Python Tutorial

Ziel dieses Tutorials ist es, einen ersten Einblick in die Programmiersprache Python zu erhalten und selbst ein kleines TicTacToe-Spiel zu programmieren. Das Tutorial richtet sich dabei an Menschen, die noch nie programmiert haben, also weder in Python noch in einer anderen Sprache. Ich empfehle, die jeweiligen Codebeispiele immer selbst auszuprobieren und ein bisschen damit rumzuspielen. Was passiert beispielsweise, wenn du in dem jeweiligen Code einen Wert änderst? Du benötigst für dieses Tutorial:

- Einen Texteditor (bspw. Sublime Text oder Notepad++)
- Einen Terminal
- Eine Installation von Python 3

ODER:

- Einen Online-Playground (bspw. https://www.programming-hero.com/code-playground/python/index.html)

Einführung

Umgang mit dem Terminal

Falls du dich für den Online-Playground entschieden haben solltest, kannst du diesen Abschnitt überspringen.

Öffne deinen Terminal. Unter Windows kannst du dafür bspw. „cmd“ verwenden, indem du auf die Windowstaste auf deiner Tastatur drückst, „cmd“ eingibst und Enter drückst. Daraufhin sollte sich ein kleines schwarzes Fenster öffnen, das ungefähr so aussieht:

Unter Linux oder MacOS musst du hierfür nach dem Programm „Terminal“ suchen und dieses öffnen.

Mit Hilfe dieser geöffneten Kommandozeile kannst du nun deinem Computer Befehle geben. Der Dateipfad, der dir links oben angezeigt wird, ist dabei der Ordner, in dem du dich gerade befindest. Möchtest du wissen, was in diesem Ordner alles enthalten ist, kannst du unter Windows den Befehl **dir** und unter Linux oder MacOS den Befehl **ls** verwenden und anschließend Enter drücken. Probiere es einmal aus!

Dir sollten nun alle Unterordner und Dateien, die in deinem Ordner liegen, aufgelistet werden. Möchtest du in einen dieser Unterordner wechseln, so gib **cd** gefolgt von dem Namen des Unterordners ein. Damit sollte sich der Pfad, der links neben deiner Eingabe steht, ändern. Möchtest du wieder zurückwechseln in den Ordner darüber, so verwende den wieder Befehl cd gefolgt von zwei Punkten, also **cd ..**

Öffne nun deinen Texteditor und gib folgendes ein:

```
print("Hello world!")
```

Speichere die Datei unter dem Namen „hello.py“ in einem Ordner deiner Wahl. Die Dateiendung „.py“ gibt an, dass es sich um ein Pythonskript handelt. Um den Code auszuführen, wechsle in deinem Terminal mit Hilfe von **cd** in den Ordner, in dem du die Datei abgelegt hast. Gib anschließend den Befehl **python hello.py** ein. In deinem Terminal sollte nun der Text „Hello world!“ erscheinen. Auf diese Weise kannst du geschriebene Python-Programme ausführen. (Hinweis: Sollte das nicht funktionieren, versuche es stattdessen mit **py hello.py** oder **python3 hello.py**)

Operationen und Variablen

Python verfügt über einige eingebaute Rechenoperationen wie zum Beispiel +, -, * und /. Gib in deinem Texteditor oder in deinem Online-Playground folgendes ein:

```
print(3+2)
```

Führe das Programm aus. print() ist eine eingebaute Funktion. Alles, was in den Klammern steht, wird erst ausgewertet und anschließend angezeigt. Du solltest nun also als Ausgabe „5“ erhalten. Probiere auch die anderen Rechenoperationen aus. Hinweis: Kommazahlen werden in Python nicht mit einem Komma, sondern einem Punkt versehen. Also 3.5 und nicht 3,5.

Darüber hinaus gibt es noch andere Operationen:

- == überprüft, ob zwei Werte gleich sind. 3==3 resultiert dabei in True, „3==5“ in False.
- != überprüft, ob zwei Werte ungleich sind. „3!=5“ resultiert dabei in True, „3!=3“ in False.
- % gibt den Rest aus, der bei einer Division übrig bleibt: „5%2“ ergibt also 1, da bei der Teilung von 5 durch 2 der Rest 1 übrig bleibt.

