FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Die Eignung von Python für Einführungskurse in das Programmieren im Vergleich zu anderen Programmiersprachen

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Magistra der Sozial- und Wirtschaftswissenschaften

im Rahmen des Studiums

Informatikmanagement

eingereicht von

Yildiz Yazicioglu
Matrikelnummer 0253072

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.Ing. Dr. Renate Motschnig
Mitwirkung: Dipl.Ing. Dr.phil. Martin Weissenböck

Wien, 19. Oktober 2015

Yildiz Yazicioglu	Renate Motschnig

Erklärung zur Verfassung der Arbeit

Yildiz Yazicioglu
Robert Blum Gasse 1/2/26, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. Oktober 2015

Yildiz Yazicioglu

Kurzfassung

In der heutigen digitalen Welt spielt Software eine bedeutende Rolle. Die Fähigkeit zur richtigen Benutzung von Software ist dabei unerlässlich. Neben der Verwendung von Programmen gewinnt zunehmend auch die Fähigkeit zur Erstellung einfacher Software – also die Programmierung – an Bedeutung.

Nun stellt sich die Frage wie man Schüler in das komplexe Gebiet der Programmierung einführen soll. Eine essentielle Rolle spielt dabei die gewählte Programmiersprache. Heute werden zur Einführung in die Programmierung meist Industriesprachen wie C oder Java verwendet. Diese Sprachen wurden dabei explizit nicht für Anfänger geschaffen. Diese Arbeit soll deshalb beleuchten wie die Verwendung der Programmiersprache Python den Einstieg in die Programmierung erleichtern kann.

Zu diesem Zweck wird zunächst im Abschnitt „Einleitung“ näher auf die Entwicklung der Programmierung eingegangen. Danach widmet sich die Arbeit grundsätzlichen Lerntheorien und spricht die Probleme des derzeitigen Unterrichts an. Als Quelle für diese Abschnitte dienen dabei sowohl klassische Literatur zu diesen Themen als auch Online-Ressourcen. Im Abschnitt „Anfängerfehler“ gibt die Arbeit an Hand von Studien aus der Literatur Aufschluss über häufige Anfängerfehler. Weiters wird in diesem Abschnitt eine für diese Arbeit durchgeführten Online-Umfrage unter Lehrern über dieses Gebiet analysiert. Im letzten Teil erfolgt schließlich eine allgemeine Analyse darüber, wie Python bestimmte Anforderungen an eine erste Programmiersprache erfüllt. Dazu werden Lösungen typischer Anfängerbeispiele in den Sprachen C, Java und Python verglichen. Zum Schluss wird auf Interviews von Lehrpersonen eingegangen. An Hand der Antworten der Lehrer wird die Eignung von Python für Einführungskurse in das Programmieren untersucht.

Die Arbeit zeigt, dass eine der großen Hürden des Einstiegs in die Programmierung das abstrakte Denken darstellt. Da diese Barriere allgemeiner Natur ist kann die Verwendung von Python hier wenig dazu beitragen diese Hürde zu verkleinern. Bei der Formulierung des abstrakten Gedankengang in eine für den Computer verständliche Form hilft die einfachere Syntax und Semantik von Python hingegen sehr wohl. Ein weiteres Problem des Anfängerunterrichts stellt die mangelnde Motivation von Schülern selbstständig Beispiele zu lösen dar. Hier hilft die direkte Rückmeldung des Interpreters und der hohe Abstraktionslevel von Python – schon einzelne Python-Statements sind ein vollständiges Programm – die Hürde zu Programmieren für die Schüler zu senken.

Abstract

In todays digital world software plays an important role. The ability to use software correctly is indispensable. In addition to the correct use of software, the creation of simple programs is a task that gained importance in recent years.

This leads us to the question of how to introduce this complex topic to beginners. The programming language plays an essential role in this process. Today most programming courses teach languages also used in the industry like C or Java. These languages were not created for beginners. This thesis therefore examines how the programming language Python helps to lower the barrier to entry for first time programming students.

For this purpose this work takes a closer look at the history of programming. Later the thesis focuses on learning theories and talks about the problems of the current teaching methods. The information behind theses subjects is taken from classic literature about these topics and online resources. In the chapter "Beginner Mistakes" the work analyzes studies to answer questions about the most common beginner mistakes. In said chapter I also analyzes an online survey for teachers about this topic. In the last part the paper focuses on how Python meets the requirements of a beginners programming language. For this purpose I take a look at typical beginner examples in the languages C, Java and Python. At the end the work discusses teacher interviews. According to the answers of the teachers I examine how Python meets the criteria of a beginners programming language.

The thesis shows that one of the biggest barriers to entry of programming is the ability of abstract thought. Since this barrier is a general one, Python is not able to significantly lower this hurdle. On the other hand the easy to understand syntax and semantic of Python helps beginners to translate their abstract thoughts into a form understandable by the computer. Another problem of programming lessons for beginners, is the lack of motivation of students to independently solve problems on their own. The direct feedback of the Python interpreter and high abstraction level of Python – even single Python statement are full programs – helps to lower the barrier for students to start programming.

Inhaltsverzeichnis

KAPITEL 1

Einleitung

1.1 · Problemstellung

Mit dem technischen Fortschritt des 19. und 20. Jahrhunderts gelang es dem Menschen, unterschiedlichste zuvor manuell ausgeführte handwerkliche Tätigkeiten auf Maschinen zu übertragen, die dieselben Tätigkeiten meist effizienter, schneller und billiger ausführen konnten. Dabei wurde jedoch das Verhalten der Maschinen nach wie vor von einer oder mehrere Personen vorgegeben und reguliert, die als Bedienungspersonal der jeweiligen Maschine zugeordnet waren. Die technische Konsequenz daraus war, dass der Bedarf nach einer Möglichkeit entstand einer Maschine ein gewisses Verhalten auf Basis der Logik vorzuschreiben bzw. einzuprogrammieren. Dies äußerte sich beispielsweise durch die Entwicklung des Lambda-Kalküls durch die Mathematiker Alonzo Church und Stephen Kleene in den 1930er-Jahren.

Einen ersten signifikanten Einfluss auf das Weltgeschehen erlangte die Programmierung von Maschinen im zweiten Weltkrieg, durch den Bau erster mechanischer Computer die zur Entschlüsselung von gegnerischem Nachrichtenverkehr genutzt wurden. Ein prominentes Beispiel hierfür ist der Computer „Colossus“ mit dem es Alan Turing gelang den deutschen Code der Enigma zu knacken und damit den zweiten Weltkrieg nach Einschätzungen von Historikern um zwei Jahre zu verkürzen.

In den folgenden Jahrzehnten wurden Programmiersprachen sowie Computer aufgrund von militärischem Bedarf weiter entwickelt, jedoch großteils von Mathematikern, Logikern und theoretischen Informatikern die sich zuvor jahrelang dem Studium der Logik gewidmet hatten. Die in den 1950er und 1960er-Jahren entwickelten Programmiersprachen (z.B. COBOL, LISP oder FORTRAN) waren vergleichsweise logikorientiert. Dadurch beherrschten hauptsächlich entsprechend ausgebildeten Forschern die Programmierung in diesen Sprachen. In den 1970er-Jahren zur Zeit der Einführung der ersten leistbaren

Heimcomputer wurde die imperativ gehaltene Programmiersprache C entwickelt.

In dieser Zeit gewannen Computer und Programmiersprachen auch im industriellen, sowie im privaten Heimgebrauch mehr und mehr an Bedeutung. Diese Tendenz wurde im Weiteren mit der Einführung des Internets und vernetzter Inhalte gefördert. Bis heute hat sich die Einflusssphäre der Maschinen mit „programmiertem Verhalten“ bis in das tägliche Leben des Menschen ausgebreitet und ist hiervon nicht mehr weg zu denken.

Dieser Einfluss hat dazu geführt, dass in den allgemeinbildenden und berufsbezogenen Bildungsinstitutionen das Fach der „Informatik“ Einzug gehalten hat, in dem jungen Menschen im Alter zwischen zehn und zwanzig Jahren der Umgang mit Computern aber auch mitunter das Programmieren von Maschinen auf hoher abstrahierter Ebene beigebracht wird.

Während bis in die 1970er-Jahre noch sehr hardwarenah und vergleichsweise unverständlich für die menschliche Intuition programmiert wurde, wurden Programmiersprachen in den folgenden Jahrzehnten mehr und mehr an die menschliche Sprache angepasst und damit einer breiteren Öffentlichkeit zugänglicher gemacht. Dabei wurden viele Mechanismen um die sich der Programmierer davor kümmern musste automatisiert und von der Maschine übernommen. Durch diese Abstrahierung entstanden die ersten Hochsprachen, die nicht mehr so hardwarenah wie beispielsweise die Assemblersprache oder C sind. Diese Hochsprachen bilden eine semantische Zwischenschicht zwischen binärem Bitcode auf dessen Basis der Computer arbeitet und der menschlichen Sprache, wobei hier in der Regel auf Grund der historischen internationalen Anerkanntheit mit Programmierkonstrukten in Englisch programmiert wird.

Diese Zwischenebene, die aus populären Programmiersprachen wie beispielsweise C/C++, Java, Python oder C# gebildet wird, ist zwar logisch aufgebaut und formal beweisbar allerdings besonders für junge Menschen in ihrer Semantik nicht immer sofort vollständig verständlich, da sie nach wie vor abstraktes und logisches Denken erfordern. Besonders wenn kompliziertere Algorithmen implementiert wurden, die einiges an Denkvermögen verlangen um ihre Funktionalität zu verstehen, kann dies oft ein Grund für Demotivation besonders bei Programmieranfängern sein.

Darüber hinaus sind für die Erstellung von Programmen Kreativität, Analytik und Erfahrung notwendig, die teilweise erst mit dem Programmieren antrainiert werden müssen. Der Erfolg des Lernvorgangs hängt dabei sehr stark vom Aufbau, der Syntax, dem Formalismus und teilweise von der Entwicklungsumgebung der zu erlernenden Programmiersprache ab. Dabei zeigen Studien [TM12; SLY13], dass Anfängerfehler sehr oft aus der Komplexität der verwendeten Elemente einer Sprache entstehen und damit unter anderem das Denken in Abstraktionen behindern. Dies führt wiederum zu verlangsamten Programmier- und Lernfortschritt, Problemen deren Ursache mangels Erfahrung nicht sofort gefunden werden kann und damit zu weniger Erfolgserlebnissen. Dies führt gera-

de bei jungen Menschen, die im Informatik-Unterricht nicht immer ein Grundinteresse an abstrahiertem, logischen Denken haben, zu Frustration und Demotivation, sodass im Endeffekt das Programmieren im Unterricht als langweilig oder als uninteressant bewertet wird.

Vergleicht man nun Programmiersprachen untereinander um eine Programmiersprache als erste zu erlernende Sprache für Schüler auszuwählen, fällt Python in die engere Wahl da es verglichen mit anderen Hochsprachen wie Java oder C++ eine einfachere Syntax hat. Deshalb soll diese Diplomarbeit Möglichkeiten evaluieren, wie die Programmiersprache Python mit ihrer Syntax schnelle Demotivation oder Frustration beim Programmieren im Unterricht vermeiden oder lindern kann. D.h. die wissenschaftliche Fragestellung hinter dieser Arbeit ist, ob und wie es möglich ist Python als erste Programmiersprache so zu verwenden, dass die anfängliche Motivation des Schülers bzw. Programmieranfängers aufrecht erhalten werden kann und dieser im weiteren Sinne für das Programmieren begeistert werden kann.

1.2 · Motivation und Zielsetzung

Der Informatikunterricht ist generell eines der schwierigeren Fächer. Die Herausforderung in diesem Fach liegt darin Schülern ein Fachgebiet näher zu bringen, das im Vergleich zu anderen Fächern sehr abstrakt gestaltet ist. Die Materie die hierbei den Schülern nähergebracht werden soll, ist ausschließlich logischer und technischer Natur und erfordert ähnlich wie die Mathematik abstraktes Denken. Dabei müssen den Schülern gerade am Anfang Sachverhalte und Paradigmen vermittelt werden, deren Nutzen besonders am Anfang nicht gleich für den Schüler ersichtlich ist und viele Folgefragen aufwerfen (z.B. was ist der Unterschied zwischen den Datentypen double und float?). Im Folgenden sollen einige Problemdimensionen im Informatikunterricht genauer erläutert werden:

Heterogenität der Schüler:

> Hier gibt es mitunter große Unterschiede, die im Informatikunterricht ausgeglichen werden müssen. Beispielsweise können große Unterschiede in der Leistungsfähigkeit bestimmter Schüler im Informatikunterricht bestehen. Darüber hinaus gibt es Unterschiede in der Herangehensweise an die Materie der Informatik bzw. des Programmierens. Weiters bringen Schüler ein unterschiedliches Grundinteresse und vor allem eine völlig unterschiedliche Affinität zu Computern und der Informatik im Allgemeinen mit in den Unterricht. Dementsprechend wird der Unterricht bei einem Schüler, der sich zuvor bereits oft mit Computern, Internet und u.U. sogar mit Programmieren beschäftigt hat viel leichter fallen als der Unterricht bei einem Schüler der Computer im privaten Bereich noch nie verwendet hat.

Fachliche und didaktische Qualifikation des Lehrers:

Nachdem das Erstellen von Programmen mitunter eine sehr komplexe Angelegenheit sein kann, in der nicht nur paradigmatische Vorgehensweisen miteinander gekreuzt werden können sondern auch methodische Herangehensweisen an eine zu lösende Aufgabe, erfordert es sehr oft besondere didaktische Fähigkeiten und vor allem Geduld um diese Materie einem Schüler im Alter zwischen vierzehn und zwanzig Jahren näher zu bringen. Hier gibt es jedoch auch mitunter Unterschiede beim Lehrpersonal, dessen fachliche-, jedoch auch didaktische Qualifikation sehr große Unterschiede aufweisen kann. Dies ist sehr oft abhängig vom Alter, der eigenen Motivation des Lehrers und der Programmiererfahrung.

Die Art des Unterrichts und strukturelle Umgebung:

Beim Erlernen einer Programmiersprache ist der übliche Frontalunterricht wie beispielsweise im Unterrichtsfach Mathematik nicht sonderlich vorteilhaft, da der Schüler in der Regel den theoretischen Anteil an einer Programmiersprache zwar bis zu einem gewissen Grad aufnehmen kann, diesen jedoch in weiterer Folge nicht in der Realität bei einem echten Programm umsetzen kann. Auch Programmieren auf einem Blatt Papier empfiehlt sich nicht da es in der Regel keinen Sinn macht sprachliche Programmkonstrukte auf Punkt und Beistrich auswendig zu lernen und danach wieder zu geben sondern eher den Sinn hinter einem Programm zu verstehen. Insofern ist auch die Art des Unterrichts eine Dimension die von entscheidender Bedeutung für dessen Erfolg ist. Des Weiteren sind die strukturellen Rahmenbedingungen entscheidend für den Informatikunterricht, da jeder Schüler in der Praxis einen eigenen Computer, sowohl in der Schule als auch privat benötigt um Programmieren zu können.

Die Wahl der Programmiersprache:

Viele Lehrer entscheiden nach eigenem Gutdünken welche Programmiersprache den Schülern beigebracht werden soll. Dabei spielen mitunter die eigenen Programmierkenntnisse in der jeweiligen Sprache eine wichtige Rolle. Oft (und das auch im tertiären Bildungssektor) wird auf eine der klassischen Hochsprachen wie Java, C/C++ oder C# zurück gegriffen, mit dem Argument dass diese die grundlegenden Mechanismen der objektorientierten Programmierung beinhalten und von diesen aus andere Programmiersprachen schnell erlernt werden können.

Nachdem es sich jedoch bei Java und C/C++ um Programmiersprachen handelt, die zwar weit verbreitet sind jedoch paradigmatisch weniger Flexibilität beinhalten als Python, soll diese Arbeit aufzeigen welche Vorteile Python als erste Programmiersprache im Informatikunterricht im Vergleich zu den anderen Programmiersprachen aufweist. Damit soll diese Arbeit helfen, Informatiklehrern eine fundierte Entscheidungsgrundlage zu bieten, weshalb Python eine bessere Lehrsprache für Programmieranfänger ist. Sie soll auch Möglichkeiten darstellen wie Anfängerfehler im Unterricht vermieden werden können und somit die Motivation der Schüler programmieren zu lernen erhalten werden

kann.

Ein weiteres Ziel dieser Arbeit ist es den Anfangsunterricht im Programmieren genauer zu untersuchen und typische Anfängerfehler aufzuzeigen. Diese sollen beschrieben werden und es sollen Wege dargestellt werden, wie diese Anfängerfehler vermieden werden können, um so die Motivation der Programmieranfänger aufrecht zu erhalten und ihre Neugier am Programmieren zu wecken.

Die wissenschaftliche Methodik liegt einerseits in der Aufarbeitung von wissenschaftlicher Fachliteratur sowie Standardwerken. Diese sind teils im Internet in speziellen Onlinebibliotheken verfügbar und teils sind sie im freien Handel erhältlich. Andererseits dienen besonders in Bezug auf Python frei abrufbare Referenzen und Dokumentationen im Internet als theoretische Grundlage für diese Arbeit.

Die Ergebnisse der Aufarbeitung sollen danach in dieser Arbeit angeführt und diskutiert werden. Um die Anfängerfehler von Programmieranfängern zu erheben, sollen Interviews und Befragung mit Lehrpersonal anhand vorbereiteter Fragen durchgeführt werden. Diese werden in weiterer Folge hier angeführt und deren Ergebnisse in zusammengefasster Form innerhalb dieser Arbeit präsentiert und interpretiert. Und zu Guter Letzt sollen aus den Ergebnissen, sowie aus dem State oft the Art zu Python Rückschlüsse gezogen werden, warum und in welcher Form sich Python als erste Programmiersprache für Programmieranfänger eignet.

1.3 · Aufbau der Arbeit

Die vorliegende Arbeit ist insgesamt in sechs Kapitel unterteilt. Im folgenden Kapitel wird die Programmiersprache Python im Generellen betrachtet. Es wird die Notwendigkeit und die historische Entwicklung der Programmiersprache erläutert und diskutiert.

In Kapitel Drei sollen in weiterer Folge die Grundlagen des Anfangsunterrichts sowie des Lernens dargestellt werden. Es soll unter anderem auf aktuell vorherrschende Lerntheorien eingegangen werden und es sollen die Ziele des Anfangsunterrichts genauer beschrieben werden. Am Ende sollen die Probleme des derzeitigen Anfangsunterrichts in Bezug auf das Programmieren erläutert werden, d.h. es soll beschrieben werden welche Herausforderungen Informatik-Lehrer beim Programmierunterricht bewältigen müssen um ihren Schülern das Programmieren näher zu bringen. Es soll aber auch beschrieben werden welchen (unter anderem auch strukturellen) Hürden Schüler im derzeitigen Informatikunterricht bzw. beim Lernen einer Programmiersprache ausgesetzt sind.

Das darauf folgende Kapitel befasst sich mit den Anfängerfehlern von Programmierschülern. Die Anfängerfehler werden aus der Befragung mit Lehrkräften erhoben, in diesem Kapitel zusammengefasst und diskutiert.

In Kapitel fünf wird Python als erste Programmiersprache ausführlich beschrieben. D.h. es werden zuerst die Anforderungen an Python als erste Programmiersprache angeführt. Dazu zählen unter anderem die Dokumentation, die Entwicklungsumgebung oder auch eine einfach Syntax oder Hilfestellungen bei der Fehlersuche. Danach wird auf die Besonderheiten von Python als Programmiersprache eingegangen. Dazu gehören beispielsweise Konzepte wie die dynamische Typisierung, bestimmte Datenstrukturen oder aber auch die Kopplung in Python, bzw. dessen Interaktionsfähigkeit mit anderen Programmiersprachen. Im nächsten Subkapitel soll Python mit anderen Programmiersprachen verglichen werden. Dies soll anhand von Anfängerbeispielen geschehen, die direkt miteinander verglichen werden.

Im letzten Kapitel sollen aus den Lehrerbefragungen, sowie den gezogenen Rückschlüssen aus dem Vergleich von Python mit anderen Programmiersprachen Schlussfolgerungen gezogen werden. D.h. hier werden die Ergebnisse dieser Arbeit zusammengefasst und interpretiert.

KAPITEL 2

Anwendungsbereiche der Programmiersprache Python

2.1 · Allgemein

Die Programmiersprache Python wurde Ende der 1980er-Jahre vom holländischen Programmierer Guido van Rossum für das Betriebssystem Amoeba entwickelt. Die Motivation hinter der Entwicklung war es die Programmiersprache „ABC“ die zu dieser Zeit zu Lehrzwecken eingesetzt wurde, durch eine weiter entwickelte Programmiersprache zu ersetzen. Die erste Vollversion (1.0) wurde im Jänner 1994 herausgegeben und beinhaltete bereits einige funktionale Features die von der Programmiersprache Lisp übernommen wurden.

Guido van Rossum spielt in der Entwicklung von Python eine sehr große richtungsweisende Rolle als „Benevolent Dictator for Life (BDFL)“, wird jedoch von einem Kern-Entwicklerteam unterstützt. In den darauffolgenden Jahren wurden zwei weitere Versionen (1.5 und 1.6) der Programmiersprache herausgegeben, die wiederum Erweiterungen für die Behandlung von komplexen Zahlen sowie Schlüsselargumente enthielten.

Im Jahr 2000 wurde die erste echte große Nachfolge-Version Python 2.0 veröffentlicht. Die wesentlichsten Neuerungen von Python 2.0 waren ein Garbage Collector der Python die Möglichkeit gibt Speicher zu verwalten und nicht mehr benötigten Programmspeicher zur Wiederverwendung frei zu geben. Weiters waren die Unterstützung des Unicode Formats sowie etliche neue Features zur Behandlung von Listen, die sehr stark von Haskell und SETL inspiriert waren, wesentliche Neuerungen. Bis zur Version 3.0 wurden sieben kleinere Releases (2.1 bis 2.7) veröffentlicht, die als wichtigste Neuerung die Vereinigung von Typen und Klassen in eine Hierarchie beinhalten. Durch die Vereinigung wurde das gesamte Objektmodell der Programmiersprache in sich konsistent und objektorientiert

gemacht.

Die Version 3.0, die im Dezember 2008 veröffentlicht wurde, ist im Gegensatz zu ihren Vorgängerversionen nicht abwärtskompatibel, da in ihr wesentliche Design-Korrekturen vorgenommen und darüber hinaus redundante Funktionen vereinheitlicht wurden. Seit 2008 wurden wiederum vier Subversionen von Python herausgegeben, wobei es sich bei der aktuellsten Version um Python 3.5 handelt.

2.2 · Anwendungsbereich

Python wird aufgrund seiner vergleichsweisen Einfachheit gern im Informatikunterricht als erste zu erlernende Programmiersprache verwendet, da mit der Sprache sehr schnell einfache und lauffähige Programme erstellt werden können. Abgesehen davon findet es jedoch auch im wissenschaftlichen Sektor sowie im industriellen Sektor immer mehr Anwendung und wird mittlerweile auch zur Implementierung größerer Applikationen herangezogen. Ein sehr einfacher Grund für dessen immer größer werdende Verbreitung ist seine kostenfreie Verfügbarkeit im Internet sowie dessen Vielseitigkeit. Besonders in der Wissenschaft wird es gern für experimentelle Zwecke verwendet, da sich mit Python Ideen mittels „Rapid Prototyping“ sehr schnell in Code umsetzen lassen und dieser Code fähig ist mit anderen (unter Umständen sehr hardwarenahen) Komponenten zu interagieren.

Aktuell wird geschätzt, dass Python von etwa einer Million Entwicklern weltweit verwendet wird. Das bedeutet, dass Python aktuell zu den Top 10 wichtigsten Programmiersprachen überhaupt zählt, neben den großen „State of the Art“-Programmiersprachen wie Java, C/C++, C#, oder PHP. Im industriellen Bereich wird es in den unterschiedlichsten Programmbranchen eingesetzt, die sich von reiner Datenverarbeitung über hardwarenahe Aufgaben bis zum Web-Engineering-Bereich erstrecken.

Eine der wichtigsten Anwenderorganisationen von Python ist zweifelsohne Google, das Python zu einer von seinen drei offiziell verwendeten Programmiersprachen zählt. Beispielsweise wurde die Google App Engine, zur Erstellung von Webapplikationen, in Python geschrieben. Weiters wurde auch die berühmte `code.google.com` Webseite mit deren Hilfe die Google Developer Community betrieben wird unter anderem mit Python implementiert. Darüber hinaus wird Python auf der Webseite von YouTube an den verschiedensten Stellen verwendet, die von der Betrachtung von Videos, über die Videoverwaltung bis zum Zugriff auf Daten von YouTube reicht.

Ein weiterer berühmter Anwender ist Dropbox, der sowohl seine Server als auch seine Dropbox-Clients hauptsächlich in Python implementiert hat und über sie die Ablage von Daten in dessen Storage regelt. Im hardwarenahen Bereich wurde Python beispielsweise verwendet um den Minicomputer Raspberry Pi zu betreiben (das „Pi“ steht für „Python Interpreter“). Im staatlichen Bereich wird Python beispielsweise von dem US–

amerikanischen Geheimdienst National Security Agency (NSA) verwendet um gesammelte Daten zu analysieren.

Pinterest verwendet als Programmiersprache sowohl serverseitig als auch clientseitig Python. Als Web-Framework kommt bei dieser Firma Django und Tornado zum Einsatz. Tornado ist ein Python-Framework, das von FriendFeed entwickelt wurde. Nach dem großen Erfolg von FriendFeed wurde die Firma von Facebook gekauft. Aufgrund der Vertrautheit des Entwicklungsteams mit Python und der Möglichkeit einfach und schnell Software in dieser Sprache zu entwickeln, wurde Python als Programmiersprache für Pinterest ausgewählt [Cho14].

Die kalifornische Universität in Irvine entwickelt dutzende Python-Anwendungen. Diese Programme bieten Dienste wie einen Zeitplan für Klassen, ein Buchungssystem für die Fakultät und ein System in dem die Schüler Daten wie Gebühren, Kurse, Prüfungen, Vorlesungen, Stundenpläne, und den Finanzstatus aktualisieren und betrachten können.

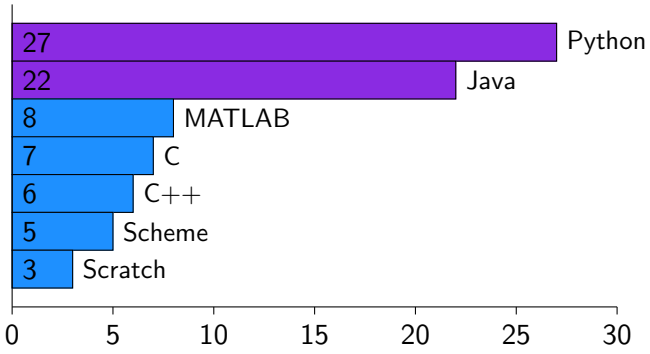
In der Fakultät für Luftfahrttechnik der Technische Universität Delft in den Niederlanden, wird im Programmierungskurs „Programmieren & Scientific Computing in Luft und Raumfahrt“ Python als Sprache gelehrt. Diese Fakultät ist die größte für Luftfahrttechnik in Europa. Andere Fakultäten wie die für „Bauingenieurwesen“ und das Doktorats-Studium werden allmählich ebenfalls auf Python umsteigen.

Das College „Smeal College of Business“ in Pennsylvania ist einer der ersten Anwender des Python-basierten Content-Management-Frameworks Zope. Die Schule prüft derzeit verschiedene Möglichkeiten, um die Leistung von Python in Business-Modeling und in der Forschung zu unterstützten und nutzbar zu machen.

„SchoolTool“ ist ein freies webbasiertes Student-Information-System für Schulen. Diese System bietet Funktionalität, wie die Anzeige von Studentenzahlen, das Einschreiben und Löschen von Schülern, das Erstellen von Abschlüssen und Abschluss-Reports, die Anzeige von Schülerleistungen und vieles weiteres. SchoolTool ist zu 100% Prozent mittels des Zope 3-Frameworks in Python geschrieben.

Python wird in sehr vielen bekannten amerikanische Universitäten eingesetzt. Philipp Guo, Forscher an der University of Rochester, hat 39 der best-bewerteten computerwissenschaftlichen Fakultäten in den USA untersucht. Von diesen 39 Top-Universitäten setzen 27 auf Python. Beispielsweise setzt das MIT, die Ivy-League-Hochschule Stanford und die Universität von Berkley auf diese Sprache (vgl [Sch14]).
Python ist auch als Sprache in der Spieleentwicklung sehr beliebt. Spiele wie „Battle Field 2“, „Snakeworlds“, „Frets On Fire“, „Metin2“ und viel mehr setzten zumindest teilweise auf die Skriptsprache [Bod13b]. Python bietet zahlreiche Libraries wie zum Beispiel Pygame und PySoy zur schnellen Spiele-Entwicklung an [Bod13b].

Abbildung 2.1: Anzahl der Anfängerkurse in verschiedenen Programmiersprachen an den 39 Top-Informatikinstituten in den USA



Python wird auch noch in weiteren Bereich wie der Wissenschaft, Finanz, Electronic-Design-Automation, Business-Software, Mikrocontroller-Programmierung, Datenbank-Programmierung, Betriebssystem-Automation und in verschiedenen Regierungseinrichtungen eingesetzt. Weitere Beispiele zu Organisationen die Python einsetzen finden sich im Wiki der Python Software Foundation [Bod13a].

So gesehen, bildet Python im generellen Sinn mehr und mehr eine flexible Alternative zu den großen Programmiersprachen wie Java, C/C++ und C#. Python findet immer mehr Verwendung in den unterschiedlichsten Einsatzbereichen für die früher andere Programmiersprachen zum Einsatz kamen. Dazu gehören in der horizontalen Dimension sowohl der Front-End-Bereich (Clients, Internet Webseiten) als auch der Back-End-Bereich (Server, Business Logik, Datenverarbeitung). In der vertikalen Dimension erstreckt sich der Einsatzbereich von Python von hardwarebezogenen Bereichen (Treiber, Betriebssystem) über Shell-Scripts bis zu weit abstrahierter Anwendungslogik (z.B. Computerspiele oder Firmenanwendungen) die ausschließlich auf dem Betriebssystem aufsetzen.

Python enthält Funktionalität um Programme zu implementieren, die eng mit dem darunter liegenden Betriebssystem zusammen arbeiten. So können mittels Python beispielsweise Dateien und Ordner durchsucht werden, andere Programme ausgeführt oder Threads behandelt werden. Des weiteren beinhaltet es alle Mechanismen die zur Interaktion mit Betriebssystemen verwendet werden. Dazu zählen unter anderem Sockets, Pipes, Kommandozeilenparameter, Stream Interfaces, Multiprozess-Behandlung etc.. Des weiteren ist es auch möglich mittels Python Anwendungen zu erstellen die eine Graphical User Interface (GUI) beinhalten. Die GUI wird in Python beispielsweise über eine API namens „Tkinter“ realisiert, die es ermöglicht plattformübergreifend (jedoch nicht komplett plattformunabhängig) GUI-Applikationen zu schreiben, die das GUI-Design des jeweiligen darunter liegenden Betriebssystems haben. Des weiteren gibt es noch etliche Erweiterungsbibliotheken um komplexere Oberflächen in Python zu erstellen.

Python kann auch internetbasierte Anwendungen verwendet werden, wie Webseiten

oder Client-Applikationen die einem User auf Webservern zur Verfügung gestellt werden. Dafür sind für Python-Erweiterungsbibliotheken verfügbar, die Python funktional zum Umgang mit den wichtigsten Internetprotokollen wie HTTP, FTP oder HTTPS ermächtigen. Abgesehen davon können auch XML und JSON basierte Dateien eingelesen, behandelt oder erstellt werden. Und darüber hinaus existieren für Python alle möglichen Frameworks zur Erstellung von Webapplikationen wie Django, web2py oder IronPython.

Eine Spezialität von Python ist dessen Interaktionsfähigkeit mit anderen Komponenten, die nicht notwendigerweise in Python geschrieben sein müssen. Dadurch kann es als eine Art Adapter zwischen Programmen fungieren die in unterschiedlichen Sprachen geschrieben sind und bis zu einem gewissen Grad deren Ausführung kontrollieren. Weiters ist Python in der Lage mittels Codegeneratoren (wie SWIG oder SIP) automatisiert Code zu erzeugen und damit Programmverhalten automatisch nachzubilden. Weiters gibt es Erweiterungs-Frameworks wie Jython oder Cython, die es Python ermöglichen innerhalb von Java bzw. C Code eingebettet zu werden, was wiederum die funktionale Mächtigkeit von Python mit der Mächtigkeit von C und Java vereint.

Um mit Daten umgehen zu können, enthält Python wiederum Interfaces über die die wichtigsten Datenbanksysteme wie Oracle, MySQL oder Postgres angesprochen werden können. Dazu enthält es eine eigene API über die Datenbanken von Python-Skripts heraus angesprochen werden können, sodass Daten aus einer Datenbank gelesen, geändert und gelöscht werden können. Ein weiteres Feature ist hier ein eigenes Python basiertes Persistenz-Framework, über das es möglich ist komplette Objekte vergleichsweise einfach in Dateien zu sichern und wieder zu laden. Und um dieselbe Funktionalität auch für Datenbanksysteme herstellen zu können gibt es beispielsweise die Frameworks SQLObject und SQLAlchemy, die eigene objektrelationale Mapper beinhalten (vergleichbar mit Java Hibernate).

Ein sehr wichtiger Anwendungsbereich für Python ist der wissenschaftliche Sektor. Aufgrund seiner Einfachheit und seiner Mächtigkeit in Bezug auf numerische Programmierung, hat sich Python sehr stark in der Wissenschaft etabliert, sei es um Experimente durchzuführen oder ein bestimmtes Verhalten in Form eines Skripts bzw. Programms zu modellieren. Python verfügt in diesem Bereich über sehr starke Erweiterungsbibliotheken, wie NumPy oder SciPy mit denen es möglich ist numerische Programme bzw. sehr rechenintensive Programme effizient zu implementieren.

Aufgrund der objektorientierten Klassenhierarchie unterstützt Python objektorientierte Grundprinzipien wie Polymorphie, die Überladung von Operatoren oder Mehrfachvererbung. Darüber hinaus kann Python-Code in Form einer Skripts implementiert werden und auch in fremdem Code eingebettet werden. Beispielsweise ist es so möglich, dass mithilfe von Python-Code Klassen und Objekte erweitert werden, die in C++, Java oder C# geschrieben wurden. Aufgrund dieser Fähigkeit wird Python sehr oft mit einer reinen Skriptsprache verwechselt, was jedoch nicht zutrifft sondern nur einen Teil seiner

Multiparadigma-Fähigkeit ausmacht.

Ein weiterer Eckpfeiler im Design von Python ist, dass einerseits objektorientiert in gekapselten Methoden bzw. Funktionen programmiert werden kann andererseits aber auch rein prozedural, wie es beispielsweise bei C der Fall ist. Durch diese Flexibilität im Programmierstil erlangt Python eine gewisse technologische Flexibilität, die besonders auf die möglichen Anwendungsbereiche eine große Auswirkung hat (vgl. Hardwarenahe Programmierung, oder Programmierung von großen Business-Objekt orientierten Applikationen).

Eine weitere Stärke von Python ist, dass es Open-Source ist und somit am Markt frei verfügbar für jeden der es verwenden möchte. Genau genommen unterliegt es einer „Python Software Foundation Lizenz“, die es erlaubt Python und dessen Code kostenfrei zu duplizieren, zu verwenden und sogar zu verkaufen.

Darüber hinaus unterstützt Python die wichtigsten Betriebssysteme die sich derzeit am Markt befinden. Dazu gehören vor allem client- und serverseitige Unix –Betriebssysteme und dessen Derivate, sowie Windows, Linux und Mac OS X. Jedoch ist Python im Vergleich zu Java nicht komplett plattformunabhängig, da sein Bytecode für jedes Betriebssystem neu kompiliert werden muss.

Python ist zwar generell in portablem „ANSI-C“-Code geschrieben jedoch wird es im Gegensatz zu Java direkt vom Betriebssystem ausgeführt. Dahingegen wird Java Bytecode nicht direkt von einem Betriebssystem gestartet sondern ist innerhalb einer Java Virtual Machine (JVM) eingebettet, die das Programm zur Laufzeit kontrolliert.

Ein weitere Stärke die Python sehr mächtig macht ist dessen Flexibilität, sowohl in Hinblick auf den Programmierstil als auch in Bezug auf seine Kombinierbarkeit, die zu einem große Teil auf dem Umstand beruht dass es in ANSI-C geschrieben ist. Python beinhaltet beispielsweise eine C-API, die es C-basierten Programmen ermöglicht in Python implementierte Strukturen aufzurufen und zur Laufzeit mit ihnen zu interagieren. Dementsprechend lässt sich Python auch mit Bibliotheken kombinieren, die für C und C++ programmiert sind, was wiederum seine funktionale Mächtigkeit erheblich steigert.

Des weiteren wurde beim Design der Programmiersprache ein Augenmerk auf Einfachheit und Lesbarkeit des Codes gelegt. Das hat wiederum zur Folge, dass Python vergleichsweise einfach zu lernen ist, da nicht sofort das objektorientierte Schema verstanden werden muss um Python-Programme zu schreiben, sondern einfache Funktionen in prozeduraler Form bereits lauffähig sind. Abgesehen davon enthält Python dem Programmieranfänger komplexere Mechanismen wie Pointer, Speicherverwaltung und Adressen (die beispielsweise in C vorkommen) vor und abstrahiert diese. Somit kann sich der Benutzer rein auf die Umsetzung des gewünschten Programmverhaltens konzentrieren.

KAPITEL 3

Anfangsunterricht

Grundlagen

Um einen tieferen Einblick in die Ziele des Informatikunterrichts zu bekommen, muss man sich mit den grundlegenden Lerntheorien auseinandersetzen. Dabei existieren drei Lerntheorien die sich hinsichtlich ihrer Ansätze wesentlich voneinander unterscheiden. Es geht um den Behaviorismus, den Kognitivismus und den Konstruktivismus. Im Allgemeinen kann man nicht sagen welche von diesen drei Lerntheorien die „beste“ ist, weil die Entscheidung sehr stark von dem Ziel und der Situation abhängt. Dennoch kann man sich ein Urteil darüber bilden, welche Lerntheorie für den Programmierunterricht die größten Erfolge zeigen würde. Im Folgenden seien diese drei fundamentalen Lerntheorien kurz vorgestellt (vgl. [Mei06; Fri14a; Fri14b; Fri14c]).

3.1 · Lerntheorien

Behaviorismus

Die Sicht auf das Lernen im Sinne des Behaviorismus beschränkt sich auf das sichtbare Verhalten von Menschen. Die Kernaussage dieser Theorie ist, dass die innerpsychischen Prozesse beim Lernen nicht berücksichtigt werden. Im Speziellen bedeutet das, dass die Vorgänge des Gehirns unsichtbar und für das Lernen von keiner Bedeutung sind. Das Gehirn können man demnach als eine Art Black Box interpretieren, von der nur der Input und der Output bekannt sind.

Das wohl bekannteste Beispiel für die Anwendung des Behaviorismus ist der klassische Frontalunterricht. Lernende bekommen verschiedene Inhalte als Reize vermittelt und die Reproduktion dieser Inhalte spielt die Rolle der Reaktion. Man spricht auch von einer sogenannten Reiz-Reaktions-Kette. Die Lernumgebung folgt dabei einem starren Ablauf, weil sie vom Lehrenden festgelegt wird. Die individuellen Faktoren spielen dabei

offensichtlich keine Rolle.

In dieser Umgebung spielt der Lehrende die Rolle einer Autoritätsperson und vermittelt dabei Wissen und Informationen. Der Lernende spielt eine passive Rolle, in dem Sinne, dass er nur auf äußere Reize durch positive oder negative Reaktionen reagiert.

Kognitivismus

Die Sicht des Kognitivismus unterstreicht, dass das Lernen kein passiver Prozess ist, sondern als aktive Verarbeitung von Informationen betrachtet wird. Dadurch unterscheidet sich der Behaviorismus sehr stark vom Kognitivismus. Die inneren Erkenntnisprozesse spielen beim Lernen eine fundamentale Rolle. Das menschliche Gehirn speichert die Erkenntnisse als sogenannte Kognitionen ab und ist in der Lage diese zu verknüpfen und auf neue Bereiche zu übertragen. Man merkt schnell, dass die individuelle Verarbeitung der Informationen nach dieser Theorie eine sehr hohe Bedeutung hat.

Das Material wird vorab ausgewählt und wirkt dadurch relativ starr. Der Lernende muss ich an die Richtlinien des Lehrenden halten, hat aber innerhalb eines vorgegebenen Rahmens die Möglichkeit auf individuelle Freiheiten.

Der Lehrende spielt dabei die Rolle eines Tutor und nimmt die zentrale Funktion bei der didaktischen Aufbereitung von Problemstellungen ein. Der Lernende kümmert sich um die eigenständige Informationsaufnahme und die Erarbeitung von Lösungsstrategien und ist somit ein aktiver Teil des Lernprozesses.

Konstruktivismus

Beim Konstruktivismus geht es um die individuelle Wahrnehmung und die individuellen Interpretationsprozesse von Informationen. Hierbei konstruiert sich das menschliche Gehirn seine eigene Wirklichkeit. Dabei kann sich die Wirklichkeit von Individuum zu Individuum durchaus unterscheiden und bietet somit Platz für eine Vielzahl von unterschiedlichen Sichtweisen und Interpretationen.

Dieser Ansatz wird dadurch geprägt, dass das Auffinden und Konstruieren von Problemen durch den Lernenden selbst geschieht. Die Prozesse sind dabei stark mit authentischen Situationen verwurzelt. Im Vordergrund steht die individuelle Konstruktion von Wissen.

Der Lehrende spielt die Rolle eines Coaches, der die eigenverantwortlichen Lernprozesse unterstützt. Durch die individuelle Konstruktion spielen die Lernenden die aktive Rolle.

Welche Methode für den Programmierunterricht?

> Programmieren lernen müssen eigentlich alle Schüler selbst. Als Lehrer hat man wirklich sehr wenige Möglichkeiten. Ich kann den Schülern eine Vorgabe geben, aber man kann niemandem Programmieren beibringen. Jeder muss es selbst lernen. Vor allem muss die Motivation von den Schülern kommen.

Die obige Aussage eines Lehrers, die im Allgemeinen auch die Meinung anderer Lehrbeauftragter in diesem Gegenstand über das Erlernen einer Programmiersprache widerspiegelt, kann bereits wichtige Indizien über die zu wählende Lerntheorie für den Programmierunterricht geben.

Ein wichtiger Faktor für die Wahl der passenden Lerntheorie ist also, dass die Schüler selbst die Initiative ergreifen und durch Übung und Auseinandersetzung mit dem Stoff sich das Programmieren unter der Verfügbarkeit eines Lehrenden beibringen. Der Behaviorismus bringt den Nachteil, dass der Lernende in die Passivität gedrängt wird und somit die Freiheit für die selbständige Auseinandersetzung mit dem Stoff fehlt. Der große Kritikpunkt bei dem Ansatz des Kognitivismus ist, dass der Lernweg bereits vorliegt und der Lernende damit nur die durch einen Lehrenden vorgegebenen Wege beschreiten kann. Das widerspricht jedoch unserer Voraussetzung, dass der Lernende Selbstinitiative und Selbstmotivation zeigen muss.

Diese Überlegungen führen geradewegs zum bereits beschriebenen Ansatz des Konstruktivismus. Der Lehrende dient als Coach für Lernenden. An dieser Stelle stellt sich jedoch die Frage wie man bei einem Lernenden die nötige Motivation wecken kann um befriedigende Ergebnisse zu erzielen.

„Kreativ sein macht Spaß und motiviert. Motivation ist die treibende Kraft hinter jeder menschlichen Tätigkeit und grundlegend für Lernprozesse“, sagt Ralf Romeike im Buch „Didaktik in der Informatik“ [SS11, Seite 356].

Unsere Erkenntnisse und das Zitat implizieren, dass eine Mischung aus Konstruktivismus und Kreativität in unserem Fall zielführend ist. Tatsächlich erfüllt der Ansatz der Lerntheorie des Konstruktionismus von Papert die von uns genannten Anforderungen an eine Lerntheorie die für den Programmierunterricht optimiert ist. Der Konstruktionismus baut auf der Idee des Konstruktivismus auf und erweitert ihn mit der Überlegung, dass das Lernen durch das eigenhändige Konstruieren eines Artefakts unterstützt werden sollte. In unserem Fall spielt das Computerprogramm die Rolle des zu konstruierenden Artefakts.

3.2 · Ziele des Anfangsunterrichts

Bevor man sich mit den Aspekten des Anfangsunterrichts beschäftigt, muss man die Ziele für den Programmierunterricht genau definieren. Was sollen die Schüler nach dem

Abschluss des Unterrichts wissen? Diese Frage lässt sich jedoch im Allgemeinen nicht beantworten. Sigrid Schubert und Andreas Schwill beschreiben in ihrem Buch „Didaktik in der Informatik“ allgemein die folgenden Ziele des Informatikunterrichts [SS11, Seite 288]:

- Der Anfangsunterricht sollte ein reduziertes, aber unverfälschtes und abgerundetes Bild der Informatik vermitteln. Dies betrifft vor allem diejenigen Schüler, die Informatik nach der Einführung wieder abwählen.
- Für die Übrigen dient der Anfangsunterricht insbesondere zur Bildung einer Grundlage, und er liefert eine Vorschau auf mögliche vertiefende Inhalte in nachfolgenden Jahrgangsstufen.

Diese Aussagen beziehen sich sehr allgemein auf den Informatikunterricht, aber man kann sie durchaus für Überlegungen bezüglich des Programmierunterrichts verwenden. Demnach gibt es zwei Arten von Schülern. Es gibt Schüler die sich vielleicht in der Zukunft viel stärker mit Thema auseinandersetzen oder sogar das Programmieren zu ihrem Beruf machen wollen. Auf der anderen Seite gibt es Schüler die den Unterricht verwenden um die Grundlagen zu lernen und um in das Programmieren hineinzuschnuppern. Die Herausforderung bei dem Unterricht ist die beiden Welten miteinander zu verbinden um einen hochqualitativen Unterricht zu bieten. Welche konkreten Inhalte sollen nun den Schülern im Anfangsunterricht beigebracht werden? Laut Schubert und Schwill sollten folgende Themen im programmiersprachlichen Zugang zum Informatikunterricht durchgenommen werden [SS11, Seite 290]:

- Einführung der elementaren Datentypen integer, real etc. und elementarer Anweisungen,
- Herstellung einfacher Programme mit Sequenz, bedingter Anweisung und Zählschleife, vorzugsweise mit mathematischen Hintergrund,
- Einbeziehung strukturierter Datentypen wie Feld, Verbund, verschiedene Schleifentypen, Zeichenkettenverarbeitung,
- Vermittlung des Parameter- und Prozedurkonzepts von Funktionen und Prozeduren, Modularisierung, erste kleinere Projekte mit anwendungsorientiertem Hintergrund,
- Einführung in Files, Referenz- und Zeigerkonzept, lineare Listen, Projekte,
- Übergang zu Rekursion im Daten- und Kontrollbereich usw.

Wie man sofort sieht, bietet dieser Ansatz ein breites Wissen über das Programmieren an und erweitert außerdem das Verständnis über die Grundkonzepte der Informatik. Das wäre ein richtiger Zugang für jemanden der in Erwägung zieht sich in der Zukunft vertiefend mit dem Programmieren zu beschäftigen. Für die andere Zielgruppe muss der

Stoff des Anfangsunterrichts viel sensibler ausgewählt werden. Im Vordergrund steht die Vermittlung von Problemlösungsstrategien statt komplizierter technischer Konzepte. Jemand der in der Zukunft kaum programmieren wird, wird von diesem Ansatz viel mehr profitieren.

Der programmiersprachliche Zugang bringt offensichtlich Nachteile mit sich, weil er die Vorkenntnisse und die Motivation der Schüler für die Teilnahme am Unterricht nicht in Erwägung zieht. Jemand der an einer Höheren Technischen Lehranstalt mit einem Schwerpunkt in Informationstechnologie ausgebildet wird, benötigt einen anderen Programmierunterricht als jemand an einer Allgemein Bildenden Höheren Schule. Diesen Nachteil kann man sehr geschickt mit der Wahl der Programmiersprache lösen. Je nach Unterrichtsziel kann man eine schlankere Sprache wählen die weniger Wissen über abstrakte Konzepte aus der Informatik erfordert und sich nur auf das Lösen von Problem konzentriert. Deswegen macht es Sinn für den Unterricht zwischen besser und schlechter geeigneten Programmiersprachen zu unterschieden.

3.3 · Probleme des derzeitigen Unterrichts

Der Programmierunterricht an Schulen ist ein relativ junges Thema. Wegen der steigenden Relevanz der Informatik in unserem Alltagsleben wird der Programmierunterricht in Schulen immer wichtiger. Das Programmieren wird nicht nur an Höheren Technischen Lehranstalten gelehrt, sondern auch an Allgemein Bildenden Höheren Schulen. Man will den Schülern ein gewisses Maß an Orientierung im Bereich der Informatik geben und nebenbei die Programmierung nutzen um einen systematischen und logischen Zugang zur Problemlösung zu geben. Ein transparenter und zielorientierter Unterricht ist wünschenswert.

Aus didaktischer Sicht gehört der Programmierunterricht auch zu den Unterrichtsfächern die die meisten Herausforderungen mit sich bringt. Die meisten Schüler tun sich am Anfang sehr schwer die programmierbaren Probleme anzugehen und selbstständig zu lösen. Es liegt in der Verantwortung des Lehrers den richtigen Zugang für die Schüler im Kontext ihrer Ausbildung zu finden und ihn strikt zu verfolgen. Der Unterricht heutzutage kümmert sich kaum um die Probleme der Schüler, die offensichtlich Schwierigkeiten mit der Erstellung ihrer ersten Programme haben. Oft werfen einfache einführende Programme viele Fragen auf die erst später beantwortet werden können und lassen die Schüler im Ungewissen über den didaktischen Verlauf des Unterrichts. Die Syntax der Programmiersprachen ist damit eine Hürde die bewältigt werden muss.

Zu einfache Programme können demotivierend auf die Schüler wirken. Um jedoch eine Programmiersprache zu erlernen muss man sich mit vielen fundamentalen Konzepten und Ideen beschäftigen. Viele Programmiersprachen wurden nicht mit dem Ziel entwickelt für Einsteiger geeignet zu sein und bringen damit viele Eigenheiten und Tücken mit sich. Bestimmte geschichtlich bedingte Konventionen mancher Sprachen sind nicht

mehr aktuell und können damit das Verständnis der Schüler in negativer Weise beeinträchtigen. Das ist bestimmt auch eine der großen Hürden im Programmierunterricht. Man muss oft mit Programmen anfangen, die viele Fragen für Schüler aufbringen und die didaktische Erfahrung der Lehrer negativ beeinflussen können.

Um diese Problematik in der Praxis zu veranschaulichen kann man ein traditionelles „Hello, World!"-Programm betrachten. Dieses Programm wird oft Einsteigern gezeigt um ihnen die grundlegende Codestruktur näher zu bringen und bietet somit das Fundament des Programmierunterrichts. Der folgende Codeabschnitt zeigt „Hello, World!" in der Programmiersprache Java:

```
class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }

}
```

Wenn ein Anfänger im Unterricht das Programm sieht, stellt er berechtigterweise sehr viele Fragen bezüglich der Syntax und der Semantik:

- Was bedeutet „`String[] args`"?
- Wieso muss die Funktion mit „`public static void`" definiert werden?
- Welche Rolle spielt das Wort „class" in unserem Programmausschnitt?
- ...

Diese Fragen kann man jedoch im Anfangsunterricht nicht beantworten, weil sie zu fortgeschritten sind und erst zu einem später Zeitpunkt im Unterricht angesprochen werden können. Dieses Programm gibt nur einen String aus und erfordert bereits viel Wissen über verschiedene Themen aus der Informatik. Für einen Schüler, der in den Unterricht einsteigt mit der Hoffnung komplexere Anwendungen zu erstellen, mag das durchaus demotivierend wirken. Wir[1] fordern deswegen eine intuitivere Programmiersprache und einen transparenteren Unterricht.

[1] Wenn gleich die Umfragen, Beispiele und Analysen selbstständig durchgeführt wurden, wurden alle Schritte mit den Betreuern diskutiert. Daher schreibe ich aus Respekt für die Zusammenarbeit oftmals wir statt ich.

KAPITEL 4

Anfängerfehler

4.1 · Typische Didaktische Problemfelder

In diesem Kapitel werden als erstes typische Anfängerfehler mittels einer Literatur-Recherche untersucht. Wir fassen dazu die Ergebnisse verschiedener Studien zu diesem Thema zusammen.

Im zweiten Teil gehen wir auf eine Lehrerbefragung ein, bei der wir verschiedene Fragen zu den häufigsten Anfänger-Fehlerquellen in Java und Python stellten. Von dreizehn gefragten Lehrer haben acht Lehrer diese Fragestellungen vollständig beantwortet. An Hand der Antworten werden wir Anfängerfehler analysieren und feststellen in welcher Programmiersprachen welche Fehler am häufigsten auftreten.

4.2 · Studien zum Thema Anfängerfehler

Programmieranfängerinnen und Programmieranfänger werden zu Beginn mit vielen verschiedenen Schwierigkeiten konfrontiert. Sie müssen nicht nur die Besonderheiten einer Programmiersprache erlernen, sondern auch verschiedenste Programmierkonzepte. Grundlegende Programmierkonzepte wie Kontrollstrukturen werden von Schülern als einfach empfunden. Studien zeigen aber, dass es schon dabei große Probleme gibt. Um ein allgemeines Problemverständnis zu entwickeln, benötigen Anfänger die Fähigkeit Programme zu erstellen und zu analysieren. Untersuchungen belegen, dass es beim Lesen und Verstehen von Code viele Schwierigkeiten gibt.

Zum Beginn wollen wir auf die Studie „Analyses of Student Programming Errors in Java Programming Courses“ [TM12] eingehen: In der Universität Samoa (NUS) werden Studenten in verschiedenen Kursen in die Programmierung eingeführt. In drei dieser Kurse, HCS181, HCS281 und HCS286 wurde gemessen welche Fehler die Studenten beim Programmieren machen. Die Nummer dieser Kurse weist dabei jeweils auf den gelehrten

Level der Lehrveranstaltung hin.

In dem Kurs HCS181 werden Konzepte wie Klassen, Methoden, Attribute, Schleifen und Kontrollstrukturen behandelt. Im fortgeschrittenen Kurs HCS281, wird auf Kontrollstrukturen, Schleifen, Fehlerbehandlung, Lesen und Schreiben auf der Konsole, und Lesen und Schreiben von Textdateien eingegangen. Im Kurs HCS286 wenden sich die Studenten schließlich Fehlerbehandlung, Lesen und Schreiben auf der Konsole, Lesen und Schreiben von Textfiles, statischen Objekten, Überladen von Funktionen, `ArrayList`, `LinkedList`, Stack, Hashtable, Trees, Polymorphismus und Interfaces zu. Als Programmiersprache kommt Java zum Einsatz. Als Entwicklungsumgebung wird JBuilder verwendet. Die Programme wurden für die Studie in JBuilder geladen. Die vom Compiler erzeugte Fehler wurden protokolliert und je nach Fehlerart kategorisiert.

Dabei wurde festgestellt, dass es einerseits Fehler gibt, die Schüler in allen Kursen machten, und andererseits Fehler, die jeweils direkt mit dem gelehrten Thema im Zusammenhang stehen. Die allgemeinen Inkorrektheiten in den Programmen der Schüler wurden in der Studie gezählt und analysiert. Sie umfassen:

- `Variable not found`
- `Identifier expected`
- `Class not found`
- `Mismatched brackets/parentheses`
- `Invalid method declaration`
- `Illegal start of type`
- `Method not found`

Tabelle 4.1: Fehler in den Kursen HCS181, HCS281 und HCS286

Fehler-Kategorie	**Häufigkeit**	**% von Gesamtzahl**
Syntax	302	94.1
Semantik	15	4.7
Logik	4	1.2
Gesamt	321	100

Tabelle 4.1 zeigt eine Einteilung der gefundenen Fehler in Kategorien und deren Auftrittshäufigkeit. Als Kategorien wurden Syntax, Semantik und Logik herangezogen. Die Tabelle zeigt, dass Syntax-Fehler die deutlich größte Kategorie bilden. Damit lässt sich feststellen, dass die Schüler nicht nur bei fortgeschrittenen Konzepten wie Objektorientierte Programmierung (OOP) Probleme haben. Viele Schwierigkeiten treten auch schon

bei der Syntax der Sprache auf.

Bei dieser Studie ist anzumerken, dass nicht auf den Aufwand eingegangen wurde, der nötig ist um die einzelnen Fehler auszubessern. Es kann also durchaus sein, dass sich nach einer zusätzlichen Abschätzung des (kognitiven) Aufwands zur Lösung der Probleme in den jeweiligen Kategorien, ein anderes Bild ergibt.

Eine weitere Studie zu dem Themengebiet Anfängerfehler ist „Determining the Barriers Faced by Novice Programmers“ [SLY13], die an der Texas A&M Universität durchgeführt wurde. Bei diesem Experiment bekamen Schüler eine Aufgabe passend zu einem Grundkonzept der Programmierung. Die Implementierung der Aufgabenstellung erfolgte in der Programmiersprache C. Die Lösungsweg der Studenten, und damit auch die aufgetretenen Fehler, wurde durch Videoaufnahme beobachtet um Probleme festzustellen.

Die Hypothese der Studie war, dass die meisten Probleme durch konzeptionelle Barrieren — also den sechs Konzepten (hinter) der Aufgabenstellungen — verursacht werden. Diese Konzepte sind:

1. Schleifen
2. Verschachtelte Abfragen
3. Switch-Anweisungen
4. Arrays
5. Funktionen
6. Dateizugriff

Dem gegenüber stehen, die in der Studie genannten, grundlegende Programmierbarrieren, wie

1. Deklaration von Header-Files
2. Syntax-Fehler und
3. der grundlegende Aufbau des Codes.

Die Studie wurde mit 20 Schüler durchgeführt, wobei diesen zufällig eine Aufgabe zu den sechs erwähnten Grundkonzepten zugewiesen wurde. Nachdem Abschluss des Experiments wurde festgestellt, dass die Hypothese der Studie falsch ist. Programmieranfänger haben also mehr Probleme mit den grundlegenden Programmierbarrieren, als mit den Grundkonzepten. Dieser Umstand ist in Abbildung 4.1 deutlich zu sehen.
Zum Schluss wollen wir auf eine Befragung von 379 Studenten an der Universität Monash vom Mai 2007 eingehen [BM07]. Diese Studie untersucht die Probleme die Studenten

Abbildung 4.1: Prozentualer Anteil der Problemgebiete der Studie „Determining the Barriers Faced by Novice Programmers“

beim Erlernen der Programmierung haben. Zu diesem Zweck wurden die Schüler zum Schwierigkeitsgrad von Programmierkonzepten und Themen des Lehrplans befragt. Die Schüler mussten dabei den wahrgenommenen Schwierigkeitsgrad auf einer sieben Punkte-Skala beurteilen.

Tabelle 4.2: Mittlere Steigerung der Schwierigkeitseinschätzung vom Verständnis hin zur Implementierung

Gebiet	**Konzeptuelle Schwierigkeit**	**Mittlere Steigerung (von 7)**	**Prozentuelle Steigerung**
Algorithmen	Hoch	0.23	3.36
Syntax	Niedrig	-0.06	-0.84
Variablen	Mittel	0.18	2.64
Bedingungen und Schleifen	Mittel	0.14	2.05
Arrays	Mittel	0.03	0.37
Methoden	Hoch	0.18	2.61
OOP Konzepte	Hoch	0.20	2.80
OOP Design	Hoch	0.09	1.24
Testen	Mittel	0.27	3.9

Besonders die Bereiche Arrays (4.17 Punkte), Methoden (3.92 Punkte), Objektorientierte Konzepte (4.92 Punkte), und Objektorientiertes Design (4.52 Punkte) wurden als Problembereiche gennant. Hier gaben vergaben die Schüler die höchsten Bewertungen für die wahrgenommene Schwierigkeit.

Bei den Themenbereichen wurden die Schüler einerseits nach der Beurteilung des Verständnisses und andererseits nach der Fähigkeit zur Implementierung befragt. Dabei schätzen die Schüler die Implementierung, außer beim Thema „Syntax“, immer als schwerer ein. Abbildung 4.2 zeigt diesen Unterschied.

4.3 · Lehrerbefragung

Im zweiten Teil dieses Kapitels wollen wir uns der Befragung von Lehrern zum Thema „Anfängerfehler“ widmen. Dabei wurden dreizehn Lehrer zu Fehlern und deren Häufigkeit befragt, wobei fünf dieser Lehrer die Fragestellung nicht beantworteten. Die Lehrer hatten die Auswahl zwischen den Antworten Nie (1), Selten (2), Manchmal (3), Häufig (4) und Sehr Häufig (5). Abbildung 4.2 zeigt typische Java-Fehler und den Mittelwert der Häufigkeit, die von den Lehrern angegeben wurde. Abbildung 4.3 zeigt zum Vergleich die Umfrage-Ergebnisse für die Sprache Python.

Während einige der in der Umfrage gezeigten Fehler sprachspezifisch sind, gibt es auch Fehler die in beiden Sprachen vorkommen können. Diese umfassen:

- Syntax-Fehler
- Probleme bei der Verwendung der Operatoren „=“ und „==“
- Problemen beim Einsatz der Schleifen „while“ und „for“
- Verwechslungen zwischen Referenzieren und Kopieren von Objekten
- Fehler bei der Ausführung des Programms
- Inkompatible Typen
- Probleme beim Einlesen von Werten von der Tastatur
- Probleme bei Listen/Arrays

An Hand dieser Fehlerquellen können wir die beiden Sprachen vergleichen.

> **Hinweis:** Bei den folgenden Prozentangaben ist zu beachten, dass sie sich auf die Punkte-Wertung der Lehrer beziehen. Eine Steigerung um 100% bedeutet also *nicht*, dass Lehrer denken, dass Schüler doppelt so viele Fehler machen.

Bei Java ergibt sich beispielsweise eine 52%-Steigerung an Syntax-Fehlern im Vergleich zu Python. Als Syntax-Fehler zählen z.B. vergessene Strichpunkte, vergessene Klammern bei Anweisungen oder Schleifen, und die Verwendung des falschen Klammertyps, also wenn Schüler z.B. runde Klammer verwenden wenn eckige Klammern erwartet werden. Die höhere Anzahl an Syntax-Fehlern bei Java war durchaus zu erwarten, da die Sprache einige Dinge voraussetzt, die in Python nicht benötigt werden, deren (optionaler) Einsatz aber auch zu keinem Syntax-Fehler führt. Zum Beispiel verlangt Python bei Abfragen keine Klammern. Auch der Strichpunkt zum Abschluss einer Anweisung ist nicht erforderlich.

Abbildung 4.2: Häufigkeit von Anfängerfehlern in der Programmiersprache Java

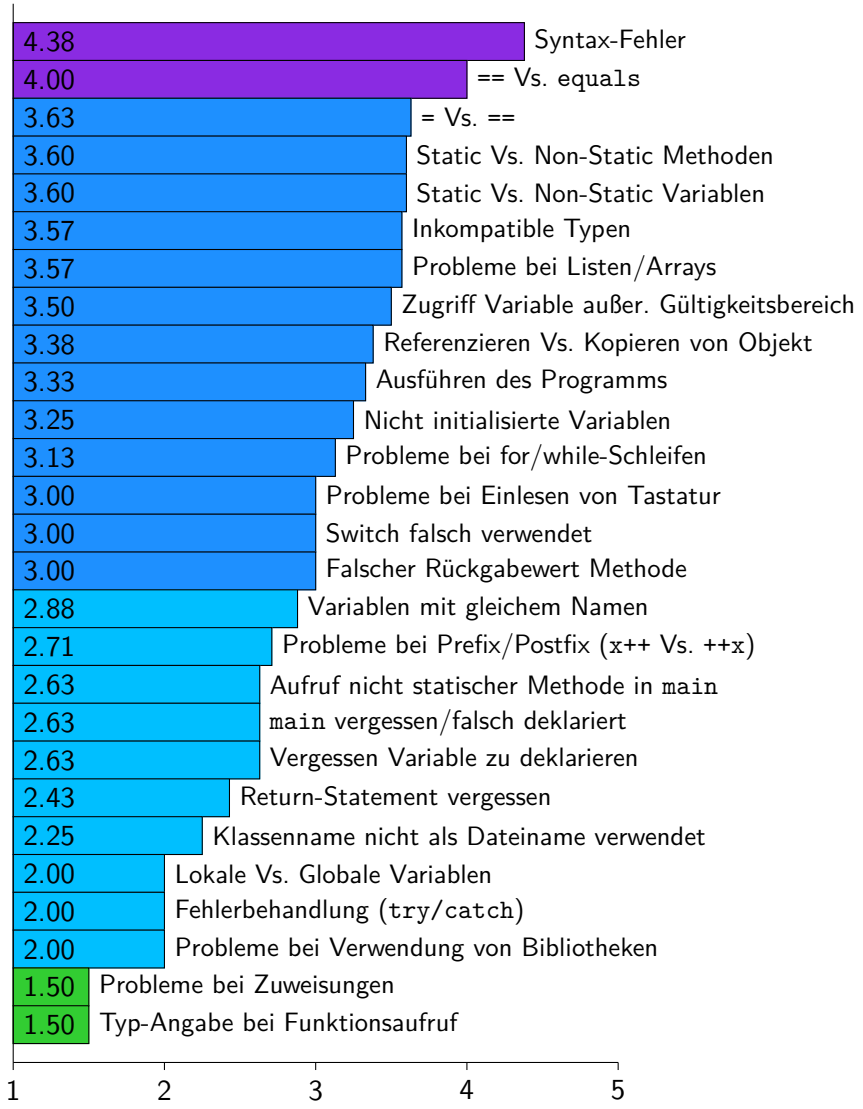
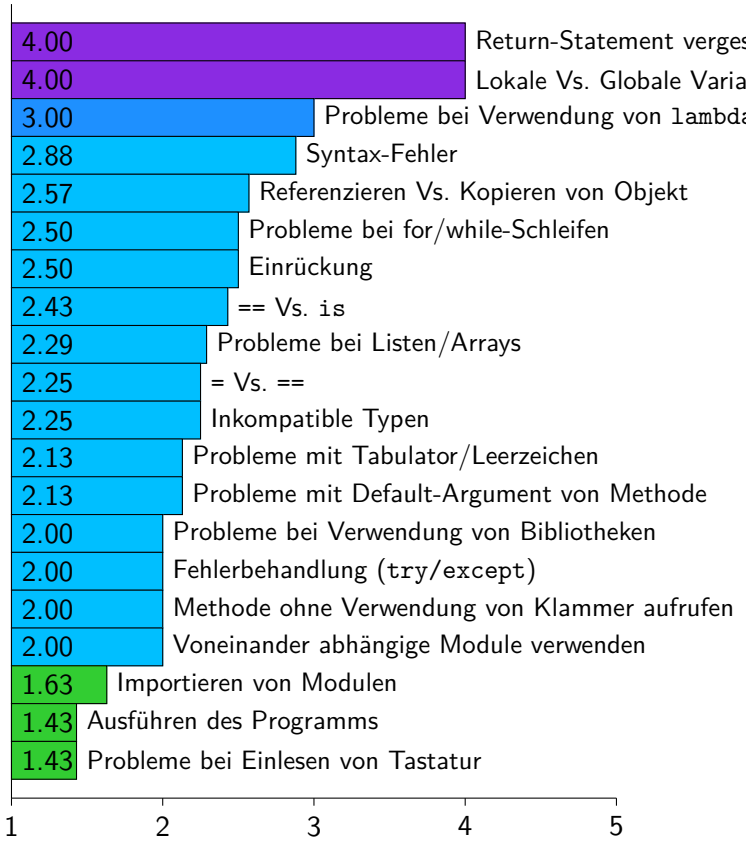


Abbildung 4.3: Häufigkeit von Anfängerfehlern in der Programmiersprache Python



Auch bei anderen Gebieten ergibt sich bei Java im Vergleich zu Python eine Steigerung der Fehlerrate (siehe Tabelle 4.3).

