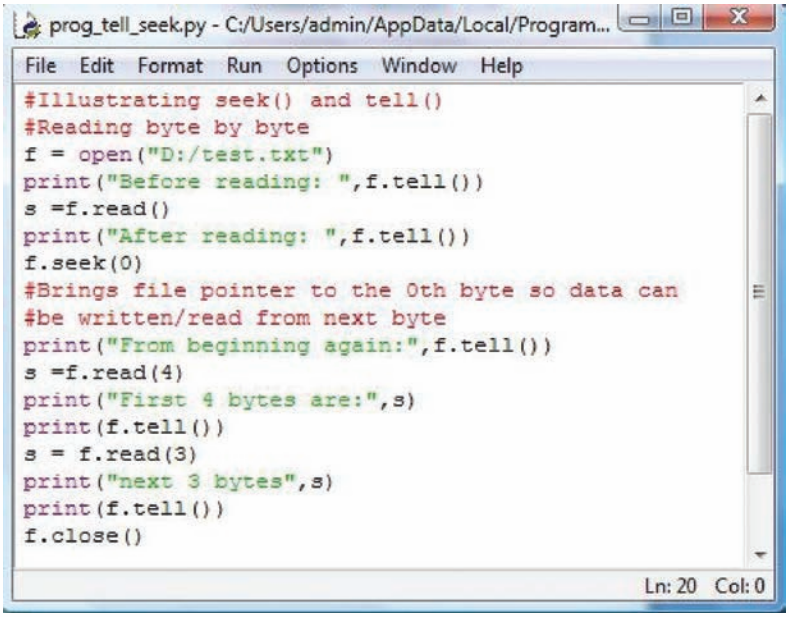
- When you open a file in reading/writing mode, the file pointer rests at 0th byte.
- When you open a file in append mode, the file pointer rests at last byte.

This is illustrated in the practical implementation that follows:



Contents of "test.txt"

```
>>>
 RESTART: C:/Users/admin/AppData/Local/Programs
k.py
Before reading:  0
After reading:  48
From beginning again: 0
First 4 bytes are: This
4
next 3 bytes  is
7
```

4.13 INTRODUCTION TO CSV

In every span of today's organizational working environment, data sharing is one of the major tasks to be carried out, largely through spreadsheets or databases.

A basic approach to share data is through the comma separated values (CSV) file.

CSV is a simple flat file in a human readable format which is extensively used to store tabular data, in a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text.

Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOffice Calc.

Already defined, CSV stands for "**comma separated values**". Thus, we can say that a comma-separated file is a delimited text file that uses a comma to separate values.

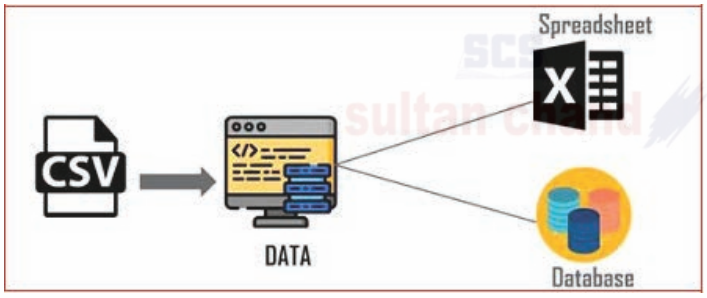


Fig. 4.1: CSV Storage Format

Fig. 4.2: Features of a CSV file

Each line in a file is known as **data/record**. Each record consists of one or more fields, separated by commas (also known as delimiters), *i.e.*, each of the records is also a part of this file. Tabular data is stored as **text** in a CSV file. The use of comma as a field separator is the source of the name for this file format. It stores our data into a spreadsheet or a database.

CTM: The most commonly used delimiter in a CSV file is usually a **comma**.

4.14 WHY USE CSV

With the use of social networking sites and its various associated applications being extensively used requires the handling of huge data. The problem arises as to how to handle and organize this large unstructured data as shown in the Fig. 4.3(a) given below.

