File

- You have been reading and writing to the standard input andoutput until now. Now, we will see how to use actual data files.
- Python provides the basic functions and methods necessary tomanipulate files by default. However, you can perform file manipulation using a file object.
- File is the collection of data that is available to a program. We can retrieve and use data stored in a file whenever we required.

Advantages:-

- Stored Data is permanent unless someone remove it.
- Stored data can be shared.
- It is possible to update or remove the data

Some Theory

Types of data used for I/O:

Text - '12345' as a sequence of unicode chars

Binary - 12345 as a sequence of bytes of its binary equivalent

Hence there are 2 file types to deal with

Text files - All program files are text files

Binary Files - Images,music,video,exe files

Text Mode – A file opened in text mode, treats its contents as if it contains text strings of the str type. When you get data from a text mode file, Python first decodes the raw bytes using either a platform-dependent encoding or, specified one.

Binary Mode – A file opened in Binary Mode, Python uses the data in the file without any decoding, binary mode file reflects the raw data in the file.

We will cover the following topics in this chapter:

Open and close files

Modes of opening a file

The file object attributes

Reading and writing files

Renaming and deleting files

File methods

The open function

Before you can read or write a file, you have to open it using Python's built-in open() function. This function creates a file object.

Syntax:

file object = open(file_name [, access_mode][, buffering])

Parameters:

The file_name argument contains the name of the file you wish to access.

Using you can determine how the file should be opened; for example, reading, writing, or appending. As far as file access modes are concerned, read (r) is the default.

Buffering is not performed if the buffering value is set to 0. When the buffering value is 1, line buffering is performed when accessing a file. If you specify more than one as the buffering value, the buffer size will be indicated. The default buffer size (default behavior) is negative.

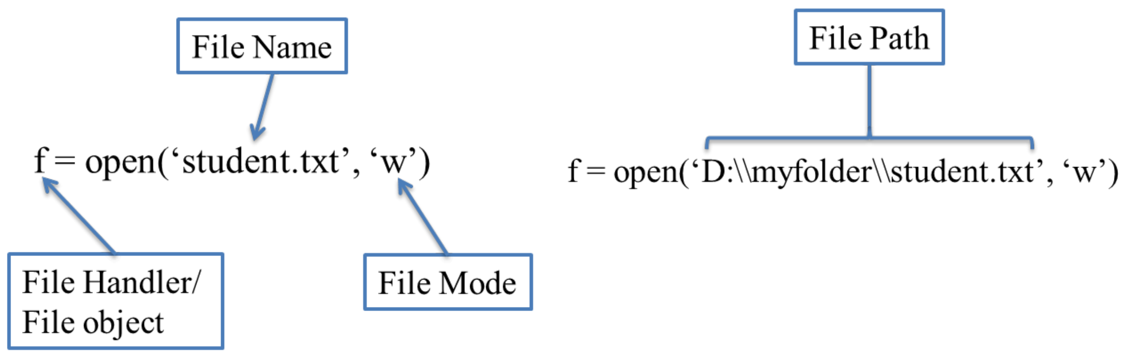
Syntax:- open(‘filename’, mode='r', buffering, encoding=None, errors=None, newline=None, closefd=True, opener=None)

encoding – name of the encoding used to decode or encode the file. It should be used only in text mode. Ex:- utf-8

errors – an optional string that specifies how encoding and decoding errors are to be handled, this cannot be used in binary mode. Some of the standard values are strict, ignore, replace etc.

newline: this parameter controls how universal newlines mode works (it only applies to text mode). It can be None, ”, ‘\n’, ‘\r’, and ‘\r\n’.

closefd – If closefd is False and a file descriptor rather than a filename was given, the underlying file descriptor will be kept open when the file is closed. If a filename is given closefd must be True (the default) otherwise an error will be raised.


File Name
f = open('student.txt', 'w')
File Handler/ File object
File Mode
File Path
f = open('D:\\myfolder\\student.txt', 'w')

The close() method

A file object's close() method flushes any unwritten information and closes it.

The reference object of the file is reassigned to another file. Python automatically closes the file, but it is still a good practice to close a file using the close()

Syntax:

fileObject.close();

close( ) – This method is used to close, opened file.

Once we close the file, file object is deleted from the memory hence file will be no longer accessible unless we open it again. If you don't explicitly close a file, Python's garbage collector will eventually destroy the object and close the open file for you, but the file may stay open for a while so You should always close opened file.

