Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Ausarbeitung

Einführung und Neuerungen mit Python 3

vorgelegt von

Mike Nguyen

Fakultät für Mathematik, Informatik und Naturwissenschaften
Fachbereich Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Sowtwareentwicklung in der Wissenschaft

Studiengang:	Software-System-Entwicklung
Matrikelnummer:	7301557
Betreuer:	Jannek Squar

Hamburg, 25.08.2020

Contents

1 Einführung

Heutzutage existieren viele verschiedene Programmiersprachen, welche natütlich ihre eigenen Porgrammierstile besitzen. Es ist wichtig die Details der Sprachen zu kennen, um dessen Potential voll ausschöpfen zu können. Wie z.B. Quellcode einsparen. Dazu kommt, dass man verschiedene Programmierarten kennen lernen sollte, um für ein vorhandendes Problem, die geeignete Sprache aussuchen zu können.

Eine von diesen Sprachen ist Python. Seit den letzten Jahren ist Python eines der bekanntesten und auch beliebtesten Sprachen weltweit geworden und hat mittlerweile Millionen Anhänger.

Aber wie kam es zu einem so großen Anstieg an Nutzern? Beziehungsweise welche Vorteile bringt die Sprache, die andere Sprachen nicht haben?
Genau darum soll es in diesem Bericht gehen. Im Folgenden wird Python näher erläutert, damit ihr euch ein besseres Bild von der Sprache machen könnt. Ihr werdet an verschiedenen Quellcode-Beispielen die Grundlagen kennen lernen. Außerdem werden die schwerwiegendsten Vor- und Nachteile an Vergleichen mit anderen Sprachen (zumeist Java) gezeigt. Dafür wird vorrausgesetzt, dass die Basis von Java berkannt ist. Und wie es in Zukunft mit Python aussieht, kann man an den gegebenen Statistiken sehen.

Weiterhin soll es speziell um Python 3, die derzeit neuste Vollversion von Python, gehen. Dadurch werden auch Version 2 und 3 gegenübergestellt, um dessen Unterschiede zu zeigen.

2 Prinzipien

Erstellt von Python's Entwickler Guido van Rossum soll "The Zen of Python" die Prinzipien von Python zusammenfassen. Aus diesen Prinzipien können auch die Vorteile der Sprache herausgelesen werden, die in Kapitel 3 – Merkmale näher erläutert werden. Hier ein kleiner Abschnitt:

1. "schön statt hässlich" bezieht sich auf die Lesbarkeit und Übersichtlichkeit der Sprache

3. "einfach statt kompliziert" soll hinauslaufen, dass die Sprache Verständlich ist

4. "komplex statt undurchschaubar" sagt aus, dass Komplexität existiert und eingesetzt werden soll, um den Quellcode übersichtlicher zu gestalten und Quellcode einzusparen.

8. "Spezialfälle sind niemals speziell genug, damit sie Regeln sprengen dürfen" dies ist ein wichtiger Schutzaspekt, der mit Python 3 behoben wurde.

10. "ein Fehler sollte nie verschwiegen werden, es sei denn er wurde absichtlich gemacht" bezieht sich auf Python's eingebaute Fehlermeldungen.

3 Geschichtlicher Hintergrund

Guido van Rossum, welcher zuvor an der Entwicklung der ABC-Sprache beteiligt war, begann 1990 eine neue Sprache zu entwickeln. Von der Komikergruppe Monthy Python inspiriert, bekam Python seinen heutigen Namen.

In der ersten Vollversion aus 1994 stand die funktionale Programmierung im Vordergrund. Eingeführt wurden hier Funktionen wie lambda, map und Filter. Version 2 wurde im Jahre 2000 veröffentlicht und beschäftigte sich nun mit der objekorientierten Programmierung. Hinzugefügt wurde ein Garbagecollector, welcher für die Speicherbereinigung zuständig ist, sowie der ASCII Zeichensatz.

Mit Version 3 gab es grundlegende Änderungen in der Sprache, sodass diese Version nicht mehr kompitabel mit älteren Versionen ist. Beispiele hierfür sind die Bibliotheken oder auch Funktionen wie print(). (siehe Kapitel 4) Die Version erschien 2008 und ist seit heute die neuste Vollversion.
Durch Python's weitergeführte Entwicklung, bekam es seit Version 3 eine großen Aufschwung an neuen Nutzern. Heute gilt Python als einer der beliebtesten Sprachen weltweit.

