

## ---: File Handling :---

**File handling is used to store data permanently in a file.**

**Basic operations:-**

1. **open('file\_name with extention','mode')** (default extention and mode are .txt and r)
2. Perform **read()/write()** operations.
3. **close()** the file.

**Type of files:**

1. **text file(.txt)** - It stores data/characters in ASCII form.
2. **binary(.dat)** - It is used to store audio, video, or image.
3. **csv(.csv)** – It is used to store data in key value format.

**Mode:-**

1. **'r' :- read mode**(default mode)-It open file in read mode. If file not exist then it give error of `FileNotFoundError: [Errno 2] No such file or directory: filename`.
2. **'w' :- write mode** – It open file in write mode and if in file previous data exist then it override with new data. If file not exist, it create new file.
3. **'a' :- append mode** - It open file in append mode and if in file, previous data exist then cursor position in last of the previous data. If file not exist, it create new file.
4. **'x' :- exclusive mode(Create mode)** – This mode is used to create a new file only.

**File object attributes –**

1. **closed:** It returns true if the file is closed and false when the file is open.
2. **encoding:** Encoding used for byte string conversion.
3. **mode:** Returns file opening mode
4. **name:** Returns the name of the file which file object holds.
5. **newlines:** Returns “\r”, “\n”, “\r\n”, None or a tuple containing all the newline types seen.

open	read mode	write mode	close
open()	read(n)	write()	closed
	read()	writelines()	close()
	readline()	writable()	
	readlines()		
	readable()		

open("file_name","mode")		
If file exist	Mode	If file not exist
Give error: file already exist.	"x"	Created a new file with given name.
Write mode, cursor present in zero index position. That means previous data will be destroyed.	"w"	Created a new file with given name.
Normal as read mode.	"r"	Give error: file not exist.
Normal as append mode with cursor position ahead of previous data.	"a"	Created a new file with given name.

## Create Mode examples:--

1. if file was not exist, then it create a new file with mention name

```

create.py > ...
1  f = open("n6.txt",'x')
2  print("file created")
3  print("Mode of file :",f.mode)
4  print("File closed or not :",f.closed)
5

```

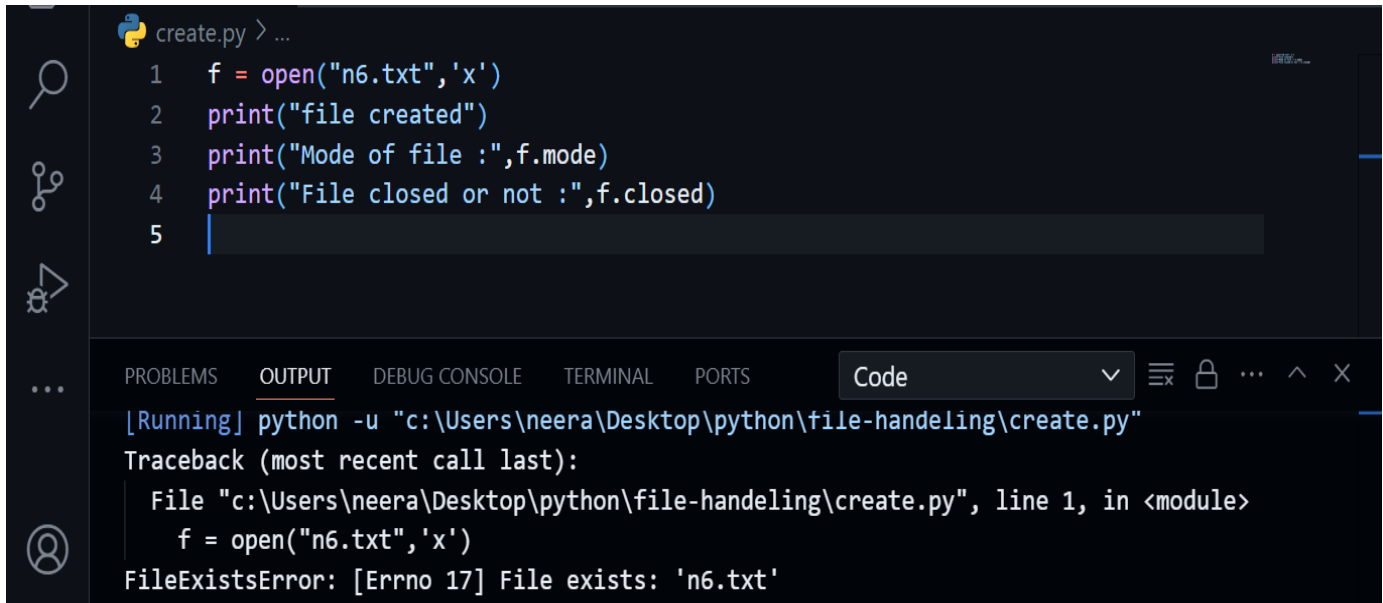
OUTPUT

```

[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\create.py"
file created
Mode of file : x
File closed or not : False