Tabelle 4.3: Vergleich der Fehlerraten Python/Java

Fehler	**Mittelwert Python**	**Mittelwert Java**	**Fehlerrate** [%]
Syntax-Fehler	2.88	4.38	52.08
= Vs. ==	2.25	3.63	61.33
Probleme bei for/while-Schleifen	2.5	3.13	25.2
Referenzieren Vs. Kopieren von Objekt	2.57	3.38	31.52
Ausführen des Programms	1.43	3.33	132.87
Inkompatible Typen	2.25	3.57	58.67
Probleme bei Einlesen von Tastatur	2.29	3.57	55.9
Probleme bei Listen und Arrays	2.29	3.57	55.9

Besonders herausheben kann man die erhöhte Anzahl der Fehler bei der Ausführung eines Programms (+130%). Hieran sieht man, dass eine interpretierte Sprache es erlaubt Code schnell und einfach auszuführen. Ein Umstand auf den wir später, im Abschnitt „Vergleich aktueller Programmiersprachen“, näher eingehen werden. In diesem Abschnitt zeigen wir auch warum das Einlesen von der Tastatur in Java für mehr Probleme sorgt als in Python (+55.9%).

Da es sehr schwierig ist alle Fehler zum Konzept Listen und Arrays aufzulisten, wurden diese Fehler in der Fragestellung zu einem Punkt zusammengefasst. Bei beiden Sprachen befinden sich Fehler zu diesem Gebiet in der oberen Hälfte der häufigsten Fehlertypen. Es zeigt sich also, dass Schüler Probleme mit diesem Konzept haben. Wiederum kommt es unter Python weniger häufig zu Fehlern bei diesem Thema. Diesen Umstand führen wir darauf zurück, dass Listen ein integraler Bestandteil von Python sind und es im Gegensatz zu Java keine Trennung zwischen Arrays und mächtigeren Listentypen wie z.B. `ArrayList` gibt.

4.4 · Schlussfolgerungen

Die Studien im ersten Abschnitt dieses Kapitels zeigten, dass Schüler nicht nur bei fortgeschritten Konzepte wie OOP Probleme haben. Auch Basis-Thematiken wie die Syntax der Sprache sorgen für viele Probleme.

Obwohl Anfängerfehler, wie z.B. Syntaxfehler, sich teilweise schnell ausbessern lassen, kosten sie natürlich nichtsdestotrotz Zeit. Wir befragten Lehrer zu Fehlerfällen um fest-

zustellen welche besonders häufig vorkommen. Viele der Fehler können sowohl in Python als auch in Java auftreten. Der Vergleich der dieser Fehler zeigte, dass sich diese in Java im Vergleich zu Python häufen. Wir schließen daraus, dass das Design und die Syntax von Python mögliche Fehlerquellen reduziert. Python bietet also einen leichteren und schnelleren Zugang für Programmieranfänger.

KAPITEL 5

Python als erste Programmiersprache

Zunächst wollen wir an dieser Stelle ein paar Gründe anführen warum Python als erste Programmiersprache zu bevorzugen ist:

- Python ist plattformunabhängig und kann daher auf allen großen Plattformen wie Windows, Linux, OS X und Mobil-Plattformen verwendet werden. Einige Linux-Distributionen sowie OS X werden mit vorinstalliertem Python-Interpreter ausgeliefert.
- Viele komplexe Anwendungen wie die Bildbearbeitung Gimp und die Geometrie-Software Cinderella lassen sich mit Python-Skripten erweitern.
- Python ist eine interaktive Sprache, die schnelles Feedback auf Eingaben gibt.
- Es gibt zahlreiche Bücher, Foren und Lernplattformen für die Programmiersprache. Renommierte Universitäten wie MIT verwenden als einführende Programmiersprache Python.
- Oft verwendete mathematische Konzepte wie komplexe Zahlen und Brüche können mittels Python sehr leicht modelliert werden.
- Python wird von großen Software-Firmen wie Microsoft und Google unterstützt.
- Python ist lesbar und einfach erlernbar, weil es verhältnismäßig wenige Regeln gibt, die der Programmausführung zugrunde liegen (vgl. [Old11, Seite 9]).
- Python-Programme benötigen weniger Entwicklungszeit, sind kompakter und lesbarer als Programme in anderen Sprachen wie z.B. C/C++ oder Java.

- Python ist eine stark abstrahierende Hochsprache, die viele abstrakte Datentypen wie flexible Listen und Dictionaries zur Verfügung stellt (vgl. [Mar15, Seite 3]).
- Man kann mittels Python schnell und leicht mit dem Filesystem arbeiten.
- Wissenschaftliche Berechnungen sind sehr schnell realisierbar. Dafür bietet Python mächtige Module wie NumPy.
- Python bietet sich als Ersatz für die numerische Berechnungs-Software MATLAB an. Im Gegensatz zu der genannten Software is Python frei und kostenlos verfügbar.
- Durch seine klare und einfache Syntax und Semantik vermindert Python den Overhead der in anderen Sprachen durch verpflichtende aber nicht notwendige Konstrukte entsteht (Boilerplate-Code).
- Durch die Verwendung des Interpreters als Taschenrechner können Programmieranfänger schnell durch ein ihnen vertrautes Konzept in die Programmierung eingeführt werden.
- Python bietet gute Libraries für die Spiele-Programmierung.

Einige dieser Punkten werden im späteren Verlauf dieses Kapitels noch detaillierter behandelt.

5.1 · Anforderungen an die erste Programmiersprache

In diesem Abschnitt wird versucht, wichtige Kriterien für die erste Programmiersprache zu finden und zu zeigen, ob Python diese Anforderungen erfüllt.

Die Anforderungen werden der Literatur [BFR13; Lin02; RNH06] entnommen und zusammengefasst. Die wichtigsten Kriterien für diese Arbeit werden dann aus diesen Kriterien ausgewählt.

Die Anforderungen sind:

- Dokumentation
- Entwicklungsumgebung
- Portabilität
- Paradigmen-Unterstützung
- Speicherverwaltung
- Einfache Syntax und Semantik
- Material und Werkzeuge im Unterricht

Diese Punkte werden im weiteren Verlauf dieses Abschnitts detailliert erklärt. Danach wird aufgezeigt, ob Python diese Kriterien erfüllt.

Dokumentation

Beim Erlernen der Programmierung stellt eine gute Dokumentation ein essentielles Instrument dar. Sie wird benötigt um allgemeine Informationen einzuholen und Programmierfehler besser verstehen zu können.

Dokumentation erleichtert die Wiederverwendung und das Verständnis von Code. Typische Fragen zur Dokumentation einer Programmiersprache sind:

- Wie ist es möglich Information über eingebaute Funktionalität zu bekommen?
- Ist es möglich die Herangehensweise von „fremden“ — also nicht selbst erstellten — Quelltext anzusehen und auch zu verstehen?

Die folgende aus dem Buch „Einführung in Python“ [LAG07] entnommene Tabelle, zeigt verschiedene Dokumentationsquellen für die Programmiersprache Python.

Tabelle 5.1: Verschiedene Dokumentationsquellen für die Programmiersprache Python

Form	Rolle
#-Kommentare	Dokumentation in der Datei
Die dir-Funktion	Liste von Attributen, über die Objekte verfügen
Docstrings: `__doc__`	Dokumentation in der Datei, angeheftet an Objekte
PyDoc: Die `help`-Funktion	Interaktive Hilfe für Objekte
PyDoc: HTML-Berichte	Modul-Dokumentation im Browser
Standard-Handbücher	Offizielle Beschreibung der Sprache und Bibliotheken
Webresourcen	Online-Tutorium, Beispiele usw.
Veröffentlichte Bücher	Kommerziell verfügbare Nachschlagewerke

Dokumentation, innerhalb eines Quelltextes wird als „Kommentar“ bezeichnet. Kommentare machen Programme leichter lesbar und verständlicher. Dies gilt besonders dann, wenn Personen die Wirkungsweise eines Programms verstehen wollen, das sie nicht selbst geschrieben haben.

Kommentare sind auch für die Autoren eines Programms eine wichtige Stütze. Durch die Verwendung von Kommentaren können sie festhalten warum sie bestimmte Entscheidungen getroffen haben. Zudem erlauben Kommentare Autoren auch nach einer längeren Pause ohne Probleme und ohne viel Einarbeitungszeit weiter an ihrem Programm zu arbeiten.

Kurze Kommentare werden in Python durch das Symbol „Doppelkreuz“ `#` erzeugt. Längere Kommentare — die mehrere Zeilen umfassen können — beginnen und enden mit drei Anführungszeichen. In vielen Sprachen stehen Kommentare, die die Aufgabe einer Funktion und ihre Parameter beschreiben vor der Funktionsdefinition. In Python beschreiben Kommentare unter Funktionen, Methoden oder Klasse diese. Diese Kommentare werden in Python auch als „Docstrings“ [KE07] bezeichnet und können über die Methode `__doc__()` angesehen werden.

```
def print_hello_world():
    """Hier ein Beispiel eines langen Kommentars,
    der mehrere Zeilen umfassen kann...
    """
    print("Hello World") # Ein einzeiliger Kommentar
```

Die Funktion `dir()` [Fou15a; LAG07] kann verwendet werde um die Liste der Attribute eines Objekts abzufragen. Dies geschieht in dem man das Objekt als Argument an die Funktion übergibt. Wenn diese Funktionen ohne Parameter aufgerufen wird, dann werden die Attribute im aktuellen Namensraum ausgegeben.

```
>>> dir()[:4] # Vier Attribute des globalen Namensraum
['__builtins__', '__doc__', '__name__', 'os']
>>> dir([])[:4] # Vier Attribute der Klasse Liste
['__add__', '__class__', '__contains__', '__delattr__']
```

Python stellt außer `dir()` auch die Funktion `help()` zur Verfügung um Dokumentation zu einem Objekt zu erhalten:

> Python verfügt über eine integrierte Hilfe. Die interaktive Shell liefert mit dem Kommando `help(thema)` ein Kurz-Doku. Das funktioniert für Module, Funktionen, Typen und einzelne Objekte. `help()` verfügt über einen interaktiven Modus, wenn es ohne Parameter aufgerufen wird. [Fel09]

Außer der Dokumentation innerhalb der interaktiven Python-Konsole gibt es auch noch eine umfangreiche Dokumentation der Sprache und ihrer Standard-Bibliotheken in HTML-Form. Diese steht in einer aktuellen Version unter https://www.python.org/doc/ zur Verfügung.

Entwicklungsumgebung

Integrierte und hochwertige Entwicklungsumgebungen sind heutzutage unerlässlich, besonders für professionelle Programmierer. Sie erwarten von einer Entwicklungsumgebung folgende Funktionalität [Obe03]:

- Sie sollte Komfortfunktionen wie automatische Code-Ergänzung zur Verfügung stellen,
- individuell konfigurierbar sein,
- eine Übersicht der verwendeten Klassen, Variablen und Methoden zeigen können,
- und Fehler und Compiler-Meldungen zum aktuellen Quelltext entweder direkt neben dem Code oder in einem separaten Fenster zeigen.

Nun stellt sich die Frage, ob diese umfangreiche und mächtige Funktionalität für Anfänger geeignet ist. Ein Integrated Development Environment (IDE) soll den Schüler beim Erlernen einer Programmiersprache unterstützen. Dabei ist wichtig, dass die IDE möglichst simpel und übersichtlich gehalten wird, um den Einstieg zu erleichtern. Eine zu große Funktionsvielfalt lenkt den Einsteiger vom wichtigen Kontext — Erlernen des Schreiben eines Programms — ab und ist daher didaktisch nicht ideal

In Python gibt es grundsätzlich drei verschiedene Wege um Quellcode zu schreiben und zu erweitern:

Python-Shell Innerhalb der Python-Shell kann Code zeilenweise eingegeben und ausgeführt werden. Diese Vorgehensweise ist vor allem für kleinere Projekte gut geeignet.

Editor Das Programm wird in einem Editor geschrieben. Die Ausführung des Programms erfolgt über die Kommandozeile.

Entwicklungsumgebung Der Quellcode wird in einem speziell dafür ausgelegtem Programm geschrieben und auch direkt dort ausgeführt.

IDLE

Bei der Installation von Python wird auch IDLE — eine einfache Entwicklungsumgebung — mitinstalliert. IDLE ist plattformunabhängig und kann somit auf allen Plattformen verwendet werden für die CPython zur Verfügung steht. Das ist ein klarer Vorteil dieser Entwicklungsumgebung, da sie Schüler unabhängig von der ihnen zur Verfügung stehenden Hardware, erlaubt Programme zu erstellen.

IDLE ist eine auf die wichtigsten Eigenschaften reduzierte Entwicklungsumgebung. Somit ist sie ideal für Anfänger geeignet. Sie bietet wichtige Features wie Syntax-Highlighting und Autovervollständigung, ohne jedoch zu umfangreiche Funktionalität und Features zur Verfügung zu stellen. Das ist essentiell, da komplexe Entwicklungsumgebungen den Lernprozess behindern können. Schüler beschäftigen sich dann mehr damit die Entwicklungsumgebung zu verstehen als mit dem Erlernen der Programmiersprache. Das kann schnell zur Demotivation der Schüler führen.

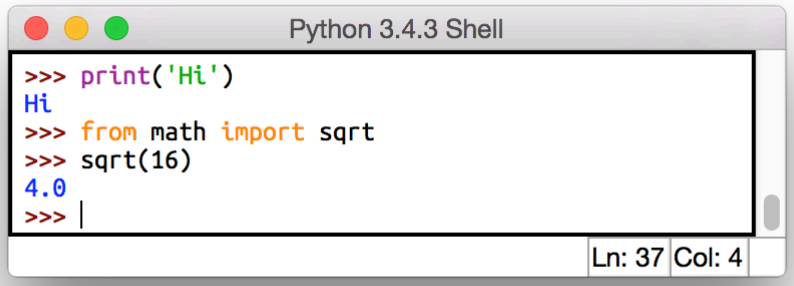


Abbildung 5.1: Ausführung einfacher Kommandos in IDLE

Texteditoren

Einer der aktuell beliebtesten Editoren ist „Sublime Text“. Er bietet Funktionen wie Autovervollständigung, Linting und Syntax-Highlighting. Außerdem kann die Funktionalität des Editors mit Plugins erweiterte werden.

Ein anderer beliebter und komfortabler Text-Editor ist Komodo. Bei Komodo handelt es sich um einen Cross-Plattform-Editor. Er verfügt über Support für mehrere Sprachen wie Python, Ruby, Javascript, PHP, Tcl, CSS, HTML und XML. Komodo bietet Syntax-Highlighting, Auto-Vervollständigung und ein „Toolbox“ genanntes Feature.

Weitere beliebte Editoren für Python finden sich unter `http://docs.python-guide.org/en/latest/dev/env/`.

Entwicklungsumgebungen

Beispiele für Entwicklungsumgebungen, die speziell auf Python zugeschnitten wurden sind Eric Python, das schon oben erwähnte IDLE und PyCharm. Weitere IDEs sind z.B. bei Wikipedia zu finden.

Neben den Entwicklungsumgebung für Quelltext steht für Python auch Software zum Entwickeln von grafischen Oberflächen, sogenannte GUI-Designer, zur Verfügung. Diese sind meist auf ein spezielles GUI-Framework ausgelegt [Wik15d]:

GUI-Toolkit	GUI-Designer
Tkinter	GUI-BUilder
WxPython	wxGlade
PyQt	Qt Designer, PySide
PyGTK	Glade

Portabilität

> Of course, the promise of 'write once, run anywhere' has been made before, most famously by Sun Microsystems for its Java language, and the Java Virtual Machine. Developers familiar with the language have an old joke based on the marketing slogan: "Write once, debug everywhere" [Kee15]

Portabilität beschreibt die Verwendbarkeit von Software in unterschiedlichen „Umgebungen". Diese Umgebung wird allgemein als Plattform bezeichnet. Sie besteht im Falle einer Programmiersprache aus der gegebenen Hardware und dem Betriebssystem, dass auf dieser Hardware ausgeführt wird.

Grundsätzlich kann man zwischen unterschiedlichen Stufen der Portabilität von Software unterscheiden. Hierzu eine relevante aber nicht vollständige Auflistung:

Portabilität des Source-Codes Hierbei ist der Source-Code eines Programmes für jede Plattform ident. Um den Code auf einer Plattform auszuführen muss dieser als erstes mittels eines Compilers in ein lauffähiges Programm übersetzt werden. Dieses Programm kann dann auf der gegebenen Plattform ausgeführt werden. Abbildung 5.2 veranschaulicht diese Schritte. Als Programmiersprachen, die dieses Konzept verwenden können C, C++, Objective C und Ada genannt werden.

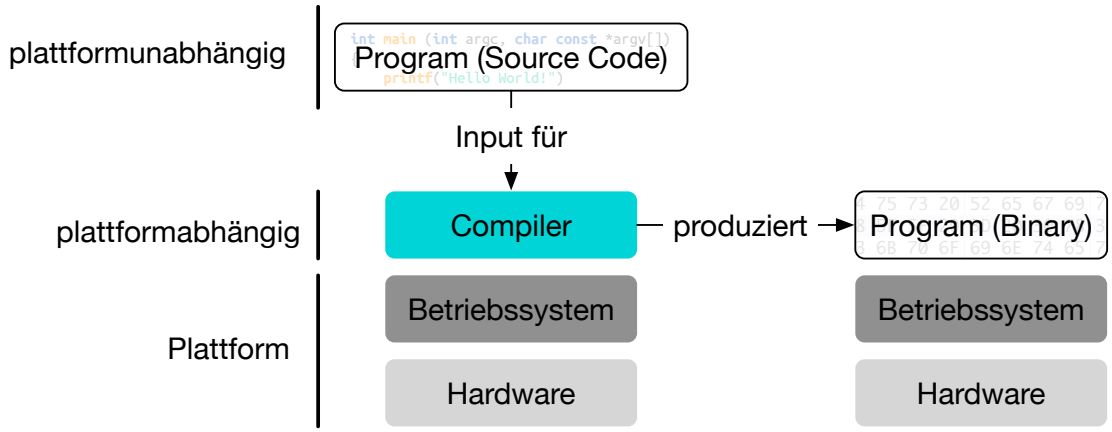


Abbildung 5.2: Plattformabhängigkeit bei kompilierten Sprachen

Eine weitere Möglichkeit besteht darin, den Übersetzungs- und Ausführungsschritt zusammenzufassen. Dabei übernimmt ein Interpreter beide Schritte. Abbildung 5.3 verdeutlicht dieses Verfahren. Die Programmiersprachen, die dieses Konzept verwenden werden auch als Skriptsprachen bezeichnet. Beispiele für populäre interpretierte Sprachen sind Python, Perl und Ruby.

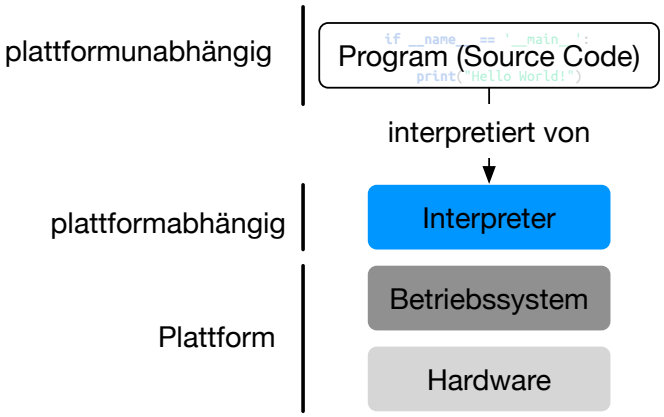


Abbildung 5.3: Plattformabhängigkeit bei interpretieren Sprachen

Portabilität von Byte-Code

Sprachen wie Java verwenden das Konzept einer sogenannten Virtual Machine (VM). Dabei wird Source-Code in eine vom Betriebssystem unabhängige Form, den sogenannten Byte-Code, übersetzt. Dieser maschinen-ähnliche Code wird dann von der Virtual Machine ausgeführt.

Dieses Konzept soll die Portabilität erhöhen. Ein Programm kann nachdem es übersetzt wurde auf jeder Plattform, auf der eine kompatible Virtual Machine existiert, ausgeführt werden. Gleichzeitig ist die Ausführung des maschinen-ähnlichen Byte-Codes schneller als die direkte Interpretation von Source Code.

Auf Grund des Geschwindigkeitsvorteils übersetzen heute viele Skriptsprachen bei der ersten Interpretation ein Skript ebenfalls in Byte-Code der dann von einer VM interpretiert wird [Pel08].

Sofern keine betriebssystem-spezifischen Funktionen verwendet werden weisen Python-Programme Source-Code-Portabilität auf. Das heißt, wenn ein Python-Interpreter für die Plattform vorhanden ist, dann kann dieser verwendet werden um Programme zu schreiben und auszuführen.

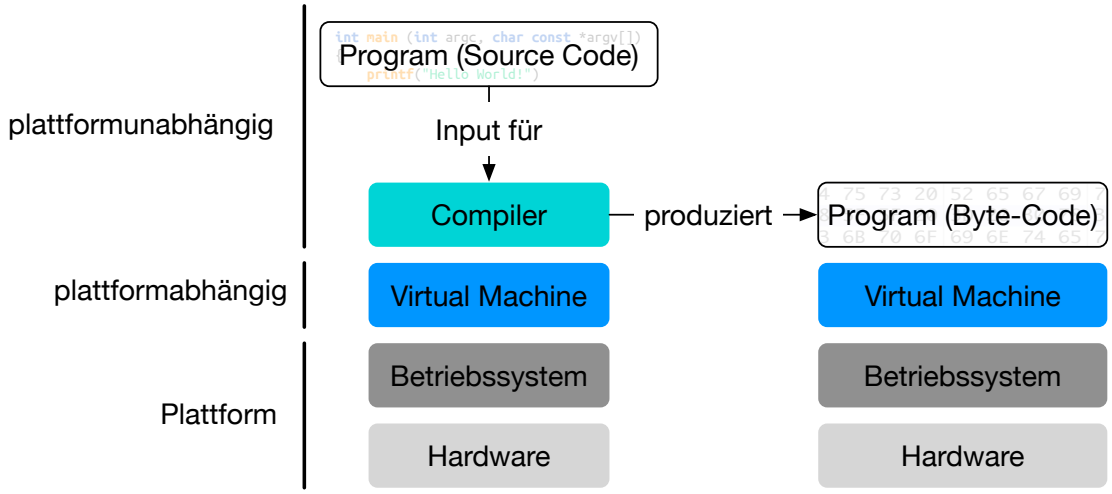


Abbildung 5.4: Plattformabhängigkeit bei der Verwendung einer Virtual Machine

Den meist verwendeten und offiziellen Python-Interpreter stellt derzeit CPython dar. Wie der Name schon verrät handelt es sich um einen Interpreter, der in der Programmiersprache C verfasst wurde. Er ist für eine weitreichende Anzahl von Plattformen verfügbar. Neben populären Betriebssystemen für PCs, wie Windows, Mac OS X und Linux, ist CPython auch auf einer Vielzahl von mobilen Plattformen, wie iOS oder Android verfügbar [Wik15a].

Um Programme in einer Sprache portabel zu gestalten ist eine umfangreiche Standard-Bibliothek notwendig. Ein sinnvolles Programm sollte verfasst werden können, ohne die Notwendigkeit betriebssystem-spezifische Funktionen direkt zu verwenden oder Grundfunktionalität selbst implementieren zu müssen. Python kann auf eine umfangreiche Standard-Bibliothek zurückgreifen. Dabei werden Funktionen für

- für Grafische User Interfaces,
- zum Dateizugriff,
- zur Verwendung von reguläre Ausdrücke,
- für Unit-Test,
- und vieles mehr

zur Verfügung gestellt. Die umfangreiche Standard-Bibliothek firmiert oft unter dem Begriff „Batteries Included“ [Bat02].

Paradigmen-Unterstützung

Einteilung der Paradigmen

> Much of this literature [about data structures] purports to be language-independent, but unfortunately it is language-independent only in the sense of Henry Ford: Programmers can use any language they want, as long as it's imperative. [Oka99]

Ein Paradigma stellt einen bestimmten „Stil“ dar, der verfolgt wird um ein Problem zu lösen. Zu qualitativ hochwertigem Code gehört die Auswahl des richtigen Paradigmas. Insbesondere ist zur Lösung mancher Aufgaben ein bestimmtes Paradigma vorgesehen [Wik15c]. Ein bekanntes Beispiel hierfür ist die mengenorientierte Sprache SQL zur Abfrage und Veränderung von Datenbeständen. Die „Programmierung“ in dieser Sprache erfolgt anhand des deklarativen Paradigmas.

Grundsätzlich kann zwischen zwei großen Gruppen von Paradigmen unterschieden werden:

- Bei der *imperativen Programmierung* wird beschrieben *wie* ein Problem gelöst wird. Der Programmierer schreibt dazu eine Folge von Befehlen die vom Computer nacheinander ausgeführt wird.
- Bei der *deklarativen Programmierung*, die oft auch als *deskriptive Programmierung* bezeichnet wird, steht im Gegensatz zum imperativen Stil das *„Was“* im Vordergrund. Das Problem wird dazu auf einer abstrakteren Ebene beschrieben als bei der imperativen Programmierung. Bei der funktionalen Programmierung — einer Untergruppe der deskriptiven Programmierung — werden z.B. Funktionen im mathematischen Sinn verwendet um ein Problem zu lösen.

Abbildung 5.5 zeigt verschiedene wichtige Durchbrüche der imperativen Programmierung. Ein essentieller Schritt stellt dabei der Weggang vom Konstrukt `goto` hinzu Schleifen (`while`, `for`) und Abfragen (`if`) dar. Der Auslöser für diese Entwicklung der sogenannten *strukturierten Sprachen* war der Artikel „Go To Statement Considered Harmful“ [Dij68] von Edsger W. Dijkstra.

Weitere wichtige Meilensteine stellen die Einführung von Prozeduren zur Unterteilung in Teilprobleme (*prozedurale Programmierung*) und die Möglichkeit zur Einteilung von Quelltext in separat übersetzbare Module (*modulare Programmierung* dar. Heute verwendete imperative Sprachen sind meist strukturiert und modular.

Bei modernen imperative Sprachen kann eine Unterscheidung in Sprachen die OOP unterstützen (C++, Java, Python) und rein prozedurale Sprachen (C) durchgeführt werden. OOP unterscheidet sich zum prozeduralen Ansatz dadurch, dass Daten und Prozeduren nicht mehr getrennt voneinander betrachtet werden. Statt dessen werden diese zu einer

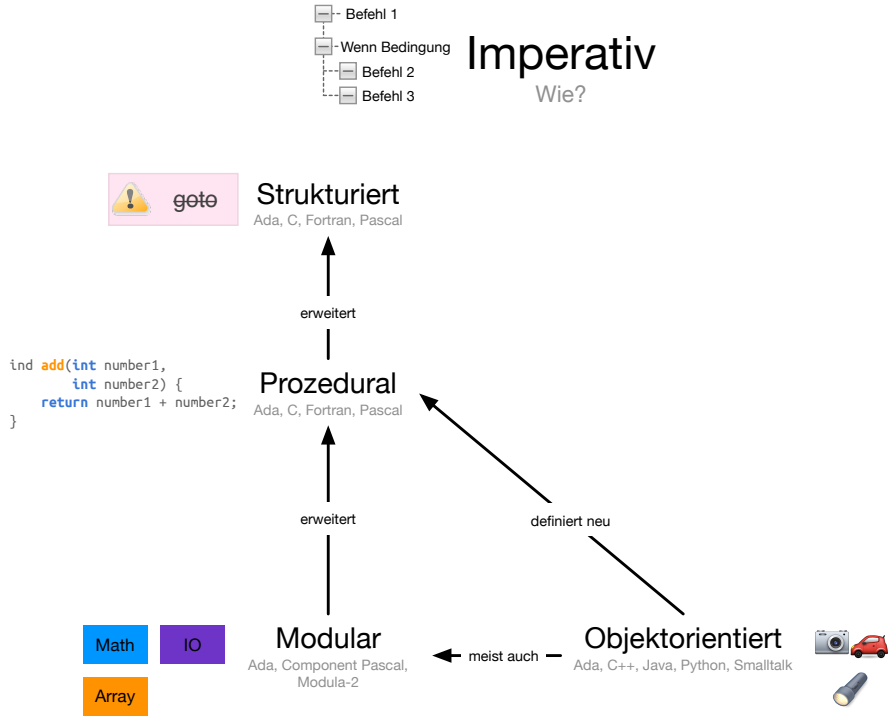


Abbildung 5.5: Wichtige Schritte der imperativen Programmierparadigmen

Einheit, den sogenannten Objekten zusammengeführt. Die Unterstützung von objektorientierten Paradigmen unterscheidet sich dabei stark nach Sprache. Während einige den Programmierer zwingen OOP zu verwenden (Java) ist bei anderen Programmiersprachen die Verwendung optional (Ada, Python).

Während imperative Sprachen heute immer noch das häufigst verwendete Werkzeug zur Programmierung darstellen, gewinnen auch zunehmend deklarative Sprachen bzw. deklarative Sprachkonstrukte an Bedeutung. Java unterstützt z.B. seit Version 1.8 funktionale Konstrukte wie anonyme Funktionen und Streams.

Abbildung 5.6 zeigt einige der beliebtesten Untergruppen der deklarative Programmierung. Anzumerken ist hier, dass es sich bei mengenorientierten Sprachen wie SQL nicht um universelle Programmiersprachen handelt. Im Gegensatz dazu sind (rein) funktionale Sprachen wie z.B. Haskell oder logische Sprachen wie Prolog „touring complete“, also gleich mächtig wie universelle imperative Programmiersprachen.

Multiparadigma-Unterstützung

Der Begriff „Multiparadigmensprache“ bedeutet, dass in einer Programmiersprache mehrere Programmierparadigmen kombiniert werden. Dadurch wird es dem Entwickler ermöglicht Aufgabenstellungen mit einem zur Problemstellung passenden Programmierstil

Abbildung 5.6: Wichtige deklarativen Programmierparadigmen

zu lösen. Alternativ kann eine Aufgabenstellung auch unter Anwendung mehrere Paradigmen gelöst werden. Python ist zum Beispiel eine Multiparadigmensprache, da in Python sowohl imperativ, als auch deklarativ programmiert werden kann.

Die Syntax und Semantik einer multiparadigmatischen Programmiersprache soll so gestaltet sein, dass die Implementierung großer Systeme mit unterschiedlichen Anforderungen besser unterstützt wird, als das bei der Umsetzung mittels reiner Kombination der unterschiedlicher Paradigmen möglich wäre.

Daher muss die Syntax einer multiparadigmatischen Programmiersprache die folgenden Anforderungen erfüllen [Gra03]:

- Anhand der Syntax sollte erkennbar sein, welches Paradigma bei der Implementierung eines Programms verwendet wurde, um die Wartbarkeit zu erleichtern.
- Es muss ein Kompromiss zwischen Knappheit und Ausführlichkeit in einem Maße geschlossen werden, sodass sich die Sprache sowohl adäquat schreiben lässt als auch lesbar ist.
- Die Syntax einer multiparadigmatischen Programmiersprache sollte es auch erlauben Programmteile in einem einzelnen Paradigma zu implementieren und das mit der syntaktischen Unterstützung, die von etablierten Einzelparadigmen geboten wird.

Nach Ludger Humbert ist es sinnvoll, mehrere Paradigmen im Informatikunterricht der Sekundarstufe II zu thematisieren.

Der Einsatz einer Multiparadigmensprache im Informatikunterricht kann den Lehr-Lernprozess fördern. Es sollte dabei möglich sein, jedes der Paradigmen „isoliert“ (und damit „sauber“) zu thematisieren, dies ist in diesem Zusammenhang wichtig zur Erarbeitung paradigmenspezifischer Fragestellungen [Hum02].

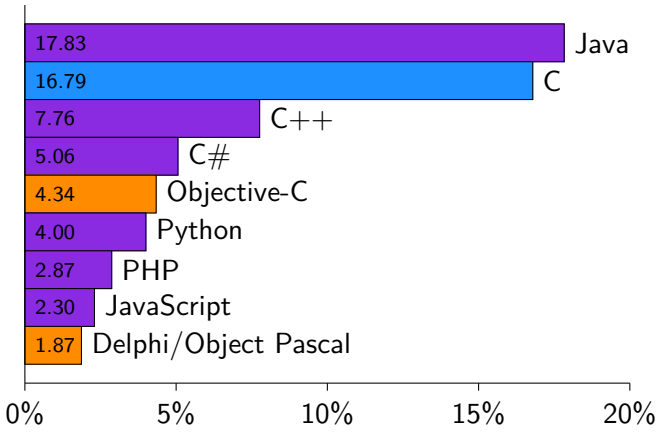
Eignung der Paradigmen für Programmiereinsteiger

Aktuell werden im europäischen Raum meist imperative Sprachen wie Java, C, oder C++ verwendet um Anfängern in die Programmierung einzuführen. Vor allem in den USA gibt es auch einige Lehranstalten die Schüler als erstes an die funktionale Programmierung heranführen. Besonders Lisp und seine Dialekte, wie Scheme und Racket, haben sich hier als Sprache für Programmieranfänger bewährt [Sci09; Fel+15].

Es stellt sich nun die Frage welches Paradigma für Anfänger am Besten geeignet ist. Dazu lässt sich feststellen, dass es hierzu keinen Konsens gibt:

> After decades of teaching programming, there is still not a consensus (and probably there will never be) on a programming paradigm and a programming language most suitable for introductory courses. [VT08]

Abbildung 5.7: Liste der populärsten Programmiersprachen im Juni 2015 laut TIOBE Index [BV15]



- Prozedurale Sprache
- (Rein) Objektorientierte Sprache
- OOP-Sprache mit Unterstützung funktionaler Programmierung

Als allgemeine Vorteile von imperativer Programmierung wird meist der höhere Verbreitungsgrad von imperativen Sprachen in der Industrie angeführt. In diesem Zusammenhang wird dabei häufig darauf hingewiesen, dass Schüler eine Sprache lernen wollen, die sie später wieder verwenden können. Während es natürlich Sinn macht eine verbreitete Sprache zu lernen, ist es auch wichtig zu sehen, dass Schüler nicht nur den Status-Quo beherrschen sollten sondern auch auf zukünftige Änderungen vorbereitet werden. Gerade in dieser Hinsicht soll als Alternative auch auf das gerade an Popularität gewinnende

Thema funktionale Paradigma eingegangen werden.

Wie Abbildung 5.7 zeigt, unterstützen die meisten populären Sprachen dieses Paradigma. Den Trend hin zur funktionalen Programmierung zeigt gerade die aktuell populärste Sprache Java, die seit Version 1.8 — veröffentlicht im März 2014 — dieses Paradigma unterstützt.

Programmierparadigmen in Python

Python unterstützt neben der prozeduralen Programmierung auch OOP und funktionale Programmierung. Der Vorteil der Sprache gerade für Anfänger liegt darin, dass kein Zwang besteht ein bestimmtes Paradigma zu verwenden. Während man bei Java gezwungen ist objektorientiert zu programmieren — die Definition zumindest einer Klasse ist nötig — kann bei Python ein Problem auch rein prozedural gelöst werden.

In den nachfolgenden Code-Listings wird die Verwendung der verschiedenen Paradigmen in Python anhand eines Beispiels gezeigt. Als Problemstellung wählen wir die Ermittlung der Summe einiger gegebener Zahlen. Listing 1 zeigt eine Lösung unter Verwendung des prozeduralen Paradigmas. Die Lösung mittels des objektorientierten Paradigmas (Listing 2) unterscheidet sich zur prozeduralen Lösung durch die „Kapselung“ der Funktion `sum` im Objekt `Numbers`. Listing 3 zeigt schließlich eine mögliche Lösung der Problemstellung mittels funktionaler Konzepte.

```
def sum(numbers):
    sum = 0
    for number in numbers:
        sum += number
    return sum

numbers = (1, 7, -2)
print("Sum of {}: {}".format(numbers, sum(numbers)))
```

Listing 1: Summenermittlung (Prozedurales Paradigma)

```
class Numbers(object):

    def __init__(self, *numbers):
        self.numbers = numbers

    def __repr__(self):
        return repr(self.numbers)

    def sum(self):
        total = 0
        for number in self.numbers:
            total += number
        return total

numbers = Numbers(1, 7, -2)
print("Sum of {}: {}".format(numbers, numbers.sum()))
```

Listing 2: Summenermittlung (Objektorientiertes Paradigma)

```
from operator import add

numbers = (1, 7, -2)
print("Sum of {}: {}".format(numbers, reduce(add, numbers)))
```

Listing 3: Summenermittlung (Funktionales Paradigma)

Die gegebenen Beispiele zeigen, dass Python die im Abschnitt „Multiparadigma-Unterstützung“ geforderten Anforderungen erfüllt:

Erkennbarkeit des verwendeten Paradigmas anhand der Syntax

Die verschiedenen Stile können in Python gut nach Verwendung von unterschiedlichen Schlüsselwörter unterschieden werden. Der objektorientierter Code verwendet beispielsweise die Begriffe `class` und `self`. Der funktionale Code kann durch die Verwendung der Funktion `reduce` erkannt werden.

Kompromiss zwischen Knappheit und Ausführlichkeit

Die Implementierung der Beispiele ist kompakt, benötigt aber keine Kenntnisse spezieller Sprachkonstrukte. Die verwendeten Konstrukte gibt es so auch anderen Sprachen die Unterstützung für die vorgestellten Paradigmen bieten (z.B. Java, Haskell).

Implementierung von Programmteilen mittels Einzelparadigmas

Die Beispiele zeigen, dass bestimmte Aufgabenstellungen mittels der Verwendung eines Paradigmas gelöst werden können. Die Paradigmen können in Python auch leicht kombiniert werden. Beispielsweise könnte man die Funktion `sum` im Objekt `Numbers` durch den in Listing 3 gegebenen funktionalen Code ersetzen:

```
...
def sum(self):
    return reduce(add, numbers)
...
```

Speicherverwaltung

Der Entwickler sollte keine Sorge bezüglich der Zuordnung und Freigabe von Speicherplatz haben [Lin02]. Stattdessen sollen sich die Schüler mit der Lösung algorithmischer Problemen befassen.

Bestimmte Programmiersprachen wie beispielsweise C++ zwingen den Entwickler aufgrund ihrer Maschinennähe, sich mit dem Speicherbereich den das jeweilige Programm benötigt zu befassen. Schüler müssen lernen, wie man nicht mehr benötigten Speicherplatz löscht, wie man einen Zeiger auf nicht belegte Speicher deklariert, und wie man einen Speicherbereich reservieren und wieder freigeben kann. Diese Arbeit ist jedoch nicht unbedingt zielführend um Programmieren zu lernen und kann unter bestimmten Umständen zum Verlust der Motivation führen, sich die Programmiersprache anzueignen.

Selbstverständlich kann das Erlernen der Speicherverwaltungssystem für den Schüler lehrreich sein. Allerdings stellt sich dabei die Frage, ob alle Schüler für diese rein technische und mentale — viel abstraktere (da auf maschinennäherer Ebene) — Arbeit reif sind, und ob man sich wirklich am Programmieranfang Gedanken über die Verwaltung des Speicherplatzes machen soll. Stattdessen ist es didaktisch sinnvoller den Fokus auf die Lösung der Aufgabenstellungen und greifbare Ergebnisse zu legen [Ram].

Python belegt automatisch den Hauptspeicher mit Objekten und gibt den Platz dafür wieder frei, sobald der betreffende Speicher nicht mehr benutzt wird. Der Schüler der Programmieren lernt muss sich nicht um die Speicherverwaltung kümmern. Diesen Prozess automatisiert Python, so dass sich der Lernende auf den eigentlichen Code konzentrieren kann [LAG07].

Für die automatisierte Speicherverwaltung verwendet Python zwei grundlegende Strategien:

1. Referenzzählung

2. Garbage Collection

Durch Referenzzählung wird der Speicherplatz gut überwacht. Jedes Objekt besitzt einen Referenzzähler, der anzeigt wie viele Objekte dieses Objekt referenzieren. Die einzelnen Objekte werden gelöscht wenn der Referenzzähler den Wert Null erreicht. Das funktioniert im Grunde sehr gut, allerdings kann es in manchen Fällen, in denen ein Objekt auf sich selbst referenziert, zu Problemen kommen. Denn durch die Referenzierung auf sich selbst wird der Referenzzähler des Objekts nicht auf null gesetzt, was bedeutet dass das Objekt nicht zum Löschen freigegeben wird, obwohl es nicht benutzt wird. Für diesen Fall ist die Garbage Collection gedacht.

Garbage Collection erfolgt über die Zuordnung und Aufhebung der Zuordnungen von Objekten. Garbage Collection kommt dann zum Einsatz, wenn keine Zuordnungen zu einem bestimmten Objekt mehr existieren. Eine Voraussetzung für die Garbage Collection ist, dass Python genügend Arbeitsspeicher zugewiesen ist, ansonsten kann die Python-Umgebung mangels Speicher abstürzen [Sch00; Dev10].

Einfache Syntax und Semantik

Strings

Python verfügt über diverse Funktionen die es erlauben Strings zu manipulieren. Einen String startet und endet entweder mit einfachen oder doppelten Anführungszeichen. Die unterschiedlichen Operationen auf Strings können direkt im Interpreter ausgeführt werden. Hierzu ein paar Beispiele:

Strings aneinander hängen

```
>>> text = "text" + "A"
>>> text
'textA'
```

Strings wiederholen

```
>>> text = "text"
>>> text * 3
'texttexttext'
```

Auf bestimmte Elemente in Strings zugreifen

```
>>> text = "Text"
>>> text[0]
'T'
```

Teile von Strings ausgeben

```
>>> text = "Ein Text"
>>> text[0:3]
'Ein'
```

Strings umdrehen

```
>>> text = "Ein Text"
>>> text[::-1]
'txeT niE'
```

Kleinste Element und größte Element in Strings ausgeben

```
>>> text = "zeichenketten"
>>> min(text)
'c'
>>> max(text)
'z'
```

Sammeltypen

Python unterscheidet mehrere Sammeltypen. Dazu gehören Listen, Dictionaries und Tupel. Listen und Tupel sind geordnete Sammlungen die verschiedene Objektarten enthalten können.

Listen

Python unterstützt heterogene Listen. Das heißt in Python müssen Elemente einer Liste nicht vom gleichem Typ sein.

Hierzu wiederum ein paar Beispiele, wobei wir hier am Anfang auch vergleichbaren Code in den Sprachen C und Java zeigen.

Erzeugen einer Liste

Python

```
liste = ["text", 3, 9.1, 'A']
```

Java

```
ArrayList liste = new ArrayList();
liste.add("text");
liste.add(3);
liste.add(9.1);
liste.add('A');
```

Ausgeben einer Liste

Python

```
print(liste)
```

Java

```
System.out.println(liste);
```

C (Homogenes Array von Zahlen)

```
printf("[");
for (int index = 0;
     index < SIZE-1;
     ++index) {
  printf("%d, ", liste[index]);
}
if (SIZE > 1) {
  printf("%d", liste[SIZE - 1]);
}
printf("]");
```

Zugriff auf Teile einer Liste

Python

```
liste2 = liste[2:4]
```

Java

```
ArrayList liste2 = new ArrayList();
for (int index = 2; index < 4; index++) {
  liste2.add(liste.get(index));
}
```

C (Homogenes Array von Zahlen)

```
int liste2[2];
for (int index = 2; index < 4; ++index) {
  liste2[index] = liste[index]
}
```

Für die restlichen Listen-Beispiele verzichten wir auf die Implementierung in anderen Programmiersprachen:

```
>>> liste = ["text", 3, 9.1, 'A']
>>> liste[0] # Zugriff auf das erste Element
'text'
>>> liste[-2] # Zugriff auf das vorletzte Element
9.1
>>> liste[2:4] # Zugriff auf die Element 3 und 4
[9.1, 'A']
>>> # Zugriff auf alle Elemente außer dem Ersten und Letzten
>>> liste[1:-1]
[3, 9.1]
>>> liste.append("...") # Einfügen eines Elements
>>> liste
['text', 3, 9.1, 'A', '...']
>>> len(liste) # Länge der Liste
5
>>> liste.pop(2) # Entfernen des dritten Elements
9.1
>>> liste
['text', 3, 'A', '...']
```

Dictionaries

Dictionaries sind in anderen Sprachen auch als „assoziative Arrays“ oder „Hashes“ bekannt. Dictionaries beinhalten Schlüssel (Keys) und deren Werte (Values). Zu den meist verwendeten Dictionary-Funktionen gehört das Setzen, Ersetzen und Löschen von Schlüssel/Wert-Paaren.

In Java wird die Dictionary-Funktionalität im Interface `Map` spezifiziert. Eine bekannte Klasse die dieses Interface implementiert ist `HashMap`. Der Unterschied zwischen Dictionaries und `HashMap` ist, dass in Python Schlüssel unveränderbare (immutable) Objekte sein müssen, damit sich der Hash-Wert des Schlüssels nicht ändert. Bei einer `HashMap` kann hingegen jedes Objekt als Schlüssel verwendet werden. Dabei muss beachtet werden, dass der Hash-Wert eines Objekt eindeutig sein muss.

Hier wiederum ein paar Beispiele zur Verwendung von Dictionaries in Python:

```
>>> person = {"david": 1972, "andreas": 2343}
>>> person
{'david': 1972, 'andreas': 2343}
>>> person["david"]
1972
>>> list(person.keys())
['david', 'andreas']
>>> "david" in person
True
>>> "David" in person
False
```

Die gleiche Funktionalität in Java:

```
HashMap<String, Integer> map = new HashMap<String, Integer>();
map.put("david", 1972);
map.put("andreas", 2343);

System.out.println(map);
System.out.println(map.keySet());
System.out.println(map.containsKey("david"));
System.out.println(map.containsKey("David"));
```

Die Implementierung der letzten zwei Beispiel-Listings in C würde entweder die Verwendung einer Nicht-Standard-Library oder eine eigene Implementierung von Hash-Tables erfordern. Da diese Implementierung für einen Programmier-Anfänger zu fordernd ist verzichten wir hier darauf ein Beispiel zu zeigen.

Tupel

Der letzte wichtige Sammlungstyp ist das sogenannte Tupel. Tupel sind vergleichbar mit Listen. Der einzige Unterschied ist, dass sie nicht veränderbar sind. Tupel starten und Enden im Gegensatz zu Listen mit runden Klammern.

An dieser Stelle stellt sich natürlich die Frage wieso man beide Sammlungstypen in einer Programmiersprache braucht. Nach Guido von Rossum, dem Erfinder von Python, sollen Tupel für heterogene und Listen für homogene Daten verwendet werden. Außerdem kann man sich sicher sein, dass wenn man mit einem Tupel arbeitet dieses einen hohen Grad an Integrität bietet, weil es nicht im Programm verändert werden kann. Tabelle 5.2 zeigt eine Zusammenfassung der vorgestellten Sammeltypen und der wichtigsten Unterschiede (vgl. [LAG07, S. 104 – 123]).

Tabelle 5.2: Unterschiede der Sammeltypen

	Listen	**Dictionaries**	**Tupel**
Ordnung	Ja	Nein	Ja
Variable Länge	Ja	Ja	Nein
Zugriff	Position	Schlüssel	Position

In Python gibt es auch spezielle Tupel- und Listenzuweisungen die im Gegensatz zur klassischen Zuweisung der einzelnen Werte eine komfortable Methode bieten um diese Sammeltypen mit Werten aufzufüllen. Man nennt diese praktische Methode auch „auspackende“ Tupel- und Listenzuweisung. Python ordnet die Objekte rechts den entsprechenden Zielen links zu. Damit kann man auch zum Beispiel Objekte innerhalb eines Tupel oder Liste auf sehr einfache Weise vertauschen. Ansonsten müsste man eine temporäre Variable definieren um den Tauschvorgang durchzuführen.

```
>>> (a, b, c) = 7, 8, 9
>>> print a, b, c
  File "<stdin>", line 1
    print a, b, c
          ^
SyntaxError: Missing parentheses in call to 'print'
>>> (a, b) = (b, a)
>>> print a, b, c
  File "<stdin>", line 1
    print a, b, c
          ^
SyntaxError: Missing parentheses in call to 'print'
```

Eine weitere Methode um Tupel und Listen zu manipulieren ist das sogenannte Slicing. Damit kann man auf ausgewählte Teile der Liste mit Hilfe eines Doppelpunktes auf einfache Art und Weise zugreifen. Außerdem kann man einfache Operatoren verwenden um Listen und Tupel zu verketten oder zu wiederholen. Alles Werkzeuge die einem Anfänger den Einstieg in die Programmierung erleichtern. Die meisten anderen Programmiersprachen bieten kaum Möglichkeiten um solche Manipulationen an jeglicher Art von Sammlungen durchzuführen.

Funktionen

Funktionen in Python werden mit dem Schlüsselword `def` eingeleitet:

```
def funktions_name(parameterliste):
    <statement-1>
    # …
    <statement-N>
```

Ein Beispiel:

```
def add(a, b):
    return a + b
```

In Gegensatz zu Python muss man in Java den Typ des Rückgabewerts und der Parametern definieren:

```
public int add(int a, int b) {
    return a+b;
}
```

Klassen und Objekte

Eine Klasse definiert man mittels des Schlüsselworts `class`.

Klassendefinition

```
class KlassenName(object):
    pass
```

„Konstruktor“

```
def __init__(self, x, y):
```

Erzeugen einer Instanz

```
x = KlassenName();
```

Hierzu ein Vergleich der grundlegenden Syntax der Elemente einer Klasse in Java und Python:

	Python	**Java**

Konstruktor	`def __init__(self,`
	`realpart,`
	`imagpart):`
	`self.realpart = realpart`
	`self.imagpart = imagpart`
		`Complex(double realpart,`
		`double imagpart) {`
		`this.realpart = realpart;`
		`this.imagpart = imagpart;`
		`}`

Erzeugen neuer Instanz	`complex = Complex(4.0,-5.5)`	`Complex complex =` `new Complex(4.0, -5.5);`

Ausgeben von Werten	`print(`
	`"real: {}, imag: {}".format(`
	`complex.realpart,`
	`complex.imagpart))`
		`System.out.println(`
		`"real: " + complex.realpart +`
		`", imag: " + complex.imagpart)`

Vererbung

Vererbung ist einer der Vorteile der objektorientierten Programmierung. Die Syntax der Vererbung in Python im Vergleich zur Vererbung in Java:

Python	Java

`class Subklasse(Superklasse):`
`  <statement-1>`
`  # …`
`  <statement-N>`
	`class Subklasse extends Superklasse {`
	`  <statement-1>`
	`  // …`
	`  <statement-N>`
	`}`

Mehrfach-Vererbung

Python unterstützt Mehrfach-Vererbung. Eine Beispiel-Klasse, die von den Superklassen `Superklasse1`, `Superklasse2` und `Superklasse3` erbt sieht etwa so aus:

```
class Subklasse(Superklasse1, Superklasse2, Superklasse3)
    <statement-1>
    # …
    <statement-N>
```

In Java existiert keine Mehrfach-Vererbung. Mittels Interfaces lässt sich dieses Feature allerdings nachahmen:

```
public Interface InterfaceSuperklasse1 {
    // Beliebige Variablendeklarationen
    // Beliebige Methodendeklarationen
}

public Interface InterfaceSuperklasse2 {
    // …
}

public class Subklasse implements InterfaceSuperklasse1,
                                  InterfaceSuperklasse2 {
    <statement-1>
    // …
    <statement-N>
}
```

Zusammenfassung

An Hand der gezeigten Beispiele sehen wir, dass die Syntax und Semantik von Python einfacher und verständlicher als ist als die der Programmiersprachen Java und C.

Material und Werkzeuge im Unterricht

An der Fakultät Mathematik der Universität Wien wird eine Programmierpraktikum angeboten. Früher wurde für dieses Programmierpraktikum die Sprache Java verwendet. Das Institut ist später auf die Sprache Python umgestiegen. Prof. Schilly bietet auf seiner Internetseite [Sch15] viele Ressourcen zum Erlernen der Programmiersprache Python an. Dort stellt er Links zu Skripten, Übungsbeispielen, nützliche Python-Tutorial-Seiten, einem Forum, einem Wiki und einer Facebook-Gruppe für dies Programmierpraktikum zur Verfügung.

Ein anderer Professor, Dieter Pahr von der Institut für Leichtbau und Struktur-Biomechanik der TU-Wien, hat die Programmiersprache seines Kurses von Java auf Python umgestellt. Seine Homepage bietet ein sehr gutes Tutorial für Python. Er hat auf seiner Webseite [Pah15] die Themen der Sprache gut aufgegliedert und diese mit einfachem Beispielen illustriert, die den Anfänger das Erlernen der Sprache erleichtern.

Python selbst bietet keine Grafikausgabe. Um 2D-Grafiken zu erstellen, kann zum Beispiel die Tk-Grafik-Bibliothek verwendet. Diese ist aber für erfahrene Programmierer gedacht und unterstützt keine 3D-Visualisierung. VPython [She] ist ein 3D-Grafik-Modul, das es erlaubt 3D-Objekte zu erstellen, zu positionieren und zu animieren. VPython erlaubt es zum Beispiel Schülern mit Vektoren zu arbeiten um physikalische Versuche zu modellieren.

Als GUI-Toolkits von Python können wir Tkinter [Bod15] und wxPython nennen. Tkinter ist eine Schnittstelle zu Tk. Tk ist eine Erweiterung von Tcl die für die GUI-Erstellung gedacht ist. Tkinter ist ein Modul, dass in Python eingebunden werden kann. Für die Erstellung einer einfachen GUI ist Tkinter ausreichend. Für komplexere Anwendungen ist es nicht geeignet. Dafür gibt es Alternativen wie WxPython, PyQt oder PyGTK.

Gato [SH15] ist eine Grafik-Animation-Toolbox, die Algorithmen auf Graphen visualisiert. Die Visualisierung erfolgt durch Effekte wie blinkende und sich abwechselnden Farben oder durch die Verknüpfungen von mathematischen Objekten. Das Programm wird auf Universität Köln entwickelt. Anwendungsgebiet dieser Software ist üblicherweise die Algorithmik und diskrete Mathematik.

Pygame [Shi15] ist ein Python-Modul zur Spieleentwicklung. Es bietet ein API für die Arbeit mit Grafik, Sound, und Eingabegeräten wie Maus und Tastatur. Ziel von Pygame ist es, die Computerspiele-Programmierung möglichst einfach zu machen indem man auf die Anwendung einer komplexen Low-Level-Programmiersprachen wie C oder C++ verzichtet.

Python bietet Bibliotheken für die einfache Kommunikation mit Micro-Controllern. Es gibt Chip-Plattformen wie „Raspberry Pi“ und „Arduino“, die man mittels Python steuern kann. Raspberry Pi-Programme lassen sich in den Sprachen Scratch, Python, C,

Ruby, Java oder Perl schreiben. Der Raspberry Pi wurde auf Universität Cambridge als Lernwerkzeuge entwickelt.

> Der Raspberry Pi ermöglicht also nicht nur (wie jeder Computer) das Programmieren eigener Software, sondern macht es sehr einfach, in die Welt des „Physical Computing“ einzusteigen, indem man eigene Projektideen mit Hilfe von computergesteuerten Sensoren, Tasten oder Leuchtdioden realisiert. [Hüb14]

Beispiele für solche Projektideen sind eine sprachgesteuerten Kaffeemaschine [LR12], ein Mediaplayer [Stü13] oder ein Wetterballon [Stü12].

5.2 · Besonderheiten von Python

Dynamische Typisierung

Man unterscheidet bei Programmiersprachen immer ob sie statische oder dynamische Typisierung verwenden. Im Allgemeinen verwenden die populärsten Programmiersprachen, wie Java und C, die statische Parametrisierung als grundlegendes Paradigma. Dabei muss grundsätzlich mit Variablen auch ein Typ deklariert werden bevor man diese mit entsprechenden Werten initialisiert. Der Typ der Variable wird dabei bereits bei der Kompilierung festgelegt. Python verwendet stattdessen eine sogenannte dynamische Typisierung. Um dieses Paradigma zu erklären muss man sich zuerst ansehen wie Python im Hintergrund mit Zuweisungen umgeht. Was passiert wenn wir die Zuweisung `a = 3` ausführen?

Zuerst wird ein Objekt mit dem Wert `3` erzeugt. Danach wird die Variable `a`, unabhängig davon ob sie bereits existiert oder nicht, erzeugt. Letztendlich wird die Variable `a` mit dem Objekt `3` verbunden und Vorgang der Zuweisung wird abgeschlossen. In Python werden solche Verbindungen zwischen Variablen und Objekten Verweise genannt, weil die Variablen auf die entsprechenden Objekte referenzieren. In diesem Sinne werden Variablen keinem Typ zugewiesen. Objekte haben hingegen sehr wohl einen Typ. Man kann sich die Variablen als Einträge in Suchtabellen vorstellen, die auf die Objekte im Hauptspeicher verweisen. Die Typ-Prüfung des Objekts wird dabei, im Gegensatz zur statischen Typisierung, erst zur Laufzeit des Programms ausgeführt. Abbildung 5.8 soll die ausgeführte Zuweisung veranschaulichen.

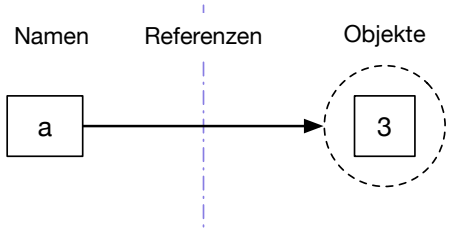


Abbildung 5.8: Eine einfache Zuweisung von Variable und Objekt [LAG07]

Man muss sich natürlich im Vorhinein im Klaren über die zur Verfügung stehenden Datentypen sein um die Ergebnisse von Berechnungen voraussagen zu können. Der Interpreter von Python kümmert sich bei Berechnungen um die Typumwandlungen selbst. Insbesondere bei Kalkulationen die Ganzzahlen und Fließkommazahlen vermischen, muss man wegen der impliziten Typumwandlungen aufpassen. Es existieren jedoch Bibliotheken die Funktionen bereitstellen, die eine explizite Typumwandlung ausführen. Beim Rechnen mit Fließkommazahlen gibt es bekanntlich oft Probleme, wenn es um die darstellbare Genauigkeit geht. Auch hierfür gibt es eine Bibliothek bei der sich die Genauigkeit und Rundung der Fließkommazahlen beliebig angeben lässt (vgl. [LAG07, Seite 73–79]).

Welche Vorteile und Nachteile bringt die dynamische Typisierung gegenüber der statischen Variante mit sich? Zuerst müssen wir betrachten welche Auswirkungen die Typüberprüfung zur Laufzeit auf den Anwender der Programmiersprache hat. Wenn die Typüberprüfung während dem Kompilieren erfolgt, werden Laufzeitfehler auf Grund falscher Typisierung natürlich sehr selten. Die einzige Ausnahme bilden Fehler bei selbst durchgeführte Typumwandlungen, die jedoch einem aufmerksamen Programmierer sehr selten passieren.

Für einem Anfänger mag die statische Typisierung jedoch unnötig kompliziert wirken und letztendlich zu mehr Fehlern führen. Wenn man noch nicht mit den Datentypen und ihren Eigenheiten vertraut ist, neigt man dazu in diesem Bereich vermehrt Fehler zu machen. Besonders explizite Typumwandlungen, die fundiertes Wissen über die im Hintergrund stattfindenden Prozesse voraussetzen, bilden für Anfänger oft eine heikle Angelegenheit. Mit einer dynamischen Programmiersprache und ihrer impliziten Typisierung bekommt ein Anfänger einen intuitiven Ansatz um seine ersten Programme zu schreiben. Das Programm wird oft in Form von Tests zur Ausführung gebracht und man kann oft direkt an die richtige Stelle springen falls ein Laufzeitfehler aufgetreten ist. Deswegen bringen Laufzeitfehler im Kontext der dynamischen Typisierung keine gravierenden Nachteile mit sich.

Bei der dynamischen Typisierung existieren auch andere Nachteile. Das wäre zum Beispiel der Overhead der bei immer gleichen Berechnungen entsteht. Da die Typüberprüfung zur Laufzeit geschieht, muss man mit etwas niedrigerer Performance rechnen. Für Anfänger im Programmieren spielt das jedoch keine große Bedeutung, da beim Lernen einer Programmiersprache die Leistung des Programms meist noch keine Rolle spielt. Natürlich ist die dynamische Typisierung auch eine Frage des persönlichen Geschmacks und wird durch steigende Erfahrung mit dem Thema Programmierung beeinflusst.

Zu den Vorteilen gehört, dass die Sprache einem hilft den richtigen Typ für das Objekt zu wählen. Das ist natürlich für Anfänger eine große Hilfe. Darüberhinaus kann man Einträge aus Dateien und Datenbanken ohne weitere Konvertierung als Zahlen behandeln. Als Anfänger kann man sich mehr auf die Logik selbst konzentrieren, da der Code mit dynamischer Typisierung flexibler und einfacher zu handhaben ist (vgl. [Fre08]). Es folgen die Vorteile und Nachteile der dynamischen Typisierung nochmal zusammengefasst.

Vorteile

- Python kümmert sich selbst um die Typisierung
- Anfänger können sich auf die Programmlogik konzentrieren
- Eingaben aus Dateien und Datenbanken sind einfacher zu handhaben

Nachteile

- Bei sich wiederholenden Berechnungen entsteht Overhead
- Frage der persönlichen Präferenzen
- Fehler werden erst zur Laufzeit entdeckt

Interaktion mit anderen Sprachen

Mit Jython wird eine Version von Python bereitgestellt, die dem Programmierer einen Zugang zu den zahlreichen Java-Bibliotheken bietet. Dazu gehören beispielsweise die bekannten Klassen AWT und Swing. Umgekehrt kann man auch Python als Skriptsprache in Java verwenden um dadurch die zahlreichen Vorteile von Python, wie die schnelle Softwareentwicklung, effektiv zu nutzen (vgl. [Seite 533 – 535][LAG07]). Listing 4 und 5 zeigen Beispiel-Code hierzu.

```
>>> from java.lang import Math
>>> Math.max(3, 5)
5L
```

Listing 4: Verwendung von Java-Code innerhalb des Jython-Interpreters

```
def show_message_as_window(msg):
    from javax.swing import JFrame, JLabel
    frame = JFrame(msg,
                   defaultCloseOperation=JFrame.EXIT_ON_CLOSE,
                   size=(100, 100),
                   visible=True)
    frame.contentPane.add(JLabel(msg))
    # ...
```

Listing 5: Verwendung von Swing innerhalb eines Python-Programms (vgl. [Jun+10b])

Python bietet mittels „Embedding“ und „Extending“ die Möglichkeit andere Programmiersprachen zu verwenden und auf diese Weise mehr Flexibilität beim Programmieren bereitzustellen. Man kann hierbei den Python-Interpreter in einer anderen Sprache einbetten oder Python mittels der Funktionen und Klassen einer anderen Sprache erweitern. Diese Funktionalität wird beispielsweise für C/C++ mit Hilfe von geeigneten Modulen realisiert (vgl. [Fou15b]). Listing 6 zeigt wie man Python innerhalb eines Java-Programms einsetzten kann.

```
public static Object createObject(Object interfaceType, String moduleName) {
    Object javaInt = null;
    // Create a PythonInterpreter object and import our Jython module
    // to obtain a reference.
    PythonInterpreter interpreter = new PythonInterpreter();
    interpreter.exec("from " + moduleName + " import " + moduleName);

    pyObject = interpreter.get(moduleName);
    // …
}
```

Listing 6: Verwendung von Python-Code innerhalb eines Java-Programms

Vorteile:

- Einfache Erweiterbarkeit und Einbettung
- Vorteile von verschiedenen Programmiersprachen nutzen

Comprehensions

Python bietet die Möglichkeit Listen, Sets und Dictionaries durch eine Notation zu erstellen, die der oft im Mathematik-Unterricht verwendeten Mengen-Notation sehr ähnelt. Damit sind keine ausgefeilten Syntaxkenntnisse notwendig um einfache Liste zu erstellen. Statt dessen kann auf die schon aus dem Mathematik-Unterricht bekannten Grundkenntnisse in der Mengenlehre zurückgreifen.

Zum berechnen der Quadrate der ganzen Zahlen von 0 bis 9 kann man zum Beispiel ein `for`-Schleife verwenden:

```
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Als Alternative bietet sich die Berechnung mittels der Funktion `map` an:

```
>>> def square(x):
...     return x**2
...
>>> squares = map(square, range(10))
>>> squares
<map object at 0x1084b9080>
```

Mit einer List-Comprehension lässt sich diese Aufgabenstellung folgendermaßen realisieren:

```
>>> squares = [x**2 for x in range(10)]
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Vorteile:

- Bereits aus dem Mathematikunterricht bekannt
- Kaum Syntaxkenntnisse notwendig

Parameter-Übergabe und Rückgabewerte

Wenn eine Funktion aufgerufen wird, werden ihre Parameter als Referenzen übergeben. Falls ein veränderliches Objekt (z.B. eine Liste oder ein Dictionary) an eine Funktion übergeben und darin verändert wird, so werden diese Veränderungen in der aufrufenden Umgebung sichtbar sein.

Beispiel:

```
>>> a = [1, 2, 3, 4, 5]
>>> def foo(x):
...     x[3] = -55 # Verändere ein Element von x
...
>>> foo(a) # Übergebe a
>>> print(a)
[1, 2, 3, -55, 5]
```

Die return-Anweisung gibt einen Wert aus der Funktion zurück. Wird kein Wert angegeben oder wird die return-Anweisung weggelassen, so wird das Objekt `None` zurückgegeben. Um mehrere Werte zurückzugeben, setzt man diese in ein Tupel:

```
def example():
  return ("Erster Wert", "Zweiter Wert")
```

Mehrere Rückgabewerte in einem Tupel können individuellen Variablen wie folgt zugewiesen werden:

```
x, y = example() # Gibt Werte in x und y zurück.
(x, y) = example() # Alternative Version - Gleiches Verhalten
```

Die Funktion apply()

Die Funktion `apply(funcname, args [, keywords])` wird verwendet, um eine Funktion indirekt auszuführen, wobei die Argumente in Form eines Tupels oder Dictionaries bereitgestellt werden. `args` ist ein Tupel mit den Positionsargumenten für die Funktion. Falls es weggelassen wird, werden keine Argumente übergeben. `kwargs` ist ein Dictionary mit Schlüsselwort-Argumenten. Die beiden folgenden Anweisungen führen zum gleichen Ergebnis:

```
foo(3,"x", name='Dave', id=12345)
apply(foo, (3,"x"), { 'name': 'Dave', 'id': 12345 })
```

Der Operator lambda

Man verwendet die lambda-Anweisung, um eine anonyme Funktion in Form eines Ausdruckes zu erzeugen: `lambda args: expression`

`args` besteht aus einer durch Kommata getrennten Liste von Argumenten und `expression` ist ein Ausdruck mit diesen Argumenten.

Beispiel:

```
>>> a = lambda x, y : x+y
>>> print(a(2, 3))
5
```

Der mit `lambda` definierte Code muss ein gültiger Ausdruck sein. Mehrfach-Anweisungen und solche wie `print`, `for` und `while` dürfen in einer `lambda`-Anweisung nicht vorkommen. Für `lambda`-Ausdrücke gelten die gleichen Sichtbarkeitsbereiche wie für Funktionen.

map() und reduce()

Die Anweisung `t = map(func, s)` wendet die Funktion `func()` auf alle Elemente in `s` an und gibt eine neue Liste `t` zurück. Für jedes Element von `t` gilt: `t[i] = func(s[i])`. Die an `map()` übergebene Funktion sollte nur ein Argument erwarten.

