Python Programming

By Neelesh Biswas, Moderator for Python Programming in The Uplift Project by GirlScript Foundation (Team: py4-team4)

File Handling

What is File?

- Collection of data stored on a secondary storage device like hard disk.
- Basically used because real life applications involve large amounts of data
- **Reason to use file**: Large amounts of data are time-consuming & can be lost if either the program is terminated or computer is turned off while doing I/O terminal
- In order to use file, we have to learn file input & output operations that is how data is read or written to a file.

Volatile & Non-Volatile Storage

Volatile Storage

- Data storage is temporary
- Directly accessed by CPU
- Storage is limited
- Power dependent
- Memory is IC Based

E.g.: RAM

Non-Volatile Storage

- Data storage is permanent
- Cannot be accessed directly by CPU
- Large storage capacity
- Power independent
- Memory is magnetic or optical based.**E.g: Pen Drive, Hard disk.**

File Path

- A File path is a path in an address of a file which is stored in a hard disk of a computer.
- Every file is identified by its path that begins from the root folder or root node.
- In **Windows**, it is written as

C:\documents\Program Files\Neelesh.txt

- In **Linux**, it is written as:

/documents/Neelesh.txt

- **Note: The characters, / and \ , used to separate folder names are known as delimiters and are specific to file system.**

Relative & Absolute Path

- A file path can be *relative* or *absolute*.
- Absolute path always contains the root & complete directory list to specify the exact location of the file.
- Relative path needs to be combined with respect to another path in order to access a file.
- **If you use a relative path from the wrong directory, then either wrong file will be accessed or no file will be accessed if file doesn't exists in the specified path.**

Types of Files

Python Supports two types of files:

1) **ASCII Text File:** A text file is a stream of characters that can be sequentially processed in a forward direction. For this reason, a text file **(.txt)** is usually opened for only one kind of operation (read, write & append).
2) **Binary File:** Contain any type of data, encoded in binary form for computer storage & processing purposes. Includes Documents, PDFs, Spreadsheets, Images, Videos, etc.

File Operations : Opening a File [open()]

Syntax in python:

```
fileObj = open("[filename]", "[access modes]")
```

Main Three Types of Access Modes:

1) Read (r)
2) Write (w)
3) Append (a)

File Operations : Closing a File `[close()]`

Syntax in Python:

```
fileObj.close()
```

File Operations : `read()` & `readline()`

The read() & readlines() methods are used to read a string from an already opened file.

Syntax:

```
file=open("Neelesh.txt","r")
print(file.read())
file.close()
```

File Operations : `Write()`

The write() & writelines() methods are used to read a string from an already opened file.

Syntax:

```
file=open("Neelesh1.txt","w")
file.write("\nI'm Teaching Python")
file.close()
```

File Operations : `writelines()`

The write() & writelines() methods are used to read a string from an already opened file.

Syntax:

```
file=open("Neelesh3.txt","w")
lines=["Hello everyone,","\nI'm Neelesh"," Welcome to Python"]
file.writelines(lines)
file.close()
print("Data Written into the file")
```

File Operations : `append()`

append() is used to write & edit an existing file.

Syntax:

```
file=open("Neelesh3.txt","a")
file.write(" Hello People")
file.close()
```

<u>Note:</u> `append() use to edit an existing file whereas write()` Overwrites an existing file.

File Operations : `with` Keyword

When dealing with file objects, it is preferable to use with keyword as it ensures that the file is closed after the program is executed even if an error occured during read/write operations.

Syntax:

```
with open("Neelesh.txt","r") as file:

    line=file.read()

    print(line)

print("Status of the file:",file.closed)
```

Syntax:

```
file=open("Neelesh.txt","r")

print(file.read())

print("Status of the file:",file.closed)
```

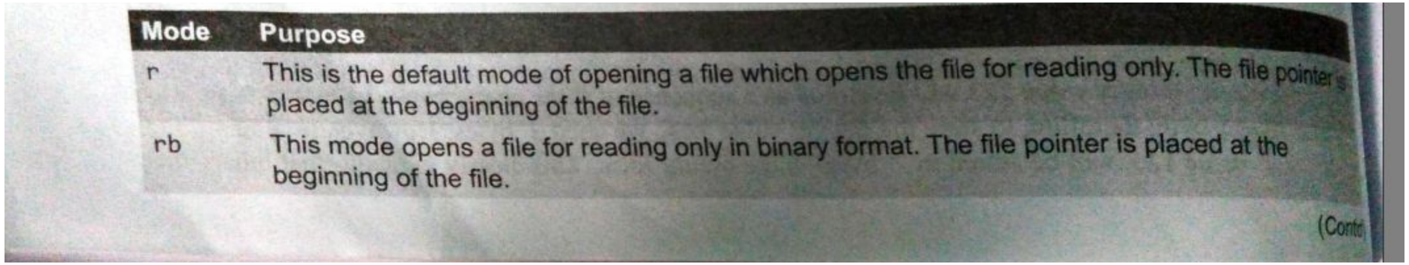
File Operations : `Splitting Words`

Python allows you to read lines from a file & splits the line based on a character. By default, this character is space but you can specify any other character to split the words to a string.

Syntax:

```
with open("Neelesh.txt","r") as file:
    line=file.readline()
    words=line.split()
    print(words)
```

More Access Modes



Mode	Purpose
r	This is the default mode of opening a file which opens the file for reading only. The file pointer is placed at the beginning of the file.
rb	This mode opens a file for reading only in binary format. The file pointer is placed at the beginning of the file.

(Contd

More Access Modes

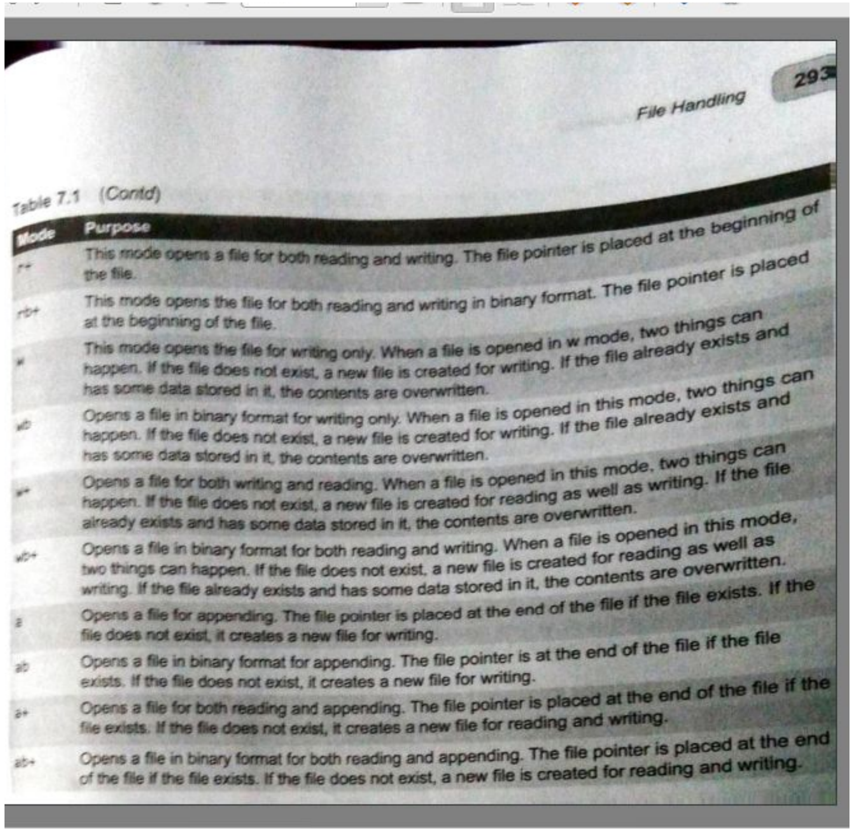


Table 7.1 (Contd)

Mode	Purpose
r+	This mode opens a file for both reading and writing. The file pointer is placed at the beginning of the file.
rb+	This mode opens the file for both reading and writing in binary format. The file pointer is placed at the beginning of the file.
w	This mode opens the file for writing only. When a file is opened in w mode, two things can happen. If the file does not exist, a new file is created for writing. If the file already exists and has some data stored in it, the contents are overwritten.
wb	Opens a file in binary format for writing only. When a file is opened in this mode, two things can happen. If the file does not exist, a new file is created for writing. If the file already exists and has some data stored in it, the contents are overwritten.
w+	Opens a file for both writing and reading. When a file is opened in this mode, two things can happen. If the file does not exist, a new file is created for reading as well as writing. If the file already exists and has some data stored in it, the contents are overwritten.
wb+	Opens a file in binary format for both reading and writing. When a file is opened in this mode, two things can happen. If the file does not exist, a new file is created for reading as well as writing. If the file already exists and has some data stored in it, the contents are overwritten.
a	Opens a file for appending. The file pointer is placed at the end of the file if the file exists. If the file does not exist, it creates a new file for writing.
ab	Opens a file in binary format for appending. The file pointer is at the end of the file if the file exists. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both reading and appending. The file pointer is placed at the end of the file if the file exists. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file in binary format for both reading and appending. The file pointer is placed at the end of the file if the file exists. If the file does not exist, a new file is created for reading and writing.