Q

Files

File: is collection of data. (General definition)

Advantages of Files regarding programing language to store the output in the file.

* Files are TWO Types
  1) Text files 2) Binary files

Basic operations performed on a data file are.

→ Naming a file
→ opening a file
→ Reading data from the file
→ writing data in to file
→ closing a file.

opening a file

You need to first create a file object that is associated with a physical file (Text file). This is called openig a file.

Syntax:-

fileobjectvariable = open("fileName", "accessmode")

the open function returns a file object for filename

The mode parameter is a string that specifies how the file will be used (for reading or writing)

Ex:- `file_obj = open("scores.txt", "r")`

"scores.txt" → this is filename; "r" → access mode

Accessmodes

Text files	Binary files
r — read	rb — read
w — write.	wb — write
a — append	ab — append
r+ — Reading & writing	rb+ — Read & write
w+ — Reading & writing	wb+ — Read & write
a+ — Reading & writing	ab+ — Read & write.

File object Attributes

Attribute	Information obtained.
→ Fileobj.closed	Returns true if the file is closed and False otherwise
→ Fileobj.mode	Returns access mode with which file has been opened.
→ Fileobj.name	Returns name of the file.

Example: program to open a file and print its attributes

program(6)

```
infile = open("student.txt", "r")
print("Name of the file :", infile.name)
print("file is closed:", infile.closed)
print("file has been opened in:", infile.mode, "mode")
```

output

Name of the file : student.txt

file is closed : False.

file has been opened in : r mode.

Reading data

→ Reading data means Reading text from the text file.

→ To read the data we have to use read() method(or) function.

program:-

```
infile = open("student.txt", "r")
read_data = infile.read()
print(read_data)
infile.close
```

student.txt

```
This is student data
Suresh
Mahesh
Mallesh
```

read() function reads all characters from the file and returns them as a string.

* If you want to read number of character from a file then

Syntax:- read(number)
→ No. of characters

program.

```
infile = open("student.txt", "r")
data = infile.read(5)
print(data)
infile.close()
```

student.txt

This is student data.

output

This !

readline()

readline() method to read the next line of line

program:

```
infile = open("class.txt", "r")
line1 = infile.readline()
line2 = infile.readline()
print(line1)
print(line2)
infile.close()
```

class.txt

B_1 class
B_2 class
B_3 class
B_4 class

output: B_1 class
B_2 class

readlines()

readlines function to read all the lines into a list of strings

program:-

```
infile = open("class.txt","r")
print(infile.readlines())
infile.close()
```

output

["B1 class\n", "B2 class\n", "B3 class\n", "B4 class"]

Writing data in the file

~~b~~ - ~~access mode to open a file~~

w - access mode only for writing to the content.

~~If the File exits, the file is opened~~

→ If the File exits, the data will be overwrite.

→ If the File not exits, then new File will be created with given File Name.

write(str)

write() method to write a string to the file

program:-

```
def main()
    outfile = open("student.txt","w") # open file for output
    outfile.write("Somewhere Road\n") # write a string to the file
    outfile.write("Sharvin\n")
    outfile.close()
main() # call the main function
```

output:-

student.txt

Somewhere
Sharvin

writelines(sequence):

writelines ( ) function to write a sequence of strings to the file. The sequence can be any iterable object producing strings, typically a list of strings.

program:-

```
outfile = open ( "write.txt";w')
lines = [" Hello world ,", "online classes", "Enjoy the python class"]
outfile . write ( lines)
outfile . close ()
```

output

write . txt

Hello world
online classes
Enjoy the Python class.

Appending data

a - access mode, file for appending data to the end of an existing file.

program

```
outfile = open("info.txt", "a") # open file for appending data
outfile.write("In python is interpreted")
outfile.close()
```

output

info.txt