

Opening and Closing Files in Python

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the **file** manipulation using a file object.

The open Function

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])
```

Here are parameter details –

- **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, i.e., 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).
- **buffering** – If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

Here is a list of the different modes of opening a file –

Sr.No	Modes & Description
1	r Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
2	rb Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
3	r+ Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
4	rb+ Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.

Sr.No	Modes & Description
5	w Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
6	wb Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
7	w+ Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
8	wb+ Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
9	a Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
10	ab Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
11	a+ Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
12	ab+ Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

The file Object Attributes

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 –

Sr.No	Modes & Description
1	file.closed Returns true if file is closed, false otherwise.
2	file.mode Returns access mode with which file was opened.
3	file.name Returns name of the file.
4	file.softspace Returns false if space explicitly required with print, true otherwise.

Example

```
#!/usr/bin/python

# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace
```

Output

This produces the following result –

```
Name of the file: foo.txt
Closed or not : False
Opening mode : wb
Softspace flag : 0
```

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

```
#!/usr/bin/python
# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
# Close opened file
fo.close()
```

Output

This produces the following result –

```
Name of the file: foo.txt
```