Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

FILE HANDLING - 2

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

1. **Writing Data to Text Files**
2. **Reading Data from a Text File**

2.4 Writing data to text files:

- For writing data 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.

1. **write()** - for writing a single string
2. **writelines()** - for writing a sequence of strings

2.4.1 The write() method:

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

Consider the following piece of code:

```
>>> myobject=open("myfile.txt",'w')

>>> myobject.write("Hey I have started#using files in Python\n")

41

>>> myobject.close()
```

- On execution, write() returns the number of characters written on to the file. Hence, 41, which is the length of the string passed as an argument, is displayed.

Note: '\n' is treated as a single character.

- If numeric data are to be written to a text file, the data need to be converted into string before writing to the file.

For example:

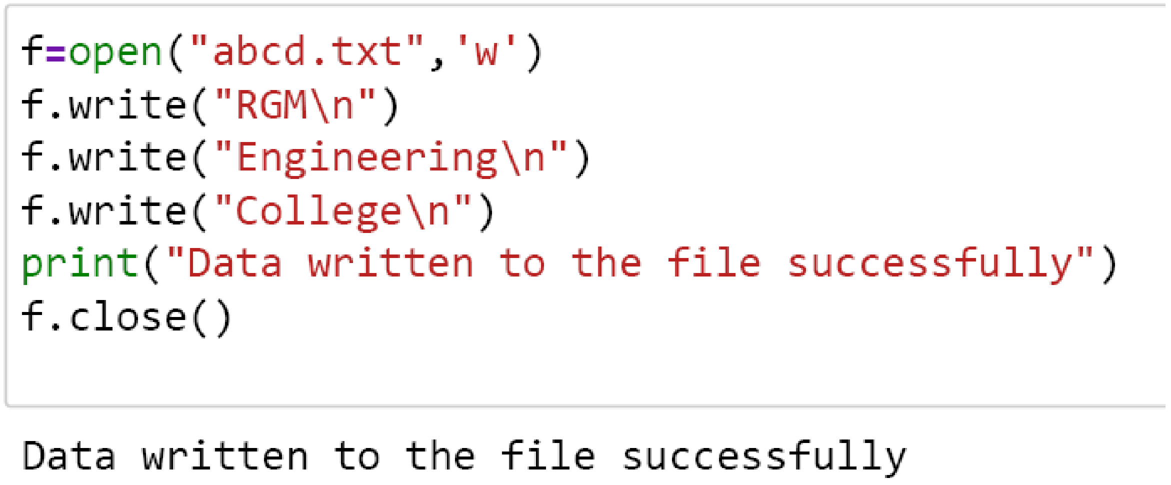
```
myobject=open("myfile.txt",'w')
marks=58
#number 58 is converted to a string using str()
myobject.write(str(marks))          ➔ 2
myobject.close()
```

Key Point:

- 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.

More Examples on write() method:

Eg 1:



```python
f=open("abcd.txt",'w')
f.write("RGM\n")
f.write("Engineering\n")
f.write("College\n")
print("Data written to the file successfully")
f.close()
```

```
Data written to the file successfully
```

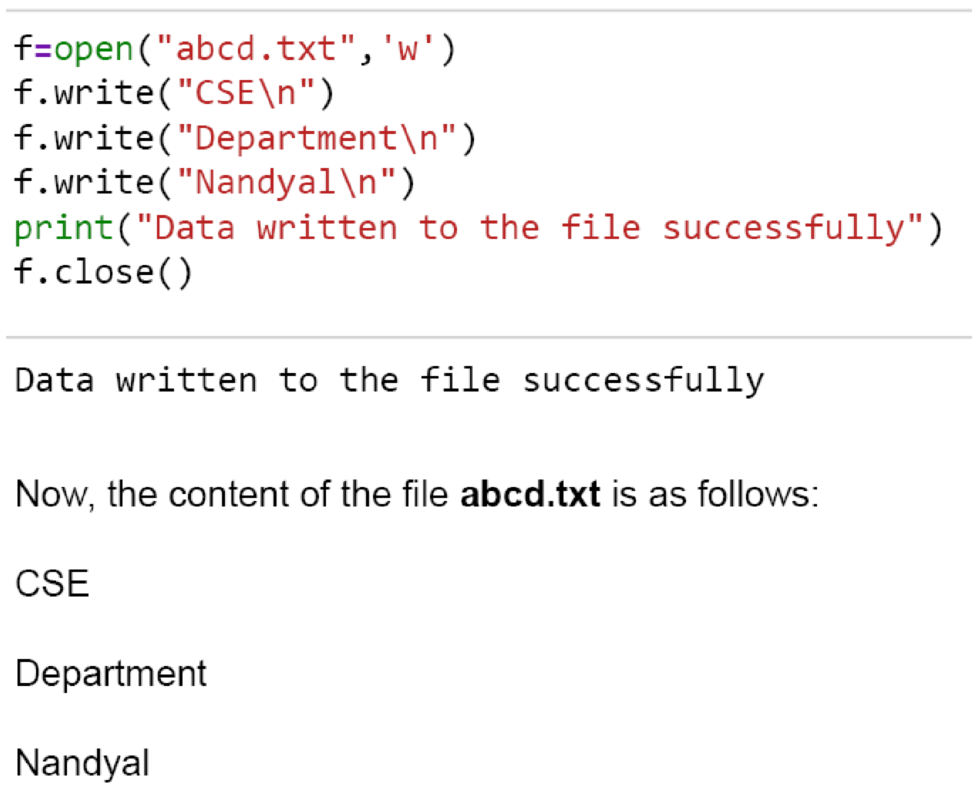
Now, the content of the file **abcd.txt** is as follows:

RGM

Engineering

College

Eg 2:



```
f=open("abcd.txt",'w')
f.write("CSE\n")
f.write("Department\n")
f.write("Nandyal\n")
print("Data written to the file successfully")
f.close()
```

```
Data written to the file successfully
```

Now, the content of the file **abcd.txt** is as follows:

CSE

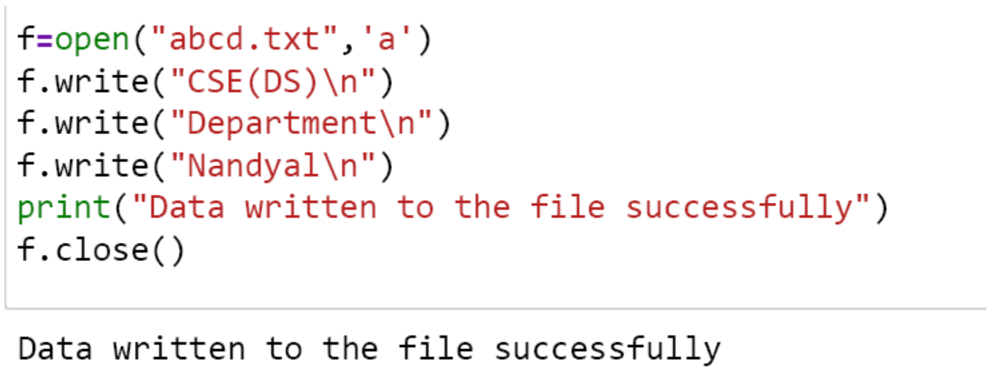
Department

Nandyal

Note:

In the above program, data present in the file will be overridden every time if we run the program. Instead of overriding if we want append operation then we should open the file as follows.