4 Merkmale

4.1 Sprachtyp

Der Sprachtyp gibt an, welche Art von Programmierung stattfindet, welche sich auf den Programmierstil und die darausfolgenden Eigenschaften auswirkt. Python beruht auf funktionaler sowie objektorientierter Programmierung. Funktionen wie Map- , Filter- und Lambda repräsentieren die funktionale Programmierung, in der das Lösen durch Funktionen im Mittelpunkt steht. Auch pure Funktioen oder immutability (nicht änderbare Werte) gehören dazu. Die objektorientierte Programmierung wird im folgenden Python-Programmausschnitt gezeigt:

```
1  class Testklasse
2      def __init__(self):
3          self.zahl = 0
4          sel.text = "Hallo"
5
6      def setZahl(self,zahl=int(input("neue Zahl:")))
7          self.zahl = zahl
8
9      def gibZahl(self)
10         print("Deine Zahl ist:",self.zahl)
11
12 if __name__ == "__main__":
13     objekt = Testklasse()
14     objekt.setZahl()
15     objekt.gibZahl()
16     print(objekt.text)
```

Listing 4.1: Testklasse

Zu sehen ist eine Klasse namens Testklasse (Zeile 1) mit einem Konstruktor (Zeile 2-4), welcher verschiedene Methoden besitzt (Zeile 6-10). Außerdem ist eine Main-Methode gegeben (Zeile 12-16), die bei der Erstellung eines Obejktes ausgeführt wird.
Das Merkmal hierbei ist, dass das Definieren von Klassen und der Erstellung von Objekten möglich ist und diese interagieren können. In dem Beipsiel interargiert das Objekt mit sich selbst durch den Aufruf der gegebenen Methoden, jedoch ist auch eine Beziehung zwischen mehreren Objekten möglich.

4.2 Syntax

Die Syntax einer Sprache gibt an, wie die Struktur eines Quellcodes aussieht. Hierbei wird ein übersichtlicher Quellcode angestrebt, um den Quellcode im Nachhinein besser verstehen zu können.

```
1 meineZahl = int(input("Suche dir eine Zahl!"))
2 if meineZahl > 1000:
3     print("ziemlich gross")
4 if meineZahl == 1000:
5     print("genau 1000!")
6 elsif meineZahl > 0:
7     print("wenigstens etwas")
8 else:
9     print("Die Zahl ist zu klein")
```

Listing 4.2: Syntax

Im obigen Beispiel können wir Python's Syntax feststellen. Im Beispiel wird der Benutzer aufgerufen eine beliebige Zahl einzugeben und je nach Eingabe, variiert die Ausgabe.Sofort fallen ein paar Unterschiede zu einem gleichfunktionalen Quellcode aus Java auf.

a) Im Gegensatz zu Java werden Befehle nicht durch Simmekolons getrennt, sondern mit Zeilenumbrüchen. Dadurch wird jeweils 1 Zeichen pro Befehl gespart und außerdem wird vermieden, dass mehrere Befehle in eine Zeile geschrieben werden, wodurch der Entwickler gezwungen ist, übersichtlicher zu programmieren.

b) Zur Strukturierung werden asnstatt geschweiften Klammern Einrückungen genutzt. Dies führt dazu, dass pro Blockeinheit 2 Zeichen gespart werden. Außerdem können so Verschachtelungen besser eingestuft werden.

Allgemein kann gesagt werden, dass so weniger Zeichen benötigt werden, desto weniger Fehler auftreten (z.B. Zeichen vergessen). Somit ist Python weniger fehleranfällig als Java. Außerdem ist die Sprache lesbarer, da weniger Zeichen benötigt werden, wodurch der benötigte Quellcode sich verringert.

