Notizen zum Programmieren mit Python

Bezeichnung von Variablen:

- Erste Zeichen muss ein Bodenstrich oder Buchstabe sein // Keine Umlaute/Sonderzeichen ausser _
- Namen dürfen keine Schlüsselwörter sein (and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, print, raise, return, True, try, with, while, yield)

Datentypen:

Float (Gleitkommazahlen):

- Alle Zahlen die einen Punkt enthalten, werden automatisch zum float (.2 , 2. , etc.)
- Division ergibt auch immer automatisch einen float (4/2 = 2 → float)
- Wissenschaftliche Schreibweise (10e4 → float)

String (str):

- Doppelte oder einfache Hochkommas (‘’’Mehrzeiligerstring’’’)
- a = """asd\ncasd""" → len = 8, **\n** zählt als ein Zeichen
- print("dsa" in "a") #bool Ausgabe: False
- f-String: f"YouthAccount **\t** from {self.name} **\t** born {self.geburtsdatum_konto_inhaber}"

- password = "passwort123"
- print(password.islower()) #isupper() sind alle Buchstaben klein? Ohne Zahlen! #bool: True
- password.isalpha() → alles Buchstaben? password.isdecimal() → alles Zahlen im String?
- password.isalnum() → alles Buchstaben und Zahlen, jedoch keine Sonderzeichen? Bool

- string = "Das ist ein Satz"
- print(string.startswith("Das")) #bool -> True // print(string.endswith("Haus")) #bool -> False

- print("Das ist\tein\nString!".split()) #Liste Ausgabe: ['Das', 'ist', 'ein', 'String!']
- print("Das-ist\tein-String!".split("-")) #Liste ['Das', 'ist\tein', 'String!']

- names = ["Hendrik", "Janek", "Torsten"] #geht nur bei Listen!!!
- a = (", ".join(names)) #print(a) → Ausgabe als ein String ‘Hendrik, Janek, Torsten’
- Er = " Er " \n print(Er.strip()) #rstrip / lstrip nur links oder rechts #Er

List: [veränderlich] → Beispiel names = [’Hendrik’, ‘Janek’, ‘Anna’, ‘Daniel’, 3, 5, 4.2]

- print(len(names)) → gibt die Anzahl der Elemente der Liste aus
- print(names[::3]) #3er Schritt Ausgabe: ['Hendrik', 'Daniel', 4.2]

- names.append(3) → hängt der Liste zu hinderst ein neues Element (hier die 3) an
- names.insert(2, "Ulf") → hängt ein weiteres Element an der x-ten Stelle der Liste an
- names.pop(1) → löscht das x-te Element aus der Liste (ohne Angabe von Index das letzte)
- del names[2] → geht nur mit Index, löscht nur dieses Element
- names.remove("Hendrik") → löscht Hendrik aus der Liste (aber nur einmal!!!)

- print(names.index("Anna"))→ zeigt den Index von Element ‘Anna’ der Liste #2
- names.extrend(names2) →zweite/r Liste/Tupel/String der Liste names hinzufügen
- names.reverse() → die Elemente der Liste wird umgekehrt →in-place!!! Originalveränderung
- names.sort() → sortiert die Liste alphabetisch/nummerisch #(reverse=True, key=len)

Dictionaries {Schlüssel : Wert}: phonebook = {"Huber": "1234", "Müller": "5678", "Heiri": "98", 1: "85"}

- keys_phonebook = phonebook.keys() #keys = 1. Element // values = 2. Element // items = Paare
- print(keys_phonebook) #Ausgabe: dict_keys(['Huber', 'Müller', 'Heiri', 1])
- for i in keys_phonebook: #Elemente mit einer Liste einzeln ausgeben **\n** print(i)

- print(phonebook.get("Huber", "kein Eintrag vorhanden")) #Einsichtname in Einträge, kEv=kEv! None
- del phonebook[1] #Löscht das Element mit Schlüssel = 1
- phonebook["Meier"] ="9999" #Eintrag hinzufügen

numpy.array() → import numpy **x[Zeile, Spalte]** **0:2 → exklusiv 2!!!**

- y = numpy.array([[1, 2], [3, 4]]) #2-D Array
- print(y > 2) # [[False False], [True True]]
- percentarry = (y/numpy.sum(y))
- x = numpy.array([[1,2,3],[4,5,6],[7,8,9]])
- erste Zeile: x[0, :] dritte Spalte: x[: , 2]
- zweite Spalte + erste beiden Zeilen x[0:2 , 1])

```
[[0.1 0.2]
 [0.3 0.4]]
```

```
def test_summe():
    if math.isclose(1, numpy.sum(percentarry),
                    abs_tol=eps):
        return "Es wurde korrekt gerechnet"
    else:    \n    return "Fehler in der Rechnung"
```

numpy.size(x, axis = 0) #0 = Zeilen / 1 = Spalten

numpy.zeros([numpy.size(my_array, axis = 0),numpy.size(my_array, axis = 1)],dtype=bool)

Import-Funktion

- import random **\n** random_number = random.randint(0, 5) #0 bis und mit 5!!!
- import calculation as calc #Modul einlesen und unter einem bestimmten Namen abspeichern
- from calculation import * #alle Module laden, diese müssen nicht mit name.funktion geladen werden!
- from calculation import faculty, addition, division #nur einzelne Module laden

---

- import random **\n** rps = ["rock", "paper", "sissors"]
- winner = random.choices(rps, k=2) #k = anzahl elemente, nur choice schreiben für 1 Element
- winner = random.sample(rps, k=2) #k = anzahl elemente, sample = keine doppelten Elemente!

Arbeiten mit Dateien

```
countries = {}
file = open("./example1.txt", "r")      ./ → aktuell        ../    1. Stufe +        ../../ 2. Stufe +
for line in file.readlines():
    line_splitted = line.split()
    countries[line_splitted[0]] = line_splitted[1]
print(countries)
file.close()
```

```
f = open('myfile.txt', 'a+') #fügt Datei am Schluss
f.write(hallo) #ein Hallo hinzu. Beim erneuten
f.seek(0) # Ausführen gibt es bereits zwei Hallo's
print(f.read()) # seek setzt Zeiger zurück, sonst
f.close() #print erst für nachfolgende leere Zeichen!
```

```
print(file.read()) #zeigt den Inhalt der ganzen Datei
print(file.readline()) #liesst eine Zeile/Zeile hüpft weiter bei mehreren Befehlen
print(file.readlines()) #generiert pro Zeile ein Element in einer Liste
```

Absolute Pfadangabe = ganzer Pfad angeben aus dem Wurzelverzeichnis C:/Users/Denis/…
Relativer Pfad = ../ vor Dateiname pro höhere Stufe/Ordner, geht in die andere Richtung als absolut

Datei-Modi

`w` = mit neuem Namen = neue Datei #bestehende Datei = löscht den Inhalt, fügt Befehle hinzu!
`a` = fügt am Ende der Datei die Codes hinzu, ohne den Rest zu löschen
`x` = bearbeitet nur Dateien, welche zuvor noch nicht existiert haben, keine Gefahr von Überschreibung
`w+` = lesen und schreiben (geht bei allen Modi)

```
file.seek(20) #Verschiebung Zeiger, Start ab bspw. Dem 20. Zeichen --> geht nur bei 'r'
print(file.tell()) #zeigt wo wir uns in der Datei befinden
print(file.readline())    \n        print(file.tell())
```

```
file.seek(20) #Verschiebung Zeiger, Start ab 20 --> geht nur bei 'r'
file.truncate() #geht nur mit 'r+', löscht alles nach dem 20. Zeichen (kann auch direkt angegeben werden)
```

CSV-Datei → import pandas

```
df = pandas.read_csv(Datei-Pfad "Datei.csv" oder "url", header = 0, #Welche Zeile gilt als Überschrift
    sep = ",", index_col=2) #Spalte welche als Index genommen wird (ohne Angabe wird nummeriert 0)
```

```
import csv
sniff = csv.Sniffer()
dialect = sniff.sniff(open("Nasdaq.csv").read())
print(dialect.__dict__)
```

```
f = open("Nasdaq.csv")
reader = csv.reader(f, dialect=dialect)
for row in reader:
    print(row)
```

JSON oder XLS-Datei → import pandas

```
df = pandas.read_json(Datei-Pfad "Datei.json" oder "url") / df = pandas.read_excel(…)
```

```
daten = [{"Planet": "Erde", "Umfang": 40075}, {"Planet": "Mars", "Umfang": 21338}]
```

```
import json
#Serialisierung = Schreiben
f = open("daten.json", "w")
json.dump(daten, f, ensure_ascii=False, indent=2)
f.close()
```

```
#Deserialisierung = Einlesen
f = open("daten.json")
neuedaten = json.load(f)
print("Daten:", neuedaten)
print("Daten als JSON:", json.dumps(neuedaten))
```

Pandas DataFrame → import pandas

Daten	Bananas	Apples	Peas	Oranges
01.05.2014	23	14	15	18
02.05.2014	25	16	18	21
03.05.2014	24	10	19	22

Speichert Daten in einen 2 dimensionalen DataFrame (Zeile / Spalte)

```
my_sales = {"Apples": [14, 16, 10], "Peas" : [15, 18, 19], "Oranges": [18, 21, 22],
            "Bananas": [23, 25, 24], "Dates": ['01.05.2014', '02.05.2014', '03.05.2014']}
```

```
df = pandas.DataFrame(my_sales) #, index=[1]
df = df.set_index(['Dates']) #Daten werden zum
df.index.name = "Daten"  #Index links aussen
df[["Bananas", "Oranges"]] #Ausgabe der Spalten
df[0:2] #Überschrift (Bananas, etc.) = 0-te Zeile
```

```
df.Oranges[0] #oder df['Oranges'][0]
df.Apples[df.Apples > 10]
df.index #Liste mit Index (hier Datum)
df.columns #Name der Spalten (hier Früchte)
df.values #Values (hier Zahlen der Matrix)
```

```
df["TOTAL"] = df[:,['APRIL','MARCH']].sum(axis=1) #Neue Spalte mit Summe aus zwei Spalten bilden
```

```
import pandas \n pandas.set_option("display.max_rows", None) #Ausgabe aller Zeilen
df = pandas.read_excel("Datei.xlsx", skiprows = 8, skipfooter = 7, names = "liste spaltennamen",
                       index_col = "Datum")
df = df.drop(columns=["L1", "L2"]) #Löscht die Spalten L1 und L2
dfjahr = df[df.index.month == 12] #filtern und neuer df erstellen
dfjahr["Inflation"] = dfjahr["Basics1939"].diff() #Differenz zur vorherigen oder n-ten Zeile bei Angabe (N)
```

urllib Modul → import urrlib / import urllib.request / import urllib.parse + import json

```
f = urllib.request.urlopen(url)
bancomaten = json.load(f)
```

urllib.parse.urlencode → for key-value pairs
urllib.parse.quote → for strings

Plots erstellen → import pandas import pylab

```
df = pandas.read_csv(url, sep=";")
df = df.set_index(["kanton_canton_cantone"])
df.index.name = "Beschäftigungskanton"
df[["2018"]].plot(kind="bar", title="Beschäftigte gesamt")
(df["2018"] - df["2013"]).plot(kind="bar", title="Entwickng")
pylab.show()
```

Objektorientierte Programmierung

Mit Hilfe des **Konstruktors** kann der Initialwert vorgegeben werden, bspw. = None

```
class Animal:
    counter = 0 #Klassenattribut  → Ausgabe via Instanziiertem Objekt owl1.counter / Animal.counter
    def __init__(self, name, age, color, food):
        self.name = name # self.name ist ein Attribut / name ist ein Parameter
        self.age = age  \n  self.color = color  \n  self.food = food
        Animal.counter += 1 #Mit der Instanziierung der Klasse wächst der Zähler um eines
    def sleep(self): #Eine Funktion in einer Klasse nennt man Methode
        print("Ich schlafe gerade ...")
```

```
class Owl(Animal): #Vererbung
    def __init__(self, name, age, color, food, hunt):
        super().__init__(name, age, color, food) #Alternative  Animal.__init__(self, name, age, color, food)
        self.hunt = hunt #weiteres Attribut hinzufügen
    def sleep(self):
        super().sleep() #Erbt die Eltern Klasse → print("Ich schlafe gerade ...") / Animal.sleep(self)
        print("Gleichgewicht halten ...")
```

```
owl1 = Owl("Eule", 2, "grau", "Mäuse", 10)
owl1.sleep() #Ausgabe: Ich schlafe gerade ...  \n  Gleichgewicht halten ...
print(owl1.hunt) # Ausgabe: 10
```

Attribut/Methode mit vorangestelltem einzelnen unterstrich sind nur für internen Gebrauch der Klasse gedacht. Dieses sollte nicht ausserhalb des Attributs/Klasse verwendet werden, ist jedoch möglich!

Das **Name Mangling**, mit zwei __ vor dem Attribut, verhindert den Zugriff auf das Attribut von ausserhalb der Klasse.

Statische Methode → @staticmethod

Man kann diese aufrufen, ohne vorher ein Objekt von der entsprechenden Klasse erzeugt zu haben

```
class Math:
    @staticmethod # braucht KEIN self
    def multiply(a, b): # Auf a und b kann ausserhalb der Methode nicht zugegriffen werden!
        return a * b #  \n  print(Math.multiply(2, 3)) #Ausgabe 6
```

Classmethod gehört zu Statische Methode, erster Parameter ist die Klasse als cls, Mehrfachvererbung!

```
import math
class Circle:
    counter = 0

    def __init__(self, radius):
        self.radius = radius
        Circle.counter += 1

    def area(self):
        return math.pi * self.radius ** 2

    @staticmethod
    def convert_area_in_radius(area):
        return math.sqrt(area/math.pi)

    @classmethod
    def from_area(cls, area):
        r = math.sqrt(area / math.pi)
        return cls(r)

    @classmethod
    def print_counter(cls):
        print(f"Anzahl erzeugter Objekte {cls.counter}" )

class Rim(Circle):
    pass

rim = Rim.from_area(20)
print(Rim.counter)
```

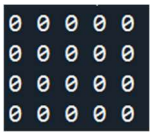
Abstract Methode

```
class Player(ABC):
    @abstractmethod
    def name(self, ) ABC als Elternklasse
```

Codeschnipsel

```
def badewetter(temperaturen):
    tage = ["Mo", "Di", "Mi" , "Do", "Fr", "Sa", "So"]
    max_temp = -1000
    max_index = -1
    for i in range(len(temperaturen)):
        if temperaturen[i] > max_temp:
            max_temp = temperaturen[i]
            max_index = i
    return tage[max_index]
print(badewetter([27,28,29,32,20,21,12]))
```

```
board = []
def print_board(board):
    for x in range(4):
        board.append(["0"]*5)
    for row in board:
        print((" ").join(row))
print_board(board)
```



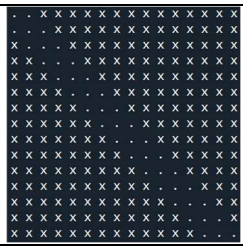
```
comp = random.randint(1, 10)    \n  user = 0
while comp != user:
    user = int(input("Enter number 1-10: "))
    if comp == user:
        print("You win!!!")
    elif comp > user:
        print("The number is bigger, try again")
    elif comp < user:
        print("The number is smaller, try again")
```

```
password = "Apfel1234"
def check_passwort(passwort):
    laenge = len(passwort)
    if laenge < 9 or laenge > 15:
        return False
    anzahl_ziffern = 0
    for i in range(len(passwort)):
        zeichen = passwort[i].upper()
        if zeichen >= '0' and zeichen <= '9':
            anzahl_ziffern += 1
        else:
            if zeichen < 'A' or zeichen > 'Z':
                return False #Sonderzeichen
    if anzahl_ziffern < 3:
        return False
    return True
print(check_passwort(password))  #True
```

```
nummern = [] \n namen = []  \n jahrgang = []
preise = []
wein_liste = wein_sortiment.split('$')
for wein in wein_liste:
    datensatz=wein.split(';')
    nummern.append(datensatz[0])
    namen.append(datensatz[1])
    jahrgang.append(datensatz[2])
    preise.append(datensatz[3])
print(wein_liste)
```

```
import re    \n    cc = "1009-2008-0703-0604"
if re.match("(\d{4}-){3}\d{4}", cc):
    print(True)
if len(cc) != 4 * 4 + 3:
    re.reject(cc)
if sum([int(x) for x in list(cc.replace("-", ""))]) % 10 == 0:
    print(True)
```

```
for i in range(15):
    for j in range(16):
        if i == j or i == j+1 or i == j-1:
            print(".", sep=" ", end=" ")
        else:
            print("x", sep=" ", end=" ")
    print("")
```



Code-Sanitisation

```
def register(name):
    replacements = {"ä": "ae", "ö": "oe", "ü": "ue"}
    for key, value in replacements.items():
        name = name.replace(key, value)
    print("Register", name)
register("Köbi Müller") #Ausgabe: Koebi Mueller
```

Exponential Waiting-Time → import random

```
class Wartezeit:
    def bereche_wartez(self, versuch_nummer: int):
        base_time = 0.5
        return base_time * random.choice
                    (range(2**versuch_nummer))
```

Quicksort - divide et impera - import random

```
def exchange(sortlist , i, j):
    temp = sortlist[i]    \n    sortlist[i] = sortlist[j]
    sortlist[j] = temp
def quicksort(values , start , end):
    if start < end:
        pivot = partition(values , start , end)
        quicksort(values , start , pivot - 1)
        quicksort(values , pivot + 1, end)
def partition(values , start , end):
    r = random.randint(start , end)
    pivot = values[r]    \n    exchange(values , r, end)
    i = start - 1
    for j in range(start , end):
        if values[j] <= pivot:
            i += 1      \n          exchange(values , i, j)
    exchange(values , i + 1, end)    \n    return i+1
liste = [9,8,33,7,2,5,13,18,6,1,7,4,5,9,10,11]
quicksort(liste, 0, len(liste)-1)  \n  print(liste)
```

Countsort

```
def count_sort(list , max):
    counted_list = [0 for x in range(max+1)]
    sorted_list = [0] * len(list)
    for i in range(len(list)):
        counted_list[list[i]] += 1
        counter = 0
        for j in range(max +1):
            val = counted_list[j]
            for i in range(0, val):
                sorted_list[counter] = j
                counter += 1
    return sorted_list
liste2 = [1,5,5,5,5,5,3,3,4,4,4,4,2,2,3]
print(count_sort(liste2, len(liste2)-1))
```

Comprehensions

Elemente werden direkt mit einer Schlaufe der Liste hinzugefügt.

Comprehensions

```
numbers = [i for i in range(1,51)]
print(numbers)
```

Alternative

```
numbers = []
for i in range(1,51):
    numbers.append(i)
print(numbers)
```

```
students2 = [("Student " + str(i), random.randint(1, 7)) for i in range(1,21)]
print(students2)
```

```
students = []
for i in range(1,21):
    students.append(("Student " + str(i), random.randint(1,7)))
print(students)
```

```
d = {"a":1, "b":2, "c":3}
d = { k.upper( ) : v for k, v in d.items()}
print(d) #{'A': 1, 'B': 2, 'C': 3}
```

```
a = list(range(8))
a =[x for x in a if x % 2 == 0]
print(a) #[0, 2, 4, 6]
```

```
a =[x+1 for x in a if x % 2 == 0]
print(a) #[1, 3, 5, 7]
```

Main

```
def main():
    if len(sys.argv) == 2 and sys.argv[1].lower() == 'test':
        pytest.main([sys.argv[0]])

if __name__ == "__main__":
    main()
```

Code-Schnippsel aussortiert

```
def greeting(first_name, last_name, academic_title = ""): #zuerst non-default Werte und
    if academic_title != "":                              #anschliessend nur noch Default-Werte
        academic_title += " "
    print("Hallo " + academic_title + first_name + " " + last_name)
    print("Herzlich Willkommen!")
greeting("Max", "Mustermann", "Dr.")
```

```
def multiply(number1, number2, *numbers):
    sum = number1*number2
    for i in numbers:
        sum *= i
    return sum
print(multiply(2,2,2,2))
```

```
user_input = ""
list = []
while user_input != "end":
    user_input = input("Enter your List with names: ")
    if user_input != "end":
        list.append(user_input)
    else:
        break
print(list)
```

Operatoren / Zeichen	Symbol	Beispiel
Exponenten	**	2**3 = 8
Modulo	%	13%7 = 6
concatenation	string1 + string2	'Hello ' + 'World' → 'Hello World'
repetition	string1 * n	'.' * 5 → '.....'
character at index (index is 0 based)	string1[5]	'abcdefg'[5] → 'f'
subtring with slicing	string1[start:stop]	'abcdefg'[2:5] → 'cde'
string length	len(string1)	len('Hello') → 5
find pattern	patternA in string1	'an' in 'apple,banana,kiwi' → True
split string	string1.split(delimiter)	'apple,banana,kiwi'.split(',') → ['apple', 'banana', 'kiwi']