# Python File Methods

Python has a set of methods available for the file object.

|  |  |
| --- | --- |
| Method | Description |
| close() | Closes the file |
| detach() | Returns the separated raw stream from the buffer |
| fileno() | Returns a number that represents the stream, from the operating system's perspective |
| flush() | Flushes the internal buffer |
| isatty() | Returns whether the file stream is interactive or not |
| read() | Returns the file content |
| readable() | Returns whether the file stream can be read or not |
| readline() | Returns one line from the file |
| readlines() | Returns a list of lines from the file |
| seek() | Change the file position |
| seekable() | Returns whether the file allows us to change the file position |
| tell() | Returns the current file position |
| truncate() | Resizes the file to a specified size |
| writable() | Returns whether the file can be written to or not |
| write() | Writes the specified string to the file |
| writelines() | Writes a list of strings to the file |

# Python File Open

File handling is an important part of any web application.

Python has several functions for creating, reading, updating, and deleting files.

## File Handling

The key function for working with files in Python is the open() function.

The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file:

"r" - Read - Default value. Opens a file for reading, error if the file does not exist

"a" - Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist

"x" - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

"t" - Text - Default value. Text mode

"b" - Binary - Binary mode (e.g. images)

## Syntax

To open a file for reading it is enough to specify the name of the file:

f = open("demofile.txt")

The code above is the same as:

f = open("demofile.txt", "rt")

Because "r" for read, and "t" for text are the default values, you do not need to specify them.

Note: Make sure the file exists, or else you will get an error.

# Python File Open

## Open a File on the Server

Assume we have the following file, located in the same folder as Python:

demofile.txt

Hello! Welcome to demofile.txt  
This file is for testing purposes.  
Good Luck!

To open the file, use the built-in open() function.

The open() function returns a file object, which has a read() method for reading the content of the file:

### Example

f = open("demofile.txt", "r")  
print(f.read())

If the file is located in a different location, you will have to specify the file path, like this:

### Example

Open a file on a different location:

f = open("D:\\myfiles\welcome.txt", "r")  
print(f.read())

## Read Only Parts of the File

By default the read() method returns the whole text, but you can also specify how many characters you want to return:

### Example

Return the 5 first characters of the file:

f = open("demofile.txt", "r")  
print(f.read(5))

## Read Lines

You can return one line by using the readline() method:

### Example

Read one line of the file:

f = open("demofile.txt", "r")  
print(f.readline())

By calling readline() two times, you can read the two first lines:

### Example

Read two lines of the file:

f = open("demofile.txt", "r")  
print(f.readline())  
print(f.readline())

By looping through the lines of the file, you can read the whole file, line by line:

### Example

Loop through the file line by line:

f = open("demofile.txt", "r")  
for x in f:  
  print(x)

## Close Files

It is a good practice to always close the file when you are done with it.

### Example

Close the file when you are finish with it:

f = open("demofile.txt", "r")  
print(f.readline())  
f.close()

Note: You should always close your files, in some cases, due to buffering, changes made to a file may not show until you close the file.

# Python File Write

## Write to an Existing File

To write to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

### Example

Open the file "demofile2.txt" and append content to the file:

f = open("demofile2.txt", "a")  
f.write("Now the file has more content!")  
f.close()  
  
#open and read the file after the appending:  
f = open("demofile2.txt", "r")  
print(f.read())

### Example

Open the file "demofile3.txt" and overwrite the content:

f = open("demofile3.txt", "w")  
f.write("Woops! I have deleted the content!")  
f.close()  
  
#open and read the file after the appending:  
f = open("demofile3.txt", "r")  
print(f.read())

Note: the "w" method will overwrite the entire file.

## Create a New File

To create a new file in Python, use the open() method, with one of the following parameters:

"x" - Create - will create a file, returns an error if the file exist

"a" - Append - will create a file if the specified file does not exist

"w" - Write - will create a file if the specified file does not exist

### Example

Create a file called "myfile.txt":

f = open("myfile.txt", "x")

Result: a new empty file is created!

### Example

Create a new file if it does not exist:

f = open("myfile.txt", "w")

# Python Delete File

## Delete a File

To delete a file, you must import the OS module, and run its os.remove() function:

### Example

Remove the file "demofile.txt":

import os  
os.remove("demofile.txt")

## Check if File exist:

To avoid getting an error, you might want to check if the file exists before you try to delete it:

### Example

Check if file exists, then delete it:

import os  
if os.path.exists("demofile.txt"):  
  os.remove("demofile.txt")  
else:  
  print("The file does not exist")

## Delete Folder

To delete an entire folder, use the os.rmdir() method:

### Example

Remove the folder "myfolder":

import os  
os.rmdir("myfolder")

Note: You can only remove empty folders.