Spotkanie 1

Wprowadzenie do Pythona

Typy proste

Typy złożone

```
6*7
```

```
42
```

```
_
```

```
42
```

```
zm=_
print(zm,type(zm))
```

```
42 <class 'int'>
```

```
zm.__dir__()
```

```
['__repr__',
 '__hash__',
 '__getattribute__',
 '__lt__',
 '__le__',
 '__eq__',
 '__ne__',
 '__gt__',
 '__ge__',
 '__add__',
 '__radd__',
 '__sub__',
 '__rsub__',
 '__mul__',
 '__rmul__',
 '__mod__',
 '__rmod__',
 '__divmod__',
 '__rdivmod__',
 '__pow__',
 '__rpow__',
 '__neg__',
```

```
 '__pos__',
 '__abs__',
 '__bool__',
 '__invert__',
 '__lshift__',
 '__rlshift__',
 '__rshift__',
 '__rrshift__',
 '__and__',
 '__rand__',
 '__xor__',
 '__rxor__',
 '__or__',
 '__ror__',
 '__int__',
 '__float__',
 '__floordiv__',
 '__rfloordiv__',
 '__truediv__',
 '__rtruediv__',
 '__index__',
 '__new__',
 'conjugate',
 'bit_length',
 'to_bytes',
 'from_bytes',
 'as_integer_ratio',
 '__trunc__',
 '__floor__',
 '__ceil__',
 '__round__',
 '__getnewargs__',
 '__format__',
 '__sizeof__',
 'real',
 'imag',
 'numerator',
 'denominator',
 '__doc__',
 '__str__',
 '__setattr__',
 '__delattr__',
 '__init__',
 '__reduce_ex__',
 '__reduce__',
 '__subclasshook__',
 '__init_subclass__',
 '__dir__',
 '__class__']
```

```
zm=7*6.4
```

```
print(zm,type(zm))
```

```
44.800000000000004 <class 'float'>
```

Typy złożone

```
#Lista
```

```
lista=list()
print(lista,type(lista))
```

```
[] <class 'list'>
```

```
lista.append('Ola')
print(lista,type(lista))
```

```
['Ola'] <class 'list'>
```

```
lista=[1,2,3,'Tola',[3,7,'Ula']]
print(lista,type(lista))
```

```
[1, 2, 3, 'Tola', [3, 7, 'Ula']] <class 'list'>
```

```
lista[3]
```

```
'Tola'
```

```
#krotki (tuple)

zm=tuple(lista)
print(zm,type(zm))
zm.
```

```
(1, 2, 3, 'Tola', [3, 7, 'Ula']) <class 'tuple'>
```

```
print(zm)
zm[4].append('XXXXXX')
print(zm)
```

```
(1, 2, 3, 'Tola', [3, 7, 'Ula'])
(1, 2, 3, 'Tola', [3, 7, 'Ula', 'XXXXXX'])
```

```
zm[2]=5
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)
```

```
Cell In[20], line 1
----> 1 zm[2]=5


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

```
# typ słownikowy

zm=dict()
print(zm,type(zm))
```

```
{} <class 'dict'>
```

```
zm['Ula']=5600

zm[(1,2)]='Nie wiem co to jest'

print(zm)
```

```
{'Ula': 5600, (1, 2): 'Nie wiem co to jest'}
```

```
zm['Ula']=7200
print(zm)
```

```
{'Ula': 7200, (1, 2): 'Nie wiem co to jest'}
```

```
zm[(1,2)]
```

```
'Nie wiem co to jest'
```

```
zm[1]
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

Cell In[26], line 1
----> 1 zm[1]


KeyError: 1
```

```
zm="Witam wszystkich uczestników kursu"

for nr,słowo in enumerate(zm.split(' ')):
    print(nr,słowo)
    print("========")
print("Koniec roboty")
```

```
0 Witam
```

```
========
1 wszystkich
========
2 uczestników
========
3 kursu
========
Koniec roboty
```