Beispiel:

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> def foo(x):
...   return 3*x
...
>>> list(map(foo,a))
[3, 6, 9, 12, 15, 18]
```

Alternativ dazu kann auch eine anonyme Funktion zur Berechnung verwendet werden:

```
>>> list(map(lambda x: 3*x, a))
[3, 6, 9, 12, 15, 18]
```

Die Funktion `map()` kann auch auf mehrere Listen angewendet werden, wie in `t = map(func, s1, s2, …, sn)`. In diesem Fall gilt für jedes Element von `t`: `t[i] = func(s1[i], s1[i], …, sn[i])` und die an `map()` übergebene Funktion muss die gleiche Anzahl von Argumenten erwarten wie die Anzahl der Listen. Das Ergebnis hat die gleiche Anzahl von Elementen wie die kürzeste Liste in `s1`, `s2`, …, `sn`.

Beispiel:

```
>>> a = [1, 2, 3]
>>> b = [100, 101, 102, 103]
>>> list(map(lambda x, y : (x, y), a, b))
[(1, 100), (2, 101), (3, 102)]
```

Die Funktion `reduce(func, s)` sammelt Daten aus einer Sequenz und gibt einen einzelnen Wert zurück (z.B. eine Summe, ein Maximum usw.). `reduce()` wendet dabei die Funktion `func()` auf die ersten beiden Elemente von `s` an. Das Ergebnis wird mit dem dritten Element von `s` kombiniert, um zu einem weiteren neuen Wert zu gelangen. Jenes Ergebnis wird wiederum mit dem vierten Element kombiniert usw., bis zum Ende der Sequenz. Die Funktion `func()` muss genau zwei Argumente erwarten und einen einzelnen Wert zurückgeben.

Beispiel:

```
>>> from functools import reduce
>>> a = [1, 2, 3, 4]
>>> def add(x, y):
...   return x+y
...
>>> reduce(add, a)
10
```

5.3 · Vergleich aktueller Programmiersprachen

In diesem Abschnitt wenden wir uns dem konkreten Vergleich von Programmiersprachen zu.

Zu diesem Zweck werden wir zunächst im Abschnitt „Anfängerbeispiele“ die Lösung einiger Aufgabenstellungen mittels des Einsatzes verschiedener Programmiersprachen zeigen. Im Vordergrund steht dabei, wie Python Schülern den Einstieg ins Programmieren, im Vergleich zu anderen Sprachen, erleichtern kann. Zur Gegenüberstellung wählen wir häufig als Anfängersprache zum Einsatz kommende Programmiersprachen wie Java und C. Die Beispiele wurden selbst implementiert. Dabei wurde darauf geachtet die Implementierung einfach zu gestalten.

Im zweiten Unterabschnitt „Lehrerbefragung“ analysieren wir schließlich Interviews und Befragungen von Informatik-Lehrern. Dieser Abschnitt soll dabei eine ergänzende Ansicht zum ersten Teil darstellen. Während der Abschnitt „Anfängerbeispiele“ sich rein dem Vergleich der Sprachen widmet werden wir im zweiten Teil stellenweise auch auf andere Themengebiete eingehen.

Anfängerbeispiele

Neben der Programmierung und dem dahinter stehenden wichtigen logischen Denken, muss ein Programmieranfänger auch wissen wie er Code übersetzen und ausführen kann. Zu diesem Zweck wollen wir als erstes darauf eingehen welche Schritte in den Sprachen C, Java und Python nötig sind um ein Beispielprogramm zu übersetzen und auszuführen.

Kompilierung und Ausführung

Als Aufgabenstellung wählen wir ein Programm, dass die Summe der Zahlen von 1 bis 10 ausgeben soll. Hierbei gehen wir davon aus, dass dieses Programm schon erstellt wurde und jetzt zur Kontrolle ausgeführt werden soll. Wir verzichten bewusst auf den Einsatz einer IDE, die zwar die Kompilierung vereinfachen kann, aber auch einen gewisse Einarbeitungszeit benötigt.

Listing 7 zeigt eine Implementierung der Aufgabenstellung in der Programmiersprache C.

```c
#include <stdio.h>

int main (int argc, char const *argv[])
{
    int sum = 0;

    for(int number = 1; number <= 10; number++)
    {
        sum += number;
    }

    printf("%d", sum);
    return 0;
}
```

Listing 7: Ein C-Programm zur Berechnung der Summe der Zahlen zwischen 1 und 10

Um das gegebene Beispiel `sum.c` in eine ausführbare Datei namens `sum` zu übersetzen kann folgender Befehl verwendet werden:

```
gcc sum.c -Wall -o sum
```

Bei obigem Befehl gehen wir davon aus, dass der freie Compiler `gcc` verwendet wird. Die Option `-Wall` kann theoretisch weggelassen werden, sollte aber zum Einsatz kommen, da sie dem Programmierer über möglicherweise fehlerhaften Code warnt. Um den erzeugten Code auszuführen kann dieser direkt mittels Eingabe des Namens ausgeführt werden:

```
./sum
```

Das angeführte Beispiel ist bewusst einfach gehalten. Bei Code mit mehreren Quelldateien ist der Kompilierungsprozess komplexer. Hier werden die Dateien einzeln übersetzt und zum Schluss verlinkt. Zu diesem Zweck kommt auch schon bei kleineren C-Projekten ein Build-System wie z.B. `Make` oder eine IDE zum Einsatz. Der komplexe Übersetzungsprozess lenkt Programmieranfänger von der eigentlichen Aufgabe — dem Erlernen einer Programmiersprache — ab.

Die Kompilierung von Projekten mit mehreren Dateien ist unter Java weniger aufwendig als unter C. Im einfachsten Fall übersetzt man die Klassen einfach einzeln und führt dann die Hauptklasse aus. Der einfachere Kompilierungsprozess kann darauf zurückgeführt werden, dass Java ca. 20 Jahre nach C entstand und man somit aus den Erfahrungen mit älteren Programmiersprachen lernen konnte.

```java
class Sum {

  public static void main(String[] args) {
      int sum = 0;

      for (int number = 1; number <= 10; number++) {
          sum += number;
      }

      System.out.print(sum);
  }
}
```

Listing 8: Ein Java-Programm zur Berechnung der Summe der Zahlen zwischen 1 und 10

Listing 8 zeigt Java-Code um die Summe der Zahlen von 1 bis 10 zu bestimmen. Hierbei ist zu beachten, dass unter Java der Name der Klasse und der Datei in der die Klassendefinition steht, übereinstimmen muss. Ein Umstand der Anfänger verwirren könnte, vor allem da die Kompilierung mittels `javac Dateiname.java` zu *keiner* Fehlermeldung führt wenn dieser Umstand nicht erfüllt ist. Statt dessen wird der ausführbare Bytecode einfach nicht erzeugt.

Wie schon erwähnt können wir eine `.java`-Datei mittels des Programms `javac` übersetzen. Gehen wir davon aus, dass unsere Quelldatei den Namen `Sum.java` trägt, dann können wir folgenden Befehl zum Übersetzen verwenden:

```
javac Sum.java
```

Die Ausführung des Programms erfolgt mittels der Java Virtual Machine durch folgenden Befehl:

```
java Sum
```

Wie wir in beiden vorhergehenden Beispielen gesehen haben ist bei kompilierten Sprache der Übersetzungsprozess vom Ausführungsschritt getrennt. Da es sich bei Python um eine interpretierte Sprache handelt werden diese beide Schritte zu einem zusammengefasst.

Listing 9 zeigt eine mögliche Implementierung unseres Beispiels in Python.

```python
print(sum(range(1, 11)))
```

Listing 9: Berechnung der Summe der ersten zehn Zahlen in Python

Zur Ausführung des Codes kann folgender Befehl verwenden werden:

```
python sum.py
```

Dabei gehen wir davon aus, dass der Code in der Datei `sum.py` gespeichert wurde. Dieses Beispiel demonstriert, dass Python den Schritt vom Quelltext hin zur Ausführung des Programms deutlich vereinfacht.

Als weiterer Vorteil von Python sollte hier erwähnt werden, dass Teile von Code direkt innerhalb der Python-Console ausgeführt werden können. Diese Fähigkeit erleichtert die schrittweise Erstellung von Code. Programmieranfänger können durch dieses Mittel schnell zu Erfolgserlebnissen kommen.

Listing 9 zeigt, dass Python-Code meist kürzer als vergleichbarer Code in klassischen kompilierten Sprachen ist. Ein Umstand den wir auch noch bei anderen Beispielen im weiteren Verlauf dieses Vergleichs sehen werden.

Codevergleich

Wie wir bereits im Abschnitt „Probleme des derzeitigen Unterrichts“ gesehen haben, wirft der Java-Code für das typische allererste Beispiel eines Programmieranfängers „Hello World“, schon sehr viele Fragen auf. Zur Erinnerung führen wir in Listing 10 nochmals den Code für „Hello World“ an.

```
1 class HelloWorld {
2
3     public static void main(String[] args) {
4         System.out.println("Hello World!");
5     }
6
7 }
```

Listing 10: Java-Code für „Hello World“

Gehen wir nun die einzelnen Zeilen durch und listen wir mögliche Fragen eines Schülers der dieses Programm verstehen will auf:

1: `class HelloWorld {`

- Was ist die Bedeutung des Wortes `class`?
- Warum müssen wir den Namen unseres Programms am Anfang angeben? Hat hier `HelloWorld` etwa eine andere Bedeutung?
- Warum brauchen wir geschwungene Klammern?

3: `public static void main(String[] args) {`

- Warum rücken wir den Text hier nach rechts?
- Welche Bedeutung haben die folgenden Wörter:
  - `public`
  - `static`
  - `void`
  - `main`
- Welche Bedeutung haben die runden Klammern?
- Was bedeutet `String`?
- Warum stehen hier eckige Klammern?
- Was bedeutet `args`?

4: `System.out.println("Hello World!");`

- Welche Bedeutung haben die folgenden Wörter:
  - `System`
  - `out`
  - `println`
- Warum stehen zwischen diesen Wörtern Punkte?
- Warum keine Leerzeichen wie zuvor?
- Wozu benötigt man Anführungszeichen?
- Warum steht hier zum Abschluss ein Strichpunkt?

Die meisten der hier gestellten Fragen werden erst im späteren Verlauf des Unterrichts besprochen. Statt direkt Antworten auf die Fragen zu liefern wird darauf verwiesen, dass diese Themengebiete für den Anfang zu komplex sind und erst später behandelt werden können. Wenn wir uns die Aufgabenstellung von „Hello World“ in Erinnerung rufen, dann sieht diese wie folgt aus:

> Schreiben Sie ein Programm, dass den Satz „Hello World!“ auf dem Bildschirm ausgibt.

Die erklärte Ziel der Aufgabe ist also zu lernen wie Zeichen auf dem Bildschirm ausgegeben werden können. Wenn wir in diesem Zusammenhang den Code von Listing 10 ansehen stellen wir fest, dass dafür nur das Statement:

```
System.out.println("Hello World!");
```

relevant ist. Das Problem ist, dass dieses Statement alleine keinen gültigen Java-Code darstellt. Möchten wir nur diese Codezeile erklären müssten wir trotzdem zumindest rudimentär auf folgende Punkte eingehen:

- Welche Aufgabe hat die Klasse `System`?
- Was ist eine Klasse überhaupt?
- Welche Aufgabe hat das Feld `out`
- Was ist ein Feld?
- Wie werden in Java Statements getrennt?

Alle diese Fragen lenken vom den eigentlichen Lernzielen des Beispiels ab:

- Wie gebe ich Zeichen auf dem Bildschirm aus?
  Antwort: Mit der Funktion `println`.
- Was ist eine Funktion?
  Antwort: Eine Funktion führt bestimme Befehle abhängig vom übergebenen Argument aus. Eine Funktion in (imperativen) Programmiersprachen verhält sich also ähnlich wie eine mathematische Funktion. Auch bei mathematischen Funktionen werden z.B. Klammern benutzt um den Wert von Argumente zu spezifizieren.
- Warum benötige ich Anführungszeichen?
  Antwort: Weil wir dadurch angeben, dass es sich um eine Zeichenkette handelt. Andernfalls würde der Computer meinen es handle sich bei `Hello World!` um einen Befehle ähnlich wie `println`.

Wir sehen also schon, dass die eigentlichen „Lernziele“ des Beispiels einiges Neues für den noch unerfahrenen Programmierer mitbringen. Würden wir auch noch die anderen Themengebiete die vom Java-Code von „Hello World“ aufgebracht werden behandeln, dann wäre der Lernende schnell überfordert. Folglich sinkt die Motivation und das Interesse an dem Wissens-Gebiet Programmierung wird gemindert.

Wir haben gezeigt, dass ein zumindest überblicksmäßiges Verständnis des Java-Codes für das typische erste Programmierbeispiel schon sehr viel Wissen über die verschiedensten Gebiete fordert. Nun wollen wir auf den in Listing 11 gezeigten Python-Code für die Aufgabenstellung eingehen.

```
print("Hello World")
```

Listing 11: Python-Code für „Hello World“

Wie wir sehen können fordert ein gründliches Verständnis der Implementierung im Gegensatz zur Java-Version keinerlei Wissen über:

- Klassen
- Objekte
- Variablen
- Klassenmethoden
- Sichtbarkeit von Methoden
- Rückgabewerten von Funktionen
- Arrays

Einige der oben angesprochenen Punkte lassen sich darauf zurückführen, dass Java den Programmierer zwingt objektorientiert zu programmieren. Dadurch kommen schon bei den einfachsten Java-Programmen OOP-Begriffe zur Sprache. Bei Python entfällt diese Hürde. Ein Programm lässt sich auch nur unter Zuhilfenahme des prozeduralen Paradigmas lösen.

Beim Vergleich dieses Beispiels zeigt sich, dass Python eine *gute Trennung der Wissensgebiete* erlaubt: Einzelne Themen können aufeinander aufbauend besprochen werden ohne in Verlegenheit zu geraten viele andere Fragestellungen gleich mitbehandeln zu müssen.

Zum Schluss wollen wir auf die Implementierung von „Hello World“ in der Programmiersprache C eingehen. Listing 12 zeigt den Code dafür.

```c
#include <stdio.h>

int main() {
  printf("Hello World!");
  return 0;
}
```

Listing 12: C-Code für „Hello World“

Wie wir sehen ist der Code einerseits ein wenig „einfacher“ als die Java-Version, da kein OOP zum Einsatz kommt, andererseits ergeben sich hier Fragestellungen, die wir im Java Code nicht hatten. Bei den folgenden Fragen werden die Themengebiete die auch schon beim Java-Code in sehr ähnlicher Form vorkamen grau hinterlegt, um den Fokus auf die neuen Fragestellungen zu lenken:

1: `#include <stdio.h>`

- Welche Bedeutung hat `include`? Warum steht vor diesem Wort eine Raute?

- Welche Bedeutung haben die Kleiner/Größer-Zeichen?
- Für was steht `stdio.h`? Handelt es sich dabei um eine Datei?

3: `int main() {`

- Welche Bedeutung hat `int`?
- Was ist `main`?
- Warum stehen hier leere runde Klammern? Kann man die auch einfach weglassen?
- Warum verwenden wir eine geschwungene Klammer?

4: `printf("Hello World!");`

> Die Fragestellungen hier sind praktisch die gleichen wie dem Python-Code von „Hello World“. Einzig auf die Verwendung des Strichpunkts muss noch extra eingegangen werden.

5: `return 0;`

- Welche Bedeutung hat `return`? Warum wird hier etwas zurückgegeben?
- Wird hier `0` zurückgegeben? Warum gerade diese Zahl?

3-6: `{ }`

- Warum schließen wir diesen Teil in geschwungenen Klammern ein?

Gesamt kann man feststellen, dass man auch bei der Implementierung in C für ein volles Verständnis des Codes einige Fragestellungen geklärt werden müssen, die nicht direkt im Zusammenhang mit den Lernzielen des Beispiels stehen.

Wir wollen nun auf ein paar weitere Beispielaufgaben eingehen und typische Implementierungen dieser Beispiele in den verschiedenen Sprachen vergleichen. Dabei gehen wir die Aufgaben in einer ähnlichen Reihenfolge durch wie sie auch im üblichen Informatikunterricht vorkommen. Hierzu als erstes eine überblicksmäßige Angabe der Aufgabenstellungen:

Summe	Schreiben Sie ein Programm, dass zwei Zahlen interaktiv einliest. Die Summe der beiden Zahlen soll ausgegeben werden.
Mittelwertabweichung	Schreiben Sie eine Methode, die alle Werte eines Arrays liefert, die mindestens 10% vom Mittelwert abweichen.
Sortieren von Dateiinhalt	Schreiben Sie Code, der den Inhalt einer Datei zeilenweise einliest. Sortieren Sie dann die eingelesenen Zeilen und speichern Sie das Ergebnis in einer neuen Datei.

Komplexe Summe Schreiben sie eine Klasse `ComplexNumber`. Objekte dieser Klasse stellen komplexe Zahlen dar. Implementieren Sie eine Methode die es erlaubt zwei Objekte vom Typ `ComplexNumber` zu addieren.

Summe

> Schreiben Sie ein Programm, dass zwei Zahlen interaktiv einliest. Die Summe der beiden Zahlen soll ausgegeben werden.

Die Lernziele dieses Beispiels sind:

- Interaktives Einlesen von Eingaben
- Einfache Operationen auf Zahlen

Listing 13 zeigt eine Implementierung des Beispiels in Java.

```
1  import java.util.Scanner;
2
3  class SumOfTwoNumbers {
4
5      public static void main(String[] args) {
6          Scanner scanner = new Scanner(System.in);
7
8          System.out.print("First number: ");
9          int number1 = scanner.nextInt();
10
11         System.out.print("Second number: ");
12         int number2 = scanner.nextInt();
13
14         int sum = number1 + number2;
15
16         System.out.println(number1 + " + " + number2 + " = " +
17                            sum);
18     }
19
20 }
```

Listing 13: Java-Code zur Berechnung der Summe von zwei Zahlen

Im Vergleich zur Java-Version von „Hello World“ ergeben sich zusätzlich folgende Fragestellungen:

1: `import java.util.Scanner;`

- Was bedeutet `import`?
- Was bedeutet `java.util.Scanner`? Ist das so etwas Ähnliches wie `System.out.println`?

6: `Scanner scanner = new Scanner(System.in);`

- Warum schreiben wir hier dreimal Scanner? Macht es einen Unterschied ob wir Groß- oder Kleinschreibung verwenden?
- Welche Bedeutung hat das Gleichheits-Zeichen?
- Wozu benötigen wir `new`?
- Ist `System.in` das Gegenteil von `System.out`?

9,12: `int number1 = scanner.nextInt();`

- Was bedeutet `int`? Was `number1`?
- Was macht `scanner.nextInt()`?
- Warum schreiben wir nicht einfach `nextInt()`?

14: `int sum = number1 + number2;`

- Hier werden wohl die beiden Zahlen addiert?
- Brauchen wir `int` um zu zeigen, dass es sich bei `sum` um eine Zahl handelt?
- Wenn das so wäre, warum benötigen wir kein `int` auf der rechten Seite?

16: `System.out.println(number1 + " + " + ...);`

- Warum soviele Pluszeichen?

Wie schon bei „Hello World“ sehen wir, dass viele der aufgeworfenen Fragen nicht direkt mit den Lernzielen in Zusammenhang stehen. Die Themengebiete Packages (Zeile 1) und Objekte (Zeile 6) werden erst später im Unterricht behandelt. Wieder zeigt sich, dass eine Trennung verschiedener Themengebiete in Java sehr schwer möglich ist.

In diesem Zusammenhang wollen wir die Implementierung des Beispiels in Python (Listing 14) analysieren.

```
1 number1 = input("First number: ")
2 number1 = int(number1)
3 number2 = input("Second number: ")
4 number2 = int(number2)
5 total = number1 + number2
6 print(number1, " + ", number2, " = ", total)
```

Listing 14: Python-Code zur Bestimmung der Summe zweier Zahlen

Wie schon bei „Hello World“ erkennen wir, dass bis auf die Lernziele praktisch nur Themen angesprochen werden, die im klassischen Programmierunterricht schon vorher unterrichtet wurden:

1. Werte
2. Typen
3. Variablen und Variablenzuweisung
4. Zugriff auf Variablenwerte

Alle diese Gebiete können in Python einzeln behandelt werden. Benötigt wir immer nur das Wissen über die vorhergehenden Themen. Im Gegensatz zu Java kann der Unterricht aufbauend gestaltet werden ohne auf Punkte des späteren Unterrichts (Packages, Objekte,…) verweisen zu müssen. Um zu zeigen, dass diese Aussage nicht aus der Luft gegriffen sind, hierzu ein triviales und gültiges Programm, dass nur aus einem Wert besteht: `42`. Hier ein Programm, dass diesen Wert ausgibt: `print(42)`. Auch Typen können in Python leicht bestimmt werden. Hierzu ein Programm, dass den Typ der Zahl 1.23 auf der Konsole ausgibt `print(1.23.__class__)`.

Nachdem wir gesehen haben, dass die angesprochenen Themen gut mittels der Programmiersprache Python vermittelt werden können, wollen wir zurück zu den „neuen“ Themengebieten des Summen-Beispiels gehen. Anhand der ersten vier Zeilen des Beispiels kann ein Programmieranfänger lernen, dass Funktionen auch Werte zurückgeben können.

Wiederum kann dieses Wissensgebiet auch einzeln erklärt werden, schließlich sind die einzelnen Zeilen auch schon für sich alleingestellt gültige Python-Programme. Falls man am Anfang noch nicht auf die Funktion `input` eingehen will, dann kann man statt dessen zum Beispiel die aus dem Mathematik-Unterricht bekannte Funktion `abs` verwenden um Rückgabewerte von Funktionen zu erklären. Die Funktion `int` kann auch extra, zum Beispiel beim Wissensgebiet der Typen, durchgenommen werden.

Gehen wir davon aus, dass dem Schüler schon vermittelt worden ist, dass Funktionen Werte zurückgeben können. In diesem Fall bleibt dem Lehrenden bei Zeile 1 und 3 nur mehr übrig dem Schüler die Wirkungsweise von `input` zu erklären. Diese Erklärung stellt im direktem Zusammenhang mit dem ersten Lernziel des Beispiels: „Interaktives Einlesen von Eingaben“.

Nachdem die Ausgabe von Zahlen schon durchgenommen wurde (Zeile 6) bleibt nur mehr die Erklärung der Addition übrig, die direkt mit Lernziel 2 „Einfache Operationen auf Zahlen“ im Zusammenhang steht.

Wir erkennen wiederum, dass es Python ermöglicht Wissensgebiete sehr schön zu trennen und einzeln durchzunehmen. Aufgabenstellungen können so gestaltet werden, dass

die neuesten Lernziele im Vordergrund stehen. Andere Gebiete die zur Lösung des Beispiels notwendig sind können schon vorher gesondert durchgenommen werden.

Kommen wir nun zur Lösung der Aufgabenstellung mittels der Programmiersprache C. Wie wir in Listing 15 erkennen benötigen wir für ein volles Verständnis der Aufgabe im Vergleich zur Lösung in Python wieder zusätzliches Wissen. Wie schon bei „Hello World“ sollte der Schüler über das Einbinden von Libraries (Zeile 1) Bescheid wissen. Weiters fordert die Aufgabe auch Wissen über den Adressoperator `&` (Zeile 8 und 11).

```
1  #include <stdio.h>
2
3  int main ()
4  {
5      int number1, number2, total;
6
7      printf("First Number: ");
8      scanf("%d", &number1);
9
10     printf("First Number: ");
11     scanf("%d", &number2);
12
13     total = number1 + number2;
14
15     printf("%d + %d = %d", number1, number2, total);
16
17     return 0;
18 }
```

Listing 15: C-Code zur Berechnung der Summe von zwei Zahlen

Ein weiteres Problem für einen Programmieranfänger kann bei diesem Beispiel die Syntax von `scanf` darstellen. Sie mag auf den ersten Blick nicht unbedingt einleuchtend wirken, außer wenn man sich schon vorher öfters mit Formatspezifikationen — z.B. im Zusammenhang mit der Funktion `printf` — auseinandergesetzt hat.

Als explizite Erleichterung der Aufgabe im Vergleich zur Python-Version kann das Weglassen der expliziten Typ-Konversion in eine Ganzzahl erwähnt werden. Diese erfolgt direkt durch die Angabe der Formatspezifikation beim Aufruf der Funktion `scanf`. Dabei sollte aber nicht unerwähnt bleiben, dass manuelle Typ-Konversionen in der Programmiersprache C früher oder später auch für Programmieranfänger ein wichtiges Thema sind. Durch die Verwendung von Zeigern und Adressoperatoren wird diese Typ-Konversion schnell wesentlich komplizierter als unter Python, dass auf die direkte Ver-

wendung von Adressen verzichtet.

Zusammenfassend lässt sich auch bei der Verwendung von C erkennen, dass für die Aufgabenstellung teilweise wieder Vorgriffe auf spätere Themenkomplexe notwendig sind. Zur vollen Verständnis des Operators `&` sollte zum Beispiel das Gebiet Zeiger und Adressen erklärt werden. Eine Thema das teilweise auch fortgeschrittene Programmierer vor Problem stellt. Hier zeigt sich, dass C als Sprache mit relativ niedrigen Abstraktionslevel nicht unbedingt gut geeignet ist Anfänger-Beispiele zu bearbeiten, da diese teilweise von einem höheren Abstraktionslevel ausgehen.

Mittelwertabweichung

> Schreiben Sie eine Methode, die alle Werte eines Arrays liefert, die mindestens 10% vom Mittelwert abweichen.

Bei diesem Beispiel beginnen wir mit der in Listing 16 gezeigten Implementierung in C.

```c
#include <math.h>

int filterMinDerivation10Percent(double values[], int length) {
  int strays_index = 0;
  double mean = 0;

  for (int index = 0; index < length; index++)
    mean += values[index] / length;

  for (int index = 0; index < length; index++)
    if (fabs(values[index] - mean) >= 0.1 * fabs(mean))
      values[strays_index++] = values[index];

  return strays_index;
}
```

Listing 16: C-Funktion zur Filterung von Werten rund um den Mittelwert

Eine Besonderheit dieses Codes stellt das Handling des Arrays dar. Statt wie in den noch folgenden Beispielen ein neues Array zu erstellen, wird das alte Array überschrieben. Zum Schluss wird die Anzahl der Element des gefilterten Arrays zurückgegeben. Diese Vorgehensweise wurde gewählt, da in C die Länge eines *dynamischen* Arrays nicht direkt an Hand der Adresse bestimmt werden kann. Beim erstellen eines neuen Arrays müssten ansonsten folgende Punkte beachtet werden:

- Speicher für das neue Array muss reserviert werden (`malloc`),

- Die Adresse des neuen Arrays muss an die Funktion übergeben werden

Beide Punkte fordern vom Programmieranfänger ein relativ ausgeprägtes Wissen über die Architektur eines Rechners. Während dieses Wissen für einen fortgeschrittenen Programmierer sicher sehr hilfreich ist, kann die Fülle an Information, die notwendig ist um dieses Beispiel — unter Verwendung eines neuen Arrays — zu lösen einen Programmieranfänger vor Probleme stellen.

Beim Code an sich fallen die klassischen dreiteiligen `for`-Schleifen auf. Dieses Konstrukt stellt für fortgeschrittene Programmierer kein größeres Problem mehr dar. Für einen Programmieranfänger ist der Programmfluss bei dieser Art der `for`-Schleife allerdings nicht unbedingt einleuchtend. Statt von links nach rechts und oben nach unten verläuft der Programmfluss teilweise auch von rechts nach links (siehe Abbildung 5.9). Diese Schwierigkeit kann durch die Verwendung einer `while`-Schleife oder besser einer sogenannten `foreach`-Schleife beseitigt werden. Leider unterstütz C, das letztgenannte Konstrukt nicht.

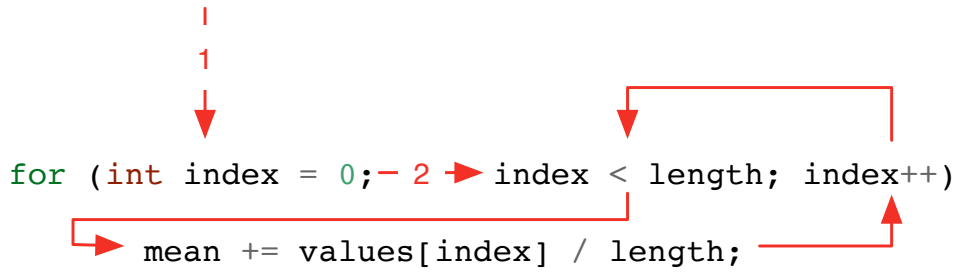


Abbildung 5.9: Programmfluss eine klassischen for-Schleife

Kommen wir nun zu der Realisierung der Aufgabenstellung in Java (siehe Listing 17). Der Code ähnelt der Version in C. Bei Java müssen wir die Länge des Arrays allerdings nicht extra angeben, da diese über das Feld `length` ermittelt werden kann. Ein weiterer Vorteil ist, dass die explizite Reservierung von Speicher für die Erstellung eines Arrays entfällt. Weiters ermöglicht uns Java auch die Verwendung einer „for-each“-Schleife, die im Vergleich zur klassischen `for`-Schleife einen logischeren Programmfluss bietet und einen Zugriff auf Elemente außerhalb des Arrays verhindert. All diese Vorteile ermöglichen es einem Programmieranfänger sich besser auf die eigentliche Aufgabenstellung zu konzentrieren.

```java
import java.util.Arrays;

public class Derivation {

    static double[]
        filterMinDerivation10Percent(double[] values) {

        int index = 0;
        double mean = 0;
        double[] strays = new double[values.length];

        for (double value : values)
            mean += value/values.length;

        for (double value : values)
            if (Math.abs(value-mean) >= 0.1 * Math.abs(mean))
                strays[index++] = value;

        return Arrays.copyOf(strays, index);
    }

}
```

Listing 17: Java-Funktion zur Filterung von Werten rund um den Mittelwert

Zum Schluss wollen wir auf die in Listing 18 vorgestellte Python-Lösung der Aufgabenstellung eingehen. Ein Punkt der sofort auffällt ist der Wegfall der `for`-Schleifen. Statt dessen kommt die Funktion `sum` und eine List-Comprehension zum Einsatz. Die Lösung in Python ähnelt einer mathematischen Lösung des Beispiels. Gerade die Nähe zur Mathematik — einem Fach, dass Schüler zu diesem Zeitpunkt schon gut kennen — ermöglicht es Programmieranfängern diesen Code schnell zu verstehen.

```python
def filterMinDerivation10Percent(values):
    mean = sum(values)/len(values)
    return [value for value in values
            if abs(value-mean) > 0.1 * abs(value)]
```

Listing 18: Python-Funktion zur Filterung von Werten rund um den Mittelwert

Die in Listing 18 gezeigte List Comprehension lässt sich mit wenig Aufwand eine äquivalente Lösung mit `for`-Schleife transformieren. Die transformierte Lösung, zu sehen in Listing 19, kann verwendet werden um Schülern die Lösung des Beispiels unter Verwen-

dung klassischer Konzepte zu lehren. Dieses Beispiel zeigt, dass Python gut geeignet ist Neulinge schrittweise in die Programmierung einzuführen. Listing 19 kann als natürliche Vorstufe der vorgestellten Lösungen in Java und Python gesehen werden.

```
def filterMinDerivation10Percent(values):
    mean = sum(values)/len(values)

    filtered_values = []
    for value in values:
        if abs(value-mean) > 0.1 * abs(value):
            filtered_values.append(value)
    return filtered_values
```

Listing 19: Verwendung einer for-Schleife zur Filterung

Sortieren von Dateiinhalt

> Schreiben Sie Code, der den Inhalt einer Datei zeilenweise einliest. Sortieren Sie dann die eingelesenen Zeilen und speichern Sie das Ergebnis in einer neuen Datei.

Wie Listing 20 zeigt ist das Schreiben und Lesen von Dateien in C mit einigermaßen großen Aufwand verbunden. Obwohl die angegebene Lösung, im Vergleich zu den Lösungen in Java und Python, die meisten Zeilen benötigt verfügt sie über den kleinsten Funktionsumfang. Wir gehen bei dem Code davon aus, dass

- die eingelesene Datei nur über eine bestimmte Zeilenanzahl verfügt,
- alle Zeilen eine maximale Länge besitzen,
- die Datei nur aus ASCII-Zeichen besteht.

Wollten wir die ersten zwei angesprochenen Punkte besser lösen, dann müssen wir uns explizit um die Freigabe von Speicher bemühen. Die Unterstützung eines andere Encodings, also die Lösung des letzten Punktes, ist für einen Programmieranfänger in der Sprache C wohl zu komplex, da die Sprache für diese Aufgabe keine Standardfunktionen zur Verfügung stellt.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_NUMBER_LINES 1024
#define MAX_LINE_LENGTH 1024

int compare(const void *first, const void *second) {
  return strcmp((char *)first, (char *)second);
}

int main() {
  char filepath_input[] = "test.txt";
  char filepath_output[] = "test_sorted.txt";
  char lines[MAX_NUMBER_LINES][MAX_LINE_LENGTH];

  int line_index = 0;

  FILE *input, *output;

  if ((input = fopen(filepath_input, "r")) == NULL) {
    fprintf(stdout, "Could not open %s for reading", filepath_input);
    return 1;
  }

  while (fgets(lines[line_index++], MAX_LINE_LENGTH, input) != NULL);

  fclose(input);

  if (ferror(input) != 0) {
    fprintf(stdout, "An error occurred while reading %s", filepath_input);
    return 1;
  }

  qsort(lines, line_index, sizeof(char) * MAX_LINE_LENGTH, compare);

  if ((output = fopen(filepath_output, "w")) == NULL) {
    fprintf(stdout, "Could not open %s for writing", filepath_input);
    return 1;
  }

  for (int line = 0; line < line_index; ++line)
    fprintf(output, "%s", lines[line]);

  fclose(output);

  return 0;
}
```

Listing 20: Sortieren der Zeilen einer Datei in C

Ein Punkt der erwähnt werden sollte ist, dass in C ein explizites Reagieren auf Fehler unerlässlich ist. In Java oder Python führen Fehler während der Programmausführung zu zumindest teilweise hilfreichen Fehlermeldungen. In C ist das nicht der Fall. Beispielsweise bricht das in Listing 20 vorgestellte Programm die Ausführung mit der kryptischen Fehlermeldung

```
Segmentation fault: 11
```

ab, falls in *Zeile 21* nicht explizit überprüft wird ob die zu öffnende Datei auch existiert.

Ein weiterer Punkt, der bei der Lösung auffällt ist die Verwendung von Zeigern. Wie schon bei der vorhergehenden Aufgabenstellung erwähnt benötigt auch ein Anfänger Wissen über dieses anfänglich komplexe Konzept. Es zeigt sich hier wiederum, dass C explizit *nicht* als Sprache für Programmieranfänger konzipiert wurde.

Kommen wir nun zu der Lösung in Java. Listing 21 zeigt eine relativ moderne Implementierung der Aufgabenstellung (Java 1.7+). Im Vergleich zur Lösung in C fällt auf, dass die Dateien nicht explizit geschlossen werden müssen. Außerdem können wir das Encoding der Dateien frei wählen (Zeile 14).

Für den Anfänger ergibt sich eine Vereinfachung, da er sich nicht direkt mit der Adressierung (Zeiger) beschäftigen muss. Die Sortierung der Linien kann mit der vorgefertigten Funktion `Collections.sort` erfolgen. Im Gegensatz zu der Lösung in C benötigen wir keine zusätzliche Hilfs-Funktion zur Sortierung.

Als Nachteile zur C-Version kann die hohe Zahl an `import`-Befehlen gesehen werden. Üblicherweise stellt die Verwendung von imports aber kein größeres Problem für Anfänger dar. Ansonsten bieten diverse IDEs auch an, dass Importieren von Libraries automatisch zu managen. Ein Eingriff vom Benutzer ist dabei nicht notwendig.

Ein Konzept, dass in der Sprache C nicht vorkommt sind die sogenannten „Generics“. Dieses Konstrukt wird in Zeile 13 verwendet um den Typ der `ArrayList` zur Speicherung der Zeilen anzugeben. Während es sich hier durchaus um ein komplexeres Konzept handelt, kann die reine Verwendung bei einer *eindimensionalen* Liste einem fortgeschrittenen Anfänger gut erklärt werden. Grundsätzlich ist der kognitive Aufwand um das zweidimensionale Array zu verstehen (C-Lösung) höher einzuschätzen, als der Aufwand der zur Verständnis der generischen `ArrayList` notwendig ist.

Insgesamt lässt sich feststellen, dass die Lösung in Java einfacher zu programmieren und verstehen ist wie die Lösung in C. Bei beiden Lösungen sollte der Lernende aber schon vorher genügend Wissen beim Programmieren einfacher Beispiele gesammelt haben.

```java
import java.util.ArrayList;
import java.util.Collections;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.file.Files;
import java.nio.file.Paths;

class SortLines {

    public static void main(String[] args) {
        ArrayList<String> lines = new ArrayList<>();
        Charset charset = Charset.forName("UTF-8");

        try (BufferedReader reader =
             Files.newBufferedReader(Paths.get("test.txt"), charset)) {

            String line = null;
            while ((line = reader.readLine()) != null)
                lines.add(line);

        } catch (IOException exception) {
            System.err.format("IOException: %s%n", exception);
        }

        Collections.sort(lines);

        try (BufferedWriter writer = Files.newBufferedWriter(
                Paths.get("test_sorted.txt"), charset)) {

            for (String line : lines) {
                writer.write(line, 0, line.length());
                writer.newLine();
            }

        } catch (IOException exception) {
            System.err.format("IOException: %s%n", exception);
        }
    }
}
```

Listing 21: Sortieren der Zeilen einer Datei in Java

Zum Schluss möchten wir auf die Python-Lösung der Aufgabenstellung eingehen. Schon die Zeilenzahl von Listing 22 zeigt uns, dass das Bearbeiten von Dateien in Python mit deutlich weniger Aufwand verbunden ist als bei den beiden anderen Programmiersprachen.

Der Programmieranfänger muss sich hier nicht wie in C darum kümmern die Datei zu schließen. Der Zeilenzugriff kann direkt mittels der schon bekannten `for`-Schleife erfolgen. Auch eine Verwendung von Generics ist nicht nötig, da Listen beliebige Datentypen aufnehmen können. Der Aufwand zum Verständnis des Beispiels liegt sicherlich deutlich unter dem der Java-Version.

```python
with open("test.txt", 'r', encoding='utf_8') as input_file:
    lines = [line for line in input_file]

lines = sorted(lines)

with open("test_sorted.txt", 'w', encoding='utf_8') as output_file:
    for line in lines:
        output_file.write(line)
```

Listing 22: Sortieren der Zeilen einer Datei in Python

Wir verwenden bei diesem Beispiel kein explizites Error-Handling, da Python auch so notwendige Information über Laufzeitfehler zur Verfügung stellt. Der folgende Text zeigt beispielsweise die Fehlermeldung falls die zu lesende Datei nicht existiert:

```
Traceback (most recent call last):
  File "sort_lines.py", line 1, in <module>
    with open("tst.txt", 'r', encoding='utf_8') as input_file:
FileNotFoundError: [Errno 2] No such file or directory: 'tst.txt'
```

Der Unterschied zur äquivalenten Fehlermeldung in C — `Segmentation fault: 11` — sind frappant. So wird dem Benutzer — also im unseren Fall dem Programmieranfänger — nicht nur die Zeile mittgeteilt in der der Fehler auftritt, sondern auch noch eine gute Beschreibung geliefert warum der Fehler auftritt.

Dieses Verhalten von Python ermöglicht — wie schon oft vorher in diesem Abschnitt festgestellt — eine gute Trennung der Themengebiete. Während in C bei dieser Aufgabenstellung Wissen über Fehlerbehandlung unerlässlich ist, und der Java-Code Kenntnisse über die Behandlung von Exceptions fordert, können wir die Behandlung von Exceptions in Python getrennt behandeln.

Komplexe Summe

> Schreiben sie eine Klasse `ComplexNumber`. Objekte dieser Klasse stellen komplexe Zahlen dar. Implementieren Sie eine Methode die es erlaubt zwei Objekte vom Typ `ComplexNumber` zu addieren.

Da C keine Unterstützung für OOP bietet, verzichten wir auf die Implementierung dieser Aufgabe in dieser Programmiersprache. Statt dessen konzentrieren wir uns auf die Implementierung in Java und Python. Wie schon im Abschnitt „Paradigmen-Unterstützung“ gesehen haben, handelt es sich bei Java — Version 1.8 und früher — um eine rein objektorientierte Sprache. Python hingegen unterstützt schon seit Version 1.0 eine Vielzahl von Programmierparadigmen [Wik15b]. Gerade in dieser Hinsicht ist ein Vergleich beiden Sprachen hinsichtlich der Unterstützung von OOP interessant. Hat hier Java auf Grund der Konzentration auf die Objektorientierte Programmierung einen Vorteil?

```java
class ComplexNumber {

    private int real;
    private int imaginary;

    ComplexNumber(int real, int imaginary) {
        this.imaginary = imaginary;
        this.real = real;
    }

    ComplexNumber add(ComplexNumber other) {
        return new ComplexNumber(real + other.real,
                                 imaginary + other.imaginary);
    }

    public String toString() {
        String sign = "";
        if (imaginary >= 0) sign = "+";

        return real + sign + imaginary + "j";
    }

}
```

Listing 23: Java-Code für eine Klasse die komplexe Zahlen repräsentiert

```python
class ComplexNumber(object):

    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary

    def __add__(self, other):
        return ComplexNumber(self.real + other.real,
                             self.imaginary + other.imaginary)

    def __repr__(self):
        return "{}{}{}j".format(self.real, "+" if self.imaginary >= 0 else "",
                                self. imaginary)
```

Listing 24: Python-Code für eine Klasse die komplexe Zahlen repräsentiert

Listing 23 zeigt eine einfache Implementierung der Aufgabenstellung in Java. Listing 24 zeigt eine gleichwertige Implementierung in Python. Wie wir sehen unterscheidet sich der Code diesmal nicht so stark wie bei den anderen Aufgabenstellungen. Einige Unterschiede die auffallen:

1. Python verwendet das Wort `self`, statt dem unter Java gebräuchlichen Schlüsselwort `this`. Anzumerken ist dabei, dass statt `self` auch jeglicher andere Name verwendet werden kann. Der erste Parameter einer Methode bezeichnet automatisch das Objekt selbst!

2. In Java ist `this` beim Variablenzugriff optional, sofern im aktuellen Bereich keine Variable mit gleichem Namen existiert (siehe Listing 23, Zeile 12). In Python muss `self` verwendet werden, wenn man auf Objektvariablen zugreifen will (siehe Listing 24, Zeile 8).

3. Python-Objekte besitzen keinen klassischen Konstruktor, statt dessen werden die spezielle Methoden `__new__()` und `__init__()` verwendet. Für jedes Objekt können diverse vordefinierte Methoden implementiert werden. In unserem Beispiel haben wir z.B. die beiden Methoden `__repr__()` und `__add__()` verwendet. Die erste Methode wird automatisch aufgerufen, wenn die String-Representation eines Objekts benötigt wird (vergleichbar zu `toString` in Java), die zweite, wenn zwischen zwei Objekten ein Plus-Zeichen steht.

4. Python unterstützt nur eine sehr limitierte Form des „Information Hiding“. Ein Zugriff auf Variablen die mit zwei Unterstrichen beginnen ist z.B. nicht direkt möglich:

```
>>> class Test:
...   def __init__(self):
...       __hidden = "Do not look at me!"
...
>>> test = Test()
>>> test.__hidden
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Test' object has no attribute '__hidden'
```

   Diese Maßnahme lässt sich aber in (Python 3) leicht umgehen. Ein indirekter Zugriff auf die Variable des gezeigten Beispiels ist zum Beispiel mit Hilfe des Befehls `test._Test__hidden` möglich.

   Java bietet im Gegensatz zu Python eine erweiterte Form des „Information Hiding“ und unterstützt damit die sogenannte Kapselung von Code. Mittels der Schlüsselwörter `private`, `protected` und `public` kann geregelt werden wo im Code der Zugriff auf bestimmte Attribute möglich ist.

   Grundsätzlich erlaubt Python jedem Programmierer selbst darüber zu bestimmen auf welche Attribute er zugreift und auf welche nicht. In Java kann der Attributzugriff hingegen beschränkt werden. Beide Philosophien bringen Vor- und Nachteile mit. Grundsätzlich kann aber gesagt werden, dass das Management von Zugriffsrechten die Programmierung komplexer macht, da über diese Komponente auch noch nachgedacht werden muss. Deshalb bietet sich das von Python gewählte einfachere Modell für Programmieranfänger an.

Abgesehen vom Attributzugriff ergeben sich zwischen den Sprachen keine größeren Unterschiede bei dieser Aufgabe. In dieser Hinsicht kann gesagt werden, dass sich durch den Multiparadigma-Ansatz von Python keine Nachteile bei der Objektorientierten Programmierung ergeben.

Zum Schluss wollen wir bei diesem Beispiel noch darauf hinweisen, dass Python direkt Unterstützung für komplexe Zahlen bietet:

```
>>> (1 + 2j) + (3 + -4j)
(4-2j)
```

Dieser Support kann zum Beispiel verwendet werden um die Korrektheit der Implementierung des Beispiels zu überprüfen.

Zusammenfassung

Tabelle 5.3 zeigt einen abschließenden Vergleich der Programmiersprachen.
Wie wir anhand der vorgestellten Beispiele zeigen konnte, zeichnen sich Implementierungen in Python im Vergleich zu C und Java durch ihre Einfachheit und Kompaktheit aus. Die Gründe für den Mehraufwand in der Programmierung bei den letztgenannten Sprachen sind vielschichtig.

Bei C ist es vor allem der niedrige Abstraktionslevel zu nennen. Die wegfallende Abstraktion des Speichers — Zugriff mittels Zeiger, explizites Speichermanagement — kann dazu führen, dass Anfänger hier vor allem im späteren Verlauf des Lernprozesse, wenn diese Themen durchgenommen werden, Probleme bekommen.

Bei Java führt vor allem die Notwendigkeit sofort objektorientiert Programmieren zu müssen dazu, dass schon ein einfachstes Programm für viele Fragen beim Anfänger sorgt. Viele der für praktisch jedes Java-Programm notwendigen Konstrukte werden erst später erklärt. Dieser Umstand führt zu einer hohen Anfangsschwelle beim Einstieg in die Programmierung.

Bei Python gibt es diese Anfangsschwelle nicht. Wie wir zeigen konnten lassen sich Themengebiete hier sehr gut trennen. Die Einführung in die Programmierung kann somit schrittweise erfolgen. Der hohe Abstraktionslevel von Python lässt eine Überführung vieler Konzepte aus dem Mathematikunterricht in die Programmierung zu. Python-Code ist vielfach sehr nah an dem was im Mathematikunterricht als sogenannter Pseudocode bekannt ist. Die Programmiersprache eignet sich damit hervorragend um die grundlegenden Konzepte der Programmierung (Variablen, Zuweisung, Abfragen, Schleifen) zu lehren.

Potentiell würden wir Java und C als gute Zweitsprachen einschätzen. Nachdem Schüler die grundlegenden Konzepte in Python gelernt haben, können Sie sich je nach Interessen-

Tabelle 5.3: *Zusammenfassender Vergleich der Programmiersprachen*

		C	Java	Python
Kompilieren, Ausführen	eine Datei	+	+	++
	mehrere Dateien	- -	•	+
Trennbarkeit Themen		•	-	++
Paradigmen		prozedural	objekt-orientiert, funktional[1]	prozedural, objekt-orientiert, funktional
Abstraktionslevel		niedrig	mittel	hoch
Komplexe Themen		Zeiger, Adressen	Generics	Meta-Programming
Codekompaktheit[2]		-	•	+
Geschwindigkeit[3]		++	+	-
Lesbarkeit		- -	•	+
Einsatzgebiete		Embedded Systems, Betriebssysteme	Business, Internet	Automation, Internet

[1] Die Unterstützung für funktionale Programmierung wurde mit Version 1.8 eingeführt.

[2] Wieviele Zeilen Code werden ca. zur Lösung der Aufgabenstellung benötigt?

[3] Die durchschnittliche Ausführungsgeschwindigkeit von einem übersetzten Programm. Dabei gehen wird davon aus, dass der gleiche Algorithmus bei den Implementierungen zum Einsatz kommt.

lage einer der beiden anderen Sprachen zuwenden. C bietet sich beispielsweise an wenn mehr über die grundsätzliche Funktionsweise eines Rechners gelehrt werden soll. Java kann verwendet werden um über objektorientierte Konzepte in einer statisch typisierten Sprache zu lehren.

Lehrerbefragung

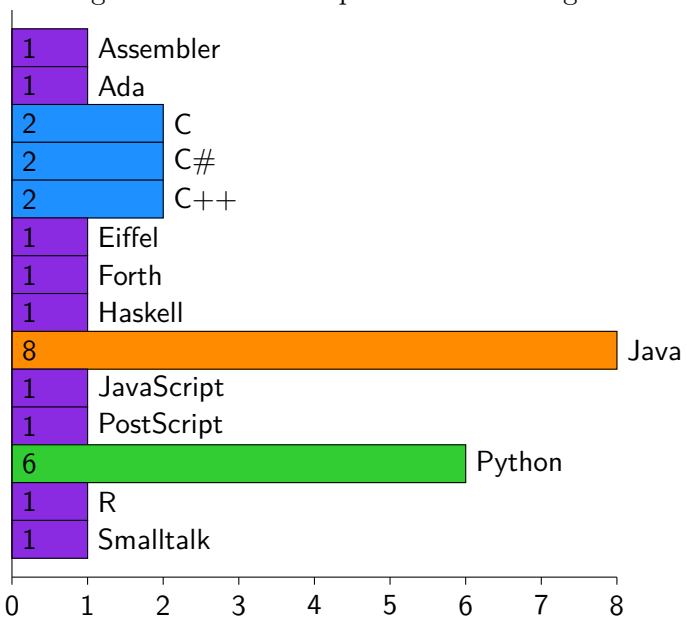
Während der Abschnitt „Anfängerbeispiele“ unsere Einschätzung zu den verschieden Programmiersprachen zeigt, kommen in diesem Teil Informatik-Lehrer zu Wort. Zu diesem Zweck wurden einerseits Interviews und andererseits eine Umfrage durchgeführt. Das Lehrpersonal für die Umfrage und Interviews ist dabei größtenteils gleich. Während uns für die Interviews 10 Personen zur Verfügung standen, waren es bei der Umfrage 12. Da die Umfrage anonym durchgeführt wurde ist kein direkter Rückschluss von interviewten auf befragtes Lehrpersonal möglich. Lehrer 1 in den Interviews ist also mit hoher Wahrscheinlichkeit ein andere Person als Lehrer 1 in den Umfragen.

Kommen wir nun zu der Einteilung dieses Abschnitts. Als erstes widmen wir uns den Daten der Interviews und der Umfrage. Danach analysieren wir das Ergebnis und versuchen auf einige der Kritikpunkte des Lehrpersonals einzugehen.

Interviews

Für die Interviews wurden zehn (männliche) Lehrpersonen zu dem Thema Anfängersprachen befragt. Sechs der befragten Personen unterrichten dabei an einer Höhere Technische Lehranstalt (HTL) während die restlichen vier Personen an der Universität tätig sind. Die Erfahrung der Lehrer reichen dabei von ca. einem Jahr bis zu ca. 39 Jahren beim erfahrensten Lehrer. Die beiden Lehrer um den Median unterrichten dabei ca. 16 bzw. 17 Jahre lang. Abbildung 5.10 zeigt welche Sprachen von den interviewten Lehrern unterrichtet werden.

Abbildung 5.10: Unterrichtssprachen der befragten Lehrer



Die Länge der geführten Interviews liegt zwischen ca. sieben Minuten für das kürzeste Interview und ca. einer Stunde und 15 Minuten für das längste Interview. Um einen Überblick über die besprochenen Themen zu geben haben wir dabei bestimmte Punkte in Tabellen zu den Gebieten:

1. Eigenschaften einer Programmiersprache für Anfänger (Tabelle 5.4, Tabelle 5.5, Tabelle 5.6),
2. Python als Anfängersprache (Tabelle 5.7, Tabelle 5.8, Tabelle 5.9), und
3. Allgemeines zum Unterricht (Tabelle 5.10, Tabelle 5.11, Tabelle 5.12)

zusammengefasst. Die Tabellen zeigen dabei auch teilweise eine Analyse des Interviews, da Aussagen der Lehrer in Kategorien gegliedert wurden.

Insgesamt äußerten sich die meisten Lehrer positiv (6 von 10) zum Einsatz von Python als erste Programmiersprache. Zwei Lehrer zeigten eine eher neutrale Haltung gegenüber der Sprache (Lehrer 2 und Lehrer 9), während zwei der befragten Personen (Lehrer 6 und Lehrer 7) eine explizit negative Haltung gegenüber Python einnahmen. Eine nähere Deutung und Analyse der Interviews zusammen mit den Daten der Umfrage erfolgt im Abschnitt „Analyse“.

Tabelle 5.4: Lehrerinterviews — Eigenschaften Anfängersprache

	Lehrer 1	**Lehrer 2**	**Lehrer 3**	**Lehrer 4**
Sprache als Motivation	Egal	Egal	Ja	Bedingt
Industriesprache (Verbreitung)	Egal	—	Egal	—
Multi-Plattform-Unterstützung	—	Wichtig	—	—
Interaktive Sprache	Vorteilhaft	—	Vorteilhaft	—
Dynamische Typisierung	Kein Problem	Nicht direkt beantwortbar	Vorteilhaft	Vorteilhaft
Sprachempfehlung	Python	Keine	—	Python
Unterrichtssprache	Python	C#	Java, Forth	Java, Processing, Python
Syntax	—	Unwichtig	—	Eher unwichtig

Tabelle 5.5: Lehrerinterviews — Eigenschaften Anfängersprache

	Lehrer 5	**Lehrer 6**	**Lehrer 7**
Sprache als Motivation	Eingeschränkt	Eingeschränkt	Nein
Industriesprache (Verbreitung)	Unwichtig	Wichtig	Wichtig
Multi-Plattform-Unterstützung	—	—	—
Interaktive Sprache	—	Egal	Vorteilhaft
Dynamische Typisierung	Vorteilhaft / Später problematisch	Problematisch	Problematisch
Sprachempfehlung	Python	Sprache mit C ähnlicher Syntax, JavaScript	C#, Java
Unterrichtssprache	Python	C, Java, Python	C#, Java, Python
Syntax	Einfache Syntax hilfreich	Wichtig	Einfache Syntax hilfreich

Tabelle 5.6: Lehrerinterviews — Eigenschaften Anfängersprache

	Lehrer 8	**Lehrer 9**	**Lehrer 10**
Sprache als Motivation	Teilweise	Ja	Ja
Industriesprache (Verbreitung)	—	Vorteilhaft	—
Multi-Plattform-Unterstützung	—	—	—
Interaktive Sprache	Hilfreich	—	—
Dynamische Typisierung	Nachteil bei großen Projekten	Kein Problem	—
Sprachempfehlung	Python	—	Python
Unterrichtssprache	Java, Python	Java	—
Syntax	—	—	einfache Syntax hilfreich

Tabelle 5.7: Lehrerinterviews — Python als Anfängersprache

	Lehrer 1	**Lehrer 2**	**Lehrer 3**	**Lehrer 4**
Einschätzung	++	•	+	+
Allgemein	Empfohlen	—	—	—
Nicht-Informatiker	—	—	—	Empfohlen
Informatiker	—	—	—	Empfohlen
System-Programmierer	—	—	—	Abgeraten
Vorteile	• Lambda-Funktionen • Decorators • List Comprehensions • Interaktive Eingabe • Bibliotheken (`argparse`, `doctest`) • Einfache Syntax • Schöne Sprache	• Plattform-Support	• Keine Trennung Objekt-/Basis -Typen • Kein Compiler	• Daten -Strukturen (Listen, Dictionaries, Sets) • Hoher Abstraktions -Level
Kritikpunkte	—	—	—	• IDE-Support

Tabelle 5.8: Lehrerinterviews — Python als Anfängersprache

	Lehrer 5	**Lehrer 6**	**Lehrer 7**
Einschätzung	+	-	-
Allgemein	—	Abgeraten	Abgeraten
Nicht-Informatiker	—	—	—
Informatiker	—	—	—
System-Programmierer	—	—	—
Vorteile	• Hoher Abstraktionslevel • Bibliotheken (`numpy`) • Verwendung als Taschenrechner	• Netzwerktechnik (Linux) • Listen	• Web-Programmierung
Kritikpunkte	• Entwicklung großer Projekte • Keine Variablendeklaration	• Keine „saubere“ Objekt-Orientierung • rechenintensive Programme	• Limitierte Möglichkeiten (Sprachmächtigkeit)

Tabelle 5.9: Lehrerinterviews — Python als Anfängersprache

	Lehrer 8	**Lehrer 9**	**Lehrer 10**
Einschätzung	++	•	++
Allgemein	Empfohlen	—	Empfohlen
Nicht-Informatiker	Empfohlen	Empfohlen	—
Informatiker	Empfohlen	Abgeraten	—
System-Programmierer	Abgeraten	Abgeraten	—
Vorteile	• Hoher Abstraktions-Level • Automatisierung • Leichte Einbindung von C-Code • Bibliotheken • Weglassen von erweiterten Konzepten (z.B. OOP) möglich	• Sehr gut für erste Schritte • Wenige Konzepte • Gute Glue-Language	• Einfache Syntax • Verwendung als Taschenrechner
Kritikpunkte		• Hürde bei Umstieg auf andere Sprache	

Tabelle 5.10: Lehrerinterviews — Unterricht

	Lehrer 1	**Lehrer 2**	**Lehrer 3**	**Lehrer 4**
Wichtige Punkte Unterricht	• Lehrer-Schüler-Verhältnis • Begeisterung Lehrer • Intensives Üben	• Intensives Üben	• Mehrere Sprachen lernen • Interpretation Programm	• Einfache Verwendung Entwicklungs-Umgebung
Schwierigkeiten Unterricht	• Algorithmisches Denken • Fehlersuche	• Algorithmisches Denken		• Abstraktes Denken (Logik)

Tabelle 5.11: Lehrerinterviews — Unterricht

	Lehrer 5	**Lehrer 6**	**Lehrer 7**
Wichtige Punkte Unterricht	• Algorithmisches Denken (Basiskonzepte)		• Selbstständiges Arbeiten • Intensives Üben
Schwierigkeiten Unterricht	• Abstraktes Denken (Logik)	• Abstraktes Denken (Logik) • Motivation der Schüler	• Abstraktes Denken (Logik) • „Problem Solving“

Tabelle 5.12: Lehrerinterviews — Unterricht

	Lehrer 8	**Lehrer 9**	**Lehrer 10**
Wichtige Punkte Unterricht	• Selbstständiges Lernen	• Selbstständiges Lernen	• Algorithmisches Denken
Schwierigkeiten Unterricht	• Debugging • Schleifen	• Pragmatik / Analogien	• Abstraktes Denken

Umfragen

Die Umfrage unter dem Lehrpersonal kann in zwei größere Themengebiete geteilt werden. Zum einen wurde gefragt in wie fern Java und Python bestimmte Anfängerkriterien erfüllen (Abbildung 5.11 und Abbildung 5.12). Zum anderen stellten wir Fragen zu der Eignung bestimmter Spracheigenschaften von Python für Programmieranfänger (Abbildung 4.3).

Bei der Erfüllung der Kriterien gab es für die befragten Personen folgende Auswahlmöglichkeiten:

- Voll und ganz (4 Punkte)
- Ausreichend (3 Punkte)
- Unzureichend (2 Punkte)
- Überhaupt nicht (1 Punkt)
- Nicht relevant (0 Punkte)

Abbildung 5.11 und Abbildung 5.12 zeigen den Mittelwert der Beantwortungen nach der beschrieben Punkte-Skala. Als nicht relevant wurden dabei von zwei Personen die Eigenschaft „Kleiner Sprachumfang“ eingeschätzt. Jeweils eine Person bewertete die Eigenschaften „Gute Unterstützung beim GUI-Design“ und „Unterstützung für Multiparadigmen“ als nicht relevant. Zusätzlich bewerte eine Person die Eigenschaft „Schritt für Schritt“ in Python zu entwickeln als nicht wichtig.

Abbildung 5.11: Java — Erfüllung von Kriterien an Anfängersprache

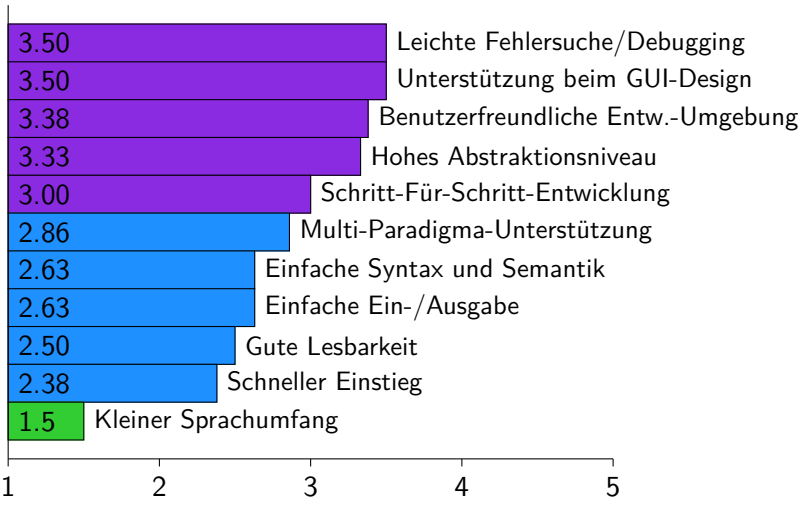
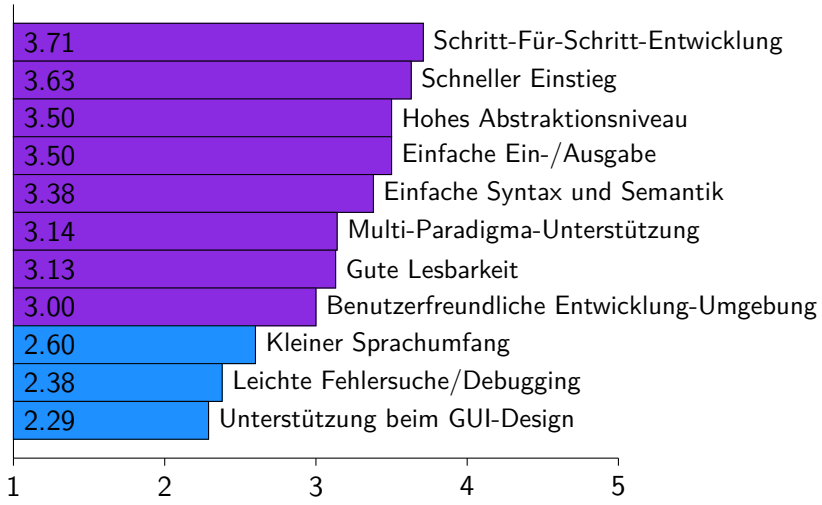


Abbildung 5.12: Python — Erfüllung von Kriterien an Anfängersprache



Bei der Befragung nach der Bewertung der Spracheigenschaften standen folgende Möglichkeiten zur Auswahl:

- Vorteil (2 Punkte)
- Weniger Vorteil (1 Punkt)
- Nachteil (-2 Punkte)
- Weniger Nachteil (-1 Punkt)
- Egal (0 Punkte)

Zwölf Personen beantworteten diese Fragestellungen vollständig. Abbildung 5.13 zeigt den Mittelwert der Antworten. Die Anzahl der Lehrer die die jeweiligen Spracheigenschaften als irrelevant (egal) bezeichneten kann Tabelle 5.13 entnommen werden.

Abbildung 5.13: Python — Vor- und Nachteile von Spracheigenschaften

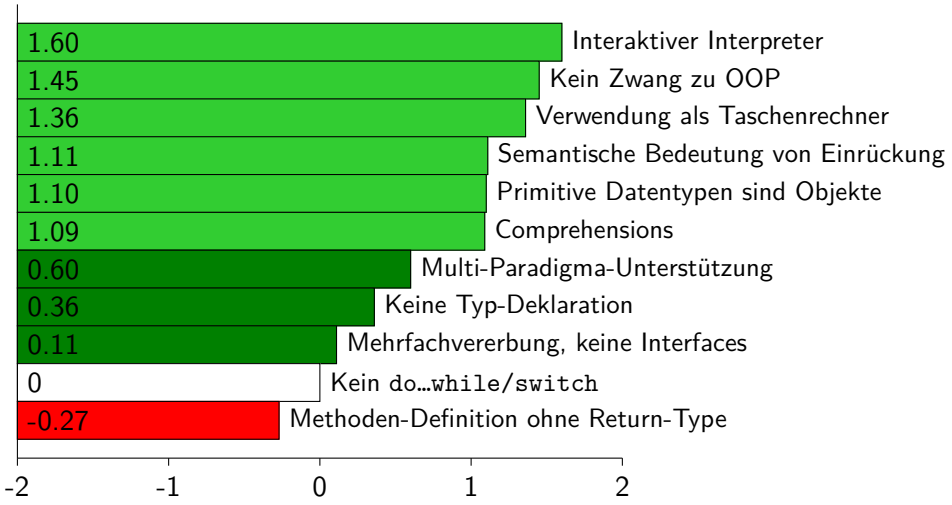


Tabelle 5.13: Anzahl der Personen die bestimmte Spracheigenschaften von Python als irrelevant bewerteten

Spracheigenschaft	**Anzahl Personen**
Kein `do…while/switch`	5
Semantische Bedeutung von Einrückung	3
Primitive Datentypen sind Objekte	2
Multi-Paradigma-Unterstützung	2
Interaktiver Interpreter	2
Verwendung als Taschenrechner	1
Kein Zwang zu OOP	1
Methoden-Definition ohne Return-Type	1
Comprehensions	1

Analyse

Die beiden vorhergehenden Abschnitte widmeten sich der reinen Darstellung der Daten aus den Interviews und den Umfragen. In diesem Abschnitt wollen wir diese Daten analysieren und auf die daraus gewonnenen Kenntnisse eingehen.

Insgesamt ergibt sich bei den Ergebnissen der Umfrage und der Interviews ein durchaus positives Bild für Python. Die Sprache wird dabei von drei Lehrer explizit als die Programmiersprache genannt, die für den Anfängerunterricht benutzt werden sollte. Die anderen Lehrer stehen Python meist auch positiv gegenüber, unterscheiden bei der Empfehlung der Sprache aber nach den Schülern die unterrichtet werden sollen. Beispielsweise wird als bevorzugte Sprache für den Bereich der technischen Informatik (Embedded Systems, Systemnahe Programmierung) meist C oder C++ genannt.

Da die gewonnenen Daten aus den Interviews relativ umfangreich sind, wollen wir diese zunächst einteilen. Dazu haben wir uns entschieden die angesprochenen Themengebiet nach der Übereinstimmung zwischen den Lehrpersonen einzuteilen. Bestimmte Gebiete wurden von den Lehrern sehr unterschiedlich kommentiert, während es bei anderen Themen praktisch keinen Widerspruch zwischen den interviewten Personen gab. Die Liste der *unumstritten Themen* inkludiert:

- **Algorithmisches Verständnis** fehlt bei Anfängern
- **Selbstständiges Lernen** von Schülern notwendig
- **Motivierung durch effektvolle Beispiele** möglich
- **Schneller Einstieg** mittels Python möglich
- **Sprachwechsel** am Anfang sollte unterlassen werden
- **Objektorientierte Konzepte** sollten nicht Teil des Anfangsunterrichts sein
- **Unterschiedliche Fachbereiche** benötigen unterschiedliche Programmiersprachen
- Die Möglichkeit **interaktiv zu programmieren** wird neutral oder positiv gesehen
- **Entwicklungsumgebungen** bei Python noch nicht so ausgereift wie unter Java
- **Multiplattform-Unterstützung** ist hilfreich

Auf die meisten dieser Themen wollen wir später noch weiter eingehen. Der letztgenannte Punkt der Multiplattform-Unterstützung wurde nur von Lehrer 2 (siehe Tabelle 5.4) explizit als wichtig angesprochen. Der ähnliche Punkt der Verbreitung einer Programmiersprache wurde hingegen von mehreren Lehrern diskutiert. Die Meinungen ob eine Anfängersprache unbedingt auch in der Industrie starke Verwendung finden sollte sind

aber unterschiedlich. Gerade deswegen listen wir diesen Punkt unter den *umstrittenen Themen*:

- Python ist eine/keine **mächtige Programmiersprache**
- Zukünftige Berufsmöglichkeiten (**Verbreitung** der Programmiersprache) sind wichtig/unwichtig
- **Dynamic Typing** stellt einen Vorteil/Nachteil dar
- Der hohe **Abstraktionslevel** von Python ist ein Vorteil/Nachteil
- Die **Syntax** einer Anfängersprache ist egal/wichtig

Nachdem wir einen Überblick über die Themengebiete der Interviews gegeben haben, wollen wir nun auf die einzelnen Punkte im weiteren Verlauf dieses Abschnitts näher eingehen. Zu diesem Zweck werden wir auch die Ergebnisse der Umfrage in die Besprechung der einzelnen Themengebiete einfließen lassen.

Algorithmisches Verständnis Von praktisch allen Lehrpersonen wurde angegeben, dass Programmieranfängern vor allem beim algorithmischen Verständnis Probleme haben. Bei der grundsätzlichen Erstellung eines Programms und dem abstrakten Denkprozess dahinter gibt es Schwierigkeiten. Die Frage ob die Programmiersprache dabei eine Rolle spielt verneinten die meisten Lehrpersonen. Lehrer 9 hingegen meinte das auch bei den Konzepten eine Programmiersprache eine Rolle spielt. Dieser Meinung schließen wir uns an.

