1 Wybrane funkcje wbudowane

1. len():

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

len(x) # liczba elementów stingu
27

l = ['Adam',185,77]

len(l) # liczba elementów listy
3

s
{'imie': 'Adam', 'wzrost': 185, 'wiek': 99}

len(s) # liczba elementów słownika
3
```

2. range():

```
range(5) # obiekt sekwencji liczb 0-4
range(0, 5)

list(range(5)) # lista
[0, 1, 2, 3, 4]

tuple(range(0,5)) # start, stop
(0, 1, 2, 3, 4)

tuple(range(1,10,2)) # start, stop, step
(1, 3, 5, 7, 9)
```

3. zip()

```
l1 = ['a','b','c']

l2 = [1,2,3]

zip(l1,l2) # tworzy obiekt iteratora
<zip at 0x7ff4482b0aa0>

list(zip(l1,l2)) # tworzy liste tupli
[('a', 1), ('b', 2), ('c', 3)]

tuple(zip(l1,l2)) # tworzy tuple tupli
(('a', 1), ('b', 2), ('c', 3))

dict(zip(l1,l2)) # tworzy słownik
{'a': 1, 'b': 2, 'c': 3}
```

```
l1 = ['a','b','c']

l2 = [1,2,3]

t = tuple(zip(l1,l2)) # tupla tupli

t
(('a', 1), ('b', 2), ('c', 3))

a1,a2 = zip(*t) # rozpakowuje tuple

a1
('a', 'b', 'c')

a2
(1, 2, 3)
```

```
l = list(zip(l1,l2)) # lista tupli

l
[('a', 1), ('b', 2), ('c', 3)]

a1,a2 = zip(*l) # rozpakowuje tuple

a1
('a', 'b', 'c')

a2
(1, 2, 3)
```

4. round()

```
4/3
1.3333333333333333

round(4/3)
1

round(4/3,1)
1.3

round(4/3,3)
1.333
```

5. enumerate()

```
l = ['a','b','c']

enumerate(l) # zwraca obiekt iteratora
<enumerate at 0x7ff448136dc0>

list(enumerate(l,5)) # lista
[(5, 'a'), (6, 'b'), (7, 'c')]

tuple(enumerate(l)) # tupla tupli
((0, 'a'), (1, 'b'), (2, 'c'))

dict(enumerate(l)) # słownik
{0: 'a', 1: 'b', 2: 'c'}

list(enumerate(l,5)) # umerowanie od 5
[(5, 'a'), (6, 'b'), (7, 'c')]
```

6. print()

```
In [711]: print(12)
12

In [712]: print(12,'xxx')
12 xxx

In [713]: print(12,'xxx',999)
12 xxx 999

In [714]: print(12,'xxx',999,sep='::')
12::xxx::999

In [715]: print(12,'xxx',999,sep='\n')
12
xxx
999

In [716]: print('\t',12,'xxx',999,)
        12 xxx 999
```

2 Warunki

```
In [746]: n = 10

In [747]: i = 5

In [748]: if 2*i > n:
     ...:     print('\t2*i>n')
     ...: elif i == n:
     ...:     print('\ti=n!!')
     ...: else:
     ...:     print('\tblabla')
     ...:
        blabla
```

3 Pętle

1. Pętla *for*:

```
In [726]: for i in range(3):
     ...:     print('\ti: ',i)
     ...:
        i:  0
        i:  1
        i:  2

In [727]: for i in range(1,10,3):
     ...:     print('\ti: ',i)
     ...:
        i:  1
        i:  4
        i:  7
```

```
In [728]: for it in 'abc':
     ...:     print('\tit: ',it)
     ...:
        it:  a
        it:  b
        it:  c

In [729]: for it in ['a','b','c']:
     ...:     print('\tit: ',it)
     ...:
        it:  a
        it:  b
        it:  c
```

```
In [740]: l = ['a','b','c']

In [741]: for i,it in enumerate(l,3):
     ...:     print('\ti=',i,',  it=',it)
     ...:
       i= 3 ,  it= a
       i= 4 ,  it= b
       i= 5 ,  it= c
```

```
In [730]: s =  {'imie':'Adam', 'wzrost':185, 'waga': 75}

In [731]: s
Out[731]: {'imie': 'Adam', 'wzrost': 185, 'waga': 75}

In [732]: for key in s.keys():
     ...:     print(key,s[key],sep=' --> ')
     ...:
imie --> Adam
wzrost --> 185
waga --> 75

In [733]: for key,val in s.items():
     ...:     print(key,val,sep=' --> ')
     ...:
imie --> Adam
wzrost --> 185
waga --> 75
```

```
In [762]: for i in range(11):
     ...:     if i%2 == 0:
     ...:         print('parzysta: ',i)
     ...:
parzysta:  0
parzysta:  2
parzysta:  4
parzysta:  6
parzysta:  8
parzysta:  10

In [763]: for i in range(11):
     ...:     if i%2 == 0 and i > 0:
     ...:         print('parzysta: ',i)
     ...:
parzysta:  2
parzysta:  4
parzysta:  6
parzysta:  8
parzysta:  10

In [764]: pr = []

In [765]: for i in range(11):
     ...:     if i%2 == 0 and i > 0:
     ...:         pr.append(i)
     ...:

In [766]: pr
Out[766]: [2, 4, 6, 8, 10]
```

2. Pętla *while*:

```
In [744]: i = 0

In [745]: while i < 5:
     ...:     print('\ti: ',i)
     ...:     i += 1
     ...:
        i:  0
        i:  1
        i:  2
        i:  3
        i:  4
```

