## Files

Files are named locations on disk to store related information. They are 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 the computer is turned off), we use files for future use of the data by permanently storing them.

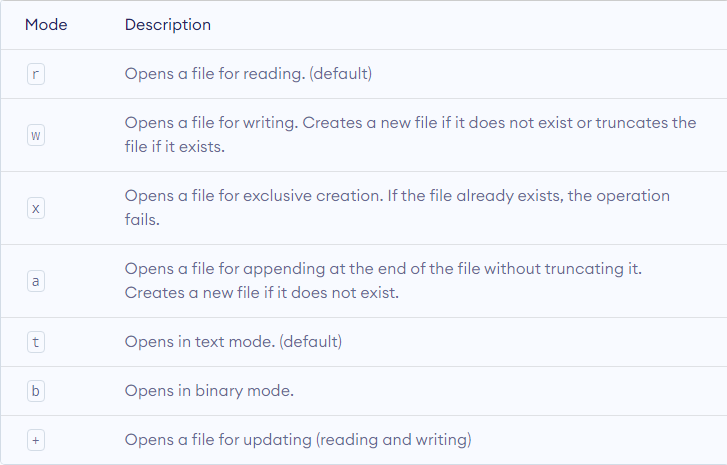
When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed so that the resources that are tied with the file are freed.

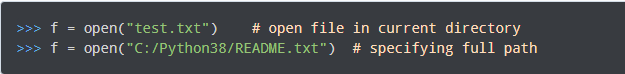
Hence, in Python, a file operation takes place in the following order:

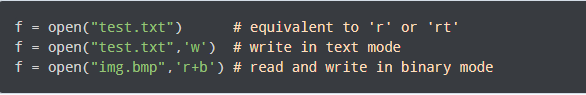
1. Open a file
2. Read or write (perform operation)
3. Close the file

## Opening Files in Python

Python has a built-in open() function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.







Unlike other languages, the character a does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings).

Moreover, the default encoding is platform dependent. In windows, it is cp1252 but utf-8 in Linux.

So, we must not also rely on the default encoding or else our code will behave differently in different platforms.

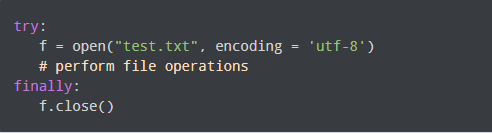
Hence, when working with files in text mode, it is highly recommended to specify the encoding type.



## Closing Files in Python

Closing a file will free up the resources that were tied with the file. It is done using the close() method available in Python.

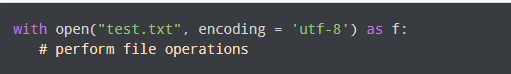
Python has a garbage collector to clean up unreferenced objects but we must not rely on it to close the file.



This way, we are guaranteeing that the file is properly closed even if an exception is raised that causes program flow to stop.

The best way to close a file is by using the with statement. This ensures that the file is closed when the block inside the with statement is exited.

We don't need to explicitly call the close() method. It is done internally.

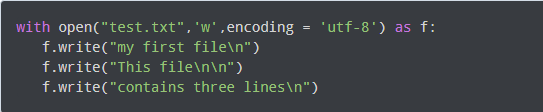


## Writing to Files in Python

In order to write into a file in Python, we need to open it in write w, append a or exclusive creation x mode.

We need to be careful with the w mode, as it will overwrite into the file if it already exists. Due to this, all the previous data are erased.

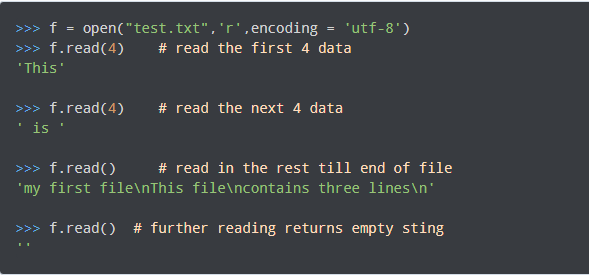
Writing a string or sequence of bytes (for binary files) is done using the write() method. This method returns the number of characters written to the file.



## Reading Files in Python

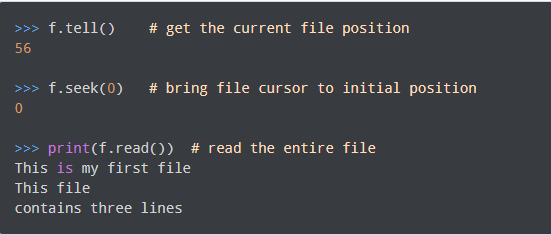
To read a file in Python, we must open the file in reading r mode.

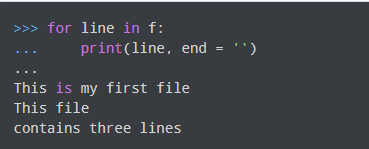
There are various methods available for this purpose. We can use the read(size) method to read in the size number of data. If the size parameter is not specified, it reads and returns up to the end of the file.

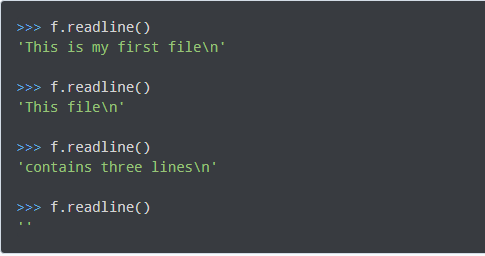


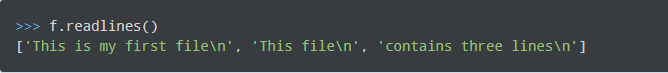
We can see that the read() method returns a newline as '\n'. Once the end of the file is reached, we get an empty string on further reading.

We can change our current file cursor (position) using the seek() method. Similarly, the tell() method returns our current position (in number of bytes).









## Python Directory

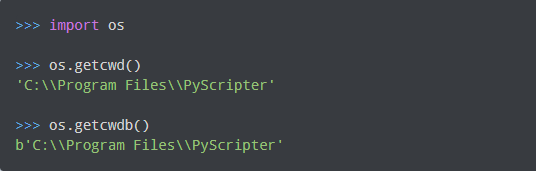
If there are a large number of files to handle in our Python program, we can arrange our code within different directories to make things more manageable.

A directory or folder is a collection of files and subdirectories. Python has the os module that provides us with many useful methods to work with directories (and files as well).

## Get Current Directory

We can get the present working directory using the getcwd() method of the os module.

This method returns the current working directory in the form of a string. We can also use the getcwdb() method to get it as bytes object.

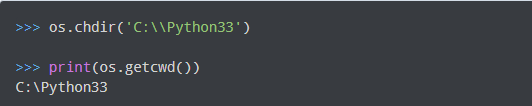


## Changing Directory

We can change the current working directory by using the chdir() method.

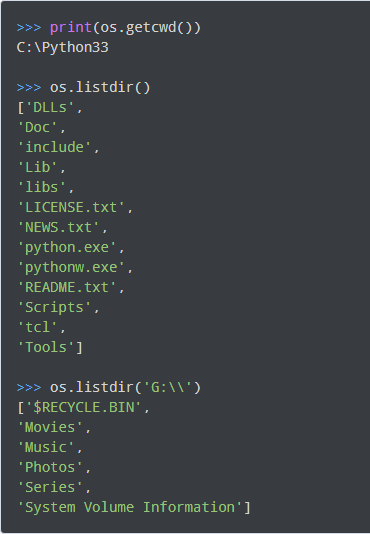
The new path that we want to change into must be supplied as a string to this method. We can use both the forward-slash / or the backward-slash \ to separate the path elements.

It is safer to use an escape sequence when using the backward slash.



## List Directories and Files

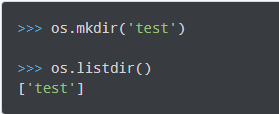
All files and sub-directories inside a directory can be retrieved using the listdir() method.



## Making a New Directory

We can make a new directory using the mkdir() method.

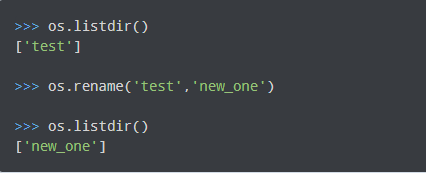
This method takes in the path of the new directory. If the full path is not specified, the new directory is created in the current working directory.



## Renaming a Directory or a File

The rename() method can rename a directory or a file.

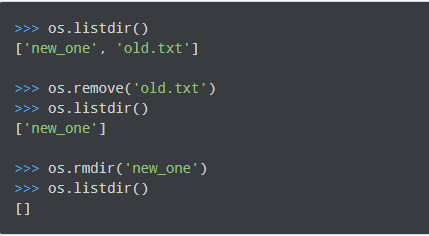
For renaming any directory or file, the rename() method takes in two basic arguments: the old name as the first argument and the new name as the second argument.



## Removing Directory or File

A file can be removed (deleted) using the remove() method.

Similarly, the rmdir() method removes an empty directory.



**Note**: The rmdir() method can only remove empty directories.