Python Cheat-Sheet

`# Kommentar`	Der Hashtag leitet eine **Kommentarzeile** ein. Diese Zeile wird nicht ausgeführt!
`:`	Leitet einen **Anweisungsblock** ein: Alle Zeilen, die danach eingerückt sind, gehören zum Anweisungsblock, z.B. zu der Funktion oder Prozedur.

`a = 42`
`print(a)`
`# erhoeht a um 2`
`a = a+2`
`print(a)`
	Variablen legt man einfach fest, indem man ihnen einen Wert zuweist. Variablennamen sind beliebig, müssen aber mit einem Buchstaben anfangen.
	Bei Zuweisungen ist immer der Variablenname links und der (neue) Wert rechts.

`a = 42` `print("DIE Antwort",a)`	Schlüsselwort für die **Ausgabe an die Konsole**. Was ausgegeben wird, steht in der Klammer. Mehrere Ausgaben können durch Komma getrennt werden.
Funktionen / Prozeduren	

`def addieren(a,b):`
`  ergebnis = a+b`
`  return ergebnis`
	Schlüsselwort, um eine **Funktion** oder **Prozedur** zu definieren:
	- Eine **Funktion** berechnet etwas; das Ergebnis wird mit `return` zurückgegeben.
	- Eine **Prozedur** tut etwas (z.B. verändert Variablen). Es gibt kein `return`.
	- Prozeduren und Funktionen können **Parameter** (einer oder mehrere) mitgegeben werden.
	Die Parameter stehen in Klammern hinter dem Namen der Prozedur bzw. Funktion.
	Innerhalb der Prozedur bzw. Funktion kann mit dem Parameter gearbeitet werden.
	Funktionen/Prozeduren ohne Parameter: leere Klammern.

`ergebnis = addieren(3,5)` `print(ergebnis)`	Funktionen und Prozeduren kann man wie rechts **aufrufen**. Man muss die richtige Zahl von Parametern angeben. Bei Funktionen muss man sich für den Rückgabewert interessieren! Der Rückgabewert wird hier in der Variable `ergebnis` gespeichert.
Bedingungen	

`x = 3`
`y = 5`
`if x == y:`
`  print("gleich!")`
`else:`
`  print("ungleich!")`
	Bedingung. Wenn die Bedingung erfüllt ist, dann wird das ausgeführt, was in dem Anweisungsblock nach dem Doppelpunkt steht.
	`else` bedeutet "sonst".

Schleifen

```
for i in range(1,10):
   zahl = i*2
   print(zahl)
```

for ist das Schlüsselwort für eine **Zähl-Schleife:** Der folgende Anweisungsblock wird wiederholt ausgeführt.
Die Zähl-Schleife hat eine Zähl-Variable, in diesem Fall i. Diese wird bei jedem Durchlauf eins hochgezählt.
Mit **`in range(..,..)`** gibt man den Bereich an.
Die erste Zahl ist einschließlich, die zweite Zahl ist ausschließlich!

```
zahl = 2
while zahl < 1000:
   # Zahl verdoppeln
   zahl = zahl * 2
   print(zahl)
```

while ist das Schlüsselwort für eine **bedingte Schleife**:
Die Schleife wird so lange ausgeführt, wie die Bedingung erfüllt ist.
In der Schleife muss man dafür sorgen, dass die Bedingung irgendwann mal erfüllt werden kann!
Sonst hat man eine sogenannte *Endlosschleife*.

Listen

```
liste = [2, 5, -3, 8]
print(liste)
```

Listen werden mit **eckigen Klammern** definiert.

```
liste = [2, 5, -3, 8]
a = max(liste)
b = min(liste)
c = len(liste)
print(a,b,c)
# gibt 8 -3 4 aus
```

Es gibt vorgefertigte Funktionen für Liste, die
Maximum
Minimum
Länge
ermitteln.

```
liste = [2, 5, -3, 8]
print(liste)
a = 7
if a in liste:
   print(a, "dabei")
else:
   print(a, "nicht dabei")
```

Mit **in** kann man überprüfen, ob ein Wert in einer Liste enthalten ist.

```
liste = [2, 5, -3, 5, 8]
a = 7
liste.append(a)
b = 5
liste.remove(b)
print(liste)
```

Mit `append` und `remove` kann man Elemente zur Liste **hinzufügen** bzw. **löschen**.
`append` fügt hinten hinzu.
`remove` löscht nur das erste Vorkommen des Elements!
Aufgerufen werden die Funktionen mit
Listenname **Punkt** `append` (bzw. `remove`).

```
# index  0  1  2   4
liste = [2, 5, -3, 8]
position = liste.index(-3)
print(position)
```

Mit dem Schlüsselwort `index` kann man den **Index** (= die Position) eines Elementes in der Liste herausfinden.
Der Index wird immer von 0 an gezählt.

```
# index  0  1  2   4
liste = [2, 5, -3, 8]
liste.insert(2,42)
print(liste)
# [2, 5, 42, -3, 8]
```

Mit `insert` kann man an einer gewünschten Position ein Element in eine Liste einfügen.
Als Parameter wird erst der Index angegeben
und dann der Wert, der eingefügt werden soll.