```
47]: # pętla 'while' i wartości: 0 - False, 1 - True

48]: i,j = 0, 0 # j = 0 czyli False

49]: while j:   # pętla nie zadziała
...:     print(f'{"i":>7}: {i}, j: {j}')
...:     i += 1
...:     if i == 3:
...:         j = None
...:

50]: i,j = 0, 1 # j = 1 czyli True

51]: while j:
...:     print(f'{"i":>7}: {i}, j: {j}')
...:     i += 1
...:     if i == 3:
...:         j = None
...:
  i: 0, j: 1
  i: 1, j: 1
  i: 2, j: 1
```

```
n [104]: # warunek, że obiekt ma jakieś wartości

n [105]: l = list('abc')

n [106]: l
ut[106]: ['a', 'b', 'c']

n [107]: while l:
    ...:     print(f'{"len":>10}: {len(l)}, {l}')
    ...:     l.pop(0)
    ...:
     len: 3, ['a', 'b', 'c']
     len: 2, ['b', 'c']
     len: 1, ['c']

n [108]: # while z else:

n [109]: l = list('abc')

n [110]: while l:
    ...:     print(f'{"len":>10}: {len(l)}, {l}')
    ...:     l.pop(0)
    ...: else:
    ...:     print(f'{"Koniec":>10}: len: {len(l)}, bool: {bool(l)}')
    ...:
     len: 3, ['a', 'b', 'c']
     len: 2, ['b', 'c']
     len: 1, ['c']
  Koniec: len: 0, bool: False
```

3. *break* i *continue*

```
In [132]: # break - kończenie pętli

In [133]: l = list('abcde')

In [134]: for i in l:
     ...:     print(f'{i:>8}')
     ...:     if i == 'c':
     ...:         print(f'Koniec pętli - i = {i}')
     ...:         break
     ...:
       a
       b
       c
Koniec pętli - i = c

In [135]: j = -1

In [136]: while 1:
     ...:     j += 1
     ...:     print(f'{l[j]:>8}')
     ...:     if l[j] == 'c':
     ...:         print(f'Koniec pętli, l[j] = {l[j]}')
     ...:         break
     ...:
       a
       b
       c
Koniec pętli, l[j] = c
```

```
In [166]: # pominięcie i kontynuacja: continue

In [167]: s = 'a1b2c3'

In [168]: for i in s:
     ...:     if i.isdigit():
     ...:         continue
     ...:     print(f'Nie jestem "digit": {i}')
     ...:
     ...:
Nie jestem "digit": a
Nie jestem "digit": b
Nie jestem "digit": c

In [169]: j = -1

In [170]: while j < len(s)-1:
     ...:     j += 1
     ...:     if s[j].isdigit():
     ...:         continue
     ...:     print(f'Nie jestem "digit": {s[j]}')
     ...:
Nie jestem "digit": a
Nie jestem "digit": b
Nie jestem "digit": c
```

4 Listy składane

```
[x for x in 'abcdef']
['a', 'b', 'c', 'd', 'e', 'f']

[2*x for x in 'abcdef']
['aa', 'bb', 'cc', 'dd', 'ee', 'ff']

[2*x.upper() for x in 'abcdef']
['AA', 'BB', 'CC', 'DD', 'EE', 'FF']

[int(x)**2 for x in '12345']
[1, 4, 9, 16, 25]
```

```
[x**3 for x in [1,2,3,4,5]]
[1, 8, 27, 64, 125]

s =  {'imie':'Adam', 'wzrost':185, 'waga': 75}

[x for x in s.keys()]
['imie', 'wzrost', 'waga']

[x for x,y in s.items()]
['imie', 'wzrost', 'waga']

[y for x,y in s.items()]
['Adam', 185, 75]
```

```
[x if x%2 == 0 else 'nan' for x in range(11)]
[0, 'nan', 2, 'nan', 4, 'nan', 6, 'nan', 8, 'nan', 10]

[x if x%3 == 0 else 'nan' for x in range(11)]
[0, 'nan', 'nan', 3, 'nan', 'nan', 6, 'nan', 'nan', 9, 'nan']

[x for x in range(11) if x%3 == 0 and x > 2]
[3, 6, 9]
```

5 Słowniki składane

```
In [381]: # dane wejściowe:

In [382]: l1, l2 = list(range(5)), ('a','b','c','d','e')

In [383]: f'{l1}{"":5}{l2}'
Out[383]: "[0, 1, 2, 3, 4]     ('a', 'b', 'c', 'd', 'e')"

In [384]: # słowniki składane (dictionary domprehension)

In [385]: {key:val for key,val in zip(l2,l1)}
Out[385]: {'a': 0, 'b': 1, 'c': 2, 'd': 3, 'e': 4}

In [386]: {key:(val,val**2,val**3) for key,val in zip(l2,l1)}
Out[386]:
{'a': (0, 0, 0),
 'b': (1, 1, 1),
 'c': (2, 4, 8),
 'd': (3, 9, 27),
 'e': (4, 16, 64)}

In [387]: {f'{key}/{val}':val for key,val in zip(l2,l1)}
Out[387]: {'a/0': 0, 'b/1': 1, 'c/2': 2, 'd/3': 3, 'e/4': 4}

In [388]: {(key,val):val+15 for key,val in zip(l2,l1)}
Out[388]: {('a', 0): 15, ('b', 1): 16, ('c', 2): 17, ('d', 3): 18, ('e', 4): 19}
```