1 Listy

1. Tworzenie listy:

```
# tworzenie listy: '[]', list()

[1,"2",'a3']
[1, '2', 'a3']

list([1,"2",'a3'])
[1, '2', 'a3']

list((1,"2",'a3'))
[1, '2', 'a3']

type([1,"2",'a3']) # spr. typu
list
```

```
In [177]: # przypisanie listy do zmiennej

In [178]: x = [1,2,3,4,5]

In [179]: print(x, type(x))
[1, 2, 3, 4, 5] <class 'list'>

In [180]: y = x

In [181]: y
Out[181]: [1, 2, 3, 4, 5]
```

```
In [182]: # konwersja 'str' --> 'list'

In [183]: x = 'a1b2c3'

In [184]: y = list(x)

In [185]: z = [x]

In [186]: print(x,y,z)
a1b2c3 ['a', '1', 'b', '2', 'c', '3'] ['a1b2c3']
```

```
# listy o elementach tego samego typu

x, y = [1,2,3] , ['av','xxx']

x
[1, 2, 3]

y
['av', 'xxx']

# listy o elementach różnego typu / zagnieżdżone

x = [99,'abc',[1,2,'xxx',['a1','b1','c1']],1000]

x
[99, 'abc', [1, 2, 'xxx', ['a1', 'b1', 'c1']], 1000]
```

2. Dostęp do elementów list / wycinki

```
# wycinki: start, stop, step

x
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

x[1:] # od indeksu '1' do końca
['1', '2', '3', '4', '5', '6', '7', '8', '9']

x[:-1] # od indeksu '0' do przedostatniego
['0', '1', '2', '3', '4', '5', '6', '7', '8']

x[3:6] # wszystkie z zakresu (bez liczby 6)
['3', '4', '5']

x[:6:2] # co drugi z zakresu od '0' do '6'
['0', '2', '4']

x[::3] # co trzeci z całego zakresu
['0', '3', '6', '9']
```

```
# zmiana wartości elementów listy

x = [1,2,3,4,5,6,7,8,9]

x
[1, 2, 3, 4, 5, 6, 7, 8, 9]

x[0] = 'abc' # zmiana wartości pierwszego el.

x
['abc', 2, 3, 4, 5, 6, 7, 8, 9]

x[2:4] = 999,'888' # zmiana dwóch: 2 i 3

x
['abc', 2, 999, '888', 5, 6, 7, 8, 9]
```

2 Tuple

1. Tworzenie tupli:

```
In [276]: # tworzenie tupli: '()', tuple()

In [277]: (1,"2",'a3')
Out[277]: (1, '2', 'a3')

In [278]: tuple((1,"2",'a3'))
Out[278]: (1, '2', 'a3')

In [279]: tuple([1,"2",'a3'])
Out[279]: (1, '2', 'a3')

In [280]: type((1,"2",'a3'))
Out[280]: tuple
```

```
In [266]: # przypisanie tupli do zmiennej

In [267]: x = (1,2,3,4,5)

In [268]: print(x, type(x))
(1, 2, 3, 4, 5) <class 'tuple'>

In [269]: y = x

In [270]: y
Out[270]: (1, 2, 3, 4, 5)
```

```
# konwersja 'str' --> 'tuple'

x = 'a1b2c3'

y = tuple(x)

z = (x,) # konieczny jest znak ','!!!

y
('a', '1', 'b', '2', 'c', '3')

z
('a1b2c3',)
```

```
# tuple o elementach tego samego typu

x, y = (1,2,3) , ('av','xxx')

x
(1, 2, 3)

y
('av', 'xxx')

# tuple o elementach różnego typu / zagnieżdżone

x = (99,'abc',[1,2,'xxx',('a1','b1','c1')],1000)

x
(99, 'abc', [1, 2, 'xxx', ('a1', 'b1', 'c1')], 1000)

type(x)
tuple
```