4.3 Kontrollstrukturen

Kontrollstrukturen steuern den Programmablauf und helfen Quellcode einzusparen.
Wie in Java existieren For- und While-Schleifen, sowie if/else Blöcke. Jedoch findet man in der Benennung von else-if, welcher sich in Python elif schreibt, einen kleinen Unteschied. Außerdem sind switch- sowie go-to-Strukturen nicht implementiert, wodurch der Entwickler diese Funktionen umschreiben muss.
Im folgenden Beispiel wird eine For-schleife gezeigt(Für Fallunterscheidung siehe 4.2):

```
1 Liste = [0,2,3,1]
2 for element in Liste[0:]:
3     print(element)
```

Listing 4.3: Syntax

Ein Liste mit 4 Elementen wird erstellt. Diese Elemente werden dann ausgegeben, indem durch die Liste gegangen wird, beginnend mit Element 0 und dann jedes Element geprinted wird.

4.4 Übersetzer

Die Übersetzungsart spielt für Programmiersprachen eine wichtige Rolle. Denn sie geben vor auf welchen Plattformen die Sprachen fungieren können oder wie schnell der Quellcode einer Sprache läuft.
Sprachen werden meist kompiliert oder interpretiert. Während Compilersprachen schnell sind, dafür plattformabhängig, sind Interpretersprachen langsam und plattformunabhängig. Python benutzt hingegen beides und vereint somit Vor- und Nachteile.
Der Compiler wird genutzt, um den Quellcode nicht wie sonst in einen Maschinencode umzuwandeln, sondern nur in einen Zwischenschritt, den sogenannten Bytecode. Dieser kann dann zur Laufzeit durch den Interpreter Zeile für Zeile ausgeführt werden. Aus diesem Grund gehört Python zu den interpretierten Sprachen, da dies im Vordergrund liegt.
Im Wesentlichen ist die Variante plattformunabhängig und zur Laufzeit schneller als gewöhnliche Interpretersprachen, da der Bytecode schneller ausgeführt werden kann. Trotzdessen ist die Sprache an einem Interpreter gebunden, sodass z.B. ein Python 2 Interpreter nicht durch ein Python 3 Interpreter ersetzbar ist, da beide unterschiedlich arbeiten. Außerdem kann Python die Schnelligkeit nicht ausnutzen (siehe Kapitel 4.6)

4.5 Interaktiv

Eine Schnittstelle zwischen Prgramm und Nutzer bietet eine Möglichkeit sich mit der Sprache genauer auseinanderzusetzen.
Python bietet einen solchen interaktiven Modus an, in welchem der Interpreter direkt angesprochen werden kann. Python Shell ist ein Terminal in welchem der Nutzer seine eigenen Eingaben als Input eingibt. Dies ermöglicht das Ausprobieren von verschiedenen Interaktionen und das Testen von kleinen Programmabschnitten.

4.6 Geschwindigkeit

Die Geschwindigkeit einer Sprache definiert den Einsatzbereich und Nutzen, denn warum sollte man eine langsame Sprache nutzen, wenn Schnellere existieren?
Trotz Python's Übersetzungvariante gehört die Sprache zu den langsamsten Sprachen. Das folgende Diagramm zeigt die benötigte Zeit verschiedener Sprachen, die benötigt wird, um die Summe von 0 bis 1000 zu errechnen.