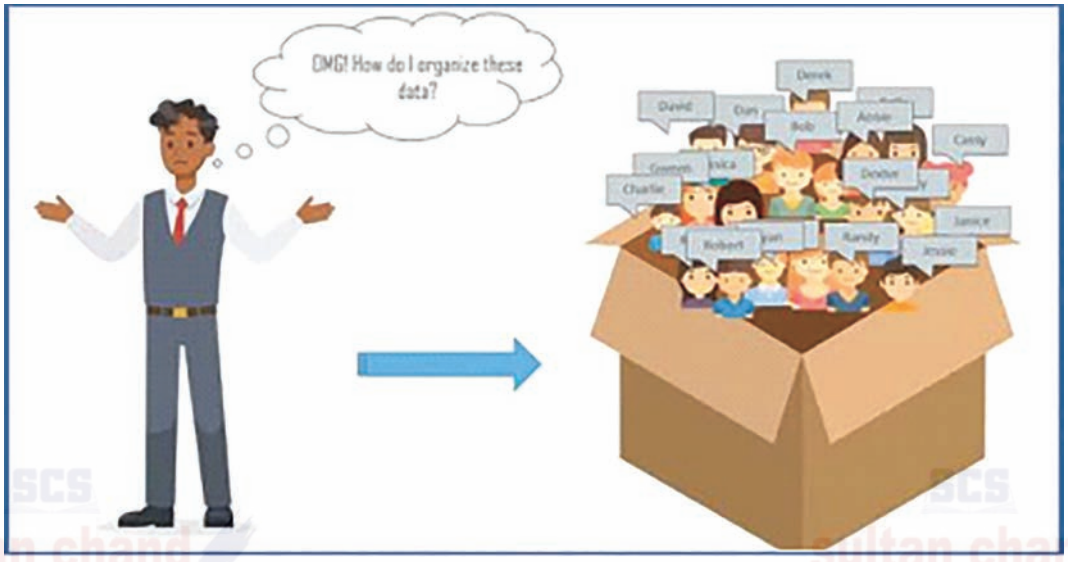


Fig. 4.3(a): Problem of Huge Data

The solution to the above problem is CSV (Fig. 4.3(b)). Thus, the CSV organizes data into a structured form and, hence, the proper and systematic organization of this large amount of data is done by CSV. Since CSV files formats are of plain text format, it makes it very easy for website developers to create applications that implement CSV.

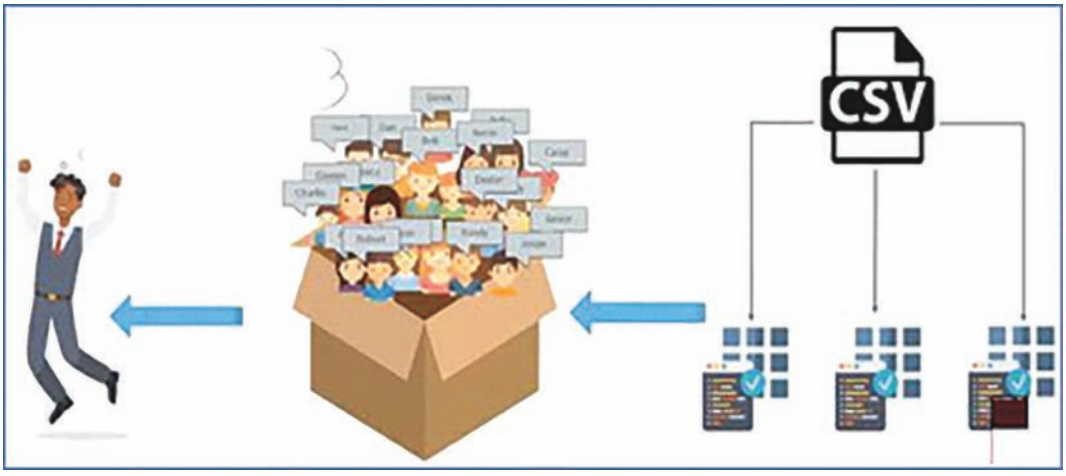


Fig. 4.3(b): CSV as a Solution

CSV files are commonly used because they are easy to read and manage, small in size, and fast to process/transfer. Because of these salient features, they are frequently used in software applications, ranging anywhere from online e-commerce stores to mobile apps to desktop tools. *For example*, Magento, an e-commerce platform, is known for its support of CSV.

Thus, in a nutshell, the several advantages that are offered by CSV files are as follows:

- CSV is faster to handle.
- CSV is smaller in size.
- CSV is easy to generate and import onto a spreadsheet or database.
- CSV is human readable and easy to edit manually.
- CSV is simple to implement and parse.
- CSV is processed by almost all existing applications.

After understanding the concept and importance of using CSV files, we will now discuss the read and write operations on CSV files.

4.15 CSV FILE HANDLING IN PYTHON

For working with CSV files in Python, there is an inbuilt module called **CSV**. It is used to read and write tabular data in CSV format.

Therefore, to perform read and write operation with CSV file, we must import **CSV module**. CSV module can handle CSV files correctly regardless of the operating system on which the files were created.

Along with this module, open() function is used to open a CSV file, and return file object. We load the module in the usual way using import:

```
>>> import csv
```

Like other files (text and binary) in Python, there are two basic operations that can be carried out on a CSV file.

1. Reading a CSV
2. Writing to a CSV.

Let us discuss these CSV operations.

