



# ОСНОВИ ФАЙЛОВОГО ВВОДУ- ВИВОДУ В PYTHON

Питання 10.2.

# Файловий ввід-вивід у Python

---

- Для відкриття файлу в Python використовується функція `open()`.
  - При виклику створюється об'єкт типу «файл», з яким далі можна буде працювати.
- Синтаксис функції `open()`:
- `my_file = open(назва_файлу [, режим_доступу][, буферизація])`
  - **назва\_файлу:** рядок, що містить назву файлу з розширенням. Наприклад, "my\_file.txt".
  - **режим\_доступу:** рядок, яким вказується режим відкриття файлу: для читання, запису, доповнення інформації тощо. За замовчуванням файл відкривається на зчитування – "r".
  - **буферизація:** ціле число. Якщо значення 0, файл відкриється без буферизації, 1 – з порядковою буферизацією, більше 1 – розмір буфера. Від'ємне число – буфер дорівнює системному.

# Режими доступу до файлу

---

<b>r</b>	Відкриває файл тільки для читання. Вказівник стоїть на початку файлу.
<b>rb</b>	Відкриває файл для читання в двійковому форматі. Вказівник стоїть на початку файлу.
<b>r+</b>	Відкриває файл для читання і запису. Вказівник стоїть на початку файлу.
<b>rb+</b>	Відкриває файл для читання і запису в довічнім форматі. Вказівник стоїть на початку файлу.
<b>w</b>	Відкриває файл тільки для запису. Вказівник стоїть на початку файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>wb</b>	Відкриває файл для запису в довічнім форматі. Вказівник стоїть на початку файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>w+</b>	Відкриває файл для читання і запису. Вказівник стоїть на початку файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>wb+</b>	Відкриває файл для читання і запису в довічнім форматі. Вказівник стоїть на початку файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>a</b>	Відкриває файл для додавання інформації в файл. Вказівник варто в кінці файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>ab</b>	Відкриває файл для додавання в двійковій форматі. Вказівник варто в кінці файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>a+</b>	Відкриває файл для додавання і читання. Вказівник варто в кінці файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.
<b>ab+</b>	Відкриває файл для додавання і читання в двійковому форматі. Вказівник варто в кінці файлу. Створює файл з ім'ям <i>назва_файла</i> , якщо такого не існує.

# Використання режимів доступу

---

```
my_file = open("some.txt", "w")
print("Назва файлу: ", my_file.name)
print("Файл закритий: ", my_file.closed)
print("У якому режимі файл відкрито: ", my_file.mode)
print("Пробіли: ", my_file.softspace)
```

- Метод файлового об'єкта `close()` автоматично закриває файл, при цьому втрачається будь-яка незбережена інформація.
  - Працювати з файлом (читати, записувати) після цього не можна.
  - Python автоматично закриває файл, якщо файловий об'єкт, до якого він прив'язаний, присвоюється іншому файлу. Проте хорошою практикою буде вручну закривати файл командою `close()`.

```
my_file = open("some.txt")
print("Назва файлу: ", my_file.name)
print("Файл закритий: ", my_file.closed)
my_file.close()
print("А тепер закритий: ", my_file.closed)
```

# Читання і запис файлів у Python

---

- Метод `write()` записує будь-який рядок у відкритий файл.
  - Рядки в Python можуть містити двійкові дані, а не лише текст.
  - Метод `write()` не додає символ переносу рядка (`'\n'`) у кінець файла.
- Метод `read()` читає рядок з відкритого файлу.
  - Синтаксис метода `read()`: `my_file.read([count])`
  - Необов'язковий параметр `count` – це кількість байт, які слід прочитати з відкритого файлу.
  - Цей метод читає інформацію з початку файлу і, якщо параметр `count` невідомий, до кінця файлу.

```
my_file = open("some.txt", "w")
my_file.write("Мені подобається
Python!\nЦе класна мова!")
my_file.close()
```

```
1 myFile = open("myText.txt", "w")
2 myFile.write("Python шикарний!!!\nСупер мова!")
3 myFile.close()
4
5
6 myFile = open("myText.txt")
7 myString = myFile.read()
8 print("Було прочитано:")
9 print(myString)
10 myFile.close()
```

```
Було прочитано:
Python шикарний!!!
Супер мова!
```

# Як дізнатися позицію вказівника у файлі в Python

---

- Після того, як викликали метод `read()` на файловому об'єкті, якщо повторно викличете `read()`, то побачите лише порожній рядок.
  - Це відбувається тому, що після першого прочитання вказівник знаходиться в кінці файлу.
  - Для того, щоб дізнатися позицію вказівника, можна використовувати метод `tell()`.

```
my_file = open("some.txt")
my_file.read(10)
print ("Я на позиції: ", my_file.tell())
my_file.close()
```

# Щоб перейти на потрібну позицію, слід використовувати метод `seek()`

---

- Синтаксис метода `seek()`.
  - Метод `seek(offset[, from])` змінює поточну позицію в файлі.
  - `my_file.seek(offset, [from])`
  - Аргумент `offset` вказує кількість байтів для переміщення.
  - Аргумент `from` задає початкову позицію, звідки байти переміщатимуться. 0 - початок файлу, 1 - поточна позиція, 2 - кінець файлу.

```
my_file = open("some.txt", "r")
print(my_file.read(10))
print("Ми знаходимось на позиції: ", my_file.tell())
# Повертаємось на початок
my_file.seek(0)
print(my_file.read(10))
my_file.close()
```

# Додавання в файл. Метод write()

---

- Якщо хочете не перезаписати файл повністю (в разі відкриття файлу в режимі 'w'), а тільки додати будь-якої текст, то файл слід відкривати в режимі 'a' - appending.
  - Після чого можна використовувати той же метод write().

```
# Видавить існуючу інформацію в some.txt та запише "Hello".  
my_file = open("some.txt", 'w')  
my_file.write("Hello")  
my_file.close()
```

```
# Залишить існуючу інформацію в some.txt та додасть "Hello".  
my_file = open("some.txt", 'a')  
my_file.write("Hello")  
my_file.close()
```



# Розширена робота з файлами в Python

---

- Для доступу до ширшої функціональності для роботи з файловою системою слід підключити бібліотеку `os`.
  - `import os`
- Отримуємо шлях до поточної папки, в якій відбувається робота програми.
  - Ця папка не обов'язково повинна співпадати з тією папкою, з якої запущено додаток, можна і поміняти її на потрібну нам.

```
print (os.getcwd())      # повертає шлях до поточної робочої папки
os.chdir('D:/')          # встановлює шлях до поточної робочої папки
```

- Спробуємо отримати список файлів у поточній директорії з розширенням `'.py'`.

```
import os
import fnmatch
for file in os.listdir('.'):
    if fnmatch.fnmatch(file, '*.py'):
        print (file)
```

# Робота з файлами

---

- Видалити файл, який існує на диску:
  - `import os`
  - `os.remove('D:/test.txt')`
- Перейменування файлу:
  - `import os`
  - `os.rename('sample.txt', 'sample1.txt')`
- Відкриємо будь-який файл за допомогою програми, якою цей файл відкривається в системі:
  - `import os`
  - `os.startfile(r'C:/Program Files/Mozilla Firefox/firefox.exe')`

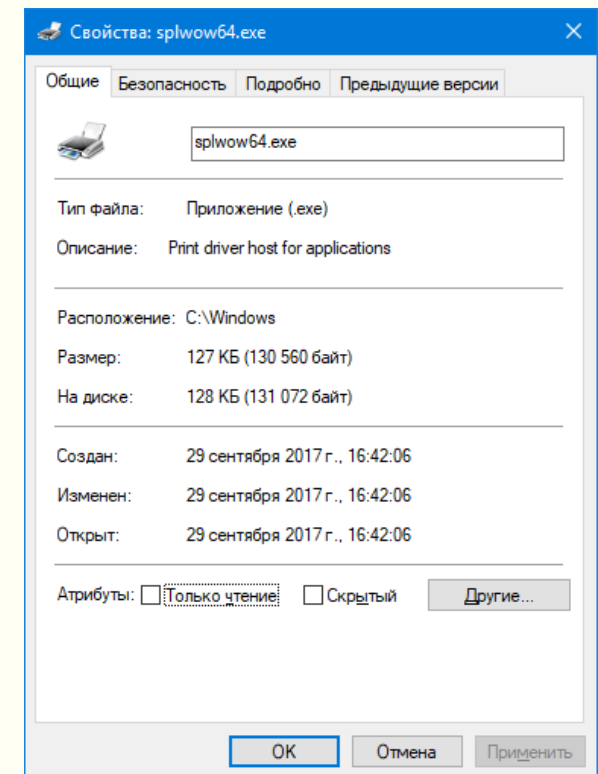
# Робота з папками

- Створимо папку за вказаним шляхом та відразу її видалимо.
  - Додатково підключимо модуль shutil.

```
1 import os.path
2 from datetime import datetime # для форматування дат
3
4 path = 'C:/Windows/splwow64.exe'
5 size = os.path.getsize(path) # розмір файлу в байтах
6 ksize = size//1024 # розмір у кілобайтах
7 # дата останнього доступу в секундах з початку епохи
8 atime = os.path.getatime(path)
9 # дата останньої модифікації в секундах з початку епохи
10 mtime = os.path.getmtime(path)
11 print ('Розмір: ', ksize, ' KB')
12 print ('Дата останнього використання: \n',
13         datetime.fromtimestamp(atime))
14 print ('Дата останнього редагування: \n',
15         datetime.fromtimestamp(mtime))
```

```
Розмір: 127 KB
Дата останнього використання:
2017-09-29 16:42:06.032440
Дата останнього редагування:
2017-09-29 16:42:06.032440
```

```
import os
import shutil
os.makedirs('D:/test/test2')
shutil.rmtree('D:/test')
```



# Надрукуємо список всіх файлів, що містяться всередині папок за визначеним шляхом (з огляду на вкладені підкаталоги)

---

```
import os
for root, dirs, files in os.walk('C:/Users/Public/'):
    for name in files:
        fullname = os.path.join(root, name)
        print(fullname + '\n')
```

- Корисні команди з модуля
  - `os.path.basename('D:/test/1/file.txt')` # повертає назву об'єкта (файл / папка) вказаного шляху
  - `os.path.dirname('D:/test/folder/fld/file.txt')` # повертає батьківський шлях до об'єкту шляху (перша частина `os.path.split`)
  - `os.path.splitext('D:/test/folder/fld/file.txt')` # розбиває шлях на шлях та розширення файла
  - `os.path.exists('D/1/temp.txt')` # чи існує шлях?
  - `os.path.isfile('D/test1.txt')` # чи є об'єкт шляху звичайним файлом? (існуючим)
  - `os.path.isdir('D:/test')` # чи є об'єкт шляху звичайною папкою? (існуючою)



# ДЯКУЮ ЗА УВАГУ!

Наступне питання: серіалізація об'єктів у мові Python. Консервування об'єктів