Möchte man das Ergebnis einer solchen Berechnung wiederverwenden, so kann man eine Variable einführen und diese anschließend für weitere Berechnungen benutzen:

```
summe = 3 + 2

print(summe % 2)
```

Hier haben wir als Namen für die Variable „summe“ gewählt. Du kannst deine Variablen aber generell benennen, wie du möchtest. Dabei sollten Variablennamen möglichst nur kleine Buchstaben oder Ziffern enthalten. Leerzeichen und einige Sonderzeichen sind nicht erlaubt. Möchtest du einen Namen wählen, der mehr als ein Wort enthält, so empfiehlt es sich, die einzelnen Wörter mit einem Unterstrich zu trennen, also bspw. summe_aus_drei_und_zwei. Wähle möglichst gute Namen, damit du später immer noch weißt, was gemeint ist.

Du kannst dabei nicht nur Zahlen in einer Variable speichern. Eine weitere Möglichkeit sind sogenannte Strings. Strings sind Zeichenketten. Auch mit ihnen kann man bestimmte Operationen ausführen:

```
hallo = "Hallo, mein Name ist "

name = "Hannah"

anderer_name = "Helena"

print(hallo + name)

print(hallo + anderer_name)

print(name == anderer_name)
```

Bedingungen

Manchmal möchte man Code nur dann ausführen, wenn eine bestimmte Bedingung erfüllt ist. Dafür können sogenannte if-Statements benutzt werden:

```
zahl = 25
if zahl % 5 == 0:
        print("Die Zahl ist durch fünf teilbar.")
```

Was genau macht der obige Code? Probiere aus, was passiert, wenn du den Wert der Variablen „zahl“ änderst. Kannst du das Programm so abändern, dass es überprüft, ob bei der Teilung durch 7 ein Rest von 2 übrigbleibt? Passe die Ausgabe des Programms, also das, was bei print in den Klammern steht, entsprechend an.

Du solltest dir merken, dass du in der Zeile nach dem Doppelpunkt einrücken musst. Alles, was dann in den darauffolgenden Zeilen eingerückt ist, gehört noch zum if-Statement und wird nur ausgeführt, wenn die entsprechende Bedingung erfüllt ist. Probiere aus, was passiert, wenn du die Einrückung weglässt.

Man kann einem Programm auch mehr als nur eine Bedingung für die Ausführung von Code geben. Probiere den untenstehenden Code aus und verändere den Wert der Variablen zahl. Was ist der Unterschied zwischen **and** und **or**?

```
zahl = 15
if zahl % 2 == 0 and zahl % 3 == 0:
        print("Die Zahl ist durch 2 und durch 3 teilbar.")
if zahl % 2 == 0 or zahl % 3 == 0:
        print("Die Zahl ist durch 2 oder durch 3 teilbar.")
```

Schleifen

Manchmal möchte man Code nicht nur einmal, sondern gleich mehrfach hintereinander ausführen. Dafür kann man sogenannte for-Schleifen benutzen.

```
for i in range(3):
        print("Hallo!")
```

Was macht dieser Code? Was passiert, wenn du range(3) durch range(5) ersetzt? Und was passiert, wenn du print("Hallo!") durch print(i) ersetzt?

Du wirst bemerkt haben, dass der Wert der Variable i bei jedem Durchlauf der Schleife um eins erhöht wird. Außerdem sollte dir aufgefallen sein, dass Python nicht bei 1 anfängt zu zählen, wie du das vermutlich gewohnt bist, sondern bei 0. Die Variable i kannst du dabei natürlich auch anders benennen.

Auch hier gilt wieder, dass du in der Zeile nach dem Doppelpunkt einrücken musst. Alles, was in den darauffolgenden Zeilen eingerückt ist, wird dann innerhalb der Schleife ausgeführt.

Innerhalb einer for-Schleife kannst du auch wieder if-Bedingungen verwenden:

```
for i in range(20):
	print(i)
	if i % 2 == 0:
		print("Die Zahl ist gerade.")
	if i % 2 == 1:
		print("Die Zahl ist ungerade.")
```

Beachte hier auch wieder die Einrücktiefen und spiele ein bisschen damit herum. Was passiert, wenn du sie veränderst?

FizzBuzz

An dieser Stelle bist du nun bereit für deine erste kleine Programmierübung. Diese Übung nennt sich FizzBuzz und wird tatsächlich manchmal in Bewerbungsgesprächen für Programmierer:innen gefordert. Die Aufgabe lautet wie folgt:

Schreibe ein Programm, das alle Zahlen der Reihe nach bis 100 ausgibt. Falls eine Zahl durch 3 teilbar ist, soll statt der Zahl der String "Fizz" ausgegeben werden. Ist sie durch 5 teilbar, soll stattdessen "Buzz" ausgegeben werden. Ist sie sowohl durch 3 als auch durch 5 teilbar, soll sowohl "Fizz" als auch "Buzz" ausgegeben werden.

TicTacToe

Anforderungen

Wir kommen nun zur eigentlichen Programmieraufgabe. Ziel ist es, dass du am Ende ein kleines TicTacToe-Spiel hast, das man zu zweit spielen kann. Dafür überlegen wir uns zuerst, was wir für ein solches Spiel brauchen:

- Wir brauchen ein Spielfeld der Größe 3x3.
- Die Spieler müssen abwechselnd die Möglichkeit haben, x bzw. o an einer bestimmten Stelle auf dem Spielfeld zu setzen.
- Es muss bestimmt werden, wann das Spiel vorbei ist und ob einer der Spieler gewonnen hat.
- Bonus: Es muss sichergestellt werden, dass die Spieler nur valide Eingaben machen können.

Das Spielfeld

Wir beginnen mit der Erstellung eines Spielfelds. Wir wissen, dass ein TicTacToe-Feld aus 9 Zellen besteht. Diese Zellen sollen zu Beginn des Spiels einen Unterstrich enthalten, also einen String "_", um anzuzeigen, dass noch kein Spieler hier etwas eingetragen hat. Zur Darstellung des Feldes können wir eine sogenannte Liste verwenden:

```
spielfeld = ["_", "_", "_", "_", "_", "_", "_", "_", "_"]
```

Eine Liste ist demnach eine Datenstruktur, die mehr als nur einen Wert enthalten kann. In unserem Fall enthält sie nun neunmal einen Unterstrich. Natürlich ist es irgendwie umständlich, in der Liste neunmal das gleiche zu schreiben. Stattdessen kann man wieder eine for-Schleife benutzen:

```
spielfeld = ["_" for i in range(9)]
```

Möchte man nur einen der Werte, den die Liste enthält, anzeigen, kann man das wie folgt:

```
print(spielfeld[2])
```

In diesem Fall wird der dritte Wert, der in der Liste steht, angezeigt. Warum nicht der zweite? Wir erinnern uns: Python fängt bei 0 an zu zählen, nicht bei 1. In unserem Fall macht das aber natürlich keinen Unterschied, weil in der Liste neunmal der gleiche Wert steht. Probiere aus, was passiert, wenn du eine Liste mit verschiedenen Werten erstellst.

Möchte man einen Wert, der in der Liste steht, verändern, so kann man das wie folgt machen:

```
spielfeld[2] = "neuer Wert"
```

Nun steht an dritter Stelle der Liste kein Unterstrich mehr, sondern "neuer Wert".

Jetzt haben wir also ein Spielfeld mit 9 Zellen. Gib die komplette Liste mit Hilfe von print() aus. Dir sollte auffallen, dass nun alle Zellen nebeneinander angezeigt werden, was natürlich nicht das ist, was wir wollen. Wir wollen 3 Zeilen untereinander mit jeweils 3 Zellen. Das kann man beispielsweise so machen:

```
print(spielfeld[0], spielfeld[1], spielfeld[2])

print(spielfeld[3], spielfeld[4], spielfeld[5])

print(spielfeld[6], spielfeld[7], spielfeld[8])
```

Das funktioniert zwar, ist aber nicht unbedingt schön, da wir dreimal hintereinander fast den gleichen Code haben. Möchte man mehrfach hintereinander den gleichen oder fast den gleichen Code ausführen, kann man eine for-Schleife benutzen. Fällt dir eine Möglichkeit ein, die Schleifenvariable i geschickt zu benutzen?

Benutzereingaben und Runden

Als nächstes sollen die Spieler unseres kleinen Spiels x bzw. o an einer bestimmten Stelle des Spielfelds setzen können. Dafür sollen sie die Möglichkeit bekommen, über den Terminal eine Zahl einzugeben. Dazu kann man die Funktion input() verwenden:

```
zelle = input("In welche Zelle möchtest du dein Zeichen setzen? ")
```

Nun wird der Spieler aufgefordert, eine Zahl einzugeben. Diese Zahl wird dabei in der Variablen zelle gespeichert. Setze an der entsprechenden Stelle im Spielfeld das Zeichen "x". Was passiert, wenn der Spieler nun eine Zahl eingibt, die größer als 8 ist?

Hinweis: Um dieses Problem kümmern wir uns später. Wir gehen jetzt erstmal davon aus, dass unsere Spieler schlau genug sind, um nur Zahlen zwischen 0 und 8 einzugeben.

Wir können nun an einer bestimmten Stelle auf dem Spielfeld ein Zeichen setzen. Das ist schon mal gut, allerdings wollen wir ja nicht nur einmal ein Zeichen setzen. Ein TicTacToe-Spiel hat bis zu 9 Runden. Was musst du machen, um deinen Code insgesamt neunmal auszuführen?

Vermutlich bist du auf die Idee gekommen, eine for-Schleife zu benutzen. Damit hat dein Spiel nun 9 Runden und der Spieler wird neunmal aufgefordert, eine Eingabe zu machen. Allerdings wird im Moment immer "x" in die jeweilige Zelle eingetragen. Wir wollen ja aber abwechselnd "x" oder "o" eintragen, je nachdem, welcher Spieler gerade dran ist. Wie können wir nun bestimmen, welcher Spieler gerade dran ist? Wir haben eine Schleifenvariable, die angibt, in welcher Runde wir uns gerade befinden. In den Runden 0, 2, 4, 6 und 8 ist der Spieler mit dem Zeichen x dran, in allen anderen der Spieler mit dem Zeichen o. Oder anders formuliert: Wenn der Wert der Schleifenvariablen durch 2 teilbar ist, müssen wir ein x setzen, sonst ein o. Schreibe den entsprechenden Code.

Das Spiel gewinnen

Im Moment hat unser Spiel immer genau 9 Runden, selbst wenn es schon vorher gewonnen wurde. Wir wollen unser Programm nun so ändern, dass es "Gewonnen!" ausgibt, wenn ein Spieler gewonnen hat, und dann abbricht.

Damit ein Spieler gewinnt, muss er sein Zeichen entweder dreimal in einer Zeile, in einer Spalte oder diagonal gesetzt haben. Überlege dir, wie du das überprüfen kannst. Welche Kombinationen gibt es? Gib in einem solchen Fall mit Hilfe von print() "Gewonnen!" (oder eine andere Siegesnachricht deiner Wahl) aus und brich die for-Schleife ab.

Hinweis: Um eine Schleife zu beenden, kannst du den Befehl **break** benutzen. Die folgende Schleife wird abgebrochen, nachdem i den Wert 5 erreicht hat:

```
for i in range(10):
        print(i)
        if i == 5:
                break
```

Nun hast du ein funktionierendes kleines Spiel! Glückwunsch 😊

Auf den folgenden Seiten findest du kleine Bonusaufgaben, die dir dabei helfen, sowohl dein TicTacToe-Spiel als auch dein FizzBuzz zu verbessern.

Bonus: elif und else

Probiere den folgenden Code aus und ändere die Werte von erste_zahl und zweite_zahl. Was macht **else**?

```
erste_zahl = 3

zweite_zahl = 5

if erste_zahl == zweite_zahl:

        print("Die Zahlen sind gleich.")

else:

        print("Die Zahlen sind nicht gleich.")
```

Mache das gleiche mit dem folgenden Code und überlege dir, was **elif** bedeutet.

```
erste_zahl = 3

zweite_zahl = 5

if erste_zahl > zweite_zahl:

        print("Die erste Zahl ist größer.")

elif erste_zahl < zweite_zahl:

        print("Die zweite Zahl ist größer.")

else:

        print("Die Zahlen sind gleich.")
```

Kannst du deine bisher geschriebenen Programme, also das FizzBuzz und das TicTacToe-Spiel, mit Hilfe von **else** und **elif** verbessern?

Bonus: Funktionen benutzen

TODO

Bonus: Benutzereingaben validieren

Unser Spiel hat noch ein paar kleine Probleme. Die Spieler können im Moment allen möglichen Blödsinn eingeben, mit dem unser Programm nichts anfangen kann. Wir wollen uns an dieser Stelle um zwei dieser Probleme kümmern:

- Sie können eine Zahl eingeben, die größer als 8 ist.
- Sie können eine Zelle, die bereits belegt ist, überschreiben.

Damit das nicht passiert, wollen wir die Eingaben der Spieler validieren.

TODO