Eg 3:



```
f=open("abcd.txt",'a')
f.write("CSE(DS)\n")
f.write("Department\n")
f.write("Nandyal\n")
print("Data written to the file successfully")
f.close()
```

```
Data written to the file successfully
```

Now, the content of the file **abcd.txt** is as follows:

CSE

Department

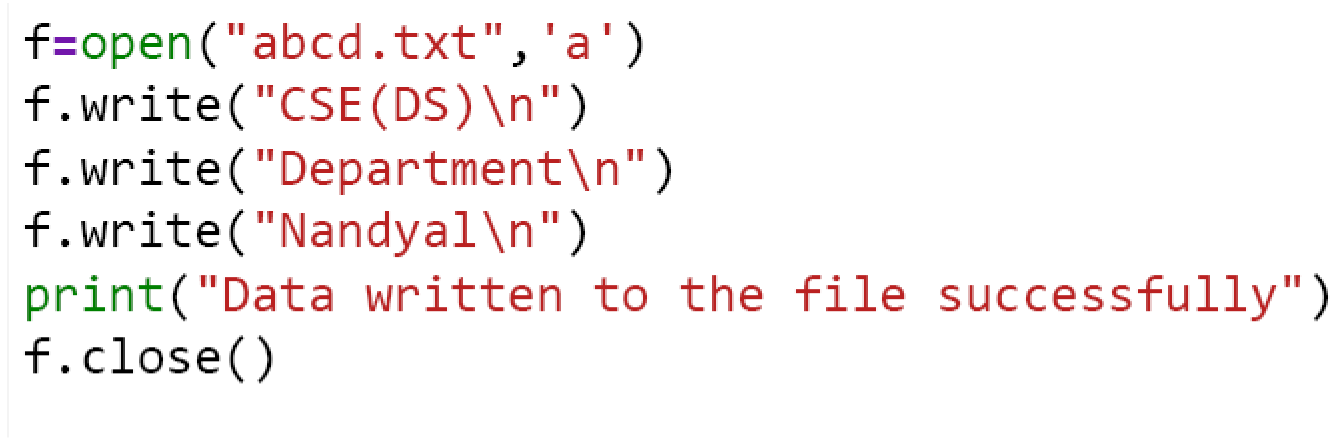
Nandyal

CSE(DS)

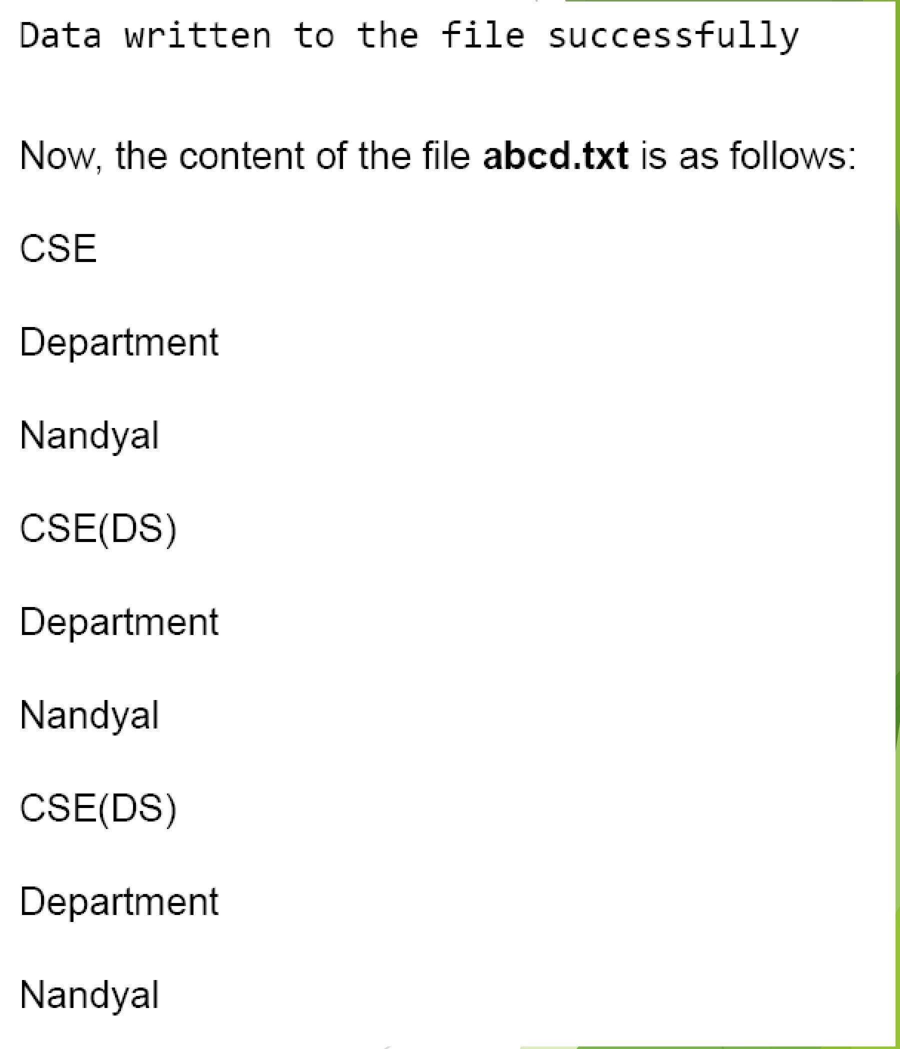
Department

Nandyal

Eg 4:



```python
f=open("abcd.txt",'a')
f.write("CSE(DS)\n")
f.write("Department\n")
f.write("Nandyal\n")
print("Data written to the file successfully")
f.close()
```



```
Data written to the file successfully
```

Now, the content of the file **abcd.txt** is as follows:

CSE

Department

Nandyal

CSE(DS)

Department

Nandyal

CSE(DS)

Department

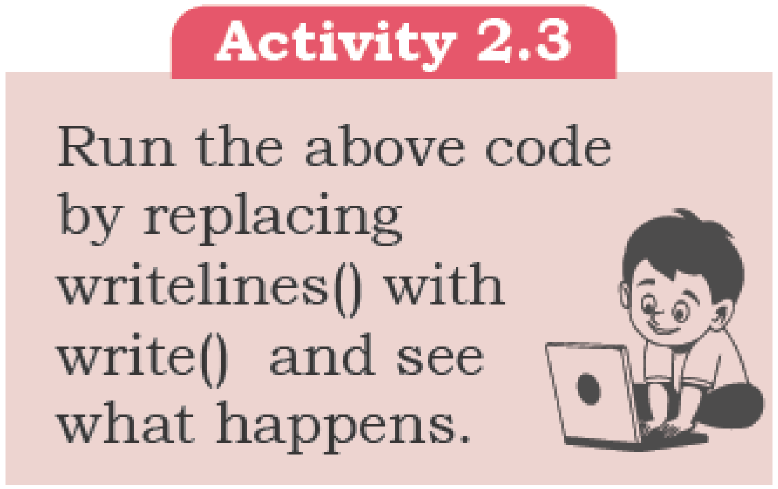
Nandyal

2.4.2 The 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.

The following code explains the use of writelines().

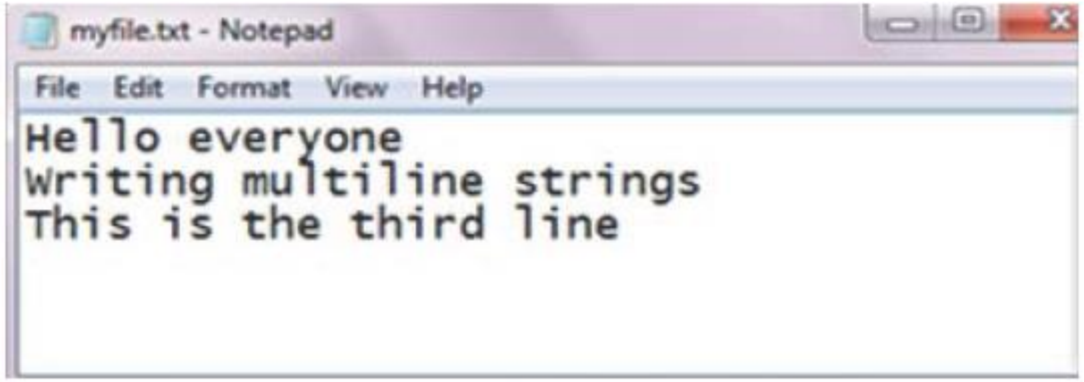
```
myobject=open("myfile.txt",'w')

lines = ["Hello everyone\n", "Writing

#multiline strings\n", "This is the

#third line"]

myobject.writelines(lines)

myobject.close()
```



Activity 2.3

Run the above code by replacing writelines() with write() and see what happens.

On opening myfile.txt, using notepad, its content will appear as shown in Figure 2.1.



```
Hello everyone
Writing multiline strings
This is the third line
```

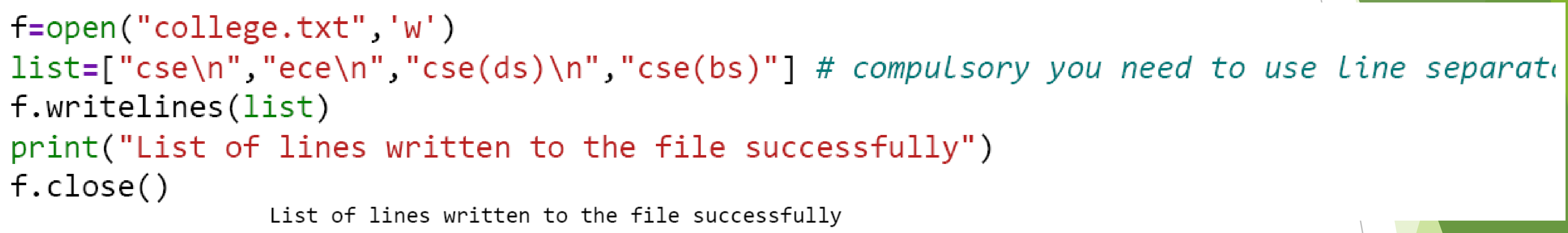
Figure 2.1: Contents of myfile.txt

Think and Reflect

Can we pass a tuple of numbers as an argument to writelines()? Will it be written to the file or an error will be generated?

More Examples on writelines() method:

Eg 1:



```
f=open("college.txt",'w')
list=["cse\n","ece\n","cse(ds)\n","cse(bs)"] # compulsory you need to use line separat
f.writelines(list)
print("List of lines written to the file successfully")
f.close()
```

```
List of lines written to the file successfully
```

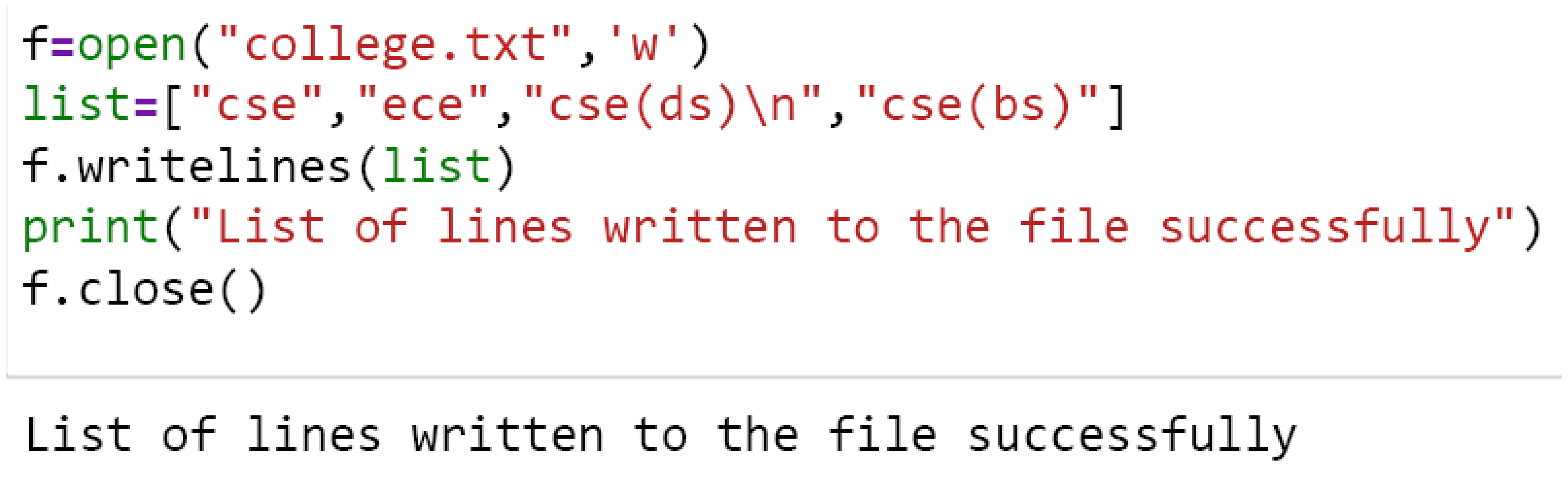
Now, the content of the file **college.txt** is as follows:

```
cse

ece

cse(ds)

cse(bs)
```

Note:

- while writing data by using write() methods, compulsory we have to provide line separator(\n),otherwise, total data should be written to a single line.

Eg 2:



```
f=open("college.txt",'w')
list=["cse","ece","cse(ds)\n","cse(bs)"]
f.writelines(list)
print("List of lines written to the file successfully")
f.close()
```

```
List of lines written to the file successfully
```

Now, the content of the file **college.txt** is as follows:

cseececse(ds)

cse(bs)

Eg 3:

```
f=open("college.txt",'w')
list=["cse\n","ece\n","cse(ds)\n","cse(bs)"]
f.writelines(list)
print("List of lines written to the file successfully")
f.close()
```

```
List of lines written to the file successfully
```

Now, the content of the file **college.txt** is as follows:

```
cse

ece
cse(ds)

cse(bs)
```

2.5 Reading Data from a Text File:

- We can write a program to read the contents of a file. Before reading a file, we must make sure that the file is opened in “r”, “r+”, “w+” or “a+” mode. There are different ways to read the contents of a file:

2.5.1 The read() method:

- This 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)

Consider the following set of statements to understand the usage of read() method:

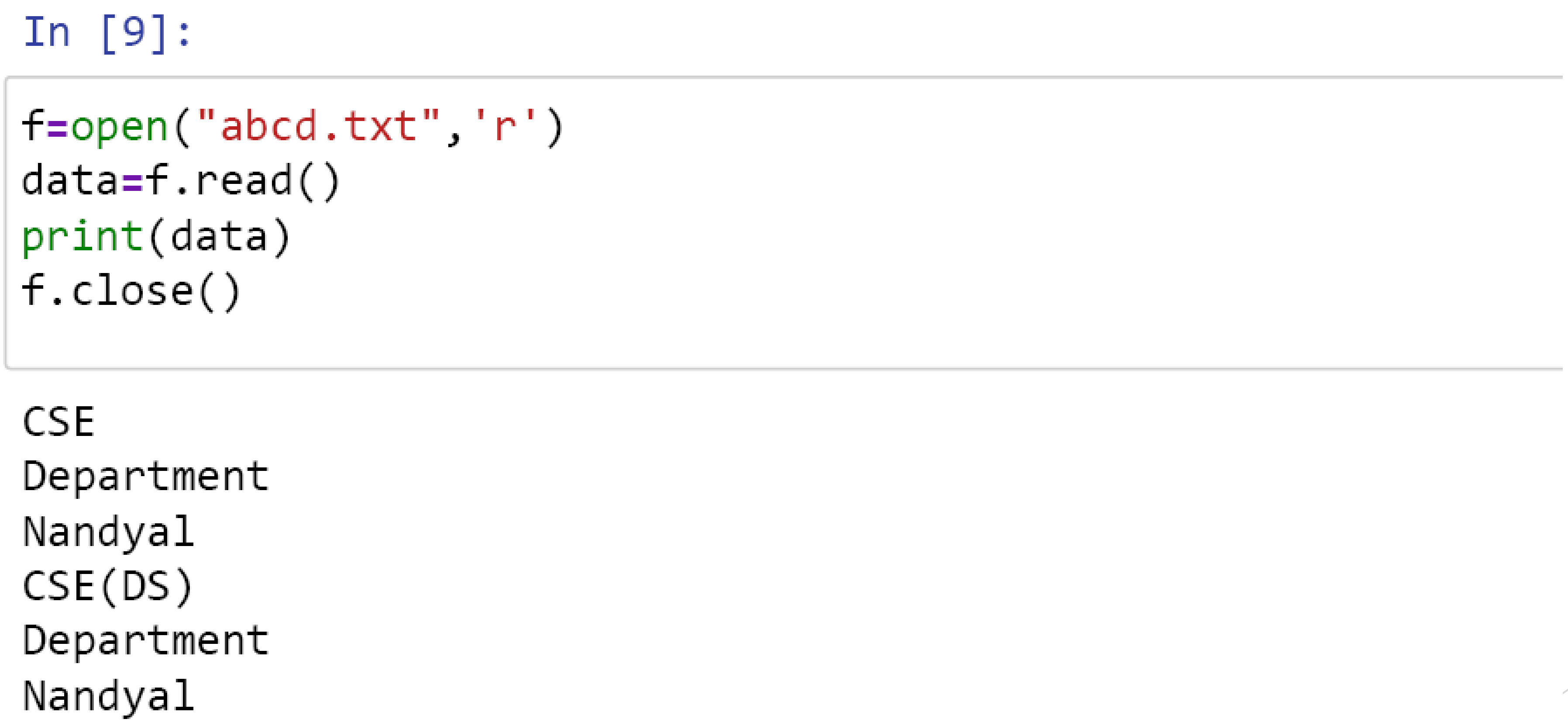
```
myobject=open("myfile.txt",'r')

data = myobject.read(10)

print(data)          ➔ 'Hello ever'

myobject.close()
```

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

```
myobject=open("myfile.txt",'r')
print(myobject.read())
➔ Hello everyone
➔ Writing multiline strings
➔ This is the third line
myobject.close()
```

More Examples on read() method:

Eg 1: Write a Python program to read total data from the specified file.

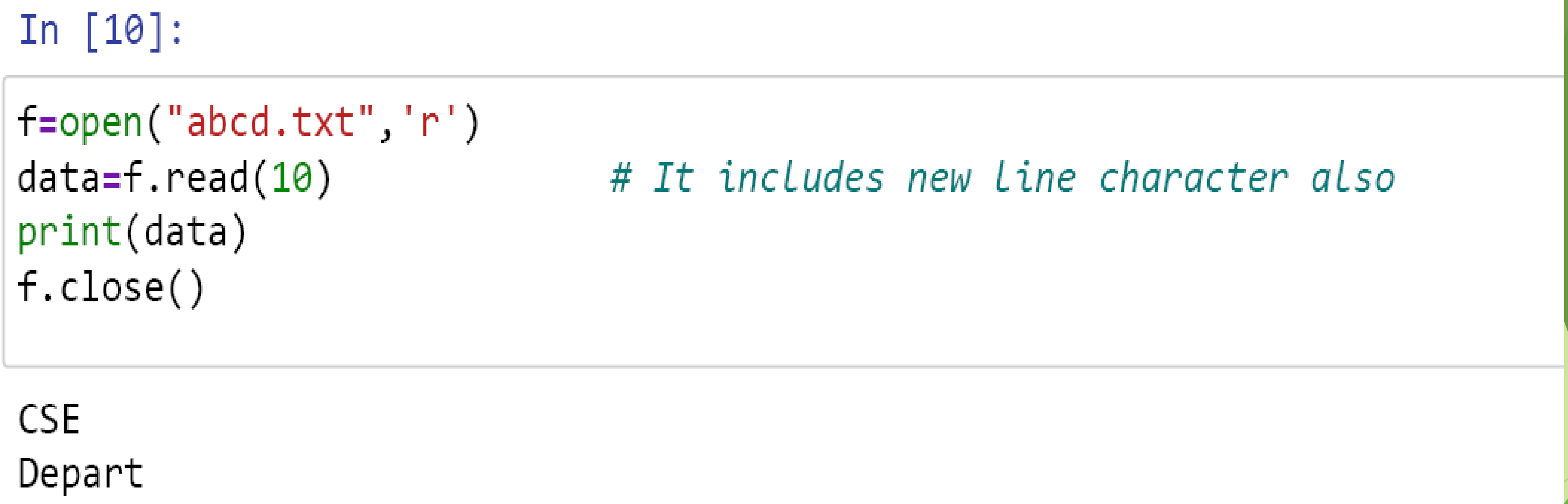


In [9]:

```python
f=open("abcd.txt",'r')
data=f.read()
print(data)
f.close()
```

```
CSE
Department
Nandyal
CSE(DS)
Department
Nandyal
```

Eg 2: Write a Python program to read only first 10 characters from the specified file.



In [10]:

```
f=open("abcd.txt",'r')
data=f.read(10)             # It includes new line character also
print(data)
f.close()
```

```
CSE
Depart
```

2.5.2 The readline([n]) method:

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

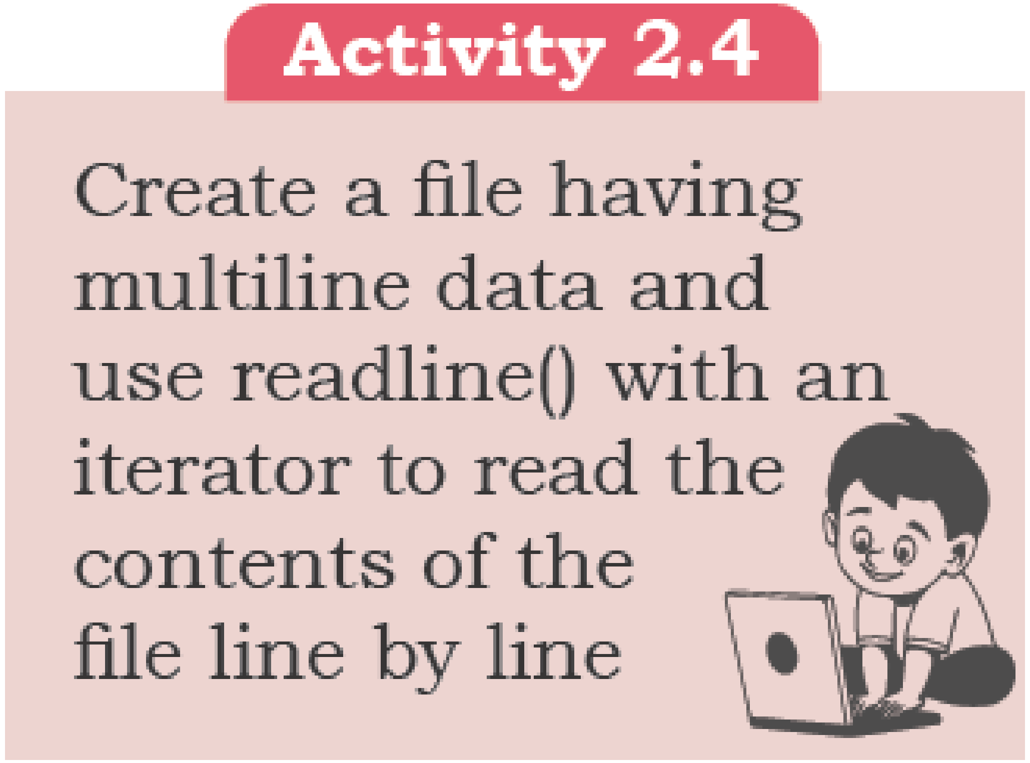
- In the following example, the second statement reads the first ten characters of the first line of the text file and displays them on the screen.

```
myobject=open("myfile.txt",'r')
myobject.readline(10)          ➔ 'Hello ever'
myobject.close()
```

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

```
myobject=open("myfile.txt",'r')
print (myobject.readline())          ➔ 'Hello everyone\n'
```

- To read the entire file line by line using the readline(), we can use a loop. This process is known as looping/iterating over a file object. It returns an empty string when EOF is reached.



Activity 2.4

Create a file having multiline data and use readline() with an iterator to read the contents of the file line by line

One Possible solution:

```
myfile = open("myfile.txt", "r")
while myfile:
    line  = myfile.readline()
    print(line)
    if line == "":
        break
myfile.close()
```

2.5.3 The readlines() method:

- The method reads all the lines and returns the lines along with newline as a list of strings. The following example uses readlines() to read data from the text file *myfile.txt.*

```
myobject=open("myfile.txt", 'r')
print(myobject.readlines())
➔ ['Hello everyone\n', 'Writing multiline strings\n', 'This is the third line']
myobject.close()
```

As shown in the above output, when we read a file using readlines() function, lines in the file become members of a list, where each list element ends with a newline character ('\n').

In case we want to display each word of a line separately as an element of a list, then we can use **split()** function. The following code demonstrates the use of split() function.

```
myobject=open("myfile.txt",'r')

d=myobject.readlines()

for line in d:

    words=line.split()

print(words)
```

Output:

```
['Hello', 'everyone']

['Writing', 'multiline', 'strings']

['This', 'is', 'the', 'third', 'line']
```

In the output, each string is returned as elements of a list. However, if splitlines() is used instead of split(), then each line is returned as element of a list, as shown in the output below:

```
for line in d:
    words=line.splitlines()
    print(words)
```

Output:

```
['Hello everyone']
['Writing multiline strings']
['This is the third line']
```

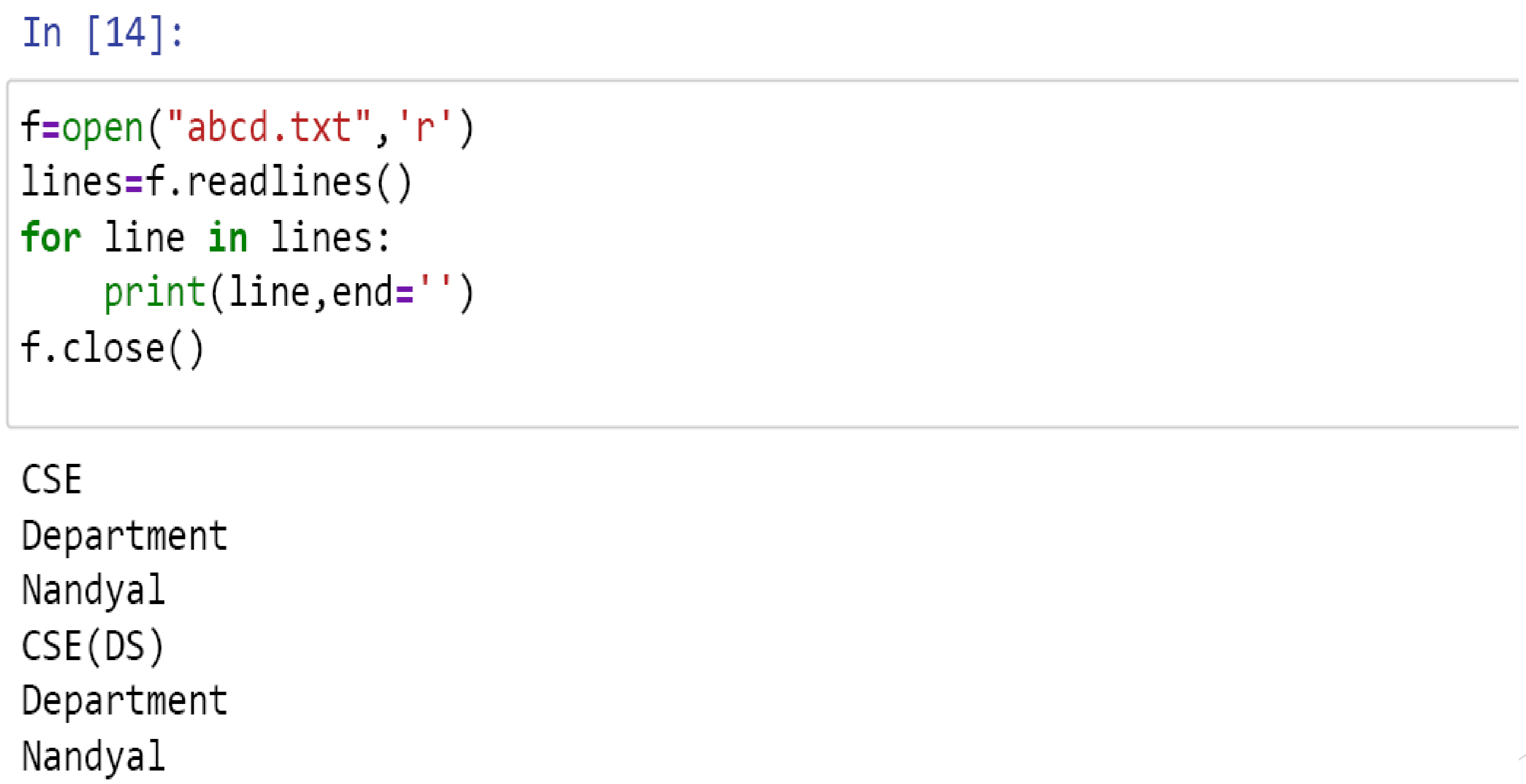
More Python programs on readlines() method:

1. Write a Python program to read data line by line from the specified file.

```
f=open("abcd.txt",'r')
line1=f.readline()
print(line1,end='') #
line2=f.readline()  #
print(line2,end='') #
line3=f.readline()
print(line3,end='')
line4=f.readline()
print(line4,end='')
line5=f.readline()
print(line5,end='')
line6=f.readline()
print(line6,end='')
f.close()
```

```
CSE
Department
Nandyal
CSE(DS)
Department
Nandyal
```

Eg 4: Write a Python program to read all lines from the specified file into a list.

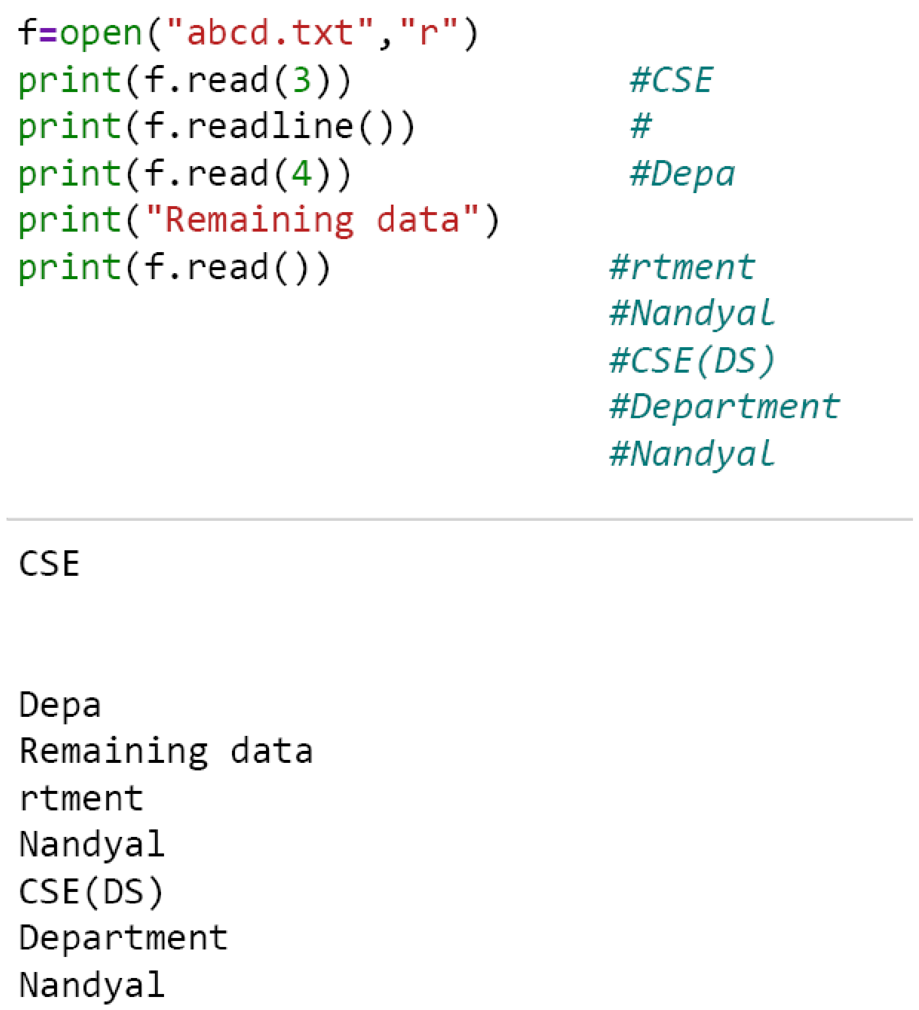


In [14]:

```
f=open("abcd.txt",'r')
lines=f.readlines()
for line in lines:
    print(line,end='')
f.close()
```

```
CSE
Department
Nandyal
CSE(DS)
Department
Nandyal
```

Eg 5:Write a Python program to demonstrate various functions to read the content from the specifie file.



```
f=open("abcd.txt","r")
print(f.read(3))             #CSE
print(f.readline())          #
print(f.read(4))             #Depa
print("Remaining data")
print(f.read())             #rtment
                            #Nandyal
                            #CSE(DS)
                            #Department
                            #Nandyal
```

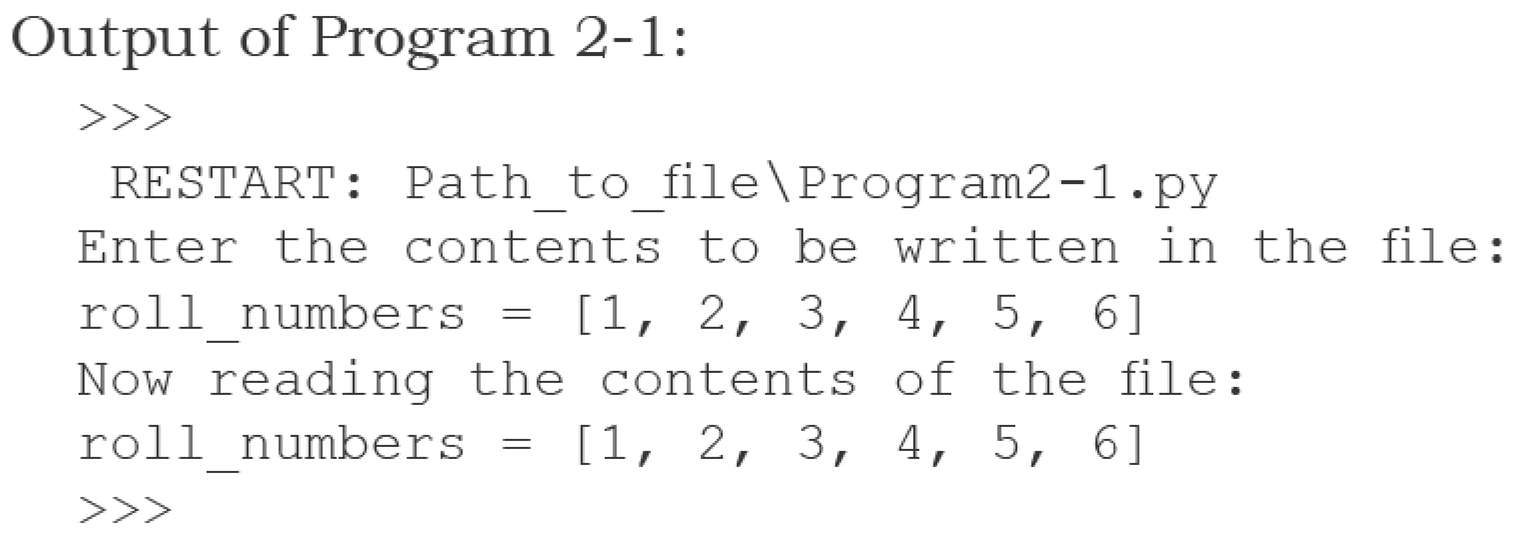
```
CSE


Depa
Remaining data
rtment
Nandyal
CSE(DS)
Department
Nandyal
```

Demo Program on writing and reading to a text file

```
fobject=open("testfile.txt","w")                    # creating a data file
sentence  =    input("Enter the contents to be written in the file: ")
fobject.write(sentence)                             # Writing data to the file
fobject.close()                                     # Closing a file
print("Now reading the contents of the file: ")
fobject=open("testfile.txt","r")                    #Looping over the file object to read the file
for str in fobject:
    print(str)
fobject.close()
```

```
Enter the contents to be written in the file: Honesty is the best policy
Now reading the contents of the file:
Honesty is the best policy
```

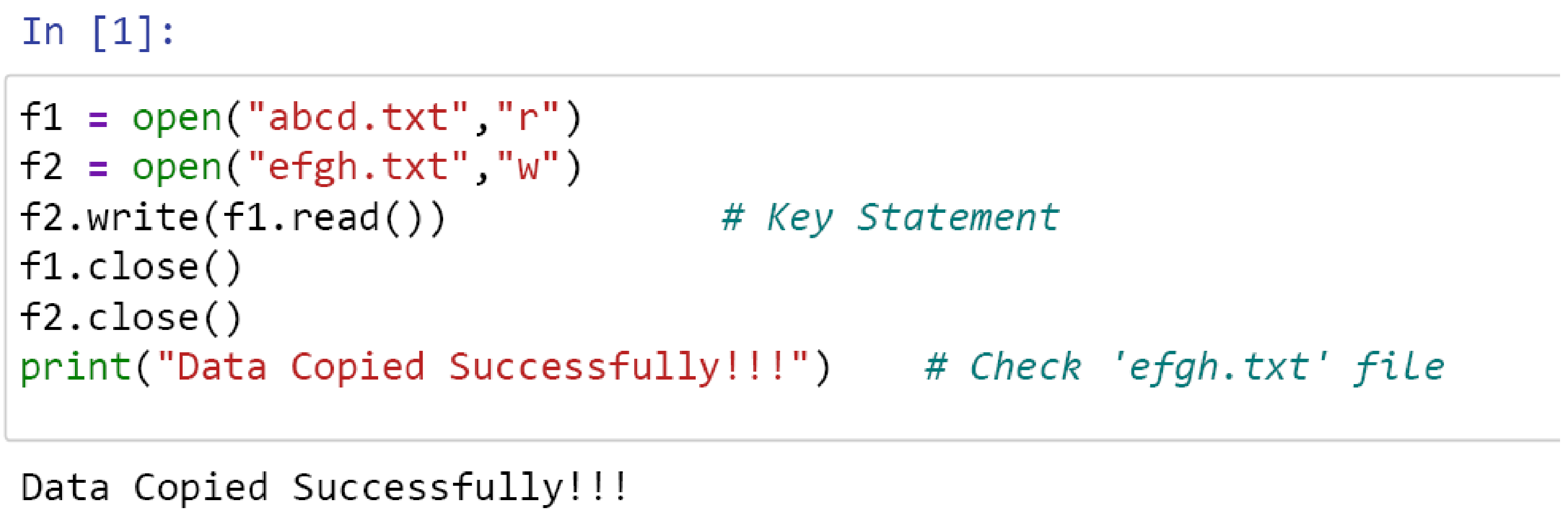
In the previous Program , the file named *testfile.txt* is opened in write mode and the file handle named *fobject* is returned. The string is accepted from the user and written in the file using write(). Then the file is closed and again opened in read mode. Data is read from the file and displayed till the end of file is reached.


Output of Program 2-1:

```
>>>
 RESTART: Path_to_file\Program2-1.py
Enter the contents to be written in the file:
roll_numbers = [1, 2, 3, 4, 5, 6]
Now reading the contents of the file:
roll_numbers = [1, 2, 3, 4, 5, 6]
>>>
```

Eg 6: Write a Python program to copy the contents of one file to another file.

Way 1:

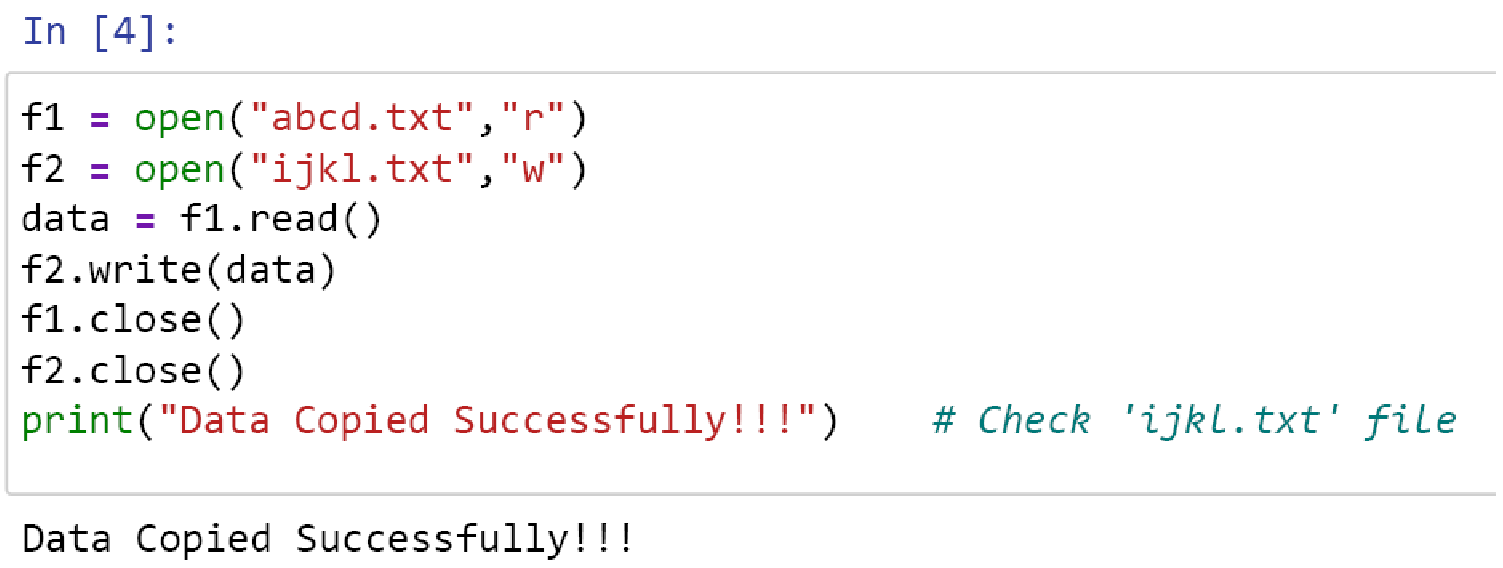


In [1]:

```
f1 = open("abcd.txt","r")
f2 = open("efgh.txt","w")
f2.write(f1.read())            # Key Statement
f1.close()
f2.close()
print("Data Copied Successfully!!!")    # Check 'efgh.txt' file
```

```
Data Copied Successfully!!!
```

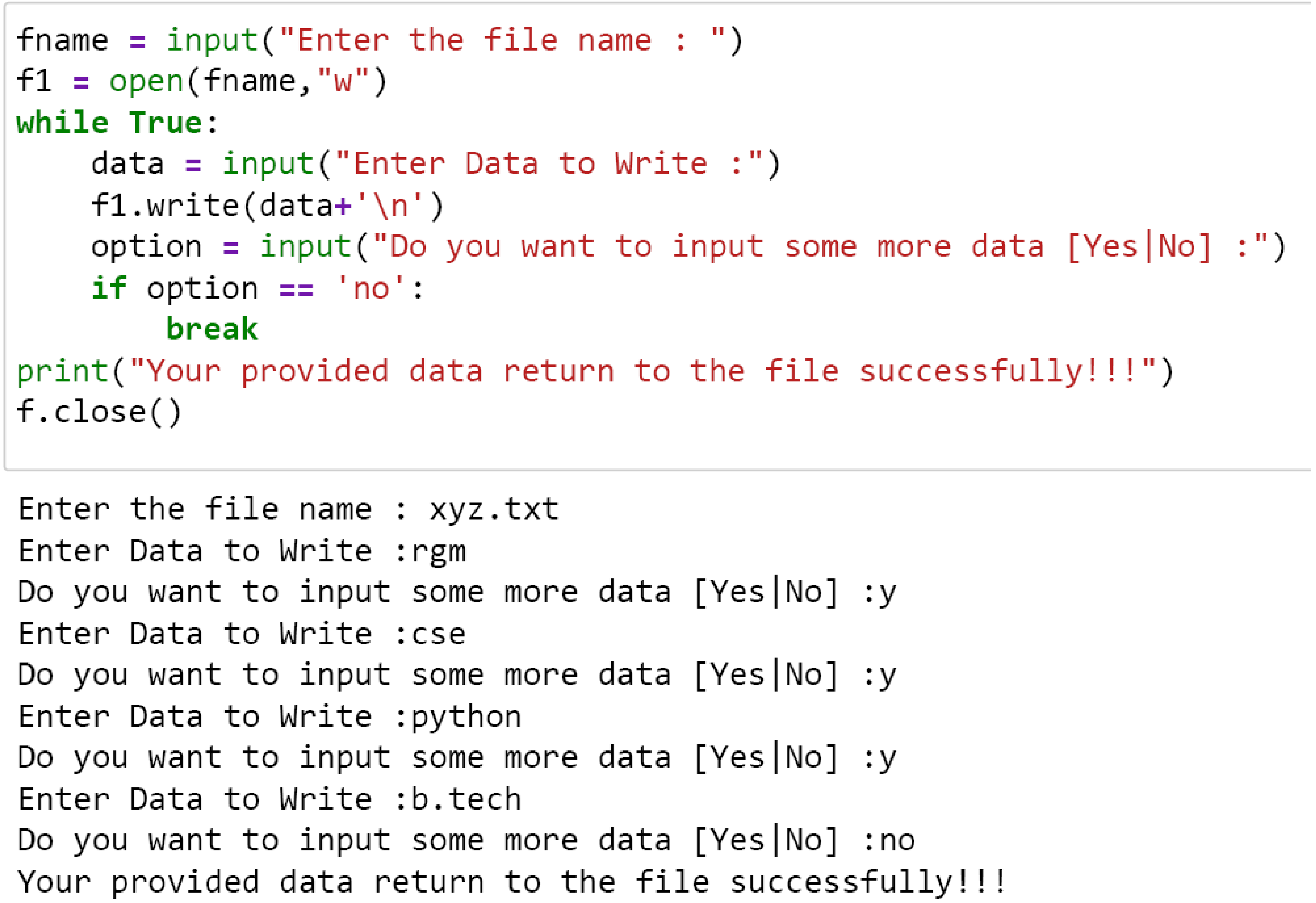
Way 2:



```
In [4]:

f1 = open("abcd.txt","r")
f2 = open("ijkl.txt","w")
data = f1.read()
f2.write(data)
f1.close()
f2.close()
print("Data Copied Successfully!!!")    # Check 'ijkl.txt' file
```

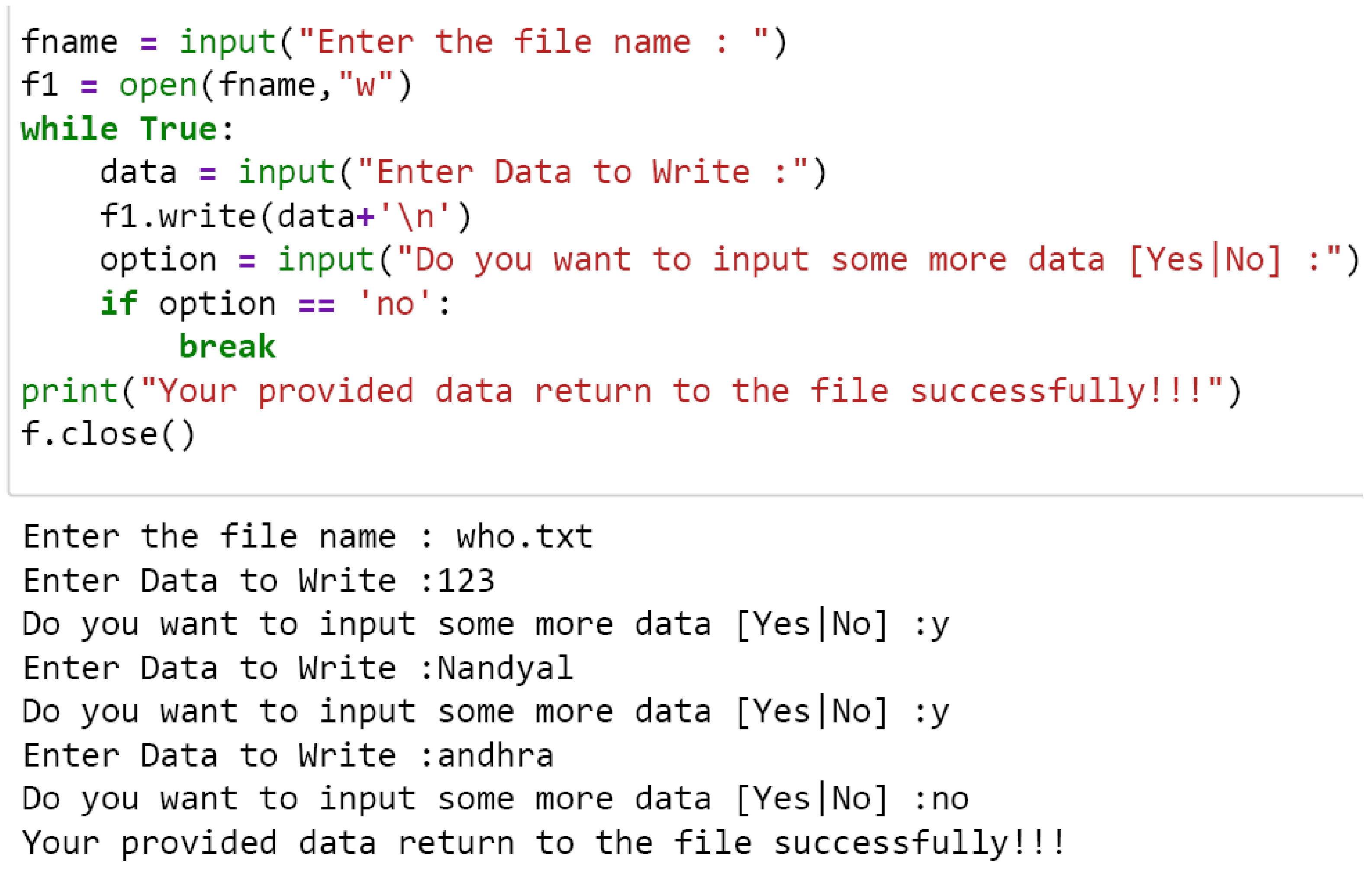
```
Data Copied Successfully!!!
```

Eg 7:Write a Python program to write the content into a file from the keyboard.



```python
fname = input("Enter the file name : ")
f1 = open(fname,"w")
while True:
    data = input("Enter Data to Write :")
    f1.write(data+'\n')
    option = input("Do you want to input some more data [Yes|No] :")
    if option == 'no':
        break
print("Your provided data return to the file successfully!!!")
f.close()
```

```
Enter the file name : xyz.txt
Enter Data to Write :rgm
Do you want to input some more data [Yes|No] :y
Enter Data to Write :cse
Do you want to input some more data [Yes|No] :y
Enter Data to Write :python
Do you want to input some more data [Yes|No] :y
Enter Data to Write :b.tech
Do you want to input some more data [Yes|No] :no
Your provided data return to the file successfully!!!
```



```python
fname = input("Enter the file name : ")
f1 = open(fname,"w")
while True:
    data = input("Enter Data to Write :")
    f1.write(data+'\n')
    option = input("Do you want to input some more data [Yes|No] :")
    if option == 'no':
        break
print("Your provided data return to the file successfully!!!")
f.close()
```

```
Enter the file name : who.txt
Enter Data to Write :123
Do you want to input some more data [Yes|No] :y
Enter Data to Write :Nandyal
Do you want to input some more data [Yes|No] :y
Enter Data to Write :andhra
Do you want to input some more data [Yes|No] :no
Your provided data return to the file successfully!!!
```

***** The '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:

- We can use this to group file operation statements within a block.
- 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. In case the user forgets to close the file explicitly or if an exception occurs, the file is closed automatically.

Eg:

```
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.

Demo Program:

```python
with open("abc.txt","w") as f:
    f.write("RGM\n")
    f.write("Engineering\n")
    f.write("College\n")
    print("Is File Closed: ",f.closed)
print("Is File Closed: ",f.closed)
```

```
Is File Closed:  False
Is File Closed:  True
```

Key Questions:

Q 1. What is the advantage of using with block while opening a file?

- The advantage of with statement is it will take care closing of file, after completing all operations automatically even in the case of exceptions also, and we are not required to close explicitly.

Q 2.What is the difference between f = open("abc.txt","w") and with open("abc.txt","w") as f: ?

In the first case:

- We have to close file explicitly

In the second case:

- We need not to close file explicitly and it will be closed automatically.

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You