What will happened if we do not close opened file:- Data of the file may be corrupted or deleted. Memory utilized by the file is not freed it may cause of insufficient memory.

Modes & Description

1 r

- Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.

2 r+

- Opens a file for both reading and writing. The file pointer placed at the beginning of the file.

3 w

- Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

4 w+

- Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.

5 a

- Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

6 a+

- Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a

The file Object Attributes

Once a file is opened and you have one file object, you can get various information related to that file. Here is a list of all attributes related to file object −

Attribute & Description

1 file.closed

Returns true if file is closed, false otherwise.

2 file.mode

Returns access mode with which file was opened.

3 file.name

Returns name of the file.

4 readable()

return boolean value indicate that whether file is readable or not

5 writable()

returns boolean value indicate that whether file is writable or not.

In [54]:
```
file=open("vishal.txt","w")
print(file)
file.close()
```

```
<_io.TextIOWrapper name='vishal.txt' mode='w' encoding='cp1252'>
```

In [8]:
```
f=open("vishal.txt","r")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in r
file name is vishal.txt
file is readable True
file is writable False
file is close True
```

In [9]:
```
f=open("vishal.txt","w")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in w
file name is vishal.txt
file is readable False
file is writable True
file is close True
```

In [10]:
```
f=open("vishal.txt","a")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in a
file name is vishal.txt
file is readable False
file is writable True
file is close True
```

In [11]:
```
f=open("vishal.txt","r+")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in r+
file name is vishal.txt
file is readable True
file is writable True
file is close True
```

In [12]:
```
f=open("vishal.txt","w+")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in w+
file name is vishal.txt
file is readable True
file is writable True
file is close True
```

In [13]:
```
f=open("vishal.txt","a+")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in a+
file name is vishal.txt
file is readable True
file is writable True
file is close True
```

default mode is r

In [55]:
```
f=open("vishal.txt")
print("file is close",f.closed)
print("file has been opened in",f.mode)
print("file name is",f.name)
print("file is readable",f.readable())
print("file is writable",f.writable())
f.close()
print("file is close",f.closed)
```

```
file is close False
file has been opened in r
file name is vishal.txt
file is readable True
file is writable False
file is close True
```

Check File exists or not

isfile() – This method is used to check whether specified file is exists or not. This method belongs to path module which is sub module of os module.

Syntax:-

import os

os.path.isfile(filename)

In [14]:
```
import os
os.path.isfile("vishal.txt")
```

Out[14]:
```
True
```

writing files

- The write() method

write(string)

writelines(list of string)

- The write() method writes any string to an open file. It does notadd a newline character to the end of the string.

Syntax:

fileObject.write(string)

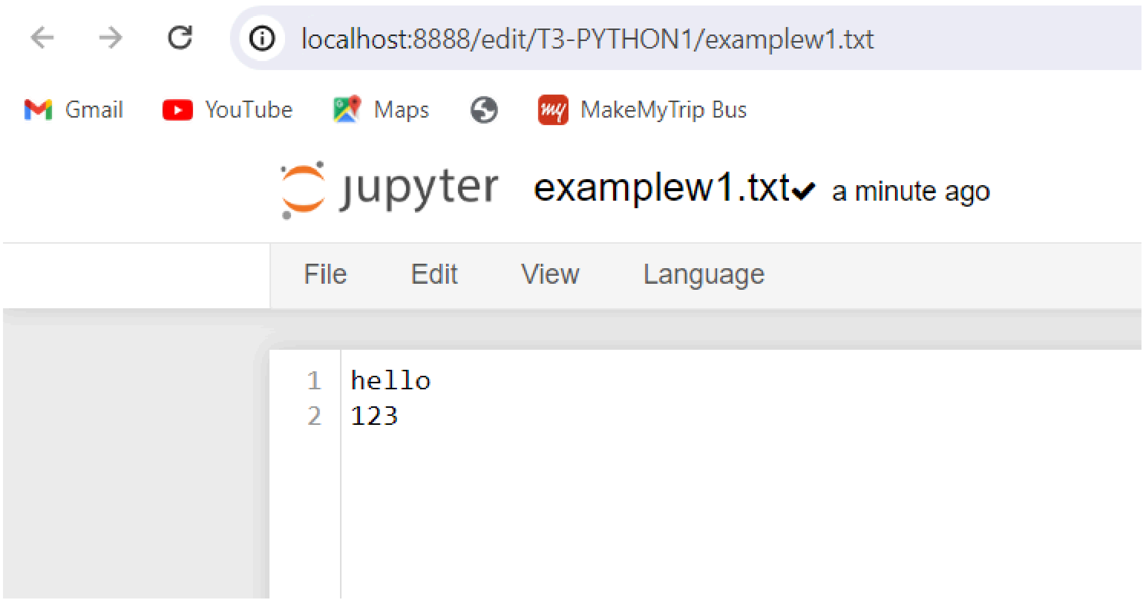
fileobject.writelines(list of string)

