Cheatography

EF Informatik (MOU) Cheat Sheet

by Jean-Pierre Mouret (jpmou) via cheatography.com/163229/cs/34189/

Datentypen

Numerisch		
int	ganze Zahlen	-3214 , 42
float	Gleitkommazahlen	3.141 , -5.76e-2 , 2E02
bool	boolesche Werte	True , False
complex	komplexe Zahlen	3.4 + 4e2j , 3J + 7
Datenstrukturen		
string	Zeichenkette	“grün” , ‘Blau’
list	allgemeine Liste (Stack)	[value, …]
tuple	**unveränderbare** Liste	(value, …)[1]
set	ungeordnete und **nicht** indizierte Liste	{value, …}
dictionary	Liste mit **key** als Index	{key:value, …}[2]

[1] Klammern optional
[2] key muss **eindeutig** sein

Typunwandlung

int(a)	a → **int**	x = int(3.0) , y = int("4")
float(a)	a → **float**	x = float(2) , y = float(" -42")
str(a)	a → **string**	x = str(2) , y = str(4.2)
set(a)	a → **set**	x = set([1 ,2,3]) , y = set()
list(a)	a → **list**	x = list(('a', 'b', 'c'))

Die Funktion **input()** gibt immer einen **String** zurück.
Soll die Eingabe z.B. einem int zugewiesen werden:
x = int(in put())

Arithmetische Operatoren

x + y	Addition	x - y	Subtraktion
x * y	Multiplikation	x / y	Division
x // y	Division ohne Rest	x % y	Modulo
x ** y	Potenz x^y		

Zuweisungskurzbefehle: **x Operator= y**
Bsp: **x += 5** entspricht **x = x + 5**

Vergleichsoperatoren

x < y	kleiner	x <= y	kleiner oder gleich
x > y	grösser	x >= y	grösser oder gleich
x == y	gleich	x != y	ungleich

Logische Operatoren

x and y	x **und** y wahr	(3 > 2) and (2 < 4) # False
x or y	x **oder** y wahr	(3 > 2) or (2 < 4) # True
not x	x **nicht** wahr	(3 > 2) # False

Mitgliedsoperationen

x in y	x Element von y	5 in (1,2,3 ,4,5) # True
x not in y	x kein Element von y	5 not in (1,2,3 ,4,5) # False

Bedingte Ausführung

if *Bedingung*: *Anweis ung(en)*	Anweisungen werden ausgeführt, falls **Bedingung** wahr und if-Konstrukt wird verlassen
elif *Bedingung*: *Anweis ung(en)*	(optional, mehrere elif-Blöcke möglich) Anweisungen werden ausgeführt, falls **Bedingung** wahr und if-Konstrukt wird verlassen
else: *Anweis ung(en)*	(optional) Anweisungen werden ausgeführt, falls keine der Bedingungen wahr waren

Schleifen

```
while Bedingung:
    Anweis ung(en)
```

```
for i in range(start, end [,s tep]):
    Anweis ung(en)
```

```
for index in Datens truktur
    Anweis ung(en)
```

```
for key in sorted (di cto nary):
    Anweis ung(en)
```

[1] Schleife geht von **start** bis **end-1**
break beendet die Schleife, **continue** den aktuellen Durchlauf

Cheatography

EF Informatik (MOU) Cheat Sheet

by Jean-Pierre Mouret (jpmou) via cheatography.com/163229/cs/34189/

Indizes und Teilsequenzen (string, list, tuple)

len(a)	6		
a[0]	0	a[5]	5
a[-1]	5	a[-2]	4
a[1:]	[1,2,3,4,5]	a[:5]	[0,1,2,3,4]
a[:-2]	[0,1,2,3]	a[1:3]	[1,2]
a[1:-1]	[1,2,3,4]	b = a[:]	klonen

Indizes und Teilsequenzen für a = [0,1,2,3,4,5]

String-, List-, Tuple-Operationen

`len(s)`	Länge von s
`sorted(s)`	Sortierte Kopie von s

String-Operationen

`str.lo wer()`	umwandelin in Kleinbuchstaben
`str.up per()`	umwandeln in Grossbuchstaben
`str.re pla ce(alt, neu)`	*alt* durch *neu* ersetzen
`str.sp lit()`	erzeugt Liste aus den Sub-Strings

List-Operationen

`lst.ap pend(e)`	hängt e an
`lst.in sert(i, e)`	fügt e for 1-tem Element ein
`lst.pop(i)`	entfernt i-tes Element
`lst.re move(e)`	entfernt 1. Element mit Wert e
`del lst[i]`	entfernt i-tes Element