TEXT FILE HANDLING

Key points:

Data File- A file is a sequence of bytes on the disk/permanent storage where a group of related data is stored. File handling in Python enables us to create, update, read, and delete the files stored on the file system through our python program.

Data File handling takes place in the following order.

1- Opening a file.

2- Performing operations (read, write) or processing data.

3- Closing the file.

Types of files in Python:

Python allows us to create and manage three types of data files.

1- Text file

2- Binary file

3- CSV file

Text file: A text file is simply a sequence of ASCII or Unicode characters.A line is a sequence of characters, stored on permanent storage. In a text file, each line is terminated by a special character, known as End Of Line (EOL). Text file can be created using any text editor. Ex. Myfile.txt.

Binary file: A binary file stores the data in the same way as stored in the memory. The .exe files, mp3 file, image files, word documents are some of the examples of binary files. We can't read a binary file using a text editor.

CSV file: CSV (Comma Separated Values) is a file format for data storage which looks like a text file. The information is organized with one record on each line and each field is separated by comma

Aspect	Text File	Binary File	CSV File (Comma-Separated Values)
Format	Contains plain text	Contains binary data	Stores tabular data in plain text
Content	Human-readable	Not human-readable	Human-readable
Character Encoding	ASCII, UTF-8	Not applicable	ASCII, UTF-8
Structure	Data is stored as lines of text	Data is stored as sequences of binary bytes	Data is organized into rows and columns
Usage	Suitable for storing textual data	Suitable for storing non-textual data	Ideal for storing structured tabular data
Example File Extensions	.txt	.jpg, .mp3, .exe	.csv

1. Opening a Text File

- Use the open() function to open a text file.
- Syntax: file_object = open("filename.txt", mode)
- Replace "filename.txt" with the name of the text file and mode with the desired file open mode.

2. Text File Modes

- 'r': Read mode. Opens a file for reading only. Raises an error if the file does not exist.
- 'r+': Read and Write mode. Opens a file for both reading and writing.
- 'w': Write mode. Opens a file for writing only. Creates a new file if it does not exist. Truncates the file if it exists.
- 'w+': Write and Read mode. Opens a file for reading and writing. Creates a new file if it does not exist. Truncates the file if it exists.
- 'a': Append mode. Opens a file for appending data. Creates a new file if it does not exist.
- 'a+': Append and Read mode. Opens a file for appending data and reading. Creates a new file if it does not exist.

3. Closing a Text File

- Always close a file after operations to release system resources.
- Use the close() method on the file object: file_object.close().

4. Opening a File Using with Clause

The with statement ensures that the file is properly closed after its suite finishes executing.

Syntax:

```
with open("filename.txt", mode) as file_object:
    # Perform file operations
```

5. Writing/Appending Data to a Text File

- Use the write() method to write data to a file.
  The write() function will write the content in the file without adding any extra characters.
  file_name.write(content)
- Use the writelines() method to write a sequence of lines to a file.
  file_name.writelines(sequence_of_lines)
- If the file is opened in write mode ('w' or 'w+'), it will overwrite existing content.
- If the file is opened in append mode ('a' or 'a+'), new data will be added to the end of the file.

6. Reading from a Text File

- Use the read() method to read the entire contents of a file as a single string if value of n is not given else it will read n characters from the current position.
  File_object.read([n])
- Use the readline() method to read a single line from the file.
  File_object.readlines()
  Note: '\n' is treated as a special character of two bytes.
- Use the readlines() method to read all lines from the file into a list.

7. seek() and tell() Methods

seek() method is used to position the file object at a particular position in a file. The syntax of seek() is:

file_object.seek(offset [, reference_point])

In the above syntax, offset is the number of bytes by which the file object is to be moved. reference_point indicates the starting position of the file object. That is, with reference to which position, the offset has to be counted. It can have any of the following values:

0 - beginning of the file

1 - current position of the file

2 - end of file

By default, the value of reference_point is 0, i.e. the offset is counted from the beginning of the file.

