Generell

Die numerischen Datentypen sind gleichermaßen zu behandeln wie in den bekannten Programmiersprachen.

Datentypen

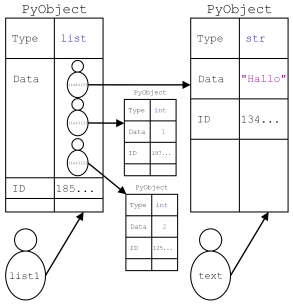
Datentyp	Beschreibung	False-Wert
NoneType	Indikator für nichts	None
Numerische Datentypen		
int	Ganze Zahlen	0
float	Gleitkommazahlen	0.0
bool	Boolesche Werte	False
complex	Komplexe Zahlen	0 + 0j
Sequenzielle Datentypen		
str	Zeichenketten oder Strings(unveränderlich)	' '
list	Listen(veränderlich)	[]
tuple	Tupel(unveränderlich)	()
bytes	Sequenz von Bytes(unveränderlich)	b' '
bytesarray	Sequenz von Bytes(veränderlich)	bytearray(b' ')
Mengen		
set	Einmalig vorkommende Objekte	set()
frozenset	Wie set jedoch unveränderlich	frozenset()
Assoziative Datentypen		
dict	Dictionary(veränderlich)	

Operatoren

Operator	Beschreibung
x // y	Ganzzahliger Quotient
x ** y	Potenzieren, x^y
+,-,...	Übliche Operation

Variablen

```
list1 = [text, 1, 2]
```



Casting

Datentyp	Klasse	Direkt	Datentyp	Klasse	Direkt
Ganzzahl	int()	3	Gleitkommazahl	float()	3.1415
Boolescher Wert	bool()	True,False	Komplexe Zahl	complex()	2 + 4j
String	str()	"HSR",'OST'	Liste	list()	[1,2,3]
Menge	set()	1,2,3	Tupel	tuple()	(1,2,3)
Unver. Menge	frozenset()	frozenset(1,2,3)	Dictionary	dict()	, "Key": 1

Eingabe und Ausgabe

```
name = input("type your name:")
print("Hello", name)
strList = ["YES", "WE", "CAN"]
for w in strList:
    print(w, end="--")
```

In:
type your name:MFG GG OG
Out:
Hello MFG GG OG
YES–WE–CAN–

Parameter	Beschreibung	Default
object(s)	Alle Objekte werden in String konvertiert	
sep='seperator'	Separierung der Objekte	' '
end='end'	Letztes Zeichen des print-Befehl	'\n'
file	Objekt mit einer Write-Methode	sys.stdout
flush	Boolscher Wert für die Output-Überprüfung	False

Listen

list.append(x)	list.extend(iterable)	list.remove(x)
list.clear()	list.index(x[, start[, end]]) → int	list.reverse()
list.copy() → list	list.insert(i, x)	list.sort(key=None, reverse=False)
list.count(x) → int	list.pop([i]) → object	

Erzeugen von Listen

```
mylist = [1, 2, 3]
alist = list([4, 5, 6])
blist = alist.copy()  # deep copy
print(f'mylist = {mylist}, alist = {alist}')
print(f'blist = {blist}')
```

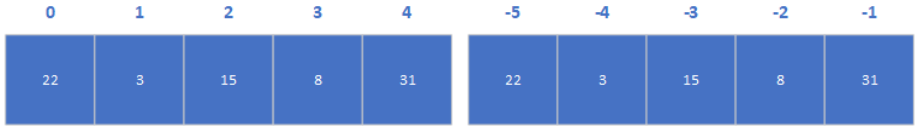
Out:
mylist = [1, 2, 3], alist = [4, 5, 6]

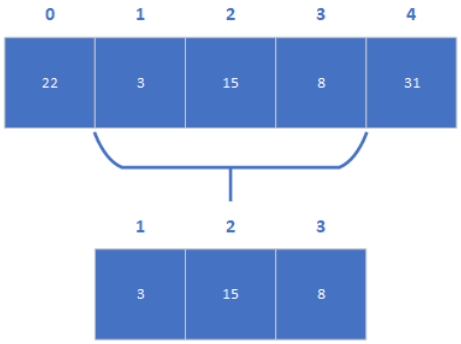
Hinzufügen/Entfernen von Elementen

```
mylist = [1, 2]
mylist.append(33)
mylist.extend([11, 22])
print(f'#1 mylist = {mylist}')
mylist.pop()
mylist.pop(2)
mylist.remove(11)
print(f'#2 mylist = {mylist}')
mylist.clear()
print(f'#3 mylist = {mylist}')
```

Out:
#1 mylist = [1, 2, 33, 11, 22]
#2 mylist = [1, 2]
#3 mylist = []

Indexierung/Slicing
slice = lst[start:end:step] 'inklusiv' start bis 'exklusiv' end






```
mylist = [22, 3, 15, 8, 31]
print(f'mylist[::] = {mylist[::]}')
print(f'mylist[1:4] = {mylist[1:4]}')
print(f'mylist[::2] = {mylist[::2]}')
print(f'mylist.index(3) = {mylist.index(3)}')
```

Out:
mylist[::] = [22, 3, 15, 8, 31]
mylist[1:4] = [3, 15, 8]
mylist[::2] = [22, 15, 31]
mylist.index(3) = 1

Sortieren

```
def myfunc(e):
    return len(e)


mylist = [4, 3, 5, 7, 3, 2]
strlist = ['BMW', 'Tesla', 'GM']
mylist.sort()
print('#1', mylist)
mylist.sort(reverse=True)
print('#2', mylist)
print('#3', mylist.count(2))
strlist.sort(key=myfunc)
print('#4', strlist)
```

Out:
#1 [2, 3, 3, 4, 5, 7]
#2 [7, 5, 4, 3, 3, 2]
#3 1
#4 ['GM', 'BMW', 'Tesla']

Tuple

Tuples sind ähnlich zu behandeln wie die Listen, nur dass sie nicht veränderbar sind.

tuple.index(x)	tuple.count(x)

```
mytupel = (1, "one", [0, 1])
print(mytupel)
```

Out:
(1, 'one', [0, 1])

Set

set.add(elem)	set.intersection(*others) → set	set.remove(elem)
set.clear()	set.intersection_update(*others)	set.symmetric_difference(other) → set
set.copy() → set	set.isdisjoint(other) → bool	set.symmetric_difference_update(other)
set.difference(*others) → set	set.issubset(other) → bool	set.union(*others) → set
set.difference_update(*others)	set.issuperset(other) → bool	set.update(*others)
set.discard(elem)	set.pop() → object	

Ein set kann ebenfalls wie eine Liste behandelt werden, wobei die Werte nur einmalig vorkommen können und sie nicht veränderbar sind. Weitere Werte können hinzugefügt oder entfernt werden.

```
myset = {"Work", "Life", 1, 2, 1}    # Duplikate werden ignoriert
myset.add("Balance")                 # {1, 2, 'Work', 'Balance', 'Life'}
```

Strings

```
string1 = "python"
string2 = """Mehrzeiliger
String"""
t = string1[2]
t = string1[-4]
```

```
# F-String example
fruit = 'apple'
color = 'red'
print(f'The fruit {fruit} has
    the color {color}')
```

Dictionary

```
moto = {
    'brand':    'honda',
    'color':    'red',
    'power':    43
    }
print(f'The motorcycle of {moto["brand
    "]} is {moto["color"]}.')
moto['year'] = 2002  # Add to dict
```

Methode	Beschreibung
clear()	Alle Elemente entfernen
copy()	Kopiert das Dictionary
keys()	Gibt eine Liste mit den Schlüsselwerten
items()	Gibt eine Liste mit tuple für alle keys
values()	Gibt eine Liste mit allen Werte zurück