

# Data Engineering

**Name: Ramireddy Preethi**

**Batch: Python Batch 2**

## Files:

Files are grouped sub scripted character arrays comprises of file.

### Types of Files in Python

- Text Files: store the data in the form of characters.
- Binary Files: store entire data in binary format i.e bytes (group of 8 bits). Binary files can be store text, images, audio and video.

### Opening a File

Reading and Writing Files in Python

#### Syntax:

```
file object = open(file_name [, access_mode][, buffering])
```

### buffering

```
In [74]: # Opening a text file with line buffering
with open(r"C:\Users\preethi\OneDrive\Desktop\example.txt", "w") as file:
    file.write("Hello, World!\n")
    file.write("This is line 2.\n")
```

```
In [75]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt", "w", buffering=4096) as file:
    file.write("this i preethi\n")
    file.write("testing how buffer works for value 2\n")
```

# Types of Modes

1) Read Mode (r) : Opens a file for reading only.

```
In [54]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("First Line:",f.readline(3)) # gives content line by line
          print("second Line:",f.readline())
          print("Third Line:",f.readline())
```

First Line: Hel  
second Line: lo, World!

Third Line: This is line 2.

```
In [37]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          for line in f:
              print(line)
```

Hello, World!

This is line 2.

```
In [34]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("Content of file:",f.readlines()) # gives list of all contents
```

Content of file: ['Hello, World!\n', 'This is line 2.\n']

```
In [34]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("Content of file:",f.readlines()) # gives list of all contents
```

Content of file: ['Hello, World!\n', 'This is line 2.\n']

```
In [41]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("Content of file:",f.readlines(10)) # gives list of contents of size upto 10 bytes
```

Content of file: ['Hello, World!\n']

```
In [47]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("Content of file:",f.readlines(14)) # gives list of contents of size upto 10 bytes
```

Content of file: ['Hello, World!\n', 'This is line 2.\n']

```
In [35]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("content:",f.read()) # prints entire content
```

content: Hello, World!  
This is line 2.

```
In [40]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
          print("content:",f.read(8)) # prints characters upto specified length
```

content: Hello, w

## 2) rb : Opens a file for reading only in binary format.

```
In [56]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","rb") as f:  
         print("content:",f.read())
```

content: b'Hello, World!\r\nThis is line 2.\r\n'

```
In [59]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","rb") as f:  
         print("content:",f.readline())
```

content: b'Hello, World!\r\n'

```
In [60]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","rb") as f:  
         print("content:",f.readlines())
```

content: [b'Hello, World!\r\n', b'This is line 2.\r\n']

```
In [64]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r+") as f:  
         print("content:",f.read())  
         f.write("writing using r+ mode\n")
```

content: Hello, World!

This is line 2.

writing using r+ mode

content:

---

```
In [67]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r+") as f:  
         print("content:",f.read())
```

content: Hello, World!

This is line 2.

writing using r+ mode

writing using r+ mode

---

```
In [73]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","rb+") as f:  
         print("content:",f.read())
```

content: b'Hello, World!\r\nThis is line 2.\r\nwriting using r+ mode\r\nwriting using r+ mode\r\n'

### 3) w,w+,wb,wb+

```
In [81]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","w") as f:
         f.write("this is preethi\n")
         f.write("working on write operations\n")
         # now after writing cursor will be at the end of file, so in order read the file
         f.seek(0)
         |
```

```
In [82]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt","r") as f:
         print(f.readline())
         print(f.readline())
```

this is preethi

working on write operations

```
In [83]: # w+
         with open("C:/Users/preethi/OneDrive/Desktop/example.txt","w") as f:
             f.write("this is preethi\n")
             f.write("working on write operations\n")
             # now after writing cursor will be at the end of file, so in order read the file
             f.seek(0)
             print(f.readline())
             print(f.readline())
```

this is preethi

working on write operations

```
In [86]: # wb
         with open("C:/Users/preethi/OneDrive/Desktop/example.txt","wb") as f:
             f.write(b"this is preethi\n") # it indicates we writing sequence of bytes
             f.write(b"working on write operations\n")
```

```
In [88]: # wb+
         with open("C:/Users/preethi/OneDrive/Desktop/example.txt","wb+") as f:
             f.write(b"this is preethi\n") # it indicates we writing sequence of bytes
             f.write(b"working on write operations\n")
             f.seek(0)
             print(f.readline())
             print(f.readline())
```

b'this is preethi\n'

b'working on write operations\n'

#### 4) a,ab,a+,ab+

```
In [94]: # a - Opens a file for appending. Adds data to the end of the file without overwriting existing content.
with open("C:/Users/preethi/OneDrive/Desktop/example.txt", "a") as file:
    file.write("Appending a new line.\n") # Appends a single line
    file.writelines(["Append line 2\n", "Append line 3\n"]) # Appends multiple lines
```

```
In [95]: with open("C:/Users/preethi/OneDrive/Desktop/example.txt", "a+") as f:
# Append new data to the file
f.write("Appending a new line.\n") # Appends a single line
f.writelines(["Append line 2\n", "Append line 3\n"]) # Appends multiple lines

# Move the file pointer to the beginning of the file for reading
f.seek(0)

# Read and print the file content
content = f.read()
print(content)
```

```
this is preethi
working on write operations
Appending a new line.
Append line 2
Append line 3
Appending a new line.
Append line 2
Append line 3
Appending a new line.
Append line 2
Append line 3
Appending a new line.
Append line 2
```

#### Closing a File

A file which is opened should be closed using the close () method. Once a file is opened but not closed, then the data of the file may be corrupted or deleted in some cases.

```
f.close();
```

## Modules and Packages

A module in Python is simply a file containing Python code, usually with a .py extension. Modules can include functions, classes, variables, and runnable code, making them reusable across different programs.

Python comes with many built-in modules (like math, os, and sys), and you can also create custom modules or install external ones via pip.

#### Use of Modules:

**Code Organization:** Modules allow you to break down complex code into smaller, manageable pieces, making your codebase cleaner and easier to maintain.

**Code Reusability:** By writing functions and classes in a module, you can reuse them across multiple scripts without duplication.

**Namespace Management:** Modules help avoid name conflicts, as each module has its own namespace. For example, math.sqrt and cmath.sqrt coexist without issues.

#### Types of Modules

- **Built-in Modules:** Part of the Python Standard Library (e.g., os, random, datetime), these modules come pre-installed with Python.

- **External Modules:** Not included with Python but can be installed via pip. Examples include requests for HTTP requests and pandas for data manipulation.
- **User-Defined Modules:** Modules created by the user. These are simply Python files that you create and import into other scripts.

## Creating and Using Custom Modules

- Create a module by saving a Python file with code you want to reuse, say mymodule.py.
- To use mymodule in another script, use the import statement:

**# mymodule.py**

```
def greet(name):
```

```
    return f'Hello, {name}!'
```

**# main.py**

```
import mymodule
```

```
print(mymodule.greet("Alice")) # Output: Hello, Alice!
```

## Importing Modules

- You can import a module in several ways:
  - **import module\_name:** Imports the entire module.
  - **import module\_name as alias:** Imports with an alias (e.g., **import numpy as np**).
  - **from module\_name import specific\_function:** Imports only specific parts (e.g., from math import sqrt).
  - **from module\_name import \*:** Imports everything from the module (not recommended for large modules as it can cause naming conflicts).

## Executing Modules as Scripts

- By adding the code **if \_\_name\_\_ == "\_\_main\_\_":** at the bottom of a module, you can make it executable as a script and reusable as a module. This block will only execute if the module is run directly (not imported).

**# example\_module.py**

```
def greet():
```

```
    print("Hello from the module!")
```

```
if __name__ == "__main__":
```

```
greet() # Runs only when executing this file directly
```

## Organizing Code into Packages

- A package is a collection of related modules organized in a directory with an `__init__.py` file (which can be empty).
- For example, a package `mypackage` with modules `module1` and `module2`:

`mypackage/`

|— `__init__.py`

|— `module1.py`

|— `module2.py`

To import from the package:

```
from mypackage import module1
```

## Common Standard Modules

- **os:** Interact with the operating system, like reading/writing files, changing directories.
- **sys:** Access system-specific parameters and functions.
- **math:** Perform mathematical operations.
- **datetime:** Work with dates and times.
- **json:** Parse and work with JSON data.
- **random:** Generate random numbers.

## Installing and Managing External Modules with pip

- **pip** is Python's package manager, used to install and manage external modules.