1 Otwieranie zamykanie pliku

Wygenerowanie danych liczbowych

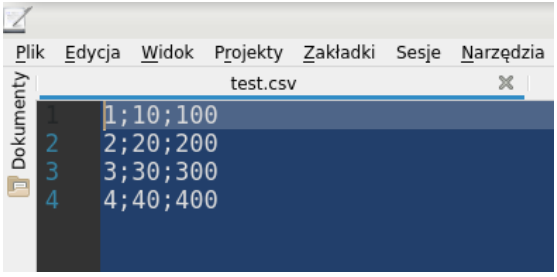
```
In [465]: # utworzenie danych tekstowych do zapisania w pliku:

In [466]: # 4 wiersze, 3 kolumny, separator: ';'

In [467]: data = [[f'{x*i:d}' for x in (1,10,100)] for i in range(1,5)]

In [468]: data
Out[468]:
[['1', '10', '100'],
 ['2', '20', '200'],
 ['3', '30', '300'],
 ['4', '40', '400']]

In [469]: data = [';'.join(x) for x in data]

In [470]: data
Out[470]: ['1;10;100', '2;20;200', '3;30;300', '4;40;400']

In [471]: data = '\n'.join(data)

In [472]: print(data)
1;10;100
2;20;200
3;30;300
4;40;400
```

##. Podstawowe użycie funkcji *open()*

- zapis danych:

```
# zapis danych do pliku 'test.csv'

# open('adres_pliku','tryb_zapisu'): podstawowe użycie

ad = '/home/u1/01tmp/test.csv'

f = open(ad,mode='w')

f.write(data)
35

f.close()
```

- widok danych zapisanych na dysku:



- odczyt danych:

```
In [479]: # odczyt danych z pliku na dysku

In [480]: ad
Out[480]: '/home/u1/01tmp/test.csv'

In [481]: f = open(ad)  # mode='r' - domyślny tryb

In [482]: dd = f.read()

In [483]: f.close()

In [484]: type(dd)
Out[484]: str

In [485]: print(dd)
1;10;100
2;20;200
3;30;300
4;40;400
```

##. Menadżer kontekstu

```
In [570]: # menadżer kontekstu z open()

In [571]: ad
Out[571]: '/home/u1/01tmp/test.csv'

In [572]: with open(ad) as f:
     ...:     print(f.read())
     ...:
1;10;100
2;20;200
3;30;300
4;40;400

In [573]: # wielokrotne odczytanie pliku

In [574]: for i in range(5):
     ...:     with open(ad) as f:
     ...:         print(f'i:{i:<4}{f.readlines()}')
     ...:
i:0   ['1;10;100\n', '2;20;200\n', '3;30;300\n', '4;40;400']
i:1   ['1;10;100\n', '2;20;200\n', '3;30;300\n', '4;40;400']
i:2   ['1;10;100\n', '2;20;200\n', '3;30;300\n', '4;40;400']
i:3   ['1;10;100\n', '2;20;200\n', '3;30;300\n', '4;40;400']
i:4   ['1;10;100\n', '2;20;200\n', '3;30;300\n', '4;40;400']
```

