ОСНОВИ ФАЙЛОВОГО ВВОДУ-ВИВОДУ В PYTHON

Питання 9.2.

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

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

Використання режимів доступу

```
my_file = open("some.txt", "w")
print("назва файла: ", my_file.name)
print("Файл закрыт: ", my_file.closed)
print("В каком режиме файл открыт: ", my_file.mode)
print("Пробелы: ", 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 файла: ", my_file.name)
print("Файл закрыт: ", my_file.closed)
my_file.close()
print("А теперь закрыт: ", 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.

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

- 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.

```
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 позиции:", 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 файла, 1 - нинішня позиція, 2 - кінець файла.

```
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().

```
# Видалить 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

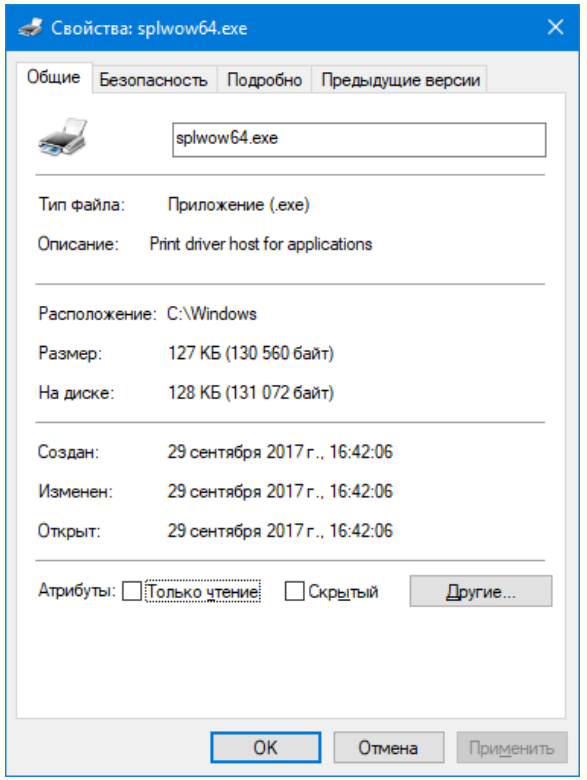
- Створимо папку за вказаним шляхом та відразу її видалимо.
  - Додатково підключимо модуль shutil.

```
import os.path
from datetime import datetime # для форматування дат

path = 'C:/Windows/splwow64.exe'
size = os.path.getsize(path) # розмір файлу в байтах
ksize = size//1024 # розмір у кілобайтах
# дата останнього доступу в секундах з початку епохи
atime = os.path.getatime(path)
# дата останньої модифікації в секундах з початку епохи
mtime = os.path.getmtime(path)
print ('Розмір: ', ksize, ' KB')
print ('Дата останнього використання: \n',
       datetime.fromtimestamp(atime))
print ('Дата останього редагування: \n',
       datetime.fromtimestamp(mtime))
```

```
Розмір:  127  KB
Дата останнього використання:
 2017-09-29 16:42:06.032440
Дата останього 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. Піклінг