



File Handling

Input - Output

- Input is any information provided to the program
 - Keyboard input
 - Mouse input
 - File input
 - Sensor input (microphone, camera, photo cell, etc.)
- Output is any information (or effect) that a program produces:
 - sounds, lights, pictures, text, motion, etc.
 - on a screen, in a file, on a disk or tape, etc.

Need of Files

- Small businesses accumulate various types of data, such as financial information related to revenues and expenses and data about employees, customers and vendors.
- Traditional file organization describes storing data in paper files, within folders and filing cabinets.
- Electronic file organization is a common alternative to paper filing; each system has its benefits and drawbacks.

File Handling

- A file stores information or data in different formats in the permanent storage/Hard disk of a computer system. The different types data includes, audio files, video files, text files, binary files, etc.
- Files can be stored with different file extensions based on the software in which the data is stored.
 - MS Word files - **doc or docx** extension
 - MS Excel files - **xls or xlsx** extension
 - Notepad files - **txt** extension
 - Wordpad files - **dat** extension
 - Video files - **mp4** extension
 - Audio files - **mp3** extension

- Basically any file can store two types of data formats.
 - **Raw/Binary data**
 - **Text/User readable data**
- **Text files:** In this type of file, Each line of text is terminated with a special character called EOL (End of Line), which is the new line character ('\n') in python by default.
- **Binary files:** In this type of file, there is no terminator for a line and the data is stored after converting it into machine-understandable binary language.

File Operations

- Any file operation follows a sequence such as
 - Open a file
 - Read or Write (Perform operation)
 - Close a file

Opening a File

- The `open()` function is used to open a file in Python. The `open()` function return a file object and the file object is used to read or modify the file contents. The syntax is
`file object = open("file name", "mode")`
- **File object** is a variable storing the file object for `open()` function. **File name** is the name of the file along with path and extension to be given. In **Mode**, we have to specify whether we want read, write or append to the file.
- We can also specify if we want to open the file in text mode or binary mode. The default is reading in text mode.

Opening a File

Mode	Description
r	Opens a file for reading. (default)
w	Opens a file for writing. Creates a new file if it does not exist or truncates the file if it exists and overwrites. Cursor/Handle is positioned at the beginning of the file.
a	Opens a file for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Opens in text mode. (default)
b	Opens in binary mode.
+	Opens a file for updating (reading and writing)

Opening a File

Mode	Description
r+	Both Read/Write [will not create the file if file is not present. Will overwrite from the beginning of the file while writing][will not truncate]
w+	Both Read/Write [Will create the file if file is not present and will write from the beginning]. For an existing file, data is truncated and over-written. The handle is positioned at the beginning of the file.
a+	Appending/Reading [append and read][will create the file if file not present]

Reading a File

- To read a file in Python, we must open the file in reading 'r' mode. There are various methods available with the file object.

Method	Description
<code>close()</code>	Closes an opened file. It has no effect if the file is already closed.
<code>read(n)</code>	Reads at most n characters from the file. Reads till end of file if it is negative or None.
<code>readline(n=-1)</code>	Reads and returns one line from the file. Reads in at most n bytes if specified.
<code>readlines(n=-1)</code>	Reads and returns a list of lines from the file. Reads in at most n bytes/characters if specified.

Reading a File

```
file=open("a.txt", 'r') #Opens a file in read mode  
print(file.read())      #Reads entire file contents  
file.close()            #Closes the file
```

```
file1=open("a.txt", mode='r')  
print(file1.read())  
file1.close()
```

```
file=open("C:\\Users\\Admin\\Desktop\\a.txt", 'rt')  
#Opens a file in read mode in text format i.e. by default read mode in  
#text format only, so no need to mention 't' along with 'r'  
print(file.read()) #Reads entire file contents  
file.close()       #Closing the file
```

```
file=open("C:/Users/Admin/Desktop/a.txt", 'r') #Try path this way  
file=open(r"C:\Users\Admin\Desktop\a.txt", 'r') #Try path this way
```

Reading a File

```
file=open("a.txt", 'r') #Opening a file in read mode
print(file.read(10)) #Reads first 10 characters only
print(file.read(10)) #From the current position it reads 10 characters
file.close() #Closing the file
```

```
file=open("a.txt", 'r') #Opening a file in read mode
print(file.readline()) #Reads first one line only
print(file.readline()) #Reads second line only
print(file.readline()) #Reads third line only
file.close() #Closing the file
```

```
file1=open("a.txt", 'r') #Opening a file in read mode
print(file1.readlines()) #Reads all the lines and display as list of lines
file1.close() #Closing the file
```

Creating a New File

- To create a new file, we use `open()` function with one of the following parameters.

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

"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

```
>>>file=open("b.txt", 'x')
```

```
>>>file=open("c.txt", 'a')
```

```
>>>file=open("d.txt", 'w')
```

Writing to a File

- There are two ways to write into a file.

Method	Description
<code>write(s)</code>	Writes the strings to the file and returns the number of characters written.
<code>writelines(lines)</code>	Writes a list of lines to the file.

write() : Inserts the string `str1` in a single line in the text file.

`file object.write(str1)`

writelines() : For a list of string elements, each string is inserted in the text file. Used to insert multiple strings at a single time.

`file object.writelines(L)` for `L = [str1, str2, str3]`

Writing to a File

```
f1=open("a.txt", 'w')
f1.write("Welcome to Python Programming")
f1.close()
```

```
f1=open("a.txt", 'w')
S1="Welcome to Python Programming"
print(f1.write(S1))  # Displays no. of characters written
f1.close()
```

```
f1=open("a.txt", 'w')
S1="Welcome to Python Programming"
L1=['first line\n', 'second line\n', 'third line']
print(f1.write(S1))  # Displays no. of characters written
print(f1.writelines(L1))
f1.close()
```

Writing to a File

```
f1=open("c.txt", 'w')  
f1.write("Welcome to Python Programming")  
f1.close()
```

```
f1=open("c.txt", 'a') #Appending data to a file  
L1=['first line\n', 'second line\n', 'third line']  
print(f1.writelines(L1))  
f1.close()
```


Writing to a File

#Writing integer values to a file

```
f1=open("aa.txt", "w")
```

```
for i in range(20):
```

```
    f1.write('%d\n'%i)
```

```
f1.close()
```

#Writing integer values to a file

```
f1=open("ab.txt", "w")
```

```
for i in range(1,11):
```

```
    f1.write(" This is line %d\n " %i)
```

```
f1.close()
```

Writing to a File

#Writing multiple integer values to a file

```
f1=open("a.txt", 'w')
for i in range(1,11):
    f1.writelines(['%d\t%i, '%d\n'%(10*i)])
f1.close()
```

#Writing floating point values to a file

```
f1=open("a.txt", 'w')
f1.write("The square root of first 10 numbers:\n")
for i in range(1,11):
    f1.writelines(['%d\t%i, '%f\n'%(i**0.5)])
f1.close()
```