4.15.1 Reading from CSV File

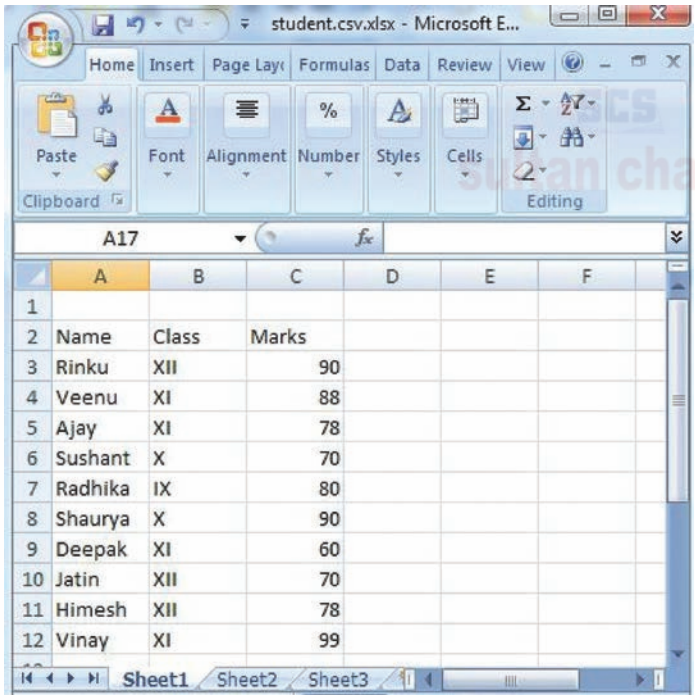
Reading from a CSV file is done using the reader object. The CSV file is opened as a text file with Python's built-in open() function, which returns a file object. This creates a special type of object to access the CSV file (reader object), using the reader() function.

The reader object is an iterable that gives us access to each line of the CSV file as a list of fields. You can also use next() directly on it to read the next line of the CSV file, or you can treat it like a list in a for loop to read all the lines of the file (as lists of the file's fields).

This is shown in the practical implementation given below.

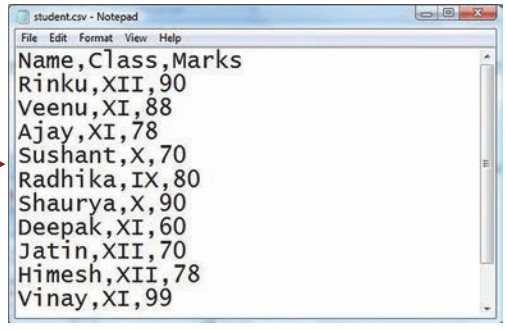
Before this, enter the student details in spreadsheet and save this file as shown:

Next step is to open the Notepad and enter the data for student.csv, which will be the equivalent for student.xls.



	A	B	C	D	E	F
1						
2	Name	Class	Marks			
3	Rinku	XII	90			
4	Veenu	XI	88			
5	Ajay	XI	78			
6	Sushant	X	70			
7	Radhika	IX	80			
8	Shaurya	X	90			
9	Deepak	XI	60			
10	Jatin	XII	70			
11	Himesh	XII	78			
12	Vinay	XI	99			

student.xls

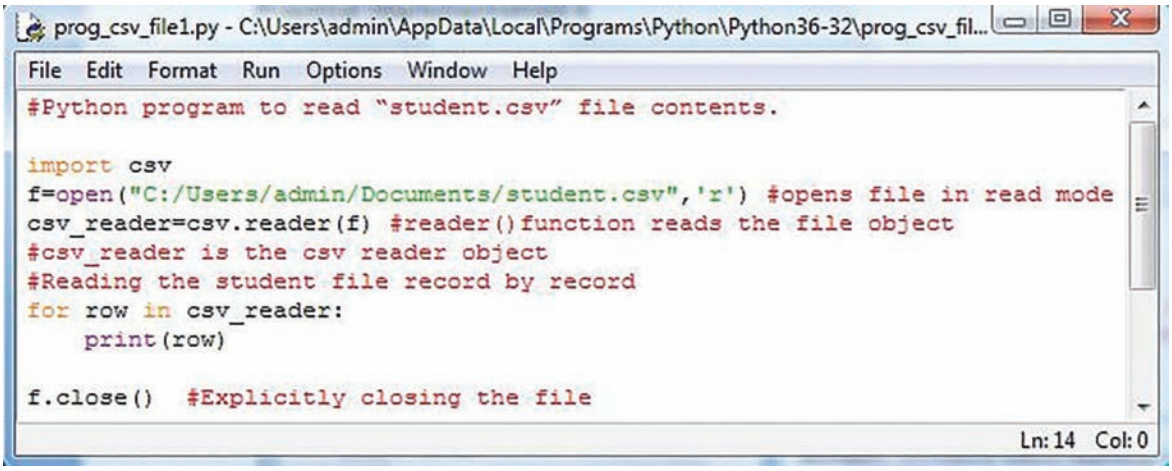


student.csv

In student.csv (notepad) file, first line is the header and remaining lines are the data/records. The fields are separated by comma, or we may say the separator character. In general, the separator character is called a delimiter, and the comma is not the only one used. Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters.