```

## 2. If file already exist, then it give error that already exist.



```
create.py > ...
1 f = open("n6.txt",'x')
2 print("file created")
3 print("Mode of file :",f.mode)
4 print("File closed or not :",f.closed)
5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\create.py"

Traceback (most recent call last):

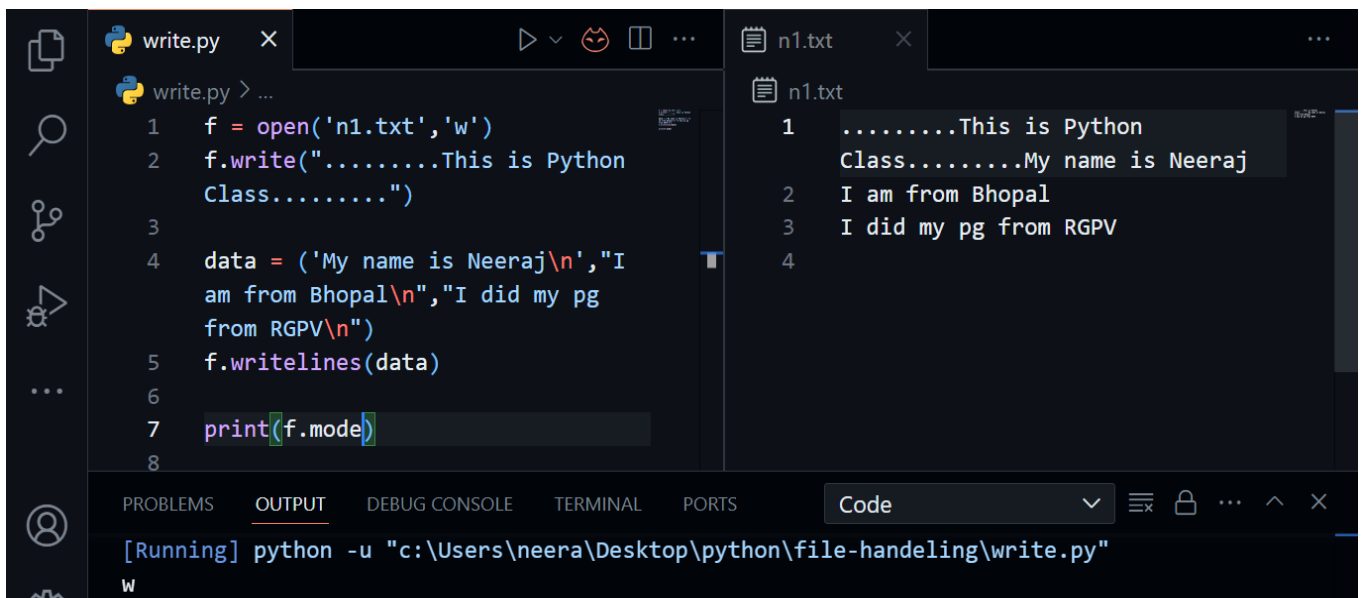
File "c:\Users\neera\Desktop\python\file-handeling\create.py", line 1, in <module>

f = open("n6.txt",'x')

FileExistsError: [Errno 17] File exists: 'n6.txt'

## Write mode example:---

### 1. if file was not exist, then it create a new file with mention name



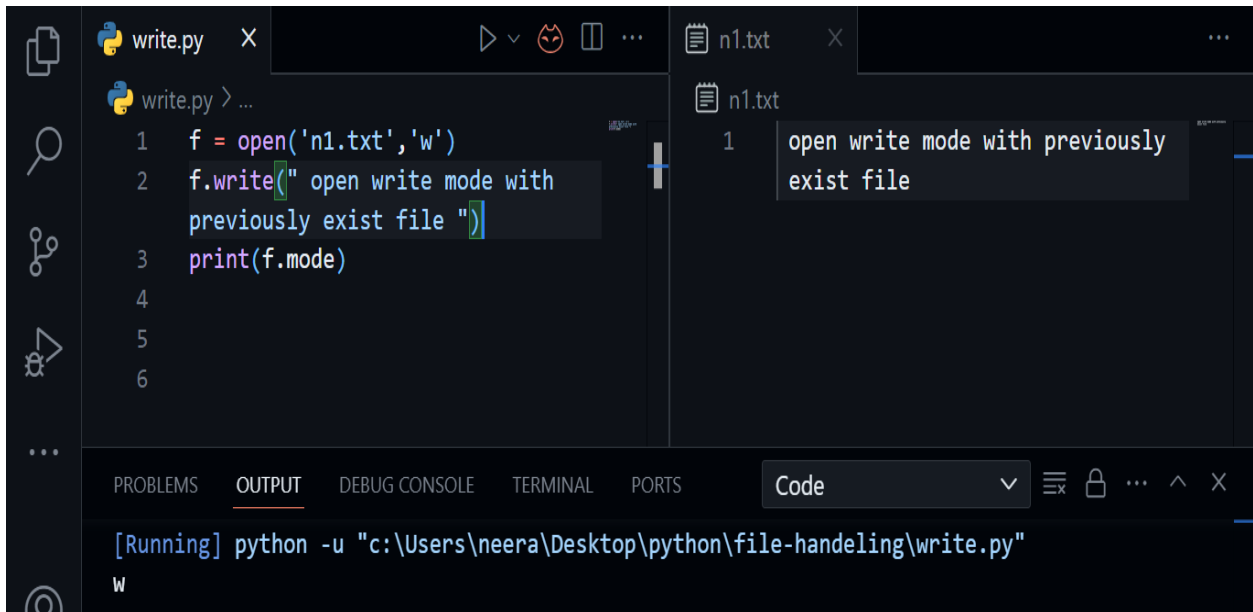
```
write.py > ...
1 f = open('n1.txt','w')
2 f.write(".....This is Python
  Class.....My name is Neeraj
  Class.....")
3
4 data = ('My name is Neeraj\n',"I
  am from Bhopal\n","I did my pg
  from RGPV\n")
5 f.writelines(data)
6
7 print(f.mode)
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\write.py"

