

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

# Python File Handling Tutorial: How To Create, Open, Read, Write

Last Updated: [October 25, 2022](#)

**An Intensive Look at Python File Handling Operations with Hands-on Examples:**

In the series of [Python tutorial for beginners](#), we learned more about [Python String Functions](#) in our last tutorial.

Python provides us with an important feature for reading data from the file and writing data into a file.

Mostly, in programming languages, all the values or data are stored in some variables which are volatile in nature.

Because data will be stored into those variables during run-time only and will be lost once the program execution is completed. Hence it is better to save these data permanently using files.



## What You Will Learn: [\[hide\]](#)

- [How Python Handle Files?](#)
- [Types Of File in Python](#)
  - [Binary files in Python](#)
  - [Text files in Python](#)
- [Python File Handling Operations](#)
  - [Python Create and Open a File](#)
  - [Python Read From File](#)
  - [Python Write to File](#)
  - [Python Append to File](#)
  - [Python Close File](#)
  - [Python Rename or Delete File](#)

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

- [Encoding in Files](#)
- [Writing and Reading Data from a Binary File](#)
- [File I/O Attributes](#)
- [Python File Methods](#)
- [Summary](#)
- [Recommended Reading](#)

## How Python Handle Files?

If you are working in a large software application where they process a large number of data, then we cannot expect those data to be stored in a variable as the variables are volatile in nature.

Hence when are you about to handle such situations, the role of files will come into the picture.

As files are non-volatile in nature, the data will be stored permanently in a secondary device like Hard Disk and using python we will handle these files in our applications.

### ***Are you thinking about how python will handle files?***

Let's take an **Example** of how normal people will handle the files. If we want to read the data from a file or write the data into a file, then, first of all, we will open the file or will create a new file if the file does not exist and then perform the normal read/write operations, save the file and close it.

Similarly, we do the same operations in python using some in-built methods or functions.

## Types Of File in Python

There are two types of files in Python and each of them are explained below in detail with examples for your easy understanding.

**They are:**

- Binary file
- Text file

### Binary files in Python

Most of the files that we see in our computer system are called binary files.

**Example:**

1. **Document files:** .pdf, .doc, .xls etc.
2. **Image files:** .png, .jpg, .gif, .bmp etc.
3. **Video files:** .mp4, .3gp, .mkv, .avi etc.
4. **Audio files:** .mp3, .wav, .mka, .aac etc.
5. **Database files:** .mdb, .accde, .frm, .sqlite etc.
6. **Archive files:** .zip, .rar, .iso, .7z etc.
7. **Executable files:** .exe, .dll, .class etc.

### ***Recommended Reading => [How to Open .7z File](#)***

All binary files follow a specific format. We can open some binary files in the normal text editor but we can't read the content present inside the file. That's because all the binary files will be encoded in the binary format, which can be understood only by a computer or machine.

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

For handling such binary files we need a specific type of software to open it.

**For Example,** You need Microsoft word software to open .doc binary files. Likewise, you need a pdf reader software to open .pdf binary files and you need a photo editor software to read the image files and so on.

## Text files in Python

Text files don't have any specific encoding and it can be opened in normal text editor itself.

### Example:

- **Web standards:** html, XML, CSS, JSON etc.
- **Source code:** c, app, js, py, java etc.
- **Documents:** txt, tex, RTF etc.
- **Tabular data:** csv, tsv etc.
- **Configuration:** ini, cfg, reg etc.

In this tutorial, we will see how to handle both text as well as binary files with some classic examples.

## 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.

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

- **'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

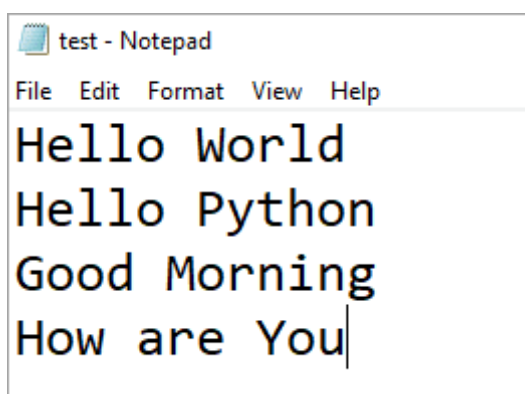
In order to read a file in python, we must open the file in read mode.