Practical Implementation–5

Write a program to read the contents of "student.csv" file.




```
>>>
 RESTART: C:/Users/admin/AppData/Local/Programs/
1.py
['Name', 'Class', 'Marks']
['Rinku', 'XII', '90']
['Veenu', 'XI', '88']
['Ajay', 'XI', '78']
['Sushant', 'X', '70']
['Radhika', 'IX', '80']
['Shaurya', 'X', '90']
['Deepak', 'XI', '60']
['Jatin', 'XII', '70']
['Himesh', 'XII', '78']
['Vinay', 'XI', '99']
[]
>>>
```

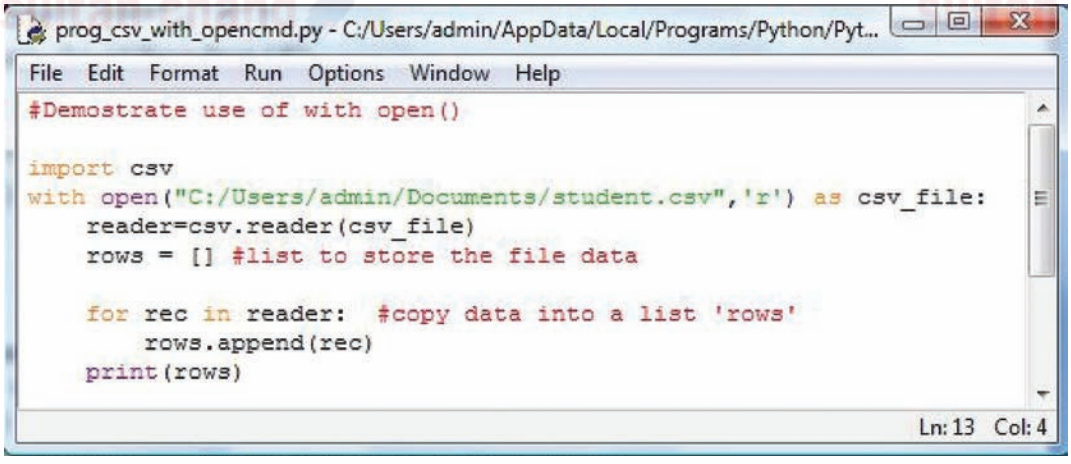
Explanation:

As seen from the above output, every record is stored in reader object in the form of a List. In the above code, we first open the CSV file in READ mode. The file object is named as **f**. The file object is converted to csv.reader object. We save the csv.reader object as **csv_reader**. The reader object is used to read records as lists from a csv file. Now, we iterate through all the rows using a for loop. When we try to print each row, one can find that row is nothing but a list containing all the field values. Thus, all the records are displayed as lists separated by comma.

In the next implementation, we will count the number of records present inside the student.csv file.

Practical Implementation–6

Write a program to read the contents of "student.csv" file using with open().

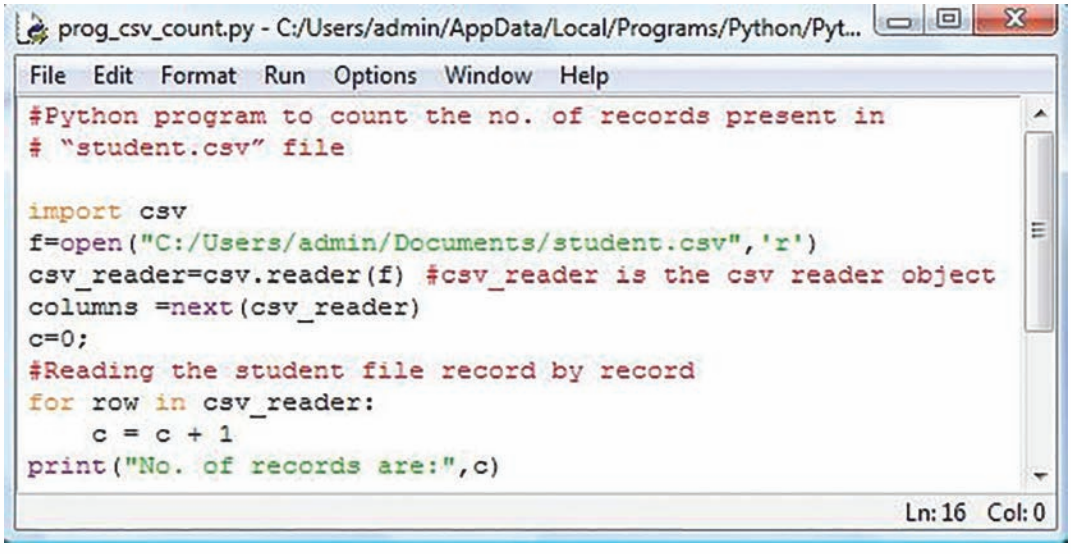



```
>>>
 RESTART: C:/Users/admin/AppData/Local/Programs/Python/Python36-32/prog_csv_
with_opencmd.py
[['Name', 'Class', 'Marks'], ['Rinku', 'XII', '90'], ['Veenu', 'XI', '88'],
['Ajay', 'XI', '78'], ['Sushant', 'X', '70'], ['Radhika', 'IX', '80'], ['Sha
urya', 'X', '90'], ['Deepak', 'XI', '60'], ['Jatin', 'XII', '70'], ['Himesh'
, 'XII', '78'], ['Vinay', 'XI', '99'], []]
```

The above modified code uses "with open()" function, the only difference being that the file being opened using with open() gets automatically closed after the program execution gets over, unlike open() where we need to give close() statement explicitly.

