Cheatography

Python Cheat Sheet

by ProjektSB19 via cheatography.com/81589/cs/19543/

Variablen und Strings

Text ausgeben

```
print("Hello world!")
```

>> Hello world!

Text mit einer Variablen ausgeben

```
msg = "Hello world!"
print(msg)
```

>> Hello world!

Verkettung (Strings kombinieren)

```
first_name = 'albert'
last_name = 'einstein'
full_name = first_name + ' ' + last_name
print(full_name)
```

>> albert einstein

Variablen werden zum Speichern von Zahlenwerten, Zeichen oder Strings verwendet. Ein String ist eine Reihe von Zeichen, die von einfachen oder doppelten Anführungszeichen umgeben sind.

Datentypen

Numerische Typen

`int`, `long`	für ganze Zahlen (long ist für besonders große Zahlen)
`float`	für Kommazahlen

Boolsche Werte

`bool`	Wahrheitswerte (true, false)

Arithmetische Operationen

`x+y`	Addition
`x-y`	Subtraktion
`x*y`	Multiplikation
`x/y`	Division
`x%y`	Rest bei ganzahliger Division
`x**y`	Potenz

Schleifen

Eine einfache While-Schleife

```
current_value = 1
while current_value <= 5:
    print(current_value)
    current_value += 1
```

Den Benutzer entscheiden lassen, wann die Schleife beendet werden soll

```
msg = ''
while msg != 'quit':
    msg = input("Wie ist deine Nachricht? ")
```

<< Hallo Welt!

```
  print(msg)
```

>> Hallo Welt!

Eine while-Schleife wiederholt einen Codeblock, solange wie eine bestimmte Bedingung erfüllt ist.

Tupel

Tupel erstellen

```
dimensionen = (1920, 1080)
```

Tupel ähneln Listen, aber die Elemente in einem Tupel können nicht geändert werden.

Listen

Liste erstellen

Strings gehören in einfache Anführungszeichen

```
liste = [elem0, elem1, elem2]
```

erstes und letztes Element erhalten

```
erstesElem = liste[0]
letztesElem = liste[-1]
```

Elemente zu einer Liste hinzufügen

```
liste = []
liste.append(elem1)
liste.append(elem2)
liste.append(elem3)
```

Listen

Durchgehen einer Liste

```
for elem in liste:
print(elem)
```

je nachdem, was `elem` für einen Typ hat, wird eine Zahl, ein Zeichen oder ein String ausgegeben

>>4

>>a

>>Hallo

>>3,46

Numerische Listen erstellen

```
quadrate = []
for x in range(1, 11):
  quadrate.append(x**2)
```

Listen

Abkürzung

```
quadrate = [x**2 for x in range(1, 11)]
```

Listen teilen

```
finalisten = ['sam', 'bob', 'ada', 'bea']
erstenBeiden = finalisten[:2]
```

Listen kopieren

```
kopieVonListe = liste[:]
```

Eine Liste speichert eine Reihe von Elementen (Zahlen, Zeichen, Strings, Objekte, ...) in einer bestimmten Reihenfolge. Du greifst auf Elemente über einen Index oder innerhalb einer Schleife zu.

Benutzereingabe

Aufforderung zur Eingabe

```
name = input("Wie ist dein Name? ")
print("Hallo, " + name + "!")
```

<< Steve

Wenn auf dem Bildschirm die Frage "Wie ist dein Name? " steht, ist man aufgefordert eine Eingabe zu tätigen. Diese wird in der Variablen **name** gespeichert.

>> Hallo, Steve!

Benutzereingabe

Aufforderung zur Zahleneingabe

```
alter = input("Wie alt bist du? ")
```

<< 16

```
alter = int(alter)
```

Hier findet eine Typumformung statt, da jede Eingabe zunächst als String gespeichert wird.

```
pi = input("Was ist der Wert von Pi? ")
```

<< 3,141592654

```
pi = float(pi)
```

Deine Programme können den Benutzer zur Eingabe auffordern. Alle Eingaben werden als String gespeichert.

Funktionen

Eine einfache Funktion

```
def begruesse_benutzer():
```

Zeigt eine Begrüssung.

```
    print("Hallo!")

begruesse_benutzer()
```

>> Hallo!

Argumente übergeben

```
def greet_user(username):
```

Zeigt eine persoenliche Begruessung an.

```
    print("Hallo, " + username + " ;D .")

greet_user('SummerBYTE')
```

>> Hallo, SummerBYTE ;D .

Standardwerte für Parameter

Funktionen (cont)

```
def make_pizza(topping='Bacon'):
```

Erstellt eine einfach belegte Pizza

```
    print("Du bekommst eine " + topping + " Pizza.")

make_pizza()
make_pizza('Pepperoni')
```

>> Du bekommst eine Bacon Pizza.

>> Du bekommst eine Pepperoni Pizza.

Rückgabe eines Wertes

```
def summiere(x, y):
```

Gibt die Summe von 2 Zahlen zurück

```
    return x + y

sum = summiere(3, 5)
print(sum)
```

>> 8

Funktionen sind gekennzeichnete Codeblöcke, die für eine bestimmte Aufgabe konzipiert sind. Informationen, die an eine Funktion übergeben werden, werden als Argument bezeichnet, und Informationen, die von einer Funktion empfangen werden, werden als Parameter bezeichnet.

Verzweigung

Vergleichsbedingungen

gleich	`x == 42`
ungleich	`x != 42`
größer als	`x > 42`
größer gleich	`x >= 42`
kleiner als	`x < 42`
kleiner gleich	`x <= 42`

Bedingungen mit Listen

```
elem1 in liste
elem3 not in liste
```

Boolesche Werte zuweisen

```
aktiv = True
pausiert = False
```

Einfache if-Verzweigung

```
if alter >= 18:
print("Du darfst wählen!")
```

>> Du darfst wählen!

if-elif-else-Verzweigungen

nur ein Codeblock wird ausgeführt

(Reihenfolge beachten!)

Verzweigung (cont)

```
if alter < 4:
  ticketPreis = 0
elif alter < 18:
  ticketPreis = 10
else:
  ticketPreis = 15
```

Verzweigungen werden verwendet, um bestimmte Bedingungen zu testen und entsprechend zu reagieren.

Klassen

Erstellen einer Klasse "Hund"

```
class Dog():
```

Stellt einen Hund dar.

```
    def __init__(self,
name):
```

Initialisiert eine Instanz der Klasse Hund.

```
        self.name = name

    def sit(self):
```

Simuliert sitzten.

```
         print(self.name
+ " sitzt.")

my_dog = Dog('Fluffy')

print(my_dog.name + "
ist ein braver Junge.")
```

>> Fluffy ist ein braver Junge.

```
my_dog.sit()
```

>> Fluffy sitzt.

Klassen

Vererbung

```
class SARDog(Dog):
```

Stelle einen Rettungshund dar.

```
     def __init__(self,
name):
```

Initialisert die Instanz des Rettungshundes.

```
super().__init__(name)

     def search(self):
```

Simuliert die Suche.

```
         print(sel-
f.name + " sucht.")

my_dog = SARDog('Fang')

print(my_dog.name + "
ist ein Rettungshund.")
my_dog.sit()
```

>> Fang ist ein Rettungshund.

```
my_dog.search()
```

>> Fang sucht.

Wörterbücher

Einfaches Wörterbuch

```
alien = {'Farbe':
'grün', 'Punkte': 5}
```

Zugriff auf einen Wert

```
print("Die Farbe des
Aliens ist " +
alien['Farbe'])
```

>> Die Farbe des Aliens ist grün

Wörterbücher (cont)

neues Schlüssel-Wert-Paar einfügen

```
alien['x_position'] = 0
```

Durchlaufen aller Schlüssel-Wert-Paare

```
lieblingszahl = {'erik':
17, 'julia': 4}
for name, nummer in
lieblingszahl.items():
  print(name + ' liebt
die ' + str(number))
```

>> erik liebt die 17

>> julia liebt die 4

Durchlaufen aller Schlüssel

```
lieblingszahl = {'erik':
17, 'julia': 4}
for name in lieblings-
zahl.keys():
  print(name + ' liebt
eine Zahl')
```

>> erik liebt eine Zahl

>> julia liebt eine Zahl

Durchlaufen aller Werte

Wörterbücher (cont)

```
lieblingszahl = {'erik':
17, 'julia': 4}
for nummer in lieblings-
zahl.values():
 print(str(nummer) + '
ist ein Liebling')
```

>> 17 ist ein Liebling

>> 4 ist ein Liebling

Wörterbücher speichern Verbindungen zwischen Informationen. Jedes Element in einem Wörterbuch ist ein Schlüssel-wertpaar.