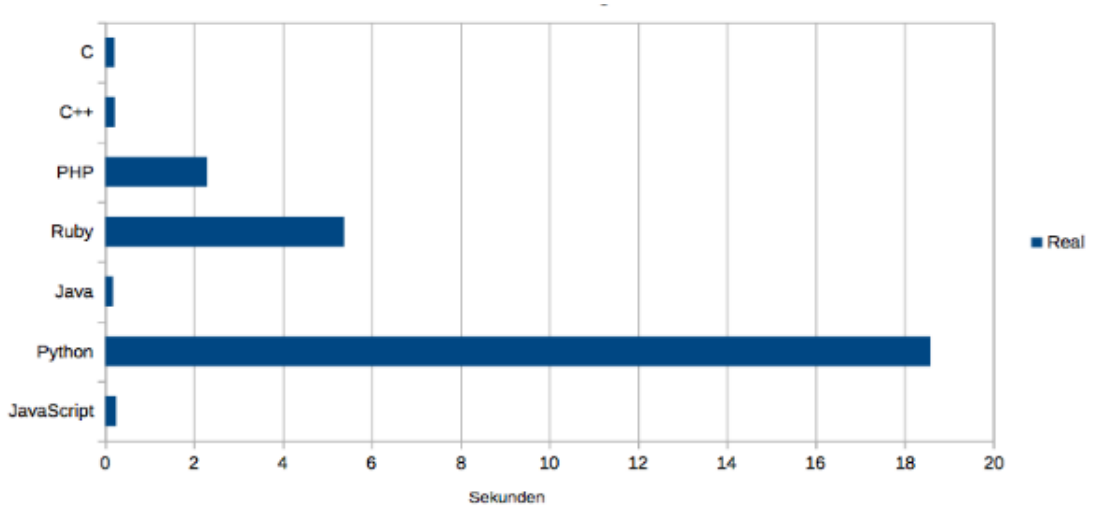

Figure 4.1: Performancevergleich

Zu sehen ist, dass die Sprachen C,C++, Java sowie Java-script mit 0,2s am wenigstens Zeit benötigen. Es folgen PHP und Ruby mit je 2,1s und 5s. Auf dem letzten Platz benötigt Python mehr als 18 Sekunden.
Ruby sowie PHP sind interpretierte Sprachen, wodurch diese langsamer als kompilierte Sprachen sind. Die schnelleren Sprachen benutzen einen Compiler. Sprachen wie Java benutzen jedoch auch die Python's Mischvariante, was nun die Frage stellt warum Python so langsam ist.
Python benutzt einen GIL (Global Interpreter Lock), sodass jeweils nur ein thread zur Zeit läuft. Bzw. Nur eine Aktion wird zur Zeit ausgefhührt, während andere Sprachen

mehrere Aktionen zeitgleich ausführen können. Der Grund hierfür ist die daraus entstehende Sicherheit, wie das Zählen von Referenzen, was bei zeitgleicher Auführung zu Problemen führen kann. Somit haben wir einen höheren Zeitaufwand, dafür einen sicheren Verlauf der im nächsten Teilkapitel erklärt wird.

4.7 Speicherverwaltung

Durch die Art der Speicherverwaltung kann ausgesagt werden wie effizient der Speicher genutzt wird. Dies bringt Folgen für das Programm, aber auch für den Entwickler.

Python benutzt eine dynamische Speicherverwaltung. Das heißt alle Variablen werden als Objekte angesehen, die verschiedene Werte annehmen können.

```
1 x = 42
2 y = "Hi"
3 z,a = True,False
4
5 x = "nicht mehr 42"
6
7 zahl = 7.4
8 print(int(zahl))
```

Listing 4.4: Speicherverwaltung

In den ersten 3 Zeilen des obigen Beispiels werden verschiedene Variablen implemtiert mit den gängigen Typen integer, String und boolean. Im Vergleich zu Java ist eine Variablendeklaration nicht vonnöten. Im nächsten Schritt wird der Variable x ein neuer Wert zugewiesen und zwar von einem Integer zu einem String. Dies ist anders als bei Java möglich. Außerdem referenziert nichts mehr die 42, sodass diese vom Garbagecollector entfernt werden kann. Man kann sich dies so vorstellen, als würde man eine Box hinstellen die den Integer 42 in sich trägt. Auf diese Box zeigen dann verschiedene Variablen und so lange eine Variable darauf zeigt bleibt sie bestehen. Zeigt x nun nicht mehr auf 42, sondern auf den String "nicht mehr auf 42", so wird die Box 42 gelöscht. Hier kommt das Multithreadproblem ins Spiel. Würden mehrere Aktionen gleichzeitig ausgeührt werden, kann es dazu führen, dass eine benötigte Box zu früh entfernt wird.

Außerdem existiert in der dynamischen Speicherverwaltung ein Sicherheitsproblem, da man nie genau wissen kann welchen Typ eine Eingabe hat. z.B. 42 oder "42". Dieses Problem wird behoben indem ein casting eingeführt wird (siehe Beispiel).

4.8 Library

Die Schnittstelle zwichen Programmierer und Programm befindet sich in der Library / Bibliothek. Diese Bibliothek enthält viele Module und Programmabschnitte, die dem Programmierer helfen soll, Quellcode einzusparen oder auf komplexen Quellcode schnell zugreifen zu können. Darum ist es wichtig sich mit den gegebenen Modulen auseinander zu setzen, um sich mit der Sprache vertraut zu machen.
Python's Bibliothek enthält viele Hilfsmodule (ein Modul ist ein Abschnitt der Bibliothek), die dem Entwickler unterstützen können. Vom Debugging bis zum User-Interface sind viele Hilfmodule gegeben.
Um die Vorteile zu nutzen müssen zuallererst die benötigten Dateien importiert werden. Denn wären alle Module schon installiert, würde dies zu Zeitverzögerungen führen. Das Importieren geschieht mit dem "import" Befehl. Will man Python's Mathebefehle nutzen so impotiert man das Modul "math": import math.

Module sind jedoch nicht nur auf die Standardbibliothek angewiesen, sondern man kann sich auch seine eigenen erstellen. Erstellt man z.B. eine Datei mit der Klasse "Testklasse" so kann eine zweite Datei auf die Klasse zugreifen, indem folgender Code benutzt wird:

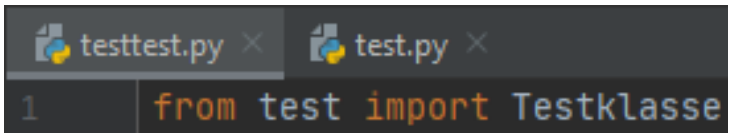

Figure 4.2: Import

generell gilt: from "Wo" import "Was"
Es gibt auch die Möglichkeit selbstentwickelte Module von Herausgebern runterzuladen und zu nutzen. Mit derzeit 41 Prozent Nutzerrate ist Django gegenwärtig das meiste genutzte Paket.

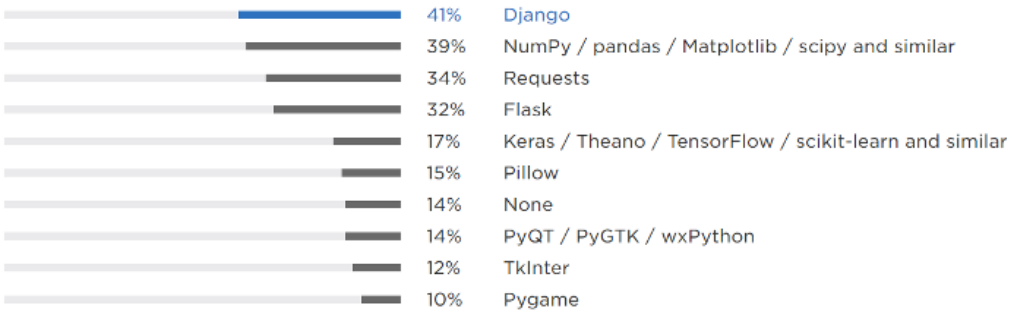

Figure 4.3: Import

4.9 Erweiterbarkeit

Natürlich haben Sprachen ihre eigenen Stärken und Schwächen. Mit dem Kombinieren von Sprachen können die eigenen Schwächen ausgeglichen werden, indem kritische Stellen durch andere Sprachen definiert werden können.
Python unterstützt das Teilprogrammieren in anderen Sprachen. Da Python eine langsame Sprache ist, können schnellere Sprachen bei zeitlich kritischen Stellen Quellcode ersetzen. Dafür müssen Pakete installiert und mit dem import-Befehl aufgerufen werden.