```
zm1,zm2=45*7,3>8

print(zm1,type(zm1))
print(zm2,type(zm2))
```

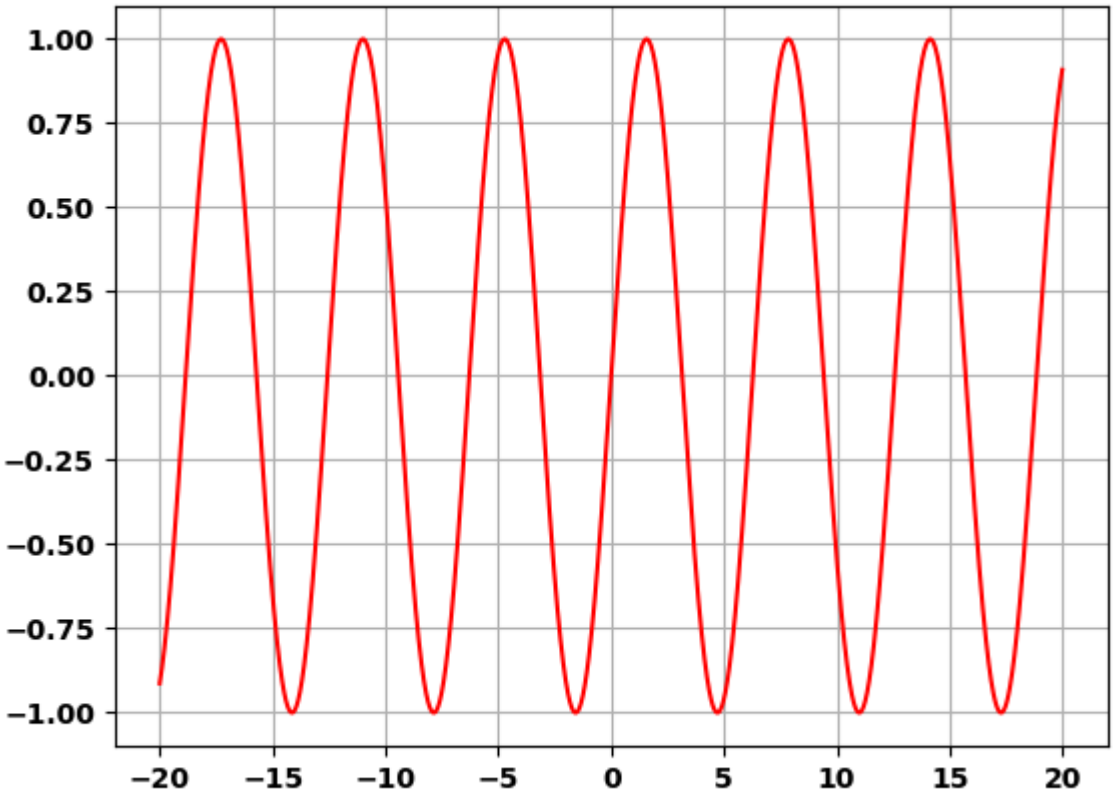
```
315 <class 'int'>
False <class 'bool'>
```

```
zm=45*7,3>8
print(zm)
```

```
(315, False)
```

```
import matplotlib.pyplot as plt
import numpy as np

x=np.arange(-20,20,0.01)
y=np.sin(x)
plt.plot(x,y,color='red')
plt.grid()
plt.show()
```



png

Wprowadzenie do przetwarzania danych z pakietami numpy i pandas

```
import numpy as np
import pandas as pd
# Wprowadzenie do numpy
lista=[4,6,7,6,5,43,12,3,7,11,3,6]
[x**2 for x in lista]

tablica=np.array(lista)
print(tablica,type(tablica), tablica.dtype)

tablica**2
```

```
[ 4  6  7  6  5 43 12  3  7 11  3  6] <class 'numpy.ndarray'> int64




array([  16,   36,   49,   36,   25, 1849,  144,    9,   49,  121,    9,
         36])
```

