lektion1

October 10, 2024

1 Lektion 1

1.1 Python als Taschenrechner

[1]:
```
1+3     # Addition  (Kommentare kann man mit '#' einfügen)
```

[1]:
```
4
```

[2]:
```
2*3+3
```

[2]:
```
9
```

[3]:
```
27/4    # Division
```

[3]:
```
6.75
```

[4]:
```
27//4   # ganzzahlige Division
```

[4]:
```
6
```

[5]:
```
27 % 4  # modulo
```

[5]:
```
3
```

Python nutzt $+, -, *, /$ für die Grundrechenarten, $**$ zum Potenzieren und $\%$ für Modulo Rechnungen.

[6]:
```
2**3    # Potenz
```

[6]:
```
8
```

[7]:
```
pow(2, 3)
```

[7]:
```
8
```

[8]:
```
? pow
```

1.2 Variablen

Variablen sind nützlich, um zum Beispiel Werte zu speichern. Das '=' ist hier kein mathematisches Gleichheitszeichen, sondern dient dazu einer Variablen einen Wert zuzuweisen. Variablennamen sind Zeichenketten, die mit einem Buchstaben anfangen müssen. Als Sonderzeichen ist nur _ (underscore) erlaubt. ## Zahlen #### ganze Zahl (integer)

[9]:
```
a_ganze_zahl = 2
```

Gleitkommazahl (float)

[10]:
```
b_gleitkomma_zahl = 3.
type(b_gleitkomma_zahl) # type ist eine eingebaute Funktion, die den Datentyp␣
↪anzeigt
```

[10]:
```
float
```

[11]:
```
a = 2
b = 3.
c = a*b
```

[12]:
```
c = a*b/5
```

[13]:
```
print(c)
```

```
1.2
```

komplexe Zahlen die imaginäre Einheit ist nicht i, sondern 1j, d.h. '1j**2=-1')

[14]:
```
1+1j
```

[14]:
```
(1+1j)
```

Verwirren Sie Python nicht, indem Sie die Namen eingebauter Funktionen (built-in) als Variablennamen verwenden.

Liste der built-in Funktionen https://docs.python.org/3/library/functions.html

[15]:
```
print = 2
```

Zeichenketten (engl. strings)

[16]:
```
txt_var = 'Hallo, Welt.'
txt_var
```

[16]:
```
'Hallo, Welt.'
```

[17]:
```
txt_var_2 = 'Hello, world!'
txt_var_2
```

[17]:
```
'Hello, world!'
```

[18]:
```
print(txt_var, txt_var_2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 1
----> 1 print(txt_var, txt_var_2)

TypeError: 'int' object is not callable
```

[19]:
```
del print # Alles wieder gut :)
```

[20]:
```
print(txt_var, txt_var_2)
```

```
Hallo, Welt. Hello, world!
```

[21]:
```
txt_var[0::2]
```

[21]:
```
'Hlo et'
```

[22]:
```
txt_var[-5]
```

[22]:
```
'W'
```

1.3 Listen (engl. list)

[23]:
```
leere_liste = list()
leere_liste
```

[23]:
```
[]
```

[24]:
```
noch_eine_leere_liste = []
noch_eine_leere_liste
```

[24]:
```
[]
```

[25]:
```
liste_mit_nullen = [0] * 5
```

[26]:
```
liste_mit_nullen
```

[26]:
```
[0, 0, 0, 0, 0]
```

[27]:
```
leere_liste.append(1)
```

[28]:
```
print(leere_liste)
```

```
[1]
```

[29]:
```
leere_liste.append('doc')
print(leere_liste)
```

```
[1, 'doc']
```

[30]:
```
len(leere_liste)
```

[30]:
```
2
```

1.3.1 Zugriff auf Elemente einer Liste (Indexzugriff)

[31]:
```
liste = [-1, 1, 2, 3, 44, 5, 6, 7]
liste
```

[31]:
```
[-1, 1, 2, 3, 44, 5, 6, 7]
```

[32]:
```
liste[4]
```

[32]:
```
44
```

Python fängt bei 0 mit dem Zählen an

[33]:
```
liste[0]
```

[33]:
```
-1
```

[34]:
```
liste[4:7] # liste[start:ende]
```

[34]:
```
[44, 5, 6]
```

Achtung! Das “ende” ist exklusiv.

[35]:
```
liste[0:4:2] # liste[start:ende:increment]
```

[35]:
```
[-1, 2]
```

[36]:
```
liste[-2]  # das vorletzte Element
```

[36]:
```
6
```

[37]:
```
liste[::-1]
```

[37]:
```
[7, 6, 5, 44, 3, 2, 1, -1]
```

[38]:
```
liste_von_listen = [[1, 2], [3, 4]]
liste_von_listen
```

[38]:
```
[[1, 2], [3, 4]]
```

[39]:
```
liste_von_listen[1][1]
```

[39]:
```
4
```

1.4 Dictionaries (Wörterbücher)

[40]:
```
leeres_dict = dict()
leeres_dict
```

[40]:
```
{}
```

[41]:
```
noch_ein_leeres_dict = {}
```

[42]:
```
ein_dict = {2: "Hallo ", 3: 'Donald'}  # Schlüssel-Werte Paare
```

Zugriff auf Inhalt des Dictionaries und aneinanderkleben von Strings

[43]:
```
ein_dict[2] + ein_dict[3] + '!'
```

[43]:
```
'Hallo Donald!'
```

1.4.1 .. Dictionaries

[44]:
```
ein_dict = {2: "Hallo ", 3: 'Donald'}
```

[45]:
```
ein_dict['s'] = '43' # neuer Eintrag mit Schlüssel 's'
ein_dict
```

[45]:
```
{2: 'Hallo ', 3: 'Donald', 's': '43'}
```

[46]:
```
print(ein_dict)
```

```
{2: 'Hallo ', 3: 'Donald', 's': '43'}
```

[47]:
```
del ein_dict[3] # Löscht den Eintrag mit Schlüssel 3
```

[48]:
```
print(ein_dict)
```

```
{2: 'Hallo ', 's': '43'}
```

[49]:
```
ein_dict['s']
```

[49]:
```
'43'
```

1.5 Tupel

[50]:
```
leeres_tupel = ()
leeres_tupel
```

[50]:
```
()
```

[51]:
```
lt = tuple()
lt
```

[51]:
```
()
```

[52]:
```
tp = ('tupel', 'lassen', 'sich', 'nicht', 'aendern')
tp
```

[52]:
```
('tupel', 'lassen', 'sich', 'nicht', 'aendern')
```

[53]:
```
tp[0] = 'Tupel'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[53], line 1
----> 1 tp[0] = 'Tupel'

TypeError: 'tuple' object does not support item assignment
```

1.6 Mengen

[54]:
```
leere_menge = set()
leere_menge
```

[54]:
```
set()
```

[55]:
```
staedte = {'Duesseldorf', 'Berlin', 'Muenchen', 'Schalke', "Dortmund"}
```

[56]:
```
'Berlin' in staedte # Ist Element in Menge enthalten?
```

[56]:
```
True
```

[57]:
```
staedte2 = {'Berlin', 'Hamburg'}
```

[58]:
```
staedte | staedte2  # Vereinigung 'union'
```

[58]:
```
{'Berlin', 'Dortmund', 'Duesseldorf', 'Hamburg', 'Muenchen', 'Schalke'}
```

1.7 Kopieren

[59]:
```
a = 1
a
```

[59]: 1

[60]:
```
b = a
b
```

[60]: 1

[61]:
```
id(a) , id(b) # a und b haben dieselbe Identität
```

[61]: (94468576917640, 94468576917640)

[62]:
```
a = 2
a
```

[62]: 2

[63]:
```
b
```

[63]: 1

[64]:
```
id(a), id(b)
```

[64]: (94468576917672, 94468576917640)

1.8 Kopieren II

[65]:
```
al = [1, [2, 3]]
```

[66]:
```
bl = al
```

[67]:
```
bl[1][0] = "Hallo"
```

[68]:
```
al
```

[68]: [1, ['Hallo', 3]]

[69]:
```
cl = al[:]
cl
```

[69]: [1, ['Hallo', 3]]

[70]:
```
cl[0] = 42
cl
```

[70]: [42, ['Hallo', 3]]

[71]:
```
al
```

[71]: [1, ['Hallo', 3]]

[72]:
```
cl[1][0] = 2018
cl
```

[72]: [42, [2018, 3]]

[73]:
```
al
```

[73]: [1, [2018, 3]]