Parameter:

String: Content to be written into the opened file.

In [15]:
```
f=open("examplew1.txt","w")
f.write("hello\n")
f.writelines(["1","2","3"])
f.close()
```

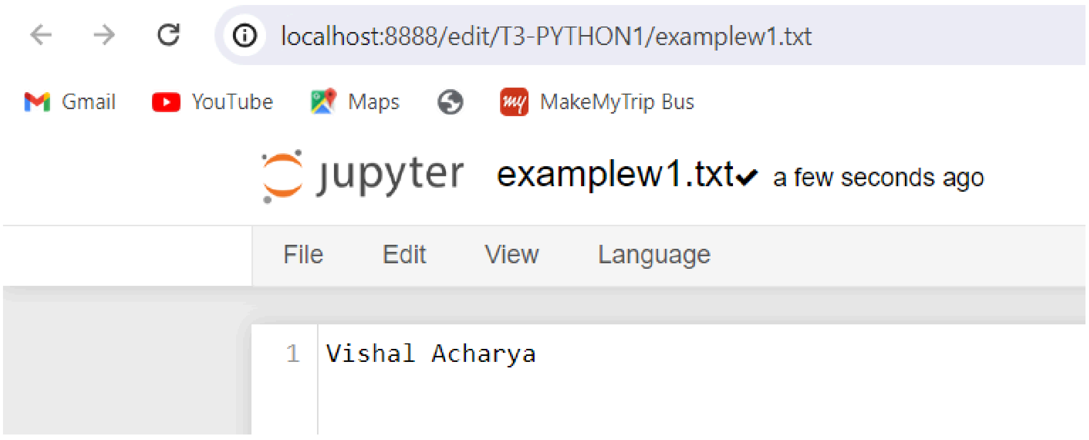
make a new file if does not exist


localhost:8888/edit/T3-PYTHON1/examplew1.txt
Gmail
YouTube
Maps
MakeMyTrip Bus
jupyter
examplew1.txt✔ a minute ago
File
Edit
View
Language

```
hello
123
```

In [16]:
```
f=open("examplew1.txt","w")
f.write("Vishal Acharya")
f.close()
```

- When you use 'w' mode and write to a file, the existing content is overwritten entirely by the new data you write into it. For instance, if you have a file named "example.txt" with some text, opening it in 'w' mode and writing new content will replace the old content with the new one.


localhost:8888/edit/T3-PYTHON1/examplew1.txt
Gmail
YouTube
Maps
MakeMyTrip Bus
jupyter
examplew1.txt✔ a few seconds ago
File
Edit
View
Language

```
Vishal Acharya
```

In [17]:
```
f=open("filew1.txt","w")
f.write("hello\n")
f.writelines(["1\n","2\n","3"])
f.close()
```

In [18]:
```
# write multiline strings
f = open('multiline.txt','w')
f.write('hello world')
f.write('\nhow are you?')
f.close()
```

In [20]:
```
# Problem with w mode
# introducing append mode
f = open('sample1.txt','a')
f.write('\nI am fine')
f.close()
```

In [21]:
```
# write lines
L = ['hello\n','hi\n','how are you\n','I am fine']

f = open("sample2.txt",'w')
f.writelines(L)
f.close()
```

three different methods for reading character data from text files:

- read(): This method reads the entire contents of the file into a string.
- readline(): This method reads a single line of the file into a string.
- readlines(): This method reads all of the lines of the file into a list of strings.

The read() method

The read() method reads a string from an open file.

