# Chapter 13
# Python and CSV Files

---

## PART – I
## I. Choose The Correct Answer

**Question** 1.
A CSV file is also known as a
(a) Flat File
(b) 3D File
(c) String File
(d) Random File
**Answer**:
(a) Flat File

**Question** 2.
The expansion of CRLF is ……
(a) Control Return and Line Feed
(b) Carriage Return and Form Feed
(c) Control Router and Line Feed
(d) Carriage Return and Line Feed
**Answer**:
(d) Carriage Return and Line Feed

**Question** 3.
Which of the following module is provided by Python to do several operations on the CSV files?
(a) py
(b) xls
(c) csv
(d) os
**Answer**:
(c) csv

**Question** 4.
Which of the following mode is used when dealing with non-text files like image or exe files?
(a) Text mode
(b) Binary mode
(c) xls mode
(d) csv mode
**Answer**:

**Question** 5.
The command used to skip a row in a CSV file is ……..
(a) next( )
(b) skip( )
(c) omit( )
(d) bounce( )
**Answer**:
(b) skip( )

**Question** 6.
Which of the following is a string used to terminate lines produced by writer( )method of csv module?
(a) Line Terminator
(b) Enter key
(c) Form feed
(d) Data Terminator
**Answer**:
(a) Line Terminator

**Question** 7.
What is the output of the following program?
import csv
d=csv.reader(open('c:\PYPRG\chl3\city.csv'))
next(d)
for row in d:
print(row)
if the file called "city.csv" contain the following details
chennai,mylapore
mumbai,andheri
(a) chennai,mylapore
(b) mumbai,andheri
(c) chennai, mumbai
(d) chennai,mylapore,mumbai,andheri
**Answer**:
(b) mumbai,andheri

**Question** 8.
Which of the following creates an object which maps data to a dictionary?
(a) listreader( )
(b) reader( )
(c) tuplereader( )
(d) DictReader( )
**Answer**:
(d) DictReader( )

**Question** 9.
Making some changes in the data of the existing file or adding more data is called
(a) Editing
(b) Appending
(c) Modification
(d) Alteration
**Answer**:
(c) Modification

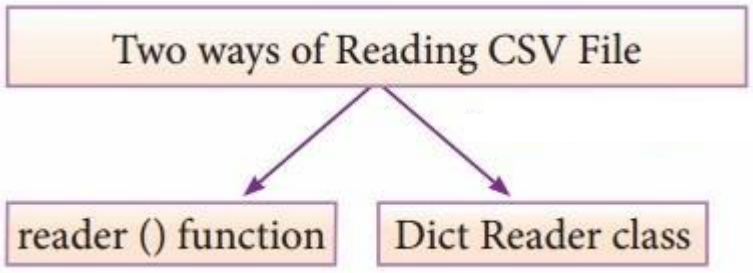
**Question** 10.
What will be written inside the file test.csv using the following program import csv
D = [['Exam'],['Quarterly'],['Halfyearly']]
csv.register_dialect('M',lineterminator = '\n')
with open('c:\pyprg\chl3\line2.csv', 'w') as f:
wr = csv.writer(f,dialect='M')
wr.writerows(D)
f.close( )
(a) Exam Quarterly Halfyearly
(b) Exam Quarterly Halfyearly
(c) Q H
(d) Exam, Quarterly, Halfearly
**Answer**:
(d) Exam, Quarterly, Halfearly

## PART – II
## II. Answer The Following Questions

**Question** 1.
What is CSV File?
**Answer**:
A CSV file is a human readable text file where each line has a number of fields , separated by commas or some other delimiter. A CSV file is also known as a Flat File. Files in the CSV format can be imported to and exported from programs that store data in tables, such as Microsoft Excel or OpenOfficeCalc.

**Question** 2.
Mention the two ways to read a CSV file using Python?
**Answer**:
Read a CSV File Using Python
There are two ways to read a CSV file.

1. Use the csv module's reader function
2. Use the DictReader class.




**Question** 3.
Mention the default modes of the File?
**Answer**:
You can specify the mode while opening a file. In mode, you can specify whether you want to read 'r', write 'w' or append 'a' to the file. You can also specify "text or binary" in which the file is to be opened.
The default is reading in text mode. In this mode, while reading from the file the data w'ould . be in the format of strings.

**Question** 4.
What is use of next( ) function?
**Answer**:
# skipping the first row(heading)
Example: next( reader)

**Question** 5.
How will you sort more than one column from a csv file? Give an example statement?
**Answer**:
To sort by more than one column you can use itemgetter with multiple indices: operator .itemgetter (1,2).
#using operator module for sorting multiple columns
sortedlist = sorted (data, key=operator.itemgetter(1))

## PART – III
## III. Answer The Following Questions

**Question** 1.
Write a note on open( ) function of python. What is the difference between the two methods?
**Answer**:
Python has a built-in function open() to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.
For Example
>>> f = openf'sample.txt") bopen file in current directory andf is file object
>>> f = open('c:\ \pyprg\ \chl3sample5.csv') #specifyingfull path

You can specify the mode while opening a file. In mode, you can specify whether you want to read 'r', write 'w' or append 'a' to the file, you can also specify "text or binary" in which the file is to be opened.
The default is reading in text mode. In this mode, while reading from the file the data would be in the format of strings.
On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

```
f = open("test.txt") # since no mode is specified the default mode it is used
#perform file operations
f.close( )
```

The above method is not entirely safe. If an exception occurs when you are performing some operation with the file, the code exits without closing the file. The best way to do this is using the "with" statement. This ensures that the file is closed when the block inside with is exited. You need not to explicitly call the close() method. It is done internally.

**Question** 2.
Write a Python program to modify an existing file?
**Answer**:

```
import csv
row= ['3', 'Meena', 'Bangalore']
with open('student.csv', 'r') as readFile:
reader= csv.reader(readFile)
lines =list(reader) # list( ) – to store each row of data as a list
lines[3] =row
with open('student.csv', 'w') as writeFile:
# returns the writer object which converts the user data with delimiter
writer= csv.writer(writeFile)
#writerows( )method writes multiple rows to a csv file
writer, writerows(lines)
readFile.close( )
writeFile.close( )
```

When we Open the student.csv file with text editor, then it will show:

| Roll No | Name | City |
|---|---|---|
| 1 | Harshini, | Chennai |
| 2 | Adhith, | Mumbai |
| 3 | Dhuruv, | Bengaluru |
| 4 | egiste, | Tiruchy |
| 5 | Venkat, | Madurai |

**Question** 3.
Write a Python program to read a CSV file with default delimiter comma (,)?

**Answer**:
CSV file with default delimiter comma(,)
The following program read a file called "sample l.csv" with default delimiter cpmma(,) and print row by row.

```
#importing csv
import csv
#opening the csv fde which is in different location with read mode
with open('c:\ \pyprg\\sample l.csv', V) as F:
#other way to open the file is f = ('c:\\pyprg\\sample l.csv', 'r')
reader = csv.reader(F)
Sprinting each line of the Data row by row print(row)
F.close( )
```

OUTPUT

```
['SNO', 'NAME', 'CITY']
[12101','RAM', 'CHENNAI']
['12102','LAVANYA','TIRUCHY']
['12103','LAKSHMAN','MADURA']
```

**Question** 4.
What is the difference between the write mode and append mode?
**Answer**:
Append mode write the value of row after the last line of the "student.csv file:"
The 'w' write mode creates a new file. If the file is already existing 'w' mode over writs it. Where as 'a' append mode add the data at the end of the file if the file already exists otherwise creates a new one.

**Question** 5.
What is the difference between reader( ) and DictReader( ) function?
**Answer**:
Reading CSV File Into A Dictionary:
To read a CSV file into a dictionary can be done by using DictReader class of csv module which works similar to the reader( ) class but creates an object which maps data to a dictionary. The keys are given by the fieldnames as parameter. DictReader works by reading the first line of the CSV and using each comma separated value in this line as a dictionary key.

The columns in each subsequent row then behave like dictionary values and can be accessed with the appropriate key (i.e. fieldname). The main difference between the csv.reader( ) and DictReader( ) is in simple terms csv. reader and csv.writer work with list/tuple, while csv.DictReader and csv.DictWriter work ' with dictionary. csv.DictReader and csv.DictWriter take additional argument fieldnames that are used as dictionary keys.

# PART – IV
# IV. Answer The Following Questions

**Question** 1.
Differentiate Excel file and CSV file?
**Answer**:
The difference between Comma-Separated Values (CSV) and eXceL Sheets(XLS) file formats is
**Answer**:
Excel:

1. Excel is a binary file that holds information about all the worksheets in a file, including both content and formatting
2. XLS files can only be read by applications that have been especially written to read their format, and can only be written in the same way.
3. Excel is a spreadsheet that saves files into its own proprietary format viz. xls or xlsx
4. Excel consumes more memory while importing data

CSV:

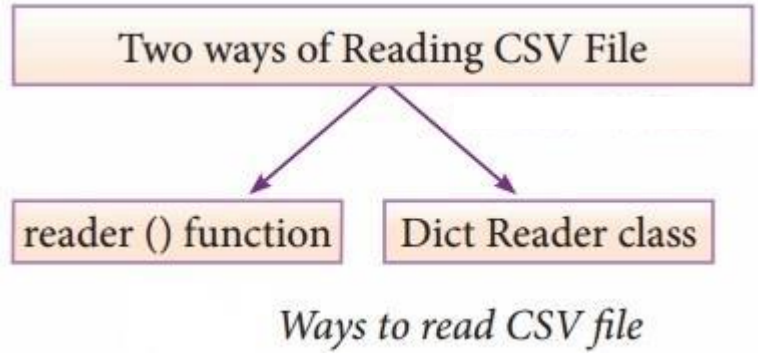
1. CSV format is a plain text format with a series of values separated by commas.
2. CSV can be opened with any text editor in Windows like notepad, MS Excel, OpenOffice, etc.
3. CSV is a format for saving tabular information into a delimited text file with extension .csv
4. Importing CSV files can be much faster, and it also consumes less memory

**Question** 2.
Tabulate the different mode with its meaning?
Python File Modes:
**Answer**:

| Mode | Description |
|---|---|
| 'r' | Open a file for reading. (default) |
| 'w' | Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists. |
| 'x' | Open a file for exclusive creation. If the file already exists, the operation fails. |
| 'a' | Open for appending at the end of the file without truncating it. Creates a new file if it does not exist. |
| 't' | Open in text mode. (default) |
| 'b' | Open in binary mode. |
| '+' | Open a file for updating (reading and writing) |

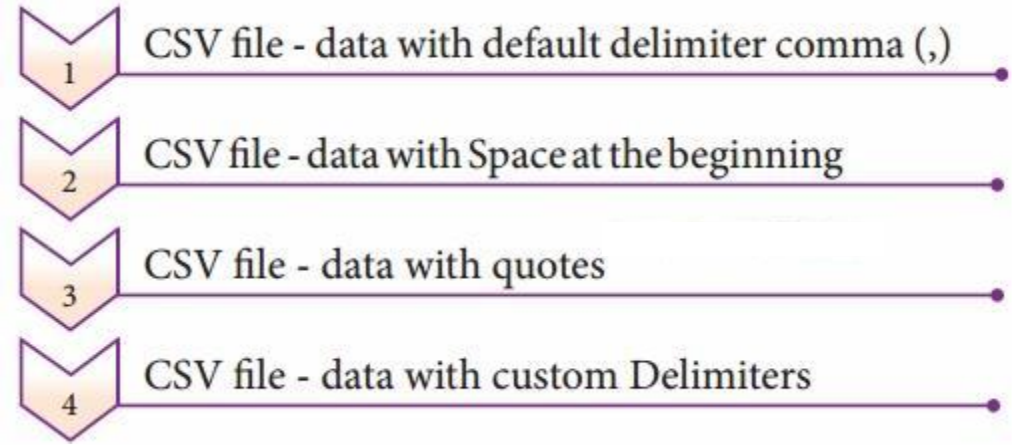
**Question** 3.
Write the different methods to read a File in Python?
**Answer**:
Read a CSV File Using Python
There are two ways to read a CSV file.

1. Use the csv module's reader function
2. Use the DictReader class.



*Ways to read CSV file*

CSV Module's Reader Function:
You can read the contents of CSV file with the help of csv.reader( )method. The reader function is designed to take each line of the file and make a list of all columns. Then, you just choose the column you want the variable data for. Using this method one can read data from csv files of different formats like quotes (" "),pipe (|) and comma(,).
The syntax for csv.reader( ) is
csv. reader (fileobject, delimiter,fmtparams)
where
file object:- passes the path and the mode of the file
delimiter:- an optional parameter containing the standard dilects like, I etc can be omitted
fmtparams:- optional parameter which help to override the default values of the dialects like skipinitialspace,quoting etc. Can be omitted




CSV file with default delimiter comma(,)
The following program read a file called "sample l.csv" with default delimiter comma(,) and print row by row.
#importing csv

```
import csv
#opening the csv file which is in different location with read mode
with open('c:\ \pyprg\\sample l.csv', 'r') as F:
#other way to open the file is f= ('c:\\pyprg\\sample l.csv', 'r)
reader = csv.reader(F)
#printing each line of the Data row by row
print( row)
F.close( )
```

OUTPUT

```
['SNO', 'NAME', 'CITY']
['12101','RAM','CHENNAI']
['12102', 'LAVANYA', 'TIRUCHY']
['12103', 'LAKSHMAN', 'MADURAI']
```

CSV files- data with Spaces at the beginning

Consider the following file "sample 2.csv" containing the following data when opened through notepad

| Topic 1 | Topic 2 | Topic 3 |
|---|---|---|
| one | two | three |
| Example 1 | Example 2 | Example 3 |

The following program read the file through Python using "csv.reader( )".

```
import csv
csv.register_dialect('myDialect',delimiter = ',',skipimtialspace=True)
F=open('c:\ \pyprg\ \sample 2.csv','r')
reader= csv.reader(F, dialect='myDialect')
for row in reader:
print( row)
F.close( )
```

OUTPUT

```
['Topic 1', 'Topic 2', 'Topic 3']
[!one', 'two', 'three']
['Example 1', 'Example 2', 'Example 3']
```

As you can see in "sample 2.csv" there are spaces after the delimiter due to which the output is also displayed with spaces.

These whitespaces can be removed, by registering new dialects using csv.register_dialect( ) class of csv module. A dialect describes the format of the csv file that is to be read. In dialects the parameter "skipinitialspace" is used for removing whitespaces after the delimiter.

CSV File-Data With Quotes

You can read the csv file with quotes, by registering new dialects using csv.register_dialect( ) class of csv module.

Here, we have quotes.csv file with following data.

```
SNO,Quotes
```

(a) “The secret to getting ahead is getting started.”
(b) “Excellence is a continuous process and not an accident.”
(c) “Work hard dream big never give up and believe yourself.”
(d) “Failure is the opportunity to begin again more intelligently.”
(e) “The successful warrior is the average man, with laser-like focus.”
The following Program read “quotes.csv” file, where delimiter is comma (,)but the quotes are within quotes (“”). import csv

```
csv.register_dialect('myDialect',delimiter = ',',quoting=csv.QUOTE_ALL,
skipinitialspace=True)
f=open('c:\\pyprg\ \quotes.csv','r')
reader= csv.reader(f, dialect-myDialect')
for row in reader:
print (row)
```

OUTPUT

```
['SNO', 'Quotes']
[(a), 'The secret to getting ahead is getting started.']
[(b), 'Excellence is a continuous process and not an accident.']
[(c), 'Work hard dream big never give up and believe yourself.']
[(d), 'Failure is the opportunity to begin again more intelligently.']
[(e), 'The successful warrior is the average man, with laser-like focus. ']
```

In the above program, register a dialect with name myDialect. Then, we used csv. QUOTE_ ALL to display all the characters after double quotes.
CSV files with Custom Delimiters
You can read CSVfile having custom delimiter by registering a new dialect with the help of csv.register_dialect( ).
In the following file called “sample 4.csv”,each column is separated with | (Pipe symbol)

| Roll No | Name | City |
|---|---|---|
| 12101 | Arun | Chennai |
| 12102 | Meena | Kovai |
| 12103 | Ram | Nellai |
| 103 | Ayush | M |
| 104 | Abinandh | M |

The following program read the file “sample4.csv” with user defined delimiter “|”

```
import csv
csv.register_dialect('myDialect', delimiter '|')
with open('c:\\pyprg\\sample 4.csv', 'r') as f:
reader= csv.reader(f, dialect-myDialect')
for row in reader:
print(row)
```

f.close( )
OUTPUT
['RollNo', 'Name', 'City']
['12101', 'Arun', 'Chennai']
['12102', 'Meena', 'Kovai']
[T21031,'Ram','Nellai']

Reading CSV File Into A Dictionary:
To read a CSV file into a dictionary can be done by using DictReader class of csv module which works similar to the reader() class but creates an object which maps data to a dictionary. The keys are given by the fieldnames as parameter. DictReader works by reading the first line of the CSV and using each comma separated value in this line as a dictionary key. The columns in each subsequent row then behave like dictionary values and can be accessed with the appropriate key (i.e. fieldname).

If the first row of your CSV does not contain your column names, you can pass a fieldnames parameter into the DictReader's constructor to assign the dictionary keys manually. The main difference between the csv.reader( ) and DictReader( ) is in simple terms csv. reader and csv.writer work with list/tuple, while csv.DictReader and csv.DictWriter work with dictionary. csv.DictReader and csv.DictWriter take additional argument fieldnames that are used as dictionary keys.

For Example:
Reading "sample 8.csv" file into a dictionary
```
import csv
filename= 'c:\ \pyprg\ \sample 8.csv'
input_file =csv.DictReader(open(filename;'r'))
for row in input_file:
print(dict(row))
print (data(row)) #dict( ) to print data
```
OUTPUT
{'ItemName '^Keyboard', 'Quantity': '48'}
{'ItemName VMonitor', 'Quantity': '52'}
{'ItemName VMouse', 'Quantity': '20'}
In the above program, DictReader( ) is used to read "sample 8.csv" file and map into a dictionary. Then, the function dict( ) is used to print the data in dictionary format without order. Remove the dict( ) function from the above program and use print(row). Check you are getting
the following output
OrderedDict([('ItemName Keyboard'), ('Quantity', '48')])
OrderedDict([('ltemName ','Monitor'), ('Quantity', '52')])
OrderedDict([('ItemName ','Mouse'), ('Quantity', '20')])

**Question** 4.
Write a Python program to write a CSV File with custom quotes?
CSV File with quote characters
**Answer**:
You can write the CSV file with custom quote characters, by registering new dialects using csv.register_dialect( ) class of csv module,

```
import csv
csvData = [['SNO',Items'], ['l','Pen'], ['2','Book'], ['3','Pencil']]
csv.register_dialect('myDialect', delimiter = '|',quotechar = "",
quoting=csv.QUOTE_ALL)
with open('c:\\pyprg\ \chl3\\quote.csv', 'w') as csvFile:
writer= csv.writer(csvFile, dialect-myDialect')
writer, write rows(csvData)
print("writing completed")
csvFile.close( )
```

When you open the "quote.csv" file in notepad, we get following output:

| Sl.No | "Items" |
|---|---|
| 1 | "Pen" |
| 2 | "Book" |
| 3 | "Pencil" |

In the above program, myDialect uses pipe (|) as delimiter and quotechar as doublequote "" to write inside the file.

**Question** 5.
Write the rules to be followed to format the data in a CSV file?
**Answer**:
Rules to be followed to format data in a CSV file:
(i) Each record (row of data) is to be located on a separate line, delimited by a line break by pressing enter key. For example:
xxx,yyy
denotes enter Key to be pressed

(ii) The last record in the file may or may not have an ending line break. For example:
PPP, qqq
yyy,xxx

(iii) There may be an optional header line appearing as the first line of the file with the same format as normal record lines. The header will contain names corresponding to the fields in the file and should contain the same number of fields as the records in the rest of the file. For example: field_ name 1,field_name 2,field_name_3 aaa,bbb,ccc
zzz,yyy,xxx CRFF( Carriage Return and Line feed)

(iv) Within the header and each record, there may be one or more fields, separated by commas. Spaces are considered part of a field and should not be ignored. The last field in the record must not be followed by a comma. For example: Red, Blue

(v) Each field may or may not be enclosed in double quotes. If fields are not enclosed with double quotes, then double quotes may not appear inside the fields. For example:
“Red”,”Blue”,”Green” #Field data with double quotes
Black,White,Yellow #Field data without double quotes

(vi) Fields containing line breaks (CRLF), double quotes, and commas should be enclosed in double-quotes. For example:
Red, “;:Blue CRLF #comma itself is a field value, so it is enclosed with double quotes Red, Blue, Green

(vii) If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be preceded with another double quote. For example:
“Red,” “Blue1: “Green”, #since double quotes is a field value it is enclosed with another double quotes,, White