For example, the statement fileObject.seek(5,0) will position the file object at 5th byte position from the beginning of the file.

tell() method returns the current file position. This function returns an integer that specifies the current position of the file object in the file. The position so specified is the byte position from the beginning of the file till the current position of the file object. The syntax of using tell() is:

file_object.tell()

Questions:

S.No.	1 Mark Questions	Answers
1.	What is the extension of regular text files? a).txt b).dat c).ppt d) .doc	A
2.	Which files can be opened in human readable form? a) binary files b) text files c) Both a and b d)None	B
3.	What is the default mode in which text file is opened? a)write b)read c)append d)None	B
4.	Which statement is correct for opening the file? a) f=open(“c:\\data.txt”,”r”) b)f=open(r”c:\data.txt”,”r”) c)Both a and b d)None	C
5.	Which of the following mode cannot be used for opening the text file? a)’r’ b)’w+’ c)’a’ d)’rb+’	D
6.	Which is correct way of closing the text file? a)f.close() b)close(f) c) Both a and b d)None	A
7.	Which statement is correct to read n bytes from text file using f as file object? a)f.read(n) b)f.readline(n) c)Both a and b d)None	A
8.	Which of the following function is used to read all the lines of the text file? a)readline() b)read() c)readlines() d)readit()	C
9.	What is the return datatype of read () function? a)string b)List c)Tuple d)Dictionary	A
10.	What is the return datatype of readlines() function? a)string b)List c)Tuple d)Dictionary	B
11.	Which function is used to write group of characters on to the text file?	B

	a)writegroup() b)write() c)writechar() d)writeall()	
12.	Which function is used to write List of strings on to the text file? a)writelist() b)writeline() c)writelines() d)writeall()	C
13.	In which mode text file should be opened to add the data at the end to the text file? a)'r' b)'w' c)'a' d)'b'	C
14.	Which of the following command can be used to open the text file in writing as well as reading mode? a)f=open("c:\data.txt",'r') b)f=open("c:\data.txt",'w+') c)f=open(c:\\data.txt",'w+) d) f=open("c:\\data.txt",'w')	C
15.	hich of the following statements is true regarding the opening modes of a file? a) While opening a file for reading, if the file does not exist, an error occurs. b) While opening a file for writing ,if the file does not exist, an error occurs. c) While opening a file for reading, if the file does not exist, a new file is created. d) None of the above.	A
16.	State True or False "csv files are special text files"	True
17.	State True or False "text files are slower in processing as they requires translation of special characters"	True
18.	Assertion(A): File opened in' w' mode always places the cursor at the beginning of the file and never generates an error. Reason(R): 'w' mode creates the file even if the file doesn't exist. a)Both A and R are true and R is the correct explanation of A. b)Both A and R are true but R is not the correct explanation of A. c)A is true but R is false. d)R is true but A is false.	A
19.	Assertion(A):Text file contains data in human readable form. Reason(R): It executes faster than any other type of the file. a)Both A and R are true and R is the correct explanation of A. b)Both A and R are true but R is not the correct explanation of A. c)A is true but R is false. d)R is true but A is false.	C
20.	Assertion(A): read()and readline() are used to read the data from the text file. Reasoning(R): readlines() function is used to read all lines from the file in the form of a List. a)Both A and R are true and R is the correct explanation of A. b)Both A and R are true but R is not the correct explanation of A. c)A is true but R is false. d)R is true but A is false.	B

	2 Mark questions	

1.
Ans
	What is the difference between read() and readline() function of text files?
	The read() function read specified number of n bytes. If n is not specified, read the entire file.
	e.g. s=f.read(10) #read 10 bytes
	s= f.read() # read the entire file
	
	The readline() function read a single line. If n is specified , read n bytes from the file.
	e.g. p=f.readline() # read single line
	p=f.readline(7) # read 7 bytes from the file	

2.
Ans
	What is the difference between readlines() and readline() function used with the text file?
	The function readlines() read all the lines of the text file and store them as elements of the List.
	e.g. s= f.readlines() # all lines of text file will be read and store in list s
	
	The function readline() will read a single line from the text file and if n is specified as the argument, read n bytes from the file.
	e.g. p=f.readline() # read single line
	p=f.readline(7) # read 7 bytes from the file	

