# Python Cheat Sheet Cheat Sheet

by python_cheat_sheet via cheatography.com/165448/cs/34643/

## while-Schleifen

```
i = input("Zum Beenden 0 eingeben: ")
while i != 0:
        i = input( "Zum Beenden 0 eingeben: ")
```

## Vererbung

```
class Banana(Fruit)
```

## continue & break

**break**

*beendet Schleife*

**continue**

*springt zum Schlei fen anfang und fährt mit der nächsten Iteration fort*

**break und else**

```
for value in values:
        if value == 5:
                pri nt( " Gef und en! ")
                break
else:
        # wird Ausgef ührt, wenn Schleife nicht mit break beendet wird
         pri nt( " Nicht gefunden :-(")
```

## For loops

```
# Strategy: Iterate over a copy
for user, status in users.c op y().it ems():
        if status == 'inact ive':
                del users[ user]
# Strategy: Create a new collection
active _users = {}
for user, status in users.i te ms():
        if status == 'active':
                act ive _us ers [user] = status
```

## Zahlen

| | |
|---|---|
| 3 | eine ganze Zahl (integer oder int) |
| 3.2 | eine Gleitpunktzahl (float) |
| 3 + 5j | eine komplexe Zahl |
| +,-,*,/ | Grundrechenarten |
| a ** b | Potenz |
| math.sqrt(a) | Wurzel |
| math.log(a, base) | Logarithmus |
| a // b | ganzzahlige Division |
| a % b | Rest der Division |
| abs(a) | Betrag |
| round(a, n) | Runden auf n Nachkommastellen |

Im math Modul sind viele weitere Funktionen

## Listen

.add()

## if...else

```
if n < 2:
        # Block wird ausgef ührt, falls n < 2
         pri nt( "Fall 1")
elif n < 4:
        # Block wird ausgef ührt, falls 2 <= n < 4
         pri nt( "Fall 2")
else:
        # Block wird ausgef ührt, falls n >= 4
         pri nt( "Fall 3")
```

## Kommentare

```
# dies ist ein Kommentar
```

## Variable zuweisen

```
x = 2
```

# Python Cheat Sheet Cheat Sheet

by python_cheat_sheet via cheatography.com/165448/cs/34643/

## Iteration

range()

```
range
```

## Match

## Escape Characters

| | |
|---|---|
| \n | neue Zeile |
| \" | " |
| \' | ' |
| \\ | \ |

## Funktionen

```
def foo():
  return s
```

## Klassen

```
class Robot:
       def __init __( name, color, weight):
                sel f._name = name
                sel f._ color = color
                sel f._ weight = weight
       def introd uce _se lf( self):
                pri nt( "Ich heiße " + self.name)
r1 = Robot( " Tom ", red, 30)
r2 = Robot( " Ter ry", blue, 40)
```

## Datenkapselung

```
Attribute einer Klasse sind normalerweise protected
getter/setter verwenden
# Getter Methode als Property
@property
def x(self):
        return self._x
# Setter Methode als Property
@x.setter
def x(self, value):
       if value < 1000:
                self._x = value
```

## Strings

| | |
|---|---|
| 'Hallo' oder "Hallo" | eine Zeichenkette/string |
| 'Blau' * 3 | BlauBlauBlau |
| 'Blau' + 'beere' | Blaubeere |
| len(a) | Länge von string a |
| int('123') | casten zu int |
| float('123') | casten zu float |
| str('123') | casten zu string |
| print("""\ Usage: thingy [OPTIONS] -h Display this usage message -H hostname Hostname to connect to """) | Multiline String |

text = ('Put several strings within parentheses ' 'to have them joined together.')

Formatieren mit f
String methods
print('The value of i is', i)
print(a, end=',')

## Wahrheitswerte/Boolean

| | |
|---|---|
| True / False | bool |
| a > b | True wenn a kleiner als b, sonst False |
| a <= b | kleiner gleich |
| a > b | größer |
| a >= b | größer gleich |
| a == b | gleich |
| a != b | ungleich |
| not | negiert Wahrheitswert |
| or | Logisches Oder |
| and | Logisches Und |

a in sequenze