2. Dostęp do elementów tupli / wycinki

```
# wycinki: start, stop, step

x = tuple('0123456789')

x
('0', '1', '2', '3', '4', '5', '6', '7', '8', '9')

x[0]
'0'

x[1:]
('1', '2', '3', '4', '5', '6', '7', '8', '9')

x[:-1]
('0', '1', '2', '3', '4', '5', '6', '7', '8')

x[3:6]
('3', '4', '5')

x[::3]
('0', '3', '6', '9')
```

```
In [322]: # zmiana wartości elementów tupli

In [323]: x = (1,2,3,4,5,6,7,8,9)

In [324]: x
Out[324]: (1, 2, 3, 4, 5, 6, 7, 8, 9)

In [325]: x[0] = 99
---------------------------------------------------------------
TypeError                                 Traceback (most r
<ipython-input-325-8cfed8487fe6> in <module>
----> 1 x[0] = 99

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

3 Słowniki

1. Tworzenie słownika:

```
# tworzenie słownika: '{}', dict()

{} # pusty słownik
{}

dict({}) # pusty słownik
{}

{'klucz1' : 'wartosc1', 'k2':'wart2'}
{'klucz1': 'wartosc1', 'k2': 'wart2'}

dict({'klucz1' : 'wartosc1', 'k2':'wart2'})
{'klucz1': 'wartosc1', 'k2': 'wart2'}

dict([('klucz1', 'wartosc1'), ('k2','wart2')]) #lista tupli / list
{'klucz1': 'wartosc1', 'k2': 'wart2'}

dict((('klucz1', 'wartosc1'), ['k2','wart2'])) # tupla list/tupli
{'klucz1': 'wartosc1', 'k2': 'wart2'}
```

```
# przypisanie słownika do zmiennej

s1 = dict({'kl1': 'v1', 'kl2': 'v2'})

s1
{'kl1': 'v1', 'kl2': 'v2'}

type(s1)
dict

s2 = s1

s2
{'kl1': 'v1', 'kl2': 'v2'}
```

```
# słownik o różnych typach kluczy i wartości

# klucze: int, float, str, tuple:

{12:'v1', 3.333:'v2', 'k1': 'v3', ('a',4): 'v4'}
{12: 'v1', 3.333: 'v2', 'k1': 'v3', ('a', 4): 'v4'}

# różne wartości: str, list, dict, tuple:

{1: '1', '1' : '1', '11a': [1,2,3], 's1': {'a': ['a'], 'b': (9,9)}}
{1: '1', '1': '1', '11a': [1, 2, 3], 's1': {'a': ['a'], 'b': (9, 9)}}
```

```
# słownik - dostęp do danych: s1[klucz] --> wartość

s = {1: '1', '1' : '1', '11a': [1,2,3], 's1': {'a': ['a'], 'b': (9,9)}}

s
{1: '1', '1': '1', '11a': [1, 2, 3], 's1': {'a': ['a'], 'b': (9, 9)}}

s[1] # prosty dostęp
'1'

s['11a']
[1, 2, 3]

s['11a'][1] # dostęp do zagnieżdżonej listy
2

s['11a'][::2] # dostęp do zagnieżdżonej listy
[1, 3]

s['s1'] # do podsłownika
{'a': ['a'], 'b': (9, 9)}

s['s1']['b'] # do podsłownika, do tupli
(9, 9)

s['s1']['b'][0] # do podsłownika, do 1 el. tupli
9
```

```
# słownik: dodawanie i zmiana danych

s =  {'imie':'Adam', 'wzrost':185}

s['waga'] = 77 # dodanie nowej pozycji

s
{'imie': 'Adam', 'wzrost': 185, 'waga': 77}

s['wiek'] = 25 # dodanie nowej pozycji

s
{'imie': 'Adam', 'wzrost': 185, 'waga': 77, 'wiek': 25}

s['waga'] = 81 # zmiana wartości

s
{'imie': 'Adam', 'wzrost': 185, 'waga': 81, 'wiek': 25}
```

4 Zbiory

```
# definicja zbioru: '{}', set():

{1,2,'xx'}
{1, 2, 'xx'}

set({1,2,'xx'})
{1, 2, 'xx'}

set((1,2,'xx'))
{1, 2, 'xx'}

set([1,2,'xx'])
{1, 2, 'xx'}

set('abcd19')
{'1', '9', 'a', 'b', 'c', 'd'}

type({1,2,'xx'})
set
```

```
l1, l2 = list(range(10)), list(range(3,13))

