Introduction

Python programs accepts the input, manipulate it & display the output. But output is availble during execution of the programs & o/p is to be entered through the keyboard.

Variables used in programs have lifetime that last till the time, program is under execution.

If we have to store data (i/p & o/p) permanently so that we can reuse it later, we uses files.

Files

A file is a collection of data stored on a secondary storage device like hard disk.

file path - file that we use are used to store on a storage medium like hard disk.

You have to give the file path to open that file.

There are 2 types of paths

- Relative Path - current working directory
  Ex file1.txt
- Absolute Path - root & complete directory list
  Ex - C:\student\documents\file1.txt

Types of files

Computer store every file as a collection of 0s & 1s i.e binary form.

Every file is basically just a series of bytes stored one after the other.

Types of files

- Text Files
- Binary Files

(1) Text Files

- A text file can be understood as a sequence of characters consisting of alphabets, numbers & other special symbol.

Ex → .txt , .csv , .py etc

- When we open a file in text editor (eg. Notepad) we see several lines of text but internally it stored in sequence of bytes consisting of 0s & 1s.
- A text file consist human readable characters.

(2) Binary Files

- Binary files are also stored in terms of bytes (0s & 1s) but unlike text files, these bytes do not represent the ASCII value of characters.
- Binary files represent the actual content such as image, audio, video, compressed version of other files.
- These files are not human readable.
- If we try to open in text editor it will show some garbage value.

Opening and closing a Text Files

Opening File

To open a file in python, we use the open() function.

syntax

```
file_object = open( file_name [, access_module ] [, buffering ] )
```

- **file_object** → This open() fun^n returns a file_object, which is stored in a variable file_object. We can use this variable to transfer data to & from the file (read & write) by calling the fun^n.

- **file_name** → The file_name argument is a string value that contains the name of the file that you want to access. If the file does not exists, it will create a new file with assigned name.

- **access_mode** → It determins the mode in which the file has to be opened.
  - read
  - write
  - append etc.
  - This is an optional paramater & the default file access mode is read (r)

- **buffering** - If the buffering values is set to 0, no buffering takes place.
  If value is set to 1, line buffering is performed while accessing file.
  If value > 1, then buffering action performed with the indicated buffer size.
  If value < 1, it will show some defaut behaviour.

File opening modes

file mode	Description	File-offset position
<r>	opens the file in read-mode only	beginning of the file.
<rb>	opens the file in binary & read mode only	beginning of the file.
<r+> or <+r>	opens the file in both read & write mode	beginning of the file.
<rb+> or <+rb>	open the file in binary read & binary write mode	beginning of the file.
<w>	open the file in write mode If file does not exist: creates a new file else: overwritten the existing file	beginning of the file
<wb>	open the file for writing in binary format. If file does not exist: creates a new file else: overwritten the existing file	beginning of the file.
<w+> or <+w>	open the file in writing & reading mode. If file does not exist: creates a new file else: overwritten the existing file	beginning of the file.
<wb+> or <+wb>	open the file for writing & reading in binary format. If file does not exist: creates a new file else: overwritten the existing file	beginning of the file

<a>	opens the file in append mode. If the file does not exist: new file will be created	end of the file.
<ab>	open the file for append in binary format. If the file does not exist: new file will be created	end of the file.
<a+> or <+a>	open the file for both appending & reading If the file does not exist: new file will be created	end of the file
<ab+> or <+ab>	open the file for both appending & reading in binary format If the file does not exist: new file will be created	end of the file.

File object Attribute

The file-object has certain attributes that tells us basic information about the file, such as -

- <file.closed>
  returns True if file is closed
- <file.mode>
  returns the access mode in which the file was opened
- <file.name>
  returns the name of the file
- <file.softspace>
  returns False if space explicitly required with print,
  True otherwise

Example

```
f = open("file.txt", "wb")
print(f.name)
print(f.closed)
print(f.mode)
print(f.softspace)
```

o/p

```
f.txt
False
wb
0
```

Closing File

- Once we are done with the read/write operations on a file, it is a good practice to close the file.
- Python provides a close() method to do so.
- while closing a file, the system frees the memory allocated to it.

syntax file_object.close()

Example -

```
# opens the file in read mode
f = open("file.txt", "r")
if f:
    print("file is opened successfully")
# closes the opened file
f.close()
```

after closing the file, we can't perform any operation.
If any exception occurs while performing some operations in the file then the program terminates without closing the file.

- we can overcome this problem using try & finally.

```
try:
    f = open("file.txt")
finally:
    f.close()
```

once the file is closed using the close() method, any attempt to use the file_object will result in an error.

flush() method

The flush() method clears the internal buffer of the file.

Files are automatically flushed while closing them. A programmer can flush file before closing it by using flush() method. syntax - file_object.flush()
(It does not require any parameters)

Example

```
# opening a file in read file
f = open("file.txt","r")
# clearing the i/p buffer
f.flush()
# reading content
fData = f.read()
print(fData)
# closing the file
f.close()
```

Opening a file using with clause

We can also open a file using with clause.

syntax

```
with open (<file-name>, access-mode) as
                                file_object :
```

- The advantage of using with clause is that any file opened using with clause is closed automatically once the control comes outside the with clause.

- In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically.

```
with open ("myfile.txt", "r+") as f :
    content = f.read()
```

Here we don't have to close the file explicitly using close() statement. Python will automatically close the file.

Reading from a text file

We can read the content of text file. Before reading a file, we must make sure that the file is opened in "r", "r+", "w+" or "a+" mode.

There are three ways to read the content of a file:

(1) read() method

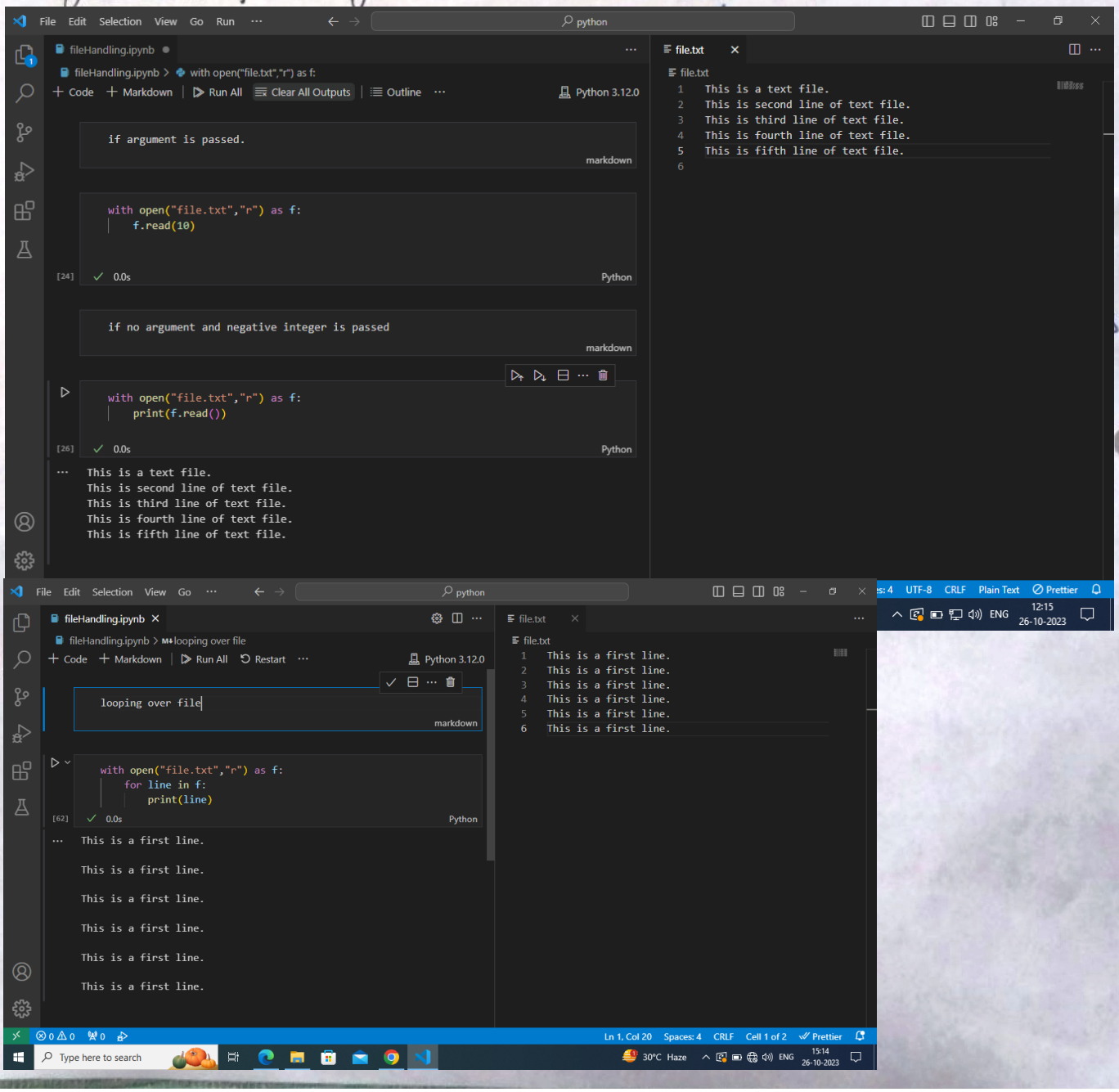
(2) readline([n]) method

(3) readlines() method

(1) The read() method

This method is used to read a specified numbers of bytes of data from a data file.

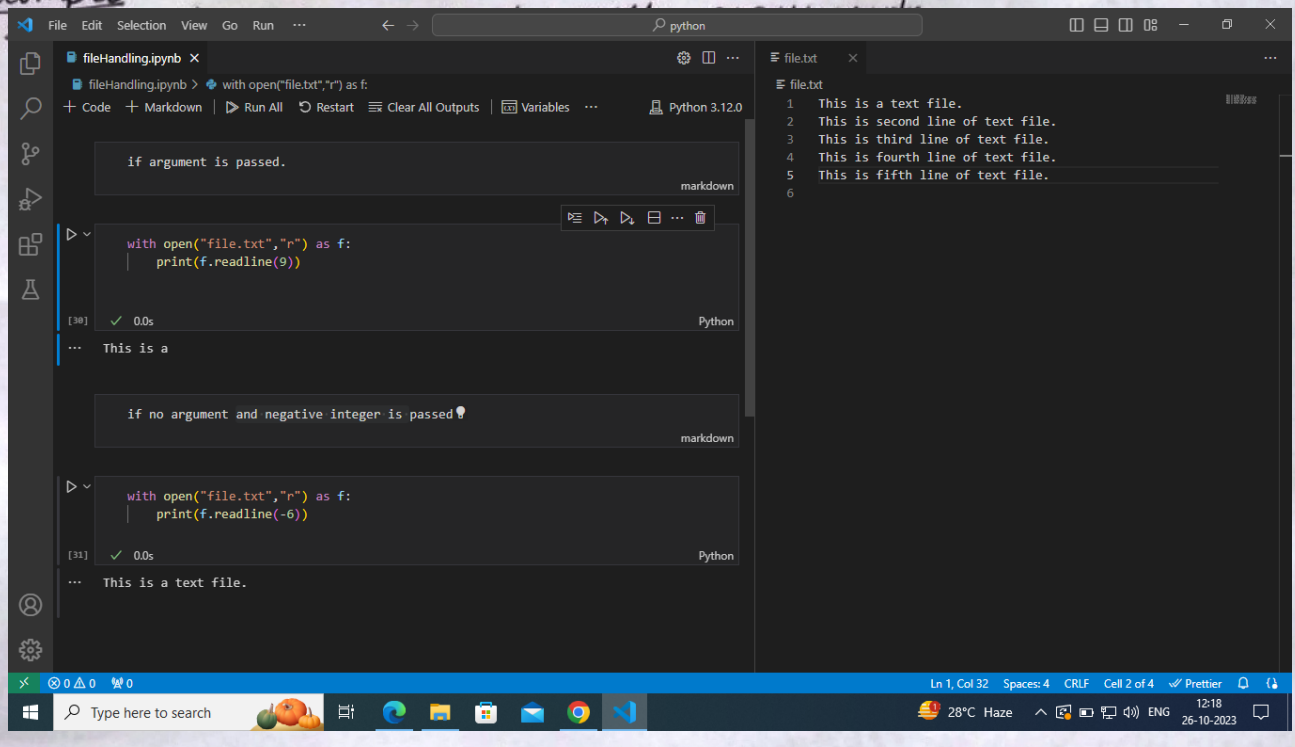
syntax file-object.read(n)



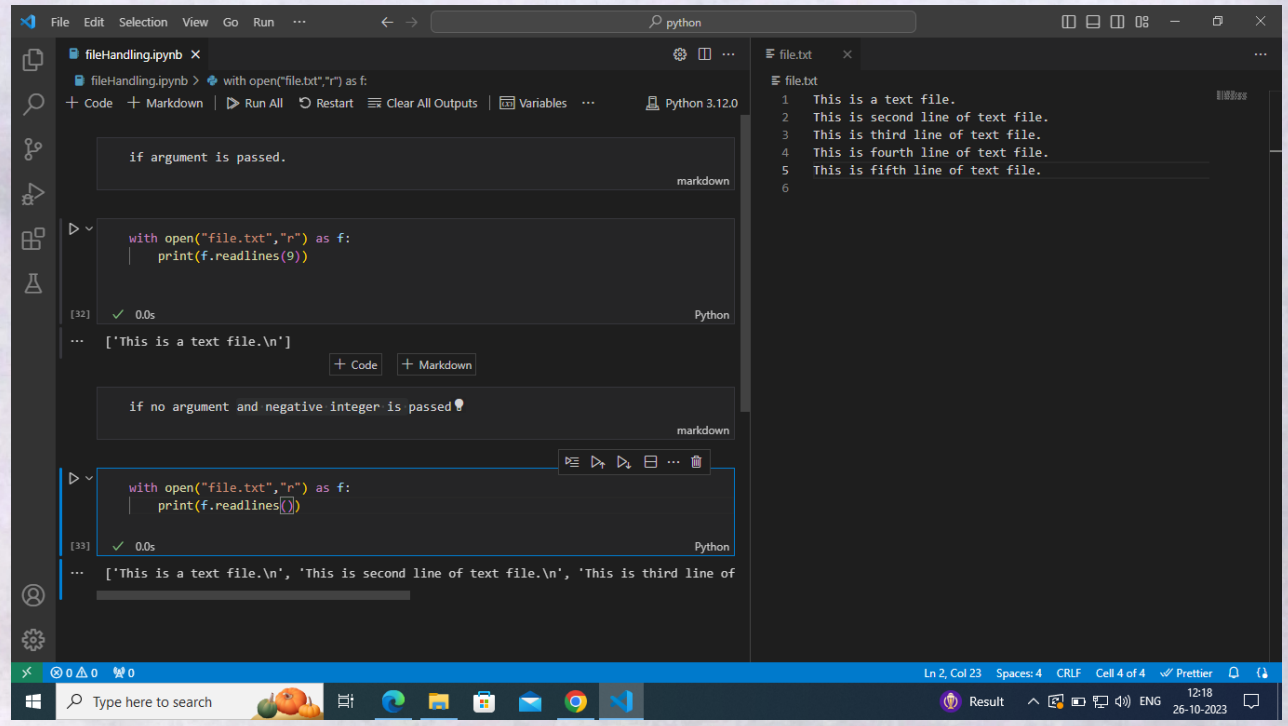
(2) readline([n]) method

This method will read the file line by line.

Example



(3) readlines() method



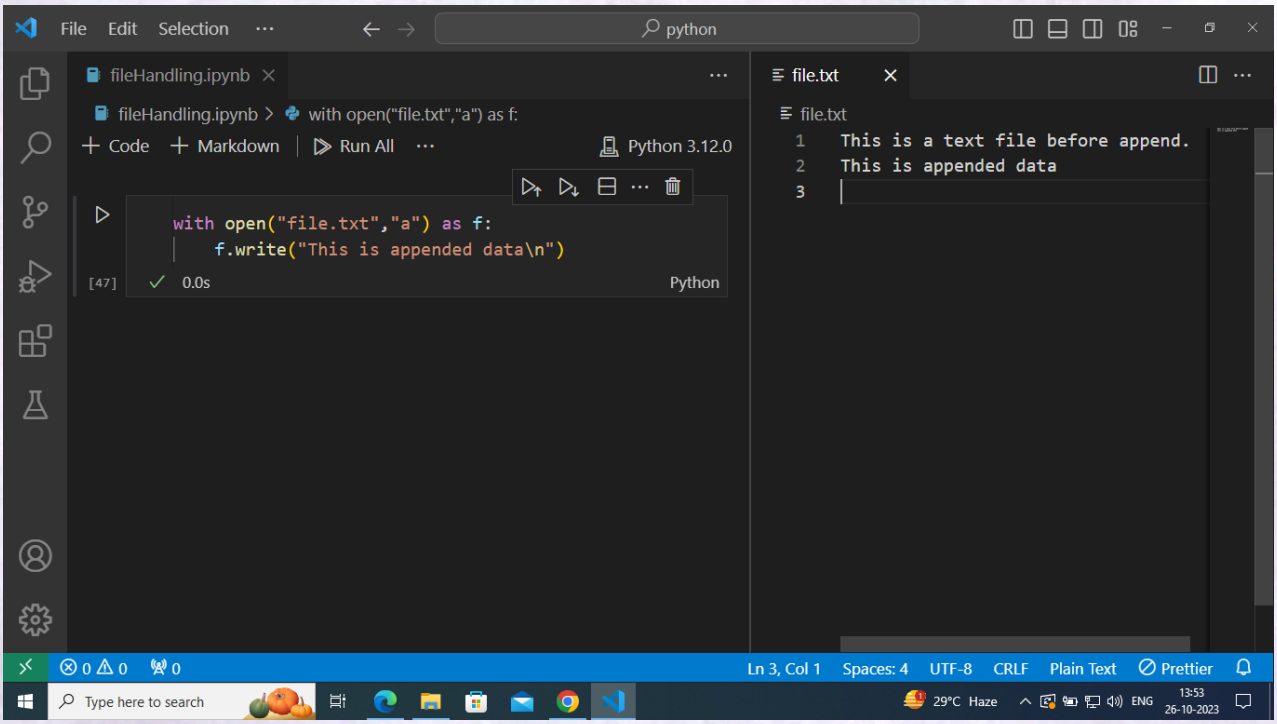
- lines in the file become members of a list, where each list element ends with a newline character ('\n').

Writing to a text file

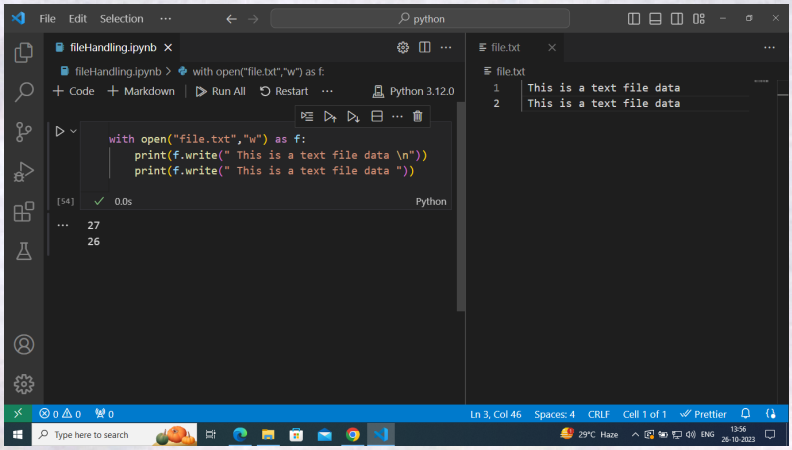
- For writing to a file, we first need to open it in write or append mode.
- If we open a file in write mode, previous data will be erased, file object will be positioned at the beginning of the file.
- If we open a file in append mode, new data will be added at the end of the previous data, file object is at the end of the file.
- There are 2 ways to write data in file-
  - write() - for writing a single string
  - writelines() - for writing a sequence of strings

(1) The write() method

open file in append mode



open file in write mode



Here, 27 & 26 are the length of string
'\n' → new line escape sequence
'\n' → treated as single character

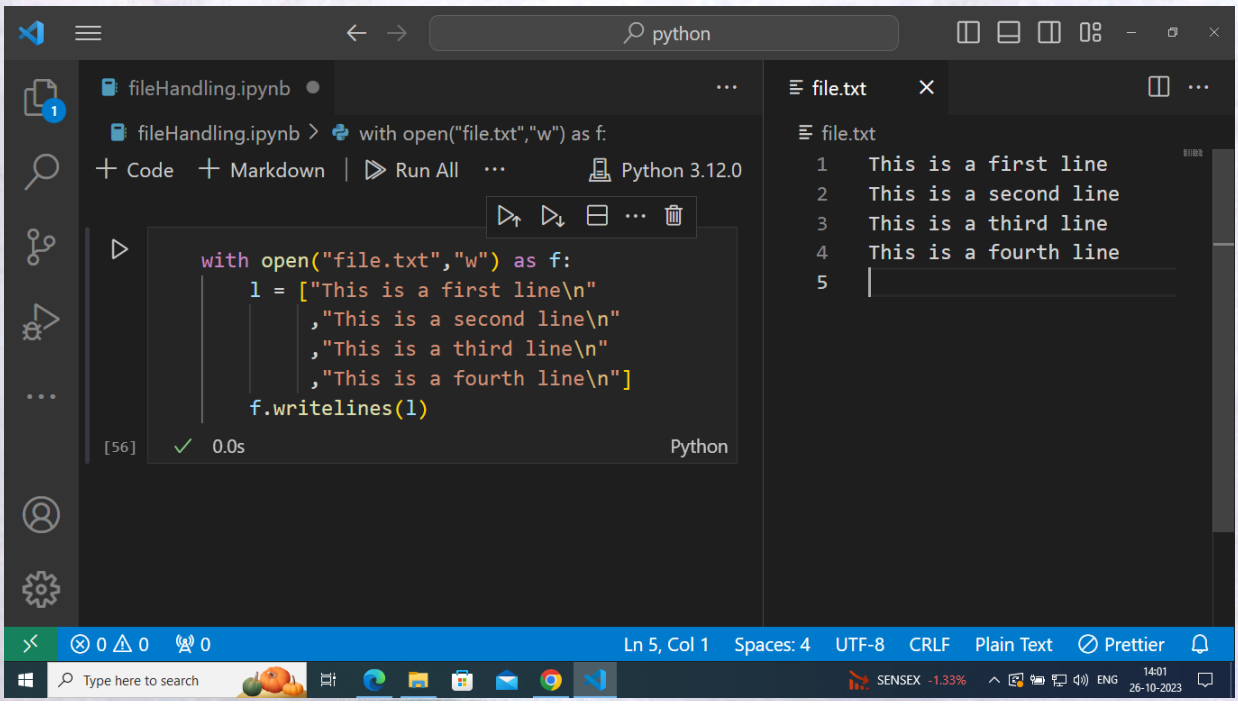
Note

The write() method writes data onto a buffer. when close() method is executed, the content from this buffer are moved to the file located in permanent storage. [This is done when we open a file without using 'with' clause.]

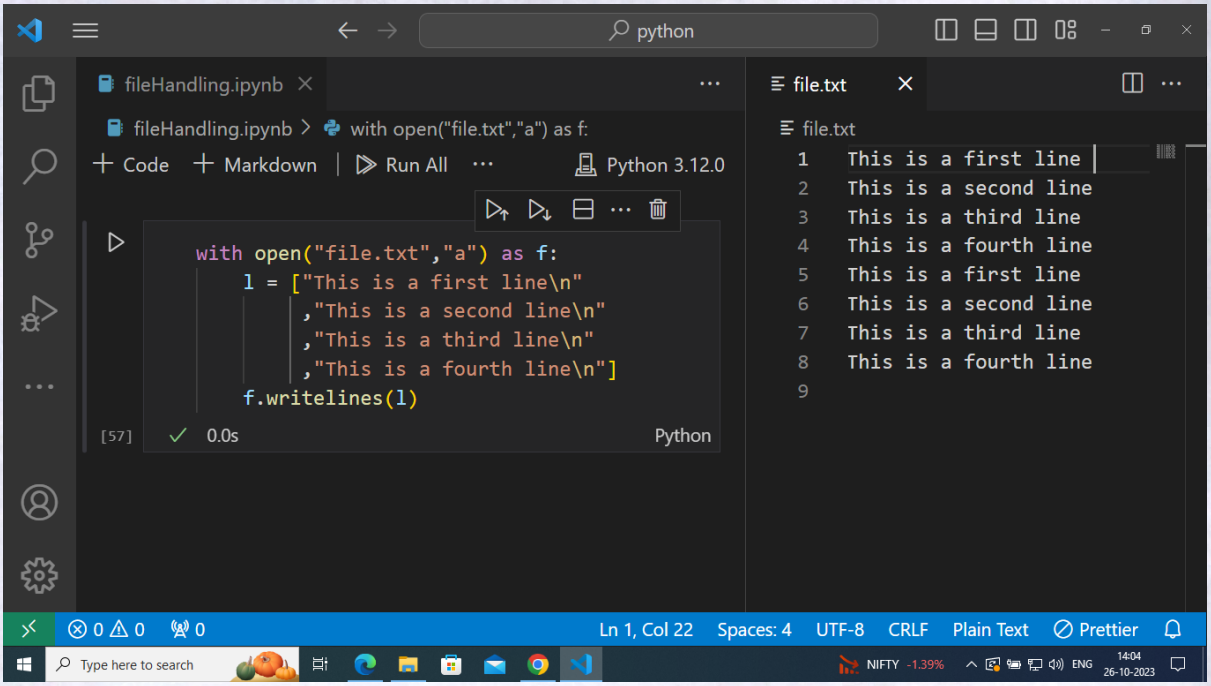
(2) writelines() method

write sequence of strings using writelines.

open file in write mode



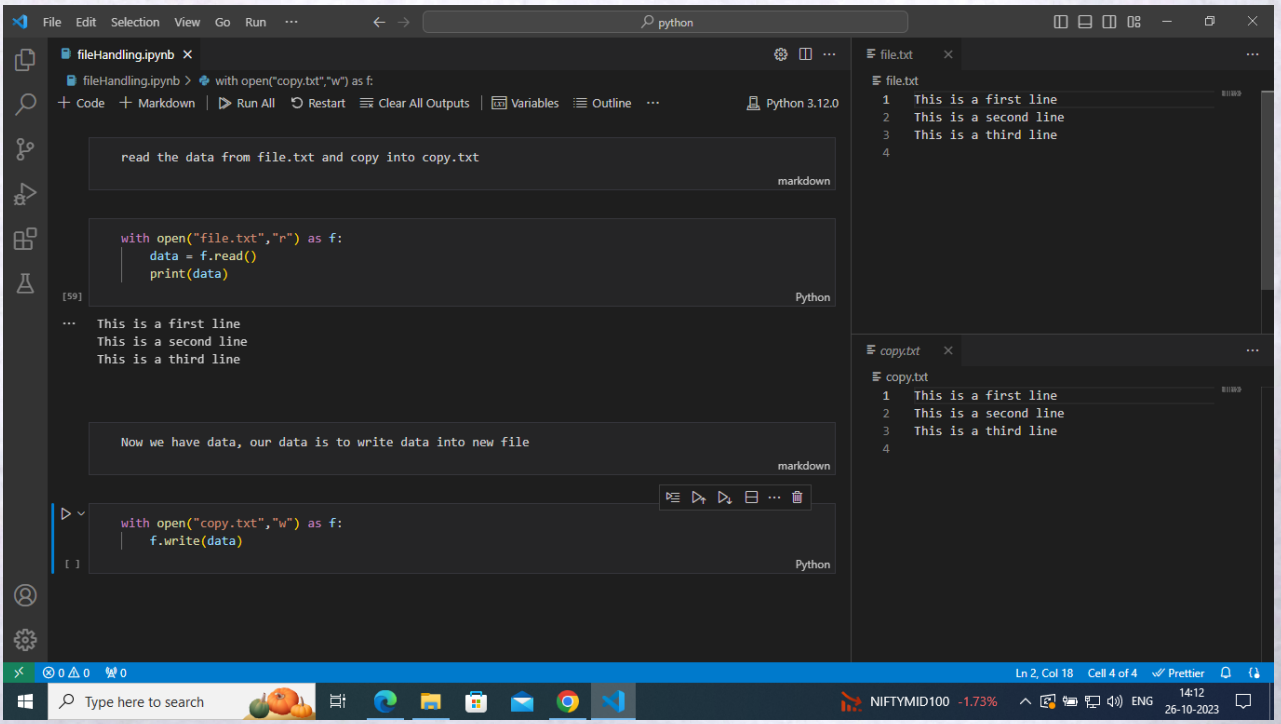
open file in append mode

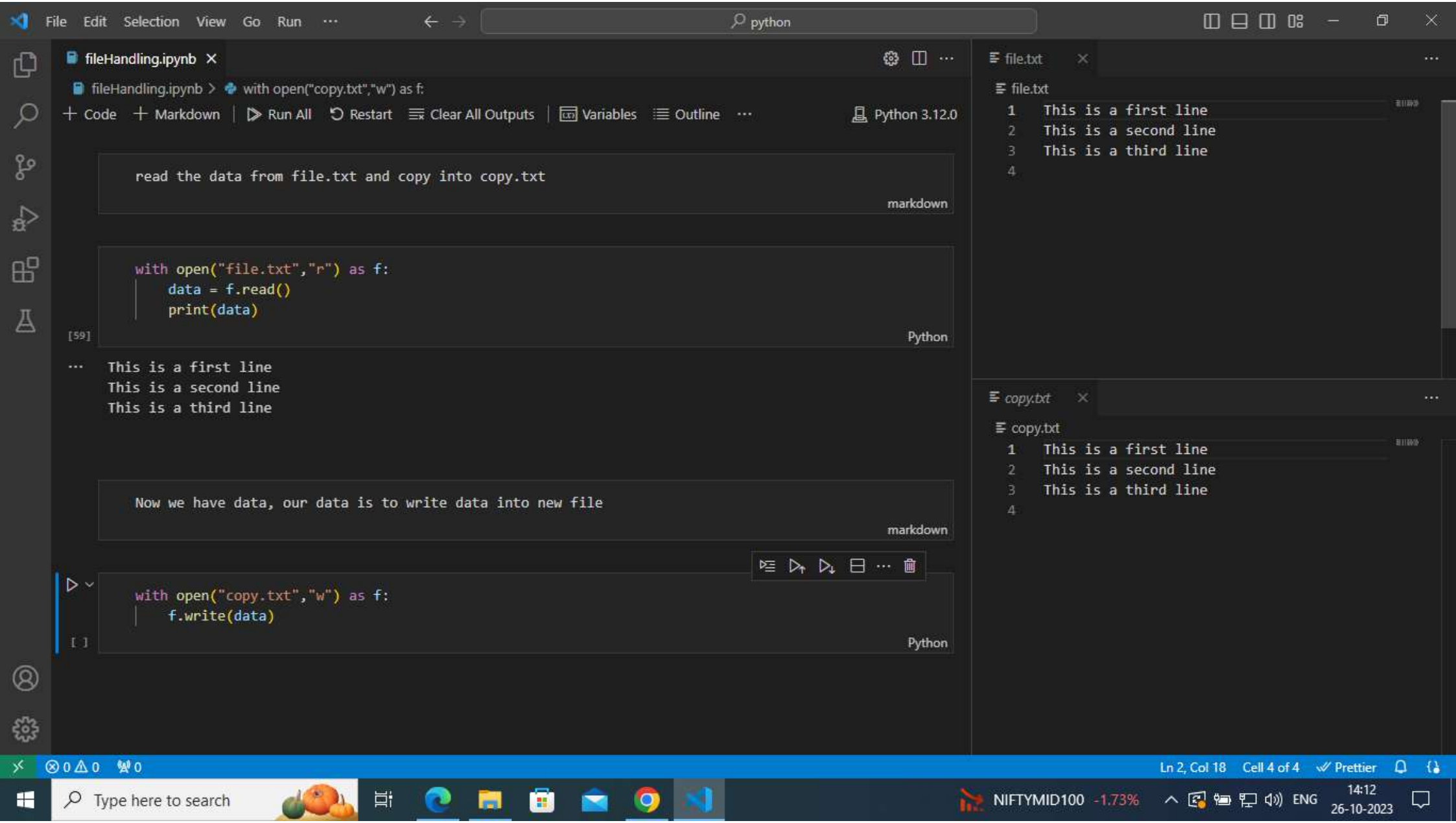


Program - Read the content from one file and copy it to another file.

OR

copy the content of the file.




File Edit Selection View Go Run ···
python
fileHandling.ipynb
fileHandling.ipynb > with open("copy.txt","w") as f:
+ Code + Markdown Run All Restart Clear All Outputs Variables Outline ···
Python 3.12.0
read the data from file.txt and copy into copy.txt
markdown
with open("file.txt","r") as f:
data = f.read()
print(data)
[59]
Python
··· This is a first line
This is a second line
This is a third line
Now we have data, our data is to write data into new file
markdown
with open("copy.txt","w") as f:
f.write(data)
[ ]
Python
file.txt
file.txt
1 This is a first line
2 This is a second line
3 This is a third line
4
copy.txt
copy.txt
1 This is a first line
2 This is a second line
3 This is a third line
4
Ln 2, Col 18 Cell 4 of 4 Prettier
Type here to search
NIFTYMID100 -1.73%
ENG
14:12
26-10-2023