Practical Implementation–7

Write a program to count the number of records present in "student.csv" file.




```
>>>
 RESTART: C:/Users/admin/AppData/Local
t.py
No. of records are: 11
```

Explanation:

In the above program, a special type of object is created to access the CSV file (reader object), which is csv_reader using the reader() function. The reader object is an iterable that gives us access to each line of the CSV file as a list of fields. The function next() is used to directly point to this list of fields to read the next line in the CSV file. .next() method returns the current row and advances the iterator to the next row.

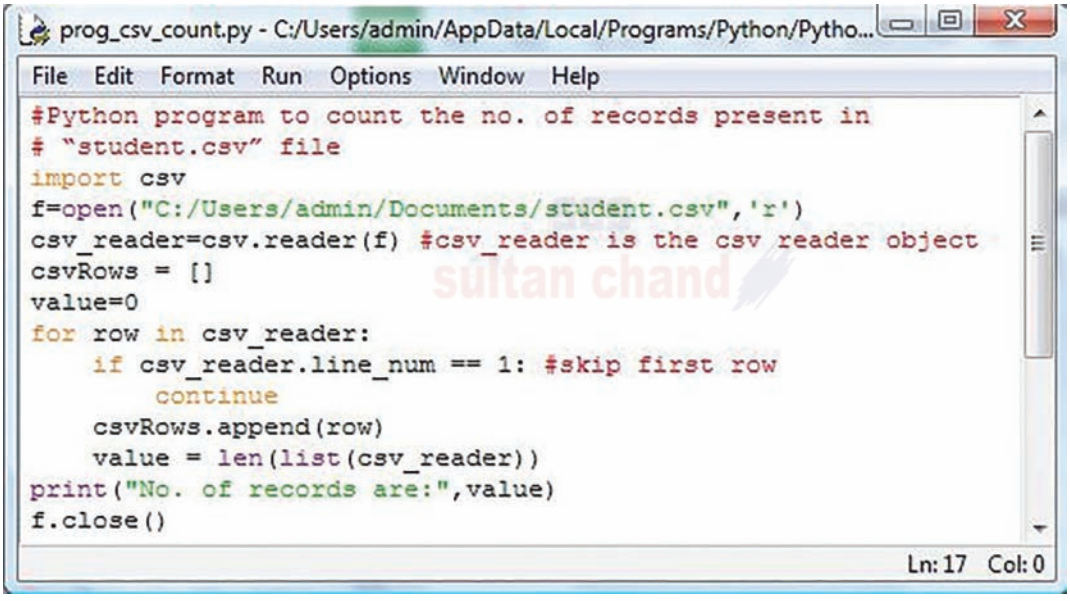
The variable 'c' is used as a counter variable to count the number of rows/records present in this file, which is finally printed and thus the output is so obtained.

One of the important observations from the output is the number of records which are being displayed as 11 instead of 10. This is so because the header (first line) in the student csv file is also treated as a record only. This limitation is overcome in the next implementation.

CTM: .next() method returns the current row and advances the iterator to the next row.

Practical Implementation–8

Program to count the exact number of records present in the csv file excluding the header.




```
>>>
 RESTART: C:/Users/admin/AppData/Local/
t.py
No. of records are: 10
```

In the above program we have used line_num object of CSV file. Our csv_reader_object has a method called line_num that returns the number of lines in our CSV.

Then, if statement checks if the line is first line or not. If the condition is true, *i.e.*, if it is the header line, then it is ignored using continue statement and the counting of records is resumed from second line onwards. Also, line_num object always stores the current line in consideration and, hence, the correct output for 10 records is so obtained.

CTM: line_num is nothing but a counter which returns the number of rows which have been iterated.

Till now we have covered the basics of how to use the CSV module to read the contents of a CSV file. Now, we will discuss about how to write to a CSV file in python.

4.15.2 Writing to CSV File

To write to a CSV file in Python, we can use the **csv.writer()** function. The csv.writer() function returns a writer object that converts the user's data into a delimited string. This string can later be used to write into CSV files using the writerow() function.

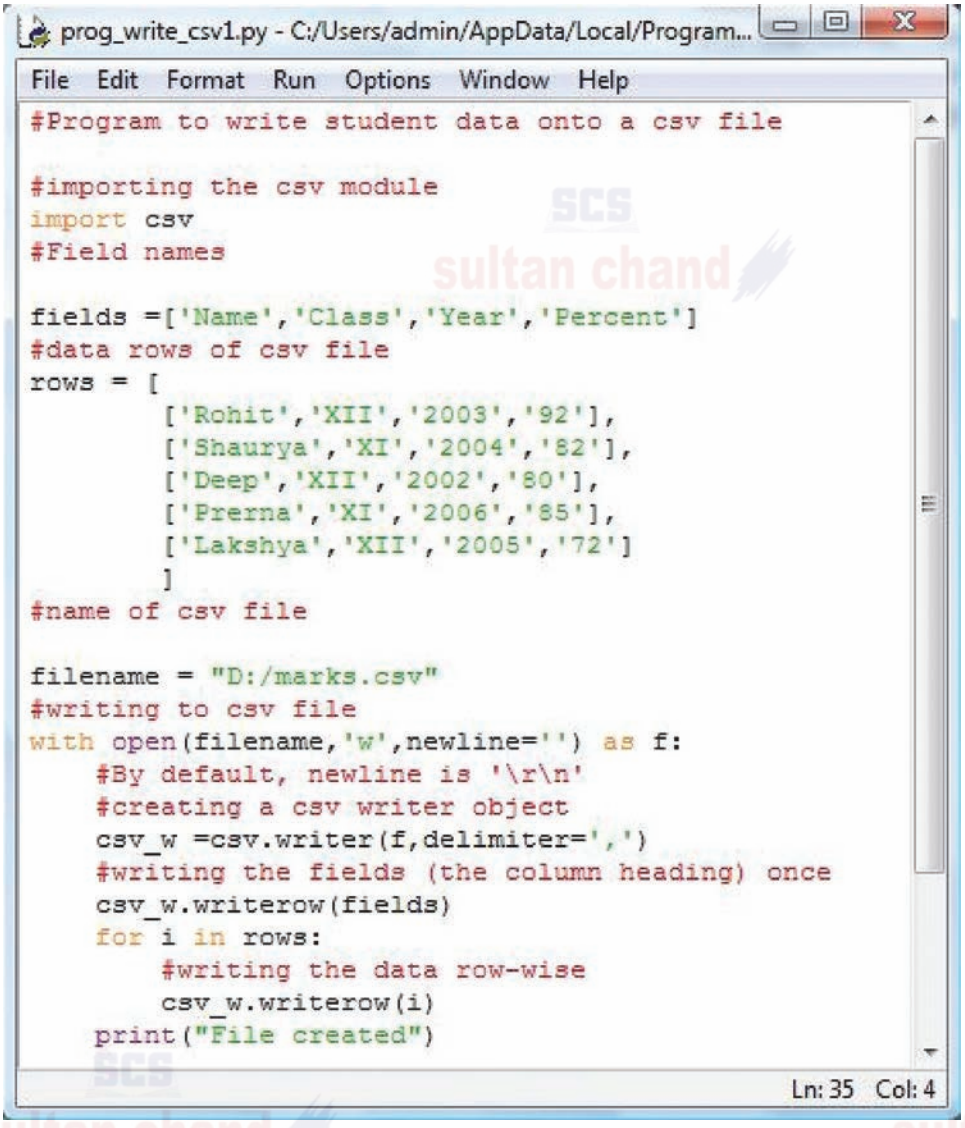
In order to write to a CSV file, we create a special type of object to write to the CSV file "**writer object**", which is defined in the CSV module, and which we create using the writer() function.

The **writerow()** method allows us to write a list of fields to the file. The fields can be strings or numbers or both. Also, while using writerow(), you do not need to add a new line character (or other EOL indicator) to indicate the end of the line, writerow() does it for you as necessary.

Let us write data onto a CSV file using writerow() method.

Practical Implementation–11

Program to write data onto "student" CSV file using writerow() method.



OBJECTIVE TYPE QUESTIONS

1. Fill in the blanks.

(a) ..... format is a text format, accessible to all applications across several platforms.
(b) ..... method is used for random access of data in a CSV file.
(c) ..... method of pickle module is used to write an object into binary file.
(d) ..... method of pickle module is used to read data from a binary file.
(e) ..... statement is given for importing csv module into your program.
(f) ..... is a string method that joins all values of each row with comma separator in CSV.
(g) ..... object contains the number of the current line in a CSV file.
(h) To end all the file contents in form of a list, ... method is used.
(i) To read all the file contents, ..., method may be used.
(j) To force Python to write the contents of file buffer on to storage file, ... method may be used.

Answers: (a) CSV (b) seek() (c) dump()
(d) load() (e) import csv (f) join()
(g) line_num (h) readlines() (i) writelines()
(j) flush()

2. State whether the following statements are True or False.

(a) CSV module can handle CSV files correctly regardless of the operating system on which the files were created.
(b) CSV module gets automatically imported into your program for reading a CSV file.
(c) The type of operation that can be performed on a file depends upon the file mode in which it is opened.
(d) Functions readline() and readlines() are essentially the same.
(e) Every record in a CSV file is stored in reader object in the form of a list.
(f) writerow() method allows us to write a list of fields to the CSV file.
(g) Comma is the default delimiter for a CSV file.
(h) tell() method of Python tells us the current position within the file.
(i) The offset argument to seek() method indicates the number of bytes to be moved.
(j) If the offset value is set to 2, beginning of the file would be taken as seek position.

Answers: (a) True (b) False (c) True (d) False (e) True (f) True
(g) True (h) True (i) True (j) False

3. Multiple Choice Questions (MCQs)

(a) Which of the following is not a valid mode to open a file?
(i) ab (ii) rw (iii) r+ (iv) w+

(b) Which statement is used to change the file position to an offset value from the start?
(i) fp.seek(offset, 0) (ii) fp.seek(offset, 1)
(iii) fp.seek(offset, 2) (iv) None of the above

(c) The difference between r+ and w+ modes is expressed as?
(i) No difference
(ii) In r+ mode, the pointer is initially placed at the beginning of the file and the pointer is at the end for w+
(iii) In w+ mode, the pointer is initially placed at the beginning of the file and the pointer is at the end for r+
(iv) Depends on the operating system

(d) What does CSV stand for?
(i) Cursor Separated Variables (ii) Comma Separated Values
(iii) Cursor Separated Values (iv) Cursor Separated Version

(e) Which module is used for working with CSV files in Python?
(i) random (ii) statistics (iii) csv (iv) math

(f) Which of the following modes is used for both writing and reading from a binary file?
(i) wb+ (ii) w (iii) wb (iv) w+

(g) Which statement is used to retrieve the current position within the file?
(i) fp.seek() (ii) fp.tell() (iii) fp.loc (iv) fp.pos

(h) What happens if no arguments are passed to the seek() method?
(i) file position is set to the start of file (ii) file position is set to the end of file
(iii) file position remains unchanged (iv) results in an error

(i) Which of the following modes will refer to binary data?
(i) r (ii) w (iii) + (iv) b

(j) Every record in a CSV file is stored in reader object in the form of a list using which method?
(i) writer() (ii) append() (iii) reader() (iv) list()

Answers: (a) (ii) (b) (i) (c) (ii) (d) (ii) (e) (iii) (f) (i)
(g) (ii) (h) (iv) (i) (iv) (j) (iii)

SOLVED QUESTIONS

1. Write a Python code to find the size of the file in bytes, number of lines and number of words.

Ans.
```
# reading data from a file and find size, lines, words
f = open('Lines.txt', 'r')
str = f.read()
size = len(str)
print('size of file n bytes ', size)
f.seek(0)
L = f.readlines()
word = L.split()
print('Number of lines ', len(L))
print('Number of words ', len(word))
f.close()
```

2. Consider the following code:
```
f = open("test", "w+")
f.write("0123456789abcdef")
f.seek(-3,2)          //Statement 1
print(f.read(2))      //Statement 2
```
Explain statement 1 and give output of statement 2.

Ans. Statement 1 uses seek() method that can be used to position the file object at a particular place in the file.

It's syntax is:
```
fileobject.seek(offset [, from_what])
```
So, f.seek(-3,2) positions the fileobject to 3 bytes before end of file.

Output of Statement 2 is:
```
de
```
It reads 2 bytes from where the file object is placed.

3. Yogendra intends to position the file pointer to the beginning of a text file. Write Python statement for the same assuming "F" is the Fileobject.

Ans.
```
F.seek(0)
```

4. Differentiate between file modes r+ and rb+ with respect to Python.

Ans. r+ opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+ opens a file for both reading and writing in binary format. The file pointer is placed at the beginning of the file.

5. In which of the following file modes the existing data of the file will not be lost?
rb, ab, w, w+b, a+b, wb, wb+, w+, r+

Ans. In file modes rb, ab, a+b and r+, data will not be lost.
In file modes w, w+b, wb, wb+ and w+, data will be truncated *i.e.* lost.

12. Write a Python program to read specific columns from a "department.csv" file and print the content of the columns, department id and department name.

Ans.
```
import csv
with open('departments.csv', newline=' ') as csvfile:
    data = csv.reader(csvfile)
    print("ID Department Name")
    print("--------------------------------")
    for row in data:
        print(row['department_id'], row['department_name'])
```

13. Explain briefly the CSV format of storing files.

Ans. The acronym CSV is short for Comma-Separated Values, which refers to a tabular data saved as plaintext where data values are separated by commas. In CSV format:

- ☞ Each row of the table is stored in one row, *i.e.*, the number of rows in a CSV file are equal to the number of rows in the table, (or sheet or database table etc.).
- ☞ The field values of a row are stored together with commas after every field value; but after the last field's value, no comma is given, just the end of line.

14. Write a menu-driven program implementing user-defined functions to perform different functions on a csv file "student" such as:

(a) Write a single record to csv

(b) Write all the records in one single go onto the csv.

(c) Display the contents of the csv file.

Ans.
```
import csv
# To create a CSV File by writing individual lines
def CreateCSV1():
    # Open CSV File
    Csvfile = open('student.csv', 'w', newline='')
    # CSV Object for writing
    Csvobj = csv.writer(Csvfile)
    while True:
        Rno = int(input("Rno:"))
        Name = input("Name:")
        Marks = float(input("Marks:"))
        Line = [Rno, Name, Marks]
        # Writing a line in CSV file
        Csvobj.writerow(Line)
        Ch = input("More(Y/N)?")
        if Ch == 'N':
            break
    Csvfile.close()          # Closing a CSV File

# To create a CSV File by writing all lines in one go
def CreateCSV2():
    # Open CSV File
    Csvfile = open('student.csv', 'w', newline='')
    # CSV Object for writing
    Csvobj =csv.writer(Csvfile)
    Lines = []
    while True:
        Rno = int(input("Rno:"))
        Name = input("Name:")
        Marks = float(input("Marks:"))
        Lines.append([Rno, Name, Marks])
        Ch = input("More(Y/N)?")
        if Ch == 'N':
            break
```

```
    # Writing all lines in CSV file
    Csvobj.writerows(Lines)
    Csvfile.close()          # Closing a CSV File

# To read and show the content of a CSV File
def ShowAll():
    # Opening CSV File for reading
    Csvfile = open('student.csv', 'r', newline='')
    # Reading the CSV content in object
    Csvobj = csv.reader(Csvfile)
    for Line in Csvobj:    # Extracting line by line content
        print(Line)
    Csvfile.close()          # Closing a CSV File

print("CSV File Handling")
while True:
    Option = input("1:CreateCSV 2:CreateCSVAll 3:ShowCSV 4:Quit ")
    if Option == "1":
        CreateCSV1()
    elif Option == "2":
       CreateCSV2()
    elif Option == "3":
        ShowAll()
    else:
        break
```

15. (a) Create a binary file "employee" that stores the records of employees and display them one by one.

(b) Display the records of all those employees who are getting salaries between 25000 to 30000.

Ans. (a)
```
import pickle
f1 = open('emp.dat','rb')
e = pickle.load(f1)
for x in e:
    print(x)
f1.close()
```

(b)
```
import pickle
f1 = open('emp.dat','rb')
e = pickle.load(f1)
for x in e:
    if(e[x]>=25000 and e[x]<=30000):
        print(x)
f1.close()
```

16. What is the output of the following code?

```
fh = open("test.txt", "r")
Size = len(fh.read())
print(fh.read(5))
```

Ans. No output.

Explanation. The fh.read() of line 2 will read the entire file content and place the file pointer at the end of file. For the fh.read(5), it will return nothing as there are no bytes to be read from EOF. Thus print() statement prints nothing.

UNSOLVED QUESTIONS

1. Write appropriate statements to do the following:

(a) To open a file named "RESULT.DAT" for output.

(b) To go to the end of the file at any time.

2. Write a program to add two more employees details to the file "emp.txt" already stored in disk.
3. How are the following codes different from one another?

(a)
```
fp = open("file.txt", 'r')
fp.read()
```

(b)
```
fp=open("file.txt", 'r')
```

4. What is the output of the following code fragment? Explain.

```
fout = file("output.txt", 'w')
fout.write("Hello, world!\n")
fout.write("How are you?")
fout.close()
file("output.txt").read()
```

5. Write the output of the following code with justification if the contents of the file ABC.txt are:
Welcome to Python Programming!

```
f1 = file("ABC.txt", "r")
size = len(f1.read())
print(size)
data = f1.read(5)
print(data)
```

6. Anant has been asked to display all the students who have scored less than 40 for Remedial Classes. Write a user-defined function to display all those students who have scored less than 40 from the binary file "Student.dat".
7. Give the output of the following snippet:

```
import pickle
list1, list2 = [2, 3, 4, 5, 6, 7, 8, 9, 10], []
for i in list1:
    if (i%2==0 and i%4==0):
        list2.append(i)
f = open("bin.dat","wb")
pickle.dump(list2, f)
f.close()
f = open("bin.dat", "rb")
data = pickle.load(f)
f.close()
for i in data:
    print(i)
```

8. Following is the structure of each record in a data file named "PRODUCT.DAT".

```
{"prod_code": value, "prod_desc": value, "stock": value}
```

The values for prod_code and prod_desc are strings, and the value for stock is an integer.
Write a function in Python to update the file with a new value of stock. The stock and the product_code, whose stock is to be updated, are to be inputted during the execution of the function.

9. Given a binary file “STUDENT.DAT”, containing records of the following type:

```
[S_Admno, S_Name, Percentage]
```

Where these three values are:

`S_Admno` – Admission Number of student (string)
`S_Name` – Name of student (string)
`Percentage` – Marks percentage of student (float)

Write a function in Python that would read contents of the file “STUDENT.DAT” and display the details of those students whose percentage is above 75.