Als Grund dafür sehen wir, dass auch schon beim überblicksmäßigen Design eines Programms, im Hinterkopf an die Realisierbarkeit in der jeweiligen Sprache gedacht wird. Gerade eine Sprache wie Python, mit ihrem hohen Abstraktionslevel bietet sich hier an. Zwischen dem Design auf einem hohen Level und der Realisierung in der Programmiersprache wird ein geringerer kognitiver Aufwand benötigt als wie bei Sprachen mit einem niedrigeren Abstraktionslevel wie C.

Teile des Designs können in Python auch schnell ausprobiert werden, ohne aufwändig notwendigen aber nicht problemspezifischen Code, schreiben zu müssen so wie zum Beispiel in Java. Das führt dazu, dass Schüler schnell zumindest einen Teil der Aufgabe lösen. Die sofortige Rückmeldung innerhalb einer interaktiven Python-Programmierumgebung hilft ihnen schnell problembehaftete Teile des Design zu verwerfen und erfolgsversprechende Wege weiter zu gehen. Dies wiederum führt schneller als in anderen Sprachen zu Erfolgserlebnissen.

Selbständiges Lernen, Motivation und schneller Einstieg Die meisten Lehrpersonen erwähnten in den Interviews, dass ein selbstständiges Lernen der Schüler unerlässlich sei. Gerade der Praxis-Teil spielt bei dem Erlernen der Programmierung eine primäre Rolle. Lehrer 8 verwies beispielsweise darauf, dass der Unterricht bei ihm nur mehr aus Programmier-Beispielen besteht. Der Theorie-Teil steht den Studenten dabei nur mehr in Form eines Skriptums zur Verfügung. Der Frontal-Unterricht in Form einer Vorlesung entfällt. Während das sicher ein extremes Beispiel darstellt verweisen auch andere Lehrer darauf, dass sie darauf achten, dass Schüler möglichst schnell selbständig werden und sich beispielsweise per Websuche über Lösungsstrategien informieren.

Damit Schüler selbständig lernen ist sicherlich Motivation ein einflussreicher Faktor. Als Motivationsfaktor wurde von einigen Lehrpersonen (z.B. Lehrer 3 und Lehrer 4) effektvolle Beispiele genannt. Im speziellen wurde hier öfters positiv auf Processing verwiesen einem Toolkit zum visuellen Design, dass sich für Programmier-Anfänger anbietet. Während Processing auf Java aufbaut gibt es unter dem Namen Processing.py auch eine Version von Processing die Python als Programmiersprache verwendet. In Punkto Visualisierung und effektvolle Beispiele kann auch VPython erwähnt werden ein Paket zur 3D-Visualisierung. Dieses wurde vor allem von Lehrer 10 propagiert. Hierbei sei aber zur erwähnen, dass dabei die 3D-Komponente als zusätzliche Schwierigkeit gesehen werden kann. In diese Richtung argumentierte zumindest Lehrer 4.

Als einen weiteren Motivationsfaktor sehen wir den schnellen Einstieg bei Python. Dieser wurde beispielsweise auch von Lehrer 1 erwähnt. Weiters schnitt Python bei diesem Thema auch in der Umfrage mit einer mittleren Wertung von 3.63 wesentlich besser ab als Java. Java bietet laut der Einschätzung der Lehrer mit einer mittleren Wertung von 2.38 nur einen unzureichend schnellen Einstieg.

Im Zusammenhang des leichten Einstiegs ins Programmieren kann auch die von Lehrer 5 und Lehrer 10 erwähnte Taschenrechner-Funktionalität von Python gennant werden. Eine Möglichkeit wäre hier Schülern diese Funktionalität zu zeigen damit sie diese beispielsweise zur Kontrolle ihrer Mathematik-Aufgaben einsetzen. Damit kann Schülern schnell vermittelt werden, dass Python auch für ihren Alltagsgebrauch nützlich sein kann. Diese Möglichkeit existiert bei einer kompilierten Sprache wie Java zum Beispiel praktisch überhaupt nicht.

Sprachwechsel Ein Sprachwechsel am Anfang des Lernprozesses wird von vielen Lehrern als problematisch beschrieben. Speziell Lehrer 2, Lehrer 4 und Lehrer 7 beschrieben diesen Wechsel als negativ. Dieser Meinung schließen wir uns an. Das Erlernen mehrerer Sprachen im weiteren Verlauf des Lebens ist, wie auch von Lehrer 5 gefordert, sicher hilfreich. Vorher sollte aber das schon erwähnte logische und algorithmische Denken mittels einer dazu geeigneten Sprache wie Python trainiert werden. Nachdem dieses ausgereift ist kann der zweite Schritt auf eine neue eventuell spezialisierte Sprache wie beispielsweise C erfolgen.

Objektorientierte Konzepte OOP-Konzepte sollten laut der Einschätzung der befragten Lehrer nicht Teil des Anfangsunterrichts sein. Interessant dabei ist, dass auch Lehrer die Java unterrichten und für diese Sprache plädieren (z.B. Lehrer 7) dieser Meinung sind. Hierzu möchten wir gerne Lehrer 8 zitieren:

> Warum ist es (Python) eine gute Einführungsprogrammiersprache? Weil diese ganzen Advanced-Konzepte kann man weglassen. Und man arbeitet trotzdem relative gut damit. Während [wenn] man hingegen bei Java weglässt, was eine Klasse ist, dann kommt man nicht mal bis zum „Hello World".

Diesen Umstand zusammen mit der schlechten Trennbarkeit von Themengebieten in Java — von Lehrer 8 als „Zirkel von Abhängigkeiten" — bezeichnet zeigt, dass es sich bei Java um keine gute erste Programmiersprache handelt. Im Gegensatz dazu ermöglicht es einem die Multiparadigma-Sprache Python auf objektorientierte Konzepte am Anfang zu verzichten und statt dessen am Anfang eine prozedurale oder funktionale Verfahrensweise einzusetzen. Beim letztgenannten Thema kann hier auch noch Lehrer 9 erwähnt werden, der neben dem imperativen Paradigma auch das funktionale Paradigma als gut für Einsteiger bewertet.

Unterschiedliche Fachbereiche Im Hinblick auf die ideale Einführungssprache wird von vielen der Lehrer darauf verwiesen, dass für unterschiedliche Schulen und Fachgebiete auch unterschiedliche Programmiersprachen zum Einsatz kommen sollten.

Wie schon vorher erwähnt wurden die Sprachen C und C++ von einigen Lehrern gennant wenn es darum geht Schüler in die Programmierung von Micro-Controllern einzuführen. Die Sprache Python wurde in diesem Zusammenhang als vorteilhaft bei den Gebieten Netzwerke (Lehrer 1, Lehrer 4, Lehrer 6), Multimedia, Linux-Scripting (Lehrer 6) und numerische Berechnungen genannt. Auch die Verwendung als „Glue-Language", also zum Verbinden von Programmen, wurde beispielsweise von Lehrer 9 positiv hervorgehoben. Der hohe Abstraktionslevel und damit geringe Overhead von von Python (Lehrer 1, Lehrer 4, Lehrer 5) qualifiziert die Sprache auch für den Bereich der Implementierung von Pseudocode.

Anders wie viele der Lehrer sehen wir auch kein Problem Python als Lehrsprache für Informatiker und im speziellen System-Programmier zu verwenden. Schließlich kann Python gut verwendet werden um zukünftige Programmierer in das algorithmische Denken einzuführen. Nach dem dieses Gebiet erschlossen wurde kann dann in Folge das gewonnene Wissen verwendet werden um die Einstiegs-Hürde in eine Zweitsprache zu verkleinern.

Interaktive Programmierung Die Möglichkeit mittels Interaktion kleine Programmteile in der Python-Shell auszuführen wurde speziell von Lehrer 1 positiv hervorgehoben:

> Was ich wichtig finde beim Programmierunterricht, dass man irgendwie interagiert. Kurze Beispiele hat. Und das es irgendwie ein dynamischer Prozess ist. Und nicht, dass es lange Durchstrecken gibt.

Auch in der Umfrage wurde der interaktive Interpreter als einer der großen Vorteile von Python gennant (siehe Tabelle 5.13). Als Nachteil wurde dieser Funktion nur von einem einzigen Lehrer gesehen, während neun Personen die Funktionalität als Vorteil einschätzen.

Entwicklungsumgebungen Beim Punkt der Entwicklungsumgebungen wurde von Lehrer 4 angeführt, dass Python-IDEs noch nicht so ausgereift sind wie die Entwicklungs-Werkzeuge unter Java. Vor allem die Autovervollständigung unter Python wurde von diesem Lehrer als noch verbesserungswürdig eingeschätzt.

Beim Anfang des Lernens sehen wir den Einsatz eine mächtigen IDE eher als Problem an. Eine Einschätzung die aber auch Lehrer 4 teilt:

> Ein wichtiges Kriterium für mich ist etwas wo der Schüler ein Fenster hat, wo er den Text so hineinschreibt wie er ihn auf der Tafel sieht und dann einen „Run“-Knopf hat. Alles was mehr ist, ist für den Schüler am Anfang zu verwirrend.

Lehrer 7 sprach im Zusammenhang der Entwicklungsumgebung davon, dass unter Python auch ein Text-Editor zur Entwicklung reicht. Er beschreibt den Wechsel auf einen IDE der danach kommt allerdings als einen mit Hürden behafteten Prozess:

> Und die haben wirklich nur im Text-Editor ihr kleines Skript geschrieben und haben ganz ganz kleine Aufgaben gelöst. Und wie die dann auf einmal gesehen haben in C#. Visual Studio… Und es gibt Klassen… Die waren komplett erschlagen von der Mächtigkeit dieser Maschinerie, die da auf sie zukommt.

Lehrer 7 spricht dabei auch von einem „Kultur-Schock“ der Schüler. Den Wechsel auf eine IDE sieht er aber als notwendig an:

> Was ich an einer Skriptsprache nett ist: Das ich sie lernen kann und anwenden kann, ohne das ich ganze große drumherum einer Entwicklungsumgebung brauche. Also ich brauche kein Visual Studio, kein Netbeans, kein Eclipse. Sondern ich kann mit einem Text-Editor schreiben, kann auf der Kommandozeile werken. Das ist zum lernen auf der einer Seite hübsch, *auf der anderen Seite ist es nicht die Realität.*

Bei diesem Punkt möchten wir allerdings widersprechen, da es einerseits durchaus auch professionelle Entwickler gibt die auf den Einsatz einer IDE verzichten. Andererseits können sich die Hilfestellungen die eine IDE bietet durchaus auch negativ auf den Lernprozess auswirken:

> An IDE, or "Integrated Development Environment" will turn you stupid. They are the worst tools if you want to be a good programmer because *they hide what's going on from you*, and your job is to know what's going on.[Sha]

Als Entwicklungsumgebung wurde von Lehrer 8 IPython-Notebook empfohlen, dass mittels Web-Interface eine erweiterte Python-Console mit der Möglichkeit grafischer Visualisierung zur Verfügung stellt. Hierbei wurde von dem Lehrer hervorgehoben, dass Dokumente die gespeichert werden auch die Ergebnisse der Auswertungen der Schüler enthalten. Somit wird es dem Lehrer ermöglicht auch noch nachträglich die Ergebnisse und Probleme der Schüler zu analysieren. Eine Möglichkeit, die gerade bei einer hohen Anzahl von Schülern vorteilhaft ist, da diese möglicherweise nicht alle gleichzeitig während des Unterrichts betreut werden können.

Umstrittene Punkte

Während es bei den vorhergehenden Punkte unter den Lehrern größtenteils einen Konsens gab waren die Meinungen bei den folgenden Themen sehr unterschiedlich.

„Mächtigkeit" von Python Lehrer 7 spricht davon, dass der Einstieg bei einer Skriptsprache wie Python wahrscheinlich leichter fällt, dann aber die Möglichkeiten mit dieser Sprache größere Projekte zu realisieren schnell ausgeschöpft sind:

> Aber ich denke die Linie auf die man dann stoßt, sind bei Skriptsprachen… Da werfe ich jetzt Python mit den anderen Sprachen einfach in einen Topf rein… Die sind dann schnell erreicht… Da ist man dann schnell am Ende der Möglichkeiten.

Lehrer 1 hingegen argumentiert gegenteilig:

> Aber was ich noch sagen wollte, dass man sehr sehr komplexe Software entwickeln kann, auf eine effiziente Art und Weise. Also Python ist nicht eine Mini-Programmiersprache.

Dieser Meinung sind wir auch. Es gibt durchaus auch große Software-Projekte die in Python realisiert wurden. Ein sehr bekanntes Beispiel hierfür ist Dropbox, ein Dienst zur Synchronisierung von Dokumenten mit über 400 Million registrierten Benutzern. Die Client-Software dieses Dienstes besteht fast ausschließlich aus Python-Code [Hof11].

Zukünftige Berufsmöglichkeiten Ob eine Anfängersprache auch in der Industrie stark verbreitet sein soll wird von den Lehrern sehr unterschiedlich gesehen. Während einige der interviewten Personen (Lehrer 6 und Lehrer 7) dieser Einschätzung sind ist es für andere Lehrer ein unwichtiger Punkt (z.B. Lehrer 1 und Lehrer 3).

Lehrer 4 spricht hierbei auch an, dass gerade der Bereich der Programmiersprachen teilweise durchaus schnellen Änderungen ausgesetzt ist. Eine Sprache die heute dominierend ist kann im Zeitraum von fünf Jahren — also dem Dauer der regulären Schulzeit in einer HTL — deutlich an Bedeutung verlieren.

Nichtsdestotrotz ist es auch für die Motivation der Schüler wichtig, dass sie sehen, dass die Programmiersprache die sie lernen auch eingesetzt wird wie Lehrer 9 bemerkt:

> Das ist auch für die Schüler interessant, wenn sie was Reales lernen. Nicht ein Spielzeug… Sie wissen gleich, das ist was Reales, dass sie wirklich verwenden können. Das ist eben kein Spielzeug, sondern etwas echtes. Da sind sie viel motivierter.

Hierbei ist zu bemerken, dass Python, als eine der populären Programmiersprachen (siehe Abbildung 5.7), diese Anforderung erfüllt. Dieser Meinung ist auch Lehrer 4.

Dynamische Typisierung Der Punkt der dynamischen Typisierung ist relativ umstritten. In der Umfrage (siehe Abbildung 5.13) wird die Eigenschaft von Python, dass man keine Typ-Deklaration benötigt, im Durchschnitt als leicht positiv wahrgenommen. Das man keinen Return-Typ angegeben muss wird hingegen als leicht negativ beurteilt.

Auch in den Interviews sind die Angaben relativ unterschiedlich. Einige Lehrer argumentieren, dass die kognitive Last der Schüler verringert wird, wenn sie keine Typen angeben müssen. Andere wiederum denken, dass es hilfreich ist wenn Schüler darüber nachdenken müssen welchen Datentyp sie in einer Variable speichern dürfen. In diesem Zusammenhang sei auch auf die Aussage von Lehrer 4 verwiesen, der auf die Wichtigkeit der Typen in Python hinweist:

> Was wir glaub ich ein bisschen unterschätzt haben, oder ich unterschätzt habe ist, dass es zwar in Python heißt, es gibt dort zwar… Man muss keine Typen hinschreiben. Man braucht Variablen nicht vereinbaren. Das verleitet irgendwie dazu, dass man das gar nicht braucht. Es ist aber so, dass in Python sehr wohl jeder Ausdruck einen Typ hat und der Schüler dann doch verstehen muss, dass es ganze Zahlen gibt, Komma-Zahlen… Das es Strings gibt. Das es Listen gibt.

Wir denken dem Vorschlag in diesem Zitat sollte Folge geleistet werden. Dank der starken Typisierung von Python kann Programmieranfängern auch leicht gezeigt werden welchen Typ ein bestimmtes Objekt hat. Zu diesem Zweck kann der Lehrer beispielsweise eine interaktive Python-Console öffnen und den Schüler mittels der Funktion `type` an Hand von ein paar Beispielen zeigen wie man den Typ bestimmen kann:

```
>>> type(1.1)
<class 'float'>
```

```
>>> type(1)
<class 'int'>
>>> type('1')
<class 'str'>
```

Abstraktionslevel Python hat im Vergleich zu Sprachen wie C einen relativ hohen Abstraktionslevel. Ob das einen Vor- oder Nachteil darstellt sehen die Lehrer unterschiedlich. Lehrer 10 argumentiert hier, dass der hohe Abstraktionslevel den Schülern bei der Erlernen der abstrakten Denkweise hilft.

Lehrer 9 hingegen meint, dass der hohe Abstraktionslevel von Python Schülern am Anfang hilft, später aber dafür sorgt, dass der zweite Schritt in die Programmierung schwieriger wird:

> Wenn man vorhat viel tiefer einzudringen, dann kommt dann eben ein schwieriger Schritt, nachdem man Python ein bisschen gemeistert hat. In anderen Sprachen wie Java, da muss man viel früher in diese schwierigen Konstrukte einsteigen, und damit hat man diese zweite Hürde eben nicht so stark. Man ist früher gezwungen, an diese komplizierten, komplexe Konzepten zu denken.

Während es sicherlich stimmt, dass man in Sprachen wie Java schon früher gezwungen wird in komplexere Konzepte einzusteigen, denken wir das das kein Vorteil ist. Schließlich sorgt gerade das Anhäufen der vielen Konzepte die man als Java-Programmieranfänger benötigt für Frustration unter den Schülern. Wir denken es ist leichter ein komplexes Thema wie die Programmierung in zwei kleineren als in einem großen Schritt zu lernen.

Syntax Die Syntax der Sprache wird von einigen Lehrern als unwichtig angesehen. Lehrer 2 geht hier sogar so weit, dass er die Wahl der Programmiersprache für Anfänger für unwesentlich hält:

> Also meiner Meinung nach, ist es im Prinzip ist vollkommen [egal], mit welcher Programmiersprache man im Unterricht beginnt. Es sollte nur nach Möglichkeit eine sein, die man auf möglichst vielen Plattformen verwenden kann.

Interessant ist hier das Lehrer 2 der in seiner Schule C# unterrichtet später auch darauf eingeht, dass die Schüler die Syntax am Anfang auswendig lernen ohne die Bedeutung (Semantik) von Keywords wie `public` oder `static` zu kennen:

> Wir fangen an, man macht eine eigene [Klasse]… Das ist eine Vorgangsweise… Wir sagen ihnen, so das wird so gemacht. Warum man das so macht, erfahren sie dann irgendwann im zweiten, dritten Jahrgang. Das sie wissen, was sie da machen. Aber jetzt fürs erstes sollen sie [das] einfach nur so hinnehmen, und so machen, dass das so ist.

Gerade hier denken wir das die Syntax, die ja den essentiellen Aufbau der Sprache vorgibt, eine wesentliche Rolle spielt. Wenn man von der überausführlichen Syntax einer Sprache wie Java oder C# hin zu einer Sprache geht, die sich wie Python auf die notwendigen Statements konzentriert, dann kann man den Schülern auch erklären welche Bedeutung die einzelnen Statements besitzen ohne auf eine spätere Erklärung verweisen zu müssen. Das sorgt dafür, dass sich die Schüler sicherer fühlen als wie in einer Sprache bei der sie am Anfang von der Hälfte der Worte nicht wissen welchen Zweck sie erfüllen.

Zusammenfassung

In den zwei vorhergehenden Abschnitten zeigten wir aus welchen Gründen Python als Erstsprache gegenüber klassischen kompilierten Sprachen wie C, C# oder Java zu bevorzugen ist. Besonders die einfache Syntax und der Wegfall des Zwangs gleich am Anfang OOP-Konzepte verwenden zu müssen machen die Sprache ideal für Anfänger. Die interaktive Console erlaubt es dabei Einsteigern schnell über die Verwendung als Taschenrechner spielerisch in die Programmierung einzusteigen.

KAPITEL 6

Schlussfolgerungen

> Now, it's my belief that Python is a lot easier than to teach to students programming and teach them C or C++ or Java at the same time because all the details of the languages are so much harder. Other scripting languages really don't work very well there either. Guido van Rossum

Wie wir am Anfang dieser Arbeit feststellten ist die Programmierung eines der Themengebiete die für viel Frustration unter Schülern sorgt. Sie wird von vielen als übermäßig komplex wahrgenommen. Obwohl Schüler heute größtenteils computer-technisch sehr versiert sind, schlägt die anfängliche Begeisterung für das Programmieren sehr schnell um, wenn Anfänger mit dem gleichzeitigem Erlernen von multiplen Themengebieten überfordert werden.

Im Laufe dieser Arbeit — unter anderem bei der Lehrerbefragung — stellte sich heraus, dass eine der Hürden für eine erfolgreiche Einführung das abstrakte und algorithmische Denken ist. Diese Barriere ist zweifelsohne vorhanden, egal welche Programmiersprache eingesetzt wird. Eine zusätzliche Hürde stellt dann allerdings die Syntax und Semantik der Sprache dar. Diese beiden Gebiete sind essentiell, da Schüler durch sie ihre abstrakten Gedanken in eine für den Computer verständliche Form bringen. Hierbei kann die einfache Syntax von Python helfen den Einstieg in die Programmierung zu erleichtern.

Im Gegensatz zu anderen Programmiersprachen, die häufig als erste Sprache zum Einsatz kommen, entstand Python als Erweiterung einer Sprache die explizit für Anfänger geschaffen wurde. Dieser Punkt wurde auch im Abschnitt „Vergleich aktueller Programmiersprachen“ deutlich. Hier behandelten wir einige der Problem von aktuellen Lehrsprachen wie Java und C, die wir hier nocheinmal kurz zusammenfassen wollen:

- Sprache wurde nicht mit Gedanken an Programmieranfänger geschaffen (C, C#, C++, Java)

- Programmiersprache enthält viele Altlasten (C++, Java).
- Versteifung auf ein Paradigma (Java, C#)
- Abstraktionslevel zu niedrig um schnell effektvolle Programme zu schaffen (C)

Die oben beschriebenen Punkte sorgen auch dafür, dass die genannten Sprache eine Syntax und Semantik besitzen die nicht direkt für Anfänger geeignet ist. Auf Python hingegen trifft keine dieser Punkte zu. Hierbei sei besonders darauf verwiesen, dass Altlasten im Gegensatz zu den anderen Sprachen aktiv behandelt werden. Beispielsweise wurde in Release 3.0 der Sprache einige Konzepte vereinheitlicht und logischer gestaltet.

Neben dem abstrakten und logischen Denken stellten wir auch fest, dass das aktive und selbstständige Üben der Schüler eine wichtige Rolle spielt. Auch hier behandelten wir Punkte wie Python hilft die Motivation zu Üben steigert. Gerade der leichte Einstieg mittels des Python-Interpreters hilft Schülern hier ohne viel Aufwand schnell Code zu schreiben. Der Einsatz der interaktiven Konsole als Taschenrechner mindert nicht nur die Einstiegshürde sondern zeigt den Schülern auch gleich, dass sie Python auch in der Praxis, beispielsweise zur Lösung von Mathematik-Aufgaben verwenden können. Python bietet sich hier als ideale Sprache für die Anwendung des Konstruktionismus an, da Schüler mit ihr schnell ein Artefakt (Computerprogramm) konstruieren können. Die Erweiterung des Artefakts und somit der nächste Schritt im Lernprozess kann ebenso schnell durch das Ausprobieren in der interaktiven Konsole erfolgen. Durch die direkte Rückmeldung in der Konsole stellt sich sofort ein Lerneffekt beim Schüler ein.

Die Erfahrungen aus dieser Arbeit, die in den vorhergehenden Zeilen nochmals zusammengefasst wurden, zeigen uns, dass Python eine wesentlich bessere Sprache für Programmieranfänger ist als viele der aktuell gelehrten Sprachen. Gerade die niedrigere Einstiegshürde sorgt dafür, dass die Probleme des aktuellen Unterrichts gemindert werden und die Motivation der Schüler, sich dem wichtigen Gebiet der Programmierung anzunehmen, steigt.

Glossar

CPython CPython stellt die Standard-Implementierung der Programmiersprache Python dar. Diese Implementierung ist in der Programmiersprache C geschrieben und stellt für viele verschiedenen Plattformen zur Verfügung. 32

Linting Linting bezeichnet den Prozess der Analyse von Quellcode auf mögliche Fehler. Dieser Prozess wird üblicherweise automatisch von einem sogenannten Linter bewerkstelligt. 33

Akronyme

BDFL Benevolent Dictator for Life. 7

GUI Graphical User Interface. 10, 33

HTL Höhere Technische Lehranstalt. 84, 98

IDE Integrated Development Environment. 32, 96

NSA National Security Agency. 9

OOP Objektorientierte Programmierung. 20, 26, 37, 38, 41, 79, 87, 95, 100

Programme

Literatur

[Bat02] Ned Batchelder. *Python: batteries included*. März 2002. URL: http://nedbatchelder.com/blog/200203/python_batteries_included.html (besucht am 13. 03. 2002).

[Bea09] David M Beazley. *Python essential reference*. Addison-Wesley Professional, 2009.

[BFR13] Joachim Birk, Steffen Friedrich und Holger Rohland. „Python im Schulinformatikunterricht“. In: (Juli 2013).

[BM07] Matthew Butler und Michael Morgan. „Learning challenges faced by novice programming students studying high level and low feedback concepts“. In: *Proceedings of the 24th ascilite Conference*. 2007, S. 2–5.

[Bod13a] Paul Boddie. *OrganizationsUsingPython - Python Wiki*. Juli 2013. URL: https://wiki.python.org/moin/OrganizationsUsingPython#Education (besucht am 24. 09. 2015).

[Bod13b] Paul Boddie. *Python Games*. Sep. 2013. URL: https://wiki.python.org/moin/PythonGames (besucht am 24. 09. 2015).

[Bod15] Jan Bodnar. *Introduction to Tkinter*. 2015. URL: http://zetcode.com/gui/tkinter/introduction/.

[BV15] TIOBE Software BV. *TIOBE Index for June 2015*. Juni 2015. URL: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

[Cho14] Tracy Chou. *Which programming languages were used to develop Pinterest's backend?* Juni 2014. URL: https://www.quora.com/Which-programming-languages-were-used-to-develop-Pinterests-backend (besucht am 24. 09. 2015).

[Chu+13] R Chudoba u. a. „Using Python for scientific computing: Efficient and flexible evaluation of the statistical characteristics of functions with multivariate random inputs“. In: *Computer Physics Communications* 184.2 (2013), S. 414–427.

[Dev10] Digi Developer. *Python Garbage Collection*. Dez. 2010. URL: http://www.digi.com/wiki/developer/index.php/Python_Garbage_Collection.

[Dij68] Edsger W Dijkstra. „Letters to the editor: go to statement considered harmful“. In: *Communications of the ACM* 11.3 (1968), S. 147–148.

[Fel+15] Matthias Felleisen u. a. *Courses using Racket*. 2015. URL: https://github.com/plt/racket/wiki/Courses-using-Racket.

[Fel09] Wolfgang Fellger. *Schlangenbeschwörung für Anfänger — Eine Einführung in Python*. Dez. 2009. URL: http://wolfgangfellger.de/pythonvortrag/folien.pdf.

[Fou15a] Python Software Foundation. *Built-in Functions*. Mai 2015. URL: https://docs.python.org/library/functions.html.

[Fou15b] Python Software Foundation. *Extending and Embedding the Python Interpreter*. Sep. 2015. URL: https://docs.python.org/3/extending.

[Fou15c] Python Software Foundation. *The Python Tutorial » Classes*. März 2015. URL: https://docs.python.org/2/tutorial/classes.html (besucht am 09. 09. 2015).

[Fre08] Alvar Freude. *Perl, PHP, Python – Ein Vergleich*. Apr. 2008. (Besucht am 01. 09. 2015).

[Fri14a] Nils Friedel. *Lerntheorien 2.0 - Folge 2 - Behaviorismus*. März 2014. URL: https://www.youtube.com/watch?v=Ji8mUhBGrRw (besucht am 21. 07. 2015).

[Fri14b] Nils Friedel. *Lerntheorien 2.0 - Folge 3 - Kognitivismus*. März 2014. URL: https://www.youtube.com/watch?v=wYVOmAf4-0E (besucht am 21. 07. 2015).

[Fri14c] Nils Friedel. *Lerntheorien 2.0 - Folge 4 - Konstruktivismus*. März 2014. URL: https://www.youtube.com/watch?v=QCOW0bSgD4g (besucht am 21. 07. 2015).

[Gmb14] CHIP Digital GmbH. *Komodo Edit*. 2014. URL: http://www.chip.de/downloads/Komodo-Edit_29950542.html.

[Gra03] Martin Grabmüller. *Multiparadigmen-Programmiersprachen*. Techn. Ber. Technischer Bericht, Technische Universität Berlin, 2003. URL: http://cs.tu-berlin.de/cs/ifb/Ahmed/RoteReihe/2003/TR2003_15.pdf.

[Hof11] Todd Hoff. *6 Lessons From Dropbox – One Million Files Saved Every 15 Minutes*. März 2011. URL: http://highscalability.com/blog/2011/3/14/6-lessons-from-dropbox-one-million-files-saved-every-15-minu.html.

[HSJ13] Rick Hoving, Gabriel Slot und Slinger Jansen. „Python: Characteristics identification of a free open source software ecosystem“. In: *Digital Ecosystems and Technologies (DEST), 2013 7th IEEE International Conference on*. IEEE. 2013, S. 13–18.

[Hüb14] Tobias Hübner. *Unterrichten mit dem Raspberry Pi*. Aug. 2014. URL: http://www.medienistik.de/Themenheft_RaspberryPi.pdf.

[Hum02] Ludger Humbert. „Welche Programmiersprache unterstützt meine Konzepte für den Informatikunterricht?“ In: *Universität Dortmund-07* (2002).

[Jun+10a] Josh Juneau u. a. *Jython and Java Integration*. 2010. URL: http://www.jython.org/jythonbook/en/1.0/JythonAndJavaIntegration.html (besucht am 27. 09. 2015).

[Jun+10b] Josh Juneau u. a. *Using Jython in an IDE*. 2010. URL: http://www.jython.org/jythonbook/en/1.0/JythonIDE.html (besucht am 27. 09. 2015).

[KE07] Peter Kaiser und Johannes Ernesti. *Docstrings*. 2007. URL: http://python.haas.homelinux.net/python_kapitel_13_003.htm#mj4f43dabfe129e9471aaf49d6abe8d0cc.

[Kee15] Christopher Keefer. *Write Once, Debug Everywhere*. Feb. 2015. URL: http://www.artandlogic.com/blog/2015/02/write-once-debug-everywhere/ (besucht am 02. 02. 2015).

[Kle15] Bernd Klein. *Python-Kurs – Funktionen*. 2009 – 2015. URL: http://www.python-kurs.eu/funktionen.php (besucht am 09. 09. 2015).

[LAG07] Mark Lutz, David Ascher und Dinu C Gherman. *Einführung in Python*. O'Reilly Germany, 2007.

[Lin02] Ingo Linkweiler. „Eignet sich die Skriptsprache Python für schnelle Entwicklungen im Softwareentwicklungsprozess“. Diss. Diploma Thesis at the University of Dortmund, Germany, 2002. URL: http://www.ingo-linkweiler.de/diplom/INGO_bomholz.pdf.

[LR12] Konrad Lischka und Ole Reißmann. *Diese Kaffeemaschine kocht aufs Wort*. Dez. 2012. URL: http://www.spiegel.de/netzwelt/web/raspberry-pi-die-besten-projekte-fuer-den-46-euro-pc-a-874993-3.html.

[Lut14] Mark Lutz. *Python pocket reference*. Ö'Reilly Media, Inc.”, 2014.

[Mak15] Matt Makai. *Development Environments*. 2015. URL: http://www.fullstackpython.com/development-environments.html.

[Mar15] Michael Markert. *Das Python 3.3 - Tutorial auf Deutsch*. Apr. 2015. URL: https://media.readthedocs.org/pdf/py-tutorial-de/python-3.3/py-tutorial-de.pdf.

[Mei06] Susanne Meier. „Didaktischer Hintergrund Lerntheorien“. In: (2006). URL: http://lehrerfortbildung-bw.de/moodle-info/schule/einfuehrung/material/2_meir_9-19.pdf.

[MSL15] Craig Miller, Amber Settle und John Lalor. „Learning Object-Oriented Programming in Python: Towards an Inventory of Difficulties and Testing Pitfalls“. In: (2015).

[Obe03] Jürgen Obermeyer. „BlueJ-eine didaktische Java-Entwicklungsumgebung“. In: *Grundfragen Multimedialer Lehre* (2003), S. 174.

[Oka99] Chris Okasaki. *Purely functional data structures*. Cambridge University Press, 1999. URL: http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf.

[Old11] Reinhard Oldenburg. *Mathematische Algorithmen im Unterricht*. Springer, 2011.

[Pah15] Dieter H. Pahr. *Grundlagen des Programmierens VU*. Apr. 2015. URL: https://www.ilsb.tuwien.ac.at/~pahr/317.530/index.html.

[Pel08] Tomas Pelka. *Comparison of Python virtual machines*. Nov. 2008. URL: http://polishlinux.org/apps/cli/comparison-of-python-virtual-machines/ (besucht am 07. 11. 2008).

[Pos06] Stefan Posch-Gruber. *Das Unterrichtsfach Informatik im Kontext informatischer Bildung*. na, 2006.

[pro15] programcreek.com. *Java vs. Python (2): Data Types*. 2008 – 2015. URL: http://www.programcreek.com/2012/09/java-vs-python-data-types/ (besucht am 09. 09. 2015).

[Ram] Stefan Ram. *C++ in der Schule*. URL: http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/c++_lernsprache_de.

[RD11] GV Rossum und F. L. Drake Jr. *Python reference manual*. iUniverse. 2011.

[RD95] Guido van Rossum und Fred L Drake. *Extending and embedding the Python interpreter*. Centrum voor Wiskunde en Informatica, 1995.

[RNH06] Raimond Reichert, Jürg Nievergelt und Werner Hartmann. *Programmieren mit Kara: ein spielerischer Zugang zur Informatik*. Springer-Verlag, 2006.

[Ros99] Guido van Rossum. *Das Python-Tutorium – Klassen*. Apr. 1999. URL: http://python.net/~gherman/publications/tut-de/online/tut/node11.html (besucht am 09. 09. 2015).

[Sch00] Neil Schemenauer. *Garbage Collection for Python*. Dez. 2000. URL: http://arctrix.com/nas/python/gc/.

[Sch14] Thorsten Schröder. *US-Universitäten: Python ist beliebteste Programmiersprache bei Einsteigern*. Juli 2014. URL: http://www.golem.de/news/us-universitaeten-python-ist-beliebteste-einstiegs-programmiersprache-1407-107748.html (besucht am 24. 09. 2015).

[Sch15] Harald Schilly. *PR Programmierpraktikum, 250129, SoSe 2015*. 2015. URL: http://harald.schil.ly/previous/pr-pp-15 (besucht am 10. 09. 2015).

[Sci09] Institute For Mathematics & Computer Science. *Schools using scheme*. 2009. URL: http://www.eimacs.com/schemers.htm.

[SH15] Alexander Schliep und Winfried Hochstättler. *Gato – Graph Animation Toolbox*. 2015. URL: http://gato.sourceforge.net/.

[Sha] Zed Shaw. *Learn C the Hard Way – Exercise 0: The Setup*. URL: http://c.learncodethehardway.org/book/ex0.html.

[She] Bruce Sherwood. *What is VPython?* URL: http://vpython.org/contents/overview.html (besucht am 10. 09. 2015).

[Shi15] Pete Shinners. *Pygame*. 2015. URL: http://www.pygame.org/hifi.html (besucht am 10. 09. 2015).

[SLY13] Pranay Kumar Sevella, Young Lee und Jeong Yang. „Determining the Barriers Faced by Novice Programmers“. Diss. Texas A&M University-Kingsville, 2013.

[SS11] Sigrid Schubert und Andreas Schwill. *Didaktik der Informatik*. Springer, 2011.

[Stü12] Moritz Stückler. *Raspberry Pi (fast) im Weltraum*. Juli 2012. URL: http://t3n.de/news/raspberry-pi-fast-weltraum-402656 (besucht am 10. 09. 2015).

[Stü13] Moritz Stückler. *Raspbmc: So baust du einen Mediaplayer für 50 Euro mit Raspberry Pi und XBMC*. Apr. 2013. URL: http://t3n.de/news/raspbmc-baust-mediaplayer-50-452902/ (besucht am 10. 09. 2015).

[Süd12] A. Südkamp. „Didaktik der Naturwissenschaften“. In: *Zeitschrift für Erziehungswissenschaft* 15 (2012), S. 175–179.

[TM12] Ioana Tuugalei und Chan Mow. „Analyses of Student Programming Errors in Java Programming Courses“. In: *Journal of Emerging Trends in Computing and Information Sciences* 3.5 (2012), S. 739–749. URL: http://www.cisjournal.org/journalofcomputing/archive/vol3no5/vol3no5_11.pdf.

[VT08] Milena Vujoševic-Janicic und Dušan Tošic. „The role of programming paradigms in the first programming courses“. In: *The Teaching of Mathematics, XI* 2 (2008), S. 63–83.

[Wik15a] Wikipedia. *CPython — Wikipedia, The Free Encyclopedia*. März 2015. URL: https://en.wikipedia.org/wiki/CPython#Supported_platforms.5B2.5D (besucht am 04. 03. 2015).

[Wik15b] Wikipedia. *History of Python — Wikipedia, The Free Encyclopedia*. Aug. 2015. URL: https://en.wikipedia.org/wiki/History_of_Python (besucht am 05. 08. 2015).

[Wik15c] Wikipedia. *Programmierparadigma — Wikipedia, Die Freie Enzyklopädie*. Juni 2015. URL: http://de.wikipedia.org/wiki/Programmierparadigma (besucht am 27. 06. 2015).

[Wik15d] Wikipedia. *Python (Programmiersprache) - Entwicklungsumgebung — Wikipedia, Die Freie Enzyklopädie*. Mai 2015. URL: http://de.wikipedia.org/wiki/Python_(Programmiersprache)#Entwicklungsumgebung.

Interviews

Da die komplette Transkription der Interviews den Umfang der Arbeit – selbst bei der Verwendung einer kleinen Schriftart – wesentlich erhöhen würde, werden die Interviews hier nur in verkürzter Form wieder gegeben. Dabei wurde darauf geachtet, dass die wichtigsten Punkte der Interviews trotzdem in der Transkription aufscheinen.

Lehrer 1

Lehrer 1 …10 000 Stunden muss man irgendwas damit gemacht haben, damit man wirklich ziemlich gut damit umgehen kann. Ich weiß nicht ob sie den kennen…

Yazicioglu [Wen]?

Lehrer 1 Malcolm Gladwell

Yazicioglu Ah, nein

Lehrer 1 Aber ja, 10 000 Stunden, dann kann man das sehr gut. Und die haben, ich würde sagen, in der Größenordnung… Also das sind alles keine Softwareentwickler. Und wenn man als Softwareentwickler arbeiten will, dann hat man eine Leidenschaft für diese Materie. Das ist einem wichtig. Und Softwareentwicklung hat ziemlich was mit kreativer und künstlerischer Gestaltung… Das machen die aber alle nicht. Und die sind alle Leute die irgendwie ihren… Ich meine halt wirklich sehr bemüht… Aber [sie] versuchen eben einen Stoff zu machen, der ihnen komplett fremd ist. Und dadurch erzählen sie dann irgendeinen Blödsinn. Es is wirklich haarsträubend. Ich kann ihnen das nur kurz zeigen, wie ich in Python anfangen würde.

Yazicioglu Sehr nett. Ja, Ja…

Lehrer 1 Und dann würde ich sie fragen: Programmieren ob sie das irgendwie kennen… Welche Vorstellung sie davon haben. Was sie glauben wie lange das dauert bis sie das lernen, bis sie so ein Programm machen können bis sie… Ich weiß nicht… Eine Funktion schreibt, die addiert oder sowas. Und dann tun sie ein bisschen herum, überlegen, und sind ein bisschen aufmerksam. Und dann zeige ich… Schaut, dass könnt ihr in fünf Minuten programmieren. Und dann sage ich schaut, dass macht ihr so… Das ist dann am Beamer… Okay, Python… Und jetzt gibt es unterschiedliche Variablen, z.B. machen wir mal eine `a=2`, eine `b=3`. Das kapieren [sie] sofort. Und dann… Jetzt werden die addiert. Ja einfach so wie man es intuitiv macht: `a+b`. Gut… Jetzt gibt es grundsätzlich auch Funktionen… Also da würde ich dann ein bisschen… Das ist dann eher so intuitiv… Was mir einfällt… Ob sie aufmerksam sind… Dann würde ich vielleicht ein wenig Datentypen auch zeigen. Und dann kommt es aber in dem Laufe der Stunde, zur ersten Funktion. Zum Beispiel, dass ich addiere. Eine Funktion, die einfach nur zwei Zahlen addiert. Dann würde ich einfach sagen, dass machen wir ähnlich wie bei der Variablenzuweisungen… Kennen sie Python?

Yazicioglu Ja, seit zwei Monaten.

Lehrer 1 Okay und diese Lambda-Funktionen finde ich wirklich sehr nett, für so ganz simple Funktionen. Und da haben wir jetzt… Gescheiter Name… Okay… Und was soll die liefern: `a+b`. Machen wir das einmal. Und wenn ich jetzt sage `add(3.4, 5)` dann liefert sie zurück. Also so schnell und einfach kann ich eine Funktion schreiben. Dann würde ich das halt ein bisschen erklären wie das mit den Variablen… Und das das halt zurückgegeben wird. Das ist es das aber dann eigentlich. Also auf Grund… Ein bisschen ausbauen… Mit den Datentypen… Vielleicht ein bisschen komplexeren Funktionen… Ziemlich gut sind da diese Definitionslisten. Also wenn ich zum Beispiel die Summe aller Quadrate haben wollte. Also es gibt ja diese `sum`-Funktion. Wenn ich irgendetwas habe: Eine Liste [dann] wird das aufsummiert. Wenn ich die Quadrate aufsummieren will, dann würde ich einfach schreiben `sumq = lambda i:`… Und dann nehme ich die `sum`-Funktion die es eh schon gibt und definiere eine Liste…

(Arbeitet an Beispiel)

Und man sieht wie irrsinnig schnell das geht. Wie interaktiv das ist… Wenn ich mit den Schülern spreche… Ich kann schauen wie sie darauf reagieren. Sie müssen durch keine Durststrecken. Sie können von der ersten Sekunde an programmieren und ganz einfache Sachen machen. Ich hab dann…

(Diskussion über nicht relevantes Thema)

Lehrer 1 Was ich wichtig finde beim Programmierunterricht, dass man irgendwie interagiert. Kurze Beispiele hat. Und das es irgendwie ein dynamische Prozess ist. Und nicht, dass es lange Durchstrecken gibt. Und wenn ich jetzt anfangen würde…

Yazicioglu Das finden sie didaktisch… Didaktisch sollte es so sein.

Lehrer 1 Didaktisch sollte es möglichst kurz sein. Wenn sie viel Werkzeuge in der Hand haben… Das sie viel machen können… Dann ist das großartig. Weil dann können sie etwas tun. Wenn es aber theoretische Konzepte gibt, die man lang erklärt bevor sie anfangen irgendwas zum programmieren oder zu machen, oder [eine] Funktion zu schreiben ist nicht angenehm.

Wenn die sowas machen können… Also nach einer Stunde sind sie in der Lage, einfache Funktionen zu schreiben, dann ist das super. Wenn ich aber zwei Einheiten brauche, um einfach. Weiß nicht, drei, vier Stunden mit der Entwicklungsumgebung habe. Dann erkläre ich noch das Klassen-Framework weiter. Dann vergeht ein Monat… Und dann sind sie alle Zombies, die das überhaupt nicht interessiert.

Yazicioglu Sie haben das Interesse verloren…

Lehrer 1 Dann haben sie das Interesse verloren. Und deshalb finde ich das wirklich ideal. Ich finde das [ist] wirklich die ideale Art und Weise wie man das macht. Ich habe keine besondere Präferenzen für irgendeine Programmiersprachen von vorne herein. Weil ich einfach schon sehr sehr viele Programmiersprache gemacht habe. Und lange in Software-Firmen in unterschiedlichsten Programmiersprachen Dinge entwickelt habe. Also mir wäre das auch komplett [egal] C# zu machen. Oder ich hab mit Kollegen gesprochen, ob wir uns nicht einigen können auf Java. Das wir Java machen, dass das zumindest ein wenig ein offenes System ist.

Aber die haben alles das Problem, dass sie eine einziges Werkzeug kennen. Das haben Sie sich lange angeeignet und da können sie nicht so leicht weg. Das ist eher das Problem. Das alles wird rationalisiert. Dann gibt es irgendwelche total blöden Argumente.

Was sie zum Beispiel sagen… Ich hab das probiert und bin daran gescheitert. Ich habe mit einem Kollegen gemeinsam unterrichtet und habe die Hälfte gehabt und meine Kollegen hat andere Hälfte gehabt. Dann haben wie… Aus ein paar Gründen hat es da wirklich Probleme gegeben. Also wir haben auch mit der Klasse Probleme gehabt.

Am Schluss vom Jahr… Mit der Methode habe ich gearbeitet… Und die waren wirklich bemüht und sehr fleißig. Aber am Schluss vom Jahr habe ich bemerkt, die können nicht programmieren. Sie können es nicht. Also eigentlich — obwohl sie sich bemüht haben — sind sie nicht in der Lage algorithmisch zu denken.

Denn habe ich mit meinem Kollegen geredet. Er hat aber gemeint: Nein, nein, das ist super gegangen. Also bei ihm hat alles wunderbar funktioniert. Dann habe ich mir gedacht: Okay, also anscheinend ist es so, dass ich Probleme gehabt das zu machen. Obwohl ich mich echt bemüht habe, echt darauf eingegangen bin und auch schon lange das mache als Lehrer. Aber beim Kollegen ist es gegangen. Warum auch immer…

Nur habe ich die Klasse in zwei Jahren wieder gehabt, und habe bemerkt, dass sie nicht programmieren können. Also das das einfach nicht stimmt. Und überhaupt so wie man HTL Schüler hat… Man nimmt den Besten, den Schlechtesten und den Mittleren heraus. Der kann nicht programmieren. Also bei uns… Ich weiß nicht ob das allgemein so ist. Also ist bei uns der Programmierunterricht so, dass die Schüler nicht programmieren können. Ich rede jetzt nicht von denen, die sehr engagiert sind, sondern der mittlere Schüler kann nicht programmieren. So ist es.

Ja, und insofern sind diese Behauptungen sind einfach komplett unsinnig. Ja, also in den einen Gegenständen… Also in Python können sie es nicht, aber in anderen Sprache können sie… Das stimmt einfach nicht. Und die Kollegen können alle nicht programmieren. Das kommt auch dazu. Weil wenn du programmierst, dann merkst du das es Spaß macht. Spaß macht das zu lösen, das Künstlerische, das Schöne. Es gibt einfach extrem schöne Lösungen.

Wenn du da vorne sitzt und kein Gefühl für schöne Lösungen und das Kreative und stolz bist auf die Lösungen bist, die du findest. Dann kriegen die Schüler das ja auch schon gar nicht mit. Das ist ja schon eine komplette Art… Komplett die falsche Art, das Problem anzugehen. Also das ist… Mir kommt vor das ist wie beim Malen. Also beim Malen, da ist dieses… Du malst ein Gemälde und freust dich, dass du das Gemälde machst und dieses schaffst. Und wenn du jetzt Leute vorne hast, die wissen wie man Grün und Blau zusammen mischt und was dann rauskommt. Was soll das sein?

Das ist das Problem, dass man überhaupt hat. Und das ist das Problem mit den Kollegen. Und deswegen behaupten die auch irgendwas. Ich meine die sind eh alle nett. Sie sind auch bemühte Lehrer, aber die verstehen davon nichts. Also es ist so. Und es kommt bei den Schülern nicht an. Und die Schüler können nicht programmieren.

(Diskussion über nicht relevantes Thema)

Yazicioglu Sie haben probiert. Das habe ich nicht so genau verstanden…

Lehrer 1 Na, es ist so. Wir haben beide [in] der ersten Klasse unterrichtet.

Yazicioglu Parallel, gleichzeitig?

Lehrer 1 Parallel, gleichzeitig… Am Schluss war ich mit dem Ergebnis nicht zufrieden. Ich habe dann zu ihm gesagt: Du ich glaube sie haben es nicht gelernt in dem Jahr. Ich hab gesagt: Resultat meiner Gruppe… Sie haben es nicht gelernt. Er hat gesagt… Er hat behauptet, seine Gruppe hat es gelernt.

Yazicioglu Behauptet?

Lehrer 1 Behauptet… Nur habe ich dann selber gesehen zwei Jahre später, dass das nicht stimmt diese Behauptung. Da waren sie schon zwei Jahre älter… Die können nicht programmieren. Das stimmt einfach nicht. Und es ist aber auch so, dass der Kollege auch nicht programmieren kann. Und zwar ganz einfach deshalb… Nur, also…

Da gibts es auch von Malcolm Gladwell ein gutes Buch. Das heißt Blink. Und da geht es darum, dass man innerhalb von Sekunden erkennen kann, ob irgendjemand… Irgendwas stimmt oder… Also bestimmte Fertigkeiten. Der hat zum Beispiel gesagt, dass es einen Mann gibt, der sieht ein Ehepaar und kann innerhalb von Sekunden sagen, ob sie geschiedene sind, ein paar Jahre oder nicht. Und dann hat man das analysiert warum… Und gesehen, es gibt bestimmte Sachen.

So, jetzt gibt es bei Softwareentwicklern bestimmte Sachen, die man sofort erkennt. Wie schnell tippt der? Also wenn jemand langsam tippt, dann ist er kein Softwareentwickler. Das ist… ich weiß nicht ob sich das jetzt geändert hat, mit diesen ganzen „Klick“-Sachen usw. Das kann ich nicht sagen. Vielleicht gibt es inzwischen welche, die nicht mehr schnell tippen können. Aber ich habe lange in Softwarefirmen gearbeitet. Die haben alle getippt, wie die Blöden. Also ganz schnell. Wenn da einer langsam war… Der war sicher kein Softwareentwickler. Und ich hab noch nie einen Softwareentwickler gesehen, der langsam tipp. Also einen guten… Es hat dann schon ein paar gegeben, aber…

Und beim Kollegen ist es so. Der macht jetzt folgendes. Wenn er zum Beispiel sagt: `if (a + 3) > 5` klammert er `a+3` ein zur Sicherheit. Aber du kennst als Softwareentwickler automatisch die Präzedenzen von den Operatoren. Das ist überhaupt kein Thema. Du kommst einfach überhaupt nicht auf die Idee. Und alleine an dieser Klammer sieht man, dass der das nicht machen kann. Und ich kenn auch seine Biografie usw. Also es ist… Er kann nicht programmieren. Er kennt diese Leidenschaft nicht dafür. Er kennt das Künstlerische nicht.

Jetzt sind diese ganzen Behauptungen unbrauchbar. Und wenn sie zu den Lehrern gehen, dann hat man das Problem, dass sie niemals Softwareentwicklern gegenübersitzen. Die das unterrichten… Wie man halt leidenschaftslos als beamteter Lehrer irgendeinen Stoff, der in einem Buch drinnen steht unterrichtet. Das ist komplett die falsche Art und Weise. Das ist… Also deshalb.

(Diskussion über nicht relevantes Thema)

Yazicioglu Finden sie dieses Argument, dass man für die Arbeitswelt, für die Industrie eine verbreitete Sprache lernen soll. Finden sie das gut…

Lehrer 1 Das finde ich…

Yazicioglu Soll man nur aus diesen Grund…

Lehrer 1 Nein, … Also da gibt es ein gutes Beispiel von einer Fachhochschule St. Pölten, die haben sich irgendwann auf OS2 sozialisiert, weil sie das Gefühl hatten, das kommt. Und das ist einfach sinnvoll, dass sie auf Industrieprodukten dann arbeiten und darauf hin-trainieren. Die haben dann halt alle keinen Job bekommen, weil OS2 nicht erfolgreich war.

Mit C# wenn man sich darauf trainiert. Ich mein das kann schon sein das es funktioniert. Aber die Firmen sind ziemlich schnell wieder weg. Das ganze C#-Konzept… Also wenn Microsoft es nicht gelingt, was ihnen bisher nicht gelungen ist, sich am Mobil-Markt mehr zu verbreiten… Dann weiß ich nicht ganz genau welche Zukunft die haben.

Und ich finde, was man vermitteln sollte, sind universelle Fertigkeiten. Die man in 30 Jahren noch hat. Und wenn man leidenschaftlich programmiert und das richtige Konzepte und die richtige Art und Weiße wie man das angeht… Dann sind die Sprache vollkommen [egal]. Das spielt überhaupt keine Rolle.

Yazicioglu Wenn man eine Sprache beherrscht, dann kann man [bei den] Anderen alles [machen]…

Lehrer 1 Oder das Denken, wenn man das machen kann. Also deshalb finde ich das absolut zweitrangig das man sich auf ein bestimmtes Industriesegment spezialisiert. Ich meine wie dynamisch das Ganze ist… Was für einen Sinn könnte das machen. Also ich mache mit ihnen nur Linux und Python und ein bisschen Bash-Scripting.

Die Sachen, die ich verwende, die sind die Sachen, die habe ich gelernt in den 80-ern. Also ziemlich unverändert. Wo gibt es das? Das über 30 Jahre sich nichts ändert. Weil es einfach ein perfektes System ist. Und das bleibt ziemlich klar und statisch. Und ich weiss nicht, was hat Microsoft vor 10 Jahre gemacht oder vor 20 Jahre, geschweige denn vor 30. Also das…

Yazicioglu Das ändert sich ganz langsam.

Lehrer 1 Nein… Also es geht wirklich nur um Fertigkeiten und Emotionen. Und es… Deswegen finde ich auch diesen Malcolm Gladwell wirklich sehr gut. Mit diesen 10 000 Stunden. Denn wie kommt ein Schüler dazu, das er 10 000 Stunden etwas macht. Indem ihm das ein Anliegen und eine Leidenschaft ist. Und… Wie schafft man, dass das er die Leidenschaft hat für das Ganze. Also das kann man nur versuchen irgendwie zu vermitteln… Und das man ein Industrie-Produkt macht weil… Ich weiß nicht… Das ist keine Motivation. Es muss ja lustvolles…

Yazicioglu Manche sagen, es kommt auf den Schüler an. Entweder… Manche Schüler wollen wirklich [ein] Softwareentwickler sein… Manche haben das Talent, die Fertigkeit. Manche nicht… Wenn man sich auf diese Schüler konzentriert, weil die nicht die Fertigkeit [besitzen]… Nur für die Schüler die leichte Sprache zu wählen… Das bringt nichts.

Lehrer 1 Erstens stimmt es nicht und zweitens der… Warte ich kann ihnen das auch zeigen…

Yazicioglu Oder eine falsche Sicherheit… Eine… Eine gefährliche Sicherheit… Die Schüler denken, dass am Anfang sehr leicht ist… Das kann ich programmieren. Aber später, wenn sie mit schweren Programmierbeispielen konfrontiert werden, dann beginnen die Probleme… Sorry

Lehrer 1 Nein, nein, das passt eh alles sehr gut…

Weil ich tu echt viele Programme schreiben immer…

…Warum jetzt Python. Als nächstes Argument, warum es gescheit ist… Jetzt habe ich eine Funktion geschrieben, die überprüfen soll ob… Genau die eine bestimmte Anzahl von Monaten liefert. Und zwar ab dem jetzigen Zeitpunkt glaube ich… Ich glaube das ist doch nicht gescheit.

Aber was ich noch sagen wollte, dass man sehr sehr komplexe Software entwickeln kann, auf eine effiziente Art und Weise. Also Python ist nicht einen Mini-Programmiersprache… Mit der man dann Dinge tun kann, mit der man dann… Sie könnten es eh dann ein wenig recherchieren… Ja, also welche große Software-Projekte in Python entwickelt worden sind.

Also wenn man sich zum Beispiel nur anschaut… Jetzt suche ich alle Python-Programme, die auf meinem Betriebssystem-Basis-Image drauf sind. Also das sind keine, die ich geschrieben habe… 6000… Genau… Also ich meine Linux… Ist auch die Frage, was man von Linux als Betriebssystem hält, aber ich denke… Wenn man nämlich auch die amerikanischen… Also Militär usw., [die] das halt irrsinnig gern nehmen. Weil es einfach ein sehr gutes ausgereiftes und stabiles Betriebssystem ist. Und die haben sich dafür entschieden.

Und Linux funktioniert eigentlich sehr evolutionär, weil jeder kann tun was er will… Ist ganz frei und offen. Und die Frage ist Softwareentwickler verwenden die… Was für Programmierplattformen verwenden die um zu arbeiten. Und man sieht in welchem Ausmaß die sich tatsächlich entschieden haben Python zu verwenden. Und das einfach ein sehr großes Ausmaß…

Also es ist jetzt nicht… Also diese Behauptung… Die ist ja auch komplett dilettantisch wieder. Die Behauptung, dass man mit Python… Das das eine Mini-Programmiersprache ist, die mit dem richtigen Leben nichts zu tun hat. Vor allem: Es gibt ja ganz… Wenn man fortgeschritten Python programmiert, da gibt es ja Decorators… Also es gibt da ja ganz sophisticated Konzepte. Man kann funktionale Programmierung machen. Es ist… Von den ganzen Programmierparadigmen ist es einfach extrem vielfältig.

Und man kann auch problemlos `public` und `private`, also wenn man das möchte… Auf diese Art und Weise, kann man genauso Software entwickeln wie man das Gefühl hat, man möchte das jetzt eigentlich tun… In einem halben Jahr mach ich das eh so. Aber am Anfang sicher noch nicht. Aber da hat man einfach die Freiheiten, dass man das tut. Ich wollte ihnen aber noch ein paar Sachen zeigen, warum das eigentlich als Programmiersprache so super ist. Und zwar das erste ist.

Yazicioglu Ich wollte auch fragen…

Lehrer 1 Diese User-Interface machen… Da gibt es eine extrem geniales Modul

Yazicioglu GUI-Erstellung?

Lehrer 1 Nichts [mit] GUI. Sondern das ist viel mächtiger, wenn man [die] Kommandozeile nimmt. Da war immer der Standard, das hat sich gennant unter Linux… Das hat sich genannt `optarg`. Da ist irgendjemand vor gar nicht allzu langer Zeit eingefallen, dass man das noch besser machen kann. Und das heißt `argparse`. Und das ist wieder vollkommen unglaublich was für mächtige Interfaces man damit machen kann. Das bekommt man mit einer GUI nie hin. Durch dieses `argparse`-Modul… Damit ist es jetzt… Alles was ich mache, mache ich ohne GUI. Genau aus dem Grund… Weil man viel mehr reinbringt irgendwie. Jetzt die Projekte die ich da gemacht habe. Das ist leider ein wenig lang her. Muss zuerst überlegen…

Oder sollen wir es überhaupt einmal durchgehen. Vielleicht ist das okay. Das ist nämlich genau wie ich es mit den Schülern machen. Zuerst gibt es mal von Stefan Schwarzer… Das ist irgend so ein Typ, und der hat eigentlich sehr nett… So eine Einführung gemacht in Python… Und da hat er das mit den Datentypen erklärt. Und da mit Operatoren. Und da schon ein bisschen mit Statements… Dann Schleifen und so… Okay, Lambda-Funktionen hat er sogar gemacht.

Und dann hat er so Beispiele dabei: Einlesen und wieder ausgeben. Das ist natürlich überhaupt sehr schön… Da zeigt er wie man das ein wenig aufwendiger macht. Und wird immer schneller und schneller… Also das Aufwendige, was Python bietet, und was man in Java auf diese Art und Weise nicht geht. Da macht er es noch kürzer.

Es ist Text schreiben und lesen in Python viel viel leichter als in Java… In Java muss man… Wie ist das… Stream… Verschachtelte Objekte verwenden, damit man endlich einen Text schreiben oder lesen kann. Aber in Python ist das ganz einfach mit einer Zeile…

Genau… Und was mir jetzt noch… Weil sie das gesagt haben… Dieses Ruby…

Yazicioglu Ruby on Rails?

Lehrer 1 Ja, ja. Also das ist von einem Japaner entwickelt worden. Ich weiß nicht heißt der [Matsumoto]… Jetzt fallt mir der Name nicht ein. Bin mir nicht ganz sicher. Jedenfalls hat der als Entwurfsziel gehabt: „Principle of Least Surprise“. Ja, also wenn man den Code liest, dann soll man das am ehesten kapieren, ohne das da irgendwas seltsames überraschendes ist. Naja das war das Entwurfsziel von Ruby. Und man hat dann einen Vergleich gemacht zwischen den einzelnen Programmiersprachen… Von der Lesbarkeit… Und da hat Python wesentlich besser abgeschnitten.

Also das muss ich sagen mit Java und so… Also Python ist einfach eine extrem lesbare Programmiersprache. Und intuitiv sehr gut verständlich. Okay, unter Programme… Dann haben wir genau Module… Genau objektorientierte… Das kommt dann ganz zum Schluss. Aber was auch ziemlich simpel ist, weil es ja eigentlich nur… Einfach nur das `class`-Konstrukt gibt, und dann hat man `def`… Und das es private ist, dass macht man halt mit einem Underline… Einfach so eine Konvention…

Yazicioglu Also dann im ersten Jahr haben sie objektorientierte [Programmierung] auch gelehrt?

Lehrer 1 Nein, ich bin ja gar nicht soweit gekommen. Weil ich ja immer nur wollte, dass die [Schüler] algorithmisch denken. Mir ist es darum gegangen, dass sie algorithmisch denken lernen. Also nicht so… Das sie ein Problem haben und den Algorithmus dazu schreiben können. Das wollte ich eigentlich haben. Und an diesem algorithmischen Denken bin ich gescheitert im ersten Jahr…

Yazicioglu Dann hängt das von der Sprache ab?

Lehrer 1 Nein, aber das ist überhaupt. Es gibt ja ein paar Sachen, die einem nicht so klar sind in der Schule. Weil irgendwie kommt einem das wahrscheinlich gut und sinnvoll vor. Aber irgendwie hat man da 20 Jahre vor sich einen beamten Lehrer. Der erzählt dann irgendwas. Und egal ob der arbeitet oder nicht arbeitet, dass spielt also eigentlich keine Rolle. Und die Schüler sind dann drinnen und kriegen das dann mit. Und hätten gerne positive Noten… Das ist aber alles.

Also das ist… Okay, und jetzt sind die halt drinnen und versuchen intuitiv zu verspüren was sie brauchen um zu einer positiven Note zu kommen. Und wenn sie das Gefühl haben, sie schaffen es ohne, dass sie sich anstrengen, dann machen sie es ohne sich anzustrengen. Und irgendwie haben sie sich nicht so wirklich angestrengt. Also es war… Ich bin ja irgendwie… Ich weiß nicht… Schon eh immer ziemlich nett zu den Schülern und dann tue ich mit den Noten… Dann habe ich oft den Nerv nicht… Wenn sie ein bisschen etwas tun, dann bekommen sie doch eine positive Note. Und weiß ich nicht. Ich hab auch keinen besonderen Notendruck. Das kommt auch noch dazu.

Yazicioglu Okay

Lehrer 1 Es haben dann alle eine positive Note bekommen, weil ich mir gedacht habe ich bin gescheitert an diesem Python-Projekt. Aber es ist nicht das Python-Projekt, an dem ich gescheitert bin, sondern dass sie algorithmisch Denken… Das zu vermitteln…

Yazicioglu Haben die Schüler dann bei dem anderen Lehrer…

Lehrer 1 Na, der hat das nur behauptet. Das stimmt nicht.

Yazicioglu Okay.

Lehrer 1 Weil der Lehrer kann nicht programmieren. Und die Schüler konnten zwei Jahre danach… Später nicht programmieren…

Yazicioglu Aha

Lehrer 1 Und zwar das war einfach nur Gerede. Also das…

Yazicioglu Weil ich denke mir wenn man in Python gescheitert [ist], wie kann man in C#…

Lehrer 1 Weil der Lehrer ja… Überhaupt mit diesem Einklammern… Der hat ja gar kein Gefühl für algorithmisches Denken. Ja, also dem ist das ja nicht wichtig, dass er… Aus seinem Kopf etwas konstruiert was schön ist. Das hat ja für den Lehrer keine Bedeutung. Deshalb… Der sagt am Schluss: Haben sie denn gewusst was `public`, `private`… Was der Unterschied ist… Und… Oder so irgendwas. Ich weiß nicht genau was dem sein Ziel ist. Aber algorithmisches Denken haben die sicher nie gekonnt.

Yazicioglu Okay. Welche Vorteile können sie…

Lehrer 1 Ich wollte ihnen… Man sieht das jetzt eh schön bei dem Code.

Yazicioglu Ah, e-Mail senden…

Lehrer 1 Das hat er aber nur hingegeben… Dann das ist so ungefähr der Kurs am Anfang. Und da brauchen sie so ungefähr, dass sie den durchhaben… So zehn Stunden. Ja, zehn Arbeitsstunden, dann haben sie das… Allerdings muss ich jetzt sagen, für die ersten mache ich es nicht so mit dem Kurs. Sehen sie eh mit Objektorientierung… Das ist schon zu weit.

Aber diese Beispiele mache ich mit ihnen. Also Hypothenuse und Ziffernsumme berechnen… Und Summe von n natürlichen Zahlen berechnen. Sowas würde ich dann mit denen machen. Okay… Das sind einfach so Übungen… Was haben wir da gemacht… Ein paar Lösungen hingegeben… Und ich mach es ihnen sogar so, dass ich ihnen immer die Lösung tippe. Schaut die Lösung für das ist das *Tipp*, *Tipp*, *Tipp*… Und ihr müsst es dann nachmachen. Sie sollen sich halt erinnern. Wenn sie ein Foto machen… Ich mein du kannst schon ein Foto machen, aber das ist die Idee nicht, weil du ähnliche Sachen lernen sollst. Du musst das einfach zusammenbekommen. Und du kannst mich eh immer fragen. So… Okay, das sind alles noch…

Ich meine das werden dann… Genau da… Matrix-Operationen… Matrix addieren… Das wäre zum Beispiel eines

Yazicioglu Gutes Beispiel…

Lehrer 1 Ja, aber da haben wir das… Das ist nämlich echt super…

(Diskussion über nicht relevantes Thema)

Lehrer 1 Das ist sehr gut. Übrigens jetzt habe ich eine Funktion geschrieben, also eigentlich ein Programm… Das kann folgendes machen. Das kann entweder das Maximum machen, eine Liste summieren oder eben die Hilfe ausgeben. Und man sieht schon ziemlich gut, dass man so eine Hilfe-Funktion hat… Dieses `foo` hab ich es genannt. Mit `-h` die Hilfe… Und jetzt kann ich ausprobieren. Wenn ich jetzt ein paar Zahlen eingebe: 7, 4… Dann muss ich ein „s“ für Summe davor… Dann summiert man es. Also ich hab Hilfe + Funktion + eine technische Dokumentation. Ah okay…

Yazicioglu Was war die Aufgabe?

Lehrer 1 Eigentlich nur ein Programm machen, dass summiert und ein Maximum macht. Und das ist eigentlich nur…

Yazicioglu Also hier steht es glaub ich nicht…

Lehrer 1 Nein, nein… So, also wie machen wir das grundsätzlich, dass wir… Da muss man mal diesen Parser initialisieren… Und mit `argparse` geht das. Und sie tun eben die einzelnen Aktionen hinzufügen… Und dann gibt es eben diesen Handler, der aufgerufen wird… Das ist die Frage, ob bei den Argumenten ein sum dabei gewesen ist… Das genau definiert worden ist, dass da die Variable `sum` heißt… Eigentlich genau anders herum. Genau da wird das zurückgeliefert.

Okay, da haben wir jetzt ein Interface. Und jetzt ist die Frage: Wie schafft man das… Hm, Java… Das man Interfaces macht, die einen eine Hilfsfunktion ausgeben… Nämlich ganz genau wie das Interface definiert ist. Wie Linux-Kommandos… Also wie Linux-Hilfen… Ich glaub, dass das in Java nicht einmal vergleichbar einfach geht. Also ich glaube da ist ein irrer Unterschied, wenn man mal anfängt Interfaces zu machen…

Yazicioglu Okay. Sie [machen] Interfaces im ersten Jahrgang.

Lehrer 1 Ja, ja… Also das Interface programmiert. Also im ersten Jahr würde ich das nicht machen. Da ist es zu kompliziert. Vor allem im ersten Jahr hab ich noch nicht geschafft, dass sie algorithmisch denken. Ich bin ja bei ganz einem anderen Problem.

(Diskussion über nicht relevantes Thema)

Yazicioglu Was ist für Sie das Ziel von Anfangsunterricht?

Lehrer 1 Also am Anfang ist es nur, dass sie algorithmisch denken können… Und das ist das beste Werkzeug. Und wenn jemand nicht algorithmisch denkt… Wirklich ohne irgendwelche Overhead ganz genau auf das, dass sie lernen algorithmisch zu denken… Und die Emotionen habt dazu dazu… Und die Schönheit von der Lösungen würdigt. Und das eigenständig Geschaffene würdigt. Das kann man damit ohne wirklich einen Overhead sehr sehr gut machen.

Yazicioglu Wegen welchen Kriterien würden sie eine Sprache wählen? Wegen der Industrie…

Lehrer 1 Nein, sicher nicht…

Yazicioglu wegen den didaktischen Ansätzen, weil [sie] eine gute Funktionalität/Bibliothek mitbringt…

Lehrer 1 Also in dem Fall… Bei Python gibt es auch… Ich weiß nicht ob sie das kennen… Das „Beautiful Codes"… Da geht es darum… Also die Schönheit von einem Software-Produkt…

Yazicioglu Effektiv Java-Code?

Lehrer 1 Nein, es heißt einfach nur „Beautiful Codes". Es ist einfach so ein Buch. Wo sie dann halt erklären wie zum Beispiel das NumPy… Also da gibt es ein paar Software-Pakete, die er dann halt vorstellt… Oder Software-Konzepte… Und der von dem „Beautiful Codes", der… Der hat dieses Schönheits-Thema.

Und dann gibt es… Da habe ich einmal eine Liste gesehen von unterschiedlichen Produkten. Und zwar wo man das Gefühl hat, die sind einfach genial und schön und harmonisch. Oder die sind halt irgendein Werkzeug. Da hat er dann… Wenn sie Bash oder so kennen… Was ich auch sehr viel verwenden… Aber da hat man nicht so sehr das Gefühl, dass ist einfach perfekt. Und bei Linux habe ich einfach das Gefühl, dass ist absolut perfekt. Und bei Python hab ich das auch.

Es sind einfach so viele geniale, harmonische Sachen drinnen… Das ist wirklich großartig. Und diese Schönheit, Harmonie, und diese Perfektion… Die verwenden zu können… Das ist sicher etwas Tolles. Das ist eine Motivation die ich habe für das Ganze. Das das einfach ein schönes Produkt ist. Das ich verwende… Mit einer schönen Philosophie… Also die Schönheit von dem Ganzen… Ich würde sagen die Schönheit…

Yazicioglu Gutes Argument, dass kann ich verwenden…

Lehrer 1 Also egal was sie machen wollen… Sie können das einfach umsetzten… Mir ist einfach algorithmisches Denken wichtig. Ich kann alles weglassen und das tun. Wenn jemand… Ich sage mal objektorientiert zu denken und objektorientierte Sachen zu machen… Also von den Kollegen, dann könnte er das genauso tun. Ja, da wäre überhaupt kein besonderer Unterschied.

(Diskussion über nicht relevantes Thema)

Lehrer 1 Ich meine was in Python auch ein ziemlich auch ein gutes Sprach… Aber das hat mit dem ersten wieder nichts zu tun… Das man in Variablen Funktionen übergeben kann. Und damit schafft man es dann Decorators zu machen… Also das heißt man kann bei diesen Klassen [bei den] Methoden [einen] Decorator angeben. Und damit wird die Funktion dann verändert und bearbeitet. Und man kann zum Beispiel zählen, wie oft eine Funktion aufgerufen worden ist. Einfach durch diesen Decorator, den man dazugibt. Ich glaube das funktioniert auch nicht in Java…

Yazicioglu Da muss man eine andere Klasse dafür schreiben…

Lehrer 1 Ja… Irgendwie…

Yazicioglu Aber warum [ist] die objektorientierte Programmierung [in Python] nicht sauber?

Lehrer 1 Ja, aber das…

Yazicioglu Mehrfachvererbung finden manche auch ist ein Nachteil… Finden sie das auch.

Lehrer 1 Ja Mehrfachvererbung finden sicher alle Sprachen als Nachteil, die das nicht anbieten können. Die finden das sicher alle sehr nachteil-haft. Und der Betrand Meyer, also der die Programmiersprache Eiffel gemacht hat. Also so ein Franzose… Der war einmal in Wien. Der hat nämlich ganz schön gezeigt… Und schön ausgearbeitet, warum Mehrfachvererbung sinnvoll ist. Was aber eher akademisch ist, weil tatsächlich kann man in der Praxis mit der Einschränkung, dass es keine Mehrfachvererbung gibt sehr gut leben kann. Also ich denken nicht, dass das

Yazicioglu Aha. Er hat das für die Eiffel-Programmiersprache…

Lehrer 1 Er hat Eiffel… Dem sein Werk ist halt Eiffel… Und da argumentiert er aber echt schön, warum Mehrfachvererbung, warum das sinnvoll ist…

Yazicioglu Wie heißt er?

Lehrer 1 Bertrand Meyer… Und Eiffel wie der Eiffel-Turm…

Yazicioglu Okay, Dann such ich [danach]… Spielt eine Programmiersprache eine Rolle bei der Motivation, Demotivation oder Erfolg? Spielt eine Programmiersprache eine Rolle dabei?

Lehrer 1 Nein, ich glaube das ist egal. Es kommt echt auf den Lehrer an. Und beim Lehrer ist es wichtig… Ich glaube es sind zwei Sachen wichtig. Erstens mal, ob man sich mag. Also sozusagen ob man den Lehrer mag. Oder ob man die Schüler mag. Das ist echt ziemlich wichtig, glaube ich. Und das nächste ist dann… Ob… Wenn man ihnen dann einen Fünfer gibt. Wenn man… Verstehen sie… Wenn alle recht gut ohne… Also wenn man die Latte…

Yazicioglu Also die Schüler am Anfang [eine] Fünf bekommen, dann gibt es keine Motivation mehr für die Programmierung meinen sie?

Lehrer 1 Nein, wenn sie… Ich weiß nicht…

Yazicioglu Besonders bei welchen Konzepten haben die Schüler Probleme? Bei der Syntax, bei der Semantik, bei der Fehlersuche oder bei den objektorientierten Konzepten. Wo haben die Schüler meistens Probleme?

Lehrer 1 Nein, ich finde das Problem das haben… Ich meine Syntax, Semantik, Fehlersuche… Ich meine… Fehlersuche ist sicher ein Problem. Aber da fragen sie dann immer mich, warum es nicht funktioniert. Weil sie den Blick nicht so haben. Weil… Also ich sehe ganz genau, wenn ich ein Programm sehe… Was daran… Es brennt sich ein… Bei denen nicht so… Die sehen das nicht so. Ich kann es nicht sagen…

Yazicioglu Sie verstehen nicht die Fehlermeldungen zu…

Lehrer 1 Ja, das durchlesen… Was hat das damit zu tun. Sie können… Aber das ist irgendwo im Denken so irgendwie. Ja… Das es ein Ausdruck ist… Das tut man dann irgendwie so verinnerlichen. Sie können nicht so genau hinschauen…

Yazicioglu Ich hatte auch…

Lehrer 1 Aber trotzdem… Das Hauptproblem ist… Also es gibt wie in der Mathematik gibt es ja auch so Lösungen. Also so Aufgabenstellungen… Und beim Programmieren hat man einfach viel mehr Freiheiten, und da ist einfach dieses algorithmische Denken. Algorithmisches Denken… Das können sie nicht… Und das ist mir damals in dem ersten Jahr nicht gelungen. Ich meine ich würde es genau so wieder probieren… Und es ist dann so, dass das mengenorientierte Denken… Das das schon gegangen ist. Also bei 17-Jährigen geht das, dass sie mengenorientiert denken lernen. Das ist ungefähr die gleiche… Also eigentlich intellektuell ist es eine größere Herausforderung. Und die 15-Jährigen… Diese eine Erfahrung… Dieses algorithmische Denken… Das hat nicht funktioniert.

Yazicioglu Dann hängt es vom Alter ab?

Lehrer 1 Alter oder… Die sind dann ja in der Dritten. Also haben sie… Zwei Jahre sind sie mit dem technischen Denken eigentlich konfrontiert gewesen. Weil in der HTL gibt es ja Netzwerktechnik usw. Das ist ja alles irgendwie so technisch. Und Mathematik… Und wenn man zwei Jahre technisch gedacht hat, dann geht es vielleicht auch besser, möglicherweise… Aber das ist alles Spekulation. Ich weiß es einfach nicht.

Yazicioglu Okay. Die Schüler in der HTL… Gehen die Schüler [nach] der ersten Klasse weg, weil sie denken, dass sie nicht programmieren können… Das sie nicht technisch…

Lehrer 1 Naja die gehen weg… Weil sie nichts machen… Sie tun nichts. Also

Yazicioglu Sie haben dieses Talent, aber sie tun nichts…

Lehrer 1 Naja, Malcolm Gladwell eben… Und das finde ich auch wirklich gut, der sagt: Egal wer 10 000 Stunden trainiert. Also es ist noch nie so gewesen, dass dann der nicht einer von den welt-besten Geigern gewesen wäre oder sowas. Also nach 10000 Stunden ist man einfach, egal was man macht extrem talentiert… Und er hat dann auch Nobelpreisträger gezeigt, die nicht so [super] gescheit waren, aber es dann doch irgendwie gemacht haben. Weil sie einfach viel Zeit investiert haben. Also ans Talent glaube ich nicht so wirklich. Also ich glaube es ist einfach… Wenn mich das begeistert und ich das dauernd mache, dann spielt das nicht so eine Rolle. Und wenn ich viele viele Stunden damit verbracht habe, dann bin ich einfach gut darin. Und zwar ausreichend. Das es wirklich so ist, dass ich eine großartige Leistung erbringen kann in dem Gebiet.

Yazicioglu Okay. Wenn wir zusammenfassen…

Lehrer 1 Also Talent ist es nicht. Ich denke es ist nur die Zeit, die man investiert. Die Zeit die sie investieren ist ob sie das Thema fasziniert. Ob sie das wollen. Dafür ist Python sehr geeignet. Weil man das… Weil man kann das recht gut, ohne mühsamen Overhead ziemlich genau den Reiz des Ganzen, des algorithmischen Denken rüberbringen. Und ja…

Yazicioglu Hilft Python da? Beim algorithmischen Denken sehr viel? Weil in Python gibt es keinen Overhead… Man beschäftigt sich mit dem nicht sondern…

Lehrer 1 Und das ist das Problem und ich löse es. Und das ist den anderen Lehrern ja komplett [egal], weil die ja nicht programmieren können. Und deshalb haben sie auch überhaupt kein… Das ist ja überhaupt ihr Thema nicht. Also das ist… Die haben nicht… Es gibt ein neues Problem, nicht vorgefertigte Beispiele, die sie seit 10 Jahren machen. Sondern es gibt ein neues Problem… Wie löse ich das? Die Schönheit des durchdenken… Und das freudige Erlebnis, dass man da eine Lösung gefunden hat. Das ist genau das was sie nicht vermitteln. Und genau das ist aber das was notwendig ist, damit man… Also damit man sich auseinandersetzt. Und dann gibt es… Die sind ja auch komplett wahnsinnig in dem Alter. Dieses… Nämlich das… Ein Schüler…

Yazicioglu Ja, also ich hab begonnen mit 18, mit 19… In diesem Alter war es sogar für mich schwer. Mit 15 glaube ich das…

Lehrer 1 Ja genau. Ein Schüler habe ich gehabt, der hat… Ich meine da war ich Klassenvorstand. Ja und der hätte etwas bringen sollen… Und dann habe ich jede Woche…

…Also ist wirklich eine gute Wahl. Also bei Python ist sicher von den… Es ist auf jeden Fall eine gute Wahl. Aber angekommen tut es auf andere Sachen. Ich glaube das Wichtigste ist das Lehrer-Schüler-Verhältnis, und das zweite ist auch, dass der Lehrer für das begeistert ist… Für das was er macht. Also das ist wichtiger.

Yazicioglu Mit Python kann man die Schüler sehr gut begeistern?

Lehrer 1 Mit Python… Aber das könnte ich ganz genauso mit Java oder C# machen. Also das wäre nicht… Da wäre es zwar ein bisschen schwieriger, aber das ist nicht das größte Problem. Mein größtes Problem ist: Habe ich ein gutes Verhältnis mit den Schülern… Und das was ich unterrichte… Bin ich davon begeistert oder nicht… Was ich aber mit C# genauso wäre. Also das wäre…

Yazicioglu Aber für die Schüler Python finden sie besser?

Lehrer 1 Nein, nein. Ich finde es sicher besser. Aber es macht jetzt wenn ich überlege… Lehrer-Schüler-Verhältnis, Begeisterung für den eigenen Gegenstand und die Auswahl der Programmiersprache… Dann würde ich sagen, Lehrer-Schüler-Verhältnis sind mindestens 50%, also mindestens… Also ob man einen Lehrer mag oder nicht… Ob man sich gut versteht oder nicht, das macht was aus. Und wenn es darum geht ob man begeistert ist… Das macht sicher von den anderen 50% aus… Drei Viertel ist Lehrer-Schüler-Verhältnis und die Begeisterung für das was man macht. Dann die Programmiersprache selber, das ist nicht sehr viel mehr. Also eigentlich ist es beim Unterricht nicht so wichtig… Also auf das kommt es jetzt nicht so an, ob das jetzt Python ist oder was anderes.

Yazicioglu Aber trotzdem würden sie Python wählen.

Lehrer 1 Ja, wenn es jetzt um diesen kleinen Teil geht, in diesem gesamten Rahmen… Dann ist Python… Python hat deswegen… Wegen Verbreitung und so… Jetzt überlege ich wegen einer Alternative… Also Ruby ist einfach nicht so verbreitet… Das könnte ich mir auch noch vorstellen. Java hat schon einen Sinn… Nämlich Java macht einen Sinn, wenn man das Gefühl hat, man möchte unter Kollegen sich abstimmen. Und das die ganzen Jahre machen. Und die Plattform nicht wechseln. Dann würde ich Java ganz gut finden. C# finde ich dann gut wenn sich die Schule dafür entscheidet sich auf einen bestimmten Hersteller zu fokussieren. Was ich aber strategisch für keine gute Idee [halte]. Also ich finde eine Fokussierung auf Microsoft nicht so gut. Also deshalb würde ich es auch nicht machen. Weil ich auch nicht weiß welche Zukunft das hat. Also im Moment ist Microsoft… Die kämpfen an allen Ecken und Enden. Also im… Also C# halte ich nicht für eine gute Idee.

(Diskussion über nicht relevantes Thema)

Lehrer 2

Yazicioglu Welche Programmiersprache würde Sie als Unterrichtssprache bevorzugen? Warum? Welche Vorteile/Nachteil können Sie nennen?

Lehrer 2 Also meiner Meinung nach, ist es im Prinzip ist vollkommen [egal], mit welcher Programmiersprache man im Unterricht beginnt. Es sollte nur nach Möglichkeit eine sein, die man auf möglichst vielen Plattformen verwenden kann.

Yazicioglu Okay

Lehrer 2 Das heisst nicht nur auf Windows, sondern auch auf Linux, auf Mac OS was auch immer. Da ist Python eh an sich eine eine gute Wahl. Wir für uns haben uns halt für C# entschieden, was jetzt auch quell-offen ist und wo wir auch kein Problem haben mit der Plattformen, weil es ja auch schon offene Bibliotheken gibt. Ich sehe jetzt ehrlich gesagt nicht das Problem bei der Wahl von der Programmiersprache, sondern generell das Problem beim Programmieren an sich, dass jetzt unabhängig von der Programmiersprache ist.

Yazicioglu Das wollte ich auch fragen: Spielt die Programmiersprache selbst eine Rolle bei der Motivation…?

Lehrer 2 Nein, Nein überhaupt nicht. Also bei uns wäre es jetzt vollkommen [egal], ob ich jetzt Java, C# oder PHP mache. Es ist vollkommen [egal] weil, gerade in der ersten Klasse sind die Probleme ganz woanders. Das [Problem ist] einfach dieses logische Denken, das Denken in Abläufen, das das von der Programmiersprache an sich komplett unabhängig ist, dass man zum Programmieren braucht. Aber das muss man erst lernen — durch Üben.

Da ist es [dann] eigentlich vollkommen [egal], ob ich dann meine strukturierte Vorgangsweise umsetze in Java, ob ich die umsetzte in C# oder ob ich die umsetzte in PHP. Das Problem, was wir gesehen haben bei den Schülern ist einfach ein ganz ein Anderes: Ein Problem zu analysieren, in kleine Teile zu zerlegen, und sich dann Schritt für Schritt der Lösung nähern. [Das] ist von der Programmiersprache her vollkommen abgekoppelt und egal.

Yazicioglu Okay. Denken Sie das eine leichtere Syntax, saubere Syntax (von einer Programmiersprache) ein wenig hilft. Haben Sie einmal mit Python programmiert? Kennen Sie Python?

Lehrer 2 Ich hab jetzt mit Python selber noch sehr wenig [gemacht], aber ich glaube es ist jetzt nicht unbedingt die Syntax das Problem. Weil die ist in vielen Programmiersprachen sehr ähnlich. Die Syntax ist etwas, dass kann man lernen. Das haben die Schüler über mindestens acht Jahre hinweg schon gelernt: Zu lernen, etwas auswendig zu lernen. Denen zu sagen, wenn du eine `for`-Schleife brauchst, dann muss die so ausschauen das geht.

Es ist eher das Problem, dass sie gar nicht zu dem Punkt kommen, dass sie wissen, dass sie da jetzt eine `for`-Schleife einsetzen müssen. Es ist auch nicht das Problem etwas auf der Konsole auszugeben oder etwas von der Konsole einzulesen. Es ist eher das Problem zu wissen wann gebe ich überhaupt [et]was aus und wann muss ich einlesen.

Yazicioglu Okay, da spielt die Programmiersprache keine Rolle…

Lehrer 2 Da ist die Programmiersprache vollkommen [egal]. Das man einfach wenn man irgendwo Methoden macht, dass die bestimmte Parameter bekommt und [etwas] zurückliefern soll… Einfach das Verständnis dafür zu haben, dass ich nicht in der Methode, auf einmal das was ich eh mitbekomme von der Konsole nocheinmal einlese. Das sind dann so die Sachen wo dann halt einfach die Übung kommen muss und wo die Schüler erst nachher oder oft zu spät drauf kommen: Hoppala, das ist wie Radfahren, dass kann ich nicht lernen, dass muss ich üben.

Yazicioglu Sie haben [es] mit verschiedenen Programmiersprachen probiert: Mit Java, C#, PHP. In allen Sprachen haben die Schüler die gleichen Probleme gehabt?

Lehrer 2 Genau.

Yazicioglu Okay. Welche didaktische Methode verwenden Sie im Unterricht, was ist für Sie didaktisch wichtig beim Programmier-Unterricht?

Lehrer 2 Naja, dass man es so vielfältige wie möglich macht, dass für jeden irgendwas dabei ist, was er [mitnehmen] kann, und vor allem auch, sehr viele Übungsbeispiele zu bieten. Die sie [die Schüler] dann vielleicht in Selbstkontrolle durchmachen.

Yazicioglu Ist es wichtig im Unterricht [etwas] Schritt für Schritt zu erklären? Finden Sie das imperative Programmierparadigma wichtig oder ist es gut, dass Schüler am Anfang objektorientiert, mit OOP, beginnen?

Lehrer 2 Also mit dem Objektorientierten können Sie am Anfang sicherlich noch gar nichts anfangen. Mann muss schon zu erst mal das normale, Strukturierte durchgehen.

Yazicioglu In Java und C# beginnt man nicht mit OOP sondern man schreibt `public static void main`. Aber das am Anfang…

Lehrer 2 Genau. Wir fangen an, man macht eine eigene [Klasse]… Das ist eine Vorgangsweise… Wir sagen ihnen, so das wird so gemacht. Warum man das so macht, erfahren sie dann irgendwann im zweiten, dritten Jahrgang. Das sie wissen, was sie da machen. Aber jetzt fürs erstes sollen sie [das] einfach nur so hinnehmen, und so machen, dass das so ist.

Yazicioglu Okay

Lehrer 2 Das man eine Klasse macht: `MethodenBox mb = new MethodenBox`… Das ist einfach so und damit haben sie auch kein Problem. Das lernen Sie auswendig und das schreiben sie hin und fertig.

Yazicioglu Das wollte ich fragen, weil manche Lehrer finden das didaktisch ganz schlecht. Die Schüler haben zuviele Problem oder Fragen. Was ist `public`? Was ist `void`? Was brauche ich? Finden Sie das ist ein Problem?

Lehrer 2 Nein, weil wir erklären es Ihnen am Anfang gar nicht.

Yazicioglu Sie sagen das später…

Lehrer 2 Wir sagen einfach, das ist so. Warum man das so macht lernen Sie dann erst später.

Yazicioglu Haben die Schüler damit Probleme?

Lehrer 2 Nein, überhaupt nicht.

Yazicioglu Am Anfang, später auch nicht?

Lehrer 2 Nein.

Yazicioglu Okay

Lehrer 2 Sie wissen, das macht man so. Dann schreiben sie das auch einfach so hin. Die Problem sind dann wie gesagt, eher das es eine Aufgabenstellung gibt und [ich] mich mal hinsetzten muss und überlegen: Was muss ich denn da jetzt überhaupt tun? Das ist eher das Problem bei dem Ganzen.

Yazicioglu Okay. Finden Sie das didaktisch gut: Das man die Schüler… Sie lernen [es jetzt] auswendig und sie lernen [es] später alles detailliert. Ist das didaktisch gut? Didaktisch okay?

Lehrer 2 Na, sagen wir so: Wenn ich von Anfang an das erklären würde, dann kämen sie nie zu irgendwelchen praktischen Sachen. Abgesehen davon, dass Objektorientierte Programmierung was ist, das ist für 15, 16-Jährige einfach noch zu komplex vom Verständnis her. Das nimmt man einfach so hin, [das] wissen sie auch. Da haben die Schüler kein Problem damit. Sie müssen sich dann auf die eigentliche Aufgaben konzentrieren.

Weil die Projekte oder die Übungen, die man Anfang macht, die sind so einfach gehalten, dass sie da die Konzepte von OOP gar nicht brauchen am Anfang. Also gerade beim programmier-einsteigen brauche ich keine Eigenschaften. Ich brauch keine…, ich muss nicht wissen, was ist `private`, was ist `public`. Das brauche ich am Anfang alles nicht. Weil die, die Programmieren lernen bei uns, [von denen] haben viele noch nicht programmiert. Und wenn man da jetzt anfängt das zu erklären, dann sind sie überfordert.

Yazicioglu Ja, stimmt.

Lehrer 2 Sie müssen zuerst mal lernen, die Kontrollstrukturen kennen lernen, diese kleinen Bausteinen richtig zusammen zu setzen, kleine Probleme zu analysieren, eine Schritt- für Schritt-Anleitung zu machen, sich eine Abfolge überlegen. Was muss ich tun? Was brauche ich für den nächsten Schritt, dass ich das machen kann? Wenn das einmal sitzt, dann kann man hergehen, und kann den nächsten Schritt gehen und kann sagen okay, jetzt kommt die nächst größere Aufgabe. Jetzt zerlegen wir die mit Hilfe [von] OOP in kleinere [Teile] und das können wir aber schon.

Yazicioglu Mh, okay.

Lehrer 2 Also nicht irgendwo von oben nach unten anfangen, weil dann…

Yazicioglu sondern von unten nach oben.

Lehrer 2 Genau

Yazicioglu Finden Sie dynamische Typisierung ist einen Nachteil? Python hat starke Typisierung. Alles wird als Objekt angesehen in Python. Finden Sie das dieses Eigenschaft ein Nachteil von Python ist? Oder [ist das] für den Anfang für die Schüler okay?

Lehrer 2 Hm, Naja. Kann man jetzt so glaube ich nicht direkt beantworten. Für den Anfang ist es wahrscheinlich schon mal gut zu wissen, womit habe ich es zu tun. Diese starke Typenorientierung ist für den Anfang sicherlich nicht schlecht. Das man sich einmal bewusst wird, womit habe ich es eigentlich zu tun. Rechne ich jetzt mal mit einer Komma-Zahl, rechne ich mit einer ganzen Zahl.

[Ich] denke mir schon, dass das sehr wichtig ist, weil sonst kommen Ergebnisse raus, und ich weiss nicht warum das nicht stimmt. Ich habe hier eh eine Variable deklariert, wenn das Ergebnis auf einmal ganz anders ausschaut, dann weiss ich nicht, wo in dieser ganzen Berechnung, wo ist der Fehler passiert.

Wo ist bei einer implizierten Typumwandlung ein Fehler passiert? Wo kann er umwandeln? Wo kann er nicht umwandeln. Wenn ich von Anfang an weiss: Okay es gibt diese und jene Typen, dann ist es wahrscheinlich gescheiter, wenn man sich [ein] bisschen mehr bewusst wird, was man da tut.

Yazicioglu Okay. Was ist für Sie das Ziel des Anfangsunterrichts? Das ist so eine allgemeine Frage. Ich weiß…

Lehrer 2 Die Schüler mal dazu bringen, über ein Problem nachzudenken.

Yazicioglu Algorithmisches Denken

Lehrer 2 …Algorithmisches Denken, analysieren, in Abläufen denken.

Yazicioglu Okay

Lehrer 2 Wenn mal die Schüler in Abläufen gelernt haben zu denken, dann ist auch das Programmieren kein Problem mehr.

Yazicioglu Egal welche Sprache. Wenn man eine Sprache gut beherrscht kann man andere Sprache auch beherrschen oder… Ich finde wenn man eine Skriptsprache beherrscht kann man traditionelle Sprache gut lernen oder hat man [dann ein] Problem?

Lehrer 2 Das hängt jetzt von der Lebenserfahrung ab sag ich einmal. Ich denke wenn man programmieren lernt, in einer Programmiersprache und man steigt auf eine andere Programmiersprache um, und man hat nicht so viel Erfahrungen beim Programmieren, dann sagt man „Um Gottes Willen“, das ist eine komplett neue Sprache. Ich muss programmieren komplett neu lernen. Wenn man aber schon viele Jahre programmiert hat und Erfahrung hat, kommt man dann zu dem Punkt, dass man sagt: Ja okay, die Syntax ist bisschen anders aber im Grunde genommen ist es das Gleiche.

Yazicioglu Dann ist es egal von welcher Sprache auf welche Sprache man umsteigt. Von Skriptsprache auf traditionelle…

Lehrer 2 Für später wenn man viel programmiert hat, ist es egal. Für die Schüler ist es natürlich jetzt in dem Alter, wo sie nicht so viele Programmiererfahrung haben, tragisch, wenn sie jetzt innerhalb der Ausbildungszeit von einer Programmiersprache auf die andere umsteigen. Das ist… Das macht man nicht.

Yazicioglu In ihrer Schule machen sie den Umstieg nicht

Lehrer 2 Nein, nein. Wir machen die kompletten fünf Jahren durch eine Programmiersprache. Weil wir genau wissen… Wir selber wissen, das ist egal ob ich jetzt in C#, oder in Java oder in PHP oder in Python irgendwas mache, oder in C++. Aber die Erfahrung haben die Schüler noch nicht. Die sehen, dass heisst jetzt nicht mehr C#, das heisst Java. Und somit wird alles, was man bisher gelernt hat… Da sagt man, das brauche ich jetzt nicht mehr — Ich lernen einer neue Programmiersprache. Das ist der Grund, warum wir sagen, wir haben uns dafür entschieden in unsere Ausbildung starten wir in der ersten [Klasse] mit einer Programmiersprache die wir bis zu fünften [Klasse] durchziehen.

Yazicioglu Für diese Schule ist [bei der] Auswahl der [Programmiersprache] die zukünftige Berufsmöglichkeit [wichtig]. Das ist… Das steht im der ersten Reihe für die Auswahl. Die Plattform soll… Sie verwenden C#- und Java-Plattform weil sie denken, dass ist in…

Lehrer 2 Wir verwenden in erster Linie C#, ja.

Yazicioglu Weil sie denken in der Arbeitswelt, in der Industrie wird C# sehr viel verwendet. Bei den Auswahlkriterien welches ist für Sie wichtig oder für diese Schule: Zukünftige Berufsmöglichkeit, didaktische Ansätze, oder Beides. Oder gibt es andere Gründe?

Lehrer 2 Ein Grund ist natürlich auch unsere eigene Erfahrung. Wir haben jetzt schon… Setzen C# jetzt schon viele Jahre ein. Haben selber entsprechende Erfahrungen mit dieser Programmiersprache und können dann Schüler dementsprechend auch in ein [wenig] komplexeren Projekten unterstützen. Nachdem C# jetzt auch als Programmiersprache für andere Plattformen verfügbar ist, ist das auch kein Thema mehr, das man mit anderen Programmiersprachen arbeitet.

Yazicioglu Okay. Dann letzte Frage. Würden sie Python als erste Sprache wählen?

Lehrer 2 Da sind wir wieder am Anfang. Es ist egal mit was man los-startet.

Yazicioglu Okay… Dann sind meine Fragen fertig.

Lehrer 2 Okay

Yazicioglu Danke schön

Lehrer 3

Yazicioglu Welche Programmiersprache würde Sie als Unterrichtssprache bevorzugen? Warum? Welche Vorteile, Nachteile gibt es für sie?

Lehrer 3 (Unverständlich)…oder ganz allgemein?

Yazicioglu Ganz allgemein. Aber es ist für die Anfänger gedacht.

Lehrer 3 Ja. Okay. Die Programmieranfänger. Ja, die Frage… Wir haben da lange darüber diskutiert im Vorjahr… Also im vorigen Frühling, Frühsommer. Es gibt da verschiedene Aspekte… Von daher würde ich da nicht unbedingt eine konkrete Programmiersprache nennen.

Also meines Erachtens, was wichtig ist… Oder… Am leichtesten kann ich sagen, warum ich bestimmte Programmiersprachen eher nicht haben will. Wenn ich frei wählen könnte. Also eben… Ich würde eine nehmen, die halt interaktiv ist, damit man direkt unmittelbar was ausprobieren kann. Gerade für Anfänger denke ich, ist das ist wichtig. Würde auch tendenziell, also eher eine nehmen, die kein statisches Type-Checking hat.

Yazicioglu Python hat dynamische Typisierung und einen interaktiven Interpreter.

Lehrer 3 Genau. Also in de Hinsicht ist Python schon mal da sehr… Fällt da durchaus rein. Also warum kein statisches Type-Checking? Weil das… Einerseits stört das denk ich auch mal den Gedankenfluss. Die Theorie ist man kriegt gleich früher die Fehler. Aber letztendlich ist das… Kriegt man teilweise Fehlermeldungen für Dinge, die keine Fehler sind. Und… Die andere Sache ist… Eben das… Das man eben beim Programmieren stärker eingeschränkt ist. Und ja… Für mich als jemanden, der es lehren will ist es insbesondere relevant, dass ich… Eine Sache die ich den Studenten beibringen will ist, dass die Interpretation von Daten oder so kommt vom Program.

Also im Prinzip… Was weiß ich… Wenn man eine Zahlen hat, ob das jetzt ein Jahres-Zahl oder eine Mengenangabe oder sonst was ist… Ist eine Frage… Die kommt von Programm her. Jetzt versucht man natürlich mit statischen Typing solche… Also schon eine gewisse von der Programmiersprache her da bestimmte Interpretation auszuschließen. Was aber nur teilweisse gelingt in den Zahlen zum Beispiel nicht. Und irgendwie… Diese… Das denke ich mir führt mehr zur Verwirrung als es hilft.

Andere Leute haben einen anderen Glauben… Das ist eine sehr religiöse Sache. Aber das ist mein Standpunkt.

Yazicioglu Okay. Sie denken, dass am Anfang zu viele Fehler… Sollen die Schüler mit so vielen Fällen… Mit Typisierung… Mit Typumwandlung… begegnen. Sondern wichtig ist es Programme zu schreiben. Schnell ein Programm zu schreiben. Hab ich das richtig verstanden.

Lehrer 3 Durchaus. Also wenn man die Fehler gerade zur Laufzeit kriegt, ist es gerade am Anfang nicht das Problem, meines Erachtens.

Yazicioglu Okay. Und würden sie Python… Da [für] eine gute [Wahl halten]?

Lehrer 3 Also ich kenne Python nicht so gut. Aber vor allen, was ich davon gehört habe und gelesen habe, ist es glaube ich keine schlechte Wahl.

Yazicioglu Und Welche Kriterien, Merkmale und Ansätze berücksichtigen sie bei der Auswahl einer Programmiersprache. Aber sie haben [das] schon erklärt. Gibt es die Ansätze… Zum Beispiel [spielt die] zukünftige Berufsmöglichkeit eine Rolle bei der Auswahl einer Sprache oder [die] didaktische Eignung. Soll man berücksichtigen… Welche Kriterien?

Lehrer 3 Also das mit den zukünftigen Berufsmöglichkeiten spielt nicht direkt eine Rolle. Weil, wenn die Studenten dann in 5 Jahren dann tatsächlich was eben machen müssen, dann müssen sie unter Umständen doch wieder ganz eine andere Programmiersprache lernen. Letztendlich geht es auch darum, dass sie möglichst ein bisschen mehrere Programmiersprache kennen lernen. Damit sie, wenn sie dann in Beruf auf eine x-te Programmiersprache treffen, die dann auch schnell lernen können.

Un eben gerade an einer Universität geht es uns eben darum, dass sie Konzepte lernen und nicht die konkreten Werkzeuge. Also die halt jetzt gerade für die nächsten 6 Monate in Berufsleben angesagt sind. Sondern, dass sie von dem Wissen, was sie haben noch in 20 Jahren profitieren können. Ja…

Yazicioglu Sie haben Konzept gesagt. Welche Konzepte am Anfang, im ersten Jahr… Wichtige Sprach…

Lehrer 3 Also allgemein: Was wird man… Wenn man jetzt eine Programmiersprache… Diese Programmiersprache lehrt… Variablen, Schleifen… Dann eben Funktionsaufrufe, Prozedur-Aufrufe, Parameter… Wie die funktionieren. Solche Dinge, die in allen Programmiersprachen irgendwie ähnlich funktionieren. Und das man eben lernt wie man mit denen umgeht. Die konkrete Programmiersprache ist dabei eigentlich gar nicht so wichtig.

Yazicioglu Denken sie, dass die Schüler sollen zuerst mit [dem] iterativen [Programmierstil] beginnen, OOP, oder ist es egal mit welchem Paradigma sie beginnen? Später

Lehrer 3 Also objektorientierte Sprachen sind ja sozusagen noch eine Ebene über den, oder aufgesetzt auf imperativen Sprachen. Von daher sehe ich schon eigentlich werd ich mit dem imperativen Paradigma anfangen. Man kann das mit einer Sprache machen, die objektorientiert ist, weil die haben ja auch eben den imperativen Unterbau. Aber, man muss das halt in der Sprache… Ja, die Frage ist halt wie das dann aufgebaut ist. Da finde ich Java zum Beispiel ein bisschen…

Yazicioglu …Man beginnt…

Lehrer 3 Nein, Java hat eigentlich einen imperativen Teil und einen objektorientierten Teil. Und die passen teilweise nicht so gut zusammen. Also im Prinzip… Man könnte jetzt mit Java imperativ programmieren lernen. Dann hat man aber viel von dem was Java ausmacht nicht gelernt. Und eben dann gibt es wieder… Also es gibt da einige komische Dinge in Java. Wobei eigentlich gibt es in jeder Programmiersprache einige komische Dinge.

Yazicioglu Gerade von Java sprechen…

Lehrer 3 Moment, ich wollte da noch zu den… Also gut zukünftige Berufsmöglichkeit… Wie gesagt… Also es gibt bei unseren Diskussionen… Hat es dann eben zwei Aspekte gegeben… Mit diesen Modesprachen… Also der eine ist wir wählen Modesprachen weil die Studenten das wollen und die dann besser motiviert sind. Also nicht so sehr weil wir glauben, dass das jetzt unbedingt nötig ist. Das sie das im Ende im Beruf die Sprache können. Weil wir glauben, wenn sie bei uns durch sind, dann lernen sie diese Sprache. Also auch wenn wir sie nicht direkt gelehrt haben, dann können sie die Sprache relativ schnell lernen.

Der andere Aspekt ist eben… Oder die andere Position wäre: Wir machen das nicht mit der Modesprache, sondern lernen etwas, dass sehr weit weg davon ist. Weil, damit es eben die Leute, die sagen wir von HTL zu uns kommen… Jetzt auch irgendwas… Nicht glauben, sie können schon alles. Damit die auch etwas haben, was sie lernen können. Also das…

Yazicioglu Das für die Schüler auch neu [ist]…

Lehrer 3 Genau. Das war halt der andere Punkt. Aber was sich im Moment bei uns durchsetzt ist der erste Standpunkt. Also wir lehren die Modesprache, weil die Studenten das so wollen.

Yazicioglu Modesprache, oder die Sprache, die Schüler bis jetzt nicht gelernt haben… [Da] sollen die Schüler auch die Aspekte sehen… Didaktische Eignung sehe ich nirgendwo. Oder…

Lehrer 3 Didaktische Eignung ist eben in den Dingen die ich erwähnt hab. Also das meines Erachtens man interaktiv mit der Sprache programmieren soll. Und eben, dass [es] keine statische Typisierung geben soll. Es kommt auf das didaktische Konzept an. Mein Kollege, der ist halt sehr auf statische Typisierung aus. Für den muss eine Sprache natürlich statisch typisiert sein, weil er das auch lehren will.

Yazicioglu Ja, mit einem Lehrer hab ich gesprochen. Statische Typisierung und einen interaktiven Interpreter hat [er] als Nachteile gesehen, der Lehrer. Weil statische Typisierung muss… Da sieht man die Fehler nicht sofort, sondern…

Lehrer 3 Bei der dynamischen meinen sie?

Yazicioglu Ah bei der dynamischen, ja… Und man erkennt die Fehler nicht sofort, sondern später. Und bei großen Programmen ist das ein bisschen problematisch denkt er.

Lehrer 3 Das ist natürlich ein Argument. Ist aber meines Erachtens für Programmieranfänger geht es ja nicht um große Programme.

Yazicioglu Stimmt. Wichtig ist das Ziel. Was man lernt am Anfang. Welches Konzept ist das? Und was ist das Ziel am Anfang?

Lehrer 3 Also eben didaktische Eignung, wenn ich jetzt wirklich meine statische Typisierung ist ein wichtiges Konzept, dass ich den Studierenden beibringen will, dann werde ich dann halt auch so eine Sprache wählen. Aber…

Yazicioglu Welche Sprache ist es… Welche Sprache unterrichten sie, die dynamische Typisierung hat. Unterrichten sie solche Sprachen?

Lehrer 3 Ich… Also in meinen Lehrveranstaltungen unterrichte ich Forth, was weder dynamisches noch statisches Type-Checking hat. Wo man… Wo wirklich der Programmierer dafür verantwortlich ist, dass er die richtigen Operationen anwendet. Und es von der Programmierung her eher wie statisches Type-Checking, weil ja die Laufzeit nicht weiß was das für Typen sind. Und die ja auch nicht, je nach Typ so ein „Plus" oder so ein „Plus" macht. Also man muss wirklich schauen… Im Programm hinschreiben, ich will jetzt ein Gleitkomma-Plus oder ich will jetzt ein Integer-Plus oder sowas.

Yazicioglu Okay. Von der Art des Unterrichts. Welches Ziel ist für sie wichtig im Anfangsunterricht? Zum Beispiel: Die Schüler sollen schnell programmieren lernen, oder was für ein Ziel betrachten sie beim Anfangsunterricht?

Lehrer 3 Naja, das sie am Ende halt die Konzepte, die ich da lehren will können und verstehen. Und anwenden können. Insbesondere halt… Was waren das für Konzepte? Müsste ich mal nachschauen…

…Die halt da vorgekommen sind.

Yazicioglu Am Anfang halten sie die aber für nicht so geeignet um Sprachelemente, um die einfachen Dinge… Algorithmus zu verstehen, oder allgemeine Dinge zu verstehen. Am Anfang finden sie [das] nicht so gut geeignet. Aber, wenn man beginnt mit Abstraktion, mit objektorientierter Programmierung, dann finden sie es gut, dass einfach so zu machen.

Lehrer 3 Also sagen wir, der Grund, warum letztendlich auf Java hinausgelaufen ist, weil der Rest der Veranstaltung auch mit Java war. Wenn ich diesen Constraint nicht gehabt hätte, hätte ich vielleicht etwas anders gemacht… Vielleicht Python… Aber wenn jetzt im Wintersemester Java kommt, und die Leute die das nicht schaffen un dann zu mir kommen. Und die dann was anderes Lernen müssen. [Das] ist halt nicht so toll. Und…

Aber ganz allgemein, sehe ich es als problematisch an, gleich im ersten Semester da gleich mit großartigen objektorientierten Konzepten zu kommen. Man kann da bisschen in der Richtung was machen… Aber die Dinge, die wir da gemacht haben im Sommer, also bis vor das Wintersemester 2014/2015 halte ich für zu ambitioniert. Also da werden dann… Zwar möglicherweise wird das dann für die Leute, die von der HTL kommen okay sein. Aber der, der da frisch anfängt hat da wenig davon.

Yazicioglu Finden sie die Schüler haben besonders die Probleme bei Java… Bei Syntax, Semantik oder beim objektorientierten Konzept? Bei welchen Punkt haben die Schüler besonders Schwierigkeiten?

Lehrer 3 Bei allen… Also bei Syntax kommt mir vor noch am wenigsten, weil das sagt eh der Compiler wenn da was nicht passt. Semantik, also ist dann durchaus… Gab es dann… Hat es durchaus einige komische Sachen, die natürlich dann auch Schwierigkeiten gemacht haben. Und die dann… Wo wir dann gerade in Übungsaufgaben… Gerade in den Prüfungsaufgaben darauf abgestellt haben… Wo ich mir denke: Wenn man eine gescheite Programmiersprache… Also wenn die Programmiersprache diese Schwäche nicht hätte, dann müssten wir uns nicht so lange mit dem Zeug beschäftigen.

Yazicioglu Können sie mir ein Beispiel geben?

Lehrer 3 Ja, ein Beispiel ist bei Vergleichen…

Yazicioglu `=` bei Vergleichsoperatoren… `eq`… `else`…

(Pause des Interviews)

Yazicioglu …Jetzt sollte es gehen…

Lehrer 3 Also es gibt in Java die Basisdatentypen wie `int` zum Beispiel. Die vergleicht man mit == und dann gibt es eben Objekttypen, Referenz-Typen. Und wenn ich die mit == vergleiche, vergleiche ich nur die Referenzen, aber nicht die Objekte selber. Und für die Objekte selber muss ich dann mit `equals` vergleichen. Und das ist halt, so eine Sache die jetzt nicht… Also einerseits diese Trennung in diese Basistypen und die Objekttypen ist natürlich eine Komplikation.

Das ist die eine Sache und eben, dass man dann bei den Vergleichen unterschiedliche oder, das… Wenn ich eben den Inhalt vergleiche will im einem Fall eben ich == verwende will, was aber im anderen Fall ganz was anderes bedeutet. Ist halt…

Yazicioglu Okay

Lehrer 3 Das ist halt so eine typische… Eine Schwäche in Java.

Yazicioglu Okay. Dann denken sie, dass für diese Probleme oder allgemein spielt [die] Programmiersprache eine Rolle für die Motivation, Begeisterung, für den Erfolg der Schüler?

Lehrer 3 Vermutlich. Denke schon — Ja. Also wenn man irgendwie… Also [ich] mein [es] kommt natürlich auch auf die anderen Dinge an. Aber wenn man irgendwie was „Cooles“ macht, dann ist man sicher motivierter, als wenn man sich irgendwie mit… [Damit] herumplagt wie man das Ding, was sowieso eigentlich kein Problem sein sollte, dem Compiler beibringt. Also, aber…

Yazicioglu Okay. Ich glaub meine Fragen… Okay, welche Lernkonzepte funktionieren im Unterricht besonders gut? Ist die Verwendung von Analogien zum Beispiel aus [der] Mathematik ein gutes Mittel für den Unterricht?

Lehrer 3 Hm,…

Yazicioglu Spielt funktionale Programmierung, funktionale Paradigmen eine besonders gute Rolle? Zum Beispiel in Python kann man [Python als Taschenrechner] verwenden. Und es gibt soviele Bibliotheken und Methoden. [Mit Python, kann man] mathematische Berechnungen sehr gut lösen. Aber…

Lehrer 3 Also ich bin mir jetzt nicht… Nicht klar welche… Also, so richtig eine Antwort auf die Frage hab ich nicht. Aber was jetzt die Mathematik betrifft: Ich weiß jetzt nicht… Ich bin da jetzt nicht so überzeugt davon, dass man… Das das jetzt besonders hilfreich ist, da mit mathematischen Konzepten umzugehen.

Ein Artikel, den ich gelesen hab… Da haben die Autoren, also das sind die Leute die das Buch „How To Design Programs“ geschrieben haben, haben eben ihr Buch verglichen mit dem Buch „Structure and Interpretation of Computer Programs“. Was beides Einführungslehrbücher sind für Programmieren. Und [die] haben eben gesagt: Ja also die… Ein Fehler, den eben das „Structure and Interpretation of Computer Programs“ macht ist, dass sie da… also… eine häufige Verwendung von z.B. Schleifen ist es das man über irgendeine Struktur drüber-iteriert. Also, da kommt die Schleife sozusagen von den Daten her.

Yazicioglu Okay

Lehrer 3 Oder eben die Kontrollstruktur, spiegelt eben die Datenstruktur wieder. Und sie meinen, das ist eben eine Sache, die einerseits nicht nur häufig vorkommt, sondern auch relativ einfach zu lernen ist. Deswegen meinen sie, dass man das auch am Anfang lernen sollte. Oder am Anfang lehren sollte — ja.

Während… Und sie meinen, dass eben sie geben dann ein Beispiel, dass eben in „Structural Interpretation of Computer Programs“ vorkommt, wo halt irgendeine Schleife [steht], die eben irgendwie mit Zahlen arbeitet, und wo es dann… Wo man irgendeine mathematische Erkenntnis braucht, um zu wissen, dass die Schleife irgendwann mal terminiert. Und das die Schleife auch genau… Also das diese Schleife auch das tut was man denkt, was sie tut soll, und so also… Und sie meinen, dass das ein wesentlich schwieriges und eigentlich nicht so häufig gebrauchtes Programmierkonzept ist — solche Art von Schleifen, wo man dann eben wirklich irgendwie aus dem Anwendungsbereich, eben in dem Fall der Mathematik noch irgendwelche Erkenntnisse braucht um das Programm überhaupt schreiben zu können.

Und sie meinen eben, dass das nicht so sinnvoll ist das zu machen und mit sowas anzufangen. Sondern lieber eben mit dem: Ich hab eine Datenstruktur, was weis ich, eine verkettete Liste, oder ein Array und mach ebene eine Schleife über die verkettete Liste oder das Array. Das das eben sozusagen geschickter ist am Anfang, weil das kommt sozusagen dann aus den Daten.

Yazicioglu Okay, geschickter als…

Lehrer 3 Geschickter dann also sagen wir… Was hätten wir. Was wär das Andere… Eine Newton-Raphson Berechnung der Quadratwurzel, oder sowas die halt, wo man die mathematische Erkenntnis braucht, das man sich da dem Wert annähert.

Yazicioglu Okay, Danke schön.

Lehrer 4

Lehrer 4 …Python versucht in der ersten Klasse

Yazicioglu Okay, (Lehrer) hat auch gesagt, dass das das nicht so gut funktioniert.

Lehrer 4 Ja wir sind gemeinsam… Es ist… Ja und nein. Es ist Python sicher eine interessante Sprache und wenn man sich wie sie es hinten geschrieben haben, also. Warum sollten… Was sind die Auswahlkriterien für die Sprache. Ist natürlich Python toll weil es ist jetzt sehr im kommen Python.

Wenn man so schaut im Internet wird sehr viel damit gemacht. Es ist halt für Schüler am Anfang ist es etwas Anders. Man muss am Anfang irgendwas finden, wo es… Ein wichtiges Kriterium für mich ist etwas wo der Schüler ein Fenster hat, wo er den Text so hineinschreibt wie er ihn auf der Tafel sieht und dann einen „Run“-Knopf hat. Alles was mehr ist, ist für den Schüler am Anfang zu verwirrend.

Yazicioglu Okay. Das mach Python super…

Lehrer 4 Das gibt es in Python. Das gibt es aber auch in Java. Wir verwenden „Processing“ heißt das. Kennen Sie…

Yazicioglu Ja, ich…

Lehrer 4 Gut… Und damit ist das dann eher ziemlich gleich. Was wir glaub ich ein bisschen unterschätzt haben, oder ich unterschätzt habe ist, dass es zwar in Python heißt, es gibt dort zwar… Man muss keine Typen hinschreiben. Man braucht Variablen nicht vereinbaren. Das verleitet irgendwie dazu, dass man das gar nicht braucht. Es ist aber so, dass in Python sehr wohl jeder Ausdruck einen Typ hat. Und der Schüler dann doch verstehen muss, das es ganze Zahlen gibt, Komma-Zahlen, das es Strings gibt, das es Listen gibt.

Yazicioglu Manche finden das ist ein Nachteil, die dynamische Typisierung. Finden sie auch, dass [das] ein Nachteil [ist]?

Lehrer 4 Als Programmierer nicht, weil ich habe jetzt bei vielen Dingen gesehen, wenn es komplexere Type werden, wo ich dann in Java viel zu lange nachdenken müsste: Was ist das für ein Typ? [Ich] glaube wenn das ein Integer, ein String ist, ist es leicht hinzuschreiben. Aber wenn es eine Liste von komplexeren Dingen ist. Das ist in Java relative aufwendig zum Schreiben. In Python ergibt sich das.

Yazicioglu Es gibt immer viele Fehlermeldungen, wegen der Typumwandlung, Casting

Lehrer 4 Ich glaub nicht, dass die…

Yazicioglu Ah, nicht…

Lehrer 4 …weggehen, wenn man Python macht. Das ist nur…

Yazicioglu Später, nicht zur Compile-Zeit, sondern…

Lehrer 4 Ja sie können später auftreten, da sage ich aber, diese Gefahr, das das erst zur Laufzeit vor Ort irgendwann Auftritt ist, relative gering, wenn ich gut getestet habe. Dann sollte diese Fehler aller gehabt haben. Also so dieses irgendwann nach halben Jahr in Betrieb kommt, dann auf einmal ein ungültiger Type, das glaube ich nicht wirklich.

Yazicioglu Manche Lehrer finden das schlimm. Die geben [ein] starkes Argument gegen Python, weil später [werden], diese Fehler entdeckt. Es ist…

Lehrer 4 Das glaub ich… Ich programmiere schon lange, in verschiedensten Sprachen. Ich habe mich am Anfang auch… Wie Python [zum] erstmal gesehen habe, was aber jetzt auch schon lange her ist, [da] waren das auch meine Bedenken. Bis ich erstmal ein bisschen mehr damit programmiert habe. Und da habe ich gesehen, dass dieses, was ich alles nicht hinschreiben muss, und damit erspare, das das bei weitem überwiegt, dass ich da vielleicht hin und wieder ein paar Unsicherheit habe.

Es ist auch stark… Also wir beginnen mit Processing, machen dann Eclipse. [Da] ist diese Übergang sehr stark, was einem die Entwicklungsumgebung abnimmt. Und Eclipse verwöhnt da die Leute sehr stark. Weiß nicht ob sie das kennen…

Yazicioglu Ja…

Lehrer 4 Diese Vervollständigung und so weiter und das geht in Python nicht.

Yazicioglu [Das] Refactoring, [die] Vervollständigung gibt es in Python…

Lehrer 4 Das gibt es in Python…

Es beginnen jetzt die ersten Tools das auch zu können. Aber gerade so bei der Vervollständigung… Es gibt einige Dinge die kann man nicht ganz einfach [machen]… Weil ja Python-Code, wenn ich [da etwas suche, dass] dem ein zu eins [entspricht], was Java hat… Weil Python Code ist ja eher das, was in Java ein Generic ist, oder in C++ ein Template ist.

Naja, doch… In Abhängigkeit was ich übergebe, dann das richtige macht. Also ist es halt schwierig. Bei C++ Template z.B. verzweifeln auch die meisten Entwicklungsumgebungen.

Yazicioglu Ja, das kenn…

Lehrer 4 Es ist ganz einfach so, dass es nicht so genau festgelegt ist was es an der Stelle geben muss. Python beginnt ja jetzt… Mit Python 3 ergibt sich diese „Pypins"… Glaube schon, dass sich, dass bei zu einem gewissen Grad durchsetzen wird. Weil es ist doch für den Entwickler eine guter Hinweiß, was da ist… Es erspart Tipparbeit. Ist ja auch Faulheit irgendwo…

Und wo ich angefangen habe… Eigentlich ist es für Schüler sehr schwierig, weil wir diesen Begriff nicht so stark haben. In Java muss ich sie hindrängen: Da musst du `Integer`, da musst du `String` herschreiben… Am Anfang gibt es eh noch nicht mehr: `Integer`, `Double`, `String`.

Aber das muss sich der Schüler genau überlegen. In Python muss er [es] sich nicht überlegen. Damit wird der Schüler aber auch nachlässig. Dann wird es auch unklarer: Was kann ich da machen? Warum passt das nicht zusammen. In Java wird er da mit der Nase darauf gestossen. Das ist ein `Integer`, und das ist `String`… Und das ist halt anders. In Python: [Eine] Variable ist [eine] Variable. Und damit sieht er halt das man nicht…

Yazicioglu Okay. Das finden sie nicht gut, dass man am Anfang…

Lehrer 4 Das ist am Anfang…

Yazicioglu Was `Integer` ist, was `float` ist…

Lehrer 4 Das ist so implizites Wissen drinnen. Es ist immer schwierig für Leute die länger programmiert haben. Also, die größte Schwierigkeiten bei Beginner, beim Anfang des Programmieren sind zwei Dinge: Erstens unsere Schüler sind, unter Anführungszeichen noch recht klein.

Yazicioglu Das wollt ich auch fragen. Danke.

Lehrer 4 Wir beginnen in der ersten Klasse. Und ich unterrichte jetzt lang genug. Und ich habe in einer Elektrotechnik-[HTL] unterrichtet. Hab dort erlebt, wie programmieren… Eigentlich haben wir in der vierten erst programmiert. Da hatten wir in der dritten Klasse ein bisschen EDV.

Ich rede da jetzt vom Anfang der 90-er jahre. Da [hatten wir] in der dritten [Klasse] ein wenig EDV und in der vierten [Klasse] wurde programmiert.

Das ist langsam vor-gewandert. Und hier, da ist jetzt eine IT-Abteilung… Da [heißt] es in der ersten Klasse voll programmieren. Und ich habe den Eindruck das es für die Schüler, gerade so bei uns in der ersten klasse, diese Abstrahieren sehr schwierig ist.

Sie können konkrete Probleme zwar durchaus lösen, aber das dann so verallgemeinern: Wie muss ich da jetzt die Schleife allgemein machen? Wie weit muss ich da zählen? Wenn ich… Das konkrete Beispiel ist klar…

Da muss ich bis 10 zählen und das 10 mal machen. Aber das jetzt zu erkennen: Das ist die Länge oder zwei mal die Länge oder die halbe Länge und sowas. Das ist für die Schüler relativ schwierig.

Das ist aber unabhängig von der Programmiersprache.

Yazicioglu Ja, das wollte ich auch… Spielt da die Programmiersprache eine Rolle?

Lehrer 4 Nein, keine Rolle. Und das war auch so: Im Großen und Ganzem haben die Schüler, wie wir Python gemacht haben, gegenüber Java, sehr ähnliche Probleme gehabt.

Yazicioglu Aha, die Probleme werden nich [verringert]

Lehrer 4 Die Probleme werden nicht geringer, nicht größer. Weil im Wirklichkeit… Sicher am Anfang in der ersten Woche ist es schwierig: Da gehört eine Strichpunkt hin, da gehört eine Klammer hin, aber das haben die Schüler relativ rasch [heraus gefunden].

Also es ist… Ich sage mal in zweiten Semester überhaupt… [Da haben die Schüler] niemals die Schwierigkeit: Wo gehört eine Klammer, wo gehört eine Strichpunkt hin oder sowas. Das haben sie gleich. Weil, wir machen nicht so viel Dinge: Das ist ein `if`, das ist ein `for`, es gibt Unterprogramme… Das wissen Sie wie man [das] schreibt.

Yazicioglu Ja…

Lehrer 4 Da ist dann nicht mehr so: Die Syntax ist anders, ob ich es einrücke oder geschwungene Klammer machen muss… Für die Schüler ist es eher diese Logik…

Yazicioglu Haben Sie gesehen dass die Schüler einen schnelleren Einstieg [mit Python schaffen] als mit Java? Oder ist [das] auch fast gleich?

Fast gleich… Man,… Es ist immer schwierig dann zu vergleichen. Wir machen Processing, da kann man relativ rasch [was mit] Grafik machen. [Wir] machen so einfache Grafiken.

In Python haben wir V-Python, Visual Python. Da kann man sehr viele Dinge auch machen… Da haben wir eigentlich 3D-Grafiken gemacht während wir [in Processing] nur 2D-Grafiken haben.

Vom Aufwand ist es ähnlich. Es ist da die Bibliothek eine Spur besser. Da kann man auch mehr Dinge machen.

Yazicioglu Die Bibliotheken [sind] in Python viel besser?

Lehrer 4 In Python… Dieses Visual-Python… Weiß nicht ob sie das kennen. Da kann man wirklich… Da gibt es so Objekte, da gibt es Bälle, und also nicht… Normal gibt es in Visual gibt es halt Rechtecke und Kreise. Das [V-Python] ist das Ganze in 3D. Das hat wirklich Kugeln, Würfeln und so Dinge mehr.

Yazicioglu Okay

Lehrer 4 Was für Schüler eine Spur schwierig ist, weil es eine dritte Dimension gibt. Anderseits, ist es eine spur effektvoller, weil da hat man schöne Grafiken. Die [kann] man, ohne das man programmieren muss, drehen. 3D-mäßig von unten anschauen, usw. Also…

Yazicioglu Das hat…

Lehrer 4 Schaut besser aus, ob es jetzt wirklich für die Schüler besser ist, wenn sie in drei Dimensionen denken müssen… Für die Schüler ist es halt wirklich diese Logik, dieses Harte, was muss ich machen… Diesen Ablauf genau beschreiben… Und das ist eigentlich die Schwierigkeit.

Yazicioglu Viele Lehrer haben das gesagt…

Lehrer 4 Ich glaub halt, dass man dann… Insofern haben wir zu früh abgebrochen, weil wir haben bis Ostern oder so gemacht. Es wäre dann sicher interessant: Wie geht es weiter? Dann kommen die interessanten Dinge, die ich glaube, die in Python bei weitem einfacher sind als wie in Java.

[Also wenn] man dann mit Listen beginnt. Da in Java so viel Detail drinnen. Da gibt es `Hashmap`, `TreeMap` usw. Wo ich sagen muss: Will ich das wirklich wissen?

Yazicioglu Aha

Lehrer 4 Weil früher… Ich habe angefangen mit Visual Basic, und Assembler, C und C++. [Da] muss man sich halt um alles kümmern. In Java muss man sich um einige Dinge nicht mehr kümmern. Da gibt es eine Garbage-Collection, [um] gewisse andere Dinge muss ich mich noch kümmern: Zum Beispiel welche Implementierung einer Hash möcht ich verwenden oder sowas.

Lehrer 4 In Python ist es noch ein Spur drüber… Das sind so Dinge wie eine Hash-Map ein Dictionary… [Das] ist dort fix in der Sprache eingebaut. Und das ist damit so natürlich zu verwenden, dass ich gar nicht nachdenke: Gibt es da Varianten oder sowas. Natürlich [wenn] ich sage, Hoch-Performance und sehr schnell, da kann man was tunen… Da muss ich C++ oder Java vielleicht doch wieder nehmen. Aber für einen Anfänger… Der will sich um die Dinge ja gar nicht kümmern. Der versteht es ja eh nicht, was da

Yazicioglu Denken sie, dass ist eine Gefahr, das ist ein Nachteil? Weil Manche glauben die Schüler müssen am Anfang alles Wissen. [Damit sie nicht in] einer späteren Phase nocheinmal einer Hürde begegnen… Weil in der späteren Phase konfrontieren…

Lehrer 4 Ja, es ist immer die Frage…

Yazicioglu OOP, Objektorientierte Programmierung-Konzept [ist] noch eine Hürde

Lehrer 4 Ja, ja… Es ist halt die Frage, muss man heutzutage noch verstehen wie eine CPU funktioniert.

Yazicioglu Wie eine?

Lehrer 4 Wie eine CPU wirklich… [Die] Grundvoraussetzung ist Programmieren. Und ich glaube nicht mehr… Früher war das so. C war nahe genug, dass man halt verstanden hat, wenn ich C schreibe, was macht die CPU.

In Java… Ja, es ist auf die Virtuelle-Maschine noch etwas anwendbar… Aber eigentlich, auch schon teilweise sehr weit weg. Und in Python sind einige Dinge… Wo man es dann klarerweise nicht mehr abschätzen kann wieviel Aufwand das ist: Ich schreibe eine Zeile hin… Was passiert das wirklich? Wieviel, wie gewaltig ist diese Operation? Und dann wenn ich mit Listen operiere… Die Listen hat 10 000 oder 100 000 Einträge… Was bedeutet das wirklich für die Maschine? Andererseits… Sie haben acht Gigabyte… Warum soll ich mich darum kümmern — ja.

Yazicioglu Ja, Da bin ich auch der Meinung…

Lehrer 4 Früher war [es] 64k. Da musste man sich um Speicher kümmern. Aber heutzutage werd ich mich um solche Dinge nicht mehr kümmern müssen.

Ich glaube schon, dass man, also… In Summe in Python dann auf Grund dieser Dinge wo ich mich nicht kümmern muss besser programmieren kann. Größere Anwendungen…

Yazicioglu Also das sehen sie als Vorteil

Lehrer 4 Das sehe ich so, ja…

Yazicioglu Und die Schüler, später… Mit dem was Java hat: `HashMap`, `ArrayList`, `LinkedList`, all diese Sachen… Haben die Schüler große Probleme [damit] nach Python? Oder können sie gleich…

Lehrer 4 Sie haben sehr viele Dinge… Also wo wir gewechselt haben… Es ist [der] Wechsel immer ein Horror. Ich glaube, es ist eine schlechte Idee mehrere Programmiersprache zu machen am Anfang. Bei uns ist es halt so, dass wir in den ersten 3 Jahren Java machen… Wobei sie natürliche nebenbei, bei der Medien-Technik bei uns, JavaScript machen müssen und PHP gemacht wird.

Wir machen bei der Netzwerktechnik dann in der vierten Klasse wirklich Python. Und zwar tief hinein Python… Schwierigere Dinge mit Dictionaries und so Dinge. Und da ist es dann kein Problem mehr. Es ist nur für Schüler am Anfang, solange man noch nicht so sattelfest im Programmieren ist… Es ist dann der Wechsel schwierig. Denn dann kommt wieder diese Frage: Wo kommt der Strichpunkt hin? In Java muss ich da nichts hinschreiben, in Python muss ich da jetzt was herschreiben. Da brauche ich Klammern, da brauche ich keine Klammern. Das ist dann schon. Wenn es ein Schema gibt, ist es für die Schüler halbwegs einsichtig.

Yazicioglu Okay, sie finden, dass der Umstieg dann…

Lehrer 4 Umstieg ist eine schlechte Idee

Yazicioglu Okay. Dann… Warum würden sie Java auswählen/bevorzugen

Lehrer 4 Ich würde eher… Also ich glaube eher ich würde Python bevorzugen inzwischen…

Yazicioglu Aha

Lehrer 4 Mir ist zwar klar, dass es einige Nachteile gibt. Wie ich vorher gesagt habe. Das ist aber sicher… Man muss jetzt sozusagen… Wir unterrichten ja hier Java seit Jahren. Es ist ja am Anfang, nicht so super unterrichtet worden, wie es jetzt ist.

Gewisse Dinge kennt man, man weiß welche Fehler die Schüler machen. Das hat natürlich in Python gefehlt. Ich tu mir viel leichter, wenn ich das schon das fünfte-mal mache, das zehnte-mal mache… Dann weiss ich, auf welchen Beispiel ich achten muss, wo es Schwierigkeiten für die Schüler gibt und so.

Beim ersten mal ist das ein bisschen schwieriger. Ich glaube da haben wir, gerade in dieser Richtung… Das [wir] diese Typen vernachlässigt haben, da einen Fehler gemacht beim Unterrichten.

Yazicioglu In späteren Phasen…

Lehrer 4 Ja, das man also… Von Anfang an, trotzdem so wie man es in Java macht, sagt: Es gibt `Integer`, das ist eine `Integer`-Variable… Auch wenn ich das nicht hinschreiben muss, muss ich das trotzdem viel stärker betonen, dass es dem Schüler klar wird.

Yazicioglu Okay. Sie würden Python wählen.

Lehrer 4 Ja. Es glaub es wär in Summe heutzutage und auch man muss sehen: Was sollen unsere Schüler programmieren? Wo sollen sie eingesetzt werden.

Yazicioglu Ja. Was ist das Ziel…

Lehrer 4 Wir haben ja hier Medientechnik und Netzwerktechniker… Wenn man jetzt im Gegensatz nimmt. Bei uns haben wir auch die Abteilung Mechatronik: Die bauen so kleine Computer und Embedded Systems, usw. Dort jedenfalls würde ich irgendwas C Ähnliches oder gleich C. Weil dort brauch ich sowas. Beim Micro-Controller-Programmieren, da kann ich nicht… Also bis auf einen… Was inzwischen auch… Aber solche Dinge muss ich in C programmieren. Also sollten sie sowas können.

Yazicioglu Ja…

Lehrer 4 Es ist gerade der Umstieg von Python gerade wieder runter… Und da sind auch so… Die Mechatroniker sind ja keine Programmierer in dem Sinn… Da ist ja… Für die ist das ein Mittel zum Zweck.

Dort jetzt in Python anfangen und dann wechseln, dann wird es total schwierig, wenn man ganz wenig Programmieren macht. Also dort bin ich auf jeden Fall dabei und sage C, oder…

Yazicioglu Okay…

Lehrer 4 Bei uns muss ich ja wirklich die Frage stellen: Wenn ich jetzt wirklich perfekt Java kann. Was… Wir sind jetzt auch nicht so sehr eine Programmierschule. Die Sprenger-Gasse… Ich weiß nicht ob sie dort waren…

Yazicioglu Ja…

Lehrer 4 Die sind ja viel stärker mit dem Programmieren. Die haben ja dreimal soviele Programmierstunden wie wir. Wir haben zwei, drei Stunden… Also zwei Stunden Übung, eine Stunde Theorie, vier Jahrgänge [lang]. Und in der fünften nur mehr zwei Stunden. Nicht soviel…

Yazicioglu Ja. Ich war dort. Aber mit einem Lehrer hab ich nicht gesprochen

Lehrer 4 Also da ist schon ein bisschen ein Unterschied. Also [bei uns sollen] nicht so sehr, echte Programmierer herauskommen… Wir machen Netzwerktechnik oder Medientechnik. Wo das [Programmieren] Hilfsmittel ist.

Gerade in diesen Bereichen glaub ich ist Python sehr stark.

Yazicioglu Okay…

Lehrer 4 Also Netzwerkprogrammierung, diese ganzen Scripts… Wenn man sich Linux anschaut… Diese ganzen Linux-Verwaltungstools werden sehr viel in Python geschrieben in der letzten Zeit oder [sind] schon in Python geschrieben. Und auch Medientechnik, so Frameworks gibt es auch sehr viele in Python. Also das ist glaub ich schon…

Yazicioglu Da finden sie [Python ist] eine gute Wahl, sehr gute Wahl weil… Skriptsprache Python gibt es… Aber wenn man Anwendungsprogrammierer sein will… Dann ist es ganz… Ganz tief hineingehen will…

Lehrer 4 Dann, Ja…

Yazicioglu Dann auch…

Lehrer 4 Aber das sind so eher diese Leute wie die Mechatroniker, die wirklich einen Micro-Controller programmieren, die irgendwo im Kernel programmieren wollen… Ja, [die] haben dann eh keine Chance, irgendwo CI… Ich seh ja der Java-Markt… Ist ja für mich auch ein sehr… Erstens ist er sehr diversitär… Es gibt verschiedene… Aber in Wirklichkeit ist er ja zusammengebrochen… Ich glaube wenn es derzeit kein Android gebe wär auch der Java-Markt schon sehr klein…

Yazicioglu Ja, weil App-Applikation [sind] sehr mächtig

Lehrer 4 Das ist ja sehr wichtig wieder. Aber auch dort gibt es ja auch Alternativen. Es gibt einige wo man vielleicht auch drüber hinweg kommt. Es versuchen einige plattformunabhängig zu programmieren. Weil es jetzt ja doch auch für eine Firma ein riesiger Aufwand, da jetzt Android, und Windows, und Mac, also iOS zu machen. Also das ist total schwierig. Wenn ich als Firma etwas anbieten möchte, und muss das drei-, vier-, fünfmal programmieren.

Yazicioglu Ja… Von der Literatur, was ich gefunden habe ist die… Mit Python kann man sehr gut Spiele programmieren. Es ist sehr gut am Anfang, für die Motivation der Schüler. Spielerisches Programmieren… Da ist es sehr gut geeignet. Wenn sie Java und Python vergleichen, dann…

Lehrer 4 Wenn man es sich bei uns anschaut, also wenn man dieses… Da darf ich ja nicht Java mit Python vergleichen um sie jetzt sozusagen… Wir machen da Processing und wir haben, Visual-Python gemacht. Und da ist nicht viel Unterschied

Yazicioglu Okay. An der TU Wien wird auch Processing. [Ich] hab mit einem Lehrer gesprochen. Er unterrichtet Processing. Er hat gesagt, er ist ganz zufrieden und erfolgreich. Sehr gut… [Das] versteckte Konzept finden sich nicht [das] ist ein Nachteil? Versteckte Typisierung

Lehrer 4 Ich sehe dort nicht das Versteckte im Vordergrund, ich sehe mehr die Ablenkung im Vordergrund. Wenn man Eclipse nimmt… Um in Eclipse das erste Programm zu schreiben… Wenn ich so vorgehe… Ich habe die Klasse das erste mal. Da erzähle ich denen an der Tafel ein kurzes Programm. Ich lese zwei Zahlen ein, bilde die Summe, gebe die Summe wieder aus. Das kann ich auf die Tafel noch halbwegs hinschreiben.

Und jetzt sollen die Schüler das umsetzen. In Eclipse muss ich ein neues Projekt machen, muss eine neue Klasse machen, und ich muss das da hineinschreiben. Und das was ich erzählt habe, [im Vergleich] zudem was jetzt da wirklich da ist, ist ein großer Unterschied.

In Processing schaffe ich es. Da ist… Ich starte das Processing. Es ist ein Fenster da. Da scheibe ich diese zehn Zeilen, fünf Zeilen hinein und es gibt ein Knopf wo „Run“ steht. Das ist für den Schüler nachvollziehbar.

Yazicioglu Gut. Und… Manche Ausbildungsbereiche wollen die Sprache wählen, [die] für zukünftige Berufsmöglichkeit gute…

Lehrer 4 Das ist in Python glaub ich auch gegeben. Bei Netzwerktechnikern, bei Medientechnikern ist Python auch. Es ist halt unter Anführungszeichen blöd, das sehr viele Sprachen aus der Ecke kommen und eine sehr ähnliche Syntax wie C haben. Zum Beispiel [die] `for`-Schleife bei Python genau dort [ist es anders]. Das ist für Schüler auch schwierig. Python macht es so… Alle Anderen machen es anders.

Yazicioglu Dann ist es schwierig…

Lehrer 4 Andererseits ist es in Java, ist ja diese foreach-Schleife ist ja auch anders. Also… Ich glaub ja sehr viel ist da in den Köpfen: Ich hab das immer schon so gemacht… Man muss ja auch in die Zukunft sehen. Es wird ja nicht bei dem bleiben… Wo werden wir in fünf Jahren sein… Und wenn ich jetzt einen Schüler unterrichte in der ersten Klasse ist es ja schwierig. Jetzt sind die bei uns fünf Jahre, dann müssen sie zum Bundesheer, dann machen sie ihren ersten Job.

Yazicioglu Mhm

Lehrer 4 Also ich muss ja praktisch jetzt unterrichten, was der in fünf Jahren brauchen wird.

Yazicioglu Ja wer weis das…

Lehrer 4 Genau. Wer weis das. Sie wissen es nicht. Ich weis es nicht. Es ist sehr schwierig.

Yazicioglu Spielt ein Programmiersprache selbst eine Rolle bei der Motivation, Demotivation, Erfolg und Begeisterung von den Schüler?

Lehrer 4 Das spielt eher die Rolle, dass man relative rasch… Gerade in ersten Klassen, rasch etwas effektvolles hat. Also nur so im Text-Modus zwei zahlen addieren… Ist zwar recht nett und schön, aber für die Schüler nicht befriedigend. Wenn man da gleich Grafiken machen kann… Wenn man relativ rasch bewegte Grafiken hat, dann ist es für die Schüler schon viel motivierender.

Yazicioglu Mit Python, kann man das sehr wohl…

Lehrer 4 Kann man das auch,ja, ja… Und was jetzt natürlich auch… Sage das ist jetzt mehr vom Unterricht her… Es ginge z.B. auch In JavaScript heutzutage. Da kann ich in jedem Browser… Das wäre großer Vorteil von JavaScript. Weil jeder hat einen Browser, wo ich das machen kann. Nur JavaScript ist im Bezug auf das Programmieren eine schreckliche Sprache. Weil entweder es geht, dann sehe ich relativ rasch etwas. Aber wenn es nicht geht, hat man keine Methoden, keine Möglichkeiten hineinzuschauen und irgendwas zu machen.

Yazicioglu [Das] Debuggen [in JavaScript] finde ich sehr schlecht…

Lehrer 4 Ja genau. Sehr schlecht, ist noch sehr nett formuliert.

Yazicioglu Ja am Anfang hatte ich so viele Schwierigkeiten. Wo finde ich meine Fehler? Wo suche ich

Lehrer 4 Das muss natürlich für einen Lehrer… Da ist ja das noch verstärkt: Ich habe ein Program, wo ich jetzt programmiere alleine… Aber ich habe da… Ich meine es sind zwei Lehrern bei 30 Schülern. Ich muss jetzt bei 15 Schüler verstehen, was nicht funktioniert. Das heißt ich habe nicht mal die Zeit mich lange hinzusetzen. Und es sollte ja so sein, dass ich auf einem Blick sehe… Wenn es schon der Schüler nicht sieht. Aber in JavaScript habe ich keine Möglichkeiten… Ich noch als Lehrer das zu sehen, der doch mehr versteht…

Yazicioglu Sie haben gesagt sie haben Projekte gemacht. Und wie war der… Mit Python haben sie es probiert. Wie war das Feedback von [den] Schülern? Hatten Schüler früher… Java gehabt oder…

Lehrer 4 Nein, die sind dann. Die kommen aus einem Gymnasium oder Hauptschule. Die haben vorher noch nie programmiert. Die sind noch sozusagen unbedarft.

Yazicioglu Okay. Und wie war [das] Feedback? Oder gab es kein Feedback?

Lehrer 4 Das könnte man nicht so sagen, dass die Schüler gesagt haben… Die Schüler haben jetzt auch nicht den Vergleich. Die haben Python gemacht… Die wissen ja nicht wie es vorher war. [Bis auf] einige [Schüler die eine Klasse wiederholen] die dann vielleicht auch meistens im Programmieren durchgefallen sind…

Yazicioglu Wie ist [der] Unterricht? Wie ist der Lehrstil. Zuerst… Unterrichten sie mit…

Lehrer 4 Wir haben also so…

Yazicioglu Beispiel. Oder wie ist das didaktisch… ihre didaktische Stilart?

Lehrer 4 Wir haben eine Stunde Theorie, zwei Stunden Übung und… Es ist ja nicht soviel Stoff sonst in der ersten Klasse. Da werden Variable, `if`, Schleife und gegen Ende Strings und Arrays…

Yazicioglu Mhm

Lehrer 4 Das ist nicht viel. Also da nur relativ wenig Mathe, Theorieblöcke wo man jetzt wirklich eine Stunde was erzählt. Weil dann… Jedes dieser Kapitel kann man eigentlich in einer Stunde, jetzt genug erzählen, wie es die Schüler brauchen in der ersten Klasse. Es ist dann viel mehr, dass man viel mehr Beispiele durchgeht… Diese Logik durchgeht. Wie komm ich da… Diese Standardmuster… Wie bilde ich eine Summe? Wie gehe ich bei einem Array überhaupt durch? Wie gehe ich bei einem String alle Elemente durch. Wenn ich suche oder so… Und das muss man mit den Schülern schon mehrfach machen.

Yazicioglu Im ersten Jahrgang, welche Sprache… Welche Konzepte unterrichten sie? Gibt es kein OOP, sondern nur…

Lehrer 4 Nein, überhaupt nicht. Nein. Also OOP kommt vor bei… Weil die String-Methoden natürlich mit dem Punkt drinnen sind… Aber das ist für die Schüler aber eh auch kein… Das ist eine Art wie man das schreibt. Mehr ist es ja für die Schüler nicht.

Yazicioglu Okay. Eigentlich, meine Fragen sind fertig. Aber ich wollte um zusammenzufassen nocheinmal ganz kurz fragen: Sie wollen Python… Weil mit Python haben sie keinen Overhead, bei der Entwicklungsumgebung zum Beispiel. Auch bei den Sprachelementen selbst. Und…

Lehrer 4 Ich glaube, dass es dann später besser wird. Es ist aber dann… Es sind ziemlich alle Sprachen gleich. Weil die Schüler ja mit anderen Dingen kämpfen als mit der Syntax. Aber es wird dann wenn es um Arrays, um Dictionaries, usw. Und überlegt: Was ich in C programmiere… Welcher Aufwand ist es ein Dictionary zu machen, oder ein dynamische Liste, die dynamisch wächst. Das ist ja…

Yazicioglu Genau. Das ist ein sehr kritischer Punkt für die anderen Lehrer. Sie sagen, dass in der späteren Phase: Es ist so schlimm Python, weil die Schüler haben sich schon gewöhnt… Warum soll ich `Integer`, warum `String`… Alles haben sie wie „Müll“ in eine Box hineingeworfen. Aber später beginnen sie… Sehen sie das als Hürde oder… Das hab ich einmal gefragt, aber das wollte ich… Entschuldigung…

Lehrer 4 Nein, nein, das passt schon. Es ist immer die Frage, was wird die Schüler einmal machen. Wenn er in so einem Bereich wie Netzwerktechnik oder Medientechnik bleibt… Das sind ja nicht Programmierer in dem Sinn sind, die riesig große Programme schreiben… Das wird immer beschränkt bleiben auf relative kleine Dinge, die die machen.

Yazicioglu Dann ist Python…

Lehrer 4 Da ist Python auf jeden Fall besser.

Yazicioglu Okay. Das finde ich auch. Weil… Bei Skriptsprachen… Bei Systemsprachen…

Lehrer 4 Ich seh Python auch nicht mehr so sehr als Skriptsprache. Das ist nur ein Detail. Das ist zwar nett, dass man es machen kann. Wahrscheinlich ist es von mir aus der Entwicklung heraus, weil ich ja mit kompilierten Sprachen gearbeitet habe. Darum bin ich es nicht so gewöhnt, das ich was ausprobiere. Wirklich auf der Kommandozeile.

Yazicioglu Mhm

Lehrer 4 [Diesen] Vorteil hat man in Python. Ansonsten… Es gibt auch Python-Compiler inzwischen… Also so ist es nicht.

Yazicioglu Okay. In der Firma, in der Industrie wird meistens Java, C++ verwendet. Wenn die Schüler später mit Java was programmieren sollen…

Lehrer 4 Es gibt auch viele Firmen schon, die mit Python arbeiten. Ja… Wobei… Es ist dann immer schwierig ein was, wieviel gibt es zu dem Thema. Das ist bei Python glaub ich aber auch schlecht. Ich hab jetzt unlängst erst mit jemanden geplaudert… Sie suchen keine Python-Programmierer, weil sie wissen, das sie keine [bekommen] am Markt. Wenn sie dort hin schreiben: Wir suchen einen Python-Programmieren, dann schränkt das zu sehr ein… Weil sich keiner meldet, weil es Keinen gibt, oder Wenige gibt. Also sie schreiben hin Programmierer… Und schreiben dann „was“. Und schreiben auch Python dazu… Das heißt, wenn jemand Python machen will, dann fühlt er sich [angesprochen], wenn jemand Java oder C++ kann sieht er sich in der Annonce genauso. Das er dann das meiste [in] Python programmieren wird ist durchaus…machbar.

Yazicioglu Ich hab gestern mit einem Lehrer in [einer] Klasse. Er hat… Er unterrichtet Java und C#. Und er hat gesagt, es gibt so große Plattformen, wo C#, Visual Studio, Eclipse und IntelliJ IDEA… Sie haben [es] probiert… Einmal mit Python zu programmieren. Aber er hat gesagt: Später haben die Schüler wirklich so starke Probleme gehabt. Sie fragen immer nach: Warum `Integer`, warum `String`… Die Plattform ist sehr mächtig, und Python-Entwicklungsumgebung und finden sie so… Ganz… schwach. Python finden sie in der späteren Phase schwach. Das finde ich nicht. Wenn man eine Programmiersprache kennt, dann sollte `Integer`, `String` oder `float` überhaupt kein Problem sein.

Lehrer 4 Ja ich seh das inzwischen… Also… An sehr vielen Dingen… Wenn man sich anschaut. Ich hab jetzt auch ein paar so Online-Kurse gemacht. Einen zu Artificial-Intelligence, und da waren die ganzen Zufallszahlen dabei. Und der hat in Python diese Suchalgorithmen gezeigt. Und das war das erste mal wo ich gesehen hab: Ja genauso muss man das erklären…

Weil da ist im Prinzip, ob ich Tiefensuche mache, oder Breitensuche oder so… Der hat das so gemacht… Der hat eine Liste von noch zu besuchenden Knoten, die in einer gewissen Reihenfolge abgearbeitet werden. Und das kann ich in Python machen. Weil da gibt es eine Liste… Das ist ein natürliches Element. Ich wüsste in Java nicht, was für ein Typ ist diese Liste. Was schreib ich dorthin. Was ist der nächste Knoten, der zu besuchen ist. Was ist das für ein Typ in Java…

Yazicioglu Sehr gutes Beispiel. Danke, das werd ich schreiben. Als Zitat…

Lehrer 4 Dann war es so. Wenn man die zwei Algorithmen nebeneinander, also den Tiefensuche, Breitensuche und A*. Wenn man die Drei nebeneinander stellt. Dann hat man gesehen, dass eine Zeile Unterschied… In Java ist da soviel Rauschen drumherum, das man das gar nicht sieht. Das da nur ein Kleinigkeit verschieden ist. Ja. Und der hat dann gezeigt, dass er Tiefensuche, Breitensuche. Der Unterschied ist eben, das man… Ob es ein Stack oder ein FIFO ist. Das sieht man aber nur in Python, dann. Das kann man in Java gar nicht mehr so formulieren.

Yazicioglu Im ersten Jahrgang, der Algorithmus…

Lehrer 4 Das war so ein Online-Kurs auf… Ich weiß nicht…

Yazicioglu Unterrichten sie aber diese…

Lehrer 4 Wir machen relative wenig Algorithmik.

Yazicioglu Weil Python ist, ähnlich wie Pseudocode…

Lehrer 4 Ja, ja. Das ist richtig. Das schaut fast so aus wie der Pseudocode. Da ist wenig Overhead.

Yazicioglu Okay

Lehrer 4 Und das lenkt dann doch ab finde ich.

Yazicioglu Danke schön

Lehrer 5

Lehrer 5 Es gibt unterschiedliche Aspekte jetzt von der Programmierung. Einerseits die Frage wenn man da nochmal nach oben geht. Welches wäre die beste erste Programmiersprache. Da gibts Umfragen, die ganz klar dahin zielen, dass Python eigentlich geeignet ist. Warum das wir sich im Interview noch ein bisschen lichten. Generell welche Sprache man in der Programmierung lernen kann…

Das ist eben das mit dem Stroustrup?… Wie er es gesagt hat… Ob man überhaupt programmieren lernen sollte, das ist praktisch eine Frage, die zuerst kommt. Soll man überhaupt programmieren lernen ja. Da hat Steve Jobs eigentlich schöne Aussage getätigt. Da gibt es auch das Video drinnen… Und da gibts andere wie Zuckerberg und Co. die dazu gesagt haben… Stroustrup hat gesagt… „Jeder sollte eigentlich Programmieren lernen, damit er versteht, wie ein Computer funktioniert."

Und mit dem was er vor sich sieht. Für die Stunden am Tag eigentlich umzugehen. Ja. Und das sind sozusagen diese Aspekte…

Yazicioglu Okay. Und sie haben gesagt es gibt fünf wichtige Programmiersprachen. Eine traditionelle, andere…

Lehrer 5 Genau. Es gibt insgesamt… Die Frage welche Sprache sollte man lernen… Da gibt es halt dann unterschiedliche Sichtweißen. Sprachen wie C++ oder Java sind natürlich sehr wichtige und weitverbreitete Sprachen.

Daneben sollte man aber auch zumindest eine Script-Sprache beherrschen. Sei es jetzt JavaScript, Python oder Ruby. Und es ist auch nicht schlecht, wenn man wirklich mit Low-Level Sprachen arbeitet, wie z.B Fortran. Das man einfach sieht: Unterschiedliche Zugänge. Jeder Sprache hat für ihren eigentlichen Gebrauch Vorteile.

Yazicioglu Sollte man [eine] Script-Sprache lernen? Das wollte ich auch fragen… Welche Vorteile…

Lehrer 5 Zum Beispiel, wenn ich jetzt eine Script-Sprache wie Python lerne. Wenn ich jetzt sehr schnell kleine Programmteile schreiben möchte um irgendetwas zu automatisieren… Da habe [ich es] bei Code in Python leicht. Da gibt es auch Beispiele im Internet… Vergleich Python und C++ oder C#… Ein und das selbe Programm hat in Python vielleicht 5 Zeilen. [Um] exakt das gleiche in C++ zu machen brauchen sie vielleicht 20 Zeilen.

Da gibts Vergleiche dazu. Das heisst, Python hat wesentlich weniger Overhead. Das ist einer der Vorteile. Sehr schnell… Sie kriegen sehr viele Bibliotheken in Python. Sie können ja nicht nur was programmieren, sondern sie können Bilder einlesen. Sie können Bilder produzieren. Sie können Statistik machen. Sie können numerische Berechnungen machen. In unterschiedlichsten Bereichen… Datenbank etc. Das heißt [das] alles haben sie in Python. Alles haben sie in C++ direkt nicht.

Das heisst, Python ist dort sehr gut wo sie schnell einen Prototyp machen wollen. Wenn sie aber in ein umfangreiches Projekte gehen. Das heißt eine große System, das wirklich stabil… Datenbank… Schnell sein soll… Dann ist Python die nicht richtige Sprache.

Dann sie müssen einfach wechseln in die Dinge wie z.B C++. Warum? Weil C++ einfach wesentlich sauberer programmiert ist. Das heisst, z.B. bei der Variablen-Deklaration. Python kennt keine Variablen-Deklaration im klassischen Sinn. Eine Variable wird dynamisch deklariert. Je nachdem was sie zuweisen.

In C++ müssen Sie sagen, das ist eine `Integer-Zahl`, und dann ist es eine `Integer`-Zahl und irgendwann wenn ein String gesetzt wird auf die `Integer`-Zahl, dann bricht das Ding ab. Bei Python merken sie gar nicht, dass das passiert.

Yazicioglu Ja. Manche Python-Gegner sagen, dass dynamische Typisierung ein Nachteil ist.

Lehrer 5 Ist ein Nachteil. Ja, das ist ein klarer Nachteil von Python. Der Vorteil des schnellen Prototyping ohne viel Overhead, kaufe ich mir einfach mit dem Nachteil, dass ich dynamisch typisiere. Sonst wäre das nicht möglich.

Oder genauso wenn man anschaut, wann ich in C++ eine Variable um-cast-en möchte. Was das für eine Riesen-Wurst an Schreiberei… Das ich eine Variable von einem Typ in den Anderen kriege. In Python ist das eine Zeile… Einmal `a = 5` und einmal `a =` ein String, Punkt fertig. Dieser ganze Cast funktioniert im Hintergrund.

Yazicioglu Es ist für große Projekte ein Nachteil. Für den Anfang sehen Sie da auch einen Nachteil, oder…

Lehrer 5 Die Frage ist, was möchte man den Leuten am Anfang beibringen? Und da… Ich finde da kann man auf diese Dinge jetzt einmal verzichten… Weil man muss Leute am Anfang generell mal sagen: Warum soll ich überhaupt programmieren? Als Informatikerin ist ihnen das hundert-prozentig klar, aber erklären Sie das z.B. einem Mediziner.

Warum soll einer Mediziner oder auch einer HTL-er, warum soll der Programmieren lernen? Der zeichnet… Der macht Berechnungen mit einem Programm oder was auch immer… Warum soll der Programmieren lernen? Das müssen sie… Und da muss man eben die grundlegende Konzepte einfach… Wie Fallunterscheidungen oder `for`-Loops, automatisiertes Einlesen/Rausschreiben… Vielleicht auch Kommunikation mit dem Betriebsystem zeigen… Aber nicht sozusagen die Feinheiten der Programmiersprache in der Typisierung zeigen.

Yazicioglu Super. [Die] Programmierung ist nicht nur für die Informatiker. Denken sie Alle… Für…

Lehrer 5 Es [sind] auch HTL-er. Sie denken wahrscheinlich auch an „AHS-ler". Einer der in die AHS geht, der ist natürlich nicht notwendigerweise Techniker… Und wahrscheinlich kein Informatiker, im Vergleich zu der Masse… Es wird ein Promille wahrscheinlich Informatiker werden, ein Prozent oder was auch immer. Aber viele werden in andere Berufe gehen.

Die haben mit Programmieren direkt nicht zu tun. Und wenn ich jetzt sage: Die dritte Schule mit 15 Jahren: Wer weiss schon mit 15 Jahren, was er mal beruflich macht.

Yazicioglu Sie denken für das. Die Schüler sollen von Anfang an ein bisschen Ahnung von Programmieren haben.

Lehrer 5 Ich glaube sie sollten von Anfang an einfach verstehen, wie ein Computer sozusagen… Eigentlich aufgebaut ist und funktioniert… Das das keine Zauber-Kiste ist, sondern das dahinter einfach Algorithmen laufen. Wenn ich irgendwo drauf klicke, wird eine Funktion gestartet, die irgendwas macht… Und ein Computer ist keine intelligentes Wesen, das irgendwie erkennt… Sondern man hat einfach nur die Information intelligent verarbeitet… Aber um das zu verstehen, muss man programmieren können.

Das heißt: Wenn ich jetzt wohin klicke, und es kommt ein Pop-Up, möchten Sie weiter machen: Ja/Nein. Dann weiß ich wenn ich eine `switch`, `case` kennen gelernt habe… Das ist nichts anderes als eine Fallunterscheidung. Und da gibt irgendein Programm… Das steht: `if`/`else`… Wenn ok macht das… Wenn nicht ok dann mach das. Und da ist jetzt, wenn ich das nie gesehen habe, wie es eigentlich funktioniert.

Zum Beispiel das Schöne ist in Python… In Python kann ich eben auch Systembefehle abschicken. Das kann ich in C++ nicht so einfach. Das heisst ich kann z.B. Files lesen oder ich kann irgendwelche Verzeichnis erstellen. Oder ich kann mir jetzt wirklich Listen von Inhalten von Verzeichnis anzeigen lassen mit sehr einfachen Dingen. Und da verstehe ich vielleicht, wenn ich einen File-Browser aufmache: Wie kommt der auf die Liste der Files.

Das das nichts anders ist… Wie dieses grafisches User Interface fragt, einfach das Betriebsystem: Naja welche Files sind denen in diesem Ordner. Zeigt das dann sozusagen einfach an, und das man dann versteht: Im Hintergrund ist ein Filesystem und das kann bedienen ohne grafisches User Interface. Das heißt wenn ich einen Command-Prompt aufmache… Sehe ich eigentlich wirklich so wie wir es vor fünfundzwanzig Jahren gesehen haben DOS oder Unix… Eigentlich das echte Betriebsystem, das dahinter läuft. Das immer noch dahinter läuft.

Nur wird das über eine grafisches User Interface sozusagen überblendet. Aber im Hintergrund läuft da trotzdem noch DOS oder Unix, bei den Unix-Maschinen, oder Unix bei den Mac-Maschinen. Das ist Linux was da beim Mac läuft, das ist kein Mac OS. Mac OS ist nur die grafische Benutzeroberfläche.

Yazicioglu Ein Vorteil von Script-Sprachen…

Lehrer 5 Der Vorteil von Script-Sprachen ist einfach, dass man sehr schnell mit dem Computer kommunizieren kann und vor allem Prozesse automatisieren kann.

Yazicioglu Danke. Und warum haben Sie…

Lehrer 5 Sie gehen auf ihre Liste…

Yazicioglu Ja. Aus diesen Gründen haben sie Python für den Unterricht ausgewählt… Andere Gründe…

Lehrer 5 Ja, also unser Unterricht, muss man schon sagen. Wir sind eine Universität und wir unterscheiden natürlich von jenen, die natürlich das erste mal programmieren lernen sollen. Weil die Leute sind bei uns… 20 Jahre… Also was meinen sie jetzt… Auf die fünfzehn-jährigen bezogen oder auf die 20-jährigen bezogen?

Yazicioglu Für mich ist es auf die fünfzehn-jährigen bezogen. Aber [die] generelle Information…

Lehrer 5 Generell haben wir hier an der Uni aus dem Grund von Java auf Python gewechselt, weil die wissenschaftliche Script-Sprache Python ist. Das heisst: Wenn sie in die Wissenschaft gehen und sie gehen z.B in unserem Bereich… In den Bereich der Simulation, der Visualisierung, der Bildbearbeitung… Überall dort finden sie sozusagen Software-Pakete, die sie mit Python skripten können.

Das heisst, alles was sie mit einem Maus klicken… Können Sie mit Python-Commands abschicken. Und da gibt es eine Vielzahl von Sprachen, eine Vielzahl von Applikationen, die das machen. Das heisst, das war ein Grund. Der zweite Grund war: Es ist sehr MATLAB ähnlich. Das heißt wenn man sich MATLAB anschaut… MATLAB ist ja auch sehr eine gängige Programmiersprache. Nur das Probleme mit MATLAB ist halt, dass es kostenpflichtig ist. Und auch wenn es nur wenige Lizenzgebühren sind… Es ist kostenpflichtig.

Und mit Python kann man praktisch MATLAB emulieren… Das ist sehr ähnlich MATLAB. Wir haben es aus dem Grund gewählt, weil wir dann fortführende Lehrveranstaltungen haben, wo wir einfach Numeric machen und mit Vektor und Matrizen arbeiten. Dort [ist] eigentlich Python auch sehr gut geeignet… Diese Dinge einfach zu machen. Es gibt auch in dem Sektor sehr viele Material, wie man Python einfach einsetzt. In dem Wissenschaftlichen…

Und das war der Grund, warum wir es gemacht haben… Ebene wie gesagt, einerseits einfach weil es sehr gängig ist, andererseits weil es sehr MATLAB ähnlich ist, und zum Dritten weil ich einfach… Weil ich einfach diesen Overhead von einer Sprache wie C++ nicht ziehe… Weil um C++ oder Java zu machen, muss ich einmal schon sehr viel erklären, bis ich auf dem Punkt hinkommen, das ich ein kleines Programm schreiben kann.

Yazicioglu Wie zum Beispiel am Anfang um „Hello World“ zu schreiben…

Lehrer 5 Genau…

Yazicioglu In Java muss man Klasse schreiben und `System.println` und `public static void main`…

Lehrer 5 Was heisst jetzt `public void main`? Wenn man das nicht erklärt: Was ist eine `public`-Klasse. Dan muss man erklären, was ist `private`-Klasse, was ist eine `protected`-Klasse. Was ist `void`? Was ist überhaupt eine Funktion? Was ist dann die Übergabe… Also ich bin so weit drinnen. Eigentlich schon… Um in Java ein Programm zu schreiben, muss eine Klasse definieren, in der Klasse muss ich eine Funktion definieren und die macht irgendwas… In Python schreibt man einfach hin `print("Hello World")`. Fertig…

Yazicioglu Ja. Java ist fokussiert objektorientiert. Um ein Programm zu schreiben, muss man in Java wirklich mit OOP beginnen. Finden sie das am Anfang. Soll man mit OOP beginnen?

Lehrer 5 Nein. Ich finde man sollte mit einem imperativen Stil beginnen. Und was mir gefällt an Python. Ich beginne am Anfang mal so: Das man sagt Python als Taschenrechner… Weil das versteht jeder. Weil einen Taschenrechner haben sie auf ihrem Handy. Einen Taschenrechner haben sie wahrscheinlich in der Schule verwendet. Wenn ich sage sie können Python so verwenden wie einen Taschenrechner. Dann hole ich sie mal ab. Das heißt sie haben das Gefühl sie wissen was das ist. Und dann sage ich: Okay, und all die Befehle, die sie in den Taschenrechner eintippen wäre ja schön wenn sie die

irgendwo untereinander gespeichert haben. Oder oft haben sie Berechnungen im Taschenrechner, die sie immer wiederholen.

Naja, dann speichern sie die in einem File, und dieses File als Python-Script. Und dann haben sie irgendwo das Gefühl verstanden zu haben, was ist jetzt der Unterschied zwischen einem Taschenrechner und einem Python-Script. Ein Python-Script ist ja schon ein Programm. So finde ich kann man das eigentlich didaktisch gut erklären. Über einen Taschenrechner das kann man in Python machen. Das kann man nicht in C++ machen oder in Java. Java-Taschenrechner gibt es nicht. Das heißt… Aber das ist von der didaktischen Seite ist eigentlich Python da sehr… Man kann da eben das… Diese Taschenrechner-Funktion ist sehr schön… Und halt dieser ganze Overhead ist nicht da. Aus diesem Grund finde ich als erste Programmiersprachen eigentlich wesentlich besser…

Es geht mir ja nicht darum zu sagen: Die eine Sprache und sonst Nix. Sondern wie von vorher von Stroustrup. Das ist eine Sprache von mehreren die du lernen solltest. Aber als erste Sprache ist es eigentlich ganz gut. Und wenn du dann verstehst: Aha das mit der dynamischen Deklaration, das ist eigentlich ein Feature, da muss man aufpassen. Es gibt dann wirklich Applikationen, wo ich aufpassen muss, was ich mache… Dann ist das eine weiterführende Sache und das muss ich ja nicht in meinem Grundkurs bringen. Weil in zwei Stunden kann ich ja nicht das bringen was andere in ihrem ganzem Studium lernen.

Yazicioglu Eine Eigenschaft von Python haben sie [erwähnt]: Taschenrechner. Es gibt noch List-Comprehensions. In Java oder C++ [ist das Erstellen von] Listen/Arrays so kompliziert… In Python… Am Anfang ist es…

Lehrer 5 Also einerseits sind diese klassischen Konzepte… Was natürlich schon C++ hat. Wo Fortran, diese Low-Level-Sprachen schon Probleme [bekommen]… Listen auf der einen Seite, diese ganzen List-Funktionen gibt es teilweise in der STL von C++, aber Fortran hat so-etwas nicht. Aber diese Maps und Dictionaries… Es gibt ja diese Maps. Aber wenn sie das anschauen, wie sie eine Map deklarieren in C++… Wie kompliziert das ist… Map vom Typ `int` auf `int` und das merken sie sich nie auswendig, weil sie es nicht jeden Tag machen. In Python schreiben sie map ist gleich, runde Klammer auf, runde Klammer zu, erster Doppelpunkt, zweiter Eintrag fertig. Das heißt von der Syntax her viel leichter zu verstehen, und doch auch diese High-Level-Dinge und doch auch…

Das sind ja nicht nur diese Standard-Dinge wie Listen und Dictionaries, sondern… Was für uns wichtig ist, sind die ganzen Arrays… Das heißt diese NumPy-Library, auch SciPy, also Scientific-Python und Numeric-Python… Also sozusagen NumPy, wo sie die ganzen Arrays eigentlich definieren als Vektoren und Matrizen… Und wo sie dann die ganzen Vektoren-/Matrizen-Operationen darauf machen können.

Das heißt… Wo sie die ganzen inneren Produkte transponieren, Matrix-Multiply, usw. Eigentlich alles machen können… Was eigentlich sehr angenehm ist, weil man sich darum nicht mehr kümmern muss. Also das gibt es ja so in dem Sinn in C++ auch nicht. Es gibt zwar irgendwo eine Mathematik-Library, in Java z.b. gibt es ganz wenige Mathematik-Libraries oder Vektor/Matrizen-Libraries. Und wenn dann hat man das Problem, dass man halt bei diesen Bibliotheken… Ja…

Zum Beispiel nur wenn ich in C++ Programme schreibe… Ich muss kompilieren. Ja… Und wenn ich dann Bibliotheken dabei habe… Ja, dann wird das schon einmal lustig… Wo sind die Bibliotheken? Wie setzte ich die Pfade. Wie mach ich das, wie mach ich das, dass ich mit einer externen Bibliothek die Dinge überhaupt kompilieren kann. In Python, richtig installiert, sage ich `import` Paket… Fertig… `import numpy` und ich kann NumPy verwenden… Ja, es gibt in Python Installer, die installieren alle Python-Pakete an der richtigen Stelle und Python findet die…

Yazicioglu So einfach…

Lehrer 5 So einfach… Natürlich, wenn es dan irgendeine Probleme gibt, dann wird es kompliziert… Weil niemand genau weiß, wo jetzt das wie passiert ist.

(Diskussion über nicht relevantes Thema)

Yazicioglu Okay. Welche Kriterien und Merkmale muss man berücksichtigen beim Anfangsunterricht. Manche haben die Kriterien für die zukünftige Berufsmöglichkeiten muss man Java wählen, oder C++. Manche sagen wieder, die didaktischen Ansätze sind sehr wichtig. Manche sagen, dass die mathematischen Berechnungen und…

Lehrer 5 Ich finde, dass die Sprache selbst zweitrangig ist. Es geht darum zu verstehen: Was ist Programmieren? Was heisst algorithmisches Denken? Umsetzen von Formeln, von Gleichungen in einer Software, die mir etwas berechnet… Womit ich das mache, das ist zweitrangig. Warum?

Es geht um die Grundlagen. Es geht ums Grundverständnis. Weniger darum, wie ich mich ausdrücke. Wenn sie z.B hier in Österreich etwas lernen auf Deutsch oder in der Türkei, etwas das Türkisch, oder in Amerika, etwas Englisch ist es eigentlich egal in welcher Sprache sie das Lernen. Sondern es geht darum was sie gelernt haben. Sie können dann das relativ leicht in jede andere Sprache übersetzen. Was sie hier gelernt haben können sie relativ leicht in ihre Muttersprache übersetzen, ins Englische übersetzen, oder was auch immer…

Wenn sie Mathematik machen, und sie müssen eine Differenzial-Gleichung lösen, es ist egal welche Sprache sie das lernen. Sie müssen lernen wie sie es lösen. Genauso ist es beim Programmieren. Es ist egal in welche Sprache sie das Programmieren lernen.

Es wichtig, dass sie programmieren lernen. Das heisst das sie die Konzepte verstehen: Von Fallunterscheidungen, von Funktionen, von Schleifen, von Klassen, Objektorientierte Programmierung… Aber wie ich das jetzt wirklich schreibe, ob ich dort hinschreiben muss `class`, Name Klammer auf, irgendwas… Oder irgendeine andere Bezeichnung, das ist eigentlich egal. Ich muss das Grundkonzept verstehen.

Yazicioglu Und [das] Grundkonzept. Denken sie, dass am Anfang, sollen [es] wirklich klarere, leichtere, Sprachelemente sein?

Lehrer 5 Zum Beispiel wenn sie erklären wollen, denn… Ein Grundkonzept z.B.: `if`/`else`-Konstrukt… Dann es ist egal welche Sprache sie dafür verwenden. Es ist… Sie müssen erklären didaktisch… Sie haben schon alles erklärt: Was ist ein Programm? Was ist eine Variable? Und da sagen sie: Jetzt komm ich an den Punkt wo ich eine Variable eingebe… Jetzt muss ich überprüfen, ob die Variable positiv oder negativ ist. Und wenn sie zum Beispiel… Oder null… Da hab ich `if`, `elif`, `else` von mir aus. So, dann muss ich den Algorithmus verstehen. Ich muss fragen, ist die Variable größer Null, ist die Variable kleiner Null oder ist die Variable Null.

Zum Beispiel… Wenn ich das verstanden habe, dann es ist egal wie ich das hinschreibe. Ob ich das jetzt in der C++ Notation hinschreibe, die ein bisschen umständlicher ist, in der Python-Notation… Aber ich habe verstanden, was das macht. Also ich finde ich… Wenn jemand sagt, das ist wichtig, dass man Java oder C++ lernen muss für den Beruf, dann finde ich das nicht… [Das] sehe ich nicht so: Weil wer kann Ihnen garantieren, dass Unternehmen, in dem Sie arbeiten C++ oder Java oder irgendwas anderes macht.

Yazicioglu Ja stimmt. Es gibt viele Firmen, die Python machen,

Lehrer 5 Es gibt welche, die machen Python… Es gibt welche, die machen Fortran… Es gibt welche die machen Visual-Basic… Es gibt welche die machen irgendwelche anderen Dinge. Und wenn sie Grundkonzept verstanden haben, dann schauen sie einfach nur nach, wie heisst den jetzt das `if`-Konstrukt in Visual Basic

Yazicioglu Sie sagen Java oder C++ ist viel verbreiteter als Python.

Lehrer 5 Ich finde es ist genauso. Es ist genauso wenn sie z.B. sagen: Sie müssen Microsoft Word lernen. Ich klicke jetzt auf diesen Knopf, dann passiert das… Ich klicke jetzt auf diesen Knopf, dann passiert das… Wenn sie das machen, dann müssen sie spätestens in zwei Jahren wieder einen Microsoft Word-Kurs machen.

Weil es geht nich darum, dass sie wissen, auf welchen Knopf sie klicken und was passiert… Sondern sie müssen intuitiv wissen: Ich habe ein weißes Blatt. Ich schreibe Text. Ich kann den Font ändern. Ich kann die Farbe ändern. Wo ist der Knopf, wo ich eben den Font ändere. Ja, und nicht ich muss auch den dritten Knopf rechts oben klicken um den Font zu ändern. Und das ist genau der Grund warum ich sage, ich muss Java lernen, damit ich im Beruf mehr Chancen habe… Das sehe ich nicht.

Ich glaube es ist wichtig, dass man die Basiskonzepte erklären… Und dann sollte man relativ leicht die Sprache haben… Ohne viel Overhead und wo man das dann demonstrieren kann. Das ein 15-jähriger auch Spaß dabei hat und irgendwas damit machen kann.

Und genau dort ist es zum Beispiel so, wo ich denke da muss man da halt auch überlegen… Zum Beispiel, das ist ja gerade in Python… Das hat den großen Vorteil, dass sie ohne viel Overhead eine grafisches User Interface machen können. Man sollte nicht zu stark auf das fokussieren. Aber es

ist doch schön für jemanden wenn er auch ein Fenster aufmachen kann und er kann das vielleicht bewegen. Die Farbe ändern… Was auch immer…

Yazicioglu Es ist in Java… Mit Java…

Lehrer 5 In Java ist [das mit dem] grafischen User-Interface relativ leicht. In C++ ist es komplett anders. In C++ können sie QT verwenden oder irgendwas. Aber es gibt keine C++ Library drinnen… Direkt wo sie das machen können.

Yazicioglu In Python gibt es TkInter

Lehrer 5 TkInter gibt es… Es gibt auch Python Qt-Bindings. Es gibt PyFltk… Was auch immer das ist

Yazicioglu Das hab ich noch schöner gefunden. Direkt… Man kann so schnell einsteigen…

Lehrer 5 Ja, sie machen… In Python merken sie das gar nicht. `import tkinter` und können ein grafisches User-Interface machen. Ob das jetzt wirklich für große Programme geeignet ist, das steht auf einem anderen Blatt Papier. Glaub ich nicht.

Aber wie gesagt. Um zu demonstrieren, was sie machen. Das sie vor einem weißen Blatt auch zeichnen können. Das sie Zahlen, die sie ausrechnen… Das sie ein Dreieck zeichnen können. Und das sie mit einer `for`-Schleife dieses Dreieck rotieren können, indem sie einfach die drei Koordinaten nehmen und mit einem Rotations-Ding machen… Und sie sehen, dieses Ding einfach bewegt.

Und ich glaub dort fängt man dann die Leute wirklich ab. Wo sie dann verstehen… Warum, wie, was und… Vor allem glaub ich für die ist das auch wichtig, dass die Comics erstellen können, das sie Computer spielen und so weiter… Das die einfach dann sehen: Was ist da eigentlich. Wie funktioniert das eigentlich. Und da muss ein bisschen Grafik dabei sein, dass denen das halt gefällt.

Yazicioglu Spielt die Auswahl [der] Programmiersprache auch eine Rolle bei der Motivation, den Erfolg? Spielt [das] eine Rolle…

Lehrer 5 In gewissen… Ich sag jetzt mal so… Vielleicht für den Erfolg, den Misserfolg. Das heist da: Wenn ich keine Programmiersprache hab, die so einen großen Overhead hat. Das ich vor lauten Bäume der Wald nicht sehe… Weiß nicht ob sie den Ausspruch kennen… Sie stehen mitten im Wald und sie sehen nur Bäume um sich. Und sie sehen eigentlich den Wald nicht. Also das von den vielen Bäumen irritiert sein…

Das gleiche bei einer Programmiersprache: Sie sehen nur `public void main` Klammer auf, Doppelpunkt, Schleife… Sie sehen eigentlich gar nicht was das Dinge macht. Das heisst, eigentlich wollten sie nur „Hello World" ausgeben, aber sie sehen nicht Hello World… Sie sehen nur `public void main`, Klammer auf, Klammer zu usw.

Irgendwann kommt ein `print`. Aber dieses `print` sehen sie schon gar nicht mehr… Weil sie sind so irritiert, dass sie vielleicht gar nicht mehr so weit lesen. Und insofern glaube ich, sollte es eine Programmiersprache sein, die die Leute abholt. Die sozusagen es ist ihnen ermöglicht Dinge zu programmieren, wo sie was sehen… Aber nicht so viel Overhead darauf haben auf dem Ganzen. Und deshalb finde ich Java oder C++ als erste Programmiersprache weniger geeignet.

Yazicioglu Super. Und wie ist ihre Meinung? Womit haben die Schüler am meisten Probleme am Anfang? Bei Semantik, bei Debugging oder [dem] Verständnis von Konzepten wie Abstrakt Denken, Abstrakt…

Lehrer 5 Ich glaub das letzte ist das größere Problem. Weil… Natürlich ist es bei der Semantik so eine Sache. Das ist in jeder Sprache so. Aber zuvor muss ich einmal den Algorithmus überlegen.

Yazicioglu Spielt dann da die Programmiersprache eine Rolle oder?

Lehrer 5 Nein, weil wir das zum Beispiel mit UML machen.

Yazicioglu Aha. Okay.

Lehrer 5 Also wenn wir das mit UML machen, dann ist es vollkommen egal welche Sprache sie verwenden. Je nachdem was sie für einen UML-Pool haben… Dann designen sie mal ihre Klassen. Was… Wie… Und dann schreiben wir dann raus Python… Und dann schreibt jemand die ganzen Klassendefinitionen raus.

Aber zu wissen welche Klassen mit welchen wie interagieren. Oder zu sagen, wie schaut es algorithmisch aus. Ja, das ist die Schwierigkeit und das ist auch die Herausforderung. Das muss man umsetzten. Man muss zunächst einmal kommen mit einem Flow-Chart, mit einem… Um zu lernen muss ich mal die Basis-Semantik kennen.

Das ist so wie in jeder Sprache, auch in einer Fremdsprache. Ich muss gewisse Vokabeln können. Ich muss wissen wie ich sozusagen die Wörter zusammen-[geben], weil sonst kenn ich mich gar nicht aus. Aber dann ist es wichtig, dass man denen sozusagen sagt, wie ist das Ganze algorithmisch zu sehen. Überlege dir einen Algorithmus, und wenn du den hast… Ich muss das mit dem prüfen. Das mit dem prüfen… Wenn das, dann das… Wenn das, dann das… Dann setzt das jetzt um in einer Programmiersprache…

Yazicioglu [Den] Algorithmus umzusetzen, von einer Sprache… Vom Denken zu einer Sprache ist… Finde ich von Python noch leichter, weil Pseudo-Code [ist] fast ähnlich… Algorithmus/Pseudocode…

Lehrer 5 Genau. Das ist auch ein Vorteil von Python. Weil Python selbst ist ja fast schon Pseudo-Code. Also wenn sie Pseudo-Code schreiben und Python anschauen… Das unterscheidet sich nicht wirklich viel.

(Diskussion über nicht relevantes Thema)

Yazicioglu Okay. Danke schön

Lehrer 6

Yazicioglu Welche Programmiersprache würde Sie als Unterrichtssprache wählen?

Lehrer 6 Das hängt davon ab: In der ersten Klasse unbedingt mit einer Sprache, mit einer C ähnlichen Syntax anfangen… Am liebsten mit einer stark typisierten [Sprache], wobei prinzipiell auch JavaScript in Frage käme.

Wenn ich Richtung Netzwerktechnik denke. Linux-Server werden hauptsächlich in Python heutzutage ge-scriptet… Da ist es notwendig, dass wir heutzutage Python lernen.

Yazicioglu Okay

Lehrer 6 Das betrifft dann aber den 4. Jahrgang.

Yazicioglu Aha. Für den ersten Jahrgang empfehlen… Würden sie nicht…

Lehrer 6 …Würde ich nicht Python empfehlen. Erstens, weil es eine Sprache ist, die völlig anders ist. Das ist so wie, wenn ich mich für eine europäische Sprache interessiere, und dann halt im Endeffekt Deutsch, Spanisch und Englisch lernen möchte, und als Startsprache wähle ich dann Finnisch.

Das hat nichts damit zu tun. Und Python ist einfach völlig anders. Das ist jetzt noch nicht die Frage, ob besser oder schlechter… Aber an Sprachen wie C++, JavaScript, PHP kommt man eigentlich nicht vorbei.

Yazicioglu Okay. Was sehen sie anders bei Python? Anders als Java, C#?

Lehrer 6 Es hat keine C-Syntax. Es hat keine C ähnliche Syntax. Ich lerne zwar vielleicht, was eine Schleife ist, aber ich muss es dann in anderen Sprache nocheinmal in einer völlig andere Art und Weisse erlernen. Weil ich muss… Alle normalen Sprachen haben geschwungene Klammern statt einer Einrückung… Was auch in Python völlig anders ist.

Python ist unheimlich mächtig was Listen betrifft. Da [ist es möglich das] sehr kompakt zu schreiben. Da kommen die andere Sprachen nicht rein. Nur was nützt mir das, wenn ich dann im Endeffekt PHP draußen programmiere oder Javascript oder C++ oder Java. Und die Chancen sind in Python… Das ich sehr viel programmiere noch… Die ist noch nicht sehr gegeben. Aber vor allem bitte als Anfängersprache halte ich es für ungeeignet.

Yazicioglu Welche Kriterien/Ansätze berücksichtigen sie bei der Auswahl einer Unterrichtssprache im Unterricht? Zukünftige Berufsmöglichkeit? Didaktische Aspekte? Oder beides? [Ist das] für sie wichtig?

Lehrer 6 Aus didaktische Sicht brauch ich… hätte ich gerne eine Sprache, die stark typisiert ist. Weil auch, wenn ich jetzt JavaScript wähle, oder Python wähle… Intern habe ich dann doch wieder Typen. Und muss sehr wohl genau wissen, was ich tue. Der Nachteil einer schwach typisierte Sprache ist für mich, immer dass ich erst gewisse Dinge zur Laufzeit feststelle, anstatt schon beim Kompilieren…

Ein bisschen helfen tun moderne Entwicklungsumgebungen, wie PyCharm, die einiges schon als Fehler markieren, was rein prinzipiell durch die Sprach-Syntax noch nicht testbar ist, aber auf Grund des Codes erkennbar ist, dass das wohl falsch sein muss. Also starke Typisierung finde ich wichtig. Ich finde es auch wichtig, dass ich eine halbwegs vernünftige Objektorientierung habe. Da wären wir ja hier bei Java und C++. Der Nachteil von C++ ist halt doch, dass es sehr komplex ist zum erlernen… Weil man sehr viel Freiheiten hat. Und standardmäßig „Out of the Box", nicht so viele Bibliotheken.

Yazicioglu Okay am Anfang sollen die Schüler OOP lernen? Oder das beginnt…

Lehrer 6 Nein, erst in der zweiten Klasse. In der ersten Klasse noch nicht. Da ist es… nur ein reines Anwenden von Objekten…

Yazicioglu Okay. Mit Java… Man programmiert… Java ist fokussiert auf [Objektorientierung]… Das… Wie lernen…

Lehrer 6 Das ist erst in der zweiten Klasse…

Yazicioglu In der ersten Klasse, welche Sprachen lernen die Schüler?

Lehrer 6 Java in der IT. In der Mechatronik selbstverständlich C, weil die werden dann im Endeffekt Embedded Systems programmieren, mit einem PIC, der €30 kostet. Und da hat Java nichts verloren.

Yazicioglu Okay. Welche Lehrkonzepte gibt es im ersten Jahr? Außer… Es gibt kein OOP sondern mal die generellen Sprachelemente, wie Schleifen, diese Sachen…

Lehrer 6 Genau. Also Kontrollstrukturen, Methoden, Strings, Arrays. Das ist so der große Brocken in der ersten Klasse.

Yazicioglu Wo haben die Schüler die meisten Probleme, bei Java? Bei Semantik, Debugging, oder Verständnis?

Lehrer 6 Ich würde sagen, unabhängig von der Sprache haben sie die größten Probleme. Das sie nicht artikulieren können. Irgendeine Form von Lösungsstrategie vom dem Algorithmus. Das hat nichts mit der Sprache zu tun

Yazicioglu Okay, [vom] Verständnis… [Das hat] mit der Logik zu tun…

Lehrer 6 Ja. Ich denke auch, dass für den durchschnittlichen Schüler, die erste Klasse fast noch ein bisschen zu früh ist. Da ist das logisch abstrakte Denken noch nicht so ausgeprägt.

Yazicioglu Okay. Da denken sie spielt die Sprache keine Rolle.

Lehrer 6 Das… Da spielt die Sprache keine Rolle. Also unabhängig von der Sprache, die Hürde haben sie immer.

Yazicioglu Okay

Lehrer 6 Da ist die Sprache völlig sekundär. C# käme auch noch in Frage zum Anfangen. Weil es ebenfalls zu den C ähnlichen Sprachen gehört.

Yazicioglu Okay. C#… Okay, bei C# haben die Schüler auch die gleichen Probleme wie in…

Lehrer 6 Da seh ich keinen Unterschied. Wir unterrichten es zwar nicht, aber zum Beispiel die HTL Ottakring. Die unterrichten C#.

Yazicioglu Ich hab auch gehört hier wird C# unterrichtet?

Lehrer 6 Nein. Das ist falsch.

Yazicioglu Okay…

Lehrer 6 Es gibt ein paar Schüler die das machen. Selbst… Aber unterrichten selber tun wir es nicht.

Yazicioglu Okay…

Lehrer 6 In keinem Jahrgang

Yazicioglu Spielt die Programmiersprache bei der Motivation, beim Erfolg, Demotivation, beim Misserfolg eine Rolle?

Lehrer 6 Naja im Laufe der Jahre haben wir immer ausprobiert mit Python zu beginnen. Das hat sich… Das war aus unsere Sicht nicht erfolgreich.

(Diskussion über nicht relevantes Thema)

Yazicioglu Warum [war] Python nicht so erfolgreich? Was finden sie…

Lehrer 6 Warum ich finde, dass Python nicht in die erste Klasse gehört? Weil…

Yazicioglu Statische Typisierung haben sie erwähnt. Aber sie haben [es] probiert.

Lehrer 6 Nein. Weil Python eine völlig andere Syntax hat als jede andere Sprache, die sie dann auch später brauchen werden. Wenn ich in die Medientechnik gehe werd ich JavaScript und PHP auf jeden Fall brauchen. Unbedingt.

Yazicioglu Weil man später diese Sprachen…

Lehrer 6 Ja. Wenn ich jetzt mit Java, oder C++, oder C# anfange, dann hab ich eine `for`-Schleife, eine `while`-Schleife. Das schaut ähnlich aus.

Yazicioglu Okay.

Lehrer 6 Ja… Eine Funktion schaut ähnlich aus. Wenn ich mehrere Befehle in einem `if` hab, mach ich nur eine Klammer in all diesen Sprachen. Python ist einfach völlig anders. Und deswegen finde ich es ist keine günstige Sprache zum anfangen.

Yazicioglu Okay. Sie haben probiert es einzusetzen. Es ist nicht erfolgreich gewesen?

Lehrer 6 Mhm.

Yazicioglu Was war der Misserfolg, bei der… Für die Schüler oder die Lehrer? Zum Erklären… Oder…

Lehrer 6 Nein. Ich hab kein Problem in der ersten Klasse Python zu erklären. Das wäre traurig, wenn das so wäre. Nein, ein Umstieg auf eine andere Sprache ist immer ein bisschen…

Yazicioglu Ah. Das…

Lehrer 6 Ansonsten… Der eigentliche Unterrichtsertrag, also das was wir mit der Sprache machen konnten war ähnlich. Da seh ich… Da spricht nichts dagegen. Wenn sie noch eine Sprache lernen müssten, dann könnte man über Python diskutieren. Aber zwei Drittel unserer Schüler werden Medientechniker. Und die brauchen dann zwei Sprachen. Die mindestens dann völlig anders sind… Das halt ich nicht für geschickt… Und das hat sich auch nicht bewährt.

Yazicioglu Okay. Sie sagen, dass Python anders ist.

Lehrer 6 Ja…

(Diskussion über nicht relevantes Thema)

Yazicioglu Es gibt in Python einen „Taschenrechner“… Man kann mit funktionalem Paradigma beginnen zuerst. Mit mathematischen Berechnungen… Finden sie es gut, dass man [etwas] Mathe-ähnliches macht… Aber ja…

Lehrer 6 Also wir haben verschiedenes ausprobiert. Wir haben eben ausprobiert relativ mathematisch zu sein. Oder in Python, mit V-Python jetzt zu zeichen. Beziehungsweise auch für Java gibt es solche Tools… Das man mit relativ wenig Aufwand zeichen kann. So dass man dann halt bei einer Schleife meinetwegen ein Schachbrett zeichnet. Das man irgendwelche… Was weiß ich… Die ersten hundert Zahlen ausgibt, die durch 3 und 7 teilbar sind.

Ich glaube, dass das für die Schüler relativ [egal] ist. Nämlich wirklich [egal]… Wo Schüler meistens sehr sehr motiviert sind, wenn sie eine recht freie Aufgabenstellung haben, wo sie ein Spiel selber programmieren dürfen. Zumindest die mittleren bis besseren Schüler sind dann sehr stark motiviert. Das ist jetzt wieder das Problem… In einer ersten Klasse geht das noch nicht. Das kann man in einer zweiten Klasse beginnen. Das sie Projekte über zwei Monate machen. Also ich hab immer wieder von Schülern das Feedback bekommen wenn sie dann in der Fünften waren: In den fünf Jahrgängen haben wir in der zweiten Klasse im Projekt am meisten gelernt. Weil sie da selbst motiviert… Sich selber Sachen zusammensuchen müssen… Was ja auch gut ist. Und es über den normalen Stoff hinüber-ausgeht.

Yazicioglu Okay

Lehrer 6 Also der… Normalerweise lernt man [etwas]. Dann muss man es in einer Übung anwenden. Dann lernt man etwas neues. Dann muss man es wieder in einer Übung anwenden. Und in zwei Monaten, wenn ich dann vergessen habe, was eine Menge ist… Und ich aber eine Menge brauche, dann hab ich es… Weil es es aus dem Kontext herausgerissen ist… Weil ich es nicht unmittelbar davor gemacht hab… Ist das sicher auch in einer dritten Klasse immer noch eine Hürde.

Yazicioglu Okay. Dann…

Lehrer 6 Also wie gesagt, ich glaube am meisten lernen sie tatsächlich bei einem Projekt. Wobei das natürlich für den Lehrer wahnsinnig anstrengend ist. Einerseits in der Betreuung, andererseits natürlich sehr schwierig in der Beurteilung.

Yazicioglu Okay…

Lehrer 6 Ein Test ist etwas einfaches. Da geb ich Punkte und die Sache hat sich. Wie beurteile ich ein Projekt?

Yazicioglu Sie denken, der [auf ein] Projekt bezogene Unterricht ist viel besser?

Lehrer 6 Ja.

Yazicioglu Man soll nicht nur die Algorithmen verstehen, sondern Projekte… Größere Aufgaben…

Lehrer 6 Ja. Wenn Schüler sich im angemessenem Schwierigkeitsgrad einerseits zur Klasse, andererseits auch zum Können selbst eine Aufgabe stellen. Man hat natürlich irgendwelche Rahmenbedingungen wie: Eine grafische Oberfläche muss dabei sein. Es muss irgendetwas über [das] Netzwerk kommunizieren. Oder… Was auch immer… Also Randbedingungen gibt es natürlich, aber im wesentlichen dürfen sie sich dort aus diesen Randbedingungen selber etwas zusammenstellen.

Yazicioglu Okay. Das wollte ich auch fragen. Was am Anfangsunterricht… Welche Konzepte lernen die Schüler? Wie soll der Anfangsunterricht didaktisch gestaltet sein? Was betrachten sie [beim] Anfangsunterricht. Sollen die Schüler schnell Programme schreiben, oder Schritt für Schritt die Sprachelemente lernen? Und danach beginnen, zu den großen…

Lehrer 6 Ich glaube so ganz trennen kann man das nicht. Man kann sich bestenfalls überlegen, mit welchen Sprachelementen fange ich an? Fange ich an mit Kontrollstrukturen oder fange ich an mit Funktionen? Aber im wesentlichen dreht es sich im Kreis. Ich brauch… Wenn ich Kontrollstrukturen habe brauche ich Variablen. Wenn ich Variablen hab… Damit muss ich praktisch anfangen, weil das kommt überall vor. Auch bei Funktionen, Methoden…

Yazicioglu Okay, ja…

Lehrer 6 Wobei ich persönlich finde: Was für Schüler sehr sehr schwierig ist… Wenn man etwas in der ersten Klasse nicht macht. Etwas schon mach… Egal… Das prägt sie dann so sehr, dass sie es in der dritten, vierten Klasse genauso tun. Also entweder nicht, oder schon…

Yazicioglu Aha

Lehrer 6 Das heist man muss sie schon in einer ersten Klasse beginnen zu lenken. Das heist zum Beispiel… Was auch in Java wichtig ist… Die Einrückung. Das sie es beinhart in einem Test Minuspunkte gibt. Oder bei einer Übung Abzüge gibt, wenn die Einrückung nicht passt. Obwohl das Programm syntaktisch und logisch richtig ist. Sonst werden sie das auch in der dritten Klasse nicht mehr richtig machen.

Yazicioglu Mhm

Lehrer 6 Wenn ich nicht von Anfang an, in der Objektorientierung jedes Attribut `private` mache, dann werden sie es auch in der fünften Klasse noch nicht tun. Es ist meiner Meinung nach geschickter, sie am Anfang auf das richtige hinzu-trainieren, und dann erst Schritt für Schritt zu erklären, warum das so ist.

Yazicioglu Okay. Darf ich vielleicht 10 oder ich weiß jetzt nicht Aufgabenstellungen…

Lehrer 6 Ja können sie gerne sehen…

…Traditionell lösen. Also als Beispiel das Dividieren, dafür braucht man in der Volksschule fast ein ganzes Jahr. Die vierte Klasse Volksschule. Versuchen sie einmal den Divisionsalgorithmus in Worte zu fassen. Exakt…

Yazicioglu Okay

Lehrer 6 Das werden wahrscheinlich viele A4-Seiten, für etwas so simples wie das Dividieren. Und das finde ich schon charakteristisch. Man braucht meiner Meinung nach auch deswegen so lange, weil dieser Algorithmus so schwer in Worte zu fassen ist. Deswegen kann ich es nur durch Übung und durch Tun.

Yazicioglu Mhm. Diese Probleme hängen… Nicht wegen der Sprache selbst, sondern wegen dem algorithmischen Denken…

Lehrer 6 Ja, aber wenn ich eine Sprache habe, wo ich etwas exakt ausdrücken kann, dann tu ich mir leicht. Was weiß ich… Mathematik hat sicher sehr gewonnen durch die arabischen Zahlen. Davor konnte man in Europa wenig Mathematik betreiben. Sicher. Oder die Null… Oder… Was weiß ich… Das Differenzieren… Durch Null… Plötzlich hat man ein Modell und kann das ausdrücken. Vorher hat man gewisse Probleme nicht lösen können. Also ich glaube schon, dass es… Nicht unbedingt nur beim Programmieren. Aber wenn ich nicht darüber reden kann, dann kann ich es nicht lösen. Oder nicht so leicht lösen.

Yazicioglu Mhm

Lehrer 6 Und ich denke, dass man mit Schülern sehr wohl auch einfach Probleme vielleicht jetzt gar nicht programmiertechnisch umgesetzt werden. Verbal versuchen es zu lösen… Sie können Lösungsstrategien nicht in Worte fassen.

Yazicioglu Ah. Dann müssen sie üben meinen sie…

(Diskussion über nicht relevantes Thema)

Lehrer 7

Yazicioglu Ich wollte [sie] fragen, welche Programmiersprache würden sie als Unterrichtssprache bevorzugen? Warum? Und welche Vorteile, Nachteile können sie mir nennen?

Lehrer 7 Also aus meiner Sicht sollte in einer berufsbildenden Schule, und das ist die HTL, einer der beiden führenden Plattformen unterrichtet werden. Das ist Java-Plattform oder die C#/.Net-Java Plattform.

Das sind die Plattformen, die de facto in der Industrie die dominierten Rollen spielen. Das heisst auch im Sinne der Vorbereitung der Schüler auf das Berufsleben soll aus meiner Sicht einer der beiden Plattformen das Thema sein. Und zwar wirklich eine. Es mach wenig Sinn, dass man Java, C# parallel macht, das bringt überhaupt nichts. Weil die Ähnlichkeiten der Sprachen sehr sehr groß sind und es geht ja nicht nur um die Programmiersprache. Es geht um die ganze Plattform.

Sprich das Betriebssystem, die Server-Technologien. Da sollte aus meiner Sicht, einer dieser beiden Plattformen, die sein, die in die Schule über die Jahre hinweg dominiert. Ich hab auch Schulen erlebt… Ich hab auch [Fachhochschulen] erlebt, wo man so im Halbjahrestakt neue Sprachen lernt. Da macht man C, dann macht man Visual Basic, dann macht man C#, dann macht man Java. Das halt ich persönlich für relativen Unfug. Es ist zwar schön wenn man auf den Lebenslauf schreiben kann: Ich hab in der Schule 20 Sprachen gelernt. Aber man beherrscht nicht eine davon in Wirklichkeit.

Unser Ansatz hier in der Schule ist der, dass wir eine Plattform nehmen, die ziehen wir konsequent über die Jahre hinweg durch. Und wenn es dann Themen gibt, Aufgabenstellungen gibt, Diplomarbeiten gibt und das haben wir dann auch immer wieder… Wo dann eine andere Plattform zu wählen ist, dann tun sich die Schüler relativ leicht, weil sie mal eine gut schon beherrschen. Das heisst, wir versuchen, nicht mit möglichst vielen Sprache bisschen was zu machen. Sondern eine Plattformen möglichst in die Tiefe durch zu exerzieren.

Was jetzt nicht heisst, dass wir nur das machen. Also die Schüler heuer in der Fünften, da haben wir als Haupt-Plattform C# gemacht. Also die .Net-Plattform. Die heuer Diplomarbeiten gemacht, mit Java auf Android, mit C auf Mikrocontroller usw.

Das heisst, wenn man in einer Plattform wirklich gut drinnen ist… Ein `if` in C zu schreiben oder eine `while` schleife in Java, oder C# zu schreiben, das ist nur mehr Syntax. Wichtig ist zu verstehen, wie die Vorgänge sind. Ja besser man eine Plattform im Griff hat, desto leichter tut man sich dann sozusagen auf andere seitlich auszuweichen.

Yazicioglu Okay. Danke. Würden Sie Python als Unterrichtssprache wählen?

Lehrer 7 Das würde ich genau aus den genannten Gründen nicht machen. Also ich würde keine Skriptsprache nehmen. Ganz egal ob das jetzt JavaScript ist oder Python, Perl oder PHP ist… Weil die haben so alle ihre Berechtigungen und ihre Plattformen und ihre Anwendungsfälle… Aber sind am Markt jetzt nicht so die dominierten Plattformen.

Und ich glaube auch, dass man sich mit einer Skriptsprache am Anfang vielleicht bisschen leichter tut. Aber das die Möglichkeiten, wie weit man dann gehen kann, doch limitiert sind. Wenn ich an eine Java-Plattform, mit Enterprise Java Beans, usw.

Da tut man sich mit Skriptsprache halt doch relativ schwer. Das heisst zum Lernen, ist es wahrscheinlich ziemlich egal. Aber ich denke die Linie auf die man dann stoßt, sind bei Skriptsprachen… Da werfe ich jetzt Python mit den anderen Sprachen einfach in ein Topf rein… Die sind dann schnell erreicht… Da ist man dann schneller am Ende der Möglichkeiten.

Yazicioglu Okay. Also ja. Sie meinen, die Schüler lernen am Anfang mit Python schnell, aber weil sie Abhängigkeiten haben… Sie denken [das] die zukünftigen Berufsmöglichkeiten dieser Plattform wichtig ist. Wenn diese Plattform nicht eine größere Rolle spielen [würde], würden sie dann [Python] trotzdem nicht wählen?

Lehrer 7 Dann würde ich…

Yazicioglu Weil…

Lehrer 7 Dann würde ich irgendeine Skriptsprache nehmen, aber ich weiß nicht ob ich Python da bevorzugen würde, weil PHP hat genauso seine Stärken und Schwächen. Wenn es um Datenbaken geht, also wenn ich einen Fokus in den Datenbankbereich hätte, dann würde ich wahrscheinlich PHP nehmen. Wenn ich einen Fokus in den Webbereich hätte, mit irgendwelchen Content-Management-Systemen, dann würd ich wahrscheinlich Python nehmen. Das würde ich dann eher von den fachlichen Schwerpunkten abhängig machen. Aber nicht von der Programmiersprache an sich. Weil die sind in ihren Möglichkeiten alle sehr ähnlich. Mit den kleinen Unterschieden, die es gibt. Aber ob das jetzt Python, PHP, Perl, JavaScript… Da sind jetzt nicht so die großen Unterschiede.

Was ich an einer Skriptsprache nett ist: Das ich sie lernen kann und anwenden kann, ohne das ich ganze große drumherum einer Entwicklungsumgebung brauche. Also ich brauche keine Visual Studio, kein Netbeans, kein Eclipse. Sondern ich kann mit einem Text-Editor schreiben, kann auf der Kommandozeile werken. Das ist zum lernen auf einer Seite hübsch, auf der anderen Seite ist es nicht die Realität.

Also kein Programmierer außer wenn er unter Linux werkt, wird dann später ein halbwegs komplexes Programm mit dem Text-Editor auf der Kommandozeile schreiben… Das heist, das ist vom Verständnis her vielleicht ganz nett. Aber das ist nicht die Praxis.

Yazicioglu Welche didaktische Ansätze sind für sie wichtig? Welche Konzepte oder Merkmale… [sind] am Anfang wichtig? [Das] die Schüler schnell lernen? Oder das die Schüler alle schwierigen Konzepte am Anfang lernen.

