5.7 WORKING WITH CSV FILES

You know that CSV files are delimited files that store tabular data (data stored in rows and columns as we see in spreadsheets or databases) where comma delimits every value, *i.e.*, the values are separated with comma. **Typically the default delimiter of CSV files is the comma (,) but modern implementations of CSV files allow you to choose a different delimiter character other than comma. Each line in a CSV file is a data record.** Each record consists of one or more fields, separated by commas (or the chosen delimiter).

Since CSV files are the *text files*, you can apply text file procedures on these and then split values using **split( )** function BUT there is a better way of handling CSV files, which is – using *csv module of Python*. In this section, we are going to talk about how you can read and write in CSV files using the *csv module* methods.

NOTE

The separator character of CSV files is called a delimiter. Default and most popular delimiter is comma. Other popular delimiters include the tab (\t), colon (:), pipe (|) and semi-colon (;) characters.

Python csv Module

The **csv** module of Python provides functionality to read and write tabular data in CSV format. It provides *two* specific types of objects – the **reader** and **writer** objects – to read and write into CSV files. The csv module's **reader** and **writer** objects read and write delimited sequences as records in a CSV file.

You will practically learn to use the reader and writer objects and the functions to read and write in CSV files in the coming sub-sections. But before you use **csv** module, make sure to import it in your program by giving a statement as :

```
import csv
```

5.7.1 Opening/Closing CSV Files

A CSV file is opened in the same way as you open any other text file (as explained in section 5.3 earlier), but make sure to do the following *two* things :

(*i*) Specify the file extension as **.csv**

(*ii*) Open the file like other text files, *e.g.*,

```
Dfile = open("stu.csv", "w")
```
← *CSV file opened in **write mode** with file handle as **Dfile***

Or

```
File1 = open("stu.csv", "r")
```
← *CSV file opened in **read mode** with file handle as **File1***

An open CSV file is closed in the same manner as you close any other file, *i.e.*, as :

```
Dfile.close()
```

Like text files, a CSV file will get created when opened in an output file mode and if it does not exist already. That is, for the file modes, "*w*", "*w*+", "*a*" "*a*+", the file will get created if it does not exist already and if it exists already, then the file modes "*w*" and "*w*+"will overwrite the existing file and the file mode "*a*" or "*a*+" will retain the contents of the file.

NOTE

The csv files are popular because of these reasons : (*i*) Easier to create, (*ii*) preferred export and import format for databases and spreadsheets, (*iii*) capable of storing large amounts of data.

5.7.1A Role of Argument newline in Opening of csv Files

While opening csv files, in the **open( )**, you can specify an additional argument **newline**, which is an optional but important argument. The role of **newline** argument is to specify how would Python handle newline characters while working with csv files.

As csv files are text files, while storing them, some types of translations occur – such as translation of **end of line (EOL) character** as per the operating system you are working on etc. Different operating systems store EOL characters differently. The MS-DOS (including Windows and OS/2), UNIX, and Macintosh operating systems all use different characters to designate the end of a line within a text file. Following table 5.4 lists the EOL characters used by different operating systems.

Table 5.4 *EOL characters used in different operating systems.*

Symbol/Char	Meaning	Operating System
CR [\r]	Carriage Return	Macintosh
LF [\n]	Line Feed	UNIX
CR/LF [\r \n]	Carriage Return/Line Feed	MS-DOS, Windows, OS/2
NULL [\0]	Null character	Other OSs

Now what would happen if you create a csv file on one operating system and use it on another. The EOL of one operating system will not be treated as EOL on another operating system. Also, if you have given a character, say '\r', in your text string (not as EOL but as part of a text string) and if you try to open and read from this file on a Macintosh system, what would happen? Mac OS will treat every '\r' as EOL – yes, including the one you gave as part of the text string.

So what is the solution? Well, the solution is pretty simple. Just suppress the EOL translation by specify third argument of **open( )** as newline = ' ' (null string – no space in quotes).

If you specify the **newline** argument while writing onto a csv file, it will create a csv file with no EOL translation and you will be able to use csv file in normal way on any platform.

NOTE

Additional optional argument as newline = '' (null string; no space in between) with file open( ) will ensure that ***no translation of end of line (EOL) character*** takes place.

That is, open your csv file as :

```
Dfile = open("stu.csv", "w", newline = '')
```
null string : no space in between
*CSV file opened in **write mode** with file handle as Dfile (no EOL translation)*

Or

```
File1 = open("stu.csv", "r", newline = '')
```
*CSV file opened in **read mode** with file handle as File1 (no EOL translation)*

Not only this is useful for working across different platforms, it is also useful when you work on the same platform for writing and reading. You will be able to appreciate the role of the *newline* argument when we talk about reading from the csv files. Program 5.21 onwards it will be clear to you.

Let us now learn to work with **csv module's** methods to write / read into CSV files.

5.7.2 Writing in CSV Files

Writing into csv files involves the conversion of the user data into the writable delimited form and then storing it in the form of csv file. For writing onto a csv files, you normally use the following *three* key players functions[2].

2. Other than the above shown functions, there are other functions too like ***DictWriter( )*** function but these are beyond the scope of the syllabus, hence we shall not cover these in this chapter.

These are :

`csv.writer()`	returns a writer object which writes data into CSV file
`<writerobject>.writerow()`	writes one row of data onto the ***writer object***
`<writerobject>.writerows()`	writes multiple rows of data onto the ***writer object***

Before we proceed, it is important for you to understand the significance of the **writer object**. Since csv files are delimited flat files and before writing onto them, the data must be in **csv-writable-delimited-form**[3], it is important to convert the received user data into the form appropriate for the csv files. **This task is performed by the writer object**. The data row written to a writer object (written using **writerow( )** or **writerows( )** functions) gets converted to csv writable delimited form and then written on to the linked csv file on the disk.

Following figure 5.8 illustrates this process.

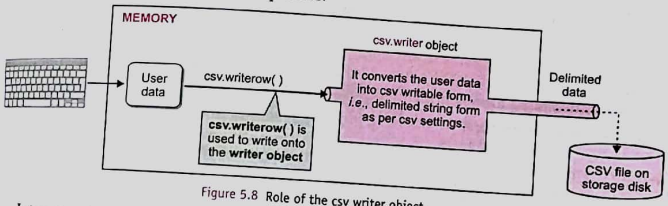


Figure 5.8 Role of the csv writer object.

Let us now learn to write onto csv files. In order to write onto a csv file, you need to do the following (**Note**, in order to appreciate the use of *newline* argument, we shall initially create csv files without the *newline* argument first and later we shall create csv files with *newline* argument too.)

(i) Import csv module

(ii) Open csv file in a file-handle (just as you open other text files), *e.g.*,

```
fh = open("student.csv", "w")
```

(iii) Create the **writer object** by using the syntax as shown below:

```
<name-of-writer-object> = csv.writer(<file-handle>, [delimiter = <delimiter character>])
```

e.g., *If you skip this argument, comma will be used as the delimiter*

```
stuwriter = csv.writer(fh)
```
you opened the file with this file handle in previous step

In the above statement, delimiter argument is not specified. When you skip the delimiter argument, the default delimiter character, which is comma, is used for separating characters. But if you specify a character with **delimiter** argument then the specified character is used as the delimiter, *e.g.*,

```
stuwriter = csv.writer(fh, delimiter = '|')
```

The above statement will create file with delimiter character as pipe symbol ('|').

1. It depends on something called ***dialect***, which by default is set for producing ***excel*** format like csv files. It can even be excel-tab where tab character is the delimiter. You can set the dialect using ***dialect*** argument but we shall only work with the default dialect and hence won't use *dialect* argument here. For further details on *dialect* argument, you may refer to the Python's documentation.

(*iv*) Obtain user data and form a Python sequence (list or tuple) out of it, *e.g.*,

```
Sturec = (11, 'Neelam', 79.0)
```

(*v*) Write the Python sequence containing user data onto the writer object using csv.writerow( ) or csv.writerows( ) functions, *e.g.*,

```
csv.writerow(Sturec)
```

Both the **writerow( )** and **writerrows( )** functions can be used for writing onto the writer object. We shall discuss about how to use **writerows( )** function little later. For now, let us focus on writing through **writerow( )** function, *i.e.*, writing single row at a time.