w

2. Write mode, cursor present in zero index position. That means previous data will be destroyed.



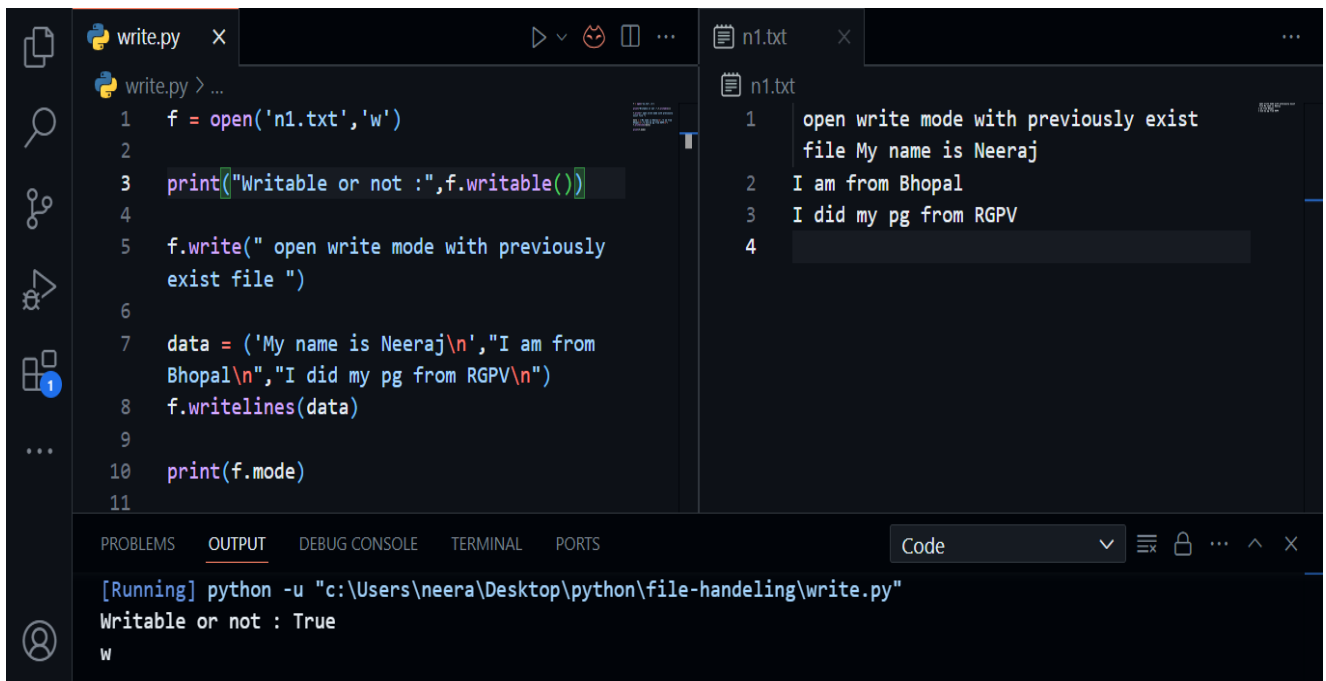
```
write.py x
write.py > ...
1 f = open('n1.txt','w')
2 f.write(" open write mode with previously exist file ")
3 print(f.mode)
4
5
6

n1.txt x
n1.txt
1 open write mode with previously exist file

[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\write.py"
W
```

### Write mode methods:

1. **write()** – it is used to write single line of data.
2. **writelines()** – It is used for multiple lines of data
3. **writable()** – To check file is writable or not.



