FILE HANDLING in Python

- Opening a file using with clause
- Writing to a text file
    - write() method
    - writelines() method
- Reading from a file
    - read() method
    - readline() method
    - readlines() method

Opening a file using with clause

- In Python, we can also open a file using with clause.

  The syntax of with clause is:

  ```
  with open (file_name, access_mode) as file_ object:
  ```

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

Opening a file using with clause

- In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically. Also, it provides a simpler syntax.

- For e.g.)

```
with open(“myfile.txt”,”r+”) as myObject:
    content = myObject.read()
```

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

Writing to a text file

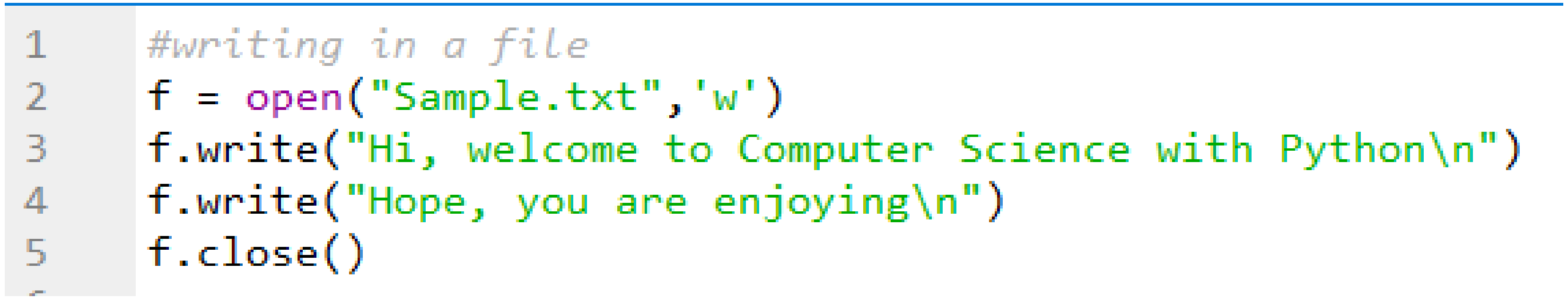
- For writing to a file, we first need to open it in write or append mode.
- If we open an existing file in write mode, the previous data will be erased, and the file object will be positioned at the beginning of the file.
- On the other hand, in append mode, new data will be added at the end of the previous data as the file object is at the end of the file.
- After opening the file, we can use the following methods to write data in the file.

  write() - for writing a single string

  writeline() - for writing a sequence of strings

Writing to a text file: write() method

- write() method takes a string as an argument and writes it to the text file.



```
#writing in a file
f = open("Sample.txt",'w')
f.write("Hi, welcome to Computer Science with Python\n")
f.write("Hope, you are enjoying\n")
f.close()
```

When we write the above statement in the interactive mode, it returns the number of characters being written on single execution of the write() method.

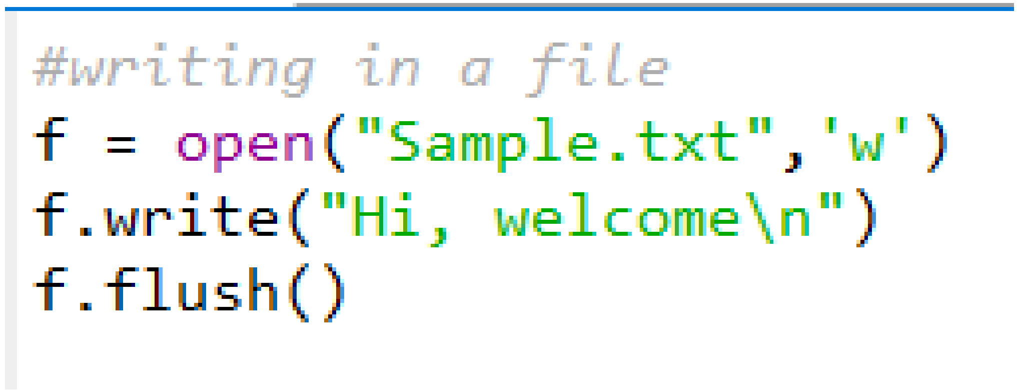
Also, we need to add a newline character (\n) at the end of every sentence to mark the end of line.

Writing to a text file: write() method

- The write() actually writes data onto a buffer. When the close() method is executed, the contents from this buffer are moved to the file located on the permanent storage.

Note:

We can also use the flush() method to clear the buffer and write contents in buffer to the file.

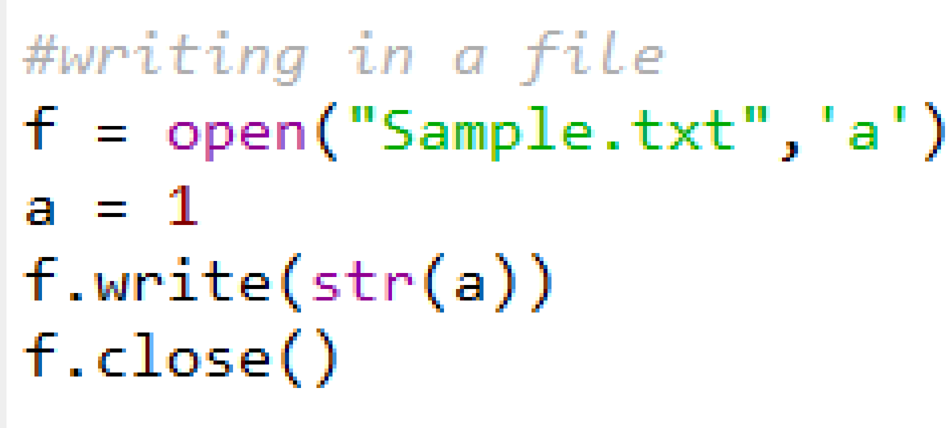


```
#writing in a file
f = open("Sample.txt",'w')
f.write("Hi, welcome\n")
f.flush()
```

Writing to a text file: write() method

- If we want to write numeric data to a text file, the data need to be converted into string before writing to the file.

for e.g.)