# operacje na zbiorach

s1,s2 = {*l1}, set(l2)

s1
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

s2
{3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

s1.intersection(s2) # część wspólna
{3, 4, 5, 6, 7, 8, 9}

s1.difference(s2) # są w s1 a nie ma w s2
{0, 1, 2}

s2.difference(s1) # są w s2 a nie ma w s1
{10, 11, 12}
```

5 Obiekty, atrybuty, metody

```
# notacja z '.', lista atrybutów i metod

x = 4.33 # liczba 'float'

x.is_integer() # spr. czy 'int'
False

x._
   as_integer_ratio() hex()              real
   conjugate()        imag
   fromhex()          is_integer()
```

```
0]: x = 'Adam' # string

1]: x.upper()  # metoda
1]: 'ADAM'

2]: x._
 capitalize()   encode()       format()       isalp
 casefold()     endswith()     format_map()   isasc
 center()       expandtabs()   index()        isdec
 count()        find()         isalnum()      isdig
```

5.1 'string' - przykłady metod

```
x = '  BMW,Audi,Fiat,Renault\n   '

x
'  BMW,Audi,Fiat,Renault\n   '

x.strip()  # usuwa białe znaki z początku i końca
'BMW,Audi,Fiat,Renault'

x.split(',') # podział na podanym znaku
['  BMW', 'Audi', 'Fiat', 'Renault\n   ']

# łączenie metod

x.strip().split(',')
['BMW', 'Audi', 'Fiat', 'Renault']
```

```
x
'  BMW,Audi,Fiat,Renault\n   '

x.replace(',',':') # zamienia podstring na nowy
'  BMW:Audi:Fiat:Renault\n   '

x.replace('Audi','VW') # zamienia podstring na nowy
'  BMW,VW,Fiat,Renault\n   '

x.count('a') # zlicza wystąpienia
2

x.count('BMW') # zlicza wystąpienia
1

x.find('a')  # znajduje indeks pierwszego wystąpienia
13

x.isdigit() # test czy str to cyfry
False

'123'.isdigit()
True
```

5.2 'list' - przykłady metod

```
l = ['Adam',185,77]

l
['Adam', 185, 77]

l.append(44.5) # dodaje element na końcu listy

l
['Adam', 185, 77, 44.5]

l.pop(-1) # usuwa i zwraca element o podanym indeksie
44.5

l
['Adam', 185, 77]

l.insert(1,'Nowak') # wstawia obiekt we wskazane miejsce

l
['Adam', 'Nowak', 185, 77]

l.remove('Nowak')  # usuwa wskazany element / nie zwraca

l
['Adam', 185, 77]
```

5.3 'dict' - przykłady metod

```
s =  {'imie':'Adam', 'wzrost':185}

s
{'imie': 'Adam', 'wzrost': 185}

s['waga'] = 77 # dodanie nowej pozycji

s['wiek'] = 25 # dodanie nowej pozycji

s
{'imie': 'Adam', 'wzrost': 185, 'waga': 77, 'wiek': 25}

s.pop('waga') # usuwa pozycję / zwraca wartość
77

s.popitem()  # usuwa ostatnią pozycję
('wiek', 25)

s
{'imie': 'Adam', 'wzrost': 185}
```

```
s =  {'imie':'Adam', 'wzrost':185}

s
{'imie': 'Adam', 'wzrost': 185}

s.keys() # zwraca klucze
dict_keys(['imie', 'wzrost'])

list(s.keys()) # konwersja na liste
['imie', 'wzrost']

s.values() # zwraca wartości
dict_values(['Adam', 185])

s.items() # zwraca tuple (key,val)
dict_items([('imie', 'Adam'), ('wzrost', 185)])
```

```
s =  {'imie':'Adam', 'wzrost':185}

s
{'imie': 'Adam', 'wzrost': 185}

s.keys() # zwraca klucze
dict_keys(['imie', 'wzrost'])

list(s.keys()) # konwersja na liste
['imie', 'wzrost']

s.values() # zwraca wartości
dict_values(['Adam', 185])

s.items() # zwraca tuple (key,val)
dict_items([('imie', 'Adam'), ('wzrost', 185)])
```