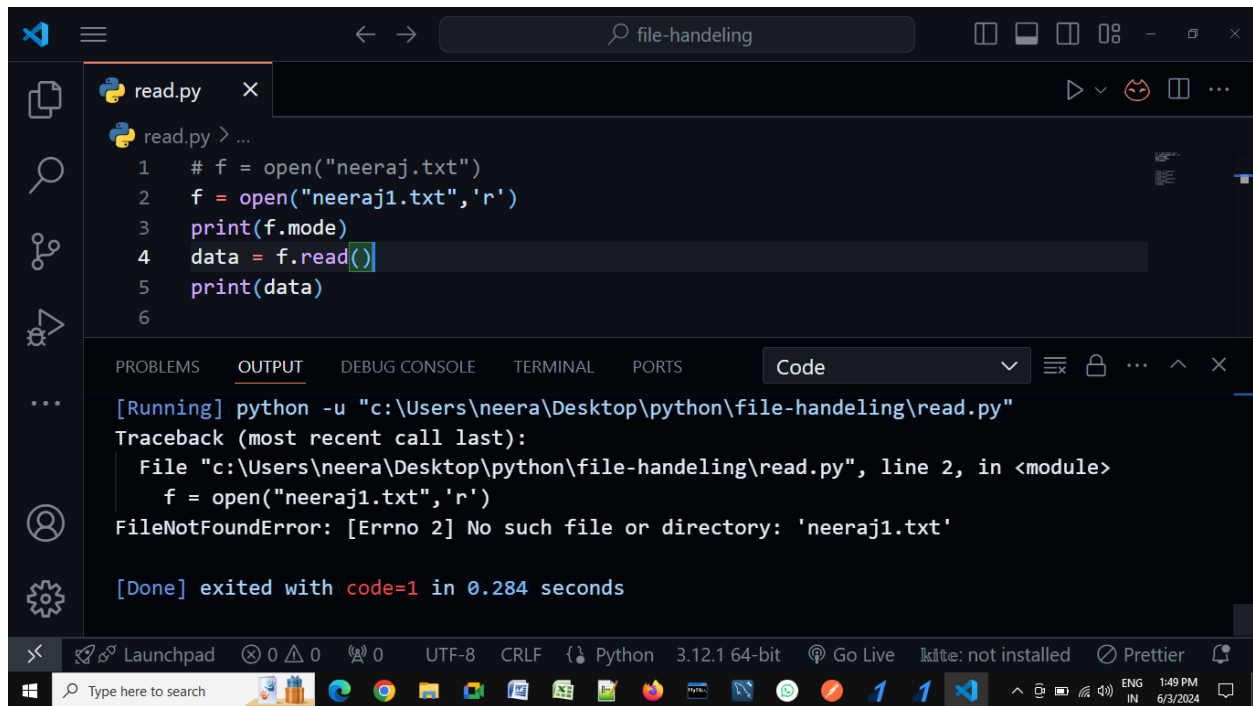
```
write.py x
write.py > ...
1 f = open('n1.txt','w')
2
3 print("Writable or not :",f.writable())
4
5 f.write(" open write mode with previously exist file ")
6
7 data = ('My name is Neeraj\n',"I am from Bhopal\n","I did my pg from RGPV\n")
8 f.writelines(data)
9
10 print(f.mode)
11

n1.txt x
n1.txt
1 open write mode with previously exist file My name is Neeraj
2 I am from Bhopal
3 I did my pg from RGPV
4

[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\write.py"
Writable or not : True
W
```

## Read mode:--

1. Give error: file not exists.



The screenshot shows the Visual Studio Code editor with a file named `read.py` open. The code in the editor is as follows:

```
1 # f = open("neeraj.txt")
2 f = open("neeraj1.txt", 'r')
3 print(f.mode)
4 data = f.read()
5 print(data)
6
```

The output window at the bottom shows the execution of the script, which results in a `FileNotFoundError` because the file `neeraj1.txt` does not exist.

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\read.py"
Traceback (most recent call last):
  File "c:\Users\neera\Desktop\python\file-handeling\read.py", line 2, in <module>
    f = open("neeraj1.txt", 'r')
FileNotFoundError: [Errno 2] No such file or directory: 'neeraj1.txt'

[Done] exited with code=1 in 0.284 seconds
```

2. If File exists, then Normal as read mode



The screenshot shows the Visual Studio Code editor with a file named `read.py` open. The code in the editor is as follows:

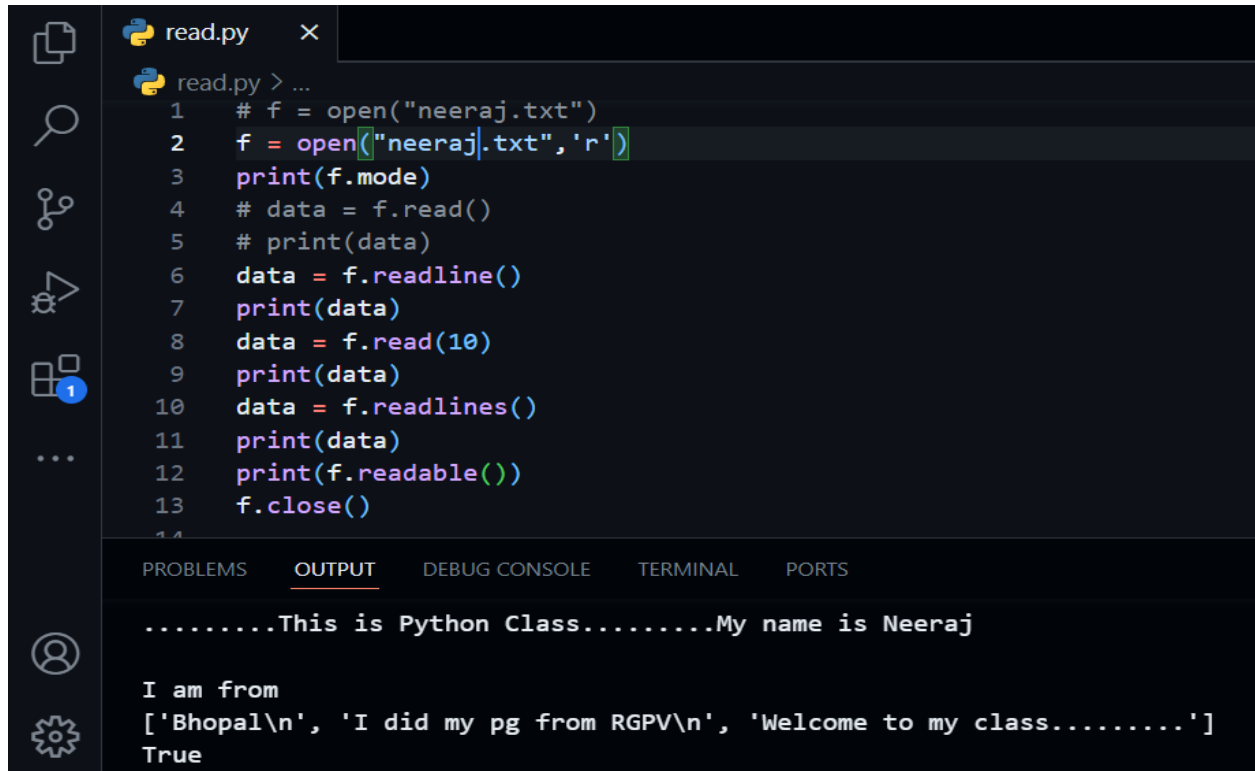
```
1 # f = open("neeraj.txt")
2 f = open("neeraj.txt", 'r')
3 print(f.mode)
4 data = f.read()
5 print(data)
6
```

The output window at the bottom shows the execution of the script, which successfully opens the file `neeraj.txt` and prints its contents.

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\read.py"
r
.....This is Python Class.....My name is Neeraj
I am from Bhopal
I did my pg from RGPV
Welcome to my class.....
```

## Read mode methods :-----

1. read(n)
2. read()
3. readline()
4. readlines()
5. readable()



```
read.py > ...
1 # f = open("neeraj.txt")
2 f = open("neeraj.txt", 'r')
3 print(f.mode)
4 # data = f.read()
5 # print(data)
6 data = f.readline()
7 print(data)
8 data = f.read(10)
9 print(data)
10 data = f.readlines()
11 print(data)
12 print(f.readable())
13 f.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

.....This is Python Class.....My name is Neeraj

I am from

['Bhopal\n', 'I did my pg from RGPV\n', 'Welcome to my class.....']

True

## Delete data, file, or folder with python:---

```
import os, shutil

os.remove('new1/n4.txt')
print(".....n4 file deleted .....")

os.remove('n4.txt')
print(".....n3 file deleted.....")

os.mkdir("new2")
print(".....new1 folder created.....")
```

```
os.chdir("new2")
print(".....change from one directory to another directory.....")

x = os.getcwd()
print(x)

os.chdir("neeraj")
print(".....change from one directory to another directory.....")

# get current working directory.....
x = os.getcwd()
print(x)

f = open('new1/n4.txt','a')
print(".....create new files within the new1 folder.....")

os.rmdir('new1')
print(".....Delete empty folder.....")

shutil.rmtree('new1')
print(".....Delete empty folder.....")

os.rename('new1',"neeraj")
print(".....Rename Folder name.....")

os.rename('n1.txt',"neeraj.txt")
print(".....Rename File name.....")
```

## tell() & seek()

**tell() :** With the help of tell() we find out the current position of cursor.



The screenshot shows a Python IDE with a file named `tellmethod.py` open. The code in the editor is as follows:

```
1 f = open("n1.txt",'r')
2
3 print("Initial position of cursor :",f.tell())
4 data = f.read(5)
5 print("New position of cursor :",f.tell())
```

The output window at the bottom shows the following output:

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\tellmethod.py"
Initial position of cursor : 0
New position of cursor : 5
```

**seek() :** With the help of seek() method, we can move cursor from our required positions.



The screenshot shows the same Python IDE with the `tellmethod.py` file. The code in the editor is as follows:

```
1 f = open("n1.txt",'r')
2
3 print("Initial position of cursor :",f.tell())
4 f.seek(10)
5 print("New position of cursor :",f.tell())
```

The output window at the bottom shows the following output:

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\tellmethod.py"
Initial position of cursor : 0
New position of cursor : 10
```

At the bottom of the IDE, a status bar indicates: `[Done] exited with code=0 in 0.323 seconds`.



## Syntax:--

**seek(attribute1, attribute2)**

attribute1 : Where we want our cursor

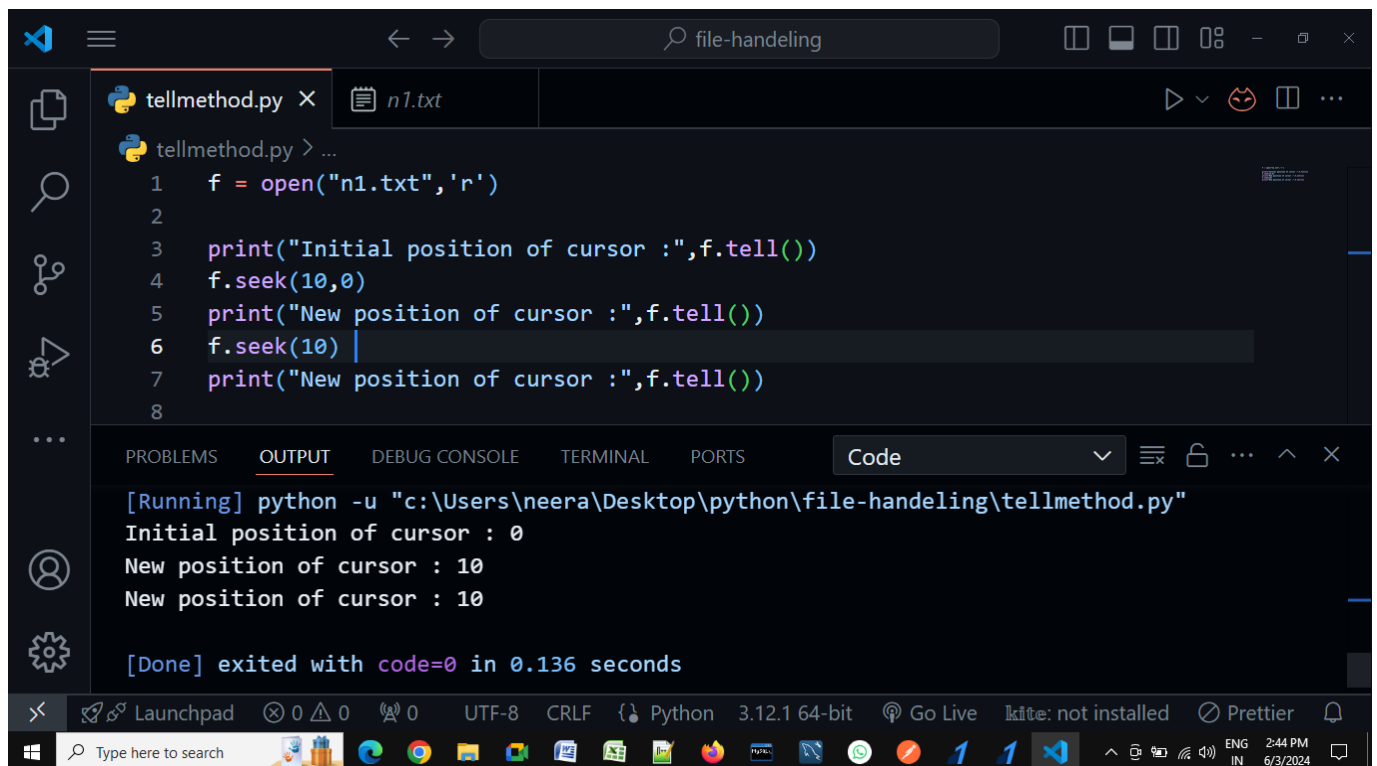
attribute2: Start from which position

1. 0 ( start from beginning )- by-default that means not required to write.
2. 1 ( start from current position )
3. 2 ( start from last position(for negative indexing))

**Note:** In python 3.2, 1 and 2 both are used only in binary mode

Examples:----

**Attribute2 : 0 ( start from beginning )-----**



The screenshot shows a Visual Studio Code editor window with a file named `n1.txt` open. The editor displays a Python script in `tellmethod.py` with the following code:

```
1 f = open("n1.txt", 'r')
2
3 print("Initial position of cursor :", f.tell())
4 f.seek(10, 0)
5 print("New position of cursor :", f.tell())
6 f.seek(10)
7 print("New position of cursor :", f.tell())
8
```

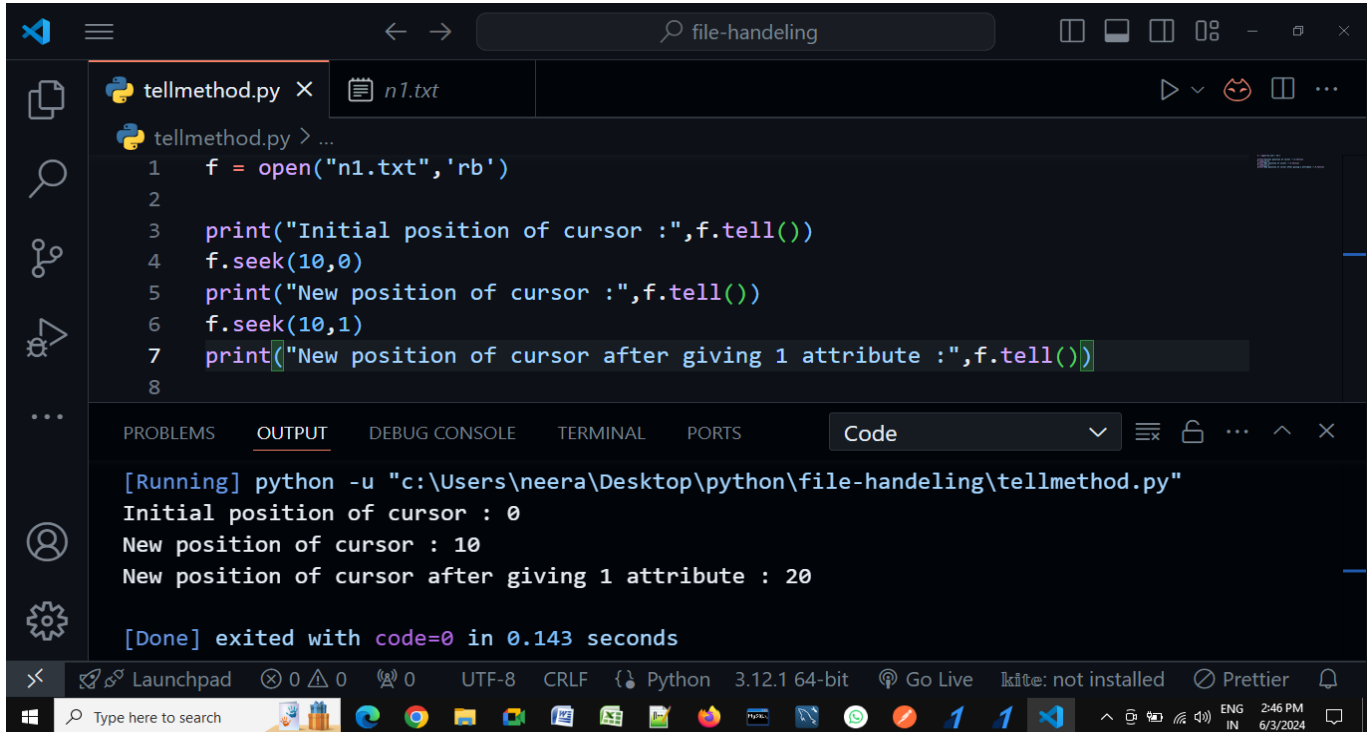
The output panel at the bottom shows the execution results:

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\tellmethod.py"
Initial position of cursor : 0
New position of cursor : 10
New position of cursor : 10

[Done] exited with code=0 in 0.136 seconds
```

The status bar at the bottom indicates the file is encoded in UTF-8, uses CRLF line endings, and is a Python 3.12.1 64-bit file. It also shows extensions like Go Live, Kite, and Prettier.

**Attribute2 : 1** (start from current position )



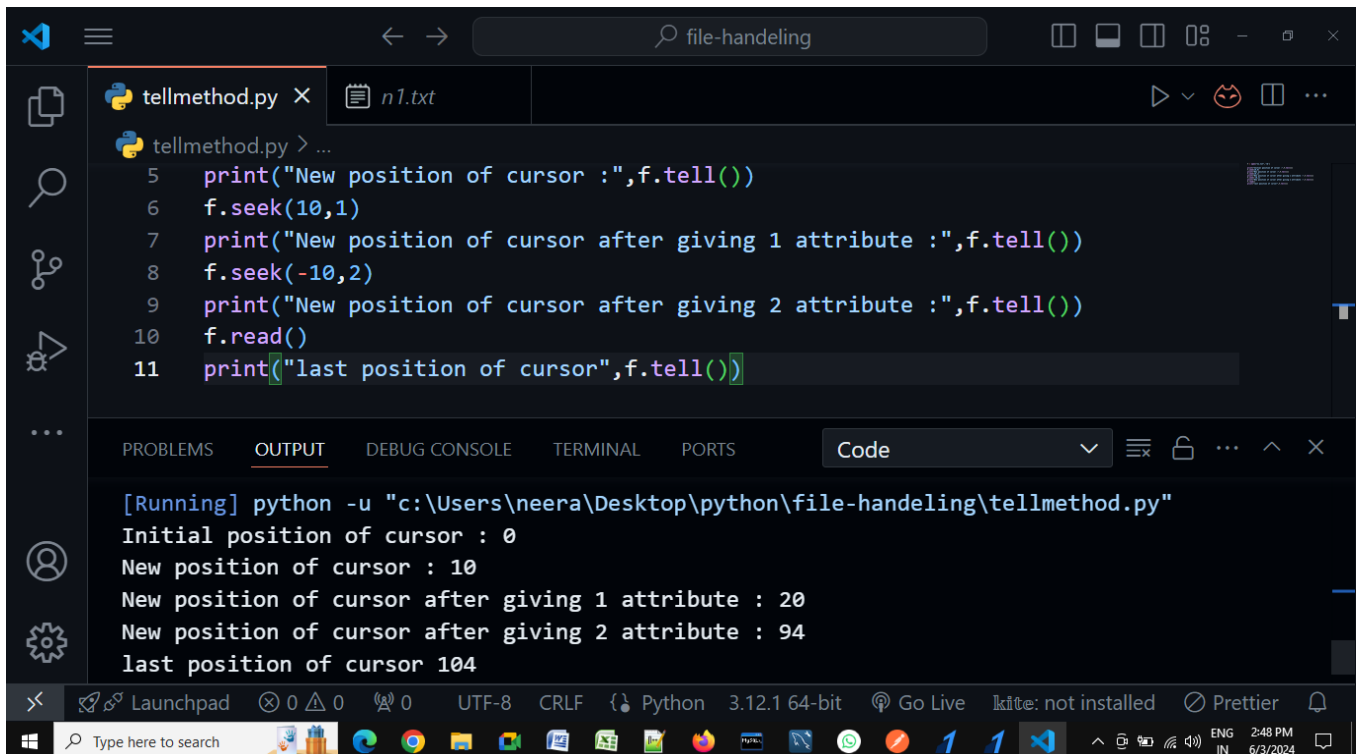
The screenshot shows the Visual Studio Code editor with a file named `tellmethod.py` open. The code in the editor is as follows:

```
1 f = open("n1.txt",'rb')
2
3 print("Initial position of cursor :",f.tell())
4 f.seek(10,0)
5 print("New position of cursor :",f.tell())
6 f.seek(10,1)
7 print("New position of cursor after giving 1 attribute :",f.tell())
8
```

The output window at the bottom shows the execution results:

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\tellmethod.py"
Initial position of cursor : 0
New position of cursor : 10
New position of cursor after giving 1 attribute : 20
[Done] exited with code=0 in 0.143 seconds
```

**Attribute2 : 2** (start from last position(for negative indexing))



The screenshot shows the Visual Studio Code editor with the same file `tellmethod.py`. The code in the editor is as follows:

```
5 print("New position of cursor :",f.tell())
6 f.seek(10,1)
7 print("New position of cursor after giving 1 attribute :",f.tell())
8 f.seek(-10,2)
9 print("New position of cursor after giving 2 attribute :",f.tell())
10 f.read()
11 print("last position of cursor",f.tell())
```

The output window at the bottom shows the execution results:

```
[Running] python -u "c:\Users\neera\Desktop\python\file-handeling\tellmethod.py"
Initial position of cursor : 0
New position of cursor : 10
New position of cursor after giving 1 attribute : 20
New position of cursor after giving 2 attribute : 94
last position of cursor 104
```

## Binary file:

Store non-simple objects like dictionaries, tuple, list, sets, nested-list objects. First all objects are serializes and after that it stores in binary\_files (like .dat ,.pkl/.pickle,.exe, .bin etc).

|----- **pickling** --> serialize --> objects converted into byte-stream.

|----- **write** --->pickle.dump(data,file)

**import pickle --**

|----- **unpickling** ->deserialize->byte-stream converted into objects.

|----- **read** ---> pickle.load(file)

**Pickling proses:-----**

```
import pickle

f = open('firstb.exe','ab+')
data = ['neeraj','jai','rahul','vishnu']
pickle.dump(data,f)
print("Data inserted successfully.....")
f.close()
```

O/P:--

Data inserted successfully.....

**Unpickling proses:-----**

```
import pickle

f = open('firstb.exe','rb+')
data = pickle.load(f)
print(data)
f.close()
```

O/P:-

['neeraj', 'jai', 'rahul', 'vishnu']