CSV files can also take the names of columns as header rows, which are written in the same way as any row of data is written, *e.g.*, to give column headings as "Rollno", "Name" and "Marks", you may write :

```
stuwriter.writerow( ['Rollno', 'Name', 'Marks'])
```

(*vi*) Once done, close the file.

You only need to write into the writer object. Rest of the work it will do itself, *i.e.*, converting the data into delimited form and then writing it onto the linked csv file.

Let us now do it practically. Following program illustrates this process.

Program 5.19 Write a program to create a CSV file to store student data (Rollno., Name, Marks). Obtain data from user and write 5 records into the file.

```
import csv
fh = open("Student.csv", "w")              #open file
stuwriter = csv.writer(fh)
stuwriter.writerow(['Rollno', 'Name', 'Marks'] )        #write header row
for i in range(5):
    print("Student record", (i+1) )
    rollno = int(input("Enter rollno:"))
    name = input("Enter name:")
    marks = float(input("Enter marks:"))
    sturec = [rollno, name, marks]          #create sequence of user data
    stuwriter.writerow(sturec)
fh.close()          #close file
```

Column headings written

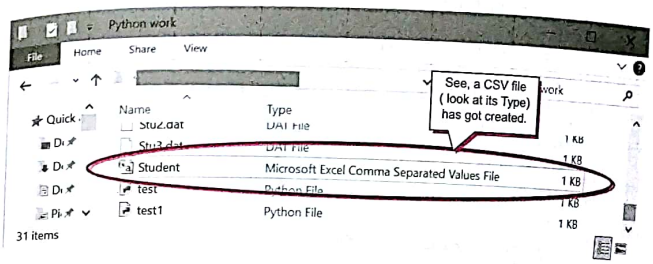
Python sequence created from user data

Python data sequence written on the writer object using writerow( )

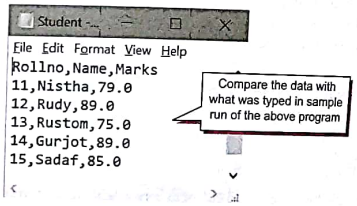
The sample run of the above program is as shown below :

```
Student record 1
Enter rollno:11
Enter name:Nistha
Enter marks:79
Student record 2
Enter rollno:12
Enter name:Rudy
Enter marks:89
Student record 3
Enter rollno:13
Enter name:Rustom
Enter marks:75
Student record 4
Enter rollno:14
Enter name:Gurjot
Enter marks:89
Student record 5
Enter rollno:15
Enter name:Sadaf
Enter marks:85
```


Now if you open the folder of your Python program, you will see that Python has created a file by the name **Student.csv**



And if you open this file in Notepad or any other ASCII editor, it shows :



The writerows( ) Function

If you have all the data available and data is not much lengthy then it is possible to write all data in one go. All you need to do is create a nested sequence out of the data and then write using the **writerows( )** function. The **writerows( )** method writes all given rows to the CSV file, *e.g.*, to write following nested sequence, you can use the **writerows( )** function :

```
Sturec = [ [11,Nistha,79.0], [12,Rudy,89.0], [13,Rustom,75.0] ]
<writerobject>.writerows(Sturec)
```

Each inner list will now be written as a separate record in the csv file.

Following program illustrates this.

Program 5.20 The data of winners of four rounds of a competitive programming competition is given as :

```
['Name', 'Points', 'Rank']
['Shradha', 4500, 23]
['Nishchay', 4800, 31]
['Ali', 4500, 25]
['Adi', 5100, 14]
```

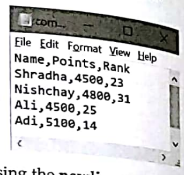
Write a program to create a csv file (compresult.csv) and write the above data into it.

```
import csv
fh = open("compresult.csv", "w")
cwriter = csv.writer(fh)
compdata = [
    ['Name', 'Points', 'Rank'],
    ['Shradha', 4500, 23],
    ['Nishchay', 4800, 31],
    ['Ali', 4500, 25],
    ['Adi', 5100, 14] ]
cwriter.writerows(compdata)
fh.close()
```

This nested sequence contains multiple records' data in the form of inner lists

Nested list written in one go using **writerows( )**

The above program will create a csv file and in Notepad, it will look like :



You can also create the nested sequence programmatically by appending one record to a list while writing using **writerows( )** function. ***Solved problem 48 performs the same.***

Please note that till now we have created csv files without using the **newline** argument, which means that EOL translation has taken place internally. How this impacts a csv file, will become clear to you soon when we start programming in the following section.

5.7.3 Reading in CSV Files

Reading from a csv file involves loading of a csv file's data, **parsing** it (*i.e.*, removing its delimitation), loading it in a *Python iterable* and then reading from this iterable. Recall that any Python sequence that can be iterated over in a for-loop is a *Python iterable*, *e.g.*, *lists*, *tuples*, and *strings* are all Python iterables.

For reading from a csv files, you normally use following *function* :

`csv.reader()`	returns a reader object which loads data from CSV file into an iterable after parsing delimited data

The **csv.reader** object does the opposite of **csv.writer** object. The *csv.reader* object loads data from the csv file, parses it, *i.e.*, removes the delimiters and returns the data in the form of a Python iterable wherefrom you can fetch one row of data at a time. (Fig. 5.9)

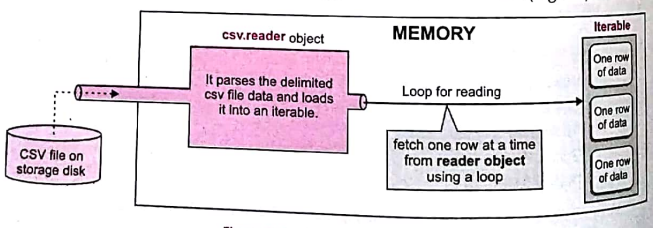


Figure 5.9 Role of csv.reader object.

Let us now learn to read from csv files. In order to read from a csv file, you need to do the following :

(*i*) Import csv module

(*ii*) Open csv file in a file-handle in read mode (just as you open other text files),

```
<file handle> = open (<csv-file>, <read mode>)
```

e.g.,

```
fh = open("student.csv", "r")
```

The file being opened must already exist otherwise an exception will get raised. Your code should be able to handle the exception *i.e.*, through **try..except**. (Alternatively you can use the **with** statement as mentioned below.)

(*iii*) Create the **reader object** by using the syntax as shown below :

```
<name-of-reader-object> = csv.reader(<file-handle>,[delimiter =<delimiter character>])
```

e.g.,

```
stureader = csv.reader(fh)
```

you opened the file with this file handle in previous step

You may also specify a delimiter character other than a comma using the **delimiter** argument, *e.g.*,

```
stureader = csv.reader(fh, delimiter = '|')
```

(*iv*) The reader object stores the parsed data in the form of iterable and thus you can fetch from it row by row through a traditional for loop, one row at a time :

```
for rec in stureader :
    print (rec)          # or do any other processing
```

Loop to fetch one row at a time in **rec** *from the iterable*

(*v*) Process the fetched single row of data as required.

(*vi*) Once done, close the file.

All the above mentioned steps are best performed through **with** statement as the **with** statement will also take care of any exception that may arise while opening/reading a file. Thus you should process the csv file as per the following syntax, which combines all the above mentioned steps :

```
with open (<csv-file>, <read mode>) as <file handle> :
    <name-of-reader-object> = csv.reader(<file-handle>)
    for <row identifier> in <reader object> :
        : # process the fetched row in <row identifier>
```

Let us do it practically.

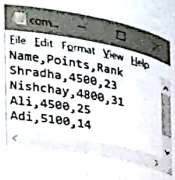
Following program is doing the same.

NOTE

Csv files are flat, text files.

Program 5.21 You created the file compresult.csv in the previous program as shown below.

Write a program to read the records of this csv file and display them.



Note. *Before we write the code for this program, recall that the given file created in the previous program,* ***was created without specifying the newline argument*** *in the file* ***open( )*** *function. So have a look at the previous program's (program 5.20) code before starting with this program's code.*

```
import csv
with open("compresult.csv", "r") as fh :
    creader = csv.reader(fh)
    for rec in creader :
        print(rec)
```

Loop to fetch one row at a time in rec from the iterable in creader

The above program will produce the result as :

```
['Name', 'Points', 'Rank']
[]
['Shradha', '4500', '23']
[]
['Nishchay', '4800', '31']
[]
['Ali', '4500', '25']
[]
['Adi', '5100', '14']
[]
```

Where have these empty rows come from ?
We never entered empty rows. Refer to the sample run of program 5.20.
Then what is the source / reason of these empty rows ?

Compare the above output with the sample run of previous program (program 5.20). We never entered empty rows. Then where have these empty rows come from? Any idea?

You guessed it right. We did not specify the **newline** argument in the file **open( )** function while creating/writing this csv file (*compresult.csv,* in this case) and thus EOL translation took place which resulted in the blank rows after every record while reading. In fact, the above shown data was internally stored as :

```
['Name', 'Points', 'Rank']\r
\n
['Shradha', '4500', '23']\r
\n
['Nishchay', '4800', '31']\r
\n
:
```

Every data record line was appended with EOL character '\r\n' on Windows OS because of EOL translation

So what is the solution ? You are right again ☺. We should have created the file with newline argument set to null string (' ') so that no EOL translation could take place.

In fact, the above situation can be handled in *two* ways :

(*i*) **Method1.** Create/write onto csv file with **newline = ' '** argument in open( ) function so this situation would not arise. ***Programs 5.23(a) and (b)*** illustrate this solution.

(*ii*) **Method2.** To read from a file which is created with EOL translation (*i.e.,* no newline argument), open it with **newline** argument set to the EOL character of the OS where the csv file was created. For instance, if you created the csv file on a Windows OS, open file for reading with **newline** argument set as '\r\n' because the EOL on Windows is stored as '\r\n', *i.e.,* as :

Now the file will be read considering every '\r\n' character as newline.

```
open(<csvfilename>, <readmode>, newline = '\r\n')
```

Recall that the table 5.4 lists EOL characters on various operating systems.

Program 5.22 handles this situation using method 2 listed here. Let us have a look at it.

Program 5.22 The csv file (compresult.csv) used in the previous program was created on Windows OS where the EOL character is '\r\n'. Modify the code of the previous program so that blank lines for every EOL are not displayed.

```
import csv
with open("compresult.csv", "r", newline = '\r\n') as fh :
    creader = csv.reader(fh)
    for rec in creader :
        print(rec)
```

Now the file will be read considering every '\r\n' character as end of line and not as a separate line

The output produced by above program is :

```
['Name', 'Points', 'Rank']
['Shradha', '4500', '23']
['Nishchay', '4800', '31']
['Ali', '4500', '25']
['Adi', '5100', '14']
```

See, no blank lines this time

The newline argument can be specified in independent open( ) statement as well as in the open( ) of with statement.

Let us now treat the earlier listed problem of blank lines in between records using the method 1 listed above. For this, we shall create the file with newline argument and then we shall not need any newline argument while reading as no EOL translation would have taken place.

Programs 5.23(*a*) and (*b*) are illustrating the same.

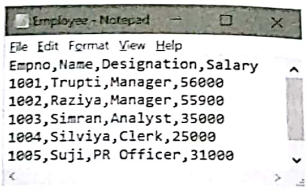
Program 5.23(a) Write a program to create a csv file by suppressing the EOL translation.

```
import csv
fh = open("Employee.csv", "w", newline = '')
ewriter = csv.writer(fh)
empdata = [
    ['Empno','Name','Designation','Salary'],
    [1001,'Trupti','Manager',56000],
```

This argument ensures that no EOL translation takes place

```
    [1002,'Raziya','Manager',55900],
    [1003,'Simran','Analyst',35000],
    [1004,'Silviya','Clerk',25000],
    [1005,'Suji','PR Officer',31000 ]
    ]
ewriter.writerows(empdata)
print("File successfully created")
fh.close()
```

The above program created a file as shown below :



Let us now read from the above created file. This time we need not specify the newline argument as no EOL translation has taken place. Program 5.23(*b*) is doing the same.

5.23(b) Write a program to read and display the contents of Employee.csv created in the previous program.

```
import csv
with open("Employee.csv", "r") as fh :
    ereader = csv.reader(fh)
    print("File Employee.csv contains :")
    for rec in ereader :
        print(rec)
```

See, we did not specify newline argument in the open( ) as no EOL translation has taken place when the file was created.

The output produced by the above program is :

```
File Employee.csv contains :
['Empno', 'Name', 'Designation', 'Salary']
['1001', 'Trupti', 'Manager', '56000']
['1002', 'Raziya', 'Manager', '55900']
['1003', 'Simran', 'Analyst', '35000']
['1004', 'Silviya', 'Clerk', '25000']
['1005', 'Suji', 'PR Officer', '31000']
```

See, no blank lines in between the records this time as the csv file was created with no EOL translation.

Check Point 5.2

1. What are text files ?
2. What are binary files ?
3. What are CSV files ?
4. Name the functions used to read and write in plain text files.
5. Name the functions used to read and write in binary files.
6. Name the functions used to read and write in CSV files.

With this we have come to the end of this chapter.

DATA FILES IN PYTHON — PriP — Progress In Python 5.1

This PriP session aims at giving you practical exposure to file handling in Python.

Fill it in PriP 5.1 under Chapter 5 of practical component-book – Progress in Computer Science with Python after practically doing it on the computer.

>>>❖<<<

LET US REVISE

- *A file in itself is a bunch of bytes stored on some storage devices like hard-disk, thumb-drive etc.*
- *The data files can be stored in three ways : (i) Text files (ii) Binary files (iii) CSV files.*
- *A text file stores information in ASCII or Unicode characters, where each line of text is terminated, (delimited) with a special character known as EOL (End of Line) character. In text files some internal manipulations take place when this EOL character is read or written.*
- *A binary file is just a file that contains information in the same format in which the information is held in memory, i.e., the file content that is returned to you is raw (with no translation or no specific encoding).*
- *The open( ) function is used to open a data file in a program through a file-object (or a file-handle).*
- *A file-mode governs the type of operations (e.g., read / write / append) possible in the opened file i.e., it refers to how the file will be used once it's opened.*
- *A text file and a csv file can be opened in these file modes : 'r' , 'w', 'a', 'r+', 'w+', 'a+'*
- *A binary file can be opened in these file modes : 'rb' , 'wb', 'ab', 'r+b' ('rb+'), 'w+b'('wb+'), 'a+b'('ab+').*
- *The three file reading functions of text files are : read( ), readline( ), readlines( )*
- *While read( ) reads some bytes from the file and returns it as a string, readline( ) reads a line at a time and readlines( ) reads all the lines from the file and returns it in the form of a list.*
- *The two writing functions for Python text files are write( ) and writelines( ).*
- *While write( ) writes a string in file, writelines( ) writes a list in a file.*
- *Pickle module's dump( ) and load( ) functions write and read into binary files.*
- *The input and output devices are implemented as files, also called standard streams.*
- *There are three standard streams : stdin (standard input), stdout (standard output) and stderr (standard error)*
- *The absolute paths are from the topmost level of the directory structure. The relative paths are relative to current working directory denoted as a dot(.) while its parent directory is denoted with two dots(..).*
- *Every open file maintains a file-pointer to determine and control the position of read or write operation in file.*
- *The seek( ) function places the file pointer at specified position.*
- *The tell( ) function returns the current position of file-pointer in open file.*
- *CSV files are delimited files that store tabular data (data stored in rows and columns as we see in spreadsheets or databases) where comma delimits every value.*
- *For writing onto a csv file, user data is written on a csv.writer object which converts the user data into delimited form and writes it on to the csv file.*
- *For reading from a csv file, **csv.reader** loads data from the csv file, parses it and makes it available in the form of an iterator wherefrom the records can be fetched row by row.*
- *CSV files should be opened with newline argument to suppress EOL translation.*

Objective Type Questions — OTQs

Multiple Choice Questions

1. Information stored on a storage device with a specific name is called a _____.
   (*a*) array (*b*) dictionary (*c*) file (*d*) tuple
2. Which of the following format of files can be created programmatically through Python to store some data ?
   (*a*) Data files (*b*) Text files
   (*c*) Video files (*d*) Binary files
3. To open a file c:\ss.txt for appending data, we use
   (*a*) file = open("c:\\ss.txt", "a") (*b*) file = open("c:\\ss.txt", "rw")
   (*c*) file = open(r"c:\ss.txt", "a") (*d*) file = open(file = "c:\ss.txt", "w")
   (*e*) file = open(file = "c:\\ss.txt", "w") (*f*) file = open("c:\res.txt")
4. To read the next line of the file from a file object *infi*, we use
   (*a*) infi.read(all) (*b*) infi.read( )
   (*c*) infi.readline( ) (*d*) infi.readlines( )
5. To read the remaining lines of the file from a file object *infi*, we use
   (*a*) infi.read(all) (*b*) infi.read( )
   (*c*) infi.readline( ) (*d*) infi.readlines( )
6. The readlines( ) method returns
   (*a*) str (*b*) a list of lines
   (*c*) a list of single characters (*d*) a list of integers
7. Which of the following mode will refer to binary data ?
   (*a*) r (*b*) w (*c*) + (*d*) b
8. In file handling, what does this term means "r, a"?
   (*a*) read, append (*b*) append, read
   (*c*) all of the mentioned (*d*) none of these
9. Which function is used to read all the characters ?
   (*a*) read( ) (*b*) readcharacters( )
   (*c*) readall( ) (*d*) readchar( )
10. Which function is used to read a single line from file ?
    (*a*) readline( ) (*b*) readlines( )
    (*c*) readstatement( ) (*d*) readfullline( )
11. Which function is used to write all the characters ?
    (*a*) write( ) (*b*) writecharacters( )
    (*c*) writeall( ) (*d*) writechar( )
12. Which function is used to write a list of strings in a file ?
    (*a*) writeline( ) (*b*) writelines( )
    (*c*) writestatement( ) (*d*) writefullline( )

13. Which of the following represents mode of both writing and reading in binary format in file.?
    (*a*) wb+ (*b*) w (*c*) wb (*d*) w+
14. Which of the following is not a valid mode to open a file ?
    (*a*) ab (*b*) rw (*c*) r+ (*d*) w+
15. What is the difference between r+ and w+ modes ?
    (*a*) No difference.
    (*b*) In r+ mode, the pointer is initially placed at the beginning of the file and for w+, the pointer is placed at the end.
    (*c*) In w+ mode, the pointer is initially placed at the beginning of the file and for r+, the pointer is placed at the end.
    (*d*) Depends on the operating system.
16. Which of the following command is used to open a file "c:\pat.txt" in read-mode only ?
    (*a*) fin = open("c:\pat.txt", "r") (*b*) fin = open("c:\\pat.txt", "r")
    (*c*) fin = open(file = "c:\pat.txt", "r+") (*d*) fin = open(file = "c:\\pat.txt", "r+")
17. Which of the following statements are true regarding the opening modes of a file ?
    (*a*) When you open a file for reading, if the file does not exist, an error occurs.
    (*b*) When you open a file for writing, if the file does not exist, an error occurs.
    (*c*) When you open a file for reading, if the file does not exist, the program will open an empty file.
    (*d*) When you open a file for writing, if the file does not exist, a new file is created.
    (*e*) When you open a file for writing, if the file exists, the existing file is overwritten with the new file.
18. Which of the following command is used to open a file "c:\pat.txt" for writing in binary format only ?
    (*a*) fout = open("c:\pat.txt", "w") (*b*) fout = open("c:\\pat.txt", "wb")
    (*c*) fout = open("c:\pat.txt", "w+") (*d*) fout = open("c:\\pat.txt", "wb+")
19. Which of the following command is used to open a file "c:\pat.txt" for writing as well reading in binary format only ?
    (*a*) fout = open("c:\pat.txt", "w") (*b*) fout = open("c:\\pat.txt", "wb")
    (*c*) fout = open("c:\pat.txt", "w+") (*d*) fout = open("c:\\pat.txt", "wb+")
20. Which of the following functions do you use to write data in the binary format ?
    (*a*) write( ) (*b*) output( ) (*c*) dump( ) (*d*) send( )

Fill in the Blanks

1. The default file-open mode is _____ mode.
2. A _____ governs the type of operations (*e.g.*, read/write/append) possible in the opened file.
3. The two types of data files can be _____ files and _____ files.
4. The other name for file object is _____ .
5. The _____ file mode will open a file for read and write purpose.
6. The _____ file mode will open a file for write and read purpose.
7. To close an open file, _____ method is used.
8. To read all the file contents in the form of a list, _____ method is used.
9. To write a list in a file, _____ method may be used.

10. To force Python to write the contents of file buffer on to storage file, _____ method may be used.
11. To read and write into binary files, _____ module of Python is used.
12. The _____ method of pickle module writes data into a binary file.
13. The _____ method of pickle module reads data from a binary file.
14. The conversion of an object hierarchy in byte stream is called _____ or _____.
15. The character that separates the values in csv files is called the _____.
16. The default delimiter of csv files is _____ .
17. The csv files are actually _____ files .
18. We can suppress EOL translation is text file by giving _____ argument in open( ).
19. The file mode to open a binary file for reading as well writing is _____ .
20. The file mode to open a text file for reading as well writing is _____ .
21. The file mode to open a text file for writing as well reading is _____ .
22. The file mode to open a binary file for writing as well reading is _____ .
23. The file mode to open a csv file for reading as well writing is _____ .
24. The file mode to open a csv file for appending as well reading is _____ .
25. To specify a different delimiter while writing into a csv file, _____ argument is used w csv.writer( ).

True/False Questions

1. When you open a file for reading, if the file does not exist, an error occurs.
2. When you open a file for writing, if the file does not exist, an error occurs.
3. When you open a file for writing, if the file exists, the existing file is overwritten with the new fi
4. The absolute paths are from the topmost level of the directory structure.
5. The relative paths are relative to the current working directory.
6. The relative path for a file always remains the same even after changing the directory.
7. The types of operations that can be carried out on a file depend upon the file mode a file is opened
8. If no path is given with a file name in the file open( ), then the file must exist in the current direc
9. Functions readline( ) and readlines( ) are essentially the same.
10. Python automatically flushes the file buffers before closing a file with close( ) function.
11. When you open a file for writing, if the file does not exist, a new file is created.
12. When you open a file for appending, if the file exists, the existing file is overwritten with the new f
13. Conversion of an object hierarchy in byte stream is called Serialisation.
14. Serialisation process is also called pickling.
15. The load( ) function of the pickle module performs pickling.
16. The dump( ) function of the pickle module performs unpickling.
17. The csv files can only take comma as delimiter.
18. The csv files are text files.

NOTE : Answers for OTQs are given at the end of the book.

Solved Problems

1. *What is the difference between read( ) and readlines( ) function?*

Solution. The read( ) reads from a file in read mode, and stores its contents in a string type variable. The readlines( ) function, reads from a file in read mode and returns a list of all lines in the file.

2. *Differentiate between the following :* [CBSE D 2015]

(*i*) `f = open('diary.txt', 'r')`

(*ii*) `f = open('diary.txt', 'w')`

Solution. (*i*) File has been opened in *read mode* with file handle *f*.

(*ii*) File has been opened in *write mode* with file handle *f*.

3. *What is the difference between readline( ) and readlines( ) function?*

Solution. The ***readline( )*** function reads from a file in read mode and returns the next line in the file or a blank string if there are no more lines. (The returned data is of string type)

The ***readlines( )*** function, also reads from a file in read mode and returns a list of all lines in the file. (The returned data is of list type)

4. *Write a single loop to display all the contents of a text file poem.txt after removing leading and trailing whitespaces.*

Solution.

```
for line in file("poem.txt") :
    print(line.strip())
```

5. *Write a function stats( ) that accepts a filename and reports the file's longest line.*

Solution.

```
def stats(filename) :
    longest = " "
    for line in file(filename) :
        if len(line) > len(longest) :
            longest = line
    print("Longest line's length =", len(longest))
    print(longest)
```

6. *What is the output of following code fragment ? Explain.*

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

Solution. The output will be :

```
'Hello, world!\nHow are you?'
```

The first line of the code is opening the file in write mode ; the next two lines write text to the file. The last line opens the file and from that reference reads the file-content. Function file( ) does the same as that of open( ). Thus file("output.txt") will give the reference to open file, on which read( ) is applied.

7. *Write a function remove_lowercase( ) that accepts two filenames, and copies all lines that do not start with a lowercase letter from the first file into the second.*

Solution.

```
def remove_lowercase(infile, outfile) :
    output = file(outfile, "w")
    for line in file(infile) :
        if not line[0] in "abcdefghijklmnopqrstuvwxyz" :
            output.write(line)
    output.close()
```

8. *What is the output of following code ?*

```
file("e:\\poem.txt", "r").readline().split()
```

Recall that poem.txt has some leading and trailing whitespaces.

Solution. ['WHY?']

9. *What is the output of following code ?*

```
file("e:\\poem.txt", "r").readline()
```

Solution.

```
'    WHY?\n'
```

10. *What is the output of following code ?*

```
fh = file("poem.txt", "r")
size = len(fh.read())
print(fh.read(5))
```

Solution. 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 so **print( )** statement prints nothing.

11. *Write a program to display all the records in a file along with line/record number.*

Solution.

```
fh = open("Result.det", "r")
count = 0
rec = " "
while True :
    rec = fh.readline()
    if rec == " " :
        break
    count = count + 1
print(count, rec, end = '')          # to suppress extra newline by print
fh.close()
```

12. *What is the output produced by following code ?*

```
obj = open("New.txt","w")
obj.write("A poem by Paramhansa Yogananda) "
obj.write("Better than Heaven or Arcadia")
```



```
obj.write("I love thee, O my India!")
obj.write("And thy love I shall give")
obj.write("To every brother nation that lives.")
obj.close()
obj1 = open("New.txt","r")
s1 = obj1.read(48)
print(s1)
obj1.close()
```

Solution. The output produced by above code will be :

```
A poem by Paramhansa Yogananda Better than Heaven
```

13. *The file "New.txt" contains the following :*

```
Better than Heaven or Arcadia
I love thee, O my India!
And thy love I shall give
To every brother nation that lives.
```

Considering the given file, what output will be produced by the following code ?

```
obj1=open("New.txt","r")
s1 = obj1.readline()
s2 = obj1.readline()
s3 = obj1.readline()
s4 = obj1.read(15)
print(s4)
obj1.close()
```

Solution. The output produced by above code is :

```
And thy love I
```

14. *Two identical files (p1.txt and p2.txt) were created by following two codes (carefully go through the two codes given below).*

(a)
```
obj = open("p1.txt","w")
obj.write("Better than Heaven or Arcadia")
obj.write("I love thee, O my India!")
obj.write("And thy love I shall give")
obj.write("To every brother nation that lives.")
obj.close( )
```

(b)
```
obj = open("p2.txt","w")
obj.write("Better than Heaven or Arcadia\n")
obj.write("I love thee, O my India!\n")
obj.write("And thy love I shall give\n")
obj.write("To every brother nation that lives.\n")
obj.close( )
```

What would be the output produced if the files are read and printed with following code.

Solution. The output produced by code (a) will be :

```
Better than Heaven or ArcadiaI love thee, O my India!And thy love
I shall giveTo every brother nation that lives.
```

The output produced by code (*b*) will be :

```
A poem by Paramhansa Yogananda
Better than Heaven or Arcadia
I love thee, O my India!
And thy love I shall give
To every brother nation that lives.
```

15. *Considering the two files p1.txt and p2.txt created in previous question, what output will be produced by following code fragments ?*

(*a*)
```
obj1 = open("p1.txt","r")
s1 = obj1.readline()
s2 = obj1.read(15)
print(s2)
obj1.close()
```

(*b*)
```
obj1 = open("p2.txt","r")
s1 = obj1.readline()
s2 = obj1.read(15)
print(s2)
obj1.close()
```

Solution. No output or blank output will be produced by code (*a*).

For code(*b*), the output produced will be :

```
Better than Hea
```

16. *Consider the file p2.txt created above. Now predict the output of following code that works with p2.txt. Explain the reason behind this output.*

```
fp1 = open("p2.txt", "r")
print(fp1.readline(20))
s1 = fp1.readline(30)
print(s1)
print(fp1.readline(25))
```

Solution. The output produced by above code will be :

```
A poem by Paramhansa
 Yogananda
Better than Heaven or Arc
```

The reason behind this output is that the first file-read line (*i.e.*, **fp1.readline(20)**, read 20 bytes from the file pointer. As just after opening the file, the file-pointer is at the beginning of the file, the 20 bytes are read from the beginning of the file which returned string as "A poem by Paramhansa\n" – this is because *readline( )* returns the read string by adding an end-line character to it (\n). So the first line of output was printed as :

```
A poem by Paramhansa
```

After the first *readline( )*, the file pointer was at the space following word 'Paramhansa', so the next *readline( )* started reading from there and read 15 character or end-of the-line, whichever is earlier. So the read string was "**Yogananda\n**" – notice the space before word Yogananda. Hence came the second line of the output.

Now the file-pointer was at the beginning of the third line and the next readline (of **fp1.readline(25)**) read 25 characters from this line and gave the last line of output.

17. *A text file contains alphanumeric text (say an.txt). Write a program that reads this text file and prints only the numbers or digits from the file.*

Solution.
```
F = open("an.txt", "r")
for line in F:
    words = line.split()
    for i in words:
        for letter in i:
            if(letter.isdigit()):
                print(letter)
```

18. *Read the code given below and answer the question :*

```
fh = open("main.txt", "w")
fh.write("Bye")
fh.close()
```

If the file contains "GOOD" before execution, what will be the contents of the file after execution of this code ?

Solution. The file would now contain **"Bye"** only, because when an existing file is opened in write mode ("w"), it truncates the existing data the file.

19. *Write a function in Python to count the number of lines in a text file 'STORY.TXT' which is starting with an alphabet 'A'.* [CBSE Sample Paper 2019-20]

Solution.
```
def COUNTLINES():
    file = open('STORY.TXT','r')
    lines = file.readlines()
    count = 0
    for w in lines:
        if w[0] == "A" or w[0] == "a":
            count = count + 1
    print("Total lines started with 'A' or 'a'", count)
    file.close()
```

20. *A given text file "data.txt" contains :*

```
Line 1\n
\n
Line 3
Line 4
\n
Line 6
```

What would be the output of following code?

```
fh = open ("data.txt", "r")
lst = fh.readlines()
print(lst[0], end = '')
print(lst[2], end = '')
print(lst[5], end = '')
print(lst[1], end = '')
print(lst[4], end = '')
print(lst[3])
```

Solution.

```
Line 1\n
 Line 3
 Line 6  \n
 \n
 Line 4
```

21. *Write code to print just the last line of a text file "data.txt".*

Solution.

```
fin = open("data.txt", "r")
lineList = fin.readlines()
print("Last line =", lineList[-1])
```

22. *Write a program that copies a text file "source.txt" onto "target.txt" barring the lines starting with a "@" sign.*

Solution.

```
def filter(oldfile, newfile) :
    fin = open(oldfile, "r")
    fout = open(newfile, "w")
    while True :
        text = fin.readline()
        if len(text) == 0:
            break
        if text[0] == "@":
            continue
        fout.write(text)
    fin.close()
    fout.close()

filter("source.txt", "target.txt")          # calling the function
```

23. *Consider the following statement. In which file mode is the file opened ? Justify your answer.*

```
with open('poem.txt') as f:
```

Solution. **Read mode**, because the file mode has not been specified and the default file mode is read mode.

24. *What is pickling process ? What is its need ?*

Solution. Objects have a specific structure which must be maintained while storing or access them. Python provides a special module – the pickle module to take care of that. The pick process serializes the objects and converts them into byte stream so that they can be stored in files.

25. *What is unpickling ?*

Solution. The "**unpickling**" is the inverse operation of pickling, whereby a byte stream (other from a binary file) is converted back into an object hierarchy.

26. *What is the difference between a regular text file and a delimited text file ?*

Solution. *Regular Text files* are the files which store the text in the same form as typed (ASC Unicode). Here the newline character ends a line and the text translations take place. These files a file extension as **.txt.**

Delimited Text files are also text files where a specific character is stored to separate the values, *i.e.*, after each value, *e.g.*, a tab or a comma after every value, *e.g.*, TSV files (*Tab Separated Values files*) or CSV files (*Comma Separated Values files*).

27. *What are csv files ?*

Solution. CSV files are *Comma Separated Values files*. These are the delimited files when mostly **comma** (**delimiter** character which separates the values) is used to separate the values stored. The CSV files can also a delimiter other than comma but comma is the default delimiter. These files take the extension as .csv.

28. *What is the significance of newline = '' argument in file open() function ?*

Solution. When **newline = ''** ensures that no end of line (EOL) translation would take place while storing the file contents on the disk. By default text files are stored with some text translations, EOL translation is one of these. By specifying **newline = ''** as argument to file open( ) function, we ensure that no EOL translation takes place. This is useful when we use the files on different platforms.

29. *What does csv.writer object do ?*

Solution. The csv.writer object adds delimitation to the user data prior to storing data in the csv file on storage disk.

30. *What is the function of csv.reader object ?*

Solution. The csv.reader object reads data from a csv file on storage disk, parses it, *i.e.*, removes the delimitation and loads the parsed data into a Python iterator wherefrom we can fetch the data row by row.

31. *What will be stored in the file school.csv after the following code is executed?*

```
data = [  [ 'DRS Delhi', 'Esha', 'Badminton'],
          ['BTS Patna', 'Abhi', 'Tennis'] ]
import csv
with open('schoo;.csv', 'w', newline ='') as csvfile:
    writer = csv.writer(csvfile, delimiter = ',')
    writer.writerow(['School', 'Nickname', 'Sport'])
    writer.writerows(data)
```

Solution.

```
'School', 'Nickname', 'Sport'
'DRS Delhi', 'Esha', 'Badminton'
'BTS Patna', 'Abhi', 'Tennis'
```

32. *A file Answer.txt contains the text as shown in Fig. 5.5 in the chapter. Write a program to remove all the lines that contain the character 'a' in this file and write other lines into another file.*

Solution.

```
myfile = open("Answer.txt", "r")
newfile = open("nswer.txt", "w")
line = " "                          # initially stored a space (a non-None value)
while line :
    line = myfile.readline()  # one line read from file
    if 'a' not in line:
        newfile.write(line)
```

```
myfile.close()
newfile.close()
print("Newly created file contains")
print("————————————————")
newfile = open("nswer.txt", "r")
line = ""
while line :
    line = newfile.readline()          # one line read from file
    print(line)
newfile.close()
```

The output produced by above program is

```
Newly created file contains
---------------------------
We know this will never occur.
How mediocre our world would be without this single most powerful letter.
```

33. *Differentiate between the following :* (Delhi 2019)

(i) `f = open ('diary. txt', 'r' )` *(ii)* `f = open ('diary. txt', 'w')`

Solution.

(i) The text file diary.txt is opened in read mode.

(ii) The text file diary.txt is opened in write mode.

34. *What will be displayed by the following code ?*

```
import pickle
Names = ['First', 'second', 'third', 'fourth', 'Fifth']
lst = []
for i in range(-1, -5, -1) :
    lst.append(Names[i])
with open('test.dat', 'wb') as fout:
    pickle.dump(lst, fout)
with open('test.dat', 'rb') as fin:
    nlist = pickle.load(fin)
print(nlist)
```

Solution.

```
['Fifth', 'fourth', 'third', 'second']
```

35. *Following code is written to update a record in a file opened with following code :*

```
import pickle
fin = open('Stu.dat', 'rb+')
try:
    while True :
        _____ = fin.tell()   # Line 1 : store file-pointer position before reading the record
        stu = pickle.load(fin)
```



```
        if stu['Marks'] in [92, 93, 94] :
            stu['Marks'] += 3          # changes made in the record
            fin.____ (_____) # Line 2 :place the file-pointer at the exact location of the record
            pickle.dump(stu, fin) # now write the updated record on the exact location
except :
    :
```

Fill in the blanks in Lines 1 and 2 to complete the code.

Solution.

Line 1 : **position = fin.tell( )** # can use any identifier in place of position

Line 2: **fin.seek (position)**

36. *Identify the error in the following code:*

```
1.  import pickle
2.  data = ['one', 2, [3, 4, 5]]
3.  with open('data2.dat', 'rb') as f:
4.      pickle.dump(data, f)
```

Solution. The file is opened in read mode and dump() function tries to write onto file, hence the error.

So the line 3 should be changed to L

```
with open('data2.dat', 'wb') as f:
```

37. *Identify the error in the following code.*

```
1.  f = open('/tmp/workfile', 'r+')
2.  f.write('0123456789abcdef')
3.  f.write('xyz8466')
4.  f.close()
5.  f.seek(0)
6.  f.read()
```

Solution. The lines 5 and 6 will raise the error as no operation can take place in a closed file – file is closed at line4. Since the file is opened in read as well write mode, we just need to bring line 4 at the end of the code.

38. *Identify the error in the following code ?*

```
1.  import csv
2.  line =  [ [1, 2,3], [4, 5, 6] ]
3.  With open(path, "w", newline ='') as csv_file :
4.      writer = csv.writer(csv_file, delimiter = |')
5.      for line in data:
6.          writer.writerow (line)
```

Solution. The give code is trying to write a nested list with *writerow( )*.

It should be replaced with *writerows( ), i.e.,* line 6 should be :

```
writer.writerows(line)
```

39. *Write a program to read a text file and display the count of lowercase and uppercase letters in the file.*
(Note : use text file Answer.txt given in Fig. 5.5 in the chapter)

Solution.

```
myfile = open("Answer.txt", "r")
ch = " "              # initially stored a space ( a non-None value)
lcount = 0                    #variable to store count of lowercase letters
```

```
ucount = 0
while ch :                          # while ch stores a Non-None value
    ch = myfile.read(1)             # one character read from file
    if ch.isupper() == True:
        ucount = ucount + 1
    else :
        lcount = lcount + 1
print("Uppercase letters in the file :", ucount)
print("Lowercase letters in the file :", lcount)
# close the file
myfile.close()
```

40. *Your recipe uses some ingredients. Write a program to store the list of ingredients in a binary file.*

Solution.

```
import pickle
ingredients = ['cucumber', 'pumpkin', 'carrot', 'peas']
with open('recipe.dat', 'wb') as fout:
    pickle.dump(ingredients, fout)
```

41. *Write a method in python to read the content from a text file story.txt line by line and display the same on screen.*

Solution. (Outside Delhi 2

```
def read_file () :
    inFile = open('story.txt','r')
    for line in inFile:
        print(line)
```

42. *Write a method in python to read lines from a text file INDIA.TXT, to find and display the occurrence of the word "India".*

(Delhi 2

For example: If the content of the file is

```
"India is the fastest growing economy.
India is looking for more investments around the globe.
The whole world is looking at India as a great market.
Most of the Indians can foresee the heights that India is
capable of reaching."
```

The output should be 4

Solution.

```
def display1( ):
    count = 0
    file = open('INDIA.TXT','r')
    for LINE in file:
        Words = LINE.split()
        for W in Words:
            if W == "India":
                count = count +1
    print ( count )
file.close()
```


43. *Write a method in python to read lines from a text file MYNOTES.TXT, and display those lines, which are starting with an alphabet 'K'.* (Outside Delhi 2017)

Solution.

```
def display():
    file = open('MYNOTES.TXT','r')
    line = file.readline()
    while line:
        if line[0]=='K' :
            print ( line )
        line = file.readline()
    file.close()
```

44. *Considering the following definition of dictionary MULTIPLEX, write a method in python to search and display all the content in a pickled file CINEMA.DAT, where MTYPE key of the dictionary is matching with the value 'Comedy'.* (similar to Delhi 2018)

```
MULTIPLEX = {'MNO':____, 'MNAME':_____, 'MTYPE':_____}
```

Solution.

```
def Search():
    file = open('CINEMA.DAT','rb')
    try:
        while True:
            MULTIPLEX = pickle.load(file)
            if MULTIPLEX ['MTYPE']=="Comedy":
                print(MULTIPLEX)
    except EOFError:
        file.close()
```

45. *Write a program to increase the salary by Rs. 2000/- of the employee having empno as 1251 in the file empl.dat.*

Solution.

```
import pickle

# declare empty dictionary object to hold read records
emp = {}
found = False

# open binary file in read and write mode
fin = open('empl.dat', 'rb+')
fin.seek(0)
# read from the file
try:
    while True :
        rpos = fin.tell()      # store file-pointer position before reading the record
        emp = pickle.load(fin)  # read record in emp dictionary from fin file handle
        if emp['Empno'] == 1251 :
            emp['Salary'] = emp['Salary'] + 2000     # changes made in the record
            fin.seek(rpos)  #place the file-pointer at the exact location of the record
            pickle.dump(emp, fin) # write the modified record
            print(emp)
            found = True
```

```
        except EOFError:
            if found == False:
                print("Sorry, no matching record found.")
            else:
                print("Record(s) successfully updated.")
        fin.close()                # close file
```

46. *Take a sample text file and find the most commonly occurring word. Also, list the frequencies of words in a text file.*

Solution

```
with open ("nanthem.txt", "r") as fh :
    contents = fh.read()
wordlist = contents.split()
wordfreq = []
high = 0
word = ''
existing =[]
for w in wordlist:
    wcount = wordlist.count(w)
    if w not in existing:
        wordfreq.append( [w,wcount])
        existing.append(w)
    if wcount > high:
        high = wcount
        word = w
print("The word '"+ word+"' occurs maximum number of times,", high, "times. ")
print("\nOther words have these frequencies :")
print(wordfreq)
```

For the text file *nanthem.txt* shown below, the above program produced the following output:

```
Jan gan man adhinayaka jaya he
Bharat bhagya vidhata
Panjab Sindh Gujarat Maratha
Dravid Utkal Banga
Vindhye Himachal Yamuna Ganga
Uchchal jaladhi tarang
Tava shubh name jage,
Tava shubh ashish mange,
Gahe tava jaya gatha.
Jan gan mangal dayak jaya he
```

Output :

```
The word 'Jaya' occurs maximum number of times, 7 times. Other words have these frequencies :
[['Jan', 2], ['gan', 2], ['man', 1], ['adhinayaka', 1], ['jaya', 3], ['he', 6],
['Bharat', 2], ['bhagya', 2], ['vidhata', 2], ['Panjab', 1], ['Sindh', 1], ['Gujarat', 1],
['Maratha', 1], ['Dravid', 1], ['Utkal', 1], ['Banga', 1], ['Vindhye', 1],['Himachal', 1],
['Yamuna', 1], ['Ganga', 1], ['Uchchal', 1], ['jaladhi', 1], ['tarang', 1], ['Tava', 2],
['shubh', 2], ['name', 1], ['jage,', 1], ['ashish', 1], ['mange,', 1], ['Gahe', 1],
['tava', 1], ['gatha.', 1], ['mangal', 1], ['dayak', 1], ['Jaya', 7]]
```

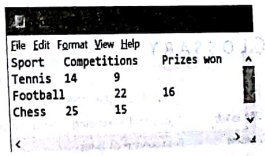

47. *Write a program to read following details of sports' performance (sport, competitions, prizes-won) of your school and store into a csv file delimited with tab character.*

Solution.

```
import csv
fh = open("Sport.csv", "w")
swriter = csv.writer(fh, delimiter='\t')
swriter.writerow( ['Sport', 'Competitions', 'Prizes won'])   #write header row
ans = 'y'
i = 1
while ans == 'y':
    print("Record", i )
    sport= input("Sport name:")
    comp = int(input("No. of competitions participated:"))
    prizes = int(input("Prizes won:"))
    srec = [ sport, comp, prizes ] # create sequence of user data
    swriter.writerow(srec)
    i = i+1
    ans = input("Want to enter more records? (y/n)..")
fh.close()
```

Sample run :

```
Record 1
Sport name:Tennis
No. of competitions participated:14
Prizes won:9
Want to enter more records? (y/n)..y
Record 2
Sport name:Football
No. of competitions participated:22
Prizes won:16
Want to enter more records? (y/n)..y
Record 3
Sport name:Chess
No. of competitions participated:25
Prizes won:15
Want to enter more records? (y/n)..n
```



48. *Write a program to get item details (code, description and price) for multiple items from the user and create a csv file by writing all the item details in one go.*

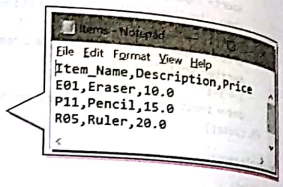
Solution.

```
import csv
fh = open("Items.csv", "w")
iwriter = csv.writer(fh)
ans = 'y'
itemrec= [['Item_Name', 'Description', 'Price']]
print("Enter item details")
While ans =='y' :
    iname = input("Enter Item code :")
```

```
    desc = input("Enter description :")
    price = float(input("Enter price:"))
    itemrec.append( [iname, desc, price])
    ans = input("Want to enter more items? (y/n)...")
else:
    iwriter.writerows(itemrec)
    print("Records written successfully.")
fh.close()
```

Sample run of the program :

```
Enter item details
Enter Item code :E01
Enter description :Eraser
Enter price:10
Want to enter more items? (y/n)...y
Enter Item code :P11
Enter description :Pencil
Enter price:15
Want to enter more items? (y/n)...y
Enter Item code :R05
Enter description :Ruler
Enter price:20
Want to enter more items? (y/n)...n
Records written successfully.
```



GLOSSARY

File	A bunch of bytes stored on some storage device.
File mode	A constant describing how a file is to be used.
Stream	A sequence of bytes.

Assignment

For Selective Assignment Solutions

Scan QR Code

Type A : Short Answer Questions/Conceptual Questions

1. What is the difference between "w" and "a" modes ?
2. What is the significance of a file-object ?
3. How is file open( ) function different from close( ) function ?
4. Write statements to open a binary file C:\Myfiles\Text1.txt in read and write mode by specifying the file path in two different formats.
5. When a file is opened for output in write mode, what happens when
   (*i*) the mentioned file does not exist (*ii*) the mentioned file does exist ?
6. What role is played by file modes in file operations ? Describe the various file mode constants and their meanings.

7. What are the advantages of saving data in : (*i*) binary form (*ii*) text form (*iii*) csv files ?
8. When do you think text files should be preferred over binary files ?
9. Write a statement in Python to perform the following operations : [CBSE D 16]
   (*a*) To open a text file "BOOK.TXT" in read mode
   (*b*) To open a text file "BOOK.TXT" in write mode
10. When a file is opened for output in append mode, what happens when
    (*i*) the mentioned file does not exist
    (*ii*) the mentioned file does exist.
11. What role is played by file modes in file operations ? Describe the various file mode constants and their meanings.
12. How many file objects would you need to create to manage the following situations ? Explain.
    (*i*) to process three files sequentially
    (*ii*) to merge two sorted files into a third file.
12. Is csv file different from a text file ? Why/why not ?
13. Is csv file different from a binary file ? Why/why not ?
14. Why are csv files popular for data storage ?
15. How do you change the delimiter of a csv file while writing into it ?
16. When and why should you suppress the EOL translation in csv file handling ?
17. If you rename a text file's extension as .csv, will it become a csv file ? Why/why not ?

Type B : Application Based Questions

1. How are following codes different from one another ?

   (*a*)
   ```
   my_file = open('poem.txt','r')
   my_file.read()
   ```

   (*b*)
   ```
   my_file = open('poem.txt','r')
   my_file.read(100)
   ```

2. If the file 'poemBTH.txt' contains the following poem (by Paramhans Yoganand) :

   God made the Earth;
   Man made confining countries
   And their fancy-frozen boundaries.
   But with unfound boundless love
   I behold the borderland of my India
   Expanding into the World.
   Hail, mother of religions, lotus, scenic beauty,and sages!

   Then what outputs will be produced by both the code fragments given in question1.
3. Consider the file poemBTH.txt given above (in previous question). What output will be produced by following code fragment ?

   ```
   obj1 = open("poemBTH.txt","r")
   s1 = obj1.readline()
   s2.readline(10)
   s3 = obj1.read(15)
   print(s3 )
   print(obj1.readline())
   obj1.close()
   ```

4. Write code to open file created in the previous question and print it in following form :

   `Name : <name>  Phone :<phone number>`

5. Consider the file "poemBTH.txt" and predict the outputs of following code fragments if the file has been opened in filepointer *file1* with the following code :

```
file1 = open("E:\\mydata\\poemBTH.txt","r+")
```

(a)
```
print("A. Output 1")
print(file1.read())
print()
```

(b)
```
print("B. Output 2")
print(file1.readline() )
print()
```

(c)
```
print("C. Output 3")
print(file1.read(9))
print()
```

(d)
```
print("D. Output 4")
print(file1.readline(9) )
```

(e)
```
print("E. Output of Readlines function is")
print(file1.readlines() )
print()
```

NOTE. Consider the code fragments in succession, *i.e.*, code (*b*) follows code (*a*), which means changes by code (*a*) remain intact when code (*b*) is executing. Similarly, code (*c*) follows (*a*) and (*b*), and so on.

6. What is the following code doing ?

```
file = open("contacts.csv", "a")
name = input("Please enter name.")
phno = input("Please enter phone number.")
file.write(name + "," + phno + "\n")
```

7. Consider the file "contacts.csv" created in above Q. and figure out what the following code is trying to do?

```
name = input("Enter name :")
file = open("contacts.csv", "r")
for line in file:
    if name in line:
        print(line)
```

8. Consider the file poemBTH.txt and predict the output of following code fragment. What exactly is the following code fragment doing ?

```
f = open("poemBTH.txt", "r")
nl = 0
for line in f:
    nl += 1
print(nl)
```

9. If you use the code of Q.8 with p1.txt (created in solved problem 14), what would be its output ?
10. Write a method in Python to read the content from a text file *diary.txt* line by line and display the same on screen. [CBSE D 2015]
11. Write a method in Python to write multiple line of text contents into a text file *mylife.txt.line*. [D 2016]
12. What will be the output of the following code ?

```
import pickle
ID = {1:"Ziva",2:"53050",3:"IT",4:"38",5:"Dunzo"}
fin = open("Emp.pkl","wb")
pickle.dump(EmpID, fin)
fin.close()
fout = open("Emp.pkl",'rb')
ID = pickle.load(fout)
print( ID [5] )
```


13. What will be the output of the following code ?

```
import pickle
List1 = ['Roza', {'a': 23, 'b': True}, (1, 2, 3), [['dogs', 'cats'], None]]
List2 = ['Rita', {'x': 45, 'y': False}, (9, 5, 3), [['insects', 'bees'], None]]
with open('data.pkl', 'wb') as f:
    f.write(list1)
with open('data.pkl', 'wb') as f:
    f.write(list2)
with open('data.pkl', 'rb') as f:
    list1 = pickle.load(f)
print(list1)
```

14. What is the output of the following considering the file *data.csv* given on the right.

```
import csv
with open('C:\data.csv','r+') as f:
  data = csv.reader(f)
  for row in data:
      if 'the' in row :
          print(row)
```

File data.csv contains:

```
Identifier;First name;Last name
901242;Riya;Verma
2070741;Laura;Grey
408129;Ali;Baig
934600;Manit;Kaur
507916;Jiva;Jain
```

15. Identify the error in the following code.

```
import csv
f = open('attendees1.csv')
csv_f = csv.writer(f)
```

16. Identify the error in the following code.

```
import csv
f = open('attendees1.csv')
csv_f = csv.reader()
for row in csv_f:
    print(row)
```

17. Identify the error in the following code.

```
import pickle
data = ['one', 2, [3, 4, 5]]
with open('data.dat', 'wb' :
    pickle.dump(data)
```

Type C : Programming Practice/Knowledge based Questions

1. Write a program that reads a text file and creates another file that is identical except that every sequence of consecutive blank spaces is replaced by a single space.
2. A file *sports.dat* contains information in following format : Event ~ Participant
   Write a function that would read contents from file *sports.dat* and creates a file named *Athletic.dat* copying only those records from *sports.dat* where the event name is "*Atheletics*".
3. A file contains a list of telephone numbers in the following form :

   Arvind 7258031
   Sachin 7259197

   The names contain only one word the names and telephone numbers are separated by white spaces. Write program to read a file and display its contents in two columns.
4. Write a program to count the words "to" and "the" present in a text file "Poem.txt".

5. Write a program to count the number of upper-case alphabets present in a text file "Article.txt".
6. Write a program that copies one file to another. Have the program read the file names from user?
7. Write a program that appends the contents of one file to another. Have the program take the filenames from the user.
8. Write a method/function **DISPLAYWORDS( )** in python to read lines from a text file **STORY.TXT**, and display those words, which are less than 4 characters. [CBSE Sample Paper 2019]
9. Write a program that reads characters from the keyboard one by one. All lower case characters get stored inside the file LOWER, all upper case characters get stored inside the file UPPER and all other characters get stored inside file OTHERS.
10. Write a function in Python to count and display the number of lines starting with alphabet 'A' present in a text file " LINES.TXT". *e.g.*, the file "LINES.TXT" contains the following lines :

```
A boy is playing there.
There is a playground.
An aeroplane is in the sky.
Alphabets & numbers are allowed in password.
```

the function should display the output as 3.

11. Write a program that counts the number of characters up to the first $ in a text file.
12. Write a program that will create an object called filout for writing, associate it with the filename *STRS.txt*. The code should keep on writing strings to it as long as the user wants.
13. Write a program that reads a text file and creates another file that is identical except that every sequence of consecutive blank spaces is replaced by a single space.
14. Write a function to count the words "to" and "the" present in a text file "POEM.TXT".
15. Consider the following definition of dictionary Member, write a method in python to write the content in a pickled file member.dat. (similar to Delhi 2015)

```
Member = {'MemberNo.': _____, 'Name':_____ }
```

16. Consider the following definition of dictionary Staff, write a method in python to search and display the content in a pickled file staff.dat, where Staffcode key of the dictionary is matching with 'S0105'.

```
Staff ={'Staffcode': _____, 'Name'= _____ }
```
(similar to Delhi 2016)

17. Considering the following definition of dictionary COMPANY, write a method in Python to search and display the content in a pickled file COMPANY.DAT, where CompID key of the dictionary is matching with the value '1005'. (similar to Delhi 2017)

```
Company = {'CompID' = _____, 'CName' = _____, 'Turnover' = ____}
```

18. Write a function in to search and display details of all trains, whose destination is "Delhi" from a binary file "TRAIN.DAT". Assuming the binary file is containing the objects of the following dictionary type:

```
Train = { 'Tno' : _____, 'From': _____ , 'To' : _____ }
```

19. Write a Python program to read a given CSV file having tab delimiter.
20. Write a Python program to write a nested Python list to a csv file in one go. After writing the CSV file read the CSV file and display the content.
21. Write a function that reads a csv file and creates another csv file with the same content, but with a different delimiter.
22. Write a function that reads a csv file and creates another csv file with the same content except the lines beginning with 'check'.

CHAPTER 5 : FILE HANDLING

Multiple Choice Questions

1. (c) 2. (b), (d) 3. (a), (c) 4. (c) 5. (d) 6. (b)
7. (d) 8. (a) 9. (a) 10. (a) 11. (a) 12. (b)
13. (a) 14. (b) 15. (b) 16. (b) 17. (a), (d), (e)
18. (b) 19. (d) 20. (c)

Fill in the Blanks

1. read 2. file mode 3. text, binary 4. file handle 5. r+ 6. w+ or a+
7. close( ) 8. readlines( ) 9. writelines( ) 10. flush( )
11. pickle 12. dump( ) 13. load( ) 14. pickling, serialisation
15. delimiter 16. comma 17. text 18. newline 19. rb+ 20. r+
21. w+ 22. wb+ 23. r+ 24. a+ 25. delimiter

True/False Questions

1. T 2. F 3. T 4. T 5. T 6. F
7. T 8. T 9. F 10. T
11. T 12. F 13. T 14. T 15. F 16. F
17. F 18. T