Syntax:

fileObject.read([count]);

Parameter:

Count: This is the number of bytes to read from an opened file. If count is missing while trying to read as much data as possible, and perhaps until the end of the file, then this method starts reading from the beginning of the file.

sample2 txt file

hello

hi

how are you

I am fine

In [22]:
```
# reading from files
# -> using read()
f = open('sample2.txt','r')
s = f.read()
print(s)
f.close()
```

```
hello
hi
how are you
I am fine
```

In [23]:
```
# reading upto n chars
f = open('sample2.txt','r')
a = f.read(10)
b=f.read(10)
c=f.read(10)
d=f.read(10)
print(a)
print(b)
print(c)
print(d)
f.close()
```

```
hello
hi
h
ow are you

I am fine
```

File readline() method

The readline() method reads a single line from the file. The string contains a trailing newline character.

Syntax:

fileObject.readline(size)

Parameters:

size - This is the number of bytes to be read from the file.

Return value:

Lines can be read from a file with this method. A non-negative size argument represents the maximum number of bytes, including the trailing newline. An incomplete line may be returned if the size argument is not given. An empty string is returned when EOF is encountered immediately.

In [3]:
```python
# readline() -> to read line by line
f = open('sample2.txt','r')
print(f.readline())
print(f.readline())
f.close()
```

```
hello

hi
```

In [2]:
```python
# readline() -> to read line by line
f = open('sample2.txt','r')
print(f.readline(2))
y=f.readline(1)
z=f.readline()
print(y,z)
f.close()
```

```
he
l lo
```

In [6]:
```python
# readline() -> to read line by line
f = open('sample2.txt','r')
print(f.readline(5))
print(f.readline(1))
print(f.readline(1))
f.close()
```

```
hello


h
```

In [7]:
```python
# reading entire using readline
f = open('sample2.txt','r')

while True:

  data = f.readline()

  if data == '':
    break
  else:
    print(data,end='')

f.close()
```

```
hello
hi
how are you
I am fine
```

File readlines() method

The reads until EOF using readline() and returns a list containing the lines.

Syntax:

fileObject.readlines(sizehint)

Parameters:

sizehint - The number of bytes to read from the file

Return value:

The method returns a list of lines. An optional sizehint argument causes whole lines (possibly rounding up to an internal buffer size) to be read instead of reading up to EOF. An empty string is returned when EOF is encountered immediately.

In [8]:
```
f = open('sample2.txt','r')
print(f.readlines())
f.close()
```

```
['hello\n', 'hi\n', 'how are you\n', 'I am fine']
```

In [10]:
```
f = open('sample2.txt','r')
print(f.readlines(20))
f.close()
```

```
['hello\n', 'hi\n', 'how are you\n']
```

In [11]:
```
f = open('sample2.txt','r')
print(f.readlines(21),end='')
f.close()
```

```
['hello\n', 'hi\n', 'how are you\n', 'I am fine']
```

Using Context Manager (With)

It's a good idea to close a file after usage as it will free up the resources

If we dont close it, garbage collector would close it

with keyword closes the file as soon as the usage is over

In [13]:
```
# with
with open("with.txt",'w') as f:
  f.write('B2B7D1')
print(f.closed)
```

```
True
```

In [14]:
```
f=open("with.txt","w")
f.write('B2B7D1')
print(f.closed)
```

```
False
```

splitting line and words

In [15]:
```
f=open("sample2.txt")
for lines in f:
    print(lines)
    for word in lines.split():
        print(">",word)
```

```
hello

> hello
hi

> hi
how are you

> how
> are
> you
I am fine
> I
> am
> fine
```

In [16]:
```
# benefit? -> to load a big file in memory
big_L = ['hello world ' for i in range(1000)]

with open('big.txt','w') as f:
  f.writelines(big_L)
```

In [17]:
```
with open('big.txt','r') as f:

  chunk_size = 10

  while len(f.read(chunk_size)) > 0:
    print(f.read(chunk_size),end='***')
    f.read(chunk_size)
```

```
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
```

```
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
d hello wo***o world he***d hello wo***o world he***d hello wo***o world h
e***d hello wo***o world he***d hello wo***o world he***d hello wo***o wor
ld he***d hello wo***o world he***d hello wo***o world he***d hello wo***o
world he***d hello wo***o world he***d hello wo***o world he***d hello wo*
**o world he***d hello wo***o world he***d hello wo***o world he***d hello
wo***o world he***d hello wo***o world he***d hello wo***o world he***d he
llo wo***o world he***d hello wo***o world he***d hello wo***o world he***
```

In [19]:
```
f=open("samp1.txt","w")
f.write("welcome student to the world of programming")
f.truncate(5)
f.close()
f=open("samp1.txt")

x=f.read()
print(x)
```

```
welco
```

In [20]:
```
f=open("sample2.txt")
for line in f:
    print(line.strip())
```

```
hello
hi
how are you
I am fine
```

File tell() method

The tell() method returns the current position of the file read/write pointer within the file.