```
# Tablice wielowymiarowe
m1=np.array([[4,7,2],[3,5,8]])
m1
```

```
array([[4, 7, 2],
       [3, 5, 8]])
```

```
# TAblice specjalne
np.zeros((4,6))
```

```
array([[0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0.]])
```

```
# TAblice specjalne
np.ones((4,6))
```

```
array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.]])
```

```
# genowanie sekwencji jako tablicy
t1=np.arange(-8,8,0.01)
t1
```

```
array([-8.  , -7.99, -7.98, ...,  7.97,  7.98,  7.99])
```

```
t2=np.linspace(-1,1,10)
t2
```

```
array([-1.        , -0.77777778, -0.55555556, -0.33333333, -0.11111111,
        0.11111111,  0.33333333,  0.55555556,  0.77777778,  1.        ])
```

```
# Indeksowanie tablicy
print(tablica[2])
print(tablica[2:5])
```

```
7
[7 6 5]
```

```
# zmiana kształtu tablicy
m1=tablica.reshape((3,4))
```

```
m1
```

```
array([[ 4,  6,  7,  6],
       [ 5, 43, 12,  3],
```

```
       [ 7, 11,  3,  6]])
```

```
m1**2
```

```
array([[  16,   36,   49,   36],
       [  25, 1849,  144,    9],
       [  49,  121,    9,   36]])
```

```
m1[0:2,0:2]
```

```
array([[ 4,  6],
       [ 5, 43]])
```

```
# Operacje na tablicach
tb1=np.array([5,3,9])
tb2=np.array([3,1,7])

print(tb1+tb2)
print(tb1*tb2)
print(tb1**2)
print(tb1>5)
```

```
[ 8  4 16]
[15  3 63]
[25  9 81]
[False False  True]
```

```
# Funkcje statystyczne na tablicach
print(tablica)
np.mean(tablica)
```

```
[ 4  6  7  6  5 43 12  3  7 11  3  6]




np.float64(9.416666666666666)
```

```
print(tablica)
print(tablica.mean())
print(tablica.std())
```

```
[ 4  6  7  6  5 43 12  3  7 11  3  6]
9.416666666666666
10.467874134173035
```

```
# Wygenerowanie tablicy losowej
np.random.seed(33)
m1=np.random.randint(0,10,(5,5))
```

```
m1
```

```
array([[4, 7, 8, 2, 2],
       [9, 9, 3, 6, 3],
       [3, 1, 7, 6, 0],
       [0, 6, 6, 0, 4],
       [8, 8, 3, 7, 9]])
```

```
# Statystyki z wierszy
print(m1.mean(axis=1))
print(m1.max(axis=1))
```

```
[4.6 6.  3.4 3.2 7. ]
[8 9 7 6 9]
```

```
# Statystyki z kolumn
print(m1.mean(axis=0))
print(m1.max(axis=0))
```

```
[4.8 6.2 5.4 4.2 3.6]
[9 9 8 7 9]
```

Wprowadzenie do pakietu pandas

```
import numpy as np
import pandas as pd
# Typ danych series

s1=pd.Series([5,7,3,2,8,9])
print(s1,type(s1))
```

```
0    5
1    7
2    3
3    2
4    8
5    9
dtype: int64 <class 'pandas.core.series.Series'>
```

```
print(s1[2])
print(s1[1:4])
```

```
3
1    7
2    3
3    2
dtype: int64
```

```
s1=pd.Series([5,7,3,2,8,9],index=['a','a','b','c','b','a'])
print(s1,type(s1))
```

```
a    5
a    7
b    3
c    2
b    8
a    9
dtype: int64 <class 'pandas.core.series.Series'>
```