3. Ans	Name the functions used to write data on the text files. Explain The two functions write() and writelines() are used to write data on the text files. a)write()- This function writes a group of characters on to the text file. e.g. s="Computer Science" f.write(s) # It will write string s to the file using file object f b) writelines()- This function write strings in List as Lines to the file. e.g. f.writelines(L) # It will write strings in List L as lines in the file using file pointer f.	
4. Ans	What is the difference between 'w' and 'a' mode used while opening a text file? When 'w' mode is used while opening the text file , it opens the file in write mode and places the cursor at the beginning of the file and truncates the data of the file. And if file doesn't exist ,it creates the file. Whereas when 'a' mode is used while opening the file, it opens the file in append mode and places the cursor at the end of the file for adding the data at the end. Here also file is created , if file doesn't exist.	
5. Ans	What is the difference between 'r+' mode and 'w+' mode used while opening the text file? With both the modes reading and writing operations can take place, but difference is that if file is opened using 'w+' mode, file is created if file doesn't exist, whereas if file is opened using 'r+' mode, error is raised if file doesn't exist.	
6.	If the focus.txt file contains the following text: ***Mindfulness, cognitive training, and a healthy lifestyle may help sharpen your focus.*** Find the output of the following code: F=open("focus.txt",'r') S=F.read(11) print(S) F.close()	

Ans	Mindfulness
7.	Find the output of the following code: F=open(“focus.txt”,’r’) S= F.readline() print(S) T=F.readline() print(T) F.close()
Ans	Mindfulness, cognitive training, and a healthy lifestyle may help sharpen your focus
8.	Find the output of the following code: F=open(“focus.txt”,’r’) L= F.readlines() for a in L: print(a.upper()) F.close()
Ans	MINDFULNESS, COGNITIVE TRAINING, AND A HEALTHY LIFESTYLE MAY HELP SHARPEN YOUR FOCUS

9.	Find the output of the following code:
	
	F=open(“focus.txt”,’a+’)
	S= “ sleep reduces stress hormones that can be harmful to the brain”
	F.write(S)
	F.seek(0) # bring cursor to the beginning of the file
	L=F.readlines()
	print(L)
	F.close()

Ans	['Mindfulness, cognitive training, and a healthy lifestyle may help\n', 'sharpen your focus sleep reduces stress hormones that can be harmful to the brain']
10.	Find the output of the following code: F=open(“wish.txt”, ’w’) F.write(“Day”) F.close() If the file contains “Good” before execution, what will be the contents of the file after execution of the above code.
Ans	After execution, file will contain “Day” only as previous data will be truncated by write operation over the file.

	3 Marks questions
1.	Write a program to read text file story.txt and count number of lines starting with letter ‘A’ or ‘a’.

Ans	F=open("story.txt",'r')
	count=0
	L=F.readlines()
	for i in L:
	  if i[0]=='A' or i[0]=='a':
	  count=count+1
	print("no. of lines starting with a=",count)
	F.close()

2.	Write a program to read the file data.txt and count number of uppercase, lowercase in it.

Ans	F=open("data.txt",'r')
	u=0
	l=0
	s=F.read()
	for i in s:
	  if i.isupper():
	  u=u+1
	  if i.islower():
	  l=l+1
	print("Number of uppercase characters=",u)
	print("Number of lowercase characters=",l)
	F.close()

3.	Write a program to read the file data.txt and count number of spaces in it.

Ans	F=open("data.txt",'r')
	space=0
	s=F.read()
	for i in s:
	  if i.isspace():
	  space=space+1
	 print("Number of spaces=",space)
	F.close()

4.	Write a program to read the file hash.txt and display the number characters up to first #.

Ans	F=open("hash.txt",'r')
	count=0
	s=F.read()
	for i in s:
	  if i!='#':
	  count=count+1
	  else:
	  break
	print("Number of characters up till # =",count)
	F.close()

5.	Write a program to read the file alphabet.txt and display all the lines in uppercase.

Ans	F=open("alphabet.txt",'r') L=F.readlines() for i in L: print(i.upper()) F.close()
6. Ans	Write a program to read the file data.txt and count number of lines present in it. F=open("data.txt",'r') L=F.readlines() print("Number of lines in the file=",len(L)) F.close()

7.
 	
Ans
	Write a program to read the file data.txt and display only the digits present in it.
	
