

---: 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



The screenshot shows a Python code editor with a dark theme. On the left, there are several icons for file operations like search, open, and save. The main area contains the following code:

```
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 
```

Below the code editor, the interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The OUTPUT tab is selected, showing the following terminal 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.

The screenshot shows a Python code editor with a dark theme. On the left is a sidebar with icons for search, file, terminal, and help. The main area has a file named 'create.py' open. The code attempts to create a file named 'n6.txt' in write mode ('x'). The output panel at the bottom shows the command being run and the resulting traceback, which includes the error message 'FileExistsError: [Errno 17] File exists: 'n6.txt''.

```
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

The screenshot shows a Python code editor with a dark theme. On the left is a sidebar with icons for search, file, terminal, and help. The main area has a file named 'write.py' open. The code uses 'w' mode to open a file named 'n1.txt' and writes some text to it. The output panel at the bottom shows the command being run and the contents of the newly created 'n1.txt' file, which contains the text '.....This is Python Class.....My name is Neeraj I am from Bhopal I did my pg from RGPV'.

```
write.py > ...
1 f = open('n1.txt','w')
2 f.write(".....This is Python
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)
```

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.

The screenshot shows the VS Code interface. On the left, the 'write.py' file contains the following code:

```

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

```

The 'OUTPUT' tab at the bottom shows the command being run:

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

The 'n1.txt' file on the right shows the output of the script:

```

1 open write mode with previously
2 exist file

```

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.**

The screenshot shows the VS Code interface. On the left, the 'write.py' file contains the following code:

```

1 f = open('n1.txt','w')
2
3 print("Writable or not :",f.writable())
4
5 f.write(" open write mode with previously
6 exist file ")
7
8 data = ('My name is Neeraj\n',"I am from
9 Bhopal\n","I did my pg from RGPV\n")
10 f.writelines(data)
11
12 print(f.mode)
13

```

The 'OUTPUT' tab at the bottom shows the command being run:

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

The 'n1.txt' file on the right shows the output of the script:

```

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

```

Read mode:--

1. Give error: file not exists.

The screenshot shows the VS Code interface with a Python file named 'read.py' open. The code attempts to read from a file named 'neeraj1.txt'. The output window shows a traceback indicating that the file was not found, resulting in a FileNotFoundError. The status bar at the bottom shows the file has exited with code=1 in 0.284 seconds.

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

```
[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 VS Code interface with the same 'read.py' file. This time, the file 'neeraj1.txt' exists, so the program runs successfully and prints its contents to the output window. The output shows a series of strings: 'r', followed by '.....This is Python Class.....My name is Neeraj', 'I am from Bhopal', 'I did my pg from RGPV', and 'Welcome to my class.....'.

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

```
[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()**

The screenshot shows the VS Code interface. On the left is a sidebar with icons for file operations like copy, paste, search, and refresh. The main area has tabs for 'PROBLEMS', 'OUTPUT' (which is selected), 'DEBUG CONSOLE', 'TERMINAL', and 'PORTS'. A code editor window titled 'read.py' contains the following Python code:

```
# f = open("neeraj.txt")
f = open("neeraj.txt", 'r')
print(f.mode)
# data = f.read()
# print(data)
data = f.readline()
print(data)
data = f.read(10)
print(data)
data = f.readlines()
print(data)
print(f.readable())
f.close()
```

Below the code editor is a terminal window showing the execution of the script and its output:

```
.....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.



A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file, search, and other tools. The main area has two tabs: 'tellmethod.py' and 'n1.txt'. The code editor contains the following Python script:

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

The 'OUTPUT' tab at the bottom shows the terminal 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.



A screenshot of the Visual Studio Code interface. The left sidebar shows icons for file, search, and other tools. The main area has two tabs: 'tellmethod.py' and 'n1.txt'. The code editor contains the following Python script:

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

The 'OUTPUT' tab at the bottom shows the terminal output:

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

[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)-----

```
tellmethod.py X n1.txt
tellmethod.py > ...
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
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code
[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
```

Attribute2 : 1 (start from current position)

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows two files: "tellmethod.py" and "n1.txt".
- Code Editor:** Displays the following Python code:

```
f = open("n1.txt",'rb')
print("Initial position of cursor :",f.tell())
f.seek(10,0)
print("New position of cursor :",f.tell())
f.seek(10,1)
print("New position of cursor after giving 1 attribute :",f.tell())
```
- Terminal:** Shows the output of running the script:

```
[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
```
- Status Bar:** Shows system information like battery level, network, and date/time (2:46 PM 6/3/2024).

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

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows two files: "tellmethod.py" and "n1.txt".
- Code Editor:** Displays the following Python code:

```
print("New position of cursor :",f.tell())
f.seek(10,1)
print("New position of cursor after giving 1 attribute :",f.tell())
f.seek(-10,2)
print("New position of cursor after giving 2 attribute :",f.tell())
f.read()
print("last position of cursor",f.tell())
```
- Terminal:** Shows the output of running the script:

```
[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
```
- Status Bar:** Shows system information like battery level, network, and date/time (2:48 PM 6/3/2024).

Binary file:

Store non-simple objects like dictionaries, tuple, list, sets, nested-list objects. First all objects are serialized 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 process:-----

```
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 process:-----

```
import pickle

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

O/P:-

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