Lehrer 7 Also mit schwierigen Konzepten würde man sie sowieso überfordern. Das heist wenn ich mir eine Plattform hernehme wie C# oder Java… Wenn man da in der ersten Klasse anfängt mit Objektorientierung und Vererbung und all diesen Dingen. Das macht keinen Sinn. Weil in der ersten Klasse sind die so damit beschäftigt, dass die ihre Klammern zählen und ihre Strichpunkte machen und das einmal in den Griff bekommen. Die kämpfen einfach einmal syntaktisch ums Überleben. Was mir viel wichtiger ist als die Programmiersprache, drum hat mich das gerade so mitgenommen… Ich finde die größte Problematik bei Schülern in den ersten Jahrgängen ist nicht jetzt die Programmiersprache sondern die Art und Weise Probleme zu lösen. Problem Solving…

Yazicioglu Also das Problem kommt nicht von [der] Syntax, Semantik… Das wollte ich auch fragen… Oder Debugging…

Lehrer 7 Nein überhaupt nicht.

Yazicioglu Sondern den Konzepten selbst…

Lehrer 7 [Das] Verständnis der Aufgabenstellung. Sie geben einem Schüler eine Aufgabenstellung hin. Es scheitert schon oft am sinn-erfassenden Lesen. Das hat nicht unbedingt etwas mit dem sprachlichen Hintergrund zu tun. Sondern einfach mit der Lese-Kompetenz.

Also wir haben oft Angaben, die sind zwei, drei Seiten lang. Jetzt liest sich kaum einer die dritte Seite durch. Man liest die ersten Zeilen und fängt zu programmieren an. Das ist einmal die erste Problematik.

Und die zweite ist die: Aus einer Aufgabenstellung wirklich eine Lösung zu bauen. Ich hab so etwas… Ich mach jetzt gerne mit der ersten Klasse einfach ein Logik-Rätsel. Eine Seite, da steht alles drinnen was man braucht. Viele Schüler lesen das da unten schon nicht mehr durch. Das Kleingedruckte interessiert mich nicht mehr. Sondern die sehen da oben… Und dann machen sie schon [ein] Kreuz. Und dann kommen sie erst darauf: Aha da gibt es Hinweise. Aber das da unten liest zum Beispiel schon mal keiner.

Und ich hab das wieder gemacht. Weil in der letzten Woche ist es ja ein bisschen so, das man so [ein] Ding gut machen kann. Weil es ja nicht mehr wirklich um Noten geht. Es haben einige gar nicht [gesehen], dass man hier die Lösung eintragen muss. Kreuz gemacht und fertig… Und nicht gesehen, dass ich hier die Lösung eintragen muss… Das heist diese Art und Weise eine Aufgabenstellung zu Sehen, Zu Lesen, zu Verstehen und eine Lösung zu bauen, das ist die größte Hürde.

Ob das dann im Programm in Java, oder C# oder C schreibe, das ist dann relativ egal. Und da finde ich es, gerade in der ersten Klasse dieses logische Denken. Dieses Problem lösen. Da haben sie die größten Probleme. Da sind sie nicht gewohnt. Wenn sie aus dem Gymnasium kommen oder auch aus der NMS kommen… Diese Art zu denken, die macht man dort nicht. Da kriegt man Stoff. Den schaut man sich an. Den lernt man auswendig. Danke das wars. Aber so an diese Dinge ranzugehen, das ist ungewohnt. Und das finde ich ist die größte Schwierigkeit. Das hat mit der Programmiersprache gar nichts zu tun.

Yazicioglu Finden sie, dass eine kleine, im gewissen Maße, eine Hilfe… Das eine Programmiersprache wenig Syntax hat… [Das sie so] sauber, klar ist, dass man sein algorithmisches Denken schnell umsetzen kann?

Lehrer 7 Definitiv. Ja.

Yazicioglu Hilft es im gewissen Maße.

Lehrer 7 Ja.

Yazicioglu Das hab ich bewundert in Python… Es ist so leicht… Man muss sich nicht mit der geschwungen Klammer beschäftigen. Und es geht so… Libraries… Sehr schnelle Lösungen. Ich hab zuerst Java gelernt und nachher [hab ich] Python ein wenig probiert. Das hat mir sehr gut gefallen. Weil man kann schnell lernen… Und [die] Motivation am Anfang [ist] sehr gut. Durch diese Motivation kann man weiter gehen. Aber das ist nur am Anfang. In der späteren Phase… Ich bin noch nicht so weit mit Python… Wie weit kann Python helfen, das…

Lehrer 7 Ja, also mit Python stoßt man dann irgendwann einmal an. Python oder eben die ganzen Skriptsprache sind nett um einmal schnell etwas zu machen… Aber dann wenn es um komplexe Aufgabenstellungen geht: z.B [Wenn sie] eine grafische Oberfläche machen wollen mit Datenbankanbindung usw. Dann wird es mit Python ganz schnell, ganz mühsam. Vor allem, wenn sie dann… Und das ist halt auch das schöne in der objektorientierten Programmierung, dass man die Dinge strukturieren kann. Ich habe da Klassen… Ich habe Methoden… Ich kann die Themen auseinander halten… Das ist meine grafische Klasse… Das ist meine Problemklasse… Wenn ich es auf eine Konsole mache, schmeisse ich die grafische [Oberfläche] weg, aber die Problemklasse habe ich immer noch…

Diese Art der strukturierte Programmierung, wo ich mehrere Klassen habe, oder mehrere Dateien habe… Das wird halt mit einer Skriptsprache dann sehr sehr mühsam… Weil da brauche ich dann wieder irgendeine Entwicklungsumgebung, um das zu zu handeln… Das kenn ich aber dann nicht zu dem Zeitpunkt… Weil ich habe es mit Python im Text-Editor gemacht… Ich muss dann erst recht hinten nach, das alle dazu lernen.

Ich bin bisschen skeptisch, dass das so viel Sinn macht. Es ist natürlich grad sowas man mit C# oder Java anfängt… Das am Anfang diese Visual Studio mit den hunderttausend Menüeinträgen oft mehr verwirrt als hilft…

Wenn man sich auf die wenigen Dingen beschränkt… Und wir haben jetzt eigentlich ein sehr gutes Konzept in der ersten Klasse… Wirklich nur ganz einfach mit `if` und `while` machen… Sonst nichts… Die Erste Klasse… Aber das in allen Schattierungen rauf und runter… Dann macht man Methoden, dann macht man Strings und Arrays. Und dann endet die erste Klasse. Und das ist eigentlich auf einem sehr einfachem Niveau. Das heißt das ganze schwere Gewicht der Objektorientierung, mit Vererbung, Überschreiben/Overloading, und all diese Dingen.

Das kommt erst in der Zweiten, wenn wir uns darüber nicht den Kopf zerbrechen müssen, warum bei einer schließende Klammer davor eine öffnende Klammer stehen muss. Das muss dan schon da sein. Und ich glaube dass man dann halt mit einer Sprache wie Python dann zwar sehr sehr schnell reinkommt… Aber man steht dann auch sehr sehr schnell an. Irgendwo muss dann wahrscheinlich doch wieder auf eine andere Sprache oder eine andere Plattform umsteigen.

Yazicioglu Okay. Dann spielt die Programmiersprache eine Rolle bei der Motivation oder Demotivation oder Begeisterung von Schüler, finden sie?

Lehrer 7 Nein, das glaub ich… Überhaupt nicht. Es ist komplett irrelevant. Das spielen viele andere Dinge mit, aber die Programmiersprache nein, glaub ich nicht.

Yazicioglu Und warum haben sie die Sprache gewählt? Das wollte ich auch fragen, aber das haben sie schon erwähnt. Wegen den zukünftigen Berufsmöglichkeiten…

Lehrer 7 Vor allem das. Genau.

Yazicioglu Welche didaktischen Ansätze sind für sie wichtig? Die Aufgabenstellung soll gut vorbereitet sein oder? Ich weis es nicht… Das werde ich…

Lehrer 7 Mir ist es besonders wichtig, dass die Schüler besonders früh lernen auf eigenen Beinen zu stehen. Das heißt: Sie [bekommen] eine Aufgabenstellung. Sie kriegen alle Hilfsmittel, die sie brauchen. Sie können mir auch Fragen stellen, klarerweise. Aber ich möchte, dass sie möglichst schnell in die Lage kommen selbständig zu arbeiten.

Ich programmiere zwar in der Theoriestunde immer etwas vor, aber ich lege sehr viel Wert darauf, dass sie dann in den Laborstunden sehr sehr selbstständig arbeiten. Sprich sie nutzen dann alle Dinge, die das Internet bietet: Google usw. Weil es kommt dann immer so… Mit zum Beispiel… Wie schreibe ich jetzt auf eine Datei. Die Antwort [bekommen] sie nicht von mir sondern: Schau nach. Such dir das raus.

Weil es wird im Programmieren gibt es nie alles, das ich irgendwo finde. Sondern da muss ich suchen… Da muss ich nachschauen. Da muss ich kopieren… Schauen… Passt das rein. Diese Art vom kreativen Herangehen, dass ist eigentlich das, dass ich möglichst schnell fördern möchte. Nicht das Frontale… Ich programmiere etwas vor. Ihr schreibt das Alles mit. Wenn das passt… Danke schön. Das ist uninteressant.

Yazicioglu Von einem Lehrer hab ich auch gehört, das Programmieren muss man selbst lernen. Ein Lehrer kann das nicht…

Lehrer 7 Absolut. Schauen sie, dass ist wie Auto fahren. Sie können sich ein Buch kaufen über Auto fahren… Das steht drinnen… Aha Kupplung, Gas und Schalten und Lenken ist ja ganz leicht. Und dann setzen sie sich in ein Auto und sagen Fahren… Und sie werden es nicht können. Das ist etwas, dass kann man nur er-üben. Und daran scheitern die auch die im Programmieren scheitern in den ersten Klassen.

Da haben wir doch einige, die gewohnt sind, zu Lernen… Eine Prüfung zu machen und fertig. Obwohl wir das immer wieder sagen… Und das glaubt uns halt nicht jeder: Das ist ein Übungsfach… Da muss man halt jede Woche oder zweimal die Woche sich hinsetzten und eine Stunde programmieren. Das kann man nur… So wie man durch Fahrstunden fahren erlernt… Kann man nur durch Programmierstunden programmieren lernen. Das kann man nicht lernen… Das muss man sich er-üben. Manche kommen dann relativ schnell darauf. Die kommen dann in den Modus rein. Jeder für sich selbst, so wie er es mag…

Und manche die glauben es immer noch nicht. Die lernen im ersten Semester gar nichts und wollen dann im März eine Prüfung machen und kommen darauf: [Das] geht nicht… Das ist leider so… Das ist wirklich ein Übungsgegenstand.

Yazicioglu Haben sie es auch einmal mit Python probiert.

Lehrer 7 Ja, ja ich hab mit Python ein bisschen probiert… Auch im Zusammenhang mit Content Management Systemen… Ja, ich hab auch mit JavaScript programmiert. Ich hab mit PHP programmiert. PHP ist wirklich sehr sehr nett wenn es um Datenbanken geht. Da gibt es sehr viele Libraries. Ich finde diese Sprachen sind alle irgendwo gleich mächtig oder gleich nicht mächtig. Ich sehe jetzt nicht die großen Unterschiede.

Yazicioglu Hat diese Schule einmal mit Python… Haben sie einmal in dieser Schule Python einzusetzen…

Lehrer 7 Also punktuell ja. Es gab immer wieder… Das hängt dann doch immer sehr vom unterrichtenden Lehrer ab… Immer wieder Klassen wo auch Python unterrichtet wurde…

Yazicioglu Wie hat das funktioniert?

Lehrer 7 Es hat sich insofern glaube ich nicht so bewährt… Weil früher oder später kommt dann der Umstieg in die… Indem Fall in die C# Ecke…

Yazicioglu Aha. Nach dem Umstieg gab es soviele Probleme?

Lehrer 7 Ja, die haben extreme Probleme gehabt. Die haben genau das Thema gehabt, dass die also… Ich hab das selbst… In der ersten Klasse… Oder ich hab die erst in der zweiten übernommen. Und die haben in der ersten Klasse Python gemacht… Eine kleine Gruppe. Und die haben wirklich nur im Text-Editor ihr kleines Skript geschrieben und haben ganz ganz kleine Aufgaben gelöst.

Und wie die dann auf einmal gesehen haben in C#. Visual Studio… Und es gibt Klassen… Die waren komplett erschlagen von der Mächtigkeit dieser Maschinerie, die da auf sie zukommt. Die waren halt

gewohnt: Programmieren ist halt so ein bisschen Text schreiben und passt schon. Aber das da viel mehr dahinter steckt.

Die haben da einen ziemlichen Kulturschock bekommen. Und das mag jetzt vielleicht ein Einzelfall sein… Wahrscheinlich kann man es auch anders machen. Aber das ist die Erfahrung die ich gemacht hab an der Schule. Wir machen eigentlich konsequent C# oder die Java-Schiene. Je nachdem wer es unterrichtet.

Yazicioglu Die Probleme [liegen] sowohl bei der Entwicklungsumgebung als auch bei der Sprache selbst. Weil… das Konzept ist ein bisschen anders…

Lehrer 7 Ja

Yazicioglu Die Typisierung in der traditionellen Sprache… Statische Typisierung, aber Python [wo] es eine dynamische [Typisierung] gibt… [Das] versteckte Konzept sieht man nachdem man die traditionelle Sprache… beginnt?

Lehrer 7 So ist es. Ich kann mich erinnern, dass sich die Schüler extrem schwer getan haben in C#. Weil sie auf einmal Variablen deklarieren mussten. Da mussten sie schreiben `int`… Warum muss ich jetzt schreiben `int`… Das ist doch [egal]… Das kann alles sein… Das kann nicht alles sein… Ja, diese Dinge…

Wo man da einfach… Das es so leicht und lässig in der Skriptsprache dann auf einmal wenn man dann eine stark typisierte Sprache hat, wo man wirklich alles… Du bist ein `Integer`… du bist ein `String`… Du bist ein `Boolean`… Okay. Da muss man dann wesentlich präziser sein… Und die haben sich schon ziemlich schwer getan damit. Und all diese Dinge mit Klammern… Und das ist dann auch… Das haben wir nie gebraucht… Jetzt brauchen wir es halt… Das ist halt so…

Yazicioglu Dann sehen sie das als Nachteil… Dynamische Typisierung…

Lehrer 7 Ja.

Yazicioglu Auf der Uni Wien die Lehrer wollen aus diesem Grund, dynamische Typisierung und Mächtigkeit der Entwicklungsumgebung… Compile-Zeit und Laufzeit… Da müssen die Schüler immer auf der Hut sein… In Python ist der aktive Interpreter, die sind sich [dessen] nicht bewusst…

Lehrer 7 Also ich finde das wesentlich besser. Gerade am Anfang. Das die Schüler sich wirklich überlegen müssen: Das ist eine Zahl, deswegen definiere ich eine Integer-Variable und nicht ich deklariere mir quasi einen Mistkübel und da kann ich rein-schmeißen was ich will: Zahlen und Strings. Das ist völlig egal… Ich finde das ist gerade am Anfang ganz wichtig vom Denkansatz.

Ich brauche eine Zahl… Okay ich deklariere eine Zahl. Ich brauche eine Zeichenkette. Okay ich deklariere einen String. Ich glaube das hilft schon ein bisschen. Wie gesagt, der Umstieg vom einen auf das andere ist dann halt ziemlich schwierig.

Yazicioglu Okay. Ich hab alles gehört was ich hören wollte.

Lehrer 7 Okay. Wunderbar.

Lehrer 8

Yazicioglu Würden Sie Python als erste Programmiersprache wählen?

Lehrer 8 Ja…

Yazicioglu Aus welchem Grund?

Lehrer 8 Also grundsätzlich muss man mal sagen, dass es halt relativ viele Programmiersprachen gibt. Warum jetzt Python ausgerechnet Python und wieso nicht irgendeine Andere, z.B Fortran? Die Antwort ist: Das halt im Unterschied zum realen Welt, wo man Gegenstände zu tun hat… Beim Programmieren es viel flexibler ist und viel stärkeren Veränderungen ausgesetzt ist. Okay…

Und ein Zeichen dafür, dass es eben stimmt ist einfach die Tatsache, dass es so viele Programmiersprachen gibt. Weil wenn alles klar wäre, welche Programmiersprache man verwendet oder welche oder

wofür man es braucht… Dann würden sich eine oder zwei Programmiersprache heraus-kristallisieren… Mit denen kann man alles machen. Aber das ist nicht so.

So und jetzt warum soll jetzt Python gut sein für einen Anfänger? Und also ich würde mal sagen: Die Frage will man damit beantworten, zu versuchen, dass man sich die Frage stellen muss, wofür man unterrichtet. Wenn man zum Beispiel Embedded Systems macht, dann ist [es] vermutlich überhaupt keine gute Idee, damit überhaupt anzufangen, Programmieren zu lernen, weil mit Python kann man bei Embedded Systems nicht so weit.

Also das heisst, man muss bereits Programmierverständnis haben, dass man sich dann auf etwas spezialisieren. Wenn man jetzt viel allgemeiner hergeht, man macht nur eine Webseite mit irgendeinem dynamischen Inhalt: Dann ist Python sicher als erste Programmiersprache viel besser… Weil der Weg zwischen der Idee und der Umsetzung ist viel kürzer und man hat alle Tools schon zur Verfügung.

[Es] kommt bisschen darauf an, was man machen will. Und warum es gut für Anfänger ist… Also das leitet sich meine Meinung nach daraus ab, dass in der 80er Jahren hat man als Anfängerprogrammiersprache gesagt… Das es Basic ist, also in den späten 70er Jahren und 80er Jahren… Nur Basic ist halt sehr einfach. Weil es noch auf dieses Lockert-Model zurückgeht und kann nicht besonders viel. Also hat keine großartigen Abstrahierungsmethoden.

Und dann gab es in Holland glaube ich, ein 10 Jahre langes Forschungsprojekt: Was könnte ein Nachfolger von Basic sein? Und das war die ABC Programmiersprache… Und eine Student dort war eben dann derjenige, der Python erfunden hat.

Yazicioglu Nach ABC…

Lehrer 8 Also ja ABC… Das war die Forschungsgruppe… Und Python ist dann ein Projekt gewesen für… Das eigentlich ursprünglich ein Ersatz war für die Bash-Skripte in Unix und Linux. Und ja…

Was ich daraus ableite, dass sehr viel Vorarbeit geleistet worden ist, um auszutesten… Welche Konzepte in der Sprache selber… Von der Syntax funktionieren? Welche funktionieren nicht? Und man ist einfach dann hergegangen und hat Umfragen gemacht. Hat…

Nicht nur aus der eigene Erfahrung gesagt hat: Okay für mich ist es gut, deswegen ist es für alle gut. Sondern hat tatsächlich wissenschaftliche fundiert herauszufinden, welche Konzepte wie gut ankommen.

Eben mit dem Fokus als Einführungsprogrammiersprache. Und daraus leitet sich eben ab, dass es eben eine gute Sprache ist. Denn viele Andere haben nicht diese rigorose vorherige Untersuchung.

Yazicioglu Kann ich diese Dokumente finden, [die zeigen] aus welchem Grund diese Sprache entwickelt wurde?

Lehrer 8 Die ABC-Sprache?

Yazicioglu Python… Gibt es dafür Dokumentation?

Lehrer 8 Ja es gibt vom… Wie heißt jetzt sein Name? Guido Van Rossum… Seine Dissertation und Diplomarbeit…

Yazicioglu Ah. Okay.

Lehrer 8 Das ist glaub ich die einfachste Antwort. Ich weiß nicht genau worum es da geht. Aber ich weis, dass es frühe Interviews gibt von ihm… Aus der Mitte der 90er Jahre, wo er gesagt hat: Das erste Anwendungsziel von Python waren die Skripte am Computer. Also die einfachen Automatisierungen vom System. Das die damit besser gelöst werden können. Also das war der erste Schritt.

Und das was er verwendet hat, war das was er als Student vorher erfahren hat. Also das heißt das was er direkt in der Diplomarbeit stehen hat, das trifft auch auf Python zu, mehr oder minder. Also ich glaub das ist ein… Ich wüsste nicht auswendig ob das jetzt auf der Wikipedia-Seite von ihm steht. Aber zumindest könnte es dort verlinkt sein. Ja…

Yazicioglu Das war ein gutes Argument.

Lehrer 8 Ja und jetzt natürlich die Frage: Python ist ungefähr in den frühen 90er Jahren entstanden und wieso soll es immer noch eine gute Programmiersprache sein? Und der Grund ist eben der, dass es zwei größere Revisionen gegeben hat bislang. Um halt gewisse Atlassen loszuwerden. Weil man sie erweitern wollte und die grundlegenden Konzepten von der Sprachen daran ausgelegt sind, dass es keine Limitierungen gibt. Weil man hat quasi ist Python ein Front-End für C ist und deswegen kann man jede C-Bibliothek, dies es gibt, in Python verwenden.

Also die einzige Schwierigkeit ist nur, wenn es ein eigenes Memory Management hat. Aber das kann man auch lösen. Das heisst es gibt keine Limitierungen, neue Bibliotheken einzubinden, alles unter einen Hut zu bringen… Und das macht sie deswegen halt so flexibel. Und vor allem ist es relativ einfach, das zu machen. Im Unterschied zu vielen anderen Programmiersprachen.

Weil zum Beispiel bei Java, wenn man dort eine externe C Bibliothek hat. Dann ist es eine Riesen-Problem. Weil man hat Overhead, mit dem Speichermanagement, man muss mit der virtuellen Maschine arbeiten. Das ist nur Kopfweh und mühsam. Und bei Python ist aber natürlich auf die Art und Weise dann eine Programmiersprache zu erweitern. Ja…

Yazicioglu Okay. Denken sie für die Anfänger kommt es nicht in Frage… Aber wenn man sich mit dieser Sprache gut entwickeln will, dann [bietet] Python gute Möglichkeiten. Einen Lehrer hab ich gefragt. Er ist eben der Meinung: Für den Anfang ist es gut Python. Aber für spätere Phase, wenn man tiefer [in die Programmierung hineingehen will]… Dann ist es nicht gut, hat er gesagt. Er hat gesagt, dass dafür Java oder C, oder C++ besser [ist]. Weil dort gibt es saubere objektorientierte Konzepte.

Lehrer 8 Also ich würde mal sagen, dass die Objektorientierte Programmierung kam mit Java… Zuerst mit C++ und dann mit Java stark auf. Und das ist so ein Trend aus den 90er Jahren. Und der Grund warum das gut ist ist, dass früher ein Programm in verschiedenen… Also der Entwicklungszyklus war relativ lang. Weil man musste ein Programm schreiben. Wenn es größer wurde in mehrere Teile teilen. Jedes wurde separat kompiliert. Und die verschiedene Arbeitsgruppen in einer Firma müssen sich koordinieren.

Und dafür gibt es zum Beispiel bei Java die Interfaces mit den… Das ist quasi… Die Contracts zwischen den einzelnen Modulen des Programms… [Das die] zusammenpassen… Und bei C++ ist der Vorläufer davon diese ganzen Header-Files. Die sind ja nichts besonderes… Allerdings sind sie nur dafür um die ganze Signaturen der Funktionen und Methoden zu definieren… Um eben über [einen] größeren Bereich hinweg Libraries zu erzeugen.

Was man jetzt aber ich sag einmal ab Beginn der 2000er Jahre hat ist, dass die Computer um viele Größenordnungen schneller sind. Und dadurch: Wenn man ein Programm entwickelt, braucht man nicht mehr die Zerteilungen in Arbeitsgruppen. Und verschiedene Bereiche… Und eine Organisation rund herum.

Das heisst, alles was mit objektorientierten und statischen Typen zu tun hat tritt in den Hintergrund. Und im Vordergrund dreht sowas wie Continuous Integration und Unit-Testing. Also zum Beispiel viele die dann sagen: Ja, Java hat eine tolle objektorientierte Architektur. Die verstehen zum Beispiel nicht, dass es extreme Limitierungen gibt wegen den statischen Typensystemen und das zum Beispiel Klassen selber keine Objekte sind oder Funktionen. Jetzt in den neuesten Java-Versionen versucht man in der Richtung zu arbeiten aber weder Funkionen noch Klassen sind Objekte.

Und die statischen Typisierungen werden nur mit den Interfaces gelöst. Aber die ganzen Sachen… C++ ist da noch schlimmer, weil ganzen Altlasten von C noch drinnen sind. Wo man dem Typensystemen selbst überhaupt nicht mehr ganz Herr wird, wenn es komplizierter wird.

Also die… Also man hat viele… Vieles, dass die Sprache Java komplex und vielschichtig macht, ist dafür da, um Probleme zu lösen, die in der Sprachen selber entstanden sind. Weil es notwendig war früher.

Yazicioglu Sehr gut. Danke.

Lehrer 8 Und den Durchblick den man in Python bekommt ist, dass eben die Objektorientierte Programmierung viel weniger ist. Viel wichtiger ist es dynamisch auf neue Anforderungen sich anzupassen. Das die Dokumentationen viel besser funktioniert. Also, dass man zum Beispiel Test, Dokumentation und den Code selber als Gesamtpaket betrachtet. Und das im Continuous-Integration verwendet. Und das funktioniert da viel besser…

Und da wird jeder beipflichten… Weil, wenn ich eine Änderung in einer Python Programm mache und parallel dazu die Tests laufen lasse [dann] ist die Zeit viel kürzer als bei jeder anderen kompilierten Sprache.

Und das Andere ist auch dass die… Der Zeitaufwand geringer ist, wenn man ihm irgendwelche Veränderungen machen will… Das leitet sich daraus ab… Also ich bin… Also ich sehe überhaupt keine Mehrwert darin, großartige objektorientierte Programmierung zu unterrichten, weil sie eben ausgedient hat.

Yazicioglu Diese sauberen objektorientierte Konzepte finden sie, dass das in Java noch besser [ist]? In Python finde ich [ist] das auch gleich.

Lehrer 8 Also ich… Vom den grundlegenden Konzepten… Also alles was man mit Java machen kann, das funktioniert auch im Python.

Yazicioglu Eigentlich finde ich in Python viel besser weil das Interface am Anfang hab ich das nicht so… Warum brauche [ich] ein Interface…

Lehrer 8 Ja genau. Das ist dafür da um verschiedene Bibliotheken… Man erzeugt zum Beispiel am Anfang fünf verschiedene Bibliotheken für einen Teil eines große Programms und die werden jeweils von verschiedene Arbeitsgruppen gemacht. Die haben sich vorher ausgemacht, wie schauen jetzt die `public`-Methoden aus. Damit man jetzt vom Hauptprogramm die einzelne Komponenten aufrufen kann. Aber dadurch das… Das ist etwas vom früher… So arbeitet ja jetzt keiner mehr.

Sondern entweder es gibt dann eine klare Bibliothek, die eh nicht mehr verändert wird. Oder man hat ein großes Programm, wo gleichzeitig an allem gearbeitet wird. Das ist halt die Realität. Und deswegen hat dieses Konzept ausgedient.

Umgekehrt kann man auch sagen, dass es in Python… Es gibt das Meta-Programming… Also dass heisst, dass alle Klassen Objekte sind, [dadurch] können sich dynamisch zur Laufzeit erzeugt und verändert werden. Und in dem Layer drinnen kann man sehr wohl das Konzept von Interfaces einbauen. Weil man einen Interceptor über alle Klasse schreiben kann, der eben sagt, dass etwas von bestimmter Art sein soll. Also man kann dynamisch Klasse erzeugen, was man in Java nicht mal könnte.

Also zum Beispiel ein gutes Beispiel ist Django. Das ist ein Web-Framework. Und dort gibt es diese Models um mit der Datenbank zu kommunizieren. Und bei den Models definiert man zum Beispiel, das ist eine ID, das ist eine Name, das ist eine Email-Adresse. Und damit die einzelnen Zeilen, die aus SQL-Datenbank herauskommen zu Objekten werden, muss ich natürlich dahinter einer Klasse geben die sagt: Das eine Feld ist die ID, das Andere ist eine Name, das Andere ist die Telefonnummer oder sowas.

Yazicioglu Stimmt…

Lehrer 8 Genau… Und diese Klasse wird zur Laufzeit erzeugt und macht… Also das Programm startet… Die Klassen werden generiert anhand von den Definitionen. Und dann beim Laden des Objekts oder beim Speichern, werden die ganzen Typen überprüft. Und so etwas äquivalentes… Es gibt in Java Hibernate zum Beispiel, aber das ist ja viel komplizierter dort…

Weil man ja alles da im Code zuerst mal definieren muss. Und es passiert nicht dynamisch.

Yazicioglu Ja. Man soll viel Allocation-Annotation machen… Viele Verbindungen machen…

Lehrer 8 Ja genau, die Annotations… Ich mein, dass ist ein bisschen ein anderes Konzept. Das ist halt… Ja ich weiß jetzt nicht genau wie man das sagen soll, dass da der Unterschied ist. Es ist auf jeden Fall viel mehr Overhead das auf die Art zu machen.

Yazicioglu Das ist ein gutes Beispiel

Lehrer 8 Aber es ist zum Beispiel etwas, was jemand, der sich viel mit Java beschäftigt überhaupt nicht versteht. Warum eine Klasse eine Objekt ist. Und warum es dynamisch erzeugt werden kann. Das funktioniert einfach deswegen, weil eine Idee hinter Python kommt von Smalltalk.

Das ist eine Programmiersprache, wo eben dieses Meta-Programming mehr oder minder vorgezeigt wurde. Wie das funktioniert. Also jeder, der sich mit Objektorientierter Programmierung beschäftigt, sollte auch wissen was das ist. Aber das bei jemand, der Java unterrichtet niemals vorkommen, weil Java das nicht kann. Also das heißt…

Yazicioglu Java kann nicht… Für Java… In Java ist es unterschiedlich. Klassen und Objekte… Und in Python, das ist…

Lehrer 8 Naja, in Python ist es auch unterschiedlich. Nur gibt es drei Stufen: Da gibt es das Objekt als Instanz einer Klasse, und eine Klasse als Instanz von einem Typ. Und dann gibt es eine Hierarchie von Typen. Und der Grundtyp ist eben Objekt. Und wenn man jetzt eine Klasse… Oder eine Klasse ableitet, dann kommt die immer oben vom Grundtyp des Namens Objekt. Deswegen sagt man wenn man eine Klasse definiert: `class A(object)`. Aber statt Objekt dort kann etwas anderes stehen, dass auch ein Typ ist. Und das ist Meta-Programming. Und in Java gibt es nur das Keyword `class`. Was man überall verwenden muss.

Yazicioglu Mhm. Aber das ist nicht so bequem.

Lehrer 8 Ja, das ist im Endeffekt so etwas als würde man einen strukturierten Datentyp in C erzeugen mit bestimmten Feldern, die halt Typen haben. Okay schön. Und dazu gibt es ein paar Methoden.

Yazicioglu Verwendet Java nicht Polymorphismus oder Generics um das zu erweitern?

Lehrer 8 Ja. Aber Polymorphismus gibt es ja auch in Python mehr oder minder. Ich meine der ist halt dann automatisch da. Un diese Generics, dass sind ja dynamische Typen. Das liegt ja einfach darin, dass man so ein strenges Typenkonzept hat. Man möchte aber dieses strenge Typenkonzept ständig brechen.

Weil… Man möchte. Also quasi wenn ich eine Methode schreibe, die für viele Anwendungszwecke dienlich sein soll. Zum Beispiel ich möchte sie für alle möglichen Listen und Sets und sonstigen Sachen verwenden können. Dann muss ich ja dafür ein Interface verwenden. Und drinnen kommen ja wieder Typen vor. Und diese Typen, wenn ich die irgendwie spezifizieren will, um zum Beispiel am Namen etwas vergleichen zu können. Also vom Namen eines Attributs von den Objekten, die drinnen in der Collection sind…

Dann brauch ich dafür auch wieder ein Interface, dass mir sagt, dass es eine Methode gibt um den Namen zu bekommen. Und dafür gibt es dann dies dynamischen Typen. Also das ist das Problem. Und bei Python gibt es das nicht. Weil da sagt man entweder… Entweder sagt man, man kann eh darauf zugreifen immer… Oder ich hab halt Tests, die alle Fälle austesten, falls das ein Problem ist. Also falls es das Attribut nicht gibt, sehe ich das es ein Problem ist, weil der Test nicht funktioniert.

So, das heißt, da ist der Ansatz ein anderer. Oder wenn man ein bisschen komplexer sein will, dann erzeugt man sich eben so eine Meta-Klasse und eben so einen Meta-Typ, der eben mir sagt was dieses Namens-Attribut sein soll, wenn es nicht definiert ist.

Yazicioglu Bei einem Attribut würde das immer gehen?

Lehrer 8 Ja, also je nachdem wie kompliziert man es haben will. Also die Ganze. Also viele… Man muss immer zurückdenken: Welche Probleme gibt es? Warum sind die entstanden? Was sind die Lösungen dafür? Ich würde jetzt nicht sagen, dass eine Sprache besonders einfacherer als die Anderen ist. Ich würde sagen, dass es bei Python so ist, dass viele Probleme gar nicht existieren.

Und wenn man es unbedingt haben will, dann gibt es schon Lösungen dafür: Die sind halt viel komplizierter… Also viel komplizierter als wenn man nur ein Anfängerbuch von Python liest. Und das geht jetzt natürlich wieder zurück. Warum ist es eine gute Einführungsprogrammiersprache? Weil, diese ganzen Advanced-Konzepte kann man weglassen. Und man arbeitet trotzdem relative gut damit. Während man hingegen bei Java weglasst, was eine Klasse ist, dann kommt man nich mal bis zum „Hello World".

Yazicioglu Finden sie das didaktisch gut am Anfang… Manche Lehrer machen am Anfang… Die Java-Lehrer… Später werden wir erklären… Was `public` ist… Was `static` ist. Und [wir werden] nur drinnen etwas schreiben was [das] Programm machen soll. Ist es gut… Didaktisch gut?

Lehrer 8 Also ich muss sagen, ich hab mehrere Jahre lang Java für Anfänger unterrichtet… Und hab immer wieder das selbe Problem gehabt, dass man eben sagt: Ja, schreib zuerst mal alles in `public static void main` hinein. Und später erklären wir dann, was das ist. Und ich finde, dass es extrem schlecht ist.

Yazicioglu Ah, gut das zu hören.

Lehrer 8 Und zwar ich kann das auch begründen. Deswegen… Weil erstens man muss etwas unterrichten. Man muss Studenten irgendwie etwas sagen, dass sie etwas tun soll, was sie nicht verstehen. Und wenn man es versucht ihnen zu erklären, ist es quasi ein Art von Kreis. Also irgendwie ein Zirkel von Abhängigkeiten, sodass man niemals zu einer Lösung kommt.

Wenn man sagt okay. Was ist das? Aha dafür eine Erklärung und das, was ich erklären soll, wird jetzt wieder verwendet, um das zu erklären, was ich erklären sollte. Und da kommt man nicht weiter.

Yazicioglu Ja. Ein Beispiel: Wenn man ein File lesen will in Java: Dann muss man Streams… Soviele Konzepte auf einmal…

Lehrer 8 Also gerade dieses `StreamReader` und `StreamWriter` und `BufferedWriter`. In dieser Verkapselung. Das ist ja eine nette Idee, aber eben glaub ich auch aus den… Ich glaub, das ist eine Idee aus dem ursprünglichen Buch der objektorientierten Programmierung. In Java… Was weiß ich… 1.1.1 ist es dann umgesetzt worden.

Das man eben diese verschachtelten Konstruktoren hat, um Funktionen in einer Klasse zu erzeugen. Weil da hat man ja eine Instanz im Konstruktor vom Nächsten… Und das wieder im Konstruktor vom Nächsten. Das ist halt so ein Muster, wie man objektorientierte Programmierung machen kann. Und aus meiner Sicht ist das ein Paradebeispiel wie man es nicht machen soll.

Und selbst wenn man mit Leuten redet, die die Java-API geschrieben haben. Die sagen, dass die das jetzt nie wieder machen und das es ein Riesen-Fehler war das zu tun. Also da gibt es Interviews wo man sich das anschauen kann.

Yazicioglu Sehr gut…

Lehrer 8 Und zum Beispiel gibt es diese… Das ist irgendwo… Hab das halt vor Jahren gesehen. Aber das ist halt eben ein Paradebeispiel dafür, wo man gedacht hat: Cool das könnte man alles machen… Das eine greift auf das andere zu, und dass ist dann so schön abstrahiert. Aber in der Praxis kommt man eben darauf, dass es ein kompletter Humbug ist. Weil das ist… Also ein…

Ein lustiges Beispiel ist ja, dass es von Google eine Java-Library gibt… Ich glaub Guava hat die früher geheißen… Und da drinnen kann man ja auch Dateien lesen. Und da wird dann einfach gesagt: Ja Datei-Lesen und standardmäßig wird halt ge-buffered. Oder wenn man es nicht macht. Also so wie in Python üblich hat man halt ein paar Attribute mit… Also ein paar Argument, die man an den Konstruktor oder die statische Methode die es erzeugt übergibt. Drinnen wird halt entschieden, was passieren soll.

Und der Standardfall ist genau das was man braucht. Und man muss jetzt nicht drei, vier verschiedene Klassen in irgendeiner Weise miteinander verschachteln. Also was überhaupt keinen Sinn macht. Also… Es wurde sogar repariert nachträglich… Weil man dann quasi eine ganz andere Library verwendet, wo die API viel besser durchdacht ist. Also deswegen so etwas überhaupt unterrichten zu müssen ist nur mühsam.

Yazicioglu Okay.

Lehrer 8 Ja… Also das ist… Hausgemachte Probleme. Und es hat in der Praxis überhaupt keinen Sinn. Und es ist mittlerweile auch überholt. Und es gibt Lösungen um das zu vermeiden. Also, das zu machen ist… Ich mein es ist… Wenn es jetzt noch unterrichtet wird [ist] es halt noch typisch akademisch: Man hat irgendwas zu erklären, was in der Praxis irrelevant ist. Und darüber kann man dann schön reden. Es nützt halt nichts. Die einfachste Lösung ist immer die Beste.

Yazicioglu Okay. Unterrichten sie Python auch?

Lehrer 8 Ja, ich hab jetzt im Sommersemester 2015 hab ich einen Python-Anfängerkurs gehabt. Da auf der Uni.

Yazicioglu Was für einen Fehler, oder Probleme haben die Schüler gehabt. Oder wenn sie [einen] Vergleich machen [zwischen] Java und Python unter den Anfängern. Welche Sprache haben die Anfänger besser, leichter oder begeisterter [aufgenommen]?

Lehrer 8 Der Unterschied ist das das ganze Curriculum gestellt wurde… Und also um das ein bisschen genauer zu erklären: In den Jahren vorher gab es eine zweistündige Vorlesung und eine Stunde Übung im PC-Labor. Das heisst, es war halt eine klassische Vorlesung, wo also ein Skriptum, ein Frontal-Unterricht passiert ist.

Wo man natürlich aus meiner Sicht beim Programmieren nichts lernt. Denn wenn man nur zuhört, funktioniert es nicht. Und dann ist halt eine Stunde im PC-Labor irgendwie über Beispiele geredet worden. Und dadurch das es nur eine Stunde ist, ist relative wenig weitergegangen, weil sie… Weil quasi viel zu wenig Zeit war, um die Beispiel im Detail durchzugehen. Und man hat halt 13 bis 14 Einheiten, die es in einem ganzen Semester gibt. [In denen] haben wir ungefähr 18 bis 20 Beispiele geschafft.

Yazicioglu In Python oder in Java.

Lehrer 8 In Java. Und das ist halt sehr wenig. Und vor allem deswegen schwierig, weil die Einführungsbeispiele am Anfang, also die erste Hälfte. Um nur die grundlegende Kontrollstrukturen zu erklären… Sind einfach extrem mühsam… Weil man muss fünf, sechsmal wiederholen: Warum man da überhaupt `public`, `static`, `void`, `main` hinschreibt.

Einmal nur ein Strichpunkt falsch… Einmal unten Klammern beim herum-probieren falsch… Da sind also sehr viele Probleme passiert, also die lächerlich erscheinen, aber das ist halt fundiert darin, dass die Programmiersprache schlecht ist. Und dann haben wir halt auf Python umgestellt. Und das im Curriculum hat sich so geändert, dass es keine Vorlesung gibt. Sondern stattdessen 3 Stunden PC-Labor.

Also das heißt: Es gibt überhaupt keine Frontal-Unterricht mehr. Im Endeffekt verwirrt das natürlich die Studenten. Weil es ist niemand mehr da, der ihnen erklärt, wie es eigentlich geht. Und sie haben ein Skriptum und können Fragen stellen im Endeffekt und müssen Beispiele vortrage.

Das hat aber den Vorteil gehabt, dass sie… Ein bisschen auch mein Lehrkonzept ist, dass man die Leute zuerst verwirren muss. Damit sie dann verstehen. Damit sie selber anfangen sich damit zu beschäftigen und nicht erwarten, dass jetzt vorne jemand steht, der ihnen Buchstabe für Buchstabe sagt, was sie hinschreiben müssen. Genau.

Yazicioglu Für den Programmierunterricht muss man [selber etwas] mache…

Lehrer 8 Genau. Und deswegen sind sie halt viel stärker gezwungen gewesen das selber zu machen. Im Endeffekt ist daraus hinausgelaufen, dass wir an die 80 Beispiele in der selben Zeit gemacht haben. Und die Schwierigkeit, war vielleicht… Von dem was gemacht wurde in den Beispielen, ist zwei bis drei mal so hoch gewesen wie bei Java.

Also Programme die mehr konnten… Komplexere Aufgaben lösen… Und man hat doch viele Themen umfasst, die man vorher gar nicht gehabt hat. Also fünfmal der Stoffumfang… Zwei- bis dreimal so viel von den Komplexität. Und eben viermal so viele Beispiele insgesamt.

Das heisst, es wurde viel mehr gemacht. Und ich habe ein viel stärkeres Gefühl gehabt, dass diejenigen, die am Anfang noch nicht so überzeugt waren, was ihnen das überhaupt bringt, am Schluss dann schon motiviert waren, das alles doch zu verstehen.

Und das war bei Java überhaupt nicht der Fall. Und das liegt einfach darin, dass das Spektrum viel breiter war an den Themen, die man machen kann… Also mit weniger Vorwissen kann man viel mehr Themen abdecken, wenn man einfach die Libraries zur Verfügung hat, die [es] unter Java gar nicht gibt. Also wenn man sich jetzt mathematische Inhalte und so anschaut.

Yazicioglu [Ist es] am Anfang wichtig den Schülern beizubringen wie man algorithmisch denken soll. Wie man programmieren soll. Dieses Konzept mit nicht so viel Overhead…

Lehrer 8 Ja… Ich weiß nicht… Bei der Vorlesung ging es halt um Mathematiker…

Yazicioglu [Also] OOP-Konzepte haben sie [nicht gemacht]?

Lehrer 8 Nein, also OOP-Konzepte… Es gab Klassen und Instanzen davon. Und ableiten… Aber das ist nur ein Nebenschauplatz. Also das Hauptziel hab ich [so] verstanden, dass… Wenn ich jetzt Daten irgendwo habe, oder irgendeine Fragestellung jetzt… Also… Und allgemein irgendwelche Formeln oder irgendwelche Daten und ich will das verarbeiten. Was ist dann ein einfacher Weg um auf die Lösung zu kommen?

Also das waren dann solche Themen wie: Ich hab eine Funktion und ich möchte die Funktion ableiten und wissen an welcher Stelle ist die… Ist dann eine Nullstelle um einen Tiefpunkt zu finden. Also klassische Schulbeispiele.

Und in Python ist es relativ einfach zu sagen: Okay so definiere ich eine Funktion. So wird sie abgeleitet. Jetzt wird sie Null gesetzt. Und dann plotte ich das. Das sind weniger als zehn Zeilen Code. Das ist relativ schnell verstanden. Und das kann man in den ersten zwei Wochen machen.

Aber man sieht in den ersten zwei Wochen bereits, dass man schon relativ viel machen kann. Oder ein Beispiel am Schluss… Weiß jetzt nicht, was wir alles gemacht haben.

Aber zum Beispiel von einer Webseite einen JSON-Datenfeed herunter zu lesen für eine Wetterprognose. Da gibt es eine Open-Weather-API. Und diesen JSON-Datenfeed dann auslesen um zu sagen, an welchen Tagen die Sonne scheint oder wann es regnet. In dem man einfach in den Strings die darin sind sucht „sun“, „rain“ oder sonstige Schlüsselwörter und dann zurückrechnet welches Datum das ist.

Also das heißt Datenverarbeitung… Wie lade ich etwas von einer Webseite runter? Wie parse ich einen JSON-Objekt, so dass ich es in Python bearbeiten kann, und wie iteriere ich dann über diesen verschachtelten Datentyp. Was sind jetzt Listen? Was sind Key-Value-Assoziierungen. Wie analysiere ich jetzt [bei] einzelnen Strings was da drinnen steht. Und wie mach ich daraus irgendwas sinnvolles. Sind auch wieder nur zehn Zeilen an Code.

Yazicioglu In Java sind das glaube ich mehr als zehn Zeilen.

Lehrer 8 Ich glaube auch. Und dadurch… Ja es ist halt mehr so ein spielerischer Zugang. Aber wenn man diese ganzen einzelnen Konzepte dann eben zusammenbaut kommt man relativ schnell zu ganz interessanten Lösungen.

Yazicioglu [Sind] die Schüler zu diesen Aufgabenstellungen gekommen in Java?

Lehrer 8 Nein, nein.

Yazicioglu Auf diesen Punkt sind sie nicht gekommen. Denn in Java verlängert dieser Prozess, die…

Lehrer 8 Ja, in Java bräuchte man sicher zwei Semester schätze ich.

Yazicioglu Sehr gutes Argument. Das mag ich.

Lehrer 8 Also mein… Vor allem ich wüsste gar nicht wie das geht. Weil man kann ja einfach nur in einem Semester eine bestimmte Anzahl von Stunden abverlangen von den Studenten. Und die Beispiele können… Also jedes Beispiel kann nur bis zum bestimmten Länge… Bis zum bestimmten Schwierigkeitsgrad gegeben werden… Also so dass es für 60, 70 Prozent von den Studierenden machbar ist. Und häufig scheitert es daran, dass eben Vieles ein Vorwissen braucht, das nicht abgedeckt ist. Das man sich selben schwer bearbeiten kann.

Und bei Python ist es viel einfacher. Weil, mir haben dann alle gesagt… Also viele haben mir dann gesagt im Endeffekt. Ja, da gibt es ein Skriptum, dass brauch ich aber eigentlich eh nicht, weil ich tipp es dann halt einfach bei Google ein. Und ja das ist genau das Lernziel. Denn ein durchschnittlicher Programmierer in der Industrie schaut ständig im Internet nach, was er jetzt gerade macht. Also das selber zu machen ist wichtig. Das man das auch lernt.

Yazicioglu Entdeckendes Lernen sollte man fördern…

Lehrer 8 Genau, das ist dann eh genau das Konzept gewesen, dass ich gemacht habe. Und sie haben sich dann gewundert, warum sie das überhaupt machen. Aber okay. So ist es halt in der Praxis. Und ein objektorientiertes Konzept hätte hier nichts gebracht bei dem Beispiel. Denn sie verwenden es ja nur. Aber was müssen sie jetzt wissen, welche Patterns es jetzt dann gibt. Das ist sinnlos.

Yazicioglu Okay. Im ersten Semester haben die Schüler [die] objektorientierten Konzepte gelernt?

Lehrer 8 Ja. Klassen und Klassen ableiten. Was, wenn man eine Methode hat und dann leitet man die Klasse ab und die Methode hat den selben Namen. Dann wird die Methode überschrieben. Wie kann ich trotzdem auf die Methode von der Elternklasse zugreifen. Solche Sachen… Das wars dann schon.

Yazicioglu In Java ist… Im ersten Semester haben die Schüler das nicht gelernt?

Lehrer 8 Ja, In Java nicht. Weil da war. Ich mein da haben wir… Da war quasi das Limit, dass man überhaupt eine eigene Klasse schreibt und dann zwei Instanzen von der Klasse hat. Und die irgendwie verwendet.

Yazicioglu Die Schüler hatten [vorher] keine Kenntnisse vom Programmieren?

Lehrer 8 Ja.

Yazicioglu Wie alt waren die Schüler? 18?

Lehrer 8 Ja, sie waren jetzt im zweiten Semester.

Yazicioglu Okay.

Lehrer 8 Ich würde mal sagen sie waren ab 18… Aber es gibt auch trotzdem welche, die waren vielleicht 50. Also… Es ist dann halt jeder drinnen, der mit dem Mathematik-Studium anfängt.

Yazicioglu Eigentlich meine Zielgruppe ist ab 14. Es ist, wenn es selbst mit der Schule ab 18 schwierig ist, wie…

Lehrer 8 Also ich würde sagen… Also ah, ab 14.

Yazicioglu HTL, erste Klasse ist meine Zielgruppe. Mein Betreuer ist in der HTL ein Direktor, und er will unbedingt Python einsetzten. Da bin ich auch [dieser] Meinung. Aber alle anderen Lehrer wollen Java, oder C++, oder C#.

Lehrer 8 Also das kommt vielleicht auch ein bisschen darauf an: Welche HTL es ist, denn wenn sie z.B.… Ich hab ja keine Ahnung. Wofür soll denn die Programmiersprache verwendet werden? Wenn es zum Beispiel Programmieren von Robotern ist, dann braucht man diese… Also irgendeinen Industrie-Roboter programmieren… Dann braucht man ja irgendeine C-Programmierungssprache, die dann zu diesen Assembler-Code kompiliert wird.

Und da verstehe ich, dass man mit Python vielleicht nicht weit kommt. Aber wenn jetzt nur allgemein Programmierung erklärt werden soll…

Yazicioglu Ja, allgemein.

Lehrer 8 Ja, dann ist… Ich würde sagen Python hat den Vorteil gerade in einer HTL, weil die ganzen… Es geht nicht darum um die Objektorientierte Programmierung. Sondern mir es geht darum, dass es Bibliotheken gibt für ganzen numerische Berechnungen. Zum Beispiel HTL assoziiere ich mich mit Integrieren, Differenzieren, Simulieren, Statistik und Visualisieren. Also das heißt: Wie z.B. mache ich jetzt einen Report als PDF-File, wo dann irgendwelche Datenanalysen oder Simulationsberechnungen drinnen sind.

Yazicioglu Dann finden sie das gut, dass die Schüler mit mathematischen Berechnungen am Anfang programmieren sollten? Mit dieser Analogie aus der Mathematik?

Lehrer 8 Ja, das ist eine guter Einstieg. Ich weiß jetzt nicht genau… Mit vierzehn, da hat man ja noch nicht… Integrieren und Differenzieren. Aber Gleichungssysteme und Funktionen, dass ist schon unterrichtet worden.

Ja und es ist auch ziemlich. Sagen wir so: Die einfachsten Beispiel, die ich mir da überlegen kann: Man geht her schaut sich an, was ist im Schulstoff bislang vorgekommen. Was weiß ich… Das Volumen eines Quaders.

Da kann man halt die Formel hinschreiben. Und später kann man dann sagen, da schreiben wir eine Klasse Quader. Die wird mit a, b und c instanziert. Und jetzt sag: Berechne mittels eine Methode das Volumen und die Oberfläche aus. Ja, das sind die einfachen Beispiele, wie ich anfangen würde.

Yazicioglu Okay. Ja…

Lehrer 8 Das ist inhaltlich nichts Neues. Aber das Programmieren ist das Neue. Und wenn dann die selben Ergebnisse rauskommen, dann versteht man es auch relative gut.

Yazicioglu Das wird die Schüler mit Programmierung vertraut machen. Weil die kennen schon die Erkenntnisse aus der Mathematik. Die wollen [das] in der Programmiersprache umsetzen.

Lehrer 8 Ja, also das man einen Anknüpfungspunkt hat. Natürlich kann man das auch in C#, Java oder C machen. Also, das ist halt nicht so lustig dann…

Yazicioglu Stimmt, weil in Python gibt es [einen] Interpreter. [Da] kann man schnell alles dort schreiben, in der Python-Konsole. Dort Funktionen schreiben und die Ergebnisse…

Lehrer 8 Also was wir im… Also was ich vorgegeben habe, im gesamten letzten Semester, war dieses IPython-Notebook. Und da hat man eben keine Konsole. Sondern man hat eben ein Web-Interface. Das ist den meisten auch vertrauter.

Yazicioglu Ah, Entwicklungsumgebung…

Lehrer 8 Nein, keine Entwicklungsumgebung. Das ist einfach ein Notebook, wo man Zellen hat… Wo man eine oder mehrere Zeilen an Python-Code hineinschreibt. Dann klickt man auf „auswerten“, also mit Shift-Enter. Und wenn es ein Ergebnis gibt… Also ein print oder ein Objekt als letztes in der letzten Zeile stehen bleibt, dann wird das zurückgegeben.

Und dann kann man die Zeile ändern oder die Zelle ändern, auswerten und dann kommt wieder eine andere Antwort zurück. Und das ganze Sheet kann man speichern und das ist dann gleichzeitig auch die Abgabe. So, dass heißt, genau das was die Studenten abgeben mit dem ganzen Code und den Ergebnissen, bekommt man dann als Übungsleiter 1:1 zurück.

Und das ist zum Beispiel, wenn man ein… Wenn man ihnen sagt sie sollen ein Skript schreiben, oder an irgendeiner anderen Datei arbeiten, dann sieht man nicht die Ergebnisse waren, die die Studenten gesehen haben. Und in dem Fall werden die Ergebnisse mitgespeichert.

Yazicioglu Wie heißt das nochmal?

Lehrer 8 IPython-Notebook. Also das ist… Oder jetzt dies verallgemeinert für eine Programmiersprache, da heißt dieses Projekt Jupyter. Also das ist… Und mit dem würde ich auch anfangen zu unterrichten. Und das hab ich jetzt ein Semester lang gemacht. Und es hat viel besser funktioniert, als ich dachte. W

Weil… Also solche zum Beispiel… Alleine die… Einfach das Plotten von Funktionen, und das man irgendwelche Bilder hat, die man verarbeitet. Und dafür dann Ergebnisse sieht. Und man kann dann eben Inhalte von Zellen kopieren und mit mehrere Varianten gleichzeitig arbeiten. Und man hat eben einen globalen State im Hintergrund, den man immer wieder verändert. Das ist genauso wie in der Konsole. Allerdings nicht diesen linearen Input/Output-Stream. Sondern man kann eben die Sachen dann besser vergleichen. Also das ist sehr lehrreich gewesen.

Yazicioglu Und dynamische Typisierung finden Sie [ist ein] Nachteil?

Lehrer 8 Also das… Also ich… Es ist bei größeren Projekte natürlich ein Nachteil. Und wenn man sich anschaut, was sind die letzten Entwicklungen bei Python… Python 3.5, was jetzt finalisiert wird. Das hat Annotation, damit man überhaupt sagen kann, welche Typen zum Beispiel bei den Signaturen sind… Oder bei einzelnen Variablen… Und jetzt gibt es dann auch Tools dafür, um das statisch überprüfen zu können… Oder bei den Test, dass es halt als Fehler vorkommt.

Lehrer 8 Das heisst, ja in der Richtung gearbeitet… Aber das ist hauptsächlich deshalb, weil es bei großen Projekten notwendig ist. Was ich viel wichtiger finde ist, dass es eben nicht dieses statische, sondern dieses Strong-Typing gibt bei Python. Das ist viel wichtiger. Strong-Typing heisst, dass man zur Laufzeit jeder Zeit wissen kann, welchen Typ hinter eine Variablen ist.

Yazicioglu Okay. Das kann man in Python sehr leicht aufrufen, fragen…

Lehrer 8 Ja, in Python kann man immer sagen `isinstance` und `type`. Und da bekommt man zu jedem Objekt zurück. Also jedes Objekt selber, weiß von welchem Typ es ist. Und bei Java und C ist das ja nicht der Fall. Die haben ja nur weak typing.

Yazicioglu Okay.

Lehrer 8 Das heisst, man hat eine Liste und dann in spitzen Klammern von Type Objekt und man gibt dann zum Beispiel String hinein. Und dann holt man es wieder zurück. Dann sind alle Strings vom Type Objekt. Und man hat keine Ahnung mehr, dass es eigentlich ein String sein soll.

Aber in Python kann das nicht passieren. Wenn ich da eine Liste habe, und ich gebe Strings hinein und ich nehme ein Objekt heraus. Dann kann dazwischen serialisieren, also auf der Festplatte speichern. Das Programm beenden. Ich kann es neu laden. und nach wie vor weiße ich welchen Typ, die einzelne Objekte haben.

Einfach weil es diese Strong-Typing gibt. Das die Typinformation dabei bleibt. Das ist natürlich ein Overhead ja. Weil da ist natürlich immer zusätzlicher Pointer mitgespeichert, der sagt, was das jetzt sein soll.

Aber also [die] Ausführungszeit meiner Programm ist schon lang kein Problem mehr. Weil wenn jemand sagt, ich hätte gerne aber eine Programmiersprache, die halt besonders schnell funktioniert: Dann sage ich halt: Ja in Python gibt es genug Tools, um dort wo es notwendig ist, das Programm schnell zu machen.

Also das eine ist, wenn man den Code in Fortran schreibt, was so ziemlich das schnellste ist… Und dann gibt es das F2Py um jede Fortran…

Yazicioglu Web2Py… F2PY…

Lehrer 8 F2PY… Also jedes… Man muss ein Modul mit Subroutinen in Fortran schreiben und das kann man direkt in Python verwenden. Das geht für die ganzen numerische Mathematik-Bibliotheken. Das schnellste, was man machen kann… Dann gibt es JIT-Compiler, wie dieses Numba… Da braucht man nur ein paar Type-Annotierungen machen und man hat bis auf paar Prozent, die Geschwindigkeit die man auch mit C haben kann. Und gleichzeitig kann man aber zum Beispiel als Compiler-Target die Nvidia-Grafikkarten haben.

Dann hat man einen JIT-Compiler, der plötzlich hundertmal schneller ist als C. Nur weil man mit Grafikkarte arbeiten kann. Und trotzdem schreibt man nach wie vor Python-Code hin. Dann gibt es noch Cython. Das ist eben ein Python nach C- oder C++-Transpiler, um Python Module zu schreiben, die in C kompiliert werden. Wenn man größere Objekte schreiben will.

Und da ist die Syntax nach wie vor so gut wie in Python. Also das heisst man hat Palette von Tools, wo man dann wenn man es braucht, auch die Geschwindigkeit hat… Wenn man mal weiss wie das Programm geht… Vorher hat man das Prototyping gemacht in Python… Man weiss so ist es richtig, und dann den Weg um den Code zu beschleunigen ist relativ klein, wenn man weiss, was man tut. Das ganze Programm in C oder in C++ neu zu schreiben, ist noch Riesen-Overhead, der da nichts bringt.

Yazicioglu Und das wollte ich noch fragen: Okay. Für die Auswahlkriterien… Welche Kriterien sind für Sie wichtige bei dem Auswahl einer Sprache für Anfänger? [Sind] zukünftige Berufsmöglichkeiten oder didaktische Ansätze wichtig? Oder…

Lehrer 8 Also ich würde sagen, eines der wichtigsten Kriterien für mich war, dass es zum Studium passt. Also bei Mathematik-Studium, das ich eine Programmiersprache haben will, die sehr gut in Mathematik ist. Ja, also ich möchte… In den ersten zwei Wochen, war es möglich Gleichungen zu lösen, Funktionen zu differenzieren, Funktionen zu plotten, ohne überhaupt zu wissen, was ein Objekt ist.

Das war relativ natürlich das hinzuschreiben. Also das heisst, man hat ein sehr schnellen Einstieg in relative Komplexe Zusammenhänge oder im Hintergrund funktioniert relative viel Komplexes. Allerdings man sieht gleich, dass es eine Sinn macht mit der Programmiersprache zu arbeiten. Weil die Features da sind, und später kann man sagen: Okay wie funktioniert die Sprache eigentlich wirklich oder warum funktioniert das jetzt so. Und dann kann man einzelne Konzepte erarbeiten.

Yazicioglu Finden sie das auch wichtig für die Informatik-Bereich mit Python zu beginnen. Oder mit… Sollen…

Lehrer 8 Also Informatik ist für mich jetzt… Da ich ein Informatiker bin weiß ich halt nur, was ich so höre. Aber da gibt es halt einerseits diese Systems-Engineering-Ecke. Und ich glaube, dass jeder, der dort anfängt bereits Programmieren kann. Das ist mein Verdacht. …Oder der dann die ersten zwei Semester zumindest übersteht. Da ist es vielleicht am Besten mit C zu arbeiten.

Aber dann gibt es vielleicht Wirtschaftsmathematik. Und da hab ich gehört, dass es da welche gibt, die als einzige Programmiersprache mit PHP arbeiten. Kann sein. Das heißt in Wirtschaftsinformatik würde ich mal ganz klar sagen, da sollte Python unterrichtet werden. Warum auch immer.

Aber das minimale Argument, dass man versteht wie eine gute Programmiersprache ist. Und in der Sprache selbst, kann ich von Webseiten bis Datenbankschnittstellen alles relativ einfach machen. Und es ist auch relativ einfach zu erlernen.

Und dann gibt es noch den Didaktik-Bereich. Da würde ich auch sagen, dass es Python als Sprache ist, die man auch gleich im Unterricht verwenden kann. Und dann… Was bleibt dann noch in der Informatik übrig.

Dann gibt es noch die Forschung, da würde ich sagen… Also quasi Computer-Science, also in der Richtung wie man Programmiersprachen macht. Da hat es natürlich nur einen Sinn vielleicht mit C++ anzufangen, oder irgendwelche funktionalen Sprachen, also… Aber das ist dann deswegen weil es dann die wissenschaftlichen Ecke sozusagen [ist] und nicht mehr die anwendungsbezogene Ecke.

Yazicioglu Okay.

Lehrer 8 Und dann gibt es noch, wenn ich sage anwendungsbezogene Ecke… Da gibt es noch die medizinische Informatik.

Yazicioglu Medieninformatik, auch Medieninformatik…

Lehrer 8 Ja ich… Da weiß ich jetzt leider nicht, was die Anwendungen sind… Aber das wird dann… Ich sage mal, da ist Python sicher auch ein guter Einstieg, weil man Daten verarbeiten kann. Also man hat kein Geschwindigkeitseinbuße, wenn man numerische Mathematik macht: Was in der medizinischen Informatik vermutlich vorkommt. Statistische Datenanalysen…

Man hat dieses R Statistikprogramm. Da gibt es gute Schnittstellen von Python nach R. Oder man kann R direkt auch unterrichten, falls das notwendig ist. Und von Medieninformatik hab ich keine Ahnung.

Yazicioglu Okay. In dem Bereich Medieninformatik und Netzwerktechnik ist Python empfehlenswert.

Lehrer 8 Bei Netzwerktechnik, das glaube ich. Weil da gibt es extrem gute Libraries für die ganzen Schnittstellen. Also zum Beispiel wenn man sich die Wettbewerbe zum Hacken… Also es gibt so Wettbewerbe um sich gegenseitig zu hacken. Oder man schaut sich an, diese 0-Day-Exploits, die veröffentlicht werden. Also wenn Sicherheitslücken in Software gefunden wurden…

Zum Beispiel das Flash-Plugin funktioniert nicht richtig. Wenn man sich anschaut was ist dann der Beispiel-Code um in einem Webbrowser eben das Flash-Plugin auszuhebeln. Da sieht man eben meistens, dass der in Python geschrieben ist. Einfach, weil es ist kein Problem irgendwelche Routinen zu schreiben die halbwegs performant die ganzen Bitmasken richtig über das Socket-Interface schicken. Man kann das dann relativ schnell prototypen, und man hat es geschafft.

Yazicioglu Okay. Und [die] letzte kurze Frage: Welche… Bei welchem Konzept haben die Schüler die meisten Probleme am Anfang. Bei Variablenzuweisung, Schleifen, oder bei den objektorientierten Konzepten, oder bei der Fehlersuche? Oder es ist alles…

Lehrer 8 Also ich sag mal es ist ein bisschen individuell. Also Debugging, da ist das erste Problem… Das ich überhaupt bevor ich sag was das Problem ist, sag ich ihnen: Lest einmal was überhaupt die Fehlermeldung ist. Und es ist quasi die Überwindung plötzlich die Fehlermeldung zu lesen und zu erkennen: Da ist nicht nur ein Fehler, und der ist irgendwo. Sondern das steht dann dabei in der und der Zeile. Und der Text ist immer wieder ein anderer.

Das heißt man muss den Text interpretieren und verstehen können. Aber das ist einfach Erfahrung. Und da muss man dann einfach… Es aktiv machen und hundert oder tausendmal mit dem Kopf gegen die Wand fahren. Und dann versteht man die Fehlermeldung zu lesen. Man muss immer nur sagen: Lies die Fehlermeldung und dann sickert es langsam hinein. Also das ist ein natürlicher Lerneffekt. Aber der ist am Anfang eine Schwierigkeit.

So etwas wie eine Variable, dass kann man relativ gut unterrichten. Und da gibt es ja auch Papers. Wenn man sich anschaut… Zum Beispiel: Ich weise der Zahl 5 `x` zu, dann schreib ich hin `y = x` und dann sagt man: Welchen Wert hat `x`? Welchen Wert hat `y`. Und zum Beispiel manche Studenten glauben, wenn man hinschreibt `y = x`, [dann] wandert der Fünfer vom `x` zum `y`. Und `x` ist Null oder `x` ist undefiniert. Aber in Wahrheit ist sowohl `x` und `y` eine Fünf. Und ja okay… Und das muss man eben gut unterrichten. Dann haben es auch so gut wie alle verstanden.

Yazicioglu Da [spielt die] Sprache [keine Rolle]?

Lehrer 8 Also das Problem ist glaub ich in jeder Sprache und es hilft dann nicht zu sagen: Ja hinter der Variablen sind ja nur Referenzen auf die Objekte. Weil sowohl der Begriff Referenz als auch der Begriff Objekt hat keinen Sinn für die Studenten.

Deswegen fange ich an zu erklären: Das ist dann halt eine Box in der etwas drinnen ist. Und der Box gibt man einen Namen. Und Es kann auch zwei Namens-Zettel geben. Also das man mit irgendwelchen bildlichen Erklärungsversuchen eine Analogie herstellt die richtig ist.

Und nicht die Studenten dann alleine gelassen werden mit einer Analogie, die falsch ist. Da muss man ziemlich genau anleiten und schauen welche Beispiele man macht. Die Schleifen, also ja… Das ist ein Riesen-Problem. Und ich weiß auch leider nicht warum.

Yazicioglu Da unterscheiden die… Wo unterscheiden [sich] die Sprachen [hier]?

Lehrer 8 Naja

Yazicioglu Es ist von der Logik so elementar…

Lehrer 8 Naja ich glaub bei den Schleifen ist es so, dass man es entweder versteht, und man weiß nicht, warum man überhaupt ein Problem hatte. Oder man hat irgendwas falsch verstanden, und dann weiß man halt nicht als Unterrichtender wo das Problem ist.

Yazicioglu Okay.

Lehrer 8 Aber das hat glaub ich jeder immer. Und solche objektorientierten Sachen… Das sind dann halt meistens das die Definition, was jetzt tatsächlich im Detail passiert die fehlen, oder man hat irgendwie ein Detail falsch verstanden wieder. Ich würde sogar sagen Schleifen sind jetzt schwieriger als das ableiten von Objekten.

Yazicioglu Ah

Lehrer 8 Ja, wirklich. Also zumindest in Python ist es die Erfahrung gewesen. Weil intuitiv wenn man jetzt sagt… Zum Beispiel man stellt die Frage: Kann eine Eltern-Klasse von einem Kind etwas erben oder kann eine Kind-Klasse von einer Eltern-Klasse etwas erben?

Yazicioglu Das ist logisch.

Lehrer 8 Also jeder der sagt, dass Kind erbt etwas von den Eltern hat recht. Jeder der sagt, die Eltern erben etwas vom Kind ist automatisch falsch. Wenn man schon die richtigen Wörter verwendet, dann sind auch die Antworten zu den Fragen richtig. Und von dem heraus in der Praxis… Wenn man es erkennt, oder sieht woran man es erkennt, dann passt alles zusammen. Aber bei Schleifen da ist es viel mysteriöser anscheinend.

Yazicioglu Okay.

Lehrer 8 Oder auch umgekehrt… Man gibt ja nicht Beispiel und sagt hier musst du eine Schleife verwenden, du musst die Variable x nennen und jetzt mach das. Sondern man gibt dann zum Beispiel ein Beispiel, wo es günstig wäre eine Schleife zu schreiben. Aber die Studenten machen es nicht, weil sie nicht daran denken. Sondern kopieren dann eine Zeile zehnmal untereinander hin.

Oder zum Beispiel [wenn man] ihnen sagt: Du kannst eine Liste verwenden. Und eine Liste kann man indizieren. Und die kann man dann locker länger machen. Dann schreiben sie trotzdem hin: `x1`, `x2`, `x3`, `x4`, `x5`… Also wie erklärt man letztendlich, dass man ein bestimmtes Konzept verwenden soll? Das Konzept erklären reicht ja nicht: Weil es muss auch die Anwendung irgendwie erzwungen werden.

