Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FILE HANDLING - 1

UNIT – III

File Handling: Introduction to Files, Types of Files, Opening and Closing a Text File, Writing data to a Text File, Reading data from a Text File, Setting offsets in a File and Illustrative examples on File I/O.

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. Introduction to Files
2. Types of Files
3. Opening and Closing a Text file.

1. INTRODUCTION

Need of Files concept:

As a part of programming requirement it is very common to store our data permanently.

Different types of requirements of data storage:

1. **Very small amount of Information:**
    - 100 students mobile numbers
2. **Huge amount of Data:**
    - SBI Customers transactions information
3. **Very Very Huge amount of Data:**
    - Videos in YouTube
    - Postings in Facebook
    - Tweets in twitter

There are multiple storage areas are there to store our data, based on our requirement, we need to choose corresponding storage area.

- To store very small amount of information, most common way of storage area is **Files concept**.
- To store very huge amount of information, most common way of storage area is **Database concept.**
- To store very very huge amount of information, most common way of storage area is **Big data concept.**

As of now we want to store very small amount of information, then obviously we should go for Files concept.

- If our data is limited and to store permanently, we should go for files concept.
- The data stored inside a python object is temporary.

While the program is executing, python objects holds data. Once program execution is completed, then automatically data won't available. This type of storage area are nothing but temporary storage areas.

Introduction To File handling In Python

- We have so far created programs in Python that accept the input, manipulate it and display the output.
- But that output is available only during execution of the program and input is to be entered through the keyboard.
- This is because, the variables used in a program have a lifetime that lasts till the time the program is under execution.

- What if we want to store the data that were input as well as the generated output permanently so that we can reuse it later?
- Usually, organisations would want to permanently store information about employees, inventory, sales, etc. to avoid repetitive tasks of entering the same data. Hence, data are stored permanently on secondary storage devices for reusability.
- We store Python programs written in script mode with a **.py** extension. Each program is stored on the secondary device as a file. Likewise, the data entered, and the output can be stored permanently into a file.

So, What is a file?

- A file is a "*named location on a secondary storage media where data are permanently stored for later access*".

2. Types of files:

- Computers store every file as a collection of 0s and 1s i.e., in binary form. Therefore, every file is basically just a series of bytes stored one after the other.
- There are mainly two types of data files — text file and binary file.
- A text file consists of human readable characters, which can be opened by any text editor.
- On the other hand, binary files are made up of non-human readable characters and symbols, which require specific programs to access its contents.

2.1 Text file

- A text file can be understood as a sequence of characters consisting of alphabets, numbers and other special symbols.
- Files with extensions like **.txt**, **.py**, **.csv**, etc. are some examples of text files.
- When we open a text file using a text editor (e.g., Notepad), we see several lines of text. However, the file contents are not stored in such a way internally. Rather, they are stored in sequence of bytes consisting of 0's and 1's.
- In ASCII, UNICODE or any other encoding scheme, the value of each character of the text file is stored as bytes.

So, while opening a text file, the text editor translates each ASCII value and shows us the equivalent character that is readable by the human being.

For example, the ASCII value 65 (binary equivalent 1000001) will be displayed by a text editor as the letter ‘A’ since the number 65 in ASCII character set represents ‘A’.

- Each line of a text file is terminated by a special character, called the End of Line (EOL). For example, the default EOL character in Python is the newline (\n).
- When a text editor or a program interpreter encounters the ASCII equivalent of the EOL character, it displays the remaining file contents starting from a new line.
- Contents in a text file are usually separated by whitespace, but comma (,) and tab (\t) are also commonly used to separate values in a text file.

2.2 Binary Files

- Binary files are also stored in terms of bytes (0s and 1s), but unlike text files, these bytes do not represent the ASCII values of characters. Rather, they represent the actual content such as image, audio, video, compressed versions of other files, executable files, etc.
- These files are not human readable. Thus, trying to open a binary file using a text editor will show some garbage values. We need specific software to read or write the contents of a binary file.

- Binary files are stored in a computer in a sequence of bytes. Even a single bit change can corrupt the file and make it unreadable to the supporting application.
- Also, it is difficult to remove any error which may occur in the binary file as the stored contents are not human readable. We can read and write both text and binary files through Python programs.

Text File	Binary File
Its Bits represent character.	Its Bits represent a custom data.
Less prone to get corrupt as change reflects as soon as made and can be undone.	Can easily get corrupted, corrupt on even single bit change
Store only plain text in a file.	Can store different types of data (audio, text,image) in a single file.
Widely used file format and can be opened in any text editor.	Developed for an application and can be opened in that application only.
Mostly .txt and .rtf are used as extensions to text files.	Can have any application defined extension.

2.3 Opening and Closing a Text File

- In real world applications, computer programs deal with data coming from different sources like databases, CSV files, HTML, XML, JSON, etc.
- We broadly access files either to write or read data from it.
- But operations on files include creating and opening a file, writing data in a file, traversing a file, reading data from a file and so on.
- Python has the **io module** that contains different functions for handling files.

2.3.1 Opening a file

- To open a file in Python, we use the open() function. The syntax of **open()** is as follows:

file_object= open(file_name, access_mode)

- This function returns a **file object** called **file handle** which is stored in the variable **file_object.** We can use this variable to transfer data to and from the file (read and write) by calling the functions defined in the Python's **io module**.
- If the file does not exist, the above statement creates a new empty file and assigns it the name we specify in the statement.

Note:

- The **file_object** establishes a link between the program and the data file stored in the permanent storage.

The **file_object** has certain attributes that tells us basic information about the file, such as:

- **<file_object.closed>** returns true if the file is closed and false otherwise.
- **<file_object.mode>** returns the access mode in which the file was opened.
- **<file_object.name>** returns the name of the file.

1. The file_name should be the name of the file that has to be opened. If the file is not in the current working directory, then we need to specify the complete path of the file along with its name.

2. The access_mode is an optional argument that represents the mode in which the file has to be accessed by the program. It is also referred to as **processing mode**.

Here mode means the operation for which the file has to be opened like **<r>** for reading, **<w>** for writing, **<r+>** for both reading and writing, **<a>** for appending at the end of an existing file.

Note: The default is the read mode.

Table 2.1 File Open Modes

File Mode	Description	File Offset position
<r>	Opens the file in read-only mode.	Beginning of the file
<rb>	Opens the file in binary and read-only mode.	Beginning of the file
<r+> or <+r>	Opens the file in both read and write mode.	Beginning of the file
<w>	Opens the file in write mode. If the file already exists, all the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<wb+> or <+wb>	Opens the file in read,write and binary mode. If the file already exists, the contents will be overwritten. If the file doesn't exist, then a new file will be created.	Beginning of the file
<a>	Opens the file in append mode. If the file doesn't exist, then a new file will be created.	End of the file
<a+> or <+a>	Opens the file in append and read mode. If the file doesn't exist, then it will create a new file.	End of the file

Consider the following example.

myObject=open(“myfile.txt”, “a+”)

- In the above statement, the file ***myfile.txt*** is opened in append and read modes. The file object will be at the end of the file.
- That means we can write data at the end of the file and at the same time we can also read data from the file using the file object named ***myObject.***

What about file location?

- If the file is available in the current working directory, then we can specify directly by specifying the file.
- If the file is available some where else, we need to use complete path.

```
f = open('D:\abc.txt','r')
```

More information about modes:

The allowed modes in Python are as follows:

1. read (r) :

- Open an existing file for read operation.
- Default type of mode is read('r') only.
- The file pointer is positioned at the beginning of the file.
- If the specified file does not exist then we will get FileNotFoundError.

2. write (w) :

- Open an existing file for write operation.
- If the file already contains some data then it will be overridden.
- If the specified file is not already available then this mode will create that file.

3. append (a):

- Open an existing file for append operation.
- It won't override existing data.
- If the specified file is not already available then this mode will create a new file.

4. read+ (r+):

- To read and write data into the file.
- The previous data in the file will not be deleted.
- The file pointer is placed at the beginning of the file.

Q. In r+ case, We can get only data what we have written in second iteration!

What will happen to the first data?

==> In r+ mode, first file pointer points to the beginning of the file, so we can successfully read the existing data. After that, if you want, you can perform write operation with the new data.

5. write+ (w+):

- To write and read data.
- It will override existing data.

6.append+ (a+):

- To append and read data from the file.
- It won’t override existing data.

7. exclusive (x):

- To open a file in exclusive mode for write operation. [don't use Existing files]

What is the difference between 'w' and 'x'?

- To use exclusive mode, the specified file should not be already available.
- It should create a new file and write the data.
- If the file already exists then we will get FileExistsError.

Note:

- All the above modes are applicable for text files. If the above modes suffixed with 'b' then these represent for binary files.

Eg:

rb,wb,ab,r+b,w+b,a+b,xb

2.3.2 Closing a file

- Once we are done with the read/write operations on a file, it is a good practice to close the file.
- Python provides a close() method to do so. While closing a file, the system frees the memory allocated to it.

The syntax of close() function is:

file_object.close()

Here, **file_object** is the object that was returned while opening the file.

- Python makes sure that any unwritten or unsaved data is flushed off (written) to the file before it is closed.
- Hence, it is always advised to close the file once our work is done. Also, if the file object is re-assigned to some other file, the previous file is automatically closed.

Various properties of File Object:

Once we opened a file and we got file object ,we can get various details related to that file by using it's properties.

i) name

- Name of opened file

ii) mode

- Mode in which the file is opened

iii) closed

- Returns boolean value indicates that file is closed or not

iv) readable()

- Returns boolean value indicates that whether file is readable or not

v) writable()

- Returns boolean value indicates that whether file is writable or not.

Eg 1:

```
pwd          # Present Working Directory
```

```
Out[1]:

'C:\\Users\\HP\\Desktop\\14. File Handling'
```

Eg 2:

```
f=open("abc.txt",'w')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  w
Is File Readable:  False
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Eg 3:

```
f=open("abc.txt")
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  r
Is File Readable:  True
Is File Writable:  False
Is File Closed :  False
Is File Closed :  True
```

Eg 4:

```
f=open("abc.txt",'a')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  a
Is File Readable:  False
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Eg 5:

```
f=open("abc.txt",'r+')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  r+
Is File Readable:  True
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Eg 6:

```
f=open("abc.txt",'w+')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  w+
Is File Readable:  True
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Eg 7:

```
f=open("abc.txt",'a+')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abc.txt
File Mode:  a+
Is File Readable:  True
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Eg 8:

```
f=open("abc.txt",'x')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
<ipython-input-6-71c0e06e2086> in <module>
----> 1 f=open("abc.txt",'x')
      2 print("File Name: ",f.name)
      3 print("File Mode: ",f.mode)
      4 print("Is File Readable: ",f.readable())
      5 print("Is File Writable: ",f.writable())

FileExistsError: [Errno 17] File exists: 'abc.txt'
```

Eg 9:

```
f=open("abcdef.txt",'x')
print("File Name: ",f.name)
print("File Mode: ",f.mode)
print("Is File Readable: ",f.readable())
print("Is File Writable: ",f.writable())
print("Is File Closed : ",f.closed)
f.close()
print("Is File Closed : ",f.closed)
```

```
File Name:  abcdef.txt
File Mode:  x
Is File Readable:  False
Is File Writable:  True
Is File Closed :  False
Is File Closed :  True
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You