Syntax:

fileObject.tell()

Parameters:

NA

Return value:

The method returns the current position of the file read/write pointer.

In [21]:
```
f=open("sample2.txt")
f.read(5)
print(f.tell())
f.read(7)
print(f.tell())
f.readline()
print(f.tell())
```

```
5
14
24
```

In [56]:
```
with open("sample3.txt","w") as f:
    f.writelines(["\n","\n","\n"])
```

In [57]:
```
with open("sample3.txt","r") as f:
    print(f.tell())
    print(f.read(1))
    print(f.tell())
    print(f.read(1))
    print(f.tell())
    print(f.read(1))
    print(f.tell())
```

```
0


2


4


6
```

File seek() method

The seek() method sets the file's current position at the offset.

There are two values for the whence argument: 0, which means absolute file positioning, and 1, which means seeking relative to the current position and 2 means seeking relative to the file’s end. There is no return value.

Note:

Using either 'a' or 'a+' for opening the file for appending will undo any seek() operation at the next write.

This method is essentially no-op when an append mode file is opened, but it remains useful for apps opened in reading mode (mode 'a+').

If the file is opened in the text mode with 't', only offsets returned by tell() are valid. A different offset causes undefined behavior.

Note that not all objects are seekable.

Syntax:

fileObject.seek(offset[, whence])

Parameters:

offset- This is the position of the read/write pointer within the file.only zero support where whence=1/2

By default, 0 indicates that the position of the file is absolute, 1 means that the position is relative to the current position, and 2 means that the position is relative to the file's end.

Return value:

It does not return any value

sample2 txt file

hello

hi

how are you

I am fin

In [1]:
```
f=open("sample2.txt")
print(f.read())
print(f.read())
f.close()
```

```
hello
hi
how are you
I am fine
```

In [9]:
```
f=open("sample2.txt")
print(f.read(5))
print(f.tell())
f.seek(0,2)
print(f.tell())
print(f.read())
f.close()
```

```
hello
5
33
```

In [23]:
```
f=open("sample2.txt")
print(f.read(10))
print(f.read(5))
f.close()
```

```
hello
hi
h
ow ar
```

In [24]:
```
f=open("sample2.txt")
print(f.read(9))
f.seek(1)
print(f.read(5))
f.close()
```

```
hello
hi

ello
```

In [25]:
```
f=open("sample2.txt")
print(f.read(3))
f.seek(3)
print(f.read(5))
f.close()
```

```
hel
lo
hi
```

In [29]:
```
%%writefile penny.txt
vishal acharya
python-1
b2,b7,d1
```

```
Writing penny.txt
```

In [30]:
```
f=open("penny.txt","r")

print(f.tell())
print(f.read(10))
f.seek(0,1)
print(f.tell())
f.close()
```

```
0
vishal ach
10
```

In [31]:
```
f=open("sample2.txt","r")
print(f.tell())
print(f.read(10))
f.seek(0,2)
print(f.tell())
f.close()
```

```
0
hello
hi
h
33
```

In [32]:
```
f=open("penny.txt","r+")
f.write("vn")
print(f.tell())
print(f.read(10))
f.seek(0,1)
print(f.read())
print(f.tell())
f.seek(0)
print(f.read())
f.close()
```

```
2
shal achar
ya
python-1
b2,b7,d1

36
vnshal acharya
python-1
b2,b7,d1
```

In [33]:
```
f=open("penny.txt","a+")
f.write("vn")
print(f.tell())
print(f.read(10))
f.seek(0,1)
print(f.read())
print(f.tell())
f.seek(0)
print(f.read())
f.close()
```

```
38


38
vnshal acharya
python-1
b2,b7,d1
vn
```

In [44]:
```
%%writefile new1.txt
B2,b7 AND d1
```

```
Writing new1.txt
```

In [47]:
```
f=open("new1.txt","w+")
f.write("best")
x=f.read()
f.seek(0)
y=f.read()
f.close()
print(x,"hi")
print("*"*50)
print(y)
```

```
 hi
**************************************************
best
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: