**Datentypen**

int (1,3,6,0,-3)

float (3.5, 123.23)

complex (2 + 3j)

Boolean (True, False)

Strings (“Hello”, “Hello, World!”)

Lists ([1,3,5,4,7])

type(Variable) -> checkt Datentyp

**Operatoren für int, float, complex**

x+y -> Summe von x&y

x-y -> Differenz von x&y

x\*y -> Multiplikation von x&y

x\*\*y -> x^y (kann auch als pow(x,y) geschrieben werden)

x/y -> Produkt von x&y

x//y -> integer division von x&y (Die Zahl wird immer abgerundet, wenn das Ergebnis kein Integer ist)

x%y -> Rest von x/y

-x -> x negativiert

+x -> x unverändert

int(x) -> x wird in Integer umgewandelt (Bsp: int(input()))

long(x) -> x wird in long integer umgewandelt (Bsp: long(input()))

float(x) -> x wird in float umgewandelt (Bsp: float(input()))

complex(re,im) -> Komplexe Zahl mit realem Teil *re* und imaginärem Teil *im* (Bsp: complex(x,y))

**Boolean Operators**

x and y -> Beide Bedingungen müssen True sein

x or y -> Eine von beiden Bedingungen muss True sein, kann auch sein dass beide True sind

not x -> Wenn x falsch -> True, sonst False

**Comparison Operators**

x==y -> True wenn x&y gleich, sonst False

x!=y -> True wenn x&y nicht gleich, sonst False

x>y -> True wenn x>y

x<y -> True wenn x<y

x>=y -> True wenn x>=y

x<=y -> True wenn x<=y

if condition: ->Testet ob Bedingung erfüllt

elif condition: -> Testet ob Bedingung erfüllt wenn vorherige if & elif False waren

else: -> Wird ausgeführt wenn alle vorherigen if & elif False waren

**Loops**

While condition: -> Wenn die Bedingung True ist, wird der Block ausgeführt. Am Ende des Blockes wird die Bedingung wieder getestet. Wenn wieder True -> Block wird nochmals ausgeführt, sonst weiter. Loop kann beendet werden mit *break*.

for x in range(y, z): -> Block wird für jede Zahl in range ausgeführt

for x in y: -> Wenn y eine Liste ist wird für jedes Element der Block einmal ausgeführt

**Accessing characters in a string**

word=”leopard”

word[0] -> l

word[1] -> e

word[-1] -> d

Bei positivem Zählen wird bei 0 begonnen, bei negativem bei -1.

word[0:3] -> leo

word[:3] -> leo

word[5:7] -> rd

word[5:] -> rd

word[::-1] -> drapoel

x+y -> Zwei Strings werden aneinander gehängt

x\*n -> wenn n eine Zahl -> x wird n mal aneinandergehängt

len(word) -> Zählt die Anzahl Characters in einem String

**Allgemeine Befehle**

import time -> um time-Befehle benutzen zu können, muss dieser Befehl ausgeführt werden

time.time() -> gibt die Zeit in Sekunden an