2 Funkcje

```
# definiowanie funkcji: słowo 'def'

def f1():  # funkcja bez argumentów
    pass   # słowo kluczowe: nic nie robi


type(f1)
function

f1()  # wywołanie funkcji

f1.__doc__  # sprawdzenie opisu (tu brak)
```

```
In [175]: def f1():  # funkcja bez argumentów
     ...:     ''' Opis funkcji:
     ...:         - wyświetla stały napis 'Hejka'
     ...:     '''
     ...:     print('Hejka')
     ...:

In [176]: f1() # wywołanie funkcji
Hejka

In [177]: print(f1.__doc__) # wyświetla opis funkcji
 Opis funkcji:
       - wyświetla stały napis 'Hejka'
```

```
In [195]: def f1(a1,a2):  # dwa argumenty wymagane
     ...:     ''' Args:
     ...:             - a1, a2:  liczby, str, etc.
     ...:         Wyświetla podane argumenty.
     ...:     '''
     ...:     newline = '\n'
     ...:     print(f"{'a1':>7}: {a1}{newline}{'a2':>7}: {a2}")
     ...:

In [196]: f1('argus1','argus2') # argumenty dopasowane wg pozycji
     a1: argus1
     a2: argus2

In [197]: f1(a2=99,a1=[1,'xxx']) # argumenty dopasowane wg nazwy
     a1: [1, 'xxx']
     a2: 99

In [198]: f1(a1=[1,'xxx'], a2=99) # argumenty dopasowane wg nazwy
     a1: [1, 'xxx']
     a2: 99

In [199]: # arumenty przekazane przez zmienne

In [200]: x,y = {1:12,'k2':(12,15)},1000

In [201]: f1(x,y)
     a1: {1: 12, 'k2': (12, 15)}
     a2: 1000

In [202]: f1(a2=y,a1=x)
     a1: {1: 12, 'k2': (12, 15)}
     a2: 1000
```

```
In [203]: def f1(a1,a2='xxxxx'):  # dwa argumenty: jeden wymagany
     ...:     ''' Args:
     ...:             - a1, a2:  liczby, str, etc.
     ...:         Wyświetla podane argumenty.
     ...:     '''
     ...:     newline = '\n'
     ...:     print(f"{'a1':>7}: {a1}{newline}{'a2':>7}: {a2}")
     ...:

In [204]: f1(x)
     a1: {1: 12, 'k2': (12, 15)}
     a2: xxxxx

In [205]: f1(a1=x)
     a1: {1: 12, 'k2': (12, 15)}
     a2: xxxxx

In [206]: f1('abc',(10,20))
     a1: abc
     a2: (10, 20)

In [207]: f1(a1='abc',a2=(10,20))
     a1: abc
     a2: (10, 20)
```

1. Zmienna liczba argumentów - *args

```
0]: # zmienna liczba argumentów '*args'

1]: def f2(*xx):
..:     '''Oblicza sumę argumentów.'''
..:     s = sum(xx)
..:     print(f'{s:>12d}')
..:

2]: f2(2)  # suma jednego argumentu
     2

3]: f2(1,1,1,1)
     4

4]: f2(*[1,2,3,4,5])
    15

5]: l = range(100)

6]: f2(*l)
  4950

7]: t = (1,2,3,4,5)

8]: f2(*t)
    15
```

```
In [287]: def f2(a1,a2=100,*xx):
     ...:     '''Wyświetla argumenty'''
     ...:     nl = '\n'   # znak nowej linii
     ...:     print(f"{'a1':>12}: {a1}{nl}{'a2':>12}: {a2}")
     ...:     print(f"{'xx':>12}: {xx}")
     ...:

In [288]: f2(1,2,3,4,5)
          a1: 1
          a2: 2
          xx: (3, 4, 5)

In [289]: f2(10,20,*(1,2,3,4,5))
          a1: 10
          a2: 20
          xx: (1, 2, 3, 4, 5)

In [290]: f2(10,*(1,2,3,4,5))
          a1: 10
          a2: 1
          xx: (2, 3, 4, 5)

In [291]: f2(10,*(1,2,3,4,5),a2=99)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-291-e83bf8e19dac> in <module>
----> 1 f2(10,*(1,2,3,4,5),a2=99)

TypeError: f2() got multiple values for argument 'a2'
```

```
In [331]: # pierwsze są argumenty pozycyjne, potem z kluczami

In [332]: def f2(a1,*xx,a2=100):
     ...:     '''Wyświetla argumenty'''
     ...:     nl = '\n'   # znak nowej linii
     ...:     print(f"{'a1':>12}: {a1}{nl}{'a2':>12}: {a2}")
     ...:     print(f"{'xx':>12}: {xx}")
     ...:

In [333]: f2(1,2,3,4,5)  # dowolna liczba arg. pozycyjnych
          a1: 1
          a2: 100
          xx: (2, 3, 4, 5)

In [334]: f2(10,*(1,2,3,4,5),a2=99) # arg. z kluczem po arg. pozycyjnych
          a1: 10
          a2: 99
          xx: (1, 2, 3, 4, 5)

In [335]: # błąd argument ze słowem kluczowym przed argumentem pozycyjnym

In [336]: f2(a1=10,*(1,2,3,4,5),a2=99) # błąd 'a1=10'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last
<ipython-input-336-dd3fe332fe49> in <module>
----> 1 f2(a1=10,*(1,2,3,4,5),a2=99) # błąd 'a1=10'

TypeError: f2() got multiple values for argument 'a1'

In [337]: f2(*(1,2,3,4,5),a1=10,a2=99) # błąd 'a1' - ponownie przypisany
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last
<ipython-input-337-728f6021aa29> in <module>
----> 1 f2(*(1,2,3,4,5),a1=10,a2=99) # błąd 'a1' - ponownie przypisany

TypeError: f2() got multiple values for argument 'a1'
```

2. **kwargs - argumenty typu słownikowego (klucz - wartość)

```
In [319]: def f2(a1,a2=100,**zz):
     ...:     '''Wyświetla argumenty'''
     ...:     nl = '\n'   # znak nowej linii
     ...:     print(f"{'a1':>12}: {a1}{nl}{'a2':>12}: {a2}")
     ...:     print(f"{'zz':>12}: {zz}")
     ...:

In [320]: f2(1,2) # bez **kwargs
          a1: 1
          a2: 2
          zz: {}

In [321]: f2(1,2,3)  # błąd! wartość bez klucza
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-321-079207332e65> in <module>
----> 1 f2(1,2,3)  # błąd! wartość bez klucza

TypeError: f2() takes from 1 to 2 positional arguments but 3 were given

In [322]: f2(1,2,a=3) # wartość prawidłowa - klucz:wartość
          a1: 1
          a2: 2
          zz: {'a': 3}

In [323]: f2(1,2,a=3,b=4, c='abc')
          a1: 1
          a2: 2
          zz: {'a': 3, 'b': 4, 'c': 'abc'}

In [324]: f2(a1=2,a=3,b=4, c='abc',a2=99)  # dowolny szyk
          a1: 2
          a2: 99
          zz: {'a': 3, 'b': 4, 'c': 'abc'}

In [325]: f2(10,a2=99,**{'a':3,'b':4, 'c':'abc'}) # rozpakowanie **
          a1: 10
          a2: 99
          zz: {'a': 3, 'b': 4, 'c': 'abc'}
```

3. *args muszą być przed **kwargs

```
7]: # wszystkie argumenty: pozyc, klucze, *args, **kwargs

8]: def f2(a1,*xx,a2=100,**zz):
...:     '''Wyświetla argumenty'''
...:     nl = '\n'   # znak nowej linii
...:     print(f"{'a1':>12}: {a1}{nl}{'a2':>12}: {a2}")
...:     print(f"{'xx':>12}: {xx}")
...:     print(f"{'zz':>12}: {zz}")
...:

9]: f2(10,2,3,4,5,a2=999,a=30,b=50)
    a1: 10
    a2: 999
    xx: (2, 3, 4, 5)
    zz: {'a': 30, 'b': 50}

0]: l = [1,2,3,4,5]

1]: sl = {'a':3,'b':4, 'c':'abc'}

2]: f2(10,*l,a2=999,**sl)
    a1: 10
    a2: 999
    xx: (1, 2, 3, 4, 5)
    zz: {'a': 3, 'b': 4, 'c': 'abc'}

3]: f2(10,*l,**sl)
    a1: 10
    a2: 100
    xx: (1, 2, 3, 4, 5)
    zz: {'a': 3, 'b': 4, 'c': 'abc'}

4]: f2(10,**sl,*l) # błąd!! **kwargs przed *args
 "<ipython-input-354-241a723be37e>", line 1
(10,**sl,*l) # błąd!! **kwargs przed *args
        ^
Error: iterable argument unpacking follows keyword argument unpacking
```