Yazicioglu Okay.

Lehrer 8 Aber das war in Python auch kein Problem. Denn da sagt man einfach, so ist es nicht ganz richtig. Da kommt das Richtige heraus, aber das kann man nicht akzeptieren. Dann schaut man sich die Lösung vom Nächsten an. Da passt es. Dann hat es jeder verstanden.

Yazicioglu Okay. Danke schön.

Lehrer 8 Hat mir Freude gemacht. Ich hab in Java für private Firmen Projekte realisiert. Also, dass ich denen ein GUI-Programm gemacht hab. Ich hab in Fortran für Forschungsprojekte irgendwelche Optimierungen gemacht.

Yazicioglu Okay, sie sehen den Unterschied ganz klar und deutlich…

Lehrer 8 Ja

Lehrer 9

Yazicioglu Welche Sprache würde Sie als Unterrichtssprache bevorzugen?

Lehrer 9 Ist praktisch in der Form nicht zu beantworten. Weil man sehr klar unterscheiden muss: Was man erreichen will… Was man von den Schülern erwarten… In einer HTL wird z.B. ganz anders ausschauen als in eine AHS. Und auch da muss man schon unterschieden, in einer AHS.

Da gibt die Leuten, die haben irgendwie so ein Spezial-Modul, [indem] sie weiter unterrichtet werden, noch für den ganzen Rest der Schulzeit. Andere kommen nur ganz kurz für ein paar Wochen in Berührung mit dem Programmieren. Sonst gar nicht mehr. Und entsprechend unterschiedlich muss da die Sprache auch sein.

Yazicioglu Okay. Meine Zielgruppe ist… [Ich fokussiere mich] auf HTL, technische Schule, Handelsakademie…

Lehrer 9 Also eine HTL… Wenn es natürlich eine einschlägige HTL ist, also Informatik oder so was. Da bin ich bei Python ein bisschen skeptisch. Python ist eine wunderbare Sprache, wenn es darum geht so die erste Schritte rein zumachen. Das kann die Sprache perfekt. Und genau deswegen auch macht [sie] es bisschen schwieriger den zweiten Schritt zu gehen.

Wenn man dann wie soll ich sagen… Die Sprache ist relativ sauber bei ganz einfachen Dingen: Wenige Konzepte… Bietet gute Möglichkeiten für die Standardlösungswege. Insofern hat sie viele Vorteile.

Yazicioglu [Bei] Listen zum Beispiel gibt es sehr viele Möglichkeiten, damit man die Probleme schneller löst als in Java.

Lehrer 9 So ist es. Genau das sind die Vorteile wenn man eben rein-schnuppert in das Programmieren. Genau diese Konzepte machen später dann aber Probleme. Weil man irgendwie auch gelernt hat, alles in diesen wenigen Sprachkonstrukten zu lösen.

Wenn man z.B. in Richtung Datenstrukturen gehen will… Ist es schon nicht mehr so einfach, weil ja, man ist strikt gebunden an das was die Sprache direkt vorgibt an guten Möglichkeiten. Wenn man andere Möglichkeiten finden will, gibt es die auch in Python… Man kann natürlich alles programmieren… Aber das ist dann eine deutlich schwierigere Sache. Weil kein Mensch mehr einsehen will, warum er was anderes machen soll, als diese einfachen Konzepten die eben von vorne herein da sind.

Also dieser zweite Schritt macht es ein bisschen schwieriger für Leute die wirklich programmieren. Also später dann beruflich programmieren… Die das zum Lebensunterhalt verwenden… Wie gesagt andere Leute, die nebenbei ein bisschen programmieren… Irgendwo rein-schnuppern… Für die ist das eine wunderbare Sprache.

Yazicioglu Für [die] späteren Ziele ist Python eine wunderbare Sprache…

Lehrer 9 Wunderbare Sprache für Leute…

Yazicioglu Aber am Anfang…

Lehrer 9 Nein, nein so würde ich es auch nicht sagen. Später muss man natürlich die Sprache wählen. Es gibt natürlich unterschiedliche Applikationen… Wenn man schnell irgendwas macht, so als „Glue“-Sprache ist Python natürlich eine gute Sprache auch in der Praxis.

Nun muss man dann unterscheiden, was wirklich das Ziel des Unterrichts ist. Wenn man als Ziel hat, Leute kurz einmal heran zu führen an das Programmieren. Auch dafür ist Python eine wunderbare Sprache.

Yazicioglu Aber es ist nicht geeignet für HTL…

Lehrer 9 Wenn man vorhat viel tiefer einzudringen, dann kommt dann eben ein schwieriger Schritt, nachdem man Python ein bisschen gemeistert hat. In anderen Sprachen wie Java, da muss man viel früher in diese schwieriger Konstrukte einsteigen, und damit hat man diese zweite Hürde eben nicht so stark.

Man ist früher gezwungen, an diese komplizierten, komplexe Konzepten zu denken.

Yazicioglu [Mit] dieser zweiten Hürden meinen Sie den noch schwierigeren Level?

Lehrer 9 Es ist einfach so, Python macht den ersten Einstieg eben sehr einfach. Und aus genau dem Grund… Wenn man über die Möglichkeiten eben, die Python an der Oberfläche bietet hinausgeben will… Da sieht man die Konzepte nicht. Die Konzepte sind versteckt in Python.

Yazicioglu Wie dynamische Typisierung?

Lehrer 9 Dynamische Typisierung… Es ist halt eine dynamisch typisierte Sprache. Das sehe ich weniger als großes Problem. Andere dynamisch typisierte Sprachen sind da

Yazicioglu Perl…

Lehrer 9 Perl ist einfach alt… Sagen wir halt einmal Ruby ist auch relativ populär.

Yazicioglu Ah, „Ruby on Rails“…

Lehrer 9 „Ruby on Rails“ ist eigentlich keine Sprache, sondern ein…

Yazicioglu Web-Applikation, Web-Anwendung…

Lehrer 9 So ein System für Webanwendungen. Bei Ruby steckt halt dahinter… Da sind die Konzepte eher sichtbar. Also auch die komplexeren Konzepte sind leichter sichtbar. Man wird nicht so… Zum Beispiel mit List-Comprehensions gefangen. List-Comprehensions sind zwar wunderbar… Aber vor lauter Vereinfachung dieses einen Konzepts sieht man halt nicht mehr drüber. Das es andere Dinge auch gibt, Dinge, wo die List-Comprehension eben nicht mehr funktioniert.

Yazicioglu Okay. Wenn die Schüler mit einer anderen Sprache programmieren müssen, dann beginnen…

Lehrer 9 Dann beginnen Sie die eher in Konzepten zu denken, die deutlich über das hinausgehen, was die Sprache eben an der Oberfläche bietet. Und dieses Denken in Konzepten wird rausgeschoben, wenn man Python verwendet.

Yazicioglu Also ich [bin] von Java zu Python… Zuerst hab ich Java gelernt und nachher Python. Python war für mich leichter, schneller, schöner…

Lehrer 9 Klar. Glaube ich ihnen gerne.

Yazicioglu Umgekehrt kenn ich mich nicht aus.

Lehrer 9 Ich meine es ist natürlich auch schwer zu sagen, wo die Leute genau hängen. Wo sie Probleme haben. Aber normalerweise ist es nicht die Syntax-Ebene. Ist es nicht die einfach Ebene, wo Python eben sehr, sehr…

Yazicioglu Wo haben die Schüler die meisten Probleme? Syntax, Semantik, Fehlersuche oder Konzepte, logische… objektorientierte Konzepte?

Lehrer 9 Es sind natürlich die Konzepte. Weil da OOP dabeisteht. Allerdings das sind schon die fortgeschrittenen Konzepte, die nicht mehr so relevant sind am Anfang. Semantik z.B kann man locker erklären. Und das kapieren die Leute auch.

Aber die Pragmatik dahinter, werden sie nicht [verstehen]. Also warum verwenden wir eine bestimmte Semantik. Da stehen eben die Modelle dahinter. Die Modelle sind nicht einfach. Die sind ziemlich komplex.

Man kann aber nicht so vorgehen, dass die Modelle zuerst erklärt und dann halt die Sprachen dazu. Weil die Modelle an sich noch weniger verständlich sind, wenn man nicht schon mit einer Sprache reingeht. Das ist irgendwie einfach so… Vom Unterricht her… Man probiert es schön langsam aufzubauen… Schön langsam kriegt man hoffentlich das Model, das dahinter steckt, mit…

Yazicioglu Okay können sie vielleicht ein Beispiel dazu geben?

Lehrer 9 Ein sehr einfaches Konzept ist die Variable. Man kann zwar locker sagen: Eine Box… Ist es ein Kästchen oder irgendwas. Eine Schublade… Solche Analogien greifen aber nur zu einem bestimmten Grad. Irgendwann bricht diese Analogien zusammen.

Yazicioglu Da spielt die Programmiersprache selbst eine Rolle?

Lehrer 9 Da spielen die Programmiersprache natürlich eine kleine Rolle. Weil die Abstraktionen eben unterschiedliche weit gehen. Das was z.B bei Python immer wieder so als große Werbe-Botschaft hat ist die Mehrfachzuweisung.

Also das man mehrere Variablen gleichzeitig zuweisen kann.

Yazicioglu Alles wird als Objekt gesehen?

Lehrer 9 Naja das hat weniger mit Objekten zu tun. Sondern… Ja… Ich kann einfach zwei Variablen links hinschreiben, zwei Werte rechts und die werden parallel zugewiesen. Damit kann ich z.B. leicht Werte vertauschen.

Yazicioglu Okay

Lehrer 9 Ja das macht einiges einfacher. Das Problem dabei ist nur, man wird davon abgehalten, dass man genau diese Problematik hinter der Reihenfolge von Anweisungen sieht.

In der Vertauschung in anderen Sprachen ist es relativ komplex. Weil man da mal wissen muss, da braucht man eine Hilfsvariable.

Yazicioglu Ja in Python gibt es keine Hilfsvariable.

Lehrer 9 Genau in Python braucht man keine Hilfsvariable. Nun kann man das jetzt verallgemeinern. Python hilft da sehr stark einen Swap einmal zu implementieren. [Das ist] sehr einfach. Aber man lernt nicht dieses Konzept, das da eine bestimmte Reihenfolge in den Zuweisungen drinnen sein muss.

Weil etwas was das in eine Schachtel liegt muss solange in der Schachtel liegen bleiben, bis man es wirklich braucht. Also bis die letzte Verwendung eben da ist. Und man darf erst überschreiben, nachdem man den alten Werte nicht mehr braucht.

Das sind die Konzepte die da bisschen verloren gehen.

Yazicioglu Dann, das mach eine Hürde…

Lehrer 9 Das macht eine Hürde später dann.

Yazicioglu Und gibt es noch Hürden…

Lehrer 9 Macht es am Anfang aber viel einfacher. Ja das sind so kleine Beispiele.

Yazicioglu Vielleicht ist es nicht für HTL geeignet [sondern] für HAK?

Lehrer 9 Na also ich würde sagen, die meisten Leute sollten eher Python programmieren. Nur gerade nicht die fachspezifischen, also die Informatiker, die wirklich die Konzepte dahinter bis ins Letzte anschauen müssen.

Für die finde ich eine weniger gut geeignete Sprache. Weil man da viel früher schon auf die essentiellen Konzepte kommen muss.

Yazicioglu In amerikanischen Universitäten… In sehr vielen Universitäten und Schulen, technischen Schulen [wird] auf Python umgestiegen.

Lehrer 9 Stimmt, Python ist dort sehr verbreitet. Allerdings auch dort so, das es bei den Major… Also auf den Universitäten, zumindest bei den Major-Studien nicht so verbreitet ist. Major heisst eben, die Informatik-Studenten… Während die Maschinenbauer, Elektrotechniker oder so durchwegs Python lernen. Für die ist auch sinnvoll.

Yazicioglu Okay. Für wissenschaftliche Sprachen ist es sehr gut geeignet.

Lehrer 9 Naja, das würde ich so nicht sagen. Also das hat weniger mit wissenschaftlich zu tun. Sondern es ist einfach die Sprache für Leute, die ab und zu ein bisschen etwas programmieren.

Yazicioglu Aha. Und nicht…

Lehrer 9 Ganz allgemein…

Yazicioglu Ohne Kenntnisse…

Lehrer 9 Die schon ein [wenig] Kenntnisse brauchen. Natürlich… Aber die nicht so total tief einsteigen. Die bleiben normalerweise auf der Ebene von Python stecken. Also das was Python gut kann, ist für diese Leute in der Regel ausreichend.

Yazicioglu Ah. Okay, jetzt verstehe ich.

Lehrer 9 Sie brauchen diesen zweiten Schritt nicht mehr machen, der dann kompliziert ist.

Yazicioglu Okay. Sie wollen nicht [ein] Programmierer sein…

Lehrer 9 Sie wollen [kein] Programmierer sein, sondern programmieren nebenbei ab und zu eine Kleinigkeit. Da ist Python eine wunderbare Sprache ja. Wenn man eben diesen zweiten Schritt nicht braucht. Wenn man auf diese tiefere Konzepte verzichten kann, die eben Python gut versteckt.

(Diskussion über nicht relevantes Thema)

Yazicioglu Okay. Diese Frage [haben sie] in gewisser Weise schon ein [wenig] beantwortet. Welche Kriterien, Ansätze, Merkmale berücksichtigen sie bei der Auswahl eine Programmiersprache im Unterricht. Ist [die] zukünftige Berufsmöglichkeit wichtig oder die didaktische Eignung?

Lehrer 9 Es ist nicht an erster Stelle, die zukünftige Berufsmöglichkeit. Aber man behält es im Hinterkopf. Also es spielt schon eine gewisse Rolle bei der Auswahl. Weniger deswegen, weil die Leute das dann im Beruf brauchen so direkt. Sondern einfach… Wie soll ich sagen…

Das ist auch für die Schüler interessant, wenn sie was Reales lernen… Nicht ein Spielzeug… Sie wissen gleich, das ist was real, dass sie wirklich verwenden können. Das ist eben kein Spielzeug, sondern etwas echtes. Da sind sie viel motivierter.

Yazicioglu In Python gibt es Libraries, mit denen man sehr gut Spiele programmieren kann. [Ist es] für den Anfang für die Schüler auch empfehlenswert, zuerst mit der Spiel-Programmierung zu beginnen? Da ist es auch motivieren finde ich.

Lehrer 9 Es ist eine Herangehensweise ja. Aber da muss man auch nicht auf der Ebene von Python bleiben. Es gibt andere Systeme auch, die eine ähnliche Herangehensweise haben. Eher die dynamische Sprachen sind auf der Linie, also wenn man Smalltalk anschaut. Squeak ist genau auf dieser Linie.

Yazicioglu Ich weiß nicht ob Smalltalk objektorientiert ist…

Lehrer 9 Ja ist objektorientiert. Das ist das Paradebeispiel für eine objektorientierte Sprache. Sagen wir mal aus der Sprache stammen die meisten objektorientierte Konzepte.

Yazicioglu Aha. Okay. Finden Sie, dass eine Sprache die alle Paradigmen [unterstützt] gut ist?

Lehrer 9 Nein, das spielt eigentlich gar keine Rolle. Die didaktische Eignung ist natürlich eine wichtige Sache, nur schwer zu greifen.

Yazicioglu Bei manchen Artikeln hab ich gelesen: [Es] ist gut, dass eine Sprache alle Paradigmen [unterstützt]. Damit man Spielraum hat. Das man imperative, funktional und objektorientiert Programmieren kann. Aber…

Lehrer 9 Das finde ich nicht so wichtig. Weil man kommt ja in Wirklichkeit nicht sehr weit am Anfang.

Yazicioglu Finden Sie wichtig, dass man zuerst mit imperativen Paradigmen zu beginnen oder ist es egal?

Lehrer 9 Es gibt unterschiedliche Ansätze, die nicht unbedingt klare Vorteile haben gegenüber den Anderen. „Object first" war eine zeitlang populär.

Yazicioglu Gibt es Artikel dafür?

Lehrer 9 Inzwischen weiß man das das nicht so ideal ist. Ja, imperative beginnen, ist einfach sehr bewehrt. Also haben sehr viele Leute gemacht und man weiss einfach wie das geht. Worauf man aufpassen muss…

Lehrer 9 Funktional beginnen, das machen immer mehr… Hat immer kleinere…

Yazicioglu Gibt es Probleme beim funktionalen Paradigma, damit man später…

Lehrer 9 Nein, da gibt es keine Probleme. Das funktionale Paradigmen würde ganz gut passen am Anfang. Es muss natürlich eingebunden sein in ein größeres Konzept. Bei uns ist das ein Probleme bisschen… Wir können nicht so direkt sagen, wir fangen jetzt mit dem einen an und dann muss ganze Studium darauf aufgebaut sein.

Yazicioglu Ja, da gibt es die Abhängigkeiten…

Lehrer 9 Ja, da gibt es sehr viele Abhängigkeiten drinnen. Aber funktionale Programmierung ist sicher gut geeignet für den Anfang… Imperative auch…

Yazicioglu Ich hab mit Herr Stefan … gesprochen. Er unterrichtet Processing. Und er ist eben der Meinung mit [dem] imperativen [Programmierparadigma] zu beginnen ist sehr gut.

Lehrer 9 Ja, ja

Yazicioglu Und Processing ist wie Python aber mit Java-Version im Hintergrund…

Lehrer 9 Ja genauso ist es. Also Processing ist einfach eine vereinfachte Variante von Java mit der Möglichkeit sehr schnell irgendwelche Applikationen zu schreiben. Für Leute… Also einfach so als Motivation… Wo die Leute gerne arbeiten.

Yazicioglu Das sehe ich so wie Python…

Lehrer 9 Ja, aber Python ist in der Beziehung anders. Wie soll ich sagen. Processing ist nicht so sehr eine Programmiersprache, als mehr ein Werkzeug für die Entwicklung.

Yazicioglu Okay.

Lehrer 9 Die Sprache die dahinter steckt ist eigentlich Java und es ist dann sehr leicht zusätzliche Konzepte von Java einzuführen. Wenn man halt in diese Richtung geht.

Yazicioglu Ich hab mit einem Lehrer in der HTL gesprochen. Sie unterrichten auch Processing. Manche Unterrichtseinheiten werden mit Processing gehalten. Er hat gesagt es ist so sehr gut gelaufen. Erfolgreich gewesen…

Lehrer 9 Also Processing es ist… Ich weiß nicht kennen sie es…

Yazicioglu Nein, selbst hab ich nicht…

Lehrer 9 Es ist eine Anwendungswerkzeug um solche Videos und solche Dinge schnell einbinden zu können. Und damit sofort irgendwas machen zu können. Eher eine Bibliothek hinter Java, die auch die Grundelemente von Java als „Glue-Sprache" verwendet. Als Glue-Sprache wie man eben auch Python verwendet. Das man fertige Teile zusammenbindet. Dafür ist eine Glue-Sprache… Dafür… Also es ist eine sehr spezialisierte Glue-Sprache. Spezialisiert hat auch den Vorteil, dass man jetzt [nicht] so irrsinnig viel drüber wissen muss. Man kommt wirklich schnell rein.

Yazicioglu Das heißt das Ziel ist damit man schnell…

Lehrer 9 Man kommt sehr schnell rein und kann sehr schnell Applikationen schreiben, die nicht primitiv sind. Sondern die wirklich etwas Gröberes machen…

Yazicioglu Finden sie die didaktische Eignung, die didaktischen Ansätze sind schnell…

Lehrer 9 Ja, ja… Man darf das nicht verwechseln jetzt. Das sind unterschiedliche Ebenen. Processing ist für den ersten Einstieg sehr gut. Aber wenn sie das jetzt mit Python vergleichen. Wenn man mit Python anfängt, muss man schon wesentlich tiefer reingehen, bevor man erste vernünftige Sachen machen kann, von einer halbwegs großen Komplexität. Wenn man in Java sowas machen will, muss man noch viel tiefer reingehen. Also das sind ganz unterschiedliche Ebenen.

Yazicioglu Mhm verstehe… Bei [der] Auswahl [einer Programmiersprache für den Unterricht] welche didaktische Ansätze [sind für sie] wichtig? Das man [einen] schnellen Einstieg hat oder alles…

Lehrer 9 Aus allem möglichem… Man muss halt auch bisschen Rücksicht nehmen auf die Kundschaft sozusagen. Also darauf was die Schüler schon wissen, oder was sie am Ende wissen müssen.

In einer HTL… Wenn man wirklich fünf jahre programmiert… Da kann man schon davon ausgehen, dass die Leute einen ganz andere Einstieg haben. Die sind wahrscheinlich sehr technische orientiert. Die wissen wie ein Prozessor funktioniert bis ins letzte Detail. Das kommt einfach am Anfang…

Da muss man ganz ganz anders anfangen, als bei Jemanden wie sie gesagt haben. Der am ende MATLAB programmieren soll und wahrscheinlich keine Ahnung vom physischen Aufbaue eines Rechners hat.

Also würde ich sagen, auf HTL-Ebene… Also mit dem hardwarenahen Ansatz ist eher eine hardwarenahe Sprache auch besser geeignet… Also C und C++ ist sicher nicht eine schlechte Wahl in dem Bereich. Obwohl es… Obwohl man sagen müsste jemand der jetzt MATLAB als Ziel hat für den ist das was Unnötiges. Also man muss sehr stark darauf eingehen, welche Arten von Leute man unterrichtet und eben was das Ziel ist.

Java ist so eine Mittel-Ding, das passt fast immer. Das hat C Syntax, also man kann irgendwie auch C sehr schnell lernen, wenn man mal Java hat. Und umgekehrt [kann] man auch sehr weit in der Abstraktion gehen. Also das ist wahrscheinlich auch einer der Gründe warum Java so populär geworden ist. Das passt fast immer irgendwie ein bisschen, aber passt auch nirgends perfekt.

Yazicioglu Nach Java ist mir C++ nicht so schwer vorgekommen. Nachdem ich Java gelernt habe…

Lehrer 9 Das glaub ich. Umgekehrt aber auch. Das würde auch relativ leicht gehen.

Yazicioglu Warum wählt [diese] Universität Java aus? Haben sie vielleicht Python probiert?

Lehrer 9 Python würde ich nicht wählen. Python finde ich für uns [keine] gute Sprache.

Yazicioglu Also für die Universitäten?

Lehrer 9 Nicht allgemein Universität, sondern speziell für Informatik. Und gerade wegen dieser zweite Hürde. Ob man Leute zuerst sozusagen in einer falschen Sicherheit wiegt: Okay ihr könnt ja programmieren und dann kommt aber die große Keule. Und diese Keule, die sollt man vermeiden

Yazicioglu Und… Eigentlich haben sie das schon gesagt: Spielt die Programmiersprache selbst eine Rolle bei der Motivation, Erfolg und Begeisterung?

Lehrer 9 Na klar. Sicher.

Yazicioglu oder Misserfolg

Lehrer 9 Ja natürlich auch. Programmieren lernen müssen eigentlich alle Schüler selbst. Als Lehrer hat man in Wirklich wenig Möglichkeiten. Ich kann Ihnen grad ein paar Aufgaben vorgeben. Aber ja, man kann niemanden Programmieren beibringen. Jeder muss selbst lernen. Also…

Yazicioglu Damit ist [die] Programmiersprache selbst [wichtig]?

Lehrer 9 Die Sprache selbst ist wichtig. Also vor allem die Motivation. Die Motivation muss von innen kommen. Mit den extrinsischen Formen der Motivation kommt man nicht wirklich weit. Das ist vielleicht einer der wichtigen Unterschiede, zwischen Programmieren lernen und irgendwas anderes lernen. Eigentlich kann man ja die selben Unterrichtstechniken überall einsetzen. Programmieren lernen hat nur den Unterschied, dass man sehr schnell sieht was die Schüler können.

Yazicioglu Ja vielleicht ist am Anfang auch nicht so gute Wahl mit Python zu unterrichten. Das der Schüler sieht: Ja ich kann programmieren. Aber im späteren…

Lehrer 9 Ja und Nein. Wie gesagt: Es kommt auf das Ziel an. Wenn das Ziel eben etwas ist, was man mit Python selbst gut erledigen kann. Da ist Python sicher auch für den Unterricht perfekte Sprache. Wenn das Ziel eben das anderes ist… Wenn es wesentlich tiefer geht, als das was Python so anbietet… Dann würde ich gar nicht mit Python beginnen, und dann umsteigen… Sondern würde ich gleich in die Richtung einsteigen, in die es dann am Enge gehen soll.

Yazicioglu Dann muss ich mir das Ziel noch klarer machen… Danke

Lehrer 9 Bitte. Also wunderbar gelöst in Python… Solange halt man nur die Anfangs-Dinger braucht. Wenn man tief in die Sprache einsteigen will hat man genau das falsche gelernt.

Yazicioglu Automatische Typumwandlung, da hat man das Problem…

Lehrer 9 Es ist sowohl in Java als auch in Python ein Problem. Das da im Hintergrund soviel passiert, das man eh nicht abschätzen kann. Und wo man dann das Gefühl kriegt man hätte was verstanden, was in Wirklichkeit ganz anders funktioniert.

Yazicioglu Okay.

Lehrer 9 Also diese geheimen Dinge. Diese Magie hinter der Sprache. Das man das auch braucht. Ist für den ersten Einstieg manchmal hilfreich. Aber in Wirklichkeit lernt man ein Konzept nicht. Das Konzept wird verschleiert.

Yazicioglu Okay. Sie finden es nicht so gut, das verschleiern… Die Schüler sollen alles am Anfang lernen.

Lehrer 9 Nein, nein, nicht alles am Anfang lernen. Aber wenn reale Probleme da sind. Dann soll man die nicht verstecken.

Yazicioglu Okay.

Lehrer 9 Man muss ja nicht weitergehen. Man kann auch die ersten Beispiele so aufbauen, dass man, auch wenn alles strikt kontrolliert wird, ohne explizite Typumwandlung auskommen. Also das ist keine echte Hürde. Es ist nur so eine Verschleierung von Komplexität. Die manche Leute gerne haben. Aber wenn man eben über diesen Punkt hinausgeht, [dann] hat man ein wichtiges Konzept nicht verstanden.

Yazicioglu Okay.

Lehrer 10

Yazicioglu Wenn Sie einen Vergleich unter der Programmiersprache machen würden, wie wäre ihre Meinung zu Python als erste Programmiersprache im Vergleich zu anderen Sprachen? Wie können sie [ihre Meinung] begründen…

Lehrer 10 Python hat eine besonders einfache Syntax. Python erlaubt einen Zugang sozusagen als großen Taschenrechner. [Es] ist daher für Schüler im Alter von 15, die ja keine besonders starke abstrakte Vorstellung haben, besonders geeignet.

Yazicioglu Gut. Wie könnte man die Verwendung von Python in Ausbildungsbereich fördern?

Lehrer 10 Dabei wäre es wichtig, dass man einerseits Seminare abhält und den Lehrer die Möglichkeit zu geben, Python kennen zu lernen. Und vor allem, dass auch diejenigen, die bereits Erfahrung mit Python gemacht haben, ihre Erfahrung weitergeben können. Also im wesentlichen geht es um Lehrerfortbildung und vielleicht auch pädagogische Konferenzen.

Yazicioglu Denken sie, dass es gut ist [Schülern] ein kleine Beispiel [zu geben], damit man die [Begeisterung] von Schülern [wecken kann]?

Lehrer 10 Ja durchaus. Ja, die Schüler sind ja doch recht positiv gestimmt. Sie nehmen das was ihnen vorgesetzt wird… Manche haben gar nicht die Kenntnisse… Also das sie unterschiedliche Sprachen kennen würden die Schüler… Aber Schüler sind sehr beeindruckt wenn man etwa beginnt ihnen vorzurechnen: 2^1 , 2^1 , 2^1 , 2^1 , 2^1 und Python rechnet immer noch brav. Das sind Beispiele, wo man in sehr kurzer Zeit einen sehr starken Eindruck gewinnen kann.

Yazicioglu Okay. Wie kann man Schüler für das Programmieren begeistern. Welche Rolle spielt dabei die Programmiersprache Python?

Lehrer 10 Wir haben die Erfahrung gemacht dass, nachdem man so ein paar grundlegende Dinge einmal besprochen haben, ein Programmpaket mit 3D-Animation besonders gut angekommen ist. Also mit einfachen Befehlen irgendwelche Kugeln, Quader, Was auch immer, Zu zeichnen… Herum zu bewegen… Das hat viel Spaß gemacht.

Yazicioglu Sie meinen Visualisierung begeistert die Schüler.

Lehrer 10 Ja. Viel spannender als eine `for`-Schleife.

Yazicioglu Da bin ich auch [ihrer] Meinung. Mit Python diese Visualisierung zu realisieren ist viel einfacher als in anderen Sprachen. Diesen Eindruck habe ich gehabt.

Lehrer 10 Ja, kann man schon… Ein bisschen präziser formuliert: Es gibt Programmpaket mit denen besonders einfach solche Dinge visualisieren kann. Und die sollte man im Unterricht einsetzen.

Yazicioglu Okay. Wie zum Beispiel TkInter…

Lehrer 10 Nein das nicht. Mir fällt jetzt der Name nicht ein, wie dieses Ding heißt. Aber wir können es dann nich herausfinden.

Yazicioglu Welche Lernkonzepte funktionieren im Programmierunterricht besonders gut? Ist die Verwendung von Analogien zum Beispiel aus Mathematik ein gutes Mittel für den Unterricht und warum?

Lehrer 10 Der Programmierunterricht war vor vielen vielen Jahren in der Stundentafel direkt unter der Mathematik angesiedelt. Und diese örtliche Nähe bei den Gegenständen hat schon ausgedrückt, dass man damals den Programmierunterricht als eine Art der Mathematik mit andere Mittel verstanden haben wollte. Der Programmierunterricht ist dann in dieser Stundentafel — die hat eine gewisse Struktur — hinuntergerutscht zu den technischen Gegenständen… Um eben auszudrücken, es ist eigentlich eine technischer Gegenstand und keine Mathematik mit anderen Mittel. Ich gebe zu ich war Schuld, dass es nach unten gerutscht ist. Das war mein Vorschlag. Okay, anderes Thema… Das gehört nicht zum Interview…

Die Problematik sehe ich darin, dass die Mathematik nur mit Zahlen rechnet. Es aber genauso wichtig ist das Schüler mit Buchstaben, mit Strings und mit Symbolen arbeiten können. Also würde man das zu sehr auf der Mathematik aufhängen, wäre der Wechsel zu Charakters, Strings usw. viel zu schwierig.

Yazicioglu Wie ist ihre Meinung? Entstehen die meisten Probleme von Schülern durch Sprachkonstrukte oder durch mangelndes Verständnis des Konzepts? Dabei kann ein Grund das Abstraktionsniveau auch sein… Ein Beispiel…

Lehrer 10 Ja, also ich glaube die Dinge sind gleich-verteilt. Wenn wir natürlich eine Sprache haben, bei der die Programmstrukturen einfacher sind, dann hat man mehr Zeit mit den Schülern an Verständnisproblemen zu arbeiten. Prinzipiell wäre im Idealfall der Programmierunterricht zumindest am Anfang ein Unterricht im Erstellen von Algorithmen und noch extremer formuliert ohne Verwendung eines Computers. Es ist ein Blödsinn. Aber das wäre einmal der abstrakte Zugang.

Wenn jetzt die Probleme bei der Syntax, Semantik auftreten durch die Sprache von vorne rein geringer sein, hat man mehr Zeit um die Probleme beim abstrakten Denken, beim Formulieren vom Algorithmen zu lösen.

Yazicioglu Beide Punkte [beeinflussen] sich gegenseitig.

Lehrer 10 Die wirken zusammen und… Formulieren wir es umgekehrt herum. Wenn ich eine Sprache habe, bei der ich meine volle Aufmerksamkeit darauf verwenden muss, damit die Schüler die Strichpunkte richtig setzen, dann wäre ich ihnen nicht erklären können… Oder habe ich wenig Zeit zu erklären, wie ein Algorithmus ausschaut. Das wär eigentlich falsch. Die Sprache ist nur ein Mittel um den Algorithmus auszudrücken. Aber sollte kein Selbstzweck sein.

Yazicioglu Okay. Wie kann man diese Probleme vermeiden. Was denken sie?

Lehrer 10 In dem man die richtige Programmiersprache wählt. Und das ist natürlich Python.

Yazicioglu Danke für [das] Interview.

Lehrer 10 Gerne.

Online-Umfrage

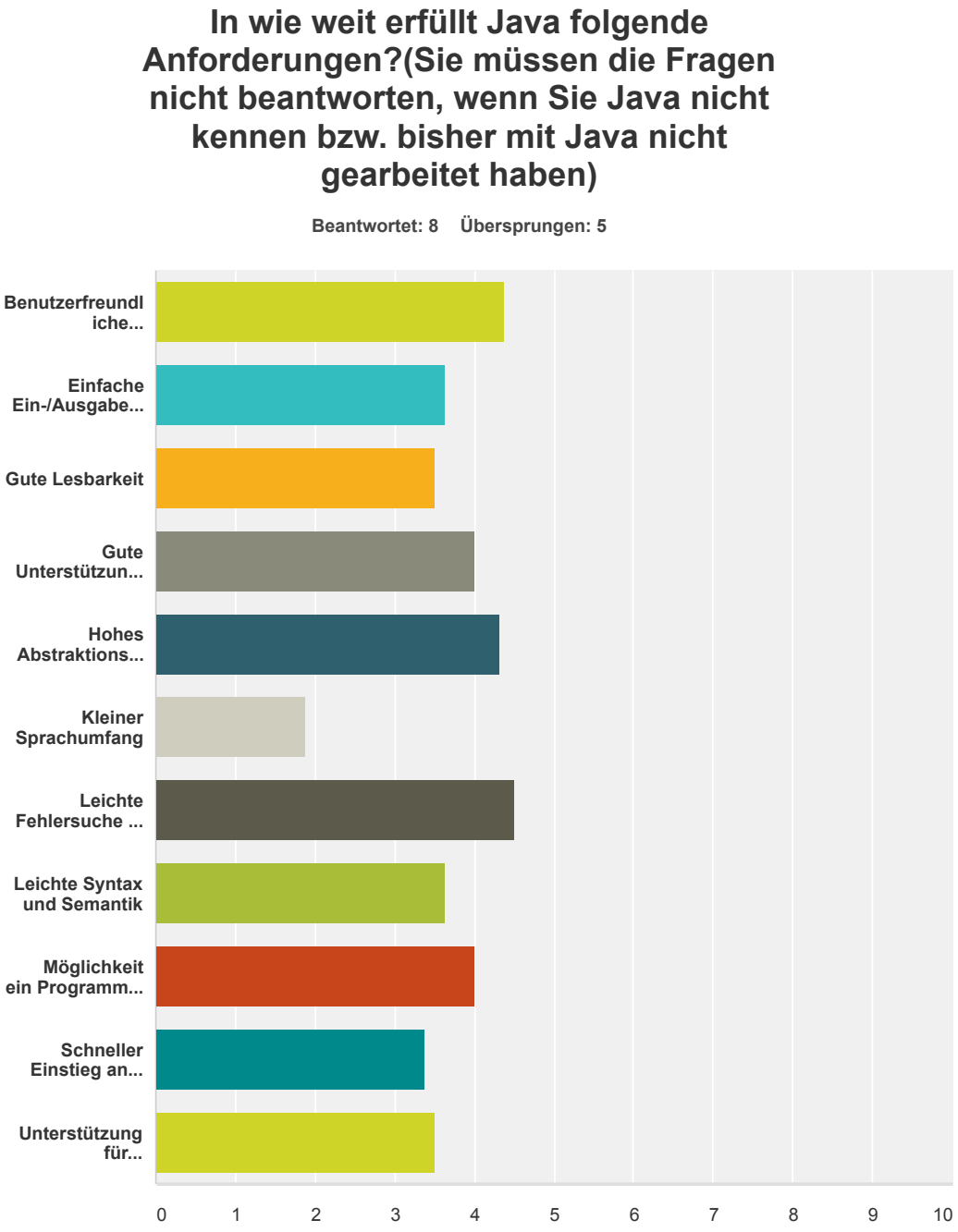
Persönliche Daten

Geben Sie bitte Ihre Daten an:

Beantwortet: 13 Übersprungen: 0

Antwortoptionen		Beantwortungen	
In welcher Schule unterrichten Sie?	**Beantwortungen**	**100,00%**	13
Seit wann sind Sie als Lehrer tätig?	**Beantwortungen**	**100,00%**	13
Welche Programmiersprachen unterrichten Sie?	**Beantwortungen**	**100,00%**	13
Welche Programmiersprache kennen Sie?	**Beantwortungen**	**61,54%**	8
Welche Schülergruppe unterrichten Sie?/Wie alt sind Ihre Schüler ungefähr?	**Beantwortungen**	**100,00%**	13

Anforderungen Java

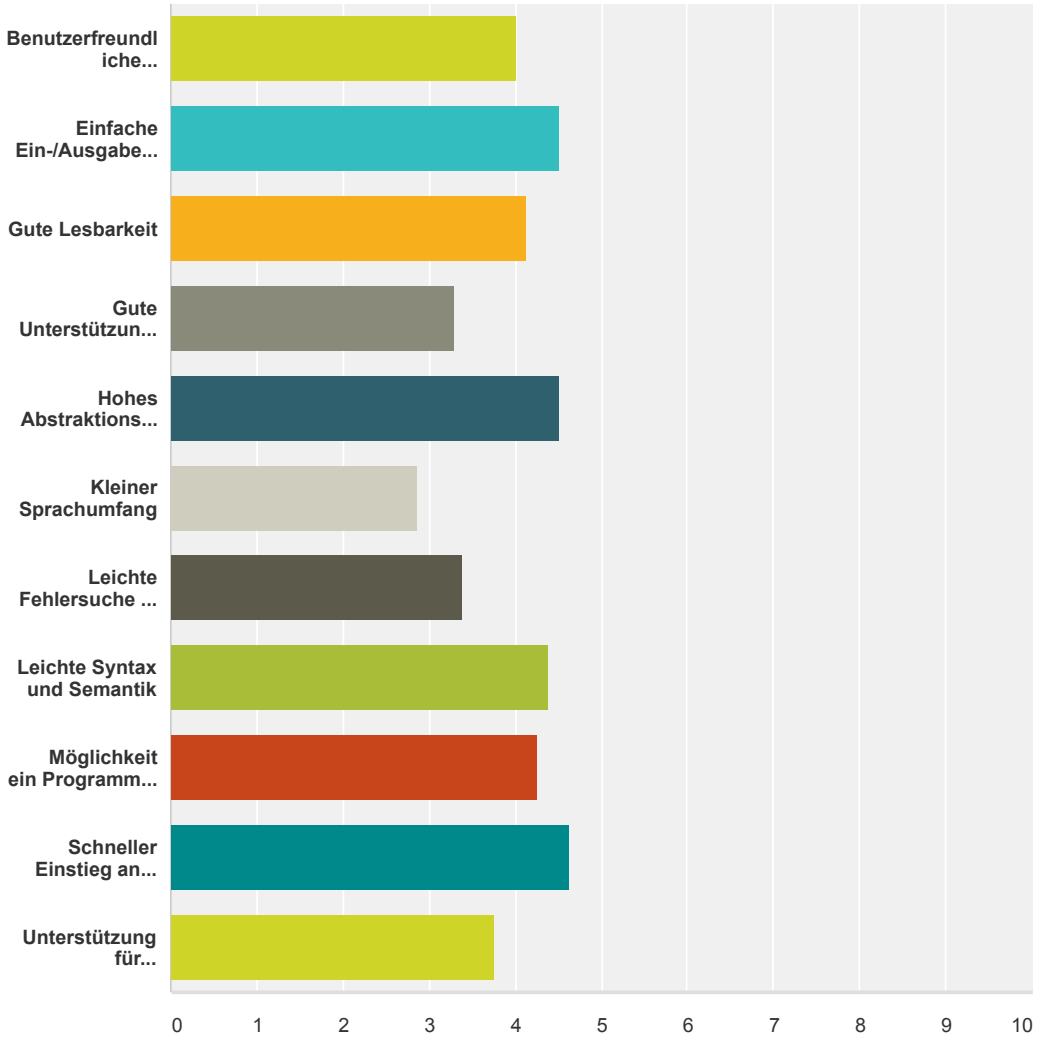


	Voll und ganz	ausreichend	unzureichend	überhaupt nicht	nicht relevant	Gesamt	Gewichteter Mittelwert
Benutzerfreundliche Entwicklungsumgebung	**50,00%** 4	**37,50%** 3	**12,50%** 1	**0,00%** 0	**0,00%** 0	8	4,38
Einfache Ein-/Ausgabe-System	**12,50%** 1	**50,00%** 4	**25,00%** 2	**12,50%** 1	**0,00%** 0	8	3,63
Gute Lesbarkeit	**12,50%** 1	**50,00%** 4	**12,50%** 1	**25,00%** 2	**0,00%** 0	8	3,50
Gute Unterstützung beim GUI-Design	**42,86%** 3	**42,86%** 3	**0,00%** 0	**0,00%** 0	**14,29%** 1	7	4,00
Hohes Abstraktionsniveau	**33,33%** 1	**66,67%** 2	**0,00%** 0	**0,00%** 0	**0,00%** 0	3	4,33
Kleiner Sprachumfang	**0,00%** 0	**0,00%** 0	**12,50%** 1	**62,50%** 5	**25,00%** 2	8	1,88
Leichte Fehlersuche und Debugging	**62,50%** 5	**25,00%** 2	**12,50%** 1	**0,00%** 0	**0,00%** 0	8	4,50
Leichte Syntax und Semantik	**0,00%** 0	**62,50%** 5	**37,50%** 3	**0,00%** 0	**0,00%** 0	8	3,63
Möglichkeit ein Programm „Schritt für Schritt“ zu entwickeln	**37,50%** 3	**37,50%** 3	**12,50%** 1	**12,50%** 1	**0,00%** 0	8	4,00
Schneller Einstieg an Programmieren	**12,50%** 1	**37,50%** 3	**25,00%** 2	**25,00%** 2	**0,00%** 0	8	3,38
Unterstützung für Multiparadigmen (OOP, Funktionale Programmierung...)	**37,50%** 3	**12,50%** 1	**25,00%** 2	**12,50%** 1	**12,50%** 1	8	3,50

Anforderungen Python

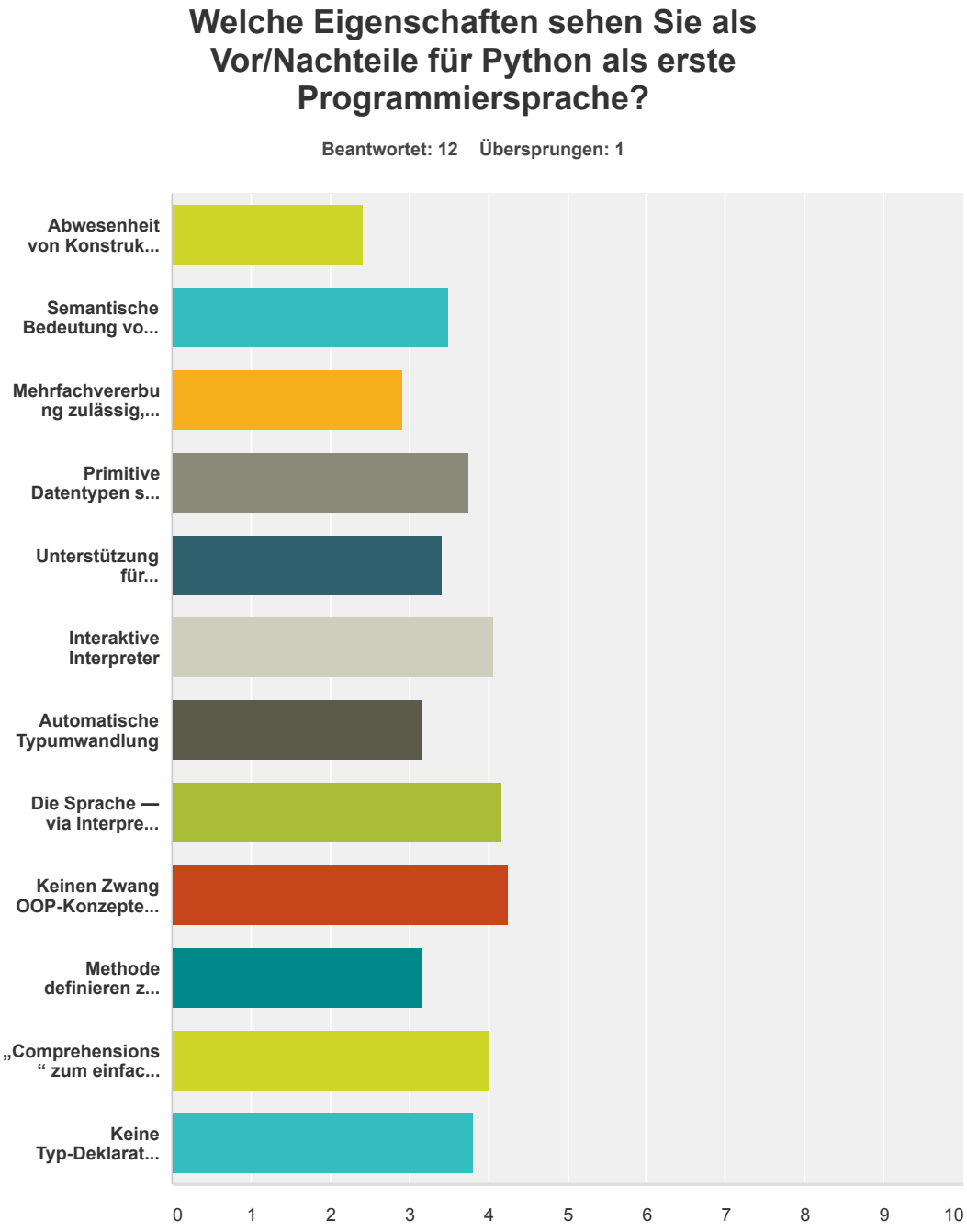
In wie weit erfüllt Python folgende Anforderungen?(Sie müssen die Fragen nicht beantworten, wenn Sie Python nicht kennen bzw. bisher mit Python nicht gearbeitet haben)

Beantwortet: 8 Übersprungen: 5



	Voll und ganz	ausreichend	unzureichend	überhaupt nicht	nicht relevant	Gesamt	Gewichteter Mittelwert
Benutzerfreundliche Entwicklungsumgebung	**50,00%** 4	**12,50%** 1	**25,00%** 2	**12,50%** 1	**0,00%** 0	8	4,00
Einfache Ein-/Ausgabe-System	**50,00%** 4	**50,00%** 4	**0,00%** 0	**0,00%** 0	**0,00%** 0	8	4,50
Gute Lesbarkeit	**50,00%** 4	**25,00%** 2	**12,50%** 1	**12,50%** 1	**0,00%** 0	8	4,13
Gute Unterstützung beim GUI-Design	**28,57%** 2	**14,29%** 1	**28,57%** 2	**14,29%** 1	**14,29%** 1	7	3,29
Hohes Abstraktionsniveau	**50,00%** 1	**50,00%** 1	**0,00%** 0	**0,00%** 0	**0,00%** 0	2	4,50
Kleiner Sprachumfang	**0,00%** 0	**57,14%** 4	**0,00%** 0	**14,29%** 1	**28,57%** 2	7	2,86
Leichte Fehlersuche und Debugging	**0,00%** 0	**50,00%** 4	**37,50%** 3	**12,50%** 1	**0,00%** 0	8	3,38
Leichte Syntax und Semantik	**50,00%** 4	**37,50%** 3	**12,50%** 1	**0,00%** 0	**0,00%** 0	8	4,38
Möglichkeit ein Programm „Schritt für Schritt“ zu entwickeln	**62,50%** 5	**25,00%** 2	**0,00%** 0	**0,00%** 0	**12,50%** 1	8	4,25
Schneller Einstieg an Programmieren	**75,00%** 6	**12,50%** 1	**12,50%** 1	**0,00%** 0	**0,00%** 0	8	4,63
Unterstützung für Multiparadigmen (OOP, Funktionale Programmierung...)	**50,00%** 4	**12,50%** 1	**12,50%** 1	**12,50%** 1	**12,50%** 1	8	3,75

Eigenschaften Python



	Vorteil	weniger Vorteil	weniger Nachteil	Nachteil	egal	Gesamt	Gewichteter Mittelwert
Abwesenheit von Konstrukten wie „do … while" oder „switch"	**8,33%** 1	**25,00%** 3	**8,33%** 1	**16,67%** 2	**41,67%** 5	12	2,42
Semantische Bedeutung von Einrückung	**58,33%** 7	**0,00%** 0	**0,00%** 0	**16,67%** 2	**25,00%** 3	12	3,50
Mehrfachvererbung zulässig, „Interface" nicht notwendig	**25,00%** 3	**16,67%** 2	**8,33%** 1	**25,00%** 3	**25,00%** 3	12	2,92
Primitive Datentypen sind Objekt	**50,00%** 6	**16,67%** 2	**8,33%** 1	**8,33%** 1	**16,67%** 2	12	3,75
Unterstützung für Multiparadigmen (OOP, Imperative, Funktional)	**33,33%** 4	**25,00%** 3	**8,33%** 1	**16,67%** 2	**16,67%** 2	12	3,42
Interaktive Interpreter	**75,00%** 9	**0,00%** 0	**0,00%** 0	**8,33%** 1	**16,67%** 2	12	4,08
Automatische Typumwandlung	**25,00%** 3	**25,00%** 3	**0,00%** 0	**41,67%** 5	**8,33%** 1	12	3,17
Die Sprache — via Interpreter — als „Taschenrechner" verwenden zu können	**58,33%** 7	**25,00%** 3	**0,00%** 0	**8,33%** 1	**8,33%** 1	12	4,17
Keinen Zwang OOP-Konzepte verwenden zu müssen	**66,67%** 8	**16,67%** 2	**0,00%** 0	**8,33%** 1	**8,33%** 1	12	4,25
Methode definieren zu können ohne den Return-Typ angeben zu müssen	**16,67%** 2	**16,67%** 2	**41,67%** 5	**16,67%** 2	**8,33%** 1	12	3,17
„Comprehensions" zum einfachen Erzeugen von Listen, Sets und Dictionaries	**50,00%** 6	**25,00%** 3	**8,33%** 1	**8,33%** 1	**8,33%** 1	12	4,00
Keine Typ-Deklaration in Python 2	**45,45%** 5	**9,09%** 1	**27,27%** 3	**18,18%** 2	**0,00%** 0	11	3,82

Kennen Sie noch weitere Beispiele von Spracheigenschaften, die Sie als Vorteil oder Nachteil von Python anführen würden?

Beantwortet: 4 Übersprungen: 9

Beantwortungen (4) Textanalyse Meine Kategorien

Kategorisieren als... | Nach Kategorie filtern | Beantwortungen durchsuchen

Anzeigen von **4** Beantwortungen

lambda-Funktionen, Generatoren, leichte Integration von Python-Scripts mit Unix-Kommandos, zB "myModul.py $(date +%s) | grep myString" sehr schlau durchdachte Klassenbibliotheken, ganz besonders den Modul "argparse",

02.08.2015 06:15 Beantwortungen von Befragten anzeigen

Vorteile: * funktionen als objekt an eine funktion übergeben. * so gut wie keine automatische typenkonvertierung (das wird hier oben falsch angenommen). stattdessen gibt es special methods (die mit __XYZ__()) um explizit die jeweils notwendige konvertierung kontrollieren zu können. * operator overloading * besseres unit testing * extrem gute bibliotheken für datenverarbeitung und visualisierung (pandas, matplotlib (bookeh, etc.), etc.) bzw. mathematische anwendugnen (sympy, numpy/scipy, sagemath, networkx, etc.) * metaprogramming, aber nur wenn es notwendig ist * gegenüber java viel effizienteres memory management * und auch sehr gute libraries zur beschleunigung des codes (numba, cython, theano, numexpr, etc.)

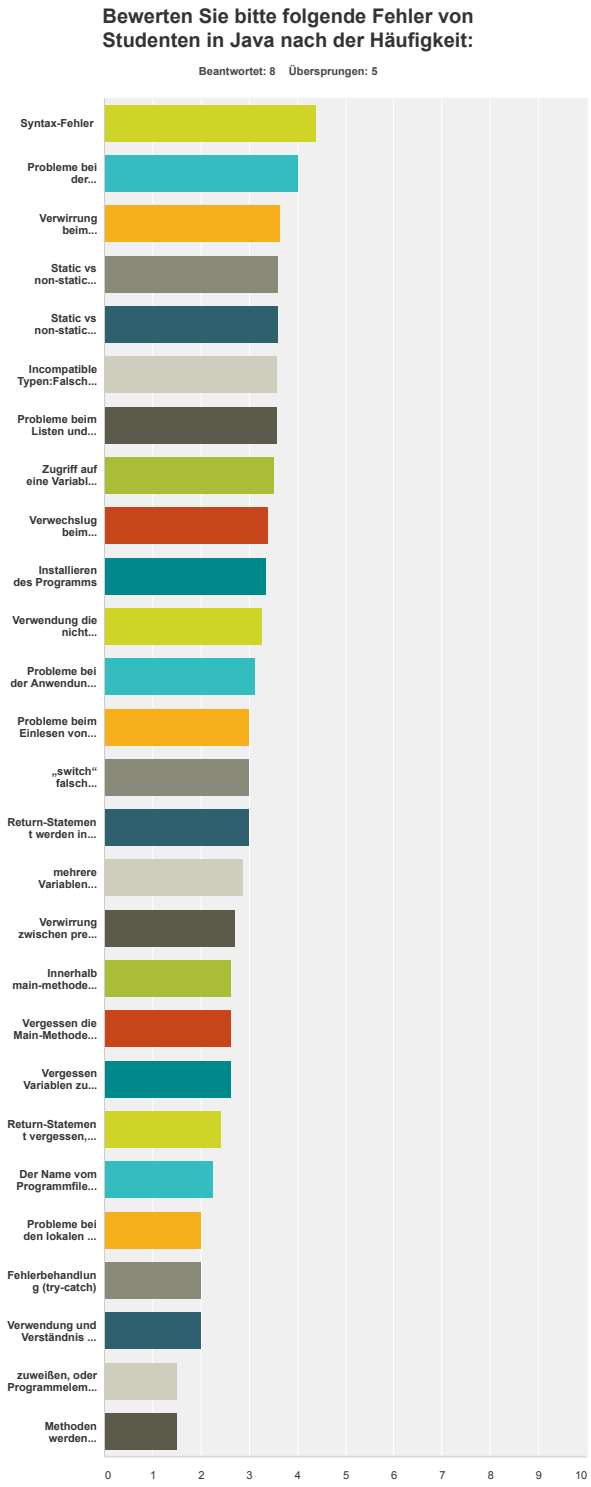
18.07.2015 14:43 Beantwortungen von Befragten anzeigen

Objekte sind überall und immer im Scope (hingegen C, C++). Man muss weniger über Speicheradressen wissen. List, Dictionaries sind schon in der Sprache beinhaltet (nicht via Standardebibliotheken zur Verfügung gestellt, wie bei Java) Standardebibliotheken wie os, urllib abstrahieren eine Menge von Komplexität, die einen Anfänger nicht berührt.

07.07.2015 11:35 Beantwortungen von Befragten anzeigen

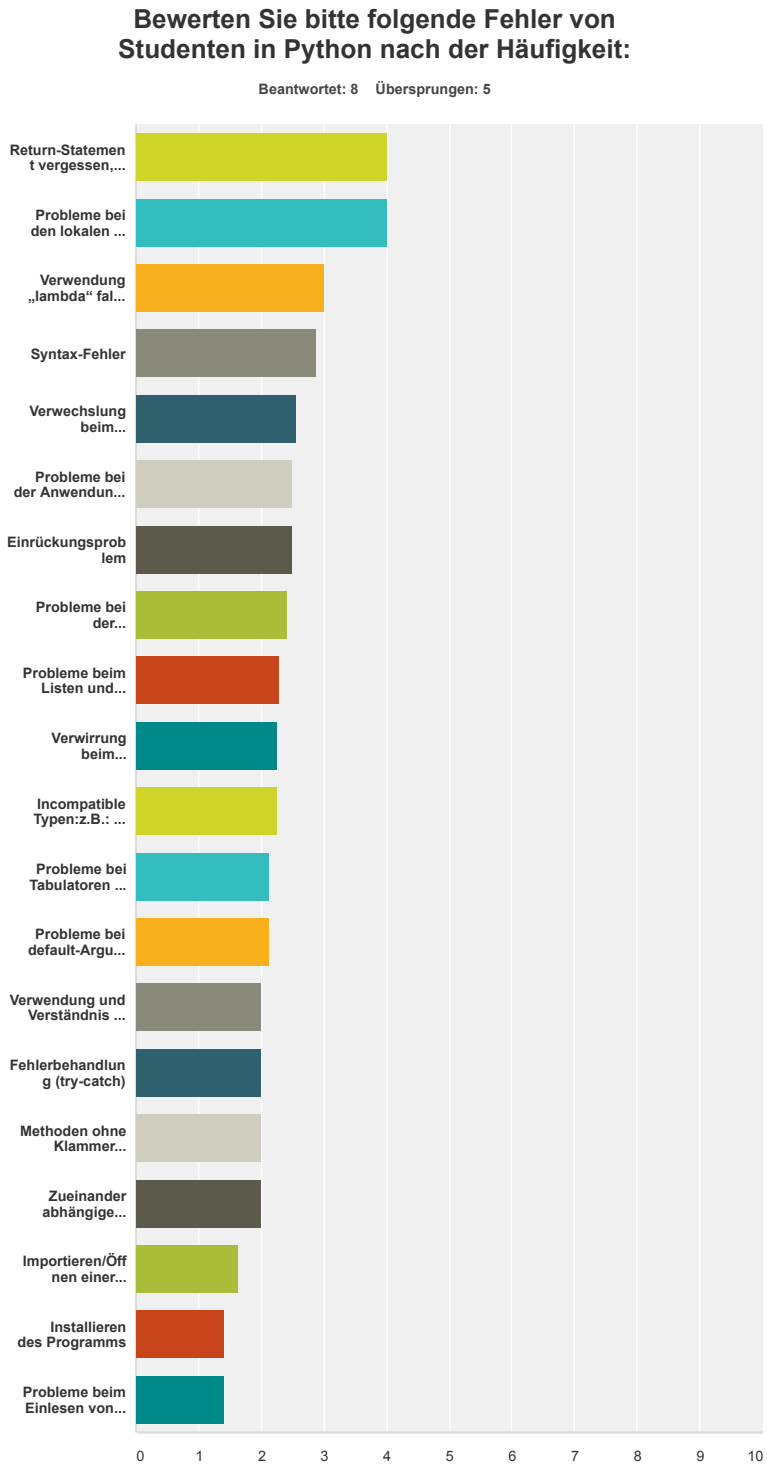
s = "Hallo Welt" print(s[-1]) # t print(s[::-1]) # tleW ollaH Vergleich von Objekte logischer Vergleich mit ==

Fehler Java


Bewerten Sie bitte folgende Fehler von Studenten in Java nach der Häufigkeit:
Beantwortet: 8 Übersprungen: 5
Syntax-Fehler
Probleme bei der...
Verwirrung beim...
Static vs non-static...
Static vs non-static...
Incompatible Typen:Falsch...
Probleme beim Listen und...
Zugriff auf eine Variabl...
Verwechslug beim...
Installieren des Programms
Verwendung die nicht...
Probleme bei der Anwendun...
Probleme beim Einlesen von...
„switch" falsch...
Return-Statemen t werden in...
mehrere Variablen...
Verwirrung zwischen pre...
Innerhalb main-methode...
Vergessen die Main-Methode...
Vergessen Variablen zu...
Return-Statemen t vergessen,...
Der Name vom Programmfile...
Probleme bei den lokalen ...
Fehlerbehandlun g (try-catch)
Verwendung und Verständnis ...
zuweißen, oder Programmelem...
Methoden werden...
0 1 2 3 4 5 6 7 8 9 10

	sehr häufig	häufig	manchmal	selten	nie	Gesamt	Gewichteter Mittelwert
Syntax-Fehler	**50,00%** 4	**37,50%** 3	**12,50%** 1	**0,00%** 0	**0,00%** 0	8	4,38
Probleme bei der Unterscheidung zwischen „==“ und „equals”	**37,50%** 3	**37,50%** 3	**12,50%** 1	**12,50%** 1	**0,00%** 0	8	4,00
Verwirrung beim Anwendungsgebiet der Operatoren „=“ und „==“	**12,50%** 1	**37,50%** 3	**50,00%** 4	**0,00%** 0	**0,00%** 0	8	3,63
Static vs non-static Methoden	**20,00%** 1	**40,00%** 2	**20,00%** 1	**20,00%** 1	**0,00%** 0	5	3,60
Static vs non-static Variablen	**20,00%** 1	**40,00%** 2	**20,00%** 1	**20,00%** 1	**0,00%** 0	5	3,60
Incompatible Typen:Falsch: String temp = a[2] + a[3];Richtig: String temp = (String)a[2] + a[3];Meldung: required:String, found int. (Casting vergessen)	**14,29%** 1	**42,86%** 3	**28,57%** 2	**14,29%** 1	**0,00%** 0	7	3,57
Probleme beim Listen und Arrays	**0,00%** 0	**57,14%** 4	**42,86%** 3	**0,00%** 0	**0,00%** 0	7	3,57
Zugriff auf eine Variable außerhalb ihres Gültigkeitsbereichs	**25,00%** 2	**37,50%** 3	**0,00%** 0	**37,50%** 3	**0,00%** 0	8	3,50
Verwechslug beim Referenzieren und Kopieren eines Objekts	**25,00%** 2	**25,00%** 2	**12,50%** 1	**37,50%** 3	**0,00%** 0	8	3,38
Installieren des Programms	**16,67%** 1	**33,33%** 2	**33,33%** 2	**0,00%** 0	**16,67%** 1	6	3,33
Verwendung die nicht initialisierte Variablen z.B: String str [] = null; int a = str.length	**0,00%** 0	**50,00%** 4	**25,00%** 2	**25,00%** 2	**0,00%** 0	8	3,25
Probleme bei der Anwendung der Schleifen „while“ und „for“	**0,00%** 0	**37,50%** 3	**37,50%** 3	**25,00%** 2	**0,00%** 0	8	3,13
Probleme beim Einlesen von Werten von der Tastatur	**16,67%** 1	**16,67%** 1	**16,67%** 1	**50,00%** 3	**0,00%** 0	6	3,00
„switch“ falsch verwendet (break/default vergessen)	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	**0,00%** 0	1	3,00
Return-Statement werden in void-methode verwendet, oder Returnytpe einer Methode zu schreiben vergessen	**0,00%** 0	**50,00%** 1	**0,00%** 0	**50,00%** 1	**0,00%** 0	2	3,00
mehrere Variablen gleichen Namens	**0,00%** 0	**25,00%** 2	**37,50%** 3	**37,50%** 3	**0,00%** 0	8	2,88
Verwirrung zwischen prefix und postfix operatorenz.B: x = 45 → y = ++x , y = x++	**0,00%** 0	**14,29%** 1	**57,14%** 4	**14,29%** 1	**14,29%** 1	7	2,71
Innerhalb main-methode nicht statische Methode (Instanz-Methode) aufgerufen	**0,00%** 0	**0,00%** 0	**62,50%** 5	**37,50%** 3	**0,00%** 0	8	2,63
Vergessen die Main-Methode zu deklarieren, oder die Funktion falsch deklariert (Falsche Reihenfolge von Keywords: static void main)	**0,00%** 0	**37,50%** 3	**12,50%** 1	**25,00%** 2	**25,00%** 2	8	2,63
Vergessen Variablen zu definieren	**0,00%** 0	**25,00%** 2	**25,00%** 2	**37,50%** 3	**12,50%** 1	8	2,63
Return-Statement vergessen, bei Methoden die einen Return-Type erwarten	**0,00%** 0	**0,00%** 0	**57,14%** 4	**28,57%** 2	**14,29%** 1	7	2,43
Der Name vom Programmfile(„filename“.java) und der Klasse sind unterschiedlich	**0,00%** 0	**25,00%** 2	**0,00%** 0	**50,00%** 4	**25,00%** 2	8	2,25
Probleme bei den lokalen und globalen Variablen	**0,00%** 0	**0,00%** 0	**50,00%** 1	**0,00%** 0	**50,00%** 1	2	2,00
Fehlerbehandlung (try-catch)	**0,00%** 0	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	1	2,00
Verwendung und Verständnis von Bibliotheksfunktionalität	**0,00%** 0	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	1	2,00
zuweißen, oder Programmelementen wie (Schleifen, Anweisungen, Verzweigungen...) nicht direkt in der Klasse schreiben, anstatt in einer Methode	**0,00%** 0	**0,00%** 0	**0,00%** 0	**50,00%** 1	**50,00%** 1	2	1,50
Methoden werden inklusive Parametertype aufgerufen:z.B.: method(int x, int y)	**0,00%** 0	**0,00%** 0	**0,00%** 0	**50,00%** 1	**50,00%** 1	2	1,50

Fehler Python



	sehr häufig	häufig	manchmal	selten	nie	Gesamt	Gewichteter Mittelwert
Return-Statement vergessen, bei Methoden die einen Return-Type erwarten	**0,00%** 0	**100,00%** 1	**0,00%** 0	**0,00%** 0	**0,00%** 0	1	4,00
Probleme bei den lokalen und globalen Variablen	**0,00%** 0	**100,00%** 1	**0,00%** 0	**0,00%** 0	**0,00%** 0	1	4,00
Verwendung „lambda“ falsch (Schwierigkeiten haben beim Verwenden "lambda")	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	**0,00%** 0	1	3,00
Syntax-Fehler	**12,50%** 1	**12,50%** 1	**25,00%** 2	**50,00%** 4	**0,00%** 0	8	2,88
Verwechslung beim Referenzieren und Kopieren eines Objekts	**0,00%** 0	**14,29%** 1	**42,86%** 3	**28,57%** 2	**14,29%** 1	7	2,57
Probleme bei der Anwendung der Schleifen „while“ und „for“	**0,00%** 0	**25,00%** 2	**12,50%** 1	**50,00%** 4	**12,50%** 1	8	2,50
Einrückungsproblem	**0,00%** 0	**25,00%** 2	**0,00%** 0	**75,00%** 6	**0,00%** 0	8	2,50
Probleme bei der Unterscheidung zwischen „==“ und „is”	**14,29%** 1	**0,00%** 0	**28,57%** 2	**28,57%** 2	**28,57%** 2	7	2,43
Probleme beim Listen und Arrays	**0,00%** 0	**14,29%** 1	**0,00%** 0	**85,71%** 6	**0,00%** 0	7	2,29
Verwirrung beim Anwendungsgebiet der Operatoren „=“ und „==“	**0,00%** 0	**0,00%** 0	**37,50%** 3	**50,00%** 4	**12,50%** 1	8	2,25
Incompatible Typen:z.B.: >>> a = „1“ + 2incompatible type for declaration can't convert xx to yy	**0,00%** 0	**12,50%** 1	**25,00%** 2	**37,50%** 3	**25,00%** 2	8	2,25
Probleme bei Tabulatoren und Leerzeichen	**0,00%** 0	**12,50%** 1	**25,00%** 2	**25,00%** 2	**37,50%** 3	8	2,13
Probleme bei default-Argument in einer Methode	**0,00%** 0	**0,00%** 0	**25,00%** 2	**62,50%** 5	**12,50%** 1	8	2,13
Verwendung und Verständnis von Bibliotheksfunktionalität	**0,00%** 0	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	1	2,00
Fehlerbehandlung (try-catch)	**0,00%** 0	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	1	2,00
Methoden ohne Klammer aufrufen	**0,00%** 0	**25,00%** 2	**0,00%** 0	**25,00%** 2	**50,00%** 4	8	2,00
Zueinander abhängige Module verwenden	**0,00%** 0	**0,00%** 0	**0,00%** 0	**100,00%** 1	**0,00%** 0	1	2,00
Importieren/Öffnen einer Module	**0,00%** 0	**0,00%** 0	**12,50%** 1	**37,50%** 3	**50,00%** 4	8	1,63
Installieren des Programms	**0,00%** 0	**0,00%** 0	**14,29%** 1	**14,29%** 1	**71,43%** 5	7	1,43
Probleme beim Einlesen von Werten von der Tastatur	**0,00%** 0	**0,00%** 0	**0,00%** 0	**42,86%** 3	**57,14%** 4	7	1,43