```
s1['a']
```

```
a    5
a    7
a    9
dtype: int64
```

```
s1[1:4]
```

```
a    7
b    3
c    2
dtype: int64
```

```
# Operacje na sieriach
s1**2
```

```
a    25
a    49
b     9
c     4
b    64
a    81
dtype: int64
```

```
# Typ DataFrame
dane={
        'Imię':['Ola','Tola','Ula','Ala','Zula','Ela'],
        'Wiek':[23,18,34,27,17,32],
        'Miasto' : ['Warszawa','Opole','Opole','Sopot','Warszawa','Sopot'],
        'Pensja' : [6700,5432,8760,6450,5670,8245]
     }

tb=pd.DataFrame(dane)
```

```
tb.to_csv(".\Dane2.csv")
```

```
tb=pd.read_csv(".\Dane2.csv")
tb
```

Unnamed: 0

Imię

Wiek

Miasto

Pensja

0

0

Ola

23

Warszawa

6700

1

1

Tola

18

Opole

5432

2

2

Ula

34

Opole

8760

3

3

Ala

27

Sopot

6450

4

4

Zula

17

Warszawa

5670

5

5

Ela

32

Sopot

8245

```
#Wstępne badanie DataFrame
tb.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  6 non-null      int64
 1   Imię        6 non-null      object
 2   Wiek        6 non-null      int64
 3   Miasto      6 non-null      object
 4   Pensja      6 non-null      int64
dtypes: int64(3), object(2)
memory usage: 368.0+ bytes
```

```
tb.shape
```

```
(6, 5)
```

```
tb.describe()
```

Unnamed: 0

Wiek

Pensja

count

6.000000

6.000000

6.000000

mean

2.500000

25.166667

6876.166667

std

1.870829

7.082843

1354.669025

min

0.000000

17.000000

5432.000000

25%

1.250000

19.250000

5865.000000

50%

2.500000

25.000000

6575.000000

75%

3.750000

30.750000

7858.750000

max

5.000000

34.000000

8760.000000

```
#Filtrowanie danych
w=tb['Wiek']>20
tb[w]
```

Unnamed: 0

Imię

Wiek

Miasto

Pensja

0

0

Ola

23

Warszawa

6700

2

2

Ula

34

Opole

8760

3

3

Ala

27

Sopot

6450

5

5

Ela

32

Sopot

8245

```
# Grupowanie danych
tb
```

Unnamed: 0

Imię

Wiek

Miasto

Pensja

0

0

Ola

23

Warszawa

6700

1

1

Tola

18

Opole

5432

2

2

Ula

34

Opole

8760

3

3

Ala

27

Sopot

6450

4

4

Zula

17

Warszawa

5670

5

5

Ela

32

Sopot

8245

```
tb.groupby('Miasto')['Pensja'].mean()
```

```
Miasto
Opole       7096.0
Sopot       7347.5
Warszawa    6185.0
Name: Pensja, dtype: float64
```

```
# Indeksowanie za pomocą metod iloc i loc
dane= {
```

```
    'Imię': ['Anna', 'Marek', 'Ewa', 'Piotr', 'Kasia'],
    'Wiek': [28, 35, 22, 45, 30],
    'Miasto': ['Warszawa', 'Kraków', 'Warszawa', 'Gdańsk', 'Kraków'],
    'Dochód': [3500, 4000, 2800, 5000, 4200]
}
tb1=pd.DataFrame(dane,index=['a','a','b','c','d'])
tb1
```

Imię

Wiek

Miasto

Dochód

a

Anna

28

Warszawa

3500

a

Marek

35

Kraków

4000

b

Ewa

22

Warszawa

2800

c

Piotr

45

Gdańsk

5000

d

Kasia

30

Kraków

4200

```
tb1.head(3)
```

Imię

Wiek

Miasto

Dochód

a

Anna

28

Warszawa

3500

a

Marek

35

Kraków

4000

b

Ewa

22

Warszawa

2800

```
tb1.tail(2)
```

Imię

Wiek

Miasto

Dochód

c

Piotr

45

Gdańsk

5000

d

Kasia

30

Kraków

4200

```
#Indeksowanie metodą loc
tb1.loc['a']
```

Imię

Wiek

Miasto

Dochód

a

Anna

28

Warszawa

3500

a

Marek

35

Kraków

4000

