Лекція 3.

Форматування даних про час та дату, читання та запис файлів в Python. Взаємодія із зовнішніми додатками

План лекції

3.1. Форматування даних про час та дату у Python.
3.2. Читання та запис файлів в Python.
3.3. Взаємодія із зовнішніми додатками.

3.1. Форматування даних про час та дату у Python

Модуль Python, який використовується для оброблення даних про час та дату, називається **datetime** [1]. Модуль **datetime** включений у більшість дистрибутивів Python як стандартна бібліотека; однак його потрібно імпортувати для використання у коді. Особливості модуля **datetime** представлені об'єктно-орієнтованою парадигмою програмування. Модуль складається з класів часу та дати. У кожного класу є свої методи, які можна викликати для роботи з екземплярами класів, які називаються об'єктами [2].

Рис.3.1 ілюструє застосування деяких основних об'єктів і методів, і включений в модуль **datetime**.

Datetime Module Terminolgy

Term	Example	Use
module	datetime	import **datetime** as dt
class	time date datetime, etc.	dt.**time** dt.**date** dt.**datetime**
object	Variables t, d, and dateAndTime	**t** = dt.time(12,31,00) **d** = dt.date(1972,12,31) **dateAndTime** = dt.dateandtime.now()
method	strftime() weekday() isoformat	t.**strftime**("%H:%M:%S") d.**weekday**() dateAndTime.**isoformat**()

Рис. 3.1. Застосування деяких основних об'єктів і методів **datetime** [3]

Метод стрипінгу використовує серію кодів форматування або директив як своїх параметрів (рис. 3.2).

Directive	Meaning
%a	Locale's abbreviated weekday name
%A	Locale's full weekday name
%b	Locale's abbreviated month name
%B	Locale's full month name
%c	Locale's appropriate date and time representation
%d	Day of the month as a number [01,31]
%H	Hour (24-hour clock) as a number [00,23]
%I	Hour (12-hour clock) as a number [01,12]
%j	Day of the year as a number [001,366]
%m	Month as a number [01,12]
%M	Minute as a number [00,59]
%p	Locale's equivalent of either AM or PM
%S	Second as a number [00,61]
%w	Weekday as a number [00,61]
%W	Week number of the year (Monday as the first day of the week) as a number [00,53]. All days in a new year preceding the first Monday are considered to be in week 0.
%y	Year without century as a number [00,99]
%Y	Year with century as a number
%Z	Time zone name (no characters if no time exists)

Рис.3.2. Список кодів форматування модуля **datetime** [3]

На рис. 3.3 показаний код Python, який використовує модуль **datetime** для подання дати та часу у форматі, який зазвичай використовується у США.

```
#load the datetime module as dt
import datetime as dt

#create a datetime object that contains the current time
currentDT = dt.datetime.now()

#view the value of currentDT
print(currentDT)
```

```
#create a new string object that contains the reformatted date and time
UDdt = currentDT.strftime('%b %d, %Y %I:%M %p')
#display the result
UDdt
```

Рис.3.3. Використання модуля **datetime** [3]

3.2. Читання та запис файлів в Python

Модуль **csv** є частиною стандартної бібліотеки Python. Модуль **csv** дозволяє читати та записувати у .csv файли. Python також має основні методи створення, відкриття та закриття зовнішніх файлів. Таблиці даних будуть існувати лише в оперативній пам'яті, поки вони не будуть збережені у файли.

Метод **open ()** використовується для створення нового файлу або для відкриття наявного файлу, який буде містити дані, які потрібно зберегти. Функція **close ()** видаляє дані з буферів і закінчує функцію запису файлу для вказаного файлу. Важливо закрити всі файли, в які не слід записувати дані. Це зберігає системні ресурси та захищає файл від пошкодження. На рис. 3.4 показаний синтаксис функції **open ()** та пояснено деякі важливі значення, які можна надати методу. Він також ілюструє використання методу **close ()**. Цей код створює файл, закриває його, а потім знову відкриває його в режимі додавання, щоб дані могли бути додані до файлу.

```
myFile = open("newText.txt", "w")
myFile.close()
myFile = open("newText.txt", "a")

print(myFile)
```

```
<open file 'newText.txt', mode 'a' at 0x729e2338>
```

Рис.3.4. Використання методів **open ()** та **close ()** модуля **datetime** [3]

Відкриття неіснуючого файлу в режимі "a" створить цей файл так само, як і в режимі "w". Різниця лише в тому, де знаходиться вказівник. Він буде вказувати на початок файлу або на кінець файлу.

На малюнку 3.5 пояснюються деякі важливі значення, які можна надати методу **open ()** . Ці параметри можуть бути об'єднані (символ "+"), щоб вказати, що повинні використовуватися як режими читання, запису та додавання.

```
open('myFile.txt', "w")
```

File open() Method Modes

Modes	Description
w	Opens an existing file that has the file name specified. If the file does not exist, it opens a new file with that name.
r	Opens an existing file in read only mode.
a	Opens an existing file and will append new data to the end of the file.

Рис.3.5. Параметри методу **open ()** [3]

Дані можна записати у файл, використовуючи метод **write ()**. Якщо файл було відкрито в режимі "a", дані будуть додані до кінця файлу. У формат, можливо, потрібно буде додати символи «\» для форматування.

Наприклад, \ n або \ r \ n символи додадуть розриви рядків у кінець записаного рядка даних. Метод файлу **read ()** читає вміст відкритого файлового об'єкта. Це показано на рис. 3.6.

```
myFile = open('newText.txt', "w")
myFile.close()

myFile = open('newText.txt', "a")
myFile.write('Line 1\n')
myFile.write('Line 2\n')
myFile.write('Line 3\n')

myFile.close()
myFile = open('newText.txt', "r")
myFile.read()
```

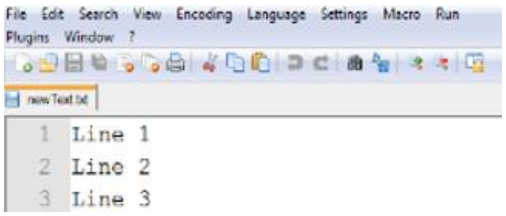

Рис. 3.6. Читання та запис даних у файл [3]

На рис. 3.6. у вхідній комірці один створюється новий файл. Файл закривається. У другій комірці файл знову відкривається в режимі додавання. У файл записуються три рядки тексту. У третій комірці файл закритий, щоб переконатися, що текст був записаний у файл. Файл знову відкривається в режимі читання, а метод **read ()** використовується для перегляду файлу. Файл також відображається у вигляді текстового редактора, який формує текст, використовуючи символи «\» для створення трьох окремих рядків.

3.3. Взаємодія із зовнішніми додатками

Python дозволяє взаємодіяти із зовнішніми додатками та операційною системою. Jupyter Notebook – це веб-програма з відкритим вихідним кодом, яка дозволяє створювати та обмінюватися документами, що містять діючий код, математичні рівняння, візуалізації та текст. Використання включає: очищення та

перетворення даних, чисельне моделювання, статистичне моделювання, візуалізація даних, машинне навчання та багато іншого [4].

У Jupyter Notebooks символ "!" дозволяє безпосередньо взаємодіяти з операційною системою. Наприклад, на рис. 3.7 показані дві команди Linux, виконані в Jupyter Notebooks. Команди починаються із символу "!".

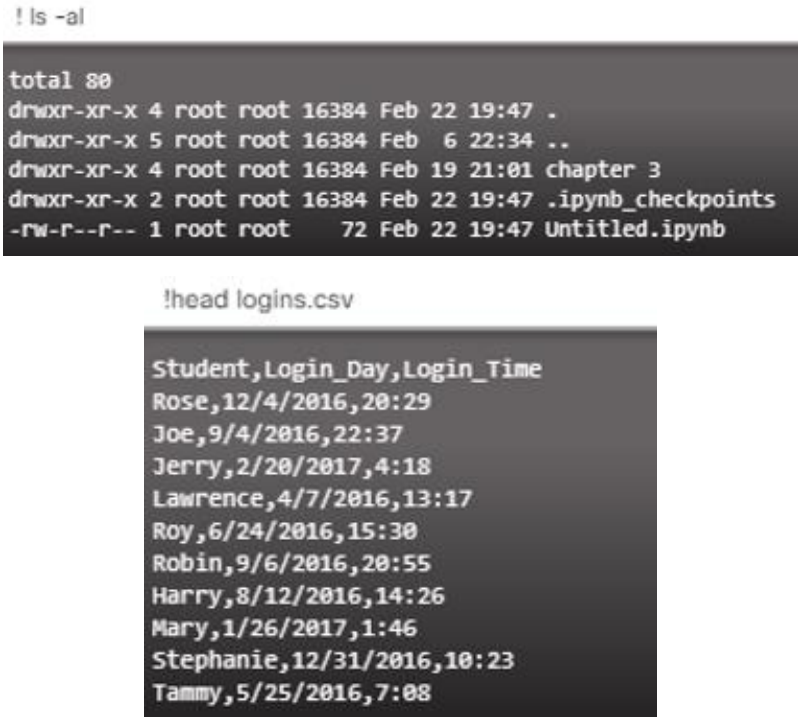

Рис. 3.7. Команди Linux, виконані в Jupyter Notebooks для взаємодії з операційною системою [3]

На рис. 3.8 показано використання модуля **subprocess** для зв'язку з зовнішньою програмою та збереження виводу команди, виданої цій програмі, в об'єкт Python. По-перше, створюється об'єкт для утримання команди, яку надсилає програма. У цьому випадку ми маємо намір надіслати команду до утиліти ping, яка доступна з оболонки Linux. Потім ми відправляємо цю команду, розділивши її на окремі слова, програмі методом **subprocess**. Нарешті, ми зберігаємо вихід команди в змінну і розділяємо її на рядок. Тоді ми можемо переглядати вміст об’єкта з друком та адресувати його окремі елементи за допомогою рядкової індексації.

```
'''import the subprocess library which is necessary for communication with external
apps'''
import subprocess
#We now execute the ping process as of from the shell:

pingCmd = 'ping -c 127.0.0.1'
process = subprocess.Popen(pingCmd.split(), stdout=subprocess.PIPE)
'''creat an object to hold the output of the process and split the output elements
into a list'''

process_output = process.communicate()[0]
process_output = process_output.split()
#view the contents of the output object
print(process_output)
```

```
['PING'. '127.0.0.1', '(127.0.0.1)', '56(84)', 'bytes', 'of', 'data.', '64', 'bytes',
 'from', '127..0.0.1', 'icmp_seq=1', 'ttl=64', 'time=0.094', 'ms', '64', 'bytes',
 'from', '127.0.0.1', icmp_seq=2', 'ttl=64', 'time=0.052', 'ms', '---', '127.0.0.1',
 'ping', 'statistics', '---', '2', 'packets', 'transmitted', '2', 'received,', '0%',
 'packet', 'loss,', 'time', '999ms', 'rtt', 'min/avg/max/mdev', '=',
 '0.052/0.073/0.094/0.021', 'ms']
```

```
#view the first five elements of the list
 process_output[0:5]
```

```
['PING', '127.0.0.1', '(127.0.0.1)', '56(84)', 'bytes']
```

Рис. 3.8. Використання модуля **subprocess** для зв'язку з зовнішньою програмою [3]

Висновок до лекції 3

Для оброблення даних про час та дату, використовується модуль Python **datetime**. Модуль **csv** дозволяє читати та записувати у .csv файли.

Python також має основні методи створення, відкриття та закриття зовнішніх файлів. Таблиці даних будуть існувати лише в оперативній пам'яті, поки вони не будуть збережені у файли. Метод **open ()** використовується для створення нового файлу або для відкриття наявного файлу, який буде містити дані, які потрібно зберегти. Функція **close ()** видаляє дані з буферів і закінчує функцію запису файлу для вказаного файлу. Python дозволяє взаємодіяти із зовнішніми додатками та операційною системою. Модуль **subprocess** використовується для зв'язку з зовнішньою програмою та збереження виводу команди, виданої цій програмі, в об'єкт Python.