	F=open("data.txt",'r')
	s=F.read()
	for letter in s:
	  if letter.isdigit():
	  print(letter)
	F.close()

8. Ans	Write a program to read the file story.txt and display second last line of the file. F=open("story.txt",'r') L=F.readlines() print(L[-2]) F.close()

9.
 	
Ans
	Write a program to read the file article.txt and count occurrences of words “the” in the file.
	
	F=open("article.txt",'r')
	count=0
	s=F.read()
	L=s.split()
	for word in L:
	  if word=="the":
	  count=count+1
	print("No. of occurences of word the=",count)
	F.close()

10.
 	
Ans
	Write a program to read the file letter.txt and display those words which has less than or equal to four characters.
	
	F=open("story.txt",'r')
	s=F.read()
	L=s.split()
	for word in L:
	  if len(word)<=4:
	  print(word)
	F.close()

	4 Marks questions

1	Write python statements for opening the following files. Also, write the Python statements to open the following files: a) a text file “example.txt” in both read and write mode b) a text file “bfile.dat” in write mode c) a text file “try.txt” in append and read mode d) a text file “btry.dat” in read only mode.
Ans	(a) F = open(‘example.txt’,”r+”) (b) F = open(“bfile.dat” , “w”) (c) F = open(‘try.txt’ , “a”) (d) F = open(‘btry’ , ’r’)
2 (i)	What is the difference between the following set of statements (a) and (b): a) P = open(“practice.txt”,”r”) P.read(10) b) with open(“practice.txt”, “r”) as P: x = P.read()
Ans	Set of statements (a) would read the file “practice.txt” and returns a string that contains first 10 characters of the text file. Set of statements (b) will read the text file “practice.txt” and returns a string that contains entire contents of the text file.
(ii)	Write a command(s) to write the following lines to the text file named hello.txt. Assume that the file is opened in append mode. “ Welcome my class” “It is a fun place” “You will learn and play”
Ans	F = open(“TFILE.txt”, ‘a’) L = [“ Welcome my class” , “It is a fun place” , “You will learn and play”] F.writelines(L) F.close()
3	Write a method/function COUNTLINES_ET() in python to read lines from a text file REPORT.TXT, and COUNT those lines which are starting either with ‘E’ and starting with ‘T’ respectively. Display the Total count separately.

Ans	def COUNTLINES_ET():
	f=open(“REPORT.TXT”)
	d=f.readlines()
	le=0
	lt=0
	for i in d:
	if i[0]==’E:
	le=le+1
	elif i[0]==’T’:
	lt=lt+1
	print(“no of line start with”,le)
	print(“no of line start with”,lt)

4	Write a function filter(oldfile, newfile) that copies all the lines of a text file “source.txt” onto “target.txt” except those lines which starts with “@” sign.
Ans	def filter(oldfile, newfile): f1 = open("oldfile","r")

	f2 = open("newfile","w")
	while True:
	text= f1.readline()
	if len(text) ==0:
	break
	if text[0] == '@':
	continue
	f2.write(text)
	f1.close()
	f2.close()

5 (i) Ans	Write a user defined function countwords() to display the total number of words present in the file from a text file "Quotes.Txt". def countwords(): s = open("Quotes.txt","r") f = s.read() z = f.split () print ("Total number of words:", len(z))

(ii)
 	
Ans
	Write a function COUNT_AND( ) in Python to read the text file "STORY.TXT" and count the number of times "AND" occurs in the file. (include AND/and/And in the counting)
	
	def COUNT_AND( ):
	count=0
	file=open('STORY.TXT','r')
	line = file.read()
	word = line.split()
	for w in word:
	if w.upper() =='AND':
	count=count+1
	print(count)
	file.close()

	5 Marks questions
1 (i) Ans	Differentiate between Text files and Binary files. **Text file**: A text file is simply a sequence of ASCII or Unicode characters.A line is a sequence of characters, stored on permanent storage. In a text file, each line is terminated by a special character, known as End Of Line (EOL). Text file can be created using any text editor. Ex. Myfile.txt. **Binary file**: A binary file stores the data in the same way as stored in the memory. The .exe files, mp3 file, image files, word documents are some of the examples of binary files. We can't read a binary file using a text editor.
(ii) Ans	Write a method COUNTWORDS() in Python to read data from text file 'ARTICLE.TXT' and display the count of words which ends with a vowel. For example, if the file content is as follows: An apple a day keeps you healthy and wise The COUNTWORDS() function should display the output as: Total words which ends with vowel = 4 def COUNTWORDS(): fil = open('ARTICLE.TXT' , 'r') data = fil.read()

```
        words = data.split()
        count = 0
        for w in words:
            if w[-1] in 'aeiouAEIOU':
                count += 1
        fil.close()
        print('Total words which ends with vowel =',count)
```

