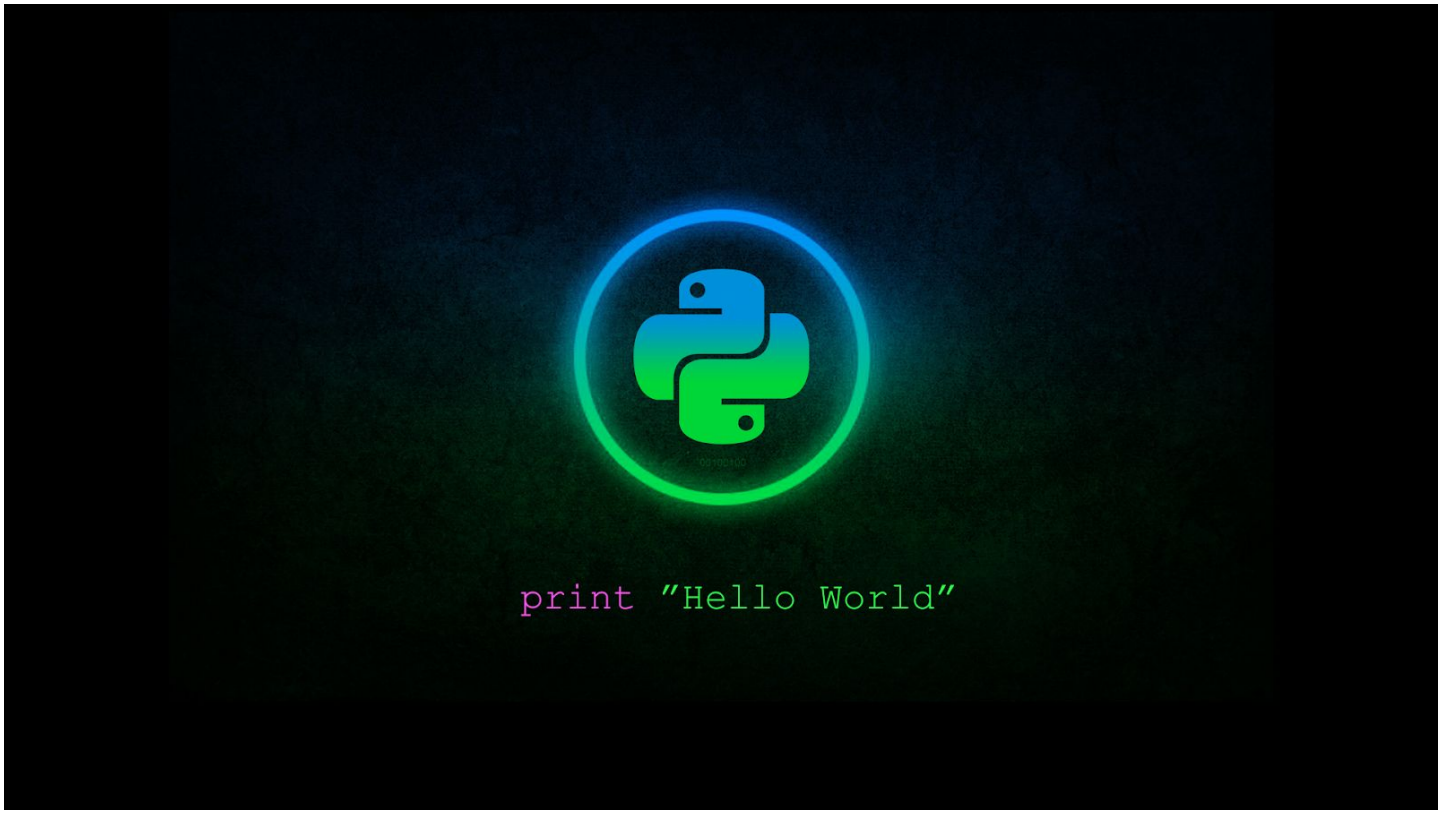
Simple Python
Persönliche Dokumentation



Erstellt von: Tim Irmler

Letzte Änderung am: 09-11-2019

1. Was ist Python

[Geben Sie eine Erklärung was Python ist]

Python ist eine Objekt-orientierte Programmiersprache. Wegen der guten Lesbarkeit hat Python enorm an Popularität gewonnen. Python ist als sehr einfache Sprache bekannt und so sehr beliebt bei Anfängern ist aber dennoch sehr mächtig.

Es gibt zwei Ansätze sowie eine Mischlösung, wie der Computer von der Programmiersprache zum Programm und dessen Ausführung kommt.

[Lesen Sie https://www.dev-insider.de/der-unterschied-von-compiler-und-interpreter-a-742282/ und erklären sie Compiler, Interpreter und die Mischlösung. In welche Kategorie gehört Python?]