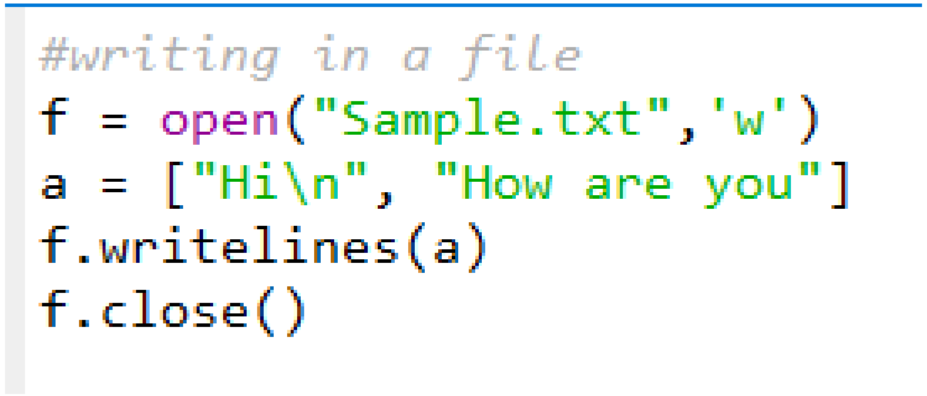


```
#writing in a file
f = open("Sample.txt",'a')
a = 1
f.write(str(a))
f.close()
```

Writing to a text file: writelines() method

- This method is used to write multiple strings to a file. We need to pass an iterable object like lists, tuple, etc. containing strings to the writelines() method.

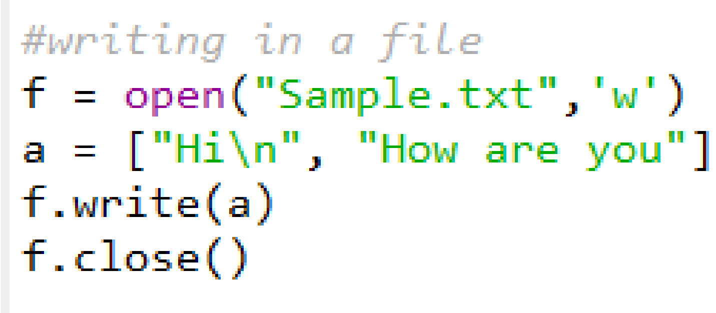
- Unlike write(), the writelines() method does not return the number of characters written in the file.



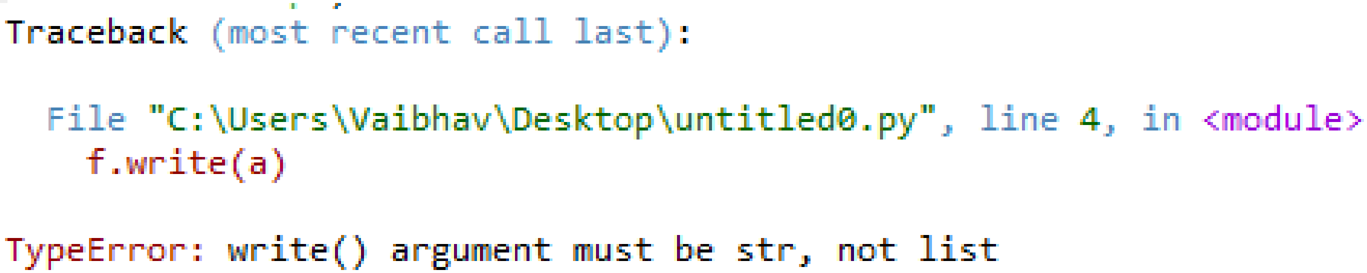
```
#writing in a file
f = open("Sample.txt",'w')
a = ["Hi\n", "How are you"]
f.writelines(a)
f.close()
```

Writing to a text file: writelines() method

- If we try to use write() method inplace of writelines() method, we will have following error:



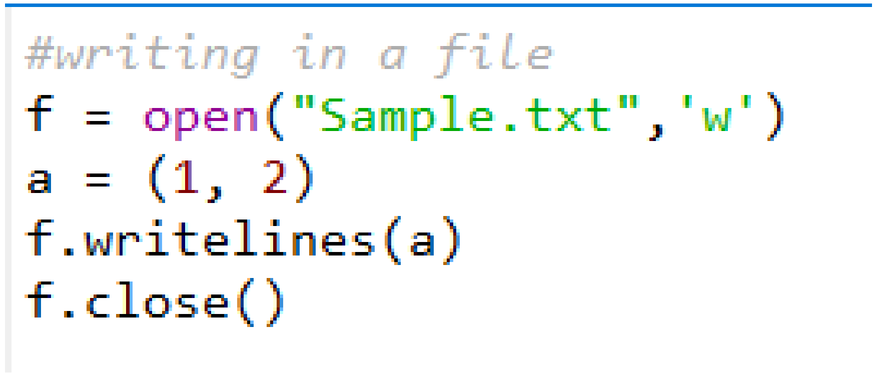
```
#writing in a file
f = open("Sample.txt",'w')
a = ["Hi\n", "How are you"]
f.write(a)
f.close()
```



