**Python File Handling Operations**

**Most importantly there are 4 types of operations that can be handled by Python on files:**

* Open
* Read
* Write
* Close

**Other operations include:**

* Rename
* Delete

Python Create and Open a File

Python has an in-built function called open() to open a file.

It takes a minimum of one argument as mentioned in the below syntax. The open method returns a file object which is used to access the write, read and other in-built methods.

**Syntax:**

file\_object = open(file\_name, mode)

Here, file\_name is the name of the file or the location of the file that you want to open, and file\_name should have the file extension included as well. Which means in **test.txt** – the term test is the name of the file and .txt is the extension of the file.

The mode in the open function syntax will tell Python as what operation you want to do on a file.

* **‘r’ – Read Mode:** Read mode is used only to read data from the file.
* **‘w’ – Write Mode:** This mode is used when you want to write data into the file or modify it. Remember write mode overwrites the data present in the file.
* **‘a’ – Append Mode:** Append mode is used to append data to the file. Remember data will be appended at the end of the file pointer.
* **‘r+’ – Read or Write Mode:** This mode is used when we want to write or read the data from the same file.
* **‘a+’ – Append or Read Mode:** This mode is used when we want to read data from the file or append the data into the same file.

**Note:** The above-mentioned modes are for opening, reading or writing text files only.

While using binary files, we have to use the same modes with the letter **‘b’** at the end. So that Python can understand that we are interacting with binary files.

* **‘wb’ –**Open a file for write only mode in the binary format.
* **‘rb’ –**Open a file for the read-only mode in the binary format.
* **‘ab’ –**Open a file for appending only mode in the binary format.
* **‘rb+’ –**Open a file for read and write only mode in the binary format.
* **‘ab+’ –**Open a file for appending and read-only mode in the binary format.

**Example 1:**

|  |
| --- |
| fo = open(“C:/Documents/Python/test.txt”, “r+”) |

In the above example, we are opening the file named ‘test.txt’ present at the location ‘C:/Documents/Python/’ and we are opening the same file in a read-write mode which gives us more flexibility.

**Example 2:**

|  |
| --- |
| fo = open(“C:/Documents/Python/img.bmp”, “rb+”) |

In the above example, we are opening the file named ‘img.bmp’ present at the location “C:/Documents/Python/”, But, here we are trying to open the binary file.

Python Read From File

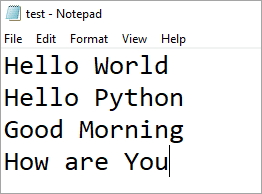
In order to read a file in python, we must open the file in read mode.

**There are three ways in which we can read the files in python.**

* read([n])
* readline([n])
* readlines()

Here, n is the number of bytes to be read.

First, let’s create a sample text file as shown below.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Sample_text_file.png)

**Now let’s observe what each read method does:**

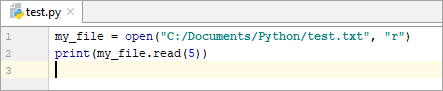
**Example 1:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.read(5)) |

**Output:**

Hello

Here we are opening the file test.txt in a read-only mode and are reading only the first 5 characters of the file using the my\_file.read(5) method.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_File_using_read_method.png)

**Output:**

[output - Reading File using my_file.read(5) method](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_File_using_read_method_output.png)

**Example 2:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.read()) |

**Output:**

Hello World  
Hello Python  
Good Morning

Here we have not provided any argument inside the read() function. Hence it will read all the content present inside the file.

**Output:**

**Example 3:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.readline(2)) |

**Output:**

He

This function returns the first 2 characters of the next line.

**Output:**

**Example 4:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.readline()) |

**Output:**

Hello World

Using this function we can read the content of the file on a line by line basis.

**Output:**

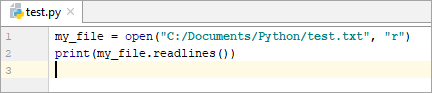
**Example 5:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.readlines()) |

**Output:**

[‘Hello World\n’, ‘Hello Python\n’, ‘Good Morning’]

Here we are reading all the lines present inside the text file including the newline characters.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_File_using_readlines_method.png)

**Output:**

[Reading_File_using_readlines()_method_output](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_File_using_readlines_method_output.png)

Now let’s see some more practical examples of reading a file.

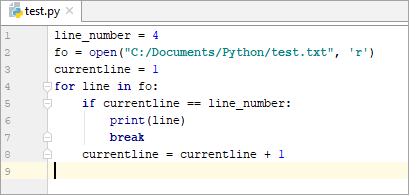
**Reading a specific line from a File**

|  |
| --- |
| line\_number = 4  fo = open(“C:/Documents/Python/test.txt”, ’r’)  currentline = 1  for line in fo:             if(currentline == line\_number):                         print(line)                         break            currentline = currentline +1 |

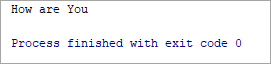
**Output:**

How are You

In the above example, we are trying to read only the 4th line from the ‘test.txt’ file using a **“for loop”**.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_specific_line_from_a_file.png)

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Reading_specific_line_from_a_file_output.png)

**Reading the entire file at once**

|  |
| --- |
| filename = “C:/Documents/Python/test.txt”  filehandle = open(filename, ‘r’)  filedata = filehandle.read()  print(filedata) |

**Output:**

Hello World  
Hello Python  
Good Morning  
How are You

**Output:**

Python Write to File

In order to write data into a file, we must open the file in write mode.

We need to be very careful while writing data into the file as it overwrites the content present inside the file that you are writing, and all the previous data will be erased.

**We have two methods for writing data into a file as shown below.**

* write(string)
* writelines(list)

**Example 1:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “w”)  my\_file.write(“Hello World”) |

The above code writes the String ‘Hello World’ into the ‘test.txt’ file.

**Before writing data to a test.txt file:**

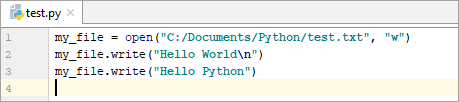
**Output:**

**Example 2:**

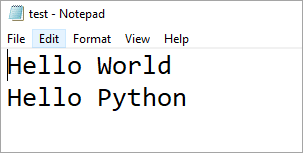
|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “w”)  my\_file.write(“Hello World\n”)  my\_file.write(“Hello Python”) |

The first line will be ‘Hello World’ and as we have mentioned \n character, the cursor will move to the next line of the file and then write ‘Hello Python’.

Remember if we don’t mention \n character, then the data will be written continuously in the text file like ‘Hello WorldHelloPython’

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Writing_data_to_file_using_write_method_with_next_line.png)

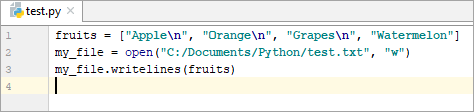
**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Writing_data_to_file_using_write_method_with_next_line_output.png)

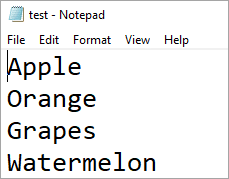
**Example 3:**

|  |
| --- |
| fruits = [“Apple\n”, “Orange\n”, “Grapes\n”, “Watermelon”]  my\_file = open(“C:/Documents/Python/test.txt”, “w”)  my\_file.writelines(fruits) |

The above code writes a **list of data** into the ‘test.txt’ file simultaneously.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Writing_List_of_data_to_file_using_writelines_method.png)

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Writing_List_of_data_to_file_using_writelines_method_output.png)

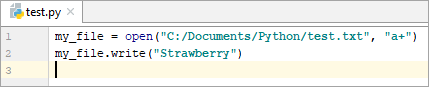
Python Append to File

To append data into a file we must open the file in ‘a+’ mode so that we will have access to both the append as well as write modes.

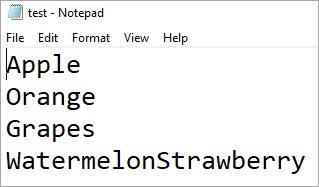
**Example 1:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “a+”)  my\_file.write (“Strawberry”) |

The above code appends the string ‘Apple’ at the **end** of the ‘test.txt’ file.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_1.png)

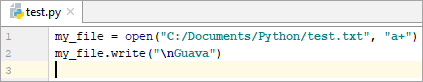
**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_1_output.png)

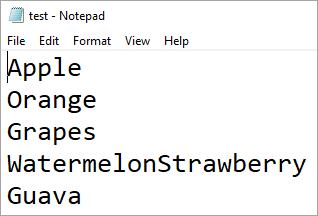
**Example 2:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “a+”)  my\_file.write (“\nGuava”) |

The above code appends the string ‘Apple’ at the end of the ‘test.txt’ file **in a new line**.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_2.png)

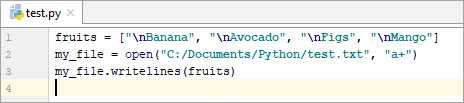
**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_2_output.png)

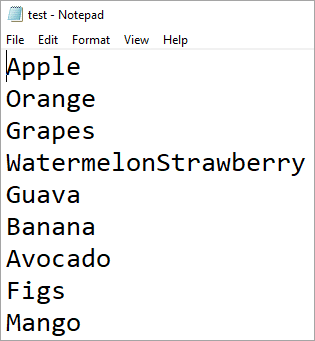
**Example 3:**

|  |
| --- |
| fruits = [“\nBanana”, “\nAvocado”, “\nFigs”, “\nMango”]  my\_file = open(“C:/Documents/Python/test.txt”, “a+”)  my\_file.writelines(fruits) |

The above code **appends a list of data** into a ‘test.txt’ file.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_3.png)

**Output:**

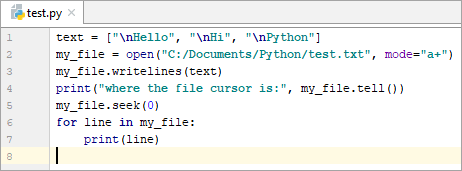
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_3_output.png)

**Example 4:**

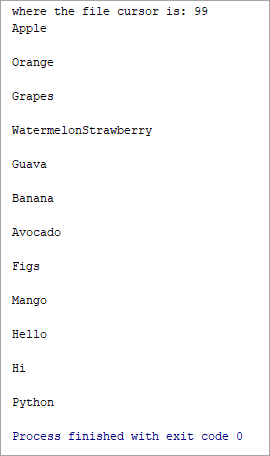
|  |
| --- |
| text=[&quot;\nHello&quot;,&quot;\nHi&quot;,&quot;\nPython&quot;]  my\_file=open(&quot;C:/Documents/Python/test.txt&quot;,mode=&quot;a+&quot;)  my\_file.writelines(text)  print(&quot;where the file cursor is:&quot;,my\_file.tell())  my\_file.seek(0)  for line in my\_file:        print(line) |

In the above code, we are appending the list of data into the ‘test.txt’ file. Here, you can observe that we have used the tell() method which prints where the cursor is currently at.

**seek(offset):** The offset takes three types of arguments namely 0,1 and 2.  
**When the offset is 0:** Reference will be pointed at the beginning of the file.  
**When the offset is 1:** Reference will be pointed at the current cursor position.  
**When the offset is 2:** Reference will be pointed at the end of the file.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_4.png)

**Output:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Appending_data_to_file_Example_4_output.png)

Python Close File

In order to close a file, we must first open the file. In python, we have an in-built method called close() to close the file which is opened.

Whenever you open a file, it is important to close it, especially, with write method. Because if we don’t call the close function after the write method then whatever data we have written to a file will not be saved into the file.

**Example 1:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “r”)  print(my\_file.read())  my\_file.close() |

**Example 2:**

|  |
| --- |
| my\_file = open(“C:/Documents/Python/test.txt”, “w”)  my\_file.write(“Hello World”)  my\_file.close() |

Python Rename or Delete File

Python provides us with an “os” module which has some in-built methods that would help us in performing the file operations such as renaming and deleting the file.

In order to use this module, first of all, we need to import the “os” module in our program and then call the related methods.

**rename() method:**

This rename() method accepts two arguments i.e. the current file name and the new file name.

**Syntax:**

os.rename(current\_file\_name, new\_file\_name)

**Example 1:**

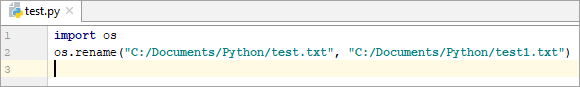
|  |
| --- |
| import os  os.rename(“test.txt”, “test1.txt”) |

Here ‘test.txt’ is the current file name and ‘test1.txt’ is the new file name.

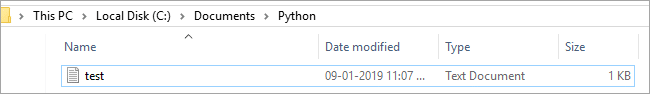
You can specify the location as well as shown in the below example.

**Example 2:**

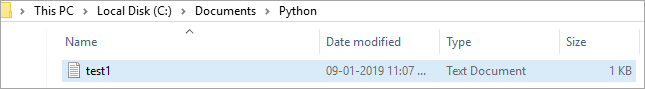
|  |
| --- |
| import os  os.rename(“C:/Documents/Python/test.txt”, “C:/Documents/Python/test1.txt”) |

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Renaming_file.png)

**Before Renaming the file:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/Before_renaming_the_file.png)

**After executing the above program**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2019/01/After_renaming_the_file.png)

**remove() method:**

We use the remove() method to delete the file by supplying the file name or the file location that you want to delete.

**Syntax:**

os.remove(file\_name)

**Example 1:**

|  |
| --- |
| import os  os.remove(“test.txt”) |

Here ‘test.txt’ is the file that you want to remove.

Similarly, we can pass the file location as well to the arguments as shown in the below example

**Example 2:**

|  |
| --- |
| import os  os.remove(“C:/Documents/Python/test.txt”) |