Python Files I/O

Python2.unit-6

What is a file?

- File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g. hard disk).
- Since, random access memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.
- When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order.

- **The *open()* Method**
- **The *close()* Method**
- **The *write()* Method**
- **The *read()* Method**
- **The *append()* Method**
- **The rename() Method**
- **The *delete()* Method**
- **The *file* object atrributes**

Opening and Closing Files:

- Until now, you have been reading and writing to the standard input and output. Now we will see how to play with actual data files.
- Python provides basic functions and methods necessary to manipulate files by default. You can do your most of the file manipulation using a **file** object.

The *open* Method:

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object which would be utilized to call other support methods associated with it.

- **Syntax:**

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

Paramters detail:

- **file_name:** The file_name argument is a string value that contains the name of the file that you want to access.
- **access_mode:** The access_mode determines the mode in which the file has to be opened ie. read, write append etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r) .

Example:

```
text_file=open("text.txt",'w')

text_file.write("Helle Hai")

text_file.close()
```

Output:

Check your Desktop for your file i.e text.txt

The *close()* Method:

The close() method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done.

Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the close() method to close a file.

- **Syntax:**

  fileObject.close();

Example:

```
text_file=open("text.txt",'w')

text_file.write("Helle Hai")

text_file.close()
```

The *write()* Method:

- The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.
- The write() method does not add a newline character ('\n') to the end of the string:

Syntax:

fileObject.write(string);

Example:

```
text_file=open("text.txt",'w')

text_file.write("Helle Hai")

text_file.close()

text_file=open("text.txt",'r')

print(text_file.read())

text_file.close()
```

Output:

Helle Hai

The *read()* Method:

The *read()* method read a string from an open file. It is important to note that Python strings can have binary data and not just text.

Syntax:

```
fileObject.read([count]);
```

Here passed parameter is the number of bytes to be read from the opend file. This method starts reading from the beginning of the file and if *count* is missing then it tries to read as much as possible, may be until the end of file.

Example:

```
text_file=open("text.txt",'w')

text_file.write("Helle Hai")

text_file.close()

text_file=open("text.txt",'r')

print(text_file.read())

text_file.close()
```

Output:

```
Helle Hai
```

The *append()* Method:

'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.

Example:

```
text_file=open("text.txt",'w')

text_file.write("Helle Hai")

text_file.close()

text_file=open("text.txt",'a')

text_file.write("Welcome to Python File Operations")

text_file.close()

text_file=open("text.txt",'r')

print(text_file.read())
```

```
text_file.close()
```

Output:

```
Helle HaiWelcome to Python File Operations
```

The rename() Method:

The *rename()* method takes two arguments, the current filename and the new filename.

Syntax:

```
os.rename(current_file_name, new_file_name)
```

Example:

```
import os
os.rename( "test1.txt", "test2.txt" )
```

The *delete()* Method:

You can use the *delete()* method to delete files by supplying the name of the file to be deleted as the argument.

Syntax:

```
os.remove(file_name)
```

Example:

```
import os
os.remove("test2.txt")
```

The *file* object atrributes:

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

Attribute	**Description**
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.
file.softspace	Returns false if space explicitly required with print, true otherwise.

Prepared by
Ramesh Yajjala
Assistant Professor(c)
Department of Computer Science and Engineering
IIIT-Srikakulam, RGUKT-AP