Interpreter lesen den Code von oben nach unten in Echtzeit, dadurch kann man gut sehen wo ein Fehler vorliegt und ihn schneller beheben. Ein Compiler schreibt zuerst den Code um in Maschinensprache. Dies nimmt am Anfang ein wenig Zeit in Anspruch. Der Code kann ein Mensch nicht verstehen, aber die Maschine schon.

Der Compreter oder Just-in-time-Compiler übersetzt das Programm erst zur Laufzeit in Maschinencode. Einerseits bietet die Hybridlösung gute Performance kompilierter Programme, andererseits ermöglicht sie die komfortable Fehlersuche interpretierter Programme.

Python ist ein Interpreter.

2. Datentypen

Ein Datentyp beschreibt eine Menge von Datenobjekten, die alle die gleiche Struktur haben und mit denen die gleichen Operationen ausgeführt werden können.

2.1 Zahlen

[Erklären sie Integer, Float und Boolean]

Integer: Eine ganze Zahl, also zum Beispiel 1 oder 32, aber nicht 4.2.

Float: Eine Dezimalzahl, also zum Beispiel 7.21.

Boolean: Kann nur eine von 2 dingen sein, entweder Wahr/True/1 oder Falsch/False/0.

[Erklären sie welche Operationen man mit Zahlen machen kann]

Plus, Minus, Mal, Durch und Hoch.

Es gibt auch noch komplexere wie zum Beispiel Modulo.

2.2 Strings

[Erklären sie was Strings sind]

Strings sind Buchstaben, Zeichen oder ganze Wörter. Eine Zahl wird dementsprechend nicht als Zahl gesehen sondern als Buchstabe.

[Erklären sie welche Operationen man mit Strings machen kann]

Da Python das ganze nicht als Zahl sieht kann man an sich keine Operationen machen. Was an sich geht ist die verschiedenen Strings aneinander zu hängen.

2.3 Listen

[Erklären sie was Listen sind]

Eine Liste ist eine einzelne Variable in der man mehrere Datentypen speichern kann. Um etwas aus der Liste abzurufen muss man NameDerListe[zahl] eingeben, wobei die Zahl bei 0 beginnt.

[Erklären sie welche Operationen man mit Listen machen kann]

Vorausgesetzt die Daten in der Liste sind als Integer oder Float gespeichert kann man dieselben Operationen machen wie oben bereits genannt.

3. Kontrollstrukturen (Statements)

3.1 Indentation

Die meisten Programmiersprachen wie C, C++, Java benutzen Klammern wie { }, um einen zusammenhängenden Code Block zusammenzufassen.

Python benutzt «Indentation», übersetzt heisst das Einschübe.

Ein Codeblock (z.B. bei einer Funktion, einer Schleife, etc.) beginnt mit mit einem Einschub und endet mit der ersten nicht eingeschobenen Codezeile. Wieviel der Einschub ist, ist frei wählbar. Es muss einfach im ganzen Code einheitlich sein, sonst wird bei der Ausführung ein Fehler aufgeworfen.

Obwohl der Einschub frei wählbar ist, hat sich für den Einschub 4 Leerzeichen durchgesetzt. Hier im Beispiel (Einschübe in gelb):

```
1. for i in range(1,11):
2.     print(i)
3.     if i == 5:
4.         break
```

3.2 Kommentare

Einfache Programme sind mit einem guten Code oft selbsterklärend. Kommentarzeilen werden vom Interpreter nicht ausgeführt, somit können Sie mit Kommentaren ihren Code beschreiben. Besonders wenn sie eigene Funktionen, Klassen etc. entwerfen kann es sinnvoll sein den Code zu kommentieren, damit Sie auch nach einem halben Jahr wissen was ihr Code macht.

Kommentare kann man entweder nur auf einer Zeile oder mehreren Zeilen schreiben. Einzeilige Kommentare werden mit einem # eingeleitet.

```
1. # Ich bin ein Kommentar
2. # Ich auch!
```

Mehrzeilige werden mit drei Anführungszeichen eingeleitet und abgeschlossen.

```
"""
Der Kommentar beginnt hier...
und hört hier auf.
"""
```

3.3 Conditions – Bedingungen

[https://www.w3schools.com/python/python_conditions.asp erklären Sie wie man logische vergleiche macht]
Vergleiche funktionieren in dem man Zwei oder mehr Variablen miteinander vergleicht.

Mögliche vergleiche wären folgende:

Gleich, Nicht gleich, Grösser, Kleiner, Grösser oder gleich und so weiter.

Gleich wird dabei mit zwei Gleichheitszeichen geschrieben: ==

Ein Code würde dann folgend aussehen:

```
a = 30
b = 200
if b > a:
        Print(“b ist grösser als a”)
```

3.4 if, elif und else

[Erklären Sie wie if, elif und else funktionieren]

if oder auf Deutsch wenn benutzt man wenn man etwas überprüfen möchte. Beispiel, wenn A gleich gross wie B ist, mache folgendes.

elif kommt dann ins Spiel wenn das vorherige if nicht zustimmt. Wir müssen aber etwas neues angeben das überprüft werden soll. Also zum Beispiel, Wenn A kleiner wie B ist, mache folgendes.

else heisst auf Deutsch sonst, wenn also die ersten Zwei nicht zutreffen dann mach einfach das. Egal wann, mach es einfach.

Beispiel:

```
a = 200
b = 30
if a == b:
        Print(“a ist gleich gross wie b”)
elif a < b:
        Print(“a ist kleiner wie b”)
else:
        Print(“b ist grösser wie a”)
```

3.5 While-Schleife

[Erklären Sie wie die while-Schleife funktionieren]

Eine while Schleife führt einen Block von Code aus solange etwas zutrifft. While wird meistens verwendet wenn man nicht genau weiss wie lange etwas dauert.

Beispiel:

```
while ichSchlafe():
        warte()
```

3.6 for

[Erklären Sie wie und für was sie die for Schleife eingesetzt werden]

Eine for Schleife ist ähnlich wie die While Schleife. Sie führt einen Block code aus so lange etwas zutrifft. Allerdings werden for Schleifen mehr für Listen verwendet, oder wenn man weiss wie lange etwas dauert.

Beispiel:

```
list = ['a','b','c']
for i in list:
        print(i)
```

3.7 break und continue

[Erklären Sie wie break und continue eingesetzt werden]

break wird eingesetzt um die aktuelle Schleife zu verlassen sobald eine Bedingung zutrifft.

continue wird eingesetzt um einen Teil der aktuellen Schleife zu überspringen sobald eine Bedingung zutrifft.

4. Funktionen

[Für alle Fragen https://www.w3schools.com/python/python_functions.asp]

[Wie werden Funktionen definiert?]

Eine Funktion wird definiert in dem man zu beginn “def” schreibt und danach den Namen der Funktion. Nach dem Namen noch 2 Klammern. Also Funktion().

[Wie werden Funktionen aufgerufen?]

In dem man irgendwann innerhalb des Codes den namen der Variable eingibt.

Wichtig ist, das man nach dem Namen der Funktion 2 Klammern setzt. Also Funktion().

[Wie übergebe ich einer Funktionen einen Wert? (Parameter)]

Wenn bei der definition der Funktion innerhalb der Klammer ein Parameter angegeben wurde, zum beispiel “x”, muss ich beim aufrufen der Funktion etwas in der Klammer angeben. Beispiel:

```
def funktion(parameter):
        print(parameter)

funktion(Tim)
funktion(Irmler)
```

Der Output wäre dann folgendes:

Tim

Irmler

[Wie definiere ich Standardwerte für Parameter?]

Beim definieren einer Funktion kann man in der Klammer direkt angeben was der Standardwert des Parameters sein soll, dieser wird nur verwendet falls er nicht geändert wird. Beispiel:

```
def funktion(parameter = “Tim”):
        print(“Ich heisse “ + parameter)
```

[Was mache ich wenn ich nicht weiss wieviele Parameter kommen?]

Falls man eine Liste benutzt, wird es innerhalb der Funktion immer noch als Liste angesehen. Um alle Inhalte der Funktion auszugeben, sollte man wieder eine for Schleife verwenden.

[Wozu wird «pass» verwendet?]

Eine Funktion darf nicht leer sein, falls man aus einem Grund eine leere Funktion haben sollte, kann man mit “pass” verhindern das man einen Fehler bekommt.

5. Modularisierung

Standardmässig kommt mit der Installation von Python die «**Python Standard Library**». Diese Bibliothek ist sehr umfangreich und bietet eine breite Palette von Funktionen. Diese sind in Module organisiert (in C Programmiert) und eröffnen z.B. die Möglichkeit auf Systemfunktionen zuzugreifen um Dateien auszulesen und abzuspeichern (Input/Output), die ohne das Einbinden nicht möglich wäre. Es gibt ganz viele Module, die Ihnen das Programmieren einfacher machen sollen, damit sie gängige Probleme, wie Zufallszahlen generieren, nicht immer neu erfinden müssen. Weitere Beispiele für Funktionen von Modulen wären z.B. diejenigen die den Code Plattformunabhängig machen.

[lesen sie https://www.python-kurs.eu/modularisierung.php und erklären wie man ganze Module und wie man Teile von Modulen einbindet in den Code]

Mit dem Befehl “import” kann man Bibliotheken einbinden. Beispiel:

```
import math
```

Um nur einzelne Funktionen/Methoden aus der Bibliothek zu importieren kann man den Namen der Funktion/Methode verwenden. Beispiel

```
from math import sin, pi
```

Hier finden sie eine Liste von den Modulen der «**Python Standard Library**» für Python 3.8.

https://docs.python.org/3/library/