



# Einführung in Python

Überarbeitete Version des Seminars von den Studierenden  
Bartel und Schuhmacher. Ihnen vielen Dank für die Freigabe  
der Originalunterlagen

Version 2.0 SH 21.10.2021

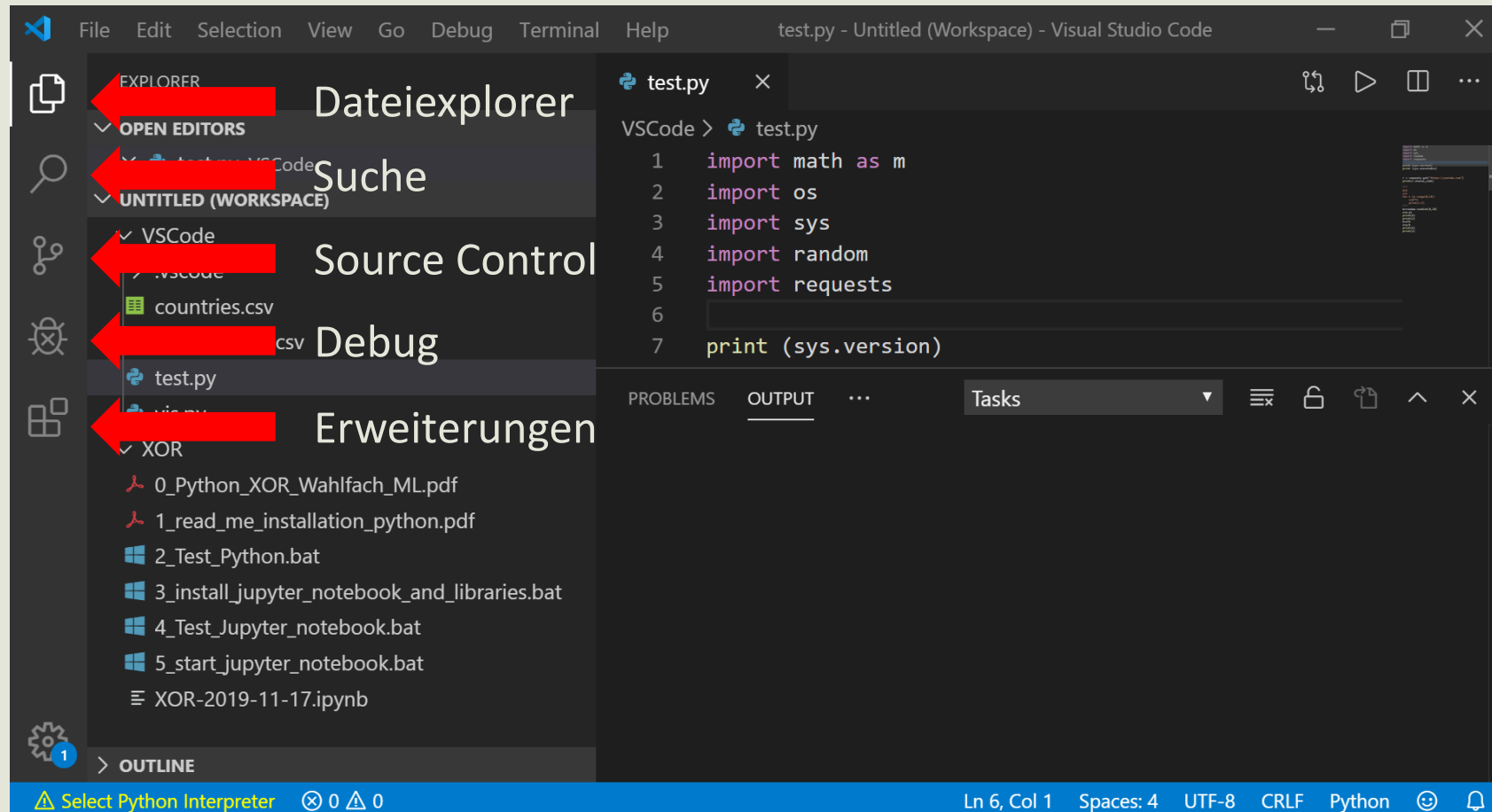
# Inhaltsverzeichnis

1. Python Grundlagen
2. Visual Studio Code
3. Erstes Testprogramm selbst schreiben
4. Python Befehle etc.
  - Bibliotheken
  - Variablen/Datentypen
  - Eingabe und Ausgabe in Konsole
  - Text aus einer Datei lesen
  - for-Schleife
  - Schreiben in eine Datei
  - while-Schleife
  - With
- 5. Python Program vis.py verstehen
- 6. Aufgabe: Einwohner Vergleich von Ländern
- 7. Quellen
- Anhang

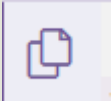

# 1. Python/Grundlagen

- höhere Programmiersprache (meist als Interpreter)
- Plattformunabhängig (Vorteil von Interpreter-Sprachen)
- gilt als einfachere Programmiersprache
- übersichtliche Syntax, wenig Schlüsselwörter, umfangreiche Standardbibliotheken
- Programme werden über das Einrücken strukturiert, nicht mit Klammern
- geschweifte Klammern und Semikolons entfallen
- 😊 nicht nach der Schlange benannt, sondern nach Monty Python (UK Komikergruppe)

## 2. Visual Studio Code: Please open visual studio code and check



## 3. 1. Test Programm schreiben Hello World

- Ctrl+N oder Oben links auf File und dann New file auswählen.
- Ctrl+Shift+S oder Save as test.py in Ihrem Datenordner der vorgegeben ist.
- Klicken auf  und Sie sehen Ihre Datei
- Nun bitte Code einfügen
- Z.B. **print**('hello world') oder gleichwertig
- `print('hello world')`
- Zum Ausführen bitte in der Mitte  drücken oder Ctrl+F5
- Das Ergebnis sehen Sie je nach Einstellung unten / Seite im Terminal



## 4. Python Grundlagen: Bibliotheken

- umfangreiche Bibliotheken
- matplotlib: Visualisierung (2D und 3D)
- Pandas: Datenverarbeitung- und Analyse
- NumPy: wissenschaftliches Rechnen (Arrays, Matrizen)
- seaborn: Visualisierung basierend auf matplotlib
- Bitte nehmen Sie genau unsere Libraries. S. 3 Install\_libraries.bat mit genau den angegebenen Versionen. Keine anderen sind in Seminaren etc. erlaubt.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
```

## 4. Variablen/Datentypen

- Variablen können ohne Datentyp eingelesen werden
- dynamische Typendeklaration (Wert und Typ )
- Datentypen werden automatisch zugewiesen  
n=5 , n wird als int definiert
- Datentypen können festgelegt werden  
n=float(5), n wird dem Datentyp float zugewiesen
- type("Variable"), gibt den Typ der Variable aus
- Unterstützt auch komplexe Zahlen

```
counter = 100 # integer Zahl  
miles = 1000.0 # floating point Zahl  
name = "Jan" # ein string  
cmplx = 10 + 4j # eine komplexe Zahl  
print(counter, miles, name, cmplx)  
print(type(counter), type(miles), type(name), type(cmplx))
```

## 4. Eingabe und Ausgabe in Konsole

- `input()` um von der Konsole einzulesen
- `print()` um in Konsole auszugeben

```
1  n=int(input("Zahl von 1 bis 5:"))
2  while n<6:
3      print("Wert:", n)
4      n = n+1
5  print("Fertig")
```



## 4. for-Schleife & Einrückungen

- range()-Funktion anstatt for(i=1, i<5, i++) // unterschied zu C
- einrücken in die for-Schleife, sonst wird es nicht von der Schleife durchlaufen
- {} werden somit nicht mehr benötigt
- des weiteren gibt es einen else block, welcher nur ausgeführt wird, wenn alle Elemente vorher durchlaufen wurden

```
1  for i in range(1, 5):  
2      print("Wert:" , i)  
3      print("fertig")
```

```
Wert: 1  
Wert: 2  
Wert: 3  
Wert: 4  
fertig
```



```
1  for i in range(1, 5):  
2      print("Wert:" , i)  
3      print("fertig")
```

```
Wert: 1  
fertig  
Wert: 2  
fertig  
Wert: 3  
fertig  
Wert: 4  
fertig
```



## 4. while-Schleife

- Verwendung der while-Schleife wie in C
- while-schleife durchlaufen → else („Korrekte Eingabe“) wird ausgeführt
- break sorgt dafür, dass die Schleife unterbrochen wird, somit wird das else der while-Schleife nicht durchlaufen
- continue → Schleifendurchlauf wird abgebrochen, springt zum Anfang der Schleife

```
1  n=int(input("Zahl zwischen 1 und 10:\t"))
2  while(n>10 or n<1):
3      if(n>10):
4          print("Zahl ist zu groß!")
5          break
6      else:
7          print("Zahl ist zu klein!")
8          break
9  else:
10     print("n=",n)
11     print("Korrekte Eingabe")
```

## 4. Text aus einer Datei lesen

- verschiedene Möglichkeiten Daten zu lesen
- 1. mithilfe der pandas Bibliothek und der Funktion `pandas.read_csv("Name")`,
- 2. ohne Bibliothek, mit der `open`-Funktion  
`open("Name.datentyp", r)`  
öffnet eine Datei und kann dann Zeilenweise ausgelesen werden

```
import pandas as pd

dataset = pd.read_csv("sample_data.csv")
print(dataset)
```

```
daten_lesen = open("sample_data.csv")
for line in daten_lesen:
    print(line)
daten_lesen.close()
```

## 4. Schreiben in eine Datei

- mithilfe der open-Funktion
- `open("Name", w)`, `w= write`
- Daten werden in die angegebene Datei geschrieben
- Datei wird erzeugt oder überschrieben

```
1  daten_lesen = open("Text.txt")
2  daten_schreiben = open("Text2.py", "w")
3
4  for line in daten_lesen:
5      print(line.rstrip())
6      daten_schreiben.write(line)
7      print('\n')
8
9  daten_lesen.close()
10 daten_schreiben.close()
```



## 4. with

- vereinfacht das lesen und schreiben der Dateien
  - Konstrukt wird öfter in Python benutzt
  - schließt Datei nach Ablauf des Blocks oder bei Ausnahmen
- ➔ Datei wird immer geschlossen, close entfällt

```
1  with open("Text.txt") as daten_lesen:
2      with open("Text2.py", "w") as daten_schreiben:
3
4          for line in daten_lesen:
5              print(line.rstrip())
6              daten_schreiben.write(line)
7              print('\n')
```



## 5. Beispielprogramm vis.py

Run Cell | Run Below | Debug Cell

# %%

# Libraries einbinden

```
import pandas as pd
from matplotlib import pyplot as plt
```

Run Cell | Run Above | Debug Cell

# %%

# Daten einlesen

# Pfad muss korrekt sein

# sampledata = pd.read\_csv(r"VSCode\sample\_data.csv")

```
sampledata = pd.read_csv("sample_data.csv")
```

# Daten als Tabelle ausgeben

```
sampledata
```

Run Cell | Run Above | Debug Cell

# %%

# Nur Spalte C ausgeben

```
sampledata.column_c
```

Run Cell | Run Above | Debug Cell

# %%

# Graph erstellen Spalte b über a und c über a

```
plt.plot(sampledata.column_a, sampledata.column_b)
plt.plot(sampledata.column_a, sampledata.column_c)
plt.title("Diagramm")
plt.xlabel("x")
plt.ylabel("y")
plt.show
```

## 6. Quellen

- ykDojo: Data and Sample Code - Intro to Data Visualization with Python.  
<https://www.csdojo.io/data> [Stand 28.10.2020]
- Programming for Everybody (Getting Started with Python) Uni Michigan,  
Coursera Kurs, <https://de.coursera.org/learn/python> [Stand 28.10.2020]
- Bernd Klein: Python 3 Tutorial zuletzt geändert im Oktober 2016  
[https://www.python-kurs.eu/python3\\_kurs.php](https://www.python-kurs.eu/python3_kurs.php) [Stand 28.10.2020]
- Ancud IT-Beratung GmbH: Machine Learning mit Python  
<https://www.ancud.de/loesungen/data-science/python/> [Stand 28.10.2020]
- Bernd Klein (2018). EINFÜHRUNG IN PYTHON 3. FÜR EIN- UND UMSTEIGER  
(3. Auflage). München, Deutschland: HANSER