Einführung in Python

Seray Cetin, BSc.

23. Jänner 2019, FH Wr. Neustadt

1 Allgemein

Python ist ein Open-Source-Software und wurde Ende 1980 von Guido van Rossum erfunden. die Programmiersprache beruht auf eine früheren Programmiersprache namens ABC.

1.1 Python Versionen

Es gibt 2 Versionen von Python, nämlich Python 3.x und Python 2.x. Nicht jedes Python 2 in der Version 3 funktioniert bzw. mit diesem kompatibel ist. Nur noch bis 2020 soll Python 2 gewartet werden.

1.2 Entwicklungsziele

- einfache, aber mächtige Programmiersprache
- Code leicht verständlich
- für Alltagsaufgaben zur Verfügung stehen und kurze Entwicklungszeiten fördern

1.3 Programmiersprache

Python ist eine moderen Programmiersprache mit folgenden Eigenschaften:

- Dynamische Typisierung: vor der Verwendung müssen Variablen nicht deklariert werden
- Objektorientierung: Klassen, Objekte mit Mehoden und Attributen
- leitsungsfähige Datentypen: int, float, string, list, tupel, dictionary, ..
- automatische Speicherverwaltung: Verwaltung von Speicher für Objekte
- Standardbibliothek: Biblioteken für Netzwerkzugriff, Unterstützung Internet-Protokolle
- Erweiterbarkeit: Module können in C/C++ geschrieben werden
- Portabilität: auf allen Betriebssystemen lauffähig
- freie Verfügbarkeit: Open-Source
- Performance

1.4 Erstes Programm

```
In [1]: print("Hello World!")
```

```
Hello World!
```

2 Code Intentation, Markdown & Basis of Jupyter

2.1 Code Intentation

In Python ist die Struktur (Formatierung) des Programmaufbaus sehr wichtig, da hier wie in Java keine Semikolon verwendet werden müssen, um das Ende der Zeile zu kennzeichnen. Das beudeutet es müssen entsprechende Einrückungen gemacht werden. Solche Codeeinrückungen wären beipsielsweise bei if-Anweisungen, Schleifen oder Funktionen wichtig. Hier dazu ein Beispiel:

```
In [ ]: if Bedingung:
            print('Bedingung ist erfüllt')
        else:
            print('Bedingung ist nicht erfüllt')
```

2.2 Markdown

Neben Code-Felder gibt es in Jupyter auch Markdown Felder. Diese sind im Prinzip nichts anderes als Textfelder. Diese Textfeld entsprechen Eins-zu-Eins den HTML-tags.

Beispiel: dicker Text Text unterstrichen Text kursiv Farbiger Text

2.3 Basis of Jupyter

Das Jupyter Notebook ist ein leistungsfähiges Werkzeug für interaktive Entwicklung und Präsentation von Data Science-Projekten. Der darin integrierter Code und die Ausgabe sind in einem einzigen Dokument. Diese könne auch zugleich heir auch visualisiert werden, auch können mathematische Gleichungen und andere Elemente kombiniert werden.

3 Variablen, Daten Typen

Variablen sind eine Art Behälter, die bestimmte Werte bzw. Ergebnisse speichern. Variable können durch andere Werte überschrieben bzw. neu gesetzte werden. Die Zuweisung der Variable wird mittel einem = durhgeführt, dieses Sysmbol darf nicht als das mathematische Gleichheitszeichen gesehen werden, sondern als Zuweisung. Hier einige Zuweisungen von Variablen und Berechnungen dieser:

```
In [2]: a = 1
        b = 2
        c = a + b
        print(c)
```

```
3
```

Variablen können bspw. eine Zahl oder ein Text beinhalten. Das bedeutet, Variablen unterscheiden sich anhand ihrer Datentypen. Folgende Datentypen stehen uns zur Verfügung:

Zahlen:

- `integer`: Ganzzahlen
- lange Ganzzahl
- Fließkommazahlen
- komplexe Zahlen

```
In [3]: x = 2    # integer
        x * 7

Out[3]: 14

In [21]: 9.0/3    # float

Out[21]: 3.0
```

Boolean: Boolean sind Subtypen von Integer. Sie können den Wert `True` (1) oder `False` (0) annehmen, je nachdem ob die Bedingung wahr der falsch ist.

```
In [4]: 1 == 1    # True

Out[4]: True

In [2]: 1 > 10    # False

Out[2]: False
```

Strings: Ist eine Sequenz von einzelnen Zeichen. Das bedeutet im Pinzip das ein String nichts weiter als eine Art Liste ist.

```
In [7]: s = "Hello, I'm "
        s + "Python!"

Out[7]: "Hello, I'm Python!"
```

Indexing von Strings: Der Buchstabe P hat den Index 0, Y hat Index 1 usw. und der letzte Buchstabe N den index 5. Indexing geht auch umgekehrt, von links nach rechts, da beginnen wir mit -1, also kann der Buchstabe N auch mit dem index -1 ausgegeben werden.

```
In [33]: text = 'PYTHON'
         print(" 0 - " + text[0])
         print(" 1 - " + text[1])
         print(" 3 - " + text[3])
         print("-1 - " + text[-1])
         print("-3 - " + text[-3])

 0 - P
 1 - Y
 3 - H
-1 - N
-3 - H
```

String - Slicing (teilen): Den Intervall der Indexe eingeben, je nachdem welchen bestimmten Bereich des Strings man ausgeben möchte.

```
In [35]: text[3:]   # gibt alle Zeichen ab dem 3. Zeichen aus

Out[35]: 'HON'
```

3.0.1 Wichtig!

Beim Deklarieren wird der Datentyp in Python nicht angegeben, dies übernimmt Python automatisch. Mit der `type()`-Funktion können die Datentypen der Variablen ausgeben werden.

```
In [20]: print(type(1))
         print(type(2.5))
         print(type("hello"))

<class 'int'>
<class 'float'>
<class 'str'>
```

Variablen in einen anderen Datentypen zu konvertieren, ist ebenfalls möglich und ganz simpel. In unserem Beispiel hier konvertieren wir einen String-Wert in ein Integer:

```
In [26]: a = "42"
         print(a)
         print(type(a))

         x = int(a)      # Konvertierung des String in Integer
         print(x)
         print(type(x))

42
<class 'str'>
42
<class 'int'>
```

4 Fließkommadarstellung

Gleitkommazahlen (= Fließkommazahlen) können auf sehr viele Arten darstellen werden. Wichtig die Kommazahl muss mit einem "Punktünd nicht mit einem "Beistrich"geschrieben. Fließkommazahlen haben den Datentypen `float` (floating point).

```
In [42]: a = 3.35678
         b = 2.          # 2.0
         c = .2          # 0.2
         d = 2e10        # 20000000000.0
```

Die Zahlen können mit `round()` gerundet werden. Optional kann die Anzahl der Nachkommastelle angegeben werden, z.B.

```
In [40]: round(a, 2)

Out[40]: 3.36
```

Das Rechnen mit Kommazahlen ist mit Rundungsfehlern verbunden, was zu einigen Ungenauigkeiten führen kann. Aber für einfache Berechnungen, sollte die Genauigkeit der float-Rechnungen ausreichend genug sein.

5 Logic & Control Flow

5.1 if-Anweisungen

Für die `if`-Anweisungen sind keine geschwungenen Klammern nötig, wie in Java, C++ und anderen Programmiersprachen. Hier ist es lediglich wichtig, dass der Code, der in die if-Klausel gehört, unter der Anweisung dementsprechend eingerückt ist. Es ist möglich gleich nach der if-Anweisung mittels `elif` (else if) weitere Bedingungen zu überprüfen bzw. hinzuzufügen. Zum Schluss ist ein `else`-Zweig optional aber nicht verplichtend.

```
In [32]: if 1 > 2:
             print("Die 1. Bedingung ist True")
         elif 2 == 2 and 2 > 1:
             print("Die 2. Bedingung ist True")
         else:
             print("Die Bedingung ist False")

Die 2. Bedingung ist True
```

6 Schleifen

Schleifen werden verwendet, wenn bestimmte Berechnungen iterativ (n-mal) durchgeführt werden müssen. Hier werden zwei Möglichkeiten gezeigt eine Schleife zu programmieren:

6.1 for-Schleife

Die `for-Schleife` benötigt eine bestimmte `range`, damit das System weiß wir oft sie durchlaufen muss.

```
In [36]: for i in range(10, 21):
             print(i*i, end=' ')

100 121 144 169 196 225 256 289 324 361 400
```

6.2 while-Schleife

Die `while-Schleife` arbeitet solange eine bestimmte Bedingung erfüllt ist.

```
In [35]: i = 0
         while i < 4:
             print ("i ist", i)
             i = i + 1
```

```
i ist 0
i ist 1
i ist 2
i ist 3
```

Natürlich können die Schleifen auch frühzeitig unterbrochen bzw. beendet werden, wenn eine bestimmte Bedindungen erreicht wird. Hierzu wird mit einer if-Anweisung die Bedingung abfragt und mit `break` die Schleife abgebrochen, wenn diese Bedingung eintrifft.

```
In [45]: for i in "Hallo Welt!":
             if i == " ":
                 break
             print (i,end='')
```

```
Hallo
```

7 Lists & Dictionaries

7.1 Listen

`Listen` werden in eckiger Klammer geschrieben. Die Elemente innerhalb der Klammer werden von einfachen Kommas getrennt.

```
In [26]: liste = ['red', 'black', 'white']
         liste
```

```
Out[26]: ['red', 'black', 'white']
```

Mit `append` werden in die Liste dahinter, neue Elemente hinzugefügt.

```
In [28]: liste.append("green")
         liste
```

```
Out[28]: ['red', 'black', 'white', 'green', 'green']
```

Die Elemente können innerhalb der Liste verschiedene Datentypen haben.

```
In [8]: mixed_list = [1, 2, 'birne']
        mixed_list
```

```
Out[8]: [1, 2, 'birne']
```

```
In [32]: mixed_list[2]
```

```
Out[32]: 'melone'
```

7.2 Dictionaries

`Dictionary` entspricht einem einfachen "Mapping". Ein Dictionary ist ein assoziatives Feld. Diese besteht aud mehreren Schlüssel-Objekt-Paaren. Jedes Objekt hat einen bestimmten Schlüssel. Hier werden geschwungene Klammer verwendet und keine eckigen, wie bei List. Im folgenden erstellen wir ein einfaches Dictionary:

```
In [5]: woerter = {"house":"Haus", "cat":"Katze", "black":"schwarz"}

In [6]: woerter["house"]

Out[6]: 'Haus'

In [8]: woerter["cat"]

Out[8]: 'Katze'
```

8 Meine erste Funktion

`Funktionen` werden definiert, um immer wiederkehrende Codezeilen über diesen auszuführen. Die Arbet wird erleichtert und der Code ist übersichtlicher. Unsere erste Funktion ist die Addition von zwei Zahlen:

```
In [29]: def zahl_addieren(zahl_1, zahl_2):
             return zahl_1 + zahl_2    # Zahlen werden addiert und zurück übergeben

In [30]: zahl_addieren(1, 2)      # Funktion mit den Zahlen afrufen

Out[30]: 3
```

Literatur

[1] Tuxcademy: Programmieren in Python - Eine praktische Einführung,
`https://www.tuxcademy.org/download/de/pyth/pyth-de-manual.pdf`

[2] Toby Donaldson: Flow of Control in Python,
`http://www.peachpit.com/articles/article.aspx?p=1312792&seqNum=3`

[3] Benjamin Pryke: Jupyter Notebook for Beginners: A Tutorial,
`https://www.dataquest.io/blog/jupyter-notebook-tutorial/`

[4] Bernd Klein. *Einführung in Python 3*. HANSER-Verlag, 2017.

[5] Mohamed Auf: Datenquellen & Datenaufbereitung - Vorlesungsunterlagen