**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.



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

#### Example 1:

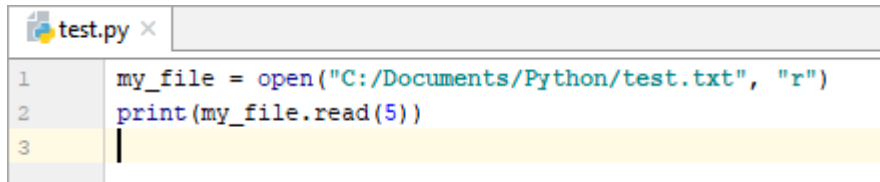
These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
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.



```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "r")
2 print(my_file.read(5))
3 |
```

**Output:**

Hello

Process finished with exit code 0

**Example 2:**

```
my_file = open("C:/Documents/Python/test.txt", "r")
print(my_file.read())
```

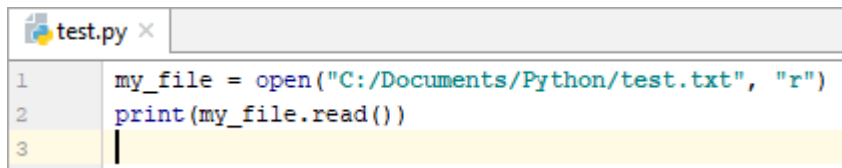
**Output:**

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.



```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "r")
2 print(my_file.read())
3 |
```

**Output:**

Hello World

Hello Python

Good Morning

Process finished with exit code 0

**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.

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "r")
2 print(my_file.readline(2))
3 |
```

**Output:**

```
He

Process finished with exit code 0
```

**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.

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "r")
2 print(my_file.readline())
3 |
```

**Output:**

```
Hello World

Process finished with exit code 0
```

**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.

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "r")
2 print(my_file.readlines())
3 |
```

**Output:**

```
['Hello World\n', 'Hello Python\n', 'Good Morning']

Process finished with exit code 0
```

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

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

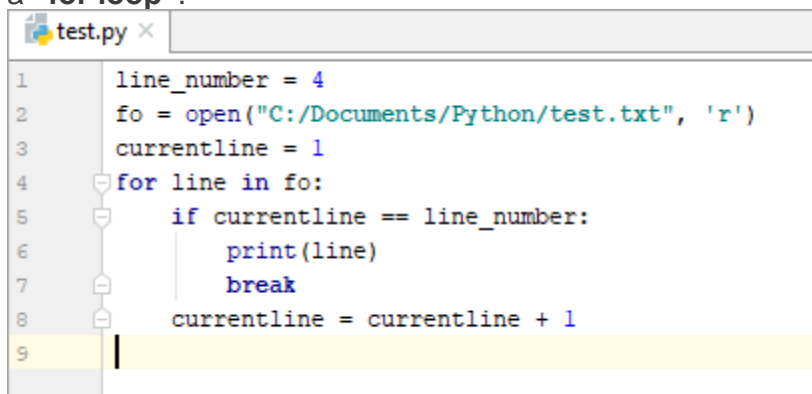
### Reading a specific line from a File

```
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 4<sup>th</sup> line from the 'test.txt' file using a "for loop".

A screenshot of a Python IDE window titled 'test.py'. The code is as follows:

```
1 line_number = 4
2 fo = open("C:/Documents/Python/test.txt", 'r')
3 currentline = 1
4 for line in fo:
5     if currentline == line_number:
6         print(line)
7         break
8     currentline = currentline + 1
9
```

The line numbers 1 through 9 are visible on the left side of the editor.

#### Output:

How are You

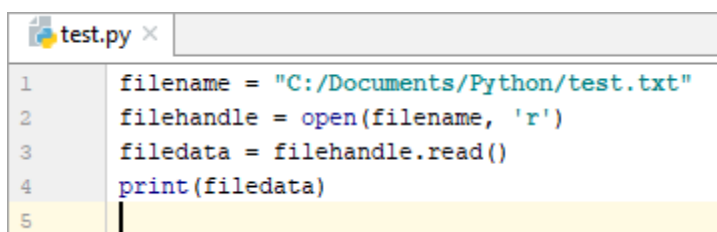
Process finished with exit code 0

### 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

A screenshot of a Python IDE window titled 'test.py'. The code is as follows:

```
1 filename = "C:/Documents/Python/test.txt"
2 filehandle = open(filename, 'r')
3 filedata = filehandle.read()
4 print(filedata)
5
```

The line numbers 1 through 5 are visible on the left side of the editor.

#### Output:

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
Hello World
Hello Python
Good Morning
How are You

Process finished with exit code 0
```

## 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.**

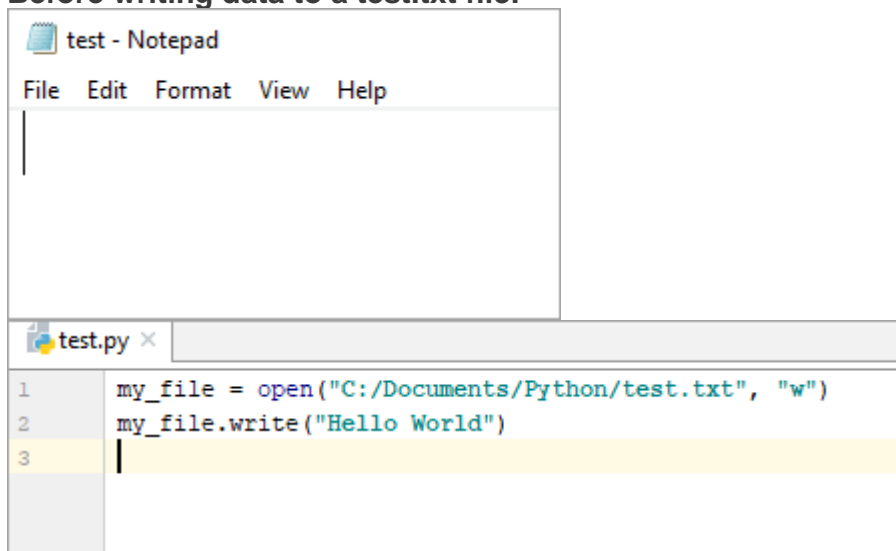
- write(string)
- writelines(list)

**Example 1:**

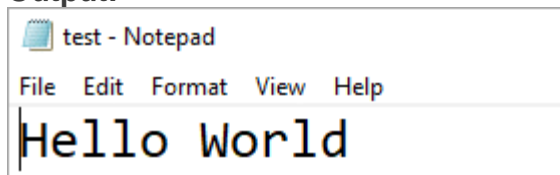
```
my_file = open("C:/Documents/Python/test.txt", "w")
my_file.write("Hello World")
```

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'.



These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

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

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "w")
2 my_file.write("Hello World\n")
3 my_file.write("Hello Python")
4 |
```

Output:

```
test - Notepad
File Edit Format View Help
Hello World
Hello Python
```

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.

```
test.py x
1 fruits = ["Apple\n", "Orange\n", "Grapes\n", "Watermelon"]
2 my_file = open("C:/Documents/Python/test.txt", "w")
3 my_file.writelines(fruits)
4 |
```

Output:

```
test - Notepad
File Edit Format View Help
Apple
Orange
Grapes
Watermelon
```

## 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.

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "a+")
2 my_file.write("Strawberry")
3 |
```

Output:

```
test - Notepad
File Edit Format View Help
Apple
Orange
Grapes
WatermelonStrawberry
```

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**.

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "a+")
2 my_file.write("\nGuava")
3 |
```

Output:

```
test - Notepad
File Edit Format View Help
Apple
Orange
Grapes
WatermelonStrawberry
Guava
```

Example 3:

```
fruits = ["\nBanana", "\nAvocado", "\nFigs", "\nMango"]
```

```
my_file = open("C:/Documents/Python/test.txt", "a+")
```

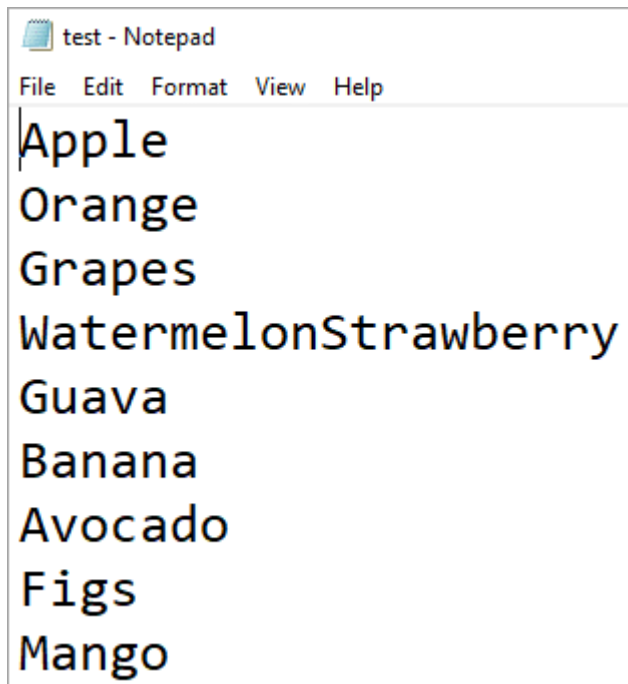
```
my_file.writelines(fruits)
```

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

```
test.py x
1 fruits = ["\nBanana", "\nAvocado", "\nFigs", "\nMango"]
2 my_file = open("C:/Documents/Python/test.txt", "a+")
3 my_file.writelines(fruits)
4 |
```

Output:

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes



**Example 4:**

```
text=["\nHello","\nHi","\nPython"]
my_file=open("C:/Documents/Python/test.txt",mode="a+")
my_file.writelines(text)
print("where the file cursor is:",my_file.tell())
my_file.seek(0)
for line in my_file:
    print(line)
```

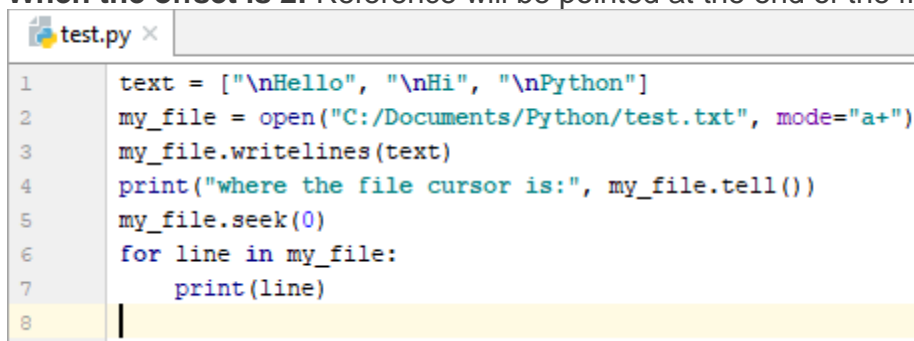
In the above code, we are appending the list of data into the 'test.txt' file. Here, you can observe that we have used the tell() method which prints where the cursor is currently at.

**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.



**Output:**

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
where the file cursor is: 99
Apple

Orange

Grapes

WatermelonStrawberry

Guava

Banana

Avocado

Figs

Mango

Hello

Hi

Python

Process finished with exit code 0
```

## 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:

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

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")
```

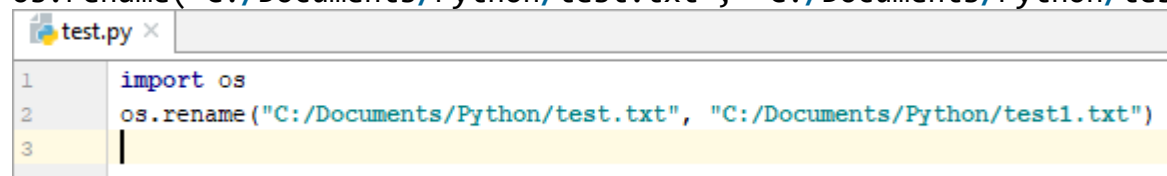
Here 'test.txt' is the current file name and 'test1.txt' is the new file name.

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

### Example 2:

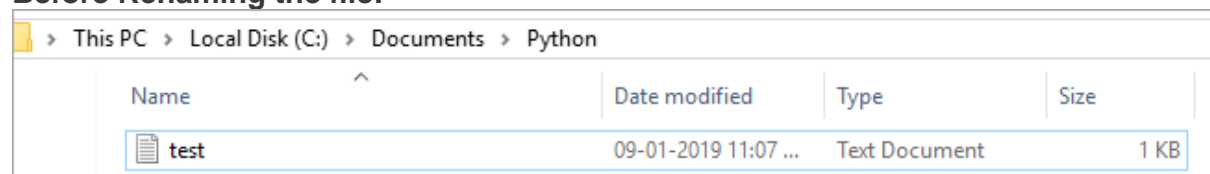
```
import os
```

```
os.rename("C:/Documents/Python/test.txt", "C:/Documents/Python/test1.txt")
```



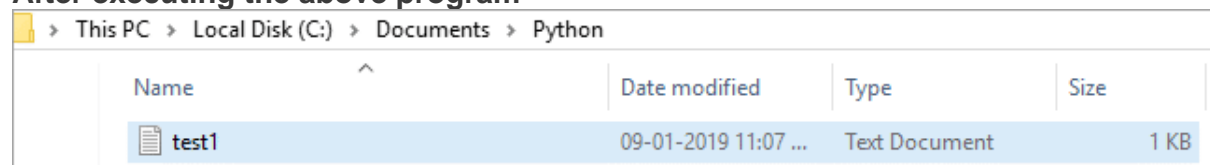
```
test.py x
1 import os
2 os.rename("C:/Documents/Python/test.txt", "C:/Documents/Python/test1.txt")
3
```

### Before Renaming the file:



This PC > Local Disk (C:) > Documents > Python				
Name	Date modified	Type	Size	
test	09-01-2019 11:07 ...	Text Document	1 KB	

### After executing the above program



This PC > Local Disk (C:) > Documents > Python				
Name	Date modified	Type	Size	
test1	09-01-2019 11:07 ...	Text Document	1 KB	

### 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
```

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
os.remove("C:/Documents/Python/test.txt")
```

## Encoding in Files

File encoding represents converting characters into a specific format which only a machine can understand.

**Different machines have different encoding format as shown below.**

- Microsoft Windows OS uses 'cp1252' encoding format by default.
- Linux or Unix OS uses 'utf-8' encoding format by default.
- Apple's MAC OS uses 'utf-8' or 'utf-16' encoding format by default.

***Let's see the encoding operation with some examples.***

**Example 1:**

```
my_file = open("C:/Documents/Python/test.txt", mode="r")
print("Microsoft Windows encoding format by default is:", my_file.encoding)
my_file.close()
```

**Output:**

Microsoft Windows encoding format by default is cp1252.

Here, I executed my program on the windows machine, so it has printed the default encoding as 'cp1252'.

**Output:**

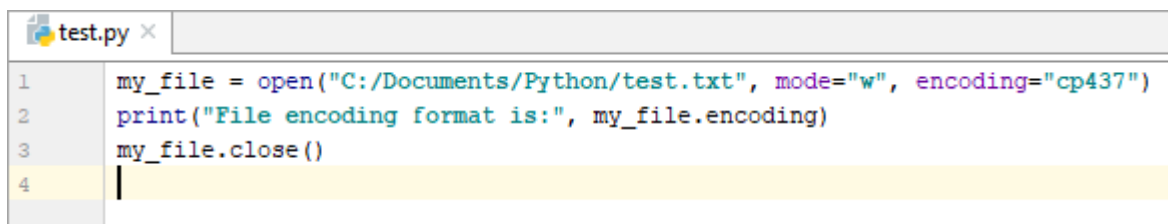
We can also change the encoding format of a file by passing it as arguments to the open function.

**Example 2:**

```
my_file = open("C:/Documents/Python/test.txt", mode="w", encoding="cp437")
print("File encoding format is:", my_file.encoding)
my_file.close()
```

**Output:**

File encoding format is: cp437

A screenshot of a code editor window titled 'test.py'. The editor contains four lines of Python code: 1. my\_file = open("C:/Documents/Python/test.txt", mode="w", encoding="cp437") 2. print("File encoding format is:", my\_file.encoding) 3. my\_file.close() 4. A blank line with a cursor. The code is syntax-highlighted, with strings in blue, keywords in green, and punctuation in red. The line numbers 1, 2, 3, and 4 are visible on the left side of the editor.

**Output:**

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
File encoding format is: cp437

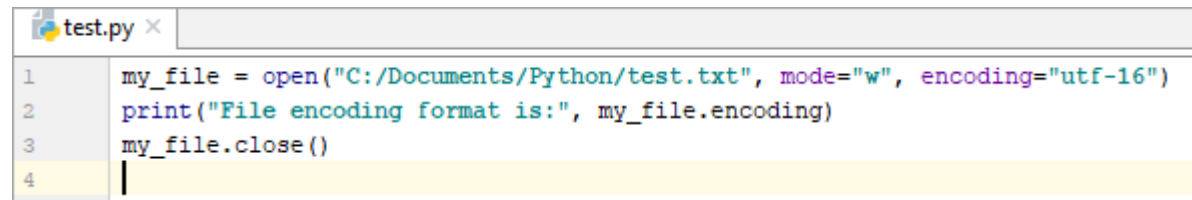
Process finished with exit code 0
```

### Example 3:

```
my_file = open("C:/Documents/Python/test.txt", mode="w", encoding="utf-16")
print("File encoding format is:", my_file.encoding)
my_file.close()
```

### Output:

File encoding format is: utf-16

A screenshot of a code editor window titled 'test.py'. The code is as follows:

```
1 my_file = open("C:/Documents/Python/test.txt", mode="w", encoding="utf-16")
2 print("File encoding format is:", my_file.encoding)
3 my_file.close()
4 |
```

### Output:

```
File encoding format is: utf-16

Process finished with exit code 0
```

## Writing and Reading Data from a Binary File

Binary files store data in the binary format (0's and 1's) which is understandable by the machine. So when we open the binary file in our machine, it decodes the data and displays in a human-readable format.

### Example:

#Let's create some binary file.

```
my_file = open("C:/Documents/Python/bfile.bin", "wb+")
message = "Hello Python"
file_encode = message.encode("ASCII")
my_file.write(file_encode)
my_file.seek(0)
bdata = my_file.read()
print("Binary Data:", bdata)
ntext = bdata.decode("ASCII")
print("Normal data:", ntext)
```

In the above example, first we are creating a binary file '**bfile.bin**' with the read and write access and whatever data you want to enter into the file must be encoded before you call the write method.

Also, we are printing the data without decoding it, so that we can observe how the data exactly looks inside the file when it's encoded and we are also printing the same data by decoding it so that it can be readable by humans.

### Output:

Binary Data: b'Hello Python'  
Normal data: Hello Python

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
test.py x
1 my_file = open("C:/Documents/Python/binaryf.dll", "wb+")
2 message = "Hello Python"
3 file_encode = message.encode("ASCII")
4 my_file.write(file_encode)
5 my_file.seek(0)
6 bdata = my_file.read()
7 print("Binary Data:", bdata)
8 ntext = bdata.decode("ASCII")
9 print("Normal data:", ntext)
10
```

Output:

## File I/O Attributes

Attribute	Description
Name	Return the name of the file
Mode	Return mode of the file
Encoding	Return the encoding format of the file
Closed	Return true if the file closed else returns false

**Example:**

```
my_file = open("C:/Documents/Python/test.txt", "a+")
print("What is the file name? ", my_file.name)
print("What is the file mode? ", my_file.mode)
print("What is the encoding format? ", my_file.encoding)
print("Is File closed? ", my_file.closed)
my_file.close()
print("Is File closed? ", my_file.closed)
```

**Output:**

```
What is the file name? C:/Documents/Python/test.txt
What is the file mode? r
What is the encoding format? cp1252
Is File closed? False
Is File closed? True
```



These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "a+")
2 print("What is the file name? ", my_file.name)
3 print("What is the file mode? ", my_file.mode)
4 print("What is the encoding format? ", my_file.encoding)
5 my_file.close()
6 print("Is File closed? ", my_file.closed)
7
```

**Output:**

```
What is the file name? C:/Documents/Python/test.txt
What is the file mode? a+
What is the encoding format? cp1252
Is File closed? False
Is File closed? True

Process finished with exit code 0
```

Let's try out a few other methods of the file.

**Example:**

```
my_file = open("C:/Documents/Python/test.txt", "w+")
my_file.write("Hello Python\nHello World\nGood Morning")
my_file.seek(0)
print(my_file.read())
print("Is file readable: ?", my_file.readable())
print("Is file writable: ?", my_file.writable())
print("File no:", my_file.fileno())
my_file.close()
```

**Output:**

```
Hello Python
Hello World
Good Morning
Is file readable:? True
Is file writable:? True
File no: 3
```

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", "w+")
2 my_file.write("Hello Python\nHello World\nGood Morning")
3 my_file.seek(0)
4 print(my_file.read())
5 print("Is file readable: ?", my_file.readable())
6 print("Is file writable: ?", my_file.writable())
7 print("File no:", my_file.fileno())
8 my_file.close()
9
```

**Output:**

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
Hello Python
Hello World
Good Morning
Is file readable: ? True
Is file writeable: ? True
File no: 3

Process finished with exit code 0
```

## Python File Methods

Function	Explanation
open()	To open a file
close()	Close an open file
fileno()	Returns an integer number of the file
read(n)	Reads 'n' characters from the file till end of the file
readable()	Returns true if the file is readable
readline()	Read and return one line from the file
readlines()	Reads and returns all the lines from the file
seek(offset)	Change the cursor position by bytes as specified by the offset
seekable()	Returns true if the file supports random access
tell()	Returns the current file location
writable()	Returns true if the file is writable
write()	Writes a string of data to the file
writelines()	Writes a list of data to the file

**Let's see what we have discussed so far in an end-end program.**

**Example:**

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

```
test.py x
1 my_file = open("C:/Documents/Python/test.txt", mode="w+")
2 print("What is the file name? ", my_file.name)
3 print("What is the mode of the file? ", my_file.mode)
4 print("What is the encoding format?", my_file.encoding)
5
6 text = ["Hello Python\n", "Good Morning\n", "Good Bye"]
7 my_file.writelines(text)
8 print("Size of the file is:", my_file.__sizeof__())
9 print("Cursor position is at byte:", my_file.tell())
10 my_file.seek(0)
11 print("Content of the file is:", my_file.read())
12 my_file.close()
13
14 file = open("C:/Documents/Python/test.txt", mode="r")
15 line_number = 3
16 current_line = 1
17 data = 0
18 for line in file:
19     if current_line == line_number:
20         data = line
21         print("Data present at current line is:", data)
22         break
23     current_line = current_line + 1
24
25 bin_file = open("C:/Documents/Python/bfile.exe", mode="wb+")
26 message_content = data.encode("utf-32")
27 bin_file.write(message_content)
28 bin_file.seek(0)
29 bdata = bin_file.read()
30 print("Binary Data is:", bdata)
31 ndata = bdata.decode("utf-32")
32 print("Normal Data is:", ndata)
33 file.close()
34 bin_file.close()
```

### Output:

```
What is the file name? C:/Documents/Python/test.txt
What is the mode of the file? w+
What is the encoding format? cp1252
Size of the file is: 192
Cursor position is at byte: 36
Content of the file is: Hello Python
Good Morning
Good Bye
Data present at current line is: Good Bye
Binary Data is: b'\xff\xfe\x00\x00G\x00\x00\x00o\x00\x00\x00o\x00\x00\x00d\x00\x00\x00
\x00\x00\x00B\x00\x00\x00y\x00\x00\x00e\x00\x00\x00'
Normal Data is: Good Bye
Process finished with exit code 0
```

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

## **Tell()**

```
text=["Hello\n","Hi\n","Python\n"]
file=open("test.txt","a+")
file.writelines(text)
print("where the file cursor is:",file.tell())
#Here, you can observe that we have used the tell() method which
prints
#where the cursor is currently at.
Hello
Hi
Python
where the file cursor is: 19
```

## **seek()**

- #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.

```
text=["Hello\n","Hi\n","Python\n"]
file=open("test.txt","a+")
file.writelines(text)
print(file.seek(0))    #0
file.close()
```

```
file = open("test.txt", "rb")
# sets Reference point to tenth
# position to the left from end
file.seek(-10, 2)  #seek(reference,offset)
print("Cursor Position :",file.tell())
#Cursor Position : 9
print(file.close())
```

These notes are only to get basic Knowledge, you'll have to study extensively separately, refer class notes

## Summary

**Enlisted below are a few pointers that can be summarized from the above tutorial:**

- We usually use a file for storing data permanently in the secondary storage as it is non-volatile in nature, so that the data may be used in the future.
- At times in some applications we may want to read the data from a text file or binary file, so we can achieve it using the in-built functions in Python like open, read, write methods etc.
- You have to be very careful while using the write method because whatever data you write into the file will be overwritten and the old data will be lost.
- In order, to prevent overwriting of data it's better to open a file in write and append mode so that data will be appended at the end of the file.
- Remember, when you open a file in the binary mode it does not accept the encoding parameter.
- You can perform renaming and deleting on a file using the rename and remove methods from the "os" module/ package.

***We hope you enjoyed this informative tutorial on Python File Handling. Our upcoming tutorial will explain more about Python Main Function.***

**[PREV Tutorial](#) | [NEXT Tutorial](#)**