4. *return*

```
# 'return' -  funkcja zwraca wynik

def f1(a1,a2):  # dwa argumenty wymagane
    '''Suma.
    '''
    return a1 + a2


def f2(a1,a2):  # dwa argumenty wymagane
    '''Iloczyn.
    '''
    return(a1 * a2)


f1(10,4)
14

s = f1(3,9)

s
12

f2(10,4)
40

il = f2(3,4)

il
12
```

3 Przestrzenie nazw

```
In [5]: # dir() - wyświetla listę nazw w danym zakresie

In [6]: print(dir(),end=',')  # wyświetl listę nazw w zakresie globalnym
['In', 'Out', '_', '_4', '__', '___', '__builtin__', '__builtins__', '__doc__', '__loader__', '_
 '_i5', '_i6', '_ih', '_ii', '_iii', '_oh', 'exit', 'get_ipython', 'quit'],
In [7]:

In [7]: # filtracja metod/atrybutów specjalnych

In [8]: [x for x in dir() if '_' not in x]
Out[8]: ['In', 'Out', 'exit', 'quit']
```

```
In [6]: # defnicja 2 funkcji - zagnieżdżone użycie

In [7]: def f1():
   ...:     a,b = [1,2,3],10
   ...:     c = a.append(b)
   ...:     print(f"{'f1_dir()':>12}: {dir()}")  # wyświetl listę nazw
   ...:

In [8]: def f2():
   ...:     x,y = 2,3
   ...:     i = x * y
   ...:     f1()
   ...:     print(f"{'f2_dir()':>12}: {dir()}")
   ...:

In [9]: f1() # nazwy funkcji f1()
    f1_dir(): ['a', 'b', 'c']

In [10]: f2() # nazwy funkcji f2()
    f1_dir(): ['a', 'b', 'c']
    f2_dir(): ['i', 'x', 'y']

In [11]: # funkcja f1() nie jest nazwą z zakresu funkcji f2()!!!
```

```
In [12]: # zmiana przestrzeni nazw globalnych

In [13]: [x for x in dir() if '_' not in x]
Out[13]: ['In', 'Out', 'exit', 'f1', 'f2', 'quit']
```

```
In [15]: # definicja funkcji f3() wewnątrz funkcji f2()

In [16]: def f2():
    ...:     x,y = 2,3
    ...:     i = x * y
    ...:     def f3():
    ...:         x3,y3 = 'a',[1,2,3]
    ...:         print(f"{'f3_dir()':>12}: {dir()}")
    ...:     f3()
    ...:     print(f"{'f2_dir()':>12}: {dir()}")
    ...:

In [17]: f2()
    f3_dir(): ['x3', 'y3']
    f2_dir(): ['f3', 'i', 'x', 'y']

In [18]: # funkcja f3() jest nazwą z zakresu przestrzeni f2()

In [19]: # zmiana przestrzeni nazw globalnych - brak zmian

In [20]: [x for x in dir() if '_' not in x]
Out[20]: ['In', 'Out', 'exit', 'f1', 'f2', 'quit']

In [21]: # f3() nie jest nazwą z zakresu globalnego
```

```
22]: # wpływ przestrzeni nazw na zmienne i ich wartości

23]: x = 1   # zmienna globalna

24]: def f2():
...:     x = 2 # zmienna lokalna f2()
...:     def f3():
...:         x = 3 # zmienna lokalna f3()
...:         print(f"{'x':>12} wewnątrz f3(): {x}")
...:     f3()
...:     print(f"{'x':>12} wewnątrz f2(): {x}")
...:

25]: f2()
       x wewnątrz f3(): 3
       x wewnątrz f2(): 2

26]: x # globalne
26]: 1

27]: # zmiana przestrzeni nazw globalnych

28]: [x for x in dir() if '_' not in x]
28]: ['In', 'Out', 'exit', 'f1', 'f2', 'quit', 'x']
```

```
In [34]: # użycie zmiennej globalnej

In [35]: x = 100 # zmienna globalna

In [36]: def f2():
    ...:     global x # deklaracja zmiennej globalnej
    ...:     def f3():
    ...:         global x
    ...:         x = 3 # zmienna lokalna f3()
    ...:         print(f"{'x':>12} wewnątrz f3(): {x}, {x**2}")
    ...:     f3()
    ...:     print(f"{'x':>12} wewnątrz f2(): {x}, {x**2}")
    ...:
    ...:     x = 10 # zmiana zmiennej globalnej!!!
    ...:

In [37]: f2()
           x wewnątrz f3(): 3, 9
           x wewnątrz f2(): 3, 9

In [38]: x  # sprawdzenie wartości - powinna zostać zmieniona
Out[38]: 10
```

```
In [45]: # odwołania do zmiennej globalnej:

In [46]: x = 100 # zmienna globalna

In [47]: def f1():
    ...:     y = x*2
    ...:     print(y)
    ...:

In [48]: def f2():
    ...:     x = x*2
    ...:     print(x)
    ...:

In [49]: f1() # znajdzie zmienną globalną
200

In [50]: f2() # błąd
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent cal
<ipython-input-50-a9e668f924c1> in <module>
----> 1 f2() # błąd

<ipython-input-48-97fa58cdffbb> in f2()
      1 def f2():
----> 2     x = x*2
      3     print(x)
      4

UnboundLocalError: local variable 'x' referenced before assignment
```

4 Prosty import modułów

1. Sposoby importowania modułów na przykładzie modułu *random*

```
4]: # trzy sposoby importowania modułów:

5]: import sys  # import całego modułu

6]: sys.path[0]    # dostęp do metod, funkcji etc.
6]: '/home/u1/anaconda3/envs/edc/bin'

7]: from math import *  # immport wszystkich nazw

8]: floor(3.333) # funkcja z modułu 'math'
8]: 3

9]: from random import choice  # import wybranej funkcjonalności

0]: choice('abcde')
0]: 'd'

1]: from random import sample as smp # import jako alias

2]: smp(range(30),5)
2]: [11, 6, 20, 10, 22]
```

2. Importowanie własnych modułów

```
In [23]: # dodanie tymczasowej ścieżki wyszukiwania modułów

In [24]: import sys   # moduł do interakcji z interpreterem

In [25]: sys.path     # lista ścieżek wyszukiwania
Out[25]:
['/home/u1/anaconda3/envs/edc/bin',
 '/home/u1/anaconda3/envs/edc/lib/python37.zip',
 '/home/u1/anaconda3/envs/edc/lib/python3.7',
 '/home/u1/anaconda3/envs/edc/lib/python3.7/lib-dynload',
 '',
 '/home/u1/.local/lib/python3.7/site-packages',
 '/home/u1/anaconda3/envs/edc/lib/python3.7/site-packages',
 '/home/u1/anaconda3/envs/edc/lib/python3.7/site-packages/IPython/extensi
 '/home/u1/.ipython']

In [26]: sys.path.append('/home/u1/01tmp/')  # tymczasowe dodanie adresu

In [27]: sys.path[-2:]
Out[27]: ['/home/u1/.ipython', '/home/u1/01tmp/']
```

```
In [33]: # przeładowanie modułu po dokonanych zmianach

In [34]: from importlib import reload

In [35]: type(reload)
Out[35]: function

In [36]: print(reload.__doc__)
Reload the module and return it.

    The module must have been successfully imported before.



In [37]: reload(sys)
Out[37]: <module 'sys' (built-in)>
```