20211007_Imformationsmanagement_FAT1

October 7, 2021

[1]:
```
# Tuples: geordnet, nicht veränderbare, Duplikaten erlaubt

thistuple = ('apple', 'banana', 'cherry')

print(thistuple)
```

```
('apple', 'banana', 'cherry')
```

[2]:
```
print(len(thistuple))
```

```
3
```

[3]:
```
# unterschiedliche Datatypen in Tuples

tuple1 = ("apple", "banana", "cherry") # Strings
tuple2 = (1, 5, 7, 9, 3) # Integer
tuple3 = (True, False, False) # Boolean
tuple4 = ("abc", 34, True, 40, "male") # Mix
```

[4]:
```
# Datatypen im Tuple darstellen

mytuple = ("apple", "banana", "cherry")
print(type(mytuple))
```

```
<class 'tuple'>
```

[5]:
```
# tuples update in Listenform

x = ('apple', 'banana', 'cherry')
y = list(x)

print(type(y))
```

```
<class 'list'>
```

[6]:
```
x = tuple(y)

print(type(x))
```

```
<class 'tuple'>
```

[7]:
```
# Elementen einfügen in dem Tuple, dadurch wir Tuple als Liste umwandeln,
# und dann erst ein Objekt einfügen, einschliesslich wird dann die Tuple wieder␣
 ↪hergestellt

thistuple = ('apple', 'banana', 'cherry')

y  = list(thistuple)

y.append('orange') #hier fügen wir ein Element ein

thistuple = tuple(y)

print(thistuple)
```

```
('apple', 'banana', 'cherry', 'orange')
```

[11]:
```
# Übung.

# Bitte definiere einen Tuple mit 5 Elementen,
# addiere einen Element am Ende mit "append" und drucke das Ergebnis aus.

Dies_ist_ein_Tuple = ('Element_0', 'Element_1', 'Element_2', 'Element_3',␣
 ↪'Element_4')

Dies_ist_eine_Liste_aus_dem_Tuple = list(Dies_ist_ein_Tuple)

Dies_ist_eine_Liste_aus_dem_Tuple.append('Element_neu')

Dies_ist_ein_Tuple = tuple(Dies_ist_eine_Liste_aus_dem_Tuple)

print(Dies_ist_ein_Tuple)
print(type(Dies_ist_ein_Tuple))
```

```
('Element_0', 'Element_1', 'Element_2', 'Element_3', 'Element_4', 'Element_neu')
<class 'tuple'>
```

[12]:
```
# Elementen können entfernt werden aus dem Tuple

Dies_ist_ein_Tuple.remove('Element_4')

print(Dies_ist_ein_Tuple)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-6e38d15155c8> in <module>
      1 # Elementen können entfernt werden aus dem Tuple
```

```
      2
----> 3 Dies_ist_ein_Tuple.remove('Element_4')
      4
      5 print(Dies_ist_ein_Tuple)

AttributeError: 'tuple' object has no attribute 'remove'
```

[13]:
```
# Elementen aus Tuples können nur entfernt werden,
# wenn sie zuerst als Listen umgewandelt werden.

Dies_ist_ein_Tuple = ('Element_0', 'Element_1', 'Element_2', 'Element_3',␣
 ↪'Element_4')

Dies_ist_eine_Liste_aus_dem_Tuple = list(Dies_ist_ein_Tuple)

Dies_ist_eine_Liste_aus_dem_Tuple.remove('Element_4')

Dies_ist_ein_Tuple = tuple(Dies_ist_eine_Liste_aus_dem_Tuple)

print(Dies_ist_ein_Tuple)
print(type(Dies_ist_ein_Tuple))
```

```
('Element_0', 'Element_1', 'Element_2', 'Element_3')
<class 'tuple'>
```

[19]:
```
Dies_ist_ein_Tuple = ('Element_0', 'Element_1', 'Element_2', 'Element_3',␣
 ↪'Element_4')

Dies_ist_eine_Liste_aus_dem_Tuple = list(Dies_ist_ein_Tuple)

Dies_ist_eine_Liste_aus_dem_Tuple.remove(Dies_ist_eine_Liste_aus_dem_Tuple[3])

Dies_ist_ein_Tuple = tuple(Dies_ist_eine_Liste_aus_dem_Tuple)

print(Dies_ist_ein_Tuple)
print(type(Dies_ist_ein_Tuple))
```

```
('Element_0', 'Element_1', 'Element_2', 'Element_4')
<class 'tuple'>
```

[14]:
```
# Set: nicht geordnet, nicht veränderbar, keine Duplikaten erlaubt

thisset = {'apple', 'banana', 'cherry', 'apple'}

print(thisset)
print(type(thisset))
```

```
{'banana', 'cherry', 'apple'}
<class 'set'>
```

[15]:
```
print(len(thisset))
```

```
3
```

[16]:
```
# Loop durch die Elemente vom Set, um bestimmte Elemente herauszufinden

thisset = {'apple', 'banana', 'cherry'}

for x in thisset:
    print(x)
```

```
banana
cherry
apple
```

[17]:
```
# Elementen können in Sets addiert werden

thisset = {'apple', 'banana', 'cherry'}
tropical = {'ananas', 'mango', 'papaya'}

thisset.update(tropical)

print(thisset)
```

```
{'mango', 'banana', 'ananas', 'cherry', 'apple', 'papaya'}
```

[22]:
```
# Dictionaries: veränderbare, Duplikaten werden nicht erlaubt!

thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year" : 1964
}

print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

[23]:
```
# elemente aus Dict darsetellen

print(thisdict['brand'])
```

```
Ford
```

[24]:
```
# es werden keine duplikaten erlaubt

thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year" : 1964,
    "year" : 2018
}

print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

[25]:
```
print(len(thisdict))
```

```
3
```

[26]:
```
# elemente vom Dictionary wechseln

thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year" : 1964
}

thisdict['year'] = 1980

print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1980}
```

[27]:
```
# If... Else... "Konditionen"
#
#   Equals: a == b
#   Not Equals: a != b
#   Lessthan: a < b
#   Lessthanorequalto: a <= b
#   Greater than: a > b
#   Greater than or equal to: a >= b
```

[28]:
```
a = 33
b = 200

if b > a:
    print('b ist grösser als a')
```

```
b ist grösser als a
```

[31]:
```
# elif: Kondition bezogen auf die vorherige Kondition
# else: Kondition bezogen auf ALLE vorherige Konditionen

a = 150
b = 90

if b > a:
    print('b ist grösser als a')
elif a == b:
    print('a und b sind gleich')
else:
    print('a ist grösser als b')
```

```
a ist grösser als b
```

[32]:
```
# mit "and" mehrere Konditionen gleichzeitig darstellen

a = 200
b = 33
c = 500

if a > b and c > a:
    print('Beide Konditionen sind wahr')
```

```
Beide Konditionen sind wahr
```

[33]:
```
# mit "or" auch mehrere Konditionen darstellen

if a > b or a > c:
    print('Mindestens eine Konditon ist wahr')
```

```
Mindestens eine Konditon ist wahr
```

[37]:
```
# mehrere "if" "elif" "else" Konditionen ineinander darstellen...

x = 19

if x > 10:
    print('x ist grösser 10',)
    if x > 20:
        print('auch grösser 20!')
    else: # weil es eine Indentation gibt, bezieht sich auf "if x>20"
        print("aber nicht über 20.")
else: # weil es keine Indentation hat, bezieht sich auf "if x>10"
    print('x ist kleiner oder gleich 10.')
```

```
x ist grösser 10
aber nicht über 20.
```

[39]:
```
'''In this program, we input a number
check if the number is positive or
negative or zero and display
an appropriate message
This time we use nested if statement'''

num = float(input("Enter a number: ")) # Eine zahl wird vom Nutzer abgefragt
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

```
Enter a number: 12
Positive number
```

[40]:
```
'''In this program,
we check if the number is positive or
negative or zero and
display an appropriate message'''

num = 3.4

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

```
Positive number
```

[42]:
```
# "while" während eine bestimmte Kondition wahr ist, dann tue etwas...

i = 1

while i <= 6:
    print(i)
    i += 1
```

```
1
2
```

```
3
4
5
6
```

[43]:
```python
# "while" kann mit "if" kombiniert werden

i = 1

while i < 6:
    print(i)
    if i == 3:
        break # while loop wird gestoppt
    i += 1
```

```
1
2
3
```

[46]:
```python
i = 1

while i < 6:
    print(i)
    i += 1
else:
    print('i ist nicht mehr kleiner 6')
```

```
1
2
3
4
5
i ist nicht mehr kleiner 6
```

[58]:
```python
a = ['Enes', 'Justin', 'Nino', 'Maria']

s = ['H4']

i = 0 # Variable die stellt die Position in der Liste []

while i < len(a):
    if a[i] == s: # Processing for item found
        print(s, 'found in list.')
        break
    i += 1
else:
    print(s, 'not found in the list')
```

```
['H4'] not found in the list
```

[59]:
```
## https://realpython.com/python-while-loop/
```

[63]:
```
age = 73
gender = 'F'

if age < 18:
    if gender == 'M':
        print('Sohn')
    else:
        print('Tochter')
elif age >= 18 and age < 65:
    if gender == 'M':
        print('Vater')
    else:
        print('Mutter')
else:
    if gender == 'M':
        print('G-Vater')
    else:
        print('G-Mutter')
```

```
G-Mutter
```

[65]:
```
a = ['foo', 'bar']

while len(a):
    print(a.pop(0))
    while len(a):
        print(a.pop(0))
```

```
foo
bar
```

[66]:
```
# Übung: Erstelle eine Liste mit 6 Elementen.
# Definiere ein neues Element.
# Nutze: "format" "while" und "if", um herauszufinden
# ob ein Wort in der Liste, die gleiche Länge hat, wie das neue Element.
```

[ ]:
```

```