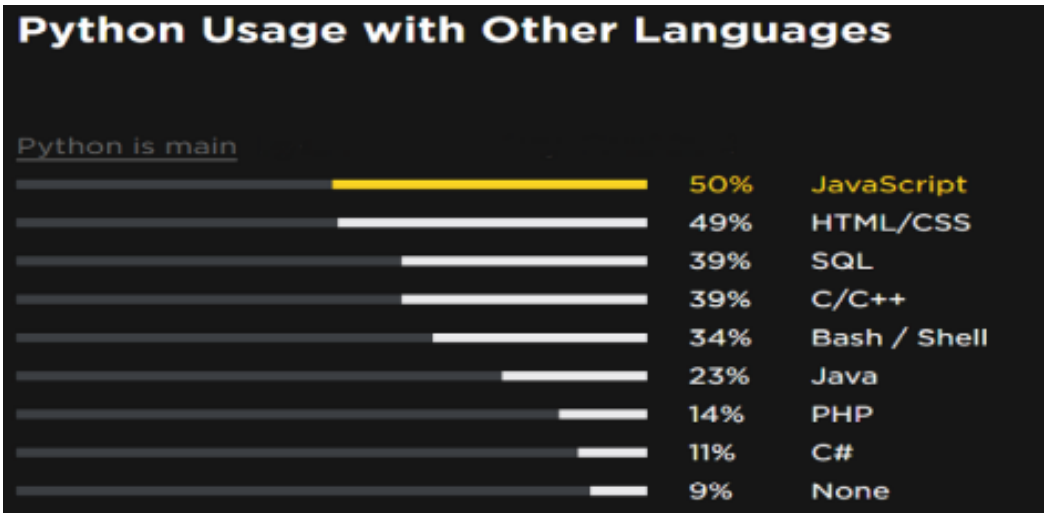

Figure 4.4: Kombination

Wie zu sehen wird Python oft mit JavaScript oder CSS kombiniert. Dabei können auch mehrere Sprachen im selben Quellcode genutzt werden. Außerdem wird Python mit 79 Prozent meist als Hauptsprache genutzt und nur 21 Prozent als Nebensprache.

4.10 Fehler

Natürlich kommt es immer wieder, dass das Programm Fehler entdeckt und mit ihnen umgehen muss. Dies soll der Entwickler natürlich mitbekommen und auch möglichst beheben. Falls in Python ein Fehler auftritt bekommt der Programmierer eine Fehlermeldung. Diese sieht folgend aus:

```
1 Traceback(most recent call last):
2     Dateipfad, Zeile, (Modul)
3         Felhlercode
4     Art des Fehlers
```

Listing 4.5: Fehlermeldung

In dieser Meldung kann festgestellt werden, wo die Zeile, die nicht richtig ausgeführt werden kann, sich befindet, das genaue Abbild des Codes, sowie die Art des Fehlers. Dabei kann zwischen verschiedenen Arten unterschieden werden: Syntax-,Logical- und Built-in-Fehlern, wie z.B. ZeroDivisionError, NameError oder TypeError.
Will man solche Fehlermeldung vermeiden, kann dies mit einem assert Befehl oder mit einem Try/Except-Block verwirklicht werden.

```
1  Liste = [0,0,2,1,0,3]
2  i = 0
3
4  for element in Liste[0:]:
5      try:
6          zahl = 1 / element
7      except ZeroDvisionError:
8          print("Element",i,"ist eine Null")
9          i += 1
10     else:
11         print("Element",i,"ist keine Null")
12         i += 1
```

Listing 4.6: Try/Except

Try/Except: In dem Tryblock wird versucht eine Aktion auszuführen, tritt hierbei ein definierter Fehler auf, übernimmt der Except-Pfad, andernfalls der else-Pfad.
Im obigen Beispiel wird eine Liste mit 6 Elementen generiert und der integer i mit dem Wert 0. Im Beispiel wird versucht alle 0en ausfindig gemacht zu werden, da diese einen ZeroDivisionError erzeugen könne (Division durch 0). Für Jedes Element in der Liste wird nun versucht 1 durch das Element zu teilen. Falls unser gesuchter Fehler auftritt, wird "Element 'i' ist eine Null" ausgegeben und wenn kein Fehler auftritt, wird "Element 'i' ist keine Null" zurückgegeben.

5 Python 3

Nun soll es um die Unterschiede zwischen der alten und neuen Versionen gehen. Diese unterscheiden sich in großer Weise. Mit Python 3 wurd Python neugestaltet und ist zum größten Teil inkompatibel mit älteren Python Versionen.

Bibliothek und Module
Während Python 2 redundante und überflüssige Funktionen besitzt, wurden diese in Version 3 weggelassen, wodurch weniger Speicher benötigt wird. Auch die Module sind neu ansortiert, wodurch ein genutzter Intepreter zwischen den unterschiedlichen Versionen unterscheiden muss.

Print() und Input
Input nimmt eine Eingabe auf, währen print/print() eine Ausgabe gibt. Print, eine Python 2 Anweisung, wird in Version 3 durch die Funktion print() ersetzt. Dieser ist einfacher zu nutzen.
Die Input-Funktion hat in Version 2 eine große Sicherheitslücke in der die Eingabe als Objekt angesehen wird. Bei einer Eingabe wie 42 kann der Typ entweder als integer oder String angesehen werden, wodurch viele Probleme entstehen können, da der Typ nicht bestimmt werden kann. Dieses Problem wird in Version 3 gelöst, indem die Eingabe immer als String angesehen wird und deshalb der Eingabetyp immer klar definiert ist.

Integer
Während long, int und float, wie in Java, gängig waren, fällt der Typen Long raus und wird nun auch durch int dargestellt. Dies hat zwar den Nachteil, dass kleine Zahlen mehr Speicher verbrauchen, jedoch ist die Unterscheidung nicht mehr vonnöten. Außerdem werden float und true division vorgestellt. Dadurch ist das Unterscheiden zwischen Division mit und ohne Rundung möglich.

String
Der ASCII-Code findet sich in der neuen Version nicht mehr als Standard. Stattdessen wird Unicode genutzt, der weltweit gängiger ist. In beiden Versionen kann auf beiden Codierungem durch Spezifizierung zugegriffen werden.

Vergleiche
In Python 2 sind viele Vergleiche durchführbar, die nicht möglich sein sollten, wie z.B. der Vergleich Liste > String gibt True zurück. Solche Vergleiche finden sich in der neuen Version nicht mehr und liefert stattdessen einen Fehler, sodass das Vergleichsporblem gelöst ist.

Schleifen

Schleifen benutzen zumeist Variablen. Diese sollen natürlich nur Einfluss innerhalb der Schleife haben, wobei geleichbenannte Variablen in anderen Codeabschnitten unbeeinflusst bleiben sollen. Dieses "Feature" wird in Version 3 eingeführt, indem Schleifen lokal bleiben anstat global.

Type Hints

Da in Python alles als Objekte angesehen wird, ist es schwer Variablen voneinander zu entscheiden. z.B. 42 kann als Zahl sowie als String interpretiert werden. Um einen spezifischen Typen zu definieren, kann in Version 3 der Type Hint genutzt werden, indessen der Typ des Objektes festgelegt wird.

assert

Um Programmfehler zu vermeiden werden assert-Anweisungen genutzt. Diese Überprüfen die Gültigkeit eines Ausdrucks und geben ggf. eine Meldung zurück. Im Vergleich zu Version 2 sind assert-Anweisungen in Version 3 kürzer und vereinheitlicht, da Redundanzen abgeschafft wurden.

6 Zukunftsaussichten

Python bietet wie in den vorherigen Kapitel bereits genannt viele Vorteile für einen Entwickler. Dazu zählt, dass der entstandene Quellcode übersichtlich und leicht zu verstehen ist. Das spricht für ein anfängerfreundliches Umfeld, wodurch Python immer weiter an neuen Nutzern gewinnt.

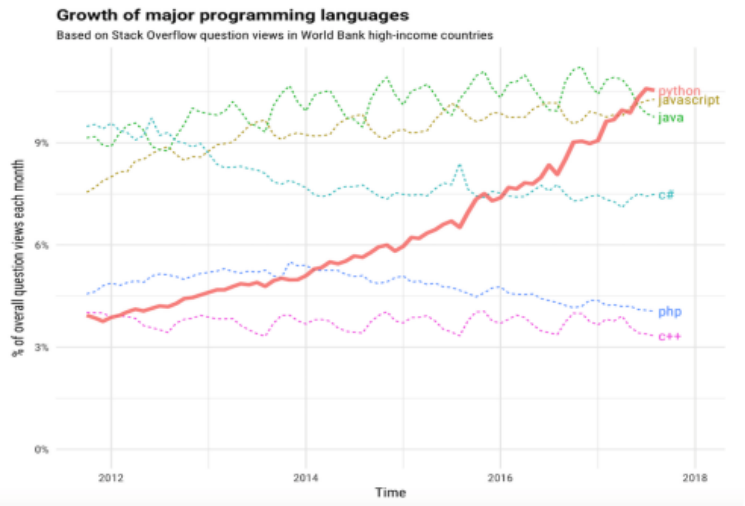

Figure 6.1: Wachstumsrate

Gezeigt wird der Wachstum an gestellten Overflow-Fragen im Zeitraum von 2012 bis 2018. Zu sehen ist, dass Python den größten Wachstum an gestellten Fragen hat, was impliziert, dass immer mehr Personen Python aktiv nutzen.

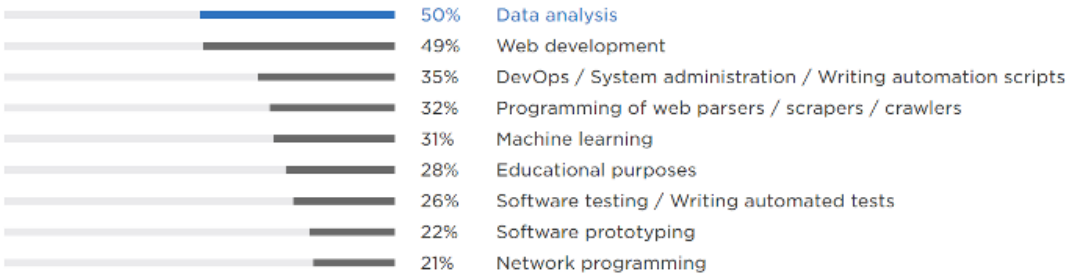

Figure 6.2: Nutzungsrate

Darum wird Python in verschiedensten Bereichen genutzt, wobei Datenverarbeitung und Webdevelopment sowie Machine Learning im Vordergrund stehen.

7 Zusammenfassung und Fazit

Zusammengefasst ist Python eine sehr übersichtliche und verständliche Sprache, die sich für den Einstieg in die Welt der Prgrammierung bestens eignet. Sie bringt viele Vorteile wie eingebaute Bibliotheken, einen interaktiven Modus und viele Hilfestellungen. Python ist zwar langsam, dafür auch sicher. Auch gibt es die Möglichkeiten die Schwächen der Sprache durch imports zu neutralisieren.

Mit Version 3.0 wurden viele Sicherheitslücken, Redundanzen und Probleme entfernt, sodass eine Entwicklung der Sprache zu spüren ist. Heutzutage wird Python in vielen Bereichen genutzt und bekommt jährlich mehr Anhänger.

Auch in Zukunft wird Python immer mehr Ansehen gewinnen, da im Laufe des Informationszeitalter die Datenwissenschaft immer weiter wachsen wird und dadurch neue Arbeitsplätze entstehen.

8 Internetquellen

- Unbekannt 21.05.20
  https://de.wikipedia.org/wiki/Python(Programmiersprache)
- Guido van Rossum 12.05.20
  https://docs.python.org/3.0/whatsnew/3.0.html
- Unbekannt 12.05.20
  https://lerneprogrammieren.de/python-2-vs-3/
- Rainer Grimm 14.05.20
  www.embedded-software-engineering.de/migration-auf-python-3-a-808804/
- Stefan Luber 14.05.20
  https://www.bigdata-insider.de/was-ist-python-a-730480/
- Alex Buerkle 16.05.20
  https://www.dev-insider.de/was-ist-python-a-843060/
- Andreas Wygrabek 21.05.20
  https://www.data-science-architect.de/python-compiliert-interpretiert/
- Serdar Yegulalp 16.05.20
  https://www.computerwoche.de/a/python-lernen-leicht-gemacht,3331244,2
- Bernd Hengelein 16.05.20
  https://cito.github.io/byte_of_python/read/features-of-python.html
- Tam Hanna 19.05.20
  entwickler.de/online/python/python-tutorial-einfuehrung-579865097.html
- Amandine Lee 19.05.20
  https://www.kite.com/blog/python/functional-programming/
- Klein 19.05.20
  https://www.python-kurs.eu/python3_kurs.php Bernd
- Sebastian Raschka 20.05.20
  https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html
- Barry Warsaw 20.05.20
  https://www.python.org/dev/peps/pep-0001/#what-is-a-pep

9 Bildquellen

- Figure 1: https://devopsworld.de/1133
- Figure 2: Eigenerstellung
- Figure 3 und 4: https://opensource.com/article/18/5/numbers-python-community-trends