2 (i) Ans

Explain seek() and tell() methods.

seek() method is used to position the file object at a particular position in a file.
The syntax of seek() is:
file_object.seek(offset [, reference_point])
For example, the statement fileObject.seek(5,0) will position the file object at 5th byte position from the beginning of the file.
tell() method returns the current file position. This function returns an integer that specifies the current position of the file object in the file. The position so specified is the byte position from the beginning of the file till the current position of the file object. The syntax of using tell() is:
file_object.tell()

(ii)

Write a function COUNT() in Python to read from a text file 'rhym.txt' and display the count of words in each line.
Example: If the content of 'rhym.txt' is as follows:
Jack and jill
Went up the hill
To enjoy
Then the COUNT() function should display output as:
Line 1 : 3
Line 2 : 4
Line 3 : 2

Ans

```
def COUNT():
    fil = open('rhym.txt')
    lines = fil.readlines()
    c = 1
    for l in lines:
        words = l.split()
        print('Line',c,':',len(words))
        c = c+1
    fil.close()
```

3 (i) Ans

Differentiate between w+ and a+ file modes.

Parameter	w+ Mode	a+ Mode
Description	Opens a file for reading and writing (truncate file).	Opens a file for reading and appending (creates file if not exists).
File Pointer	Points at the beginning of the file after opening.	Points at the end of the file after opening.
Write Operation	Overwrites existing content with new data.	Appends new data to the end of the file.
Read Operation	Reads from the beginning of the file.	Reads from the beginning of the file.
File Truncation	Truncates the file to zero length if it exists.	Does not truncate the file; preserves existing content.

	Creation	Creates a new file if it does not exist.	Creates a new file if it does not exist.
	Usage	Useful for scenarios where existing content needs to be overwritten or a new file needs to be created.	Useful for scenarios where data needs to be appended to an existing file without losing the existing content.

(ii) Write a function WE_WORDS() in Python to read from a text file 'TEXT.TXT' and display the count of words which starts with 'WE'.
Example: If the content of 'TEXT.TXT' is as follows:
WE MUST WELCOME ALL WEATHER FROM WEST
Then the WE_WORDS() function should display output as:
TOTAL WORDS STARTING WITH WE = 4

Ans

```
def WE_COUNT():
    fil = open('TEXT.TXT')
    data = fil.read()
    words = data.split()
    count = 0
    for w in words:
        if w.startswith('WE'):
            count = count + 1
    print('TOTAL WORDS STARTING WITH WE=',count)
    fil.close()
```

4 (i) What will be the return datatype of the following methods:
read()
readlines()

Ans
read() – String
readlines() – List

(ii) A pre-existing text file data.txt has some words written in it. Write a python function displaywords() that will print all the words that are having length greater than 3.
Example:
For the fie content:
A man always wants to strive higher in his life He wants to be perfect.
The output after executing displayword() will be: Always wants strive higher life wants perfect

Ans

```
def displaywords():
        f = open('data.txt','r')
        s = f.read()
        lst = s.split() for x in lst:
        if len(x)>3:
                print(x, end=" ")
        f.close()
```

5 (i) Explain the use of seek() method.

Ans **seek() method** is used to position the file object at a particular position in a file.
The syntax of seek() is:
file_object.seek(offset [, reference_point])
In the above syntax, offset is the number of bytes by which the file object is to be moved. reference_point indicates the starting position of the file object. That is,

	with reference to which position, the offset has to be counted. It can have any of the following values: 0 - beginning of the file 1 - current position of the file 2 - end of file By default, the value of reference_point is 0, i.e. the offset is counted from the beginning of the file.

