

# File IO in Python

File is a named location on disk to store related information. It is 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 computer is turned off, we use files for future use of the data.

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 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

## Open file :

Python has a built-in function `open()` 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.

```
Fd1 = open("Marvellous.txt",'r')
```

Search file in current directory as we provide relative path of file

```
Fd2 = open("/Users/marvellous/Desktop/Today/Marvellous.txt")
```

Search in specific path as we provide absolute path

We can specify the mode while opening a file.

In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file.

We also specify if we want to open the file in text mode or binary mode.

The default is reading in text mode. In this mode, we get strings when reading from the file.

On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

```
f = open("Marvellous.txt")           # equivalent to 'r' or 'rt'
f = open("Marvellous.txt",'w')       # write in text mode
f = open("LogoMarvellous.bmp",'r+b') # read and write in binary mode
```

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.

```
f = open("Marvellous.txt",mode = 'r',encoding = 'utf-8')
```

### Python File Modes

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

### Close File :

When we are done with operations to the file, we need to properly close the file. Closing a file will free up the resources that were tied with the file and is done using Python close() method. Python has a garbage collector to clean up unreferenced objects but, we must not rely on it to close the file.

```
f = open("Marvellous.txt",encoding = 'utf-8')
```

```
# perform file operations
```

```
f.close()
```

### Read data from file:

Consider below file that we refer for reading

Marvellous.txt

Marvellous Infosystems by Piyush Manohar Khairnar  
Karve Road Pune 411004  
Educating for better tomorrow..

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

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

```
fd = open("Marvellous.txt",'r',encoding = 'utf-8')
```

```
fd.read(10) # read the first 10 data
```

Output :

"Marvellous"

```
fd.read(12) # read the next 12 data
```

Output :

"Infosystems"

```
fd.read() # read in the rest till end of file
```

Output :

by Piyush Manohar Khairnar

Karve Road Pune 411004

Educating for better tomorrow..

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

```
print("Current file position is",fd.tell()) # get the current file position
```

```
fd.seek(0) # bring file cursor to initial position
```

```
print("Contents of Whole file")
```

```
print(fd.read())
```

## Writing data into file :

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. All previous data are erased.

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

```
fd = open("Marvellous.txt",'w+a',encoding = 'utf-8')
```

```
fd.write("Python : Automation and Machine Learning\n")
```

```
fd.write("Angular : Web Development\n")
```

## File IO methods in Python

Method	Description
close()	Close an open file. It has no effect if the file is already closed.
detach()	Separate the underlying binary buffer from the <code>TextIOBase</code> and return it.
fileno()	Return an integer number (file descriptor) of the file.
flush()	Flush the write buffer of the file stream.
isatty()	Return <code>True</code> if the file stream is interactive.
read( <code>n</code> )	Read atmost <code>n</code> characters form the file. Reads till end of file if it is negative or <code>None</code> .
readable()	Returns <code>True</code> if the file stream can be read from.
readline( <code>n</code> =-1)	Read and return one line from the file. Reads in at most <code>n</code> bytes if specified.
readlines( <code>n</code> =-1)	Read and return a list of lines from the file. Reads in at most <code>n</code> bytes/characters if specified.
seek( <code>offset</code> , <code>from</code> = <code>SEEK_SET</code> )	Change the file position to <code>offset</code> bytes, in reference to <code>from</code> (start, current, end).
seekable()	Returns <code>True</code> if the file stream supports random access.
tell()	Returns the current file location.
truncate( <code>size</code> = <code>None</code> )	Resize the file stream to <code>size</code> bytes. If <code>size</code> is not specified, resize to current location.
writable()	Returns <code>True</code> if the file stream can be written to.
write( <code>s</code> )	Write string <code>s</code> to the file and return the number of characters written.
writelines( <code>lines</code> )	Write a list of <code>lines</code> to the file.

## Consider below application which demonstrates file IO

```
fd = open("Marvellous.txt",'r')

print("Information about file : ",fd)

print("Contents of Whole file")
print(fd.read())

print("Reading single line from file")
print(fd.readline())

print("Current file position is",fd.tell())    # get the current file position

fd.seek(0)    # bring file cursor to initial position

print("Contents of Whole file")
print(fd.read())

fd.close()

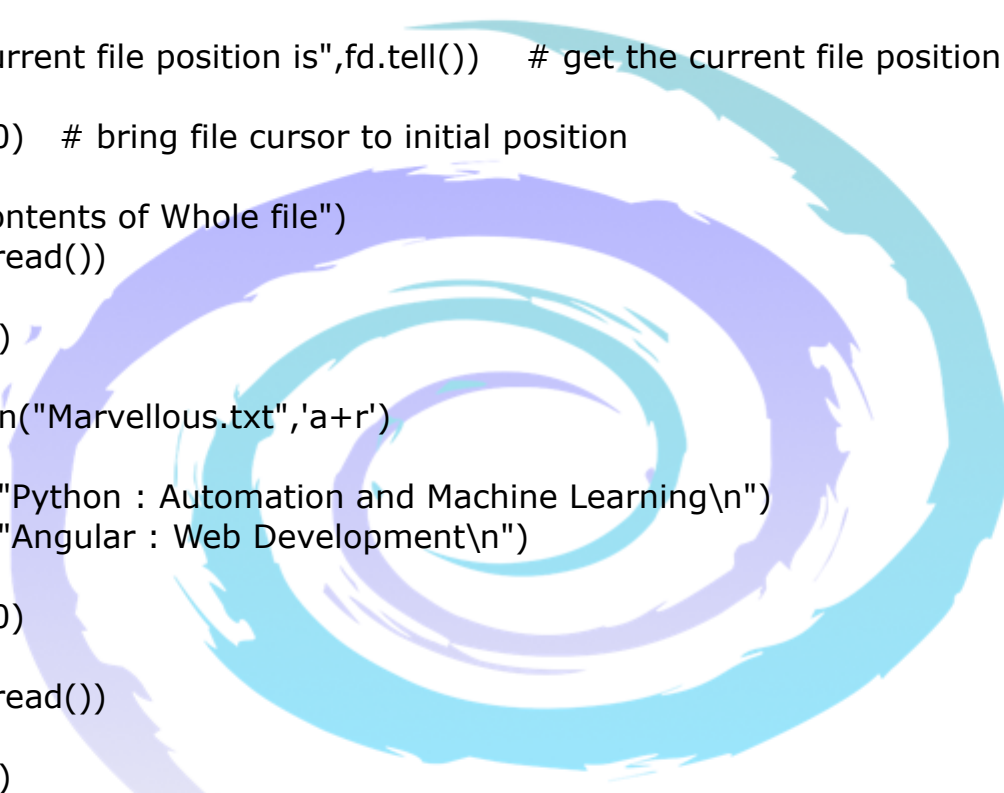
fd = open("Marvellous.txt",'a+r')

fd.write("Python : Automation and Machine Learning\n")
fd.write("Angular : Web Development\n")

fd.seek(0)

print(fd.read())

fd.close()
```



## Output of above application

```
MacBook-Pro-de-MARVELLOUS: Today marvellous$ python
FileIO.py
('Information about file : ', <open file 'Marvellou
s.txt', mode 'r' at 0x10397c5d0>)
Contents of Whole file
Marvellous Infosystems by Piyush Manohar Khairnar
Karve Road Pune 411004
Educating for better tomorrow..Python : Automation
and Machine Learning
Angular : Web Development

Reading single line from file

('Current file position is', 171)
Contents of Whole file
Marvellous Infosystems by Piyush Manohar Khairnar
Karve Road Pune 411004
Educating for better tomorrow..Python : Automation
and Machine Learning
Angular : Web Development
Marvellous Infosystems by Piyush Manohar Khairnar
Karve Road Pune 411004
Educating for better tomorrow..Python : Automation
and Machine Learning
Angular : Web Development
Python : Automation and Machine Learning
Angular : Web Development
```