WHAT IS A FILE? ¶

File is named location on disk to store related information.It is used to permanently store data in a non-valatile memory(e.g. hard disk).

FILES IN PYTHON

When we want to read from a file 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, following file operation are availabe.

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

OPEN A FILE

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

In [5]:

```
f = open("hcafe.txt") #open file in current directory
```

In [6]:

```
f = open("/home/anurag/Lms_Python_Course/Exception and File Handling/hcafe.txt") #
```

PYTHON FILE MODES

We can specify the mode while opening a file. For example we specify whether we want to read 'r', write 'w' or append 'a' to the file. We also specify if we want to open a file in text mode or binary mode.

- **TEXT MODE** : It is the default reading mode.In this mode, we get strings when reading from files.
- **BINARY MODE** : Binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

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)

Mode	Description
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

In [7]:

```
f = open("hcafe.txt")      # equivalent to 'r'
```

In [8]:

```
f = open("hcafe.txt",'w')  # write in text mode
```

In [9]:

```
f = open("hcafe.txt",'r+b') # read and write in binary mode
```

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

In [10]:

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

CLOSE A 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.

In [11]:

```
f = open("hcafe.txt", encoding = 'utf-8')
# perform some file operations
f.close()
```

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

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

In [14]:

```
with open("hcafe.txt",encoding = 'utf-8') as f:
    print (f)
    # perform file operations
```

```
<_io.TextIOWrapper name='hcafe.txt' mode='r' encoding='utf-8'>
```

WRITE TO A FILE 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.

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.

In [15]:

```
with open("hcafe.txt", 'w', encoding = 'utf-8') as f:
        f.write("These are the lecture notes of file handling in python\n")
        f.write("hope you would like it\n")
        f.write("thank you\n")
```

We must include the newline characters ourselves to distinguish different lines.

READ FILES IN PYTHON :

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

We can use the read(size) method to read, where size parameter tells that the number of data(characters) we want to read from file.

If size is not specified, it reads and returns up to the end of the file.

In [17]:

```
f = open("hcafe.txt",'r',encoding = 'utf-8')
f.read(5)
```

Out[17]:

```
'These'
```

In [18]:

```
f.read(4)    # read the next 6 data ' are'
```

Out[18]:

```
' are'
```

In [19]:

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

Out[19]:

```
' the lecture notes of file handling in python\nhope you would like it
\nthank you\n'
```

In [20]:

```
f.read()  # further reading returns empty sting
```

Out[20]:

```
''
```

In [ ]:

```

```