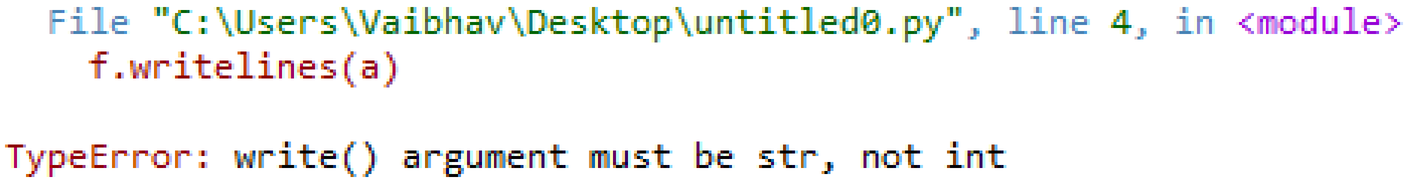
```
Traceback (most recent call last):

  File "C:\Users\Vaibhav\Desktop\untitled0.py", line 4, in <module>
    f.write(a)

TypeError: write() argument must be str, not list
```

Writing to a text file: writelines() method

- If we try to pass a tuple of numbers as an argument to writelines(), we will have following error:



```
#writing in a file
f = open("Sample.txt",'w')
a = (1, 2)
f.writelines(a)
f.close()
```



```
  File "C:\Users\Vaibhav\Desktop\untitled0.py", line 4, in <module>
    f.writelines(a)

TypeError: write() argument must be str, not int
```

Reading from a text file

- We can read the contents of a file. But 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 contents of a file:
  - read() method,
  - readline() method, and
  - readlines() method

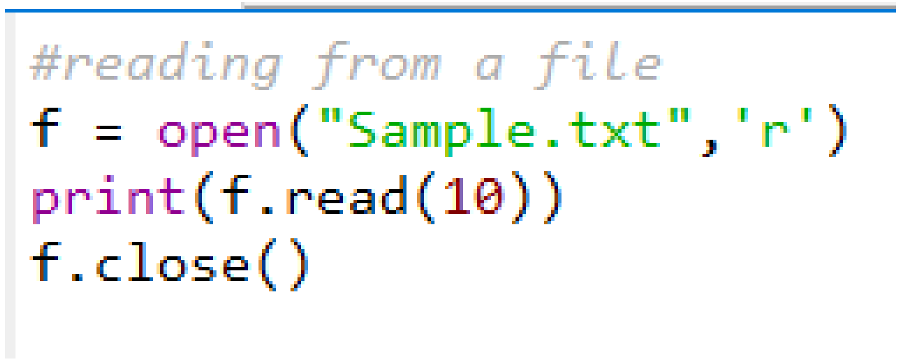
Reading from a text file: read() method

- read() method is used to read a specified number of bytes of data from a data file.
- The syntax of read() method is:

  file_object.read(n)

  where n is number of bytes of data from a data file

for e.g.)

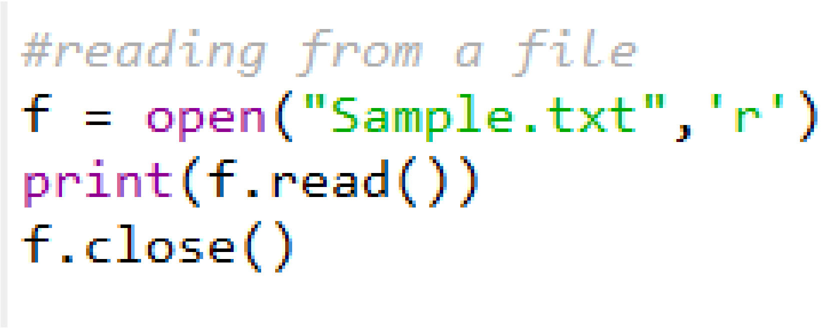


```
#reading from a file
f = open("Sample.txt",'r')
print(f.read(10))
f.close()
```

Reading from a text file: read() method

- If no argument or a negative number is specified in read(), the entire file content is read.

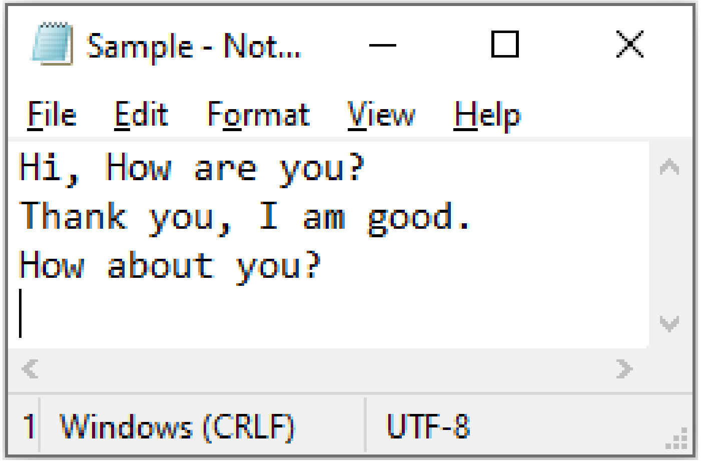
for e.g.)