(ii)
 	
Ans
	A pre-existing text file info.txt has some text written in it. Write a python function countvowel() that reads the contents of the file and counts the occurrence of vowels(A,E,I,O,U) in the file.
	
	def countvowels():
	f = open('info.txt', 'r')
	s = f.read()
	count = 0
	for x in s:
	if x in 'AEIOU':
	count+=1
	print(count) f.close()

BINARY FILE HANDLING IN PYTHON

Binary files store data in the binary format (that is, in the form of 0's and 1's) which is understandable by the machine. So when we open the binary file in our machine, it decodes the data and displays it in a human-readable format. It is important to note that the binary file contents can be displayed correctly using only specialized applications that can read binary data. If we open any binary file using a normal text editor like a notepad, we may see strange characters.

Examples of binary files include files stored with the extension of .dat, .doc, .docx, .mp4, etc. As you may relate now, these files can be opened correctly using specific applications only, that are different for each file extension. Try opening any of these binary files using notepad, and observe the magic (file opens but with unreadable contents). In this chapter of binary file handling, we'll learn to create such files (in their simple forms), modify its contents and display its contents properly.

Binary File Modes:

File mode governs the type of operations (read/write/append) that is possible in the opened file. It refers to how the file will be used once it's opened.

File Mode Description:

rb: Read Only: Opens existing file for read operation

wb: Write Only: Opens file for write operation. If the file does not exist, the file is created. If a file exists, it overwrites data.

ab: Append: Opens file in write mode. If a file exists, data will be appended at the end.

rb+: Read and Write: File should exist, Both read and write operations can be performed.

wb+: Write and Read: File created if it does not exist, If file exists, file is truncated.

ab+: Write and Read: File created if does not exist, If file exists data is truncated.

Writing data to a Binary File:

Pickle is a special python package (module) that is used to generate data in binary format. Pickle comes with few methods like load() and dump( ) to read and write data in binary format.

Pickle Module: Python Pickle is used to serialize and deserialize a python object structure. Any object on python can be pickled so that it can be saved on disk.

Pickling: Pickling is the process whereby a Python object hierarchy is converted into a byte stream.

Unpickling: A byte stream is converted into object hierarchy.

To use the picking methods in a program, we have to import the pickle module using import keyword.

Example:

import pickle #don’t write pickel

In this module, we shall discuss two of its useful functions, which are:

i. dump( ) : To store/write the object data to the file.

ii. load( ) : To read the object data from a file and return the object data.

Syntax:

Write the object to the file:

pickle.dump(List_name, file-object) #To write a list object into a binary file

Read the object from a file:

pickle.load(file-object)

Example:

```
import pickle
list =[ ] # empty list
while True:
        roll = input("Enter student Roll No:")
        sname = input("Enter student Name :")
        student = {"roll":roll,"name":sname} # create a dictionary
        list.append(student) # add dictionary as element in list
        choice= input("Want to add more record(y/n) :")
        if(choice=='n'):
                break
file = open("student.dat","wb")#open file in binary & write mode
pickle.dump(list, file) #imp: first data, then file handle
file.close( )
```

OUTPUT:

```
Enter student Roll No:1201
Enter student Name :Anil
Want to add more record(y/n) :y
Enter student Roll No:1202
Enter student Name :Sunil
Want to add more record(y/n) :n
```

Read data from a Binary File:

To read the data from a binary file, we have to use the load( ) function of the pickle module.

Example:

```
import pickle
file = open("student.dat", "rb")
list = pickle.load(file)
print(list)
file.close( )
```

OUTPUT:

[{'roll': '1201', 'name': 'Anil'}, {'roll': '1202', 'name': 'Sunil'}]

To update a record in Binary File:

Locate the record to be updated by searching for it. Make changes in the loaded record in memory. Write back onto the file at the exact location of the record.

```
import pickle
roll = input('Enter roll number whose name you want to update in binary file :')
file = open("student.dat", "rb+")
list = pickle.load(file)
found = 0
lst = [ ]
for x in list:
        if roll in x['roll']:
                found = 1
        x['name'] = input('Enter new name: ')
        lst.append(x)
        if found == 1:
                file.seek(0)
        pickle.dump(lst, file)
        print("Record Updated")
else:
        print('roll number does not exist')
file.close( )
```

OUTPUT:

Enter roll number whose name you want to update in binary file :1202

Enter new name: Harish

Record Updated

Deleting a record from binary file:

```
import pickle
roll = input('Enter roll number whose record you want to delete:')
file = open("student.dat", "rb+")
list = pickle.load(file)
found = 0
lst = []
for x in list:
        if roll not in x['roll']:
            lst.append(x)
        else:
            found = 1
        if found == 1:
            file.seek(0)
            pickle.dump(lst, file)
        print("Record Deleted ")
else:
        print('Roll Number does not exist')
file.close( )
```

OUTPUT:

Enter roll number whose record you want to delete:1201

Record Deleted

Searching a record in a binary file:

```
import pickle
roll = input('Enter roll number that you want to search in binary file :')
file = open("student.dat", "rb")
list = pickle.load(file)
file.close( )
for x in list:
        if roll in x['roll']:
            print("Name of student is:", x['name'])
            break
        else:
            print("Record not found")
```

OUTPUT:

Enter roll number that you want to search in binary file :1202

Name of student is: Harish

tell( ) and seek( ) methods:

tell( ): It returns the current position of cursor in file.

Example:

```
fout=open("story.txt","w")
fout.write("Welcome Python")
print(fout.tell( ))
fout.close( )
```

Output:

15

seek(offset, reference_point): Change the cursor position by bytes as specified by the offset, from the reference point.

Example:

```
fout=open("story.txt","w")
fout.write("Welcome Python")
fout.seek(5)
print(fout.tell( ))
fout.close( )
```

Output:

5

1 Mark Questions

1. The process of converting byte stream back to the original structure is known as

a. Picklingb. b. Unpickling c. Packing d. Zipping

2. Which file mode is used to handle binary file for reading.

a. rb b. rw c. r d. w

3. Which of the following is not a correct statement for binary files?

a. Easy for carrying data into buffer b. Much faster than other file systems

c. Characters translation is not required d. Every line ends with new line character '\n'

4. Which one of the following is correct statement?

a. import – pickle b. pickle import c. import pickle d. All the above

5. Which of the following file mode opens a file for append or read a binary file and moves the files pointer at the end of the file if the file already exist otherwise create a new file?

a. a b. ab c. ab+ d. a+

6. Which of the following file mode opens a file for reading and writing both as well as overwrite the existing file if the file exists otherwise creates a new file?

a. w b. wb+ c. wb d. rwb

7. Mr Sharma is working on a binary file and wants to write data from a list to a binary file. Consider list object as l1, binary file sharma_list.dat, and file object as f. Which of the following can be the correct statement for him?

a. f = open('sum_list','wb'); pickle.dump(l1,f)

b. f = open('sum_list','rb'); l1=pickle.dump(f)

c. f = open('sum_list','wb'); pickle.load(l1,f)

d. f = open('sum_list','rb'); l1=pickle.load(f)

8. Every file has its own identity associated with it. This is known as:

a. icon b. extension c. format d. file type

9. EOL in a file stands for :

a. End of Lines b. End of Line c. End of List d. End of Location

10. Which of the following file types allows you to store large data files in the computer memory?

a. Binary Files b. Text Files c. CSV Files d. None of these

2 Marks Questions

1. Write a program in python to write and read structure, dictionary to the binary file.

2. BINARY file is unreadable and open and close through a function only so what are the advantages of using binary file

3. Write a statement to open a binary file name sample.dat in read mode and the file sample.dat is placed in a folder (name school) existing in c drive.

4. When do you think text files should be preferred over binary files?

5. Consider a binary file employee.dat containing details such as empno:ename:salary (separator ':') write a python function to display details of those employees who are earning between 20000 and 30000(both values inclusive)

6. Differentiate between pickle.load() and pickle.dump() methods with suitable examples.

7. A binary file “Book.dat” has structure [BookNo, Book_Name, Author, Price].Write a user defined function CreateFile() to input data for a record and add to Book.dat

8. A binary file “STUDENT.DAT” has structure (admission_number, Name, Percentage). Write a function countrec() in Python that would read contents of the file “STUDENT.DAT” and display the details of those students whose percentage is above 75.

9. A binary file “Store.dