



Files:

- File handling is an important part of any web application.
- Python has several functions for creating, reading, updating, and deleting files.
- 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).
- 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.
 - ✓ Open a file
 - ✓ Read or write (perform operation)
 - ✓ Close the file

File Handling:



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

`f=open(filename, mode)`

- 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)



Access Modes of the Files:

Access Modes of the Files

Mode	Description
"r"	Opens the file in read only mode and this is the default mode.
"w"	Opens the file for writing. If a file already exists, then it'll get overwritten. If the file does not exist, then it creates a new file.
"a"	Opens the file for appending data at the end of the file automatically. If the file does not exist it creates a new file.
"r+"	Opens the file for both reading and writing.
"w+"	Opens the file for reading and writing. If the file does not exist it creates a new file. If a file already exists then it will get overwritten.
"a+"	Opens the file for reading and appending. If a file already exists, the data is appended. If the file does not exist it creates a new file.
"x"	Creates a new file. If the file already exists, the operation fails.
"rb"	Opens the binary file in read-only mode.
"wb"	Opens the file for writing the data in binary format.
"rb+"	Opens the file for both reading and writing in binary format.

Open a File :



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

```
f = open("demofile.txt")
```

```
f = open("demoFile.txt", "r")
```

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

```

>>> f = open("mainfile.txt")
>>> f.read()
Traceback (most recent call last):
  File "<ipython>...", line 1, in <module>
    f = open("mainfile.txt")
FileNotFoundError: [Errno 2] No such file or directory: 'mainfile.txt'

```

Open a File :



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

```
Hello everyone  
welcome to programmin in python  
This elective subject for sixth semester  
Happy Learning
```

- 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:

```
>>> f=open('example.txt','r')
```

```
>>> print(f.read())
```

```
Hello everyone
```

```
welcome to programmin in python
```

```
This elective subject for sixth semester
```

```
Happy Learning
```

Open a File :



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

```
Hello everyone  
welcome to programin in python  
This elective subject for sixth semester  
Happy Learning
```

- Return the 9 first characters of the file:

```
>>> f=open('example.txt','r')  
>>> print(f.read(9))  
Hello eve
```



File Object Attributes


- When the Python `open()` function is called, it returns a file object called a file handler. Using this file handler, you can retrieve information about various file attributes

List of File Attributes

Attribute	Description
<code>file_handler.closed</code>	It returns a Boolean True if the file is closed or False otherwise.
<code>file_handler.mode</code>	It returns the access mode with which the file was opened.
<code>file_handler.name</code>	It returns the name of the file.

File Methods to Read and Write Data

List of Methods Associated with the File Object

Method	Syntax	Description
 read()	file_handler. read([size])	This method is used to read the contents of a file up to a size and return it as a string. The argument <i>size</i> is optional, and, if it is not specified, then the entire contents of the file will be read and returned.
readline()	file_handler.readline()	This method is used to read a single line in file.
readlines()	file_handler.readlines()	This method is used to read all the lines of a file as list items.
write()	file_handler. write(string)	This method will write the contents of the string to the file, returning the number of characters written. If you want to start a new line, you must include the new line character.
writelines()	file_handler. writelines(sequence)	This method will write a sequence of strings to the file.
tell()	file_handler.tell()	This method returns an integer giving the file handler's current position within the file, measured in bytes from the beginning of the file.
seek()	file_handler. seek(offset, from_what)	This method is used to change the file handler's position. The position is computed from adding <i>offset</i> to a reference point. The reference point is selected by the <i>from_what</i> argument. A <i>from_what</i> value of 0 measures from the beginning of the file, 1 uses the current file position, and 2 uses the end of the file as the reference point. If the <i>from_what</i> argument is omitted, then a default value of 0 is used, indicating that the beginning of the file itself is the reference point.

Read Lines:



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

```
Hello everyone  
welcome to programmin in python  
This elective subject for sixth semester  
Happy Learning
```

- Read one line of the file:

```
f=open('example.txt','r')  
print(f.readline())  
print(f.readline())
```

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

```
Hello everyone  
  
welcome to programmin in python
```

Read Lines:



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

```
>>> f=open('example.txt','r')
```

```
>>> print(f.readline())
```

```
Hello everyone
```

```
>>> print(f.readline())
```

```
['welcome to programmin in python\n', 'This elective subject for sixth semester\n', 'Happy Learning\n']
```

```
>>> f=open('example.txt','r')
```

```
>>> for i in f:
```

```
    print(i)
```

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





Close Files:

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

```
f=open('example.txt','r')
```

```
for i in f
```

```
    print(i)
```

```
f.close()
```

To check ,file is closed

- `print(f.closed)`

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.

Write to an Existing Files:



- 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
- Open the file "example.txt" and append content to the file:

```
f=open('example.txt','a')  
f.write(' Learning python is fun')  
f.close()
```

```
f=open('example.txt','r')  
print(f.read())
```

File Positions:



- The `tell()` method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.
- The `seek(offset[, from])` method changes the current file position. The offset argument indicates the number of bytes to be moved. The from argument specifies the reference position from where the bytes are to be moved.
- `f.tell()` # get the current file position
- `f.seek(0)` # bring file cursor to initial position