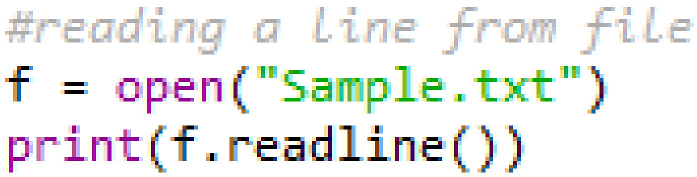
```
#reading from a file
f = open("Sample.txt",'r')
print(f.read())
f.close()
```

Reading from text file: readline() method

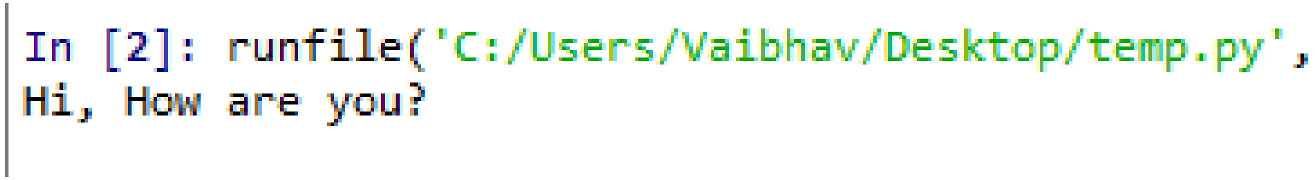
- This method reads one complete line from a file where each line terminates with a newline (\n) character.



```
Hi, How are you?
Thank you, I am good.
How about you?
```



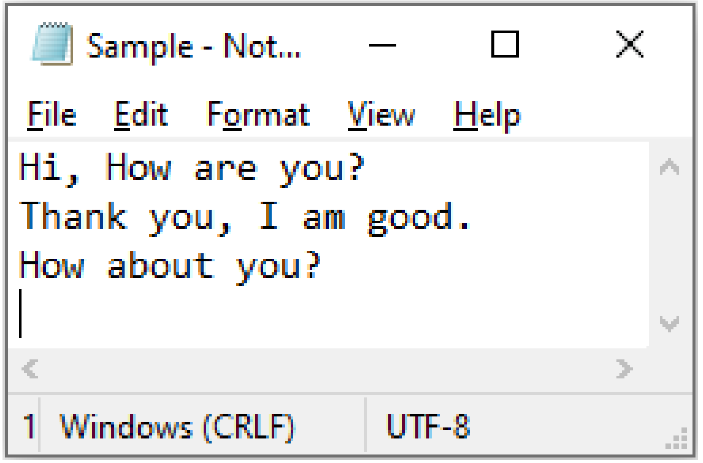
```python
#reading a line from file
f = open("Sample.txt")
print(f.readline())
```



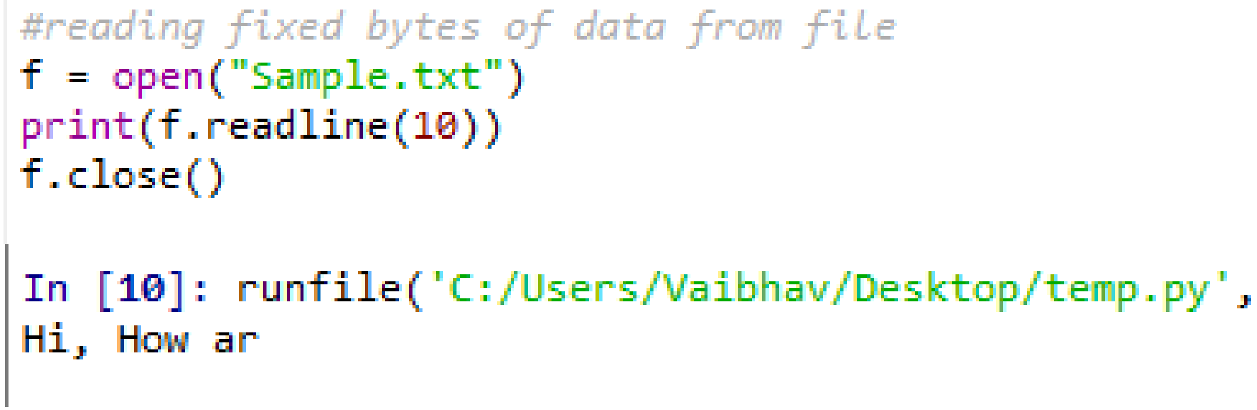
```
In [2]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
Hi, How are you?
```

Reading from text file: readline() method

- However, It can also be used to read a specified number (n) of bytes of data from a file but maximum up to the newline character (\n).



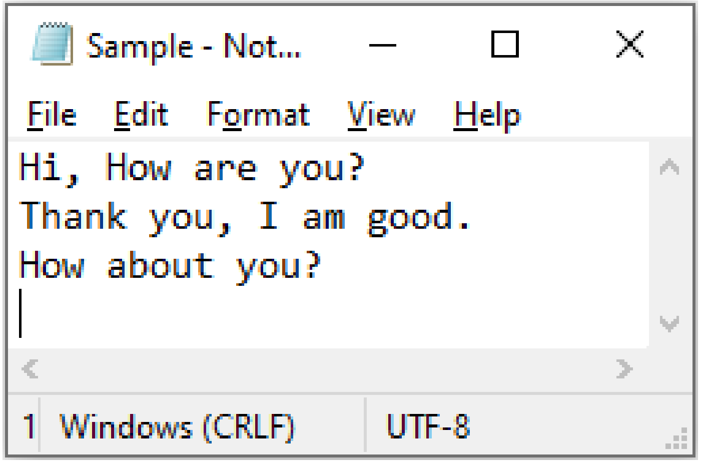
```
Hi, How are you?
Thank you, I am good.
How about you?
```



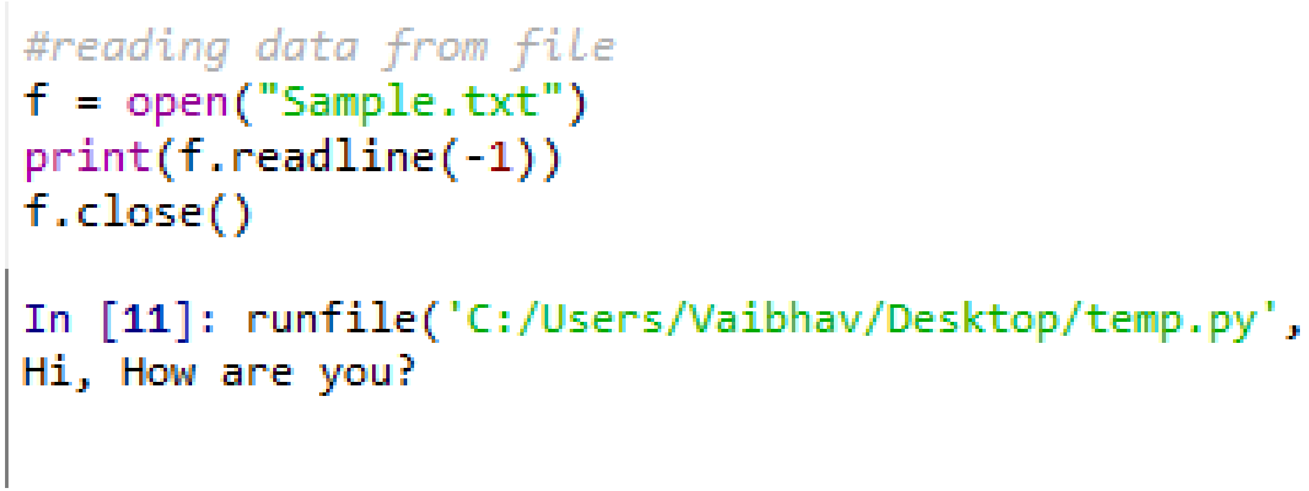
```
#reading fixed bytes of data from file
f = open("Sample.txt")
print(f.readline(10))
f.close()

In [10]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
Hi, How ar
```

Reading from text file: readline() method

- If no argument or a negative number is specified, it reads a complete line and returns string.



```
Hi, How are you?
Thank you, I am good.
How about you?
```

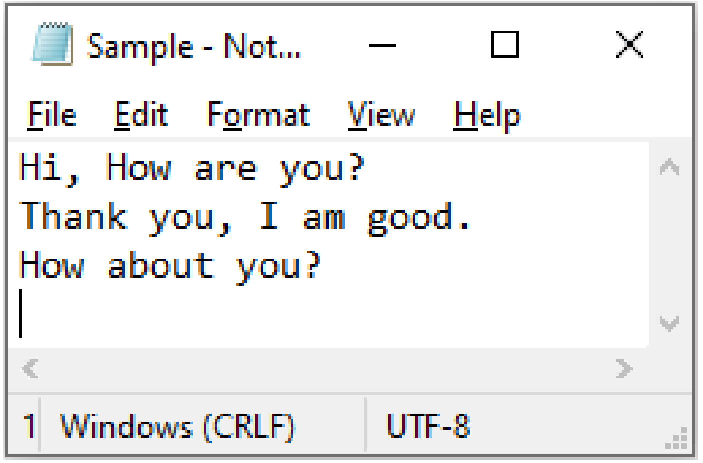


```
#reading data from file
f = open("Sample.txt")
print(f.readline(-1))
f.close()
```

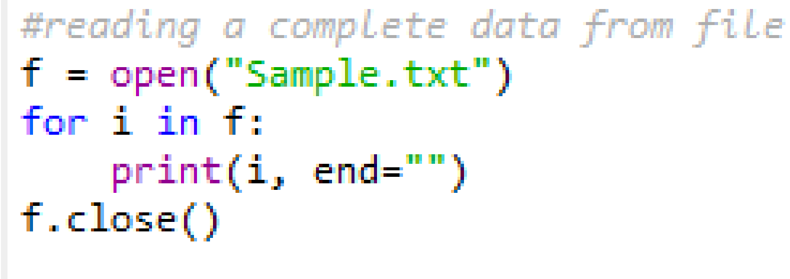
```
In [11]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
Hi, How are you?
```

Reading from text file: readline() method

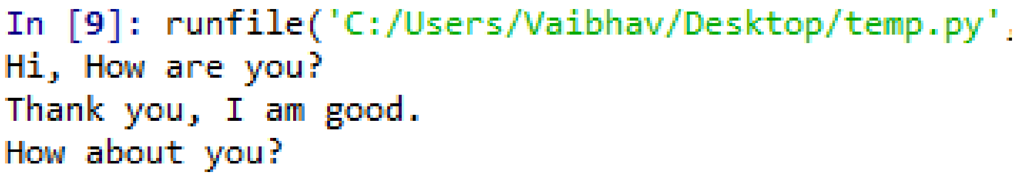
- To read the entire file line by line using the readline(), we can use a loop.



```
Hi, How are you?
Thank you, I am good.
How about you?
```



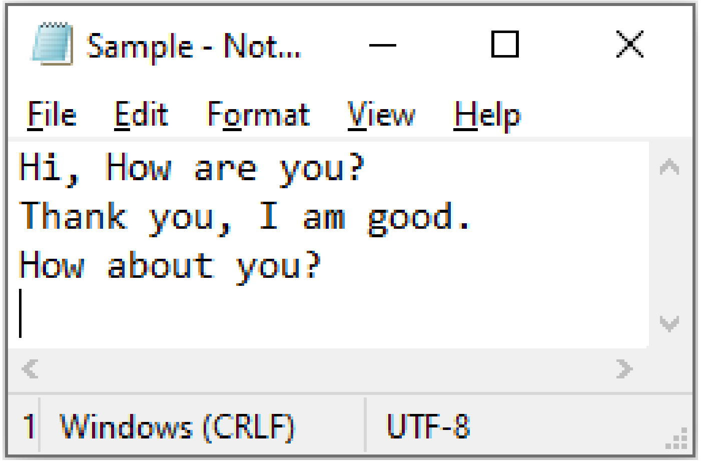
```
#reading a complete data from file
f = open("Sample.txt")
for i in f:
    print(i, end="")
f.close()
```



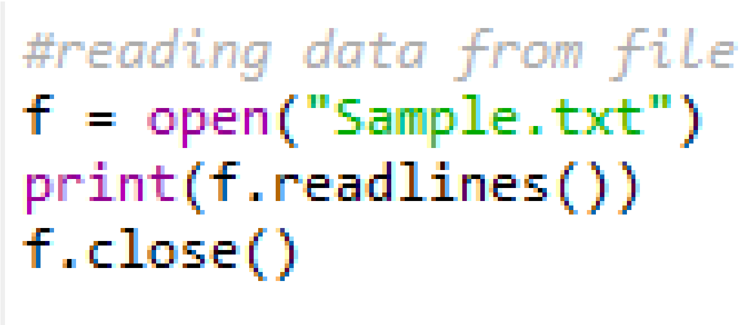
```
In [9]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
Hi, How are you?
Thank you, I am good.
How about you?
```

Reading from text file: readlines() method

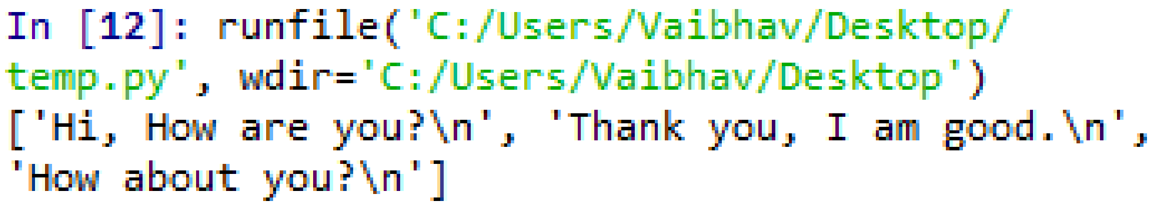
- The method reads all the lines and returns the lines along with newline as a list of strings.



```
Hi, How are you?
Thank you, I am good.
How about you?
```



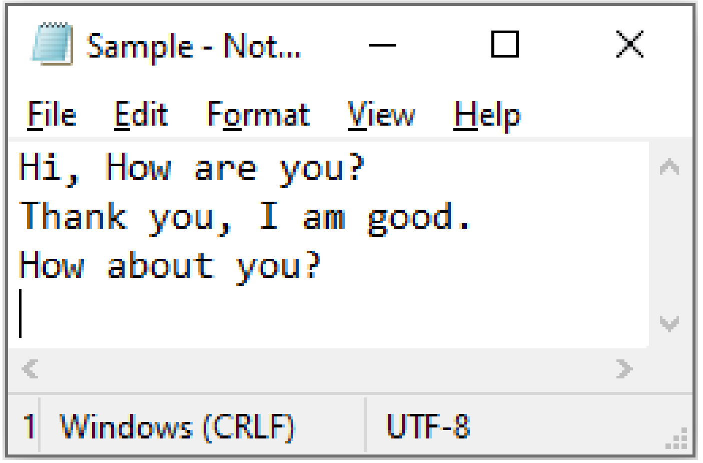
```python
#reading data from file
f = open("Sample.txt")
print(f.readlines())
f.close()
```



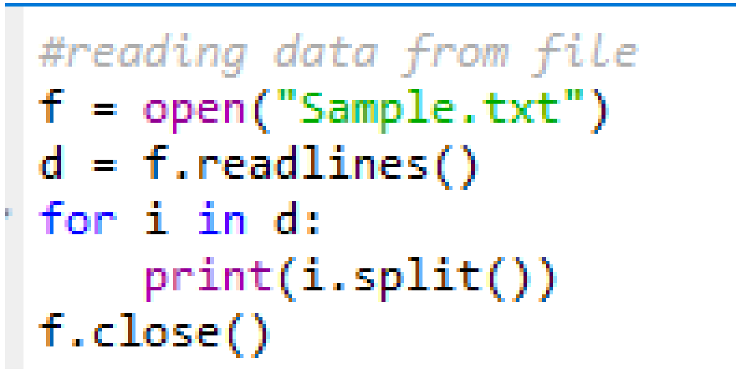
```
In [12]: runfile('C:/Users/Vaibhav/Desktop/
temp.py', wdir='C:/Users/Vaibhav/Desktop')
['Hi, How are you?\n', 'Thank you, I am good.\n',
'How about you?\n']
```

Reading from text file: readlines() method

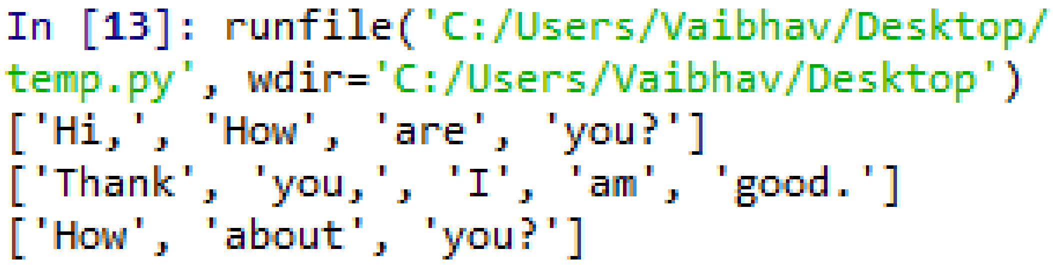
- In case we want to display each word of a line separately as an element of a list, then we can use split() function.



```
Hi, How are you?
Thank you, I am good.
How about you?
```



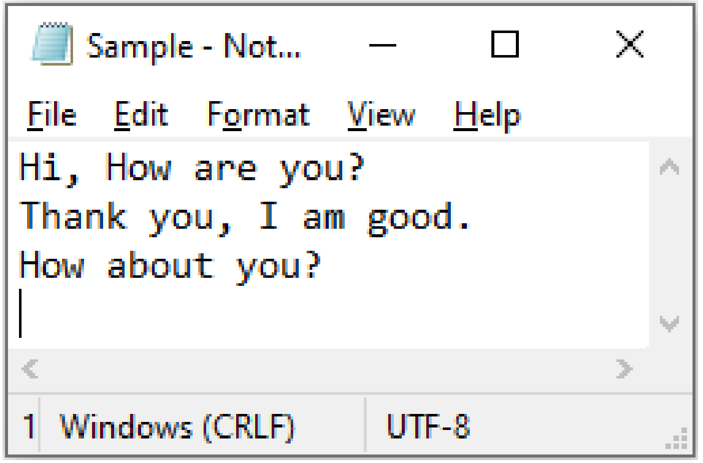
```python
#reading data from file
f = open("Sample.txt")
d = f.readlines()
for i in d:
    print(i.split())
f.close()
```



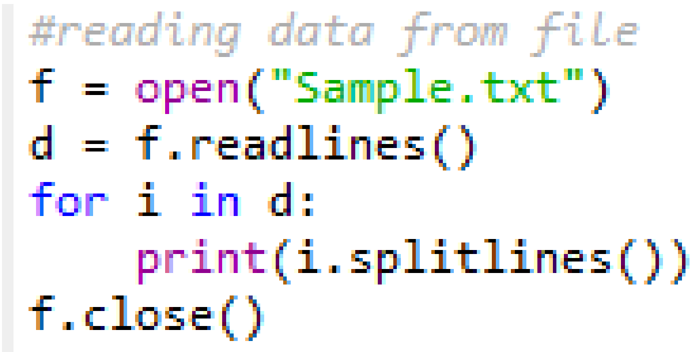
```
In [13]: runfile('C:/Users/Vaibhav/Desktop/
temp.py', wdir='C:/Users/Vaibhav/Desktop')
['Hi,', 'How', 'are', 'you?']
['Thank', 'you,', 'I', 'am', 'good.']
['How', 'about', 'you?']
```

Reading from text file: readlines() method

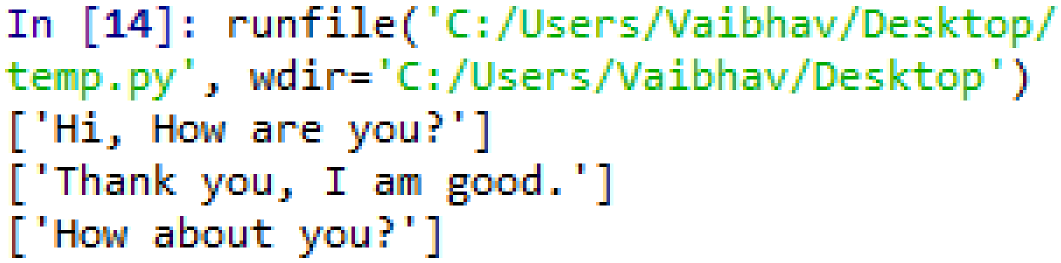
- However, if *splitlines()* is used instead of split(), then each line is returned as element of a list,



```
Hi, How are you?
Thank you, I am good.
How about you?
```



```
#reading data from file
f = open("Sample.txt")
d = f.readlines()
for i in d:
    print(i.splitlines())
f.close()
```



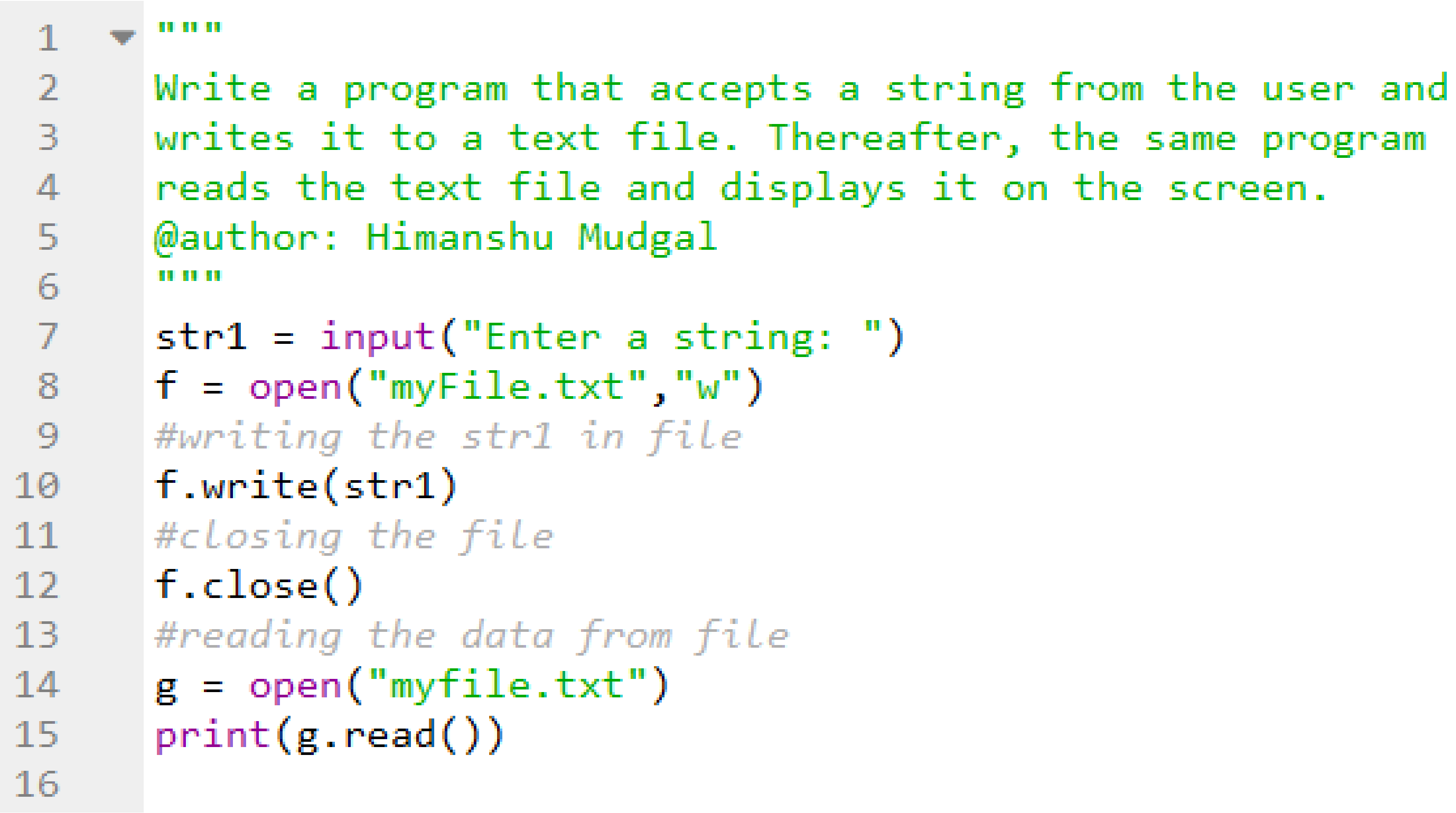
```
In [14]: runfile('C:/Users/Vaibhav/Desktop/
temp.py', wdir='C:/Users/Vaibhav/Desktop')
['Hi, How are you?']
['Thank you, I am good.']
['How about you?']
```

Program on file handling

- Sample Program:

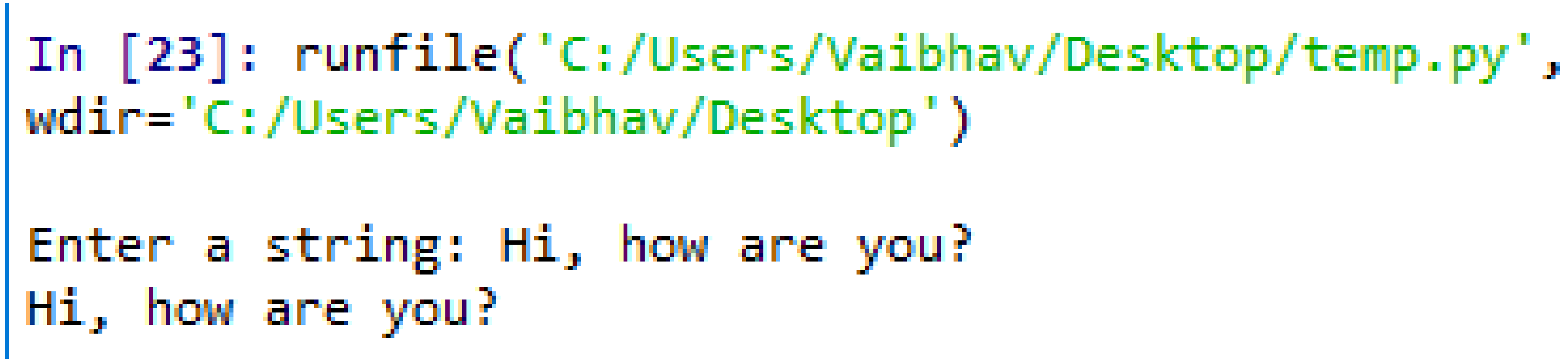
  Write a program that accepts a string from the user and writes it to a text file. Thereafter, the same program reads the text file and displays it on the screen.

Code:



```python
"""
Write a program that accepts a string from the user and
writes it to a text file. Thereafter, the same program
reads the text file and displays it on the screen.
@author: Himanshu Mudgal
"""
str1 = input("Enter a string: ")
f = open("myFile.txt","w")
#writing the str1 in file
f.write(str1)
#closing the file
f.close()
#reading the data from file
g = open("myfile.txt")
print(g.read())

```

Output:



```
In [23]: runfile('C:/Users/Vaibhav/Desktop/temp.py',
wdir='C:/Users/Vaibhav/Desktop')

Enter a string: Hi, how are you?
Hi, how are you?
```