```
tb1['Wiek']
```

```
a    28
a    35
b    22
c    45
d    30
Name: Wiek, dtype: int64
```

```
tb1.loc[['a','c'],['Wiek','Miasto']]
```

Wiek

Miasto

a

28

Warszawa

a

35

Kraków

c

45

Gdańsk

```
tb1.loc['a':'c','Wiek':'Dochód']
```

Wiek

Miasto

Dochód

a

28

Warszawa

3500

a

35

Kraków

4000

b

22

Warszawa

2800

c

45

Gdańsk

5000

```
#Indeksowanie metod iloc
tb1.iloc[2]
```

```
Imię           Ewa
Wiek            22
Miasto    Warszawa
Dochód        2800
Name: b, dtype: object
```

```
tb1.iloc[2,3]
```

```
np.int64(2800)
```

```
tb1.iloc[1:3,0:2]
```

Imię

Wiek

a

Marek

35

b

Ewa

22

```
tb1.iloc[[0,2],[2,3]]
```

Miasto

Dochód

a

Warszawa

3500

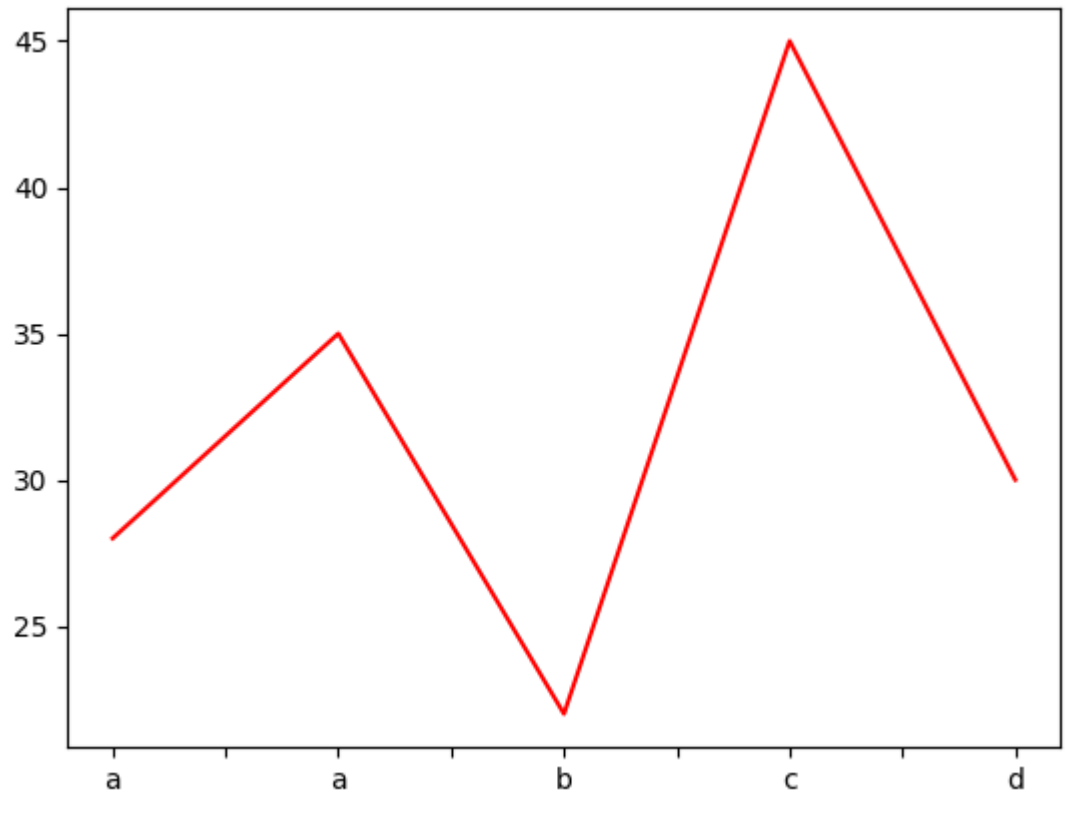
b

Warszawa

2800

```
tb1['Wiek'].plot(color='red')
```

```
<Axes: >
```



png