Python Spickzettel

Grundsätzliches:

- Code wird über Einrückungen strukturiert; Code im gleichen Anweisungsblock gleich weit eingerückt
- Kommentare werden mit # eingeleitet
- ''' text ''' markiert einen Kommentarblock

Code importieren

```
import <Paket>                  Software aus Paket importieren
                                Aufruf von Funktion des Pakets
from <Paket> import <Modul>
                                Aufruf von Funktion des Pakets
import <Paket> as np            importieren mit Kurznamen
```

```
import numpy
numpy.sin(x)
from numpy import sin
sin(x)
np.sin(x)
```

Rechenoperationen

```
Zuweisung:          =                                   a = 3
Grundrechenarten:    +   -   *   /                      x + 3, a * b, …
Rest bei Division:  %                                   17 % 3
Potenzieren:        **                                  2 ** 3  → 8
```

Mathematische Funktionen (auch für Arrays)

```
                    import numpy
Konstanten:         numpy.pi, numpy.e, numpy.euler_gamma
Funktionen:         numpy.sqrt(x), numpy.exp(x), numpy.log(x), numpy.log10(x)
                    numpy.sin(x),  numpy.cos(x), numpy.tan(x)
                    numpy.sinh(x), numpy.cosh(x), …,
Betrag, Runden:     numpy.abs(x),  numpy.around(x)
Typumwandlung:      numpy.int_(x), numpy.float_(x)      (Ganze Zahl ↔ Fließkommazahl)
```

Datentypen

```
bool          Wahrheitswert          True, False
int           Ganze Zahl             23456
float         Fließkommazahl         3.1415
complex       Komplexe Zahl          1.+2.j
str           Zeichenkette (String)  'Hallo'
tuple         Tuple                  (1, 3, 5)          Tuples sind nicht veränderbar !
list          Liste                  [1, 3, 5]          Listen sind veränderbar: a[1] = 2
                                     range(5) → [0, 1, 2, 3, 4]
dictionary    Key-Value-Paare        {zahl:5, text:'Hi'}
numpy.array   numpy-Array            numpy.array( [1., 3., 5.] )
      erlaubt Vektoroperationen:     x = numpy.array([1, 2, 3])   x ** 2       → array([1, 4, 9])
                                     a = numpy.array([1, 4, 9])   numpy.sqrt(a) → array([1, 2, 3])
       a = numpy.array([1, 2, 3])    b = numpy.array([2, 3, 4])   a * b        → array([2, 6, 12])
```

Listen und Felder (Arrays)

```
l = [2, 4, 6, 8, 10]              |   len(l)                  Länge der Liste
l[0]        das erste Element     |   l.append(<Element>)     Element anhängen
l[-1]       das letzte Element    |   l.insert(<index>,<Element>)   einsetzen
l[2:4]      Liste der Elemente 2 bis 3  |   l.remove(<Element>)     Element entfernen
range(5,8)   → [5, 6, 7]          |   l.index(<Element>)      Index von Element
range(0,6,2) → [0, 2, 4]          |   l.sort()                Liste sortieren
```

```
import numpy as np
np.zeros(5)             → array([0., 0., 0., 0., 0., 0.])
np.ones(3)              → array([1., 1., 1.])
np.linspace(0., 2., 5)  → array(0., 0.5, 1., 1.5, 2.)
np.arange(0., 2., 0.5)  → array(0., 0.5, 1., 1.5)
```

Kontrollstrukturen

Schleifen:

```
for variable in range(n):
    <Anweisungsblock>

for variable in <Liste>:
    <Anweisungsblock>

while <Bedingung>:
    <Anweisungsblock>
```

Eine Schleife kann mit `break` abgebrochen werden

Der Anweisungsbllock kann mit `continue` übersprungen werden

Verzweigungen:

```
if <Bedingung>:
    <Anweisungsblock>

if <Bedingung>:
    <Anweisungsblock>
else:
    <Anweisungsblock>

if <Bedingung1>:
    <Anweisungsblock>
elif <Bedingung2>:
    <Anweisungsblock>
else:
    <Anweisungsblock>
```

Verknüpfung von Bedingungen:

```
if x < 7 and x != 5
if x == 2 or x == 4
```

Funktionen

Definition:

```
def name(Parameter):
  Anweisungsblock
```

Aufruf:

```
name(Parameter)
```

Beispiel:

```
def maximum(x,y):
    if x >= y:
        return x
    else:
        return y
```

Funktionen können beliebig viele Parameter haben. Klammern sind immer notwendig, auch wenn keine Parameter vorhanden sind

`return` beendet eine Funktion und gibt einen Wert zurück. Eine Funktion muss keinen Rückgabewert haben.

Variablen im aufrufenden Programm sind in Funktionen nur lesbar. Wenn eine Variable verändert werden soll, muss sie mit `global <Variable>` in der Funktion definiert werden.

Ein- und Ausgabe

```
input([prompt])
print(x)
```

```
x = input("Welche Zahl?")
print("Die Antwort ist: ", 6*7)
```

Zufallszahlen und Statistik

Im Modul `numpy.random` gibt es Funktionen zum Erzeugen von Zufallszahlen

```
import numpy as np
np.random.random()              Zufallszahl zwischen 0. und 1.
np.random.random(10)            Array mit 10 Zufallszahlen zwischen 0. und 1.
np.random.normal()              gaußverteilte Zufallszahl mit Mittelwert 0. und Streuung 1.
np.random.normal(1., 2., 10)    Array mit 10 gaußverteilten Zufallszahlen  Mittelwert 1, Streuung 2

data = np.random.rand(100)
np.sum(data)                    Berechnet Summe der Elemente von data
np.std(data)                    Berechnet Standardabweichung der Elemente
bc, be = np.histogram(data, bins)   Berechnet Histogramm für bins Intervalle:
                                bc: Häufigkeiten,  be: Intervallgrenzen
```

Visualisierung

Das Modul `matplotlib.pyplot` enthält viele grafische Funktionen

```
import matplotlib.pyplot as plt
```

`plt.plot(x, y)`	Array y gegen Array x auftragen
`plt.errorbar(x,y, yerr=<val>, fmt='ro')`	Datenpunkte (x, y) mit Fehlerbalken
`plt.bar(x, y, align='center')`	Balkendiagramm
`bc, be, p = plt.hist(data, bins)`	Histogramm (`bc`: Häufigkeiten, `be`: Intervallgrenzen)
`plt.show()`	Grafik auf Bildschirm anzeigen
`plt.savefig(<Dateiname>)`	Grafik in Datei abspeichern

Beispiel:

```
import numpy as np
import maplotlib.pyplot as plt
x = linspace(0.001, 10., 200)
y = np.sin(np.pi * x / x)
plt.plot(x, y)
plt.show()
```

Beschriftungen:

`plt.xlabel(Text)`	Beschriftung der x-Achse
`plt.ylabel(Text)`	Beschriftung der y-Achse
`plt.title(Text)`	Titel des Graphen
`plt.colorbar()`	Schaltet Farbbalken ein
`plt.legend(Position)`	Schaltet Legende ein, `(Position: 'best', 'upper right', 'lower left', etc.)`

komplexere Darstellung:

```
fig = plt.figure(<Name>, figsize=(x,y) ) , ax = fig.add_subplot(c,r,n), ax.plot(x,y, label=<txt>),
       ax.set_title(<name>), ax.set_xlabel(), ax.set_ylabel(), ax.set_xlim(),ax,set_ylim(),
       ax.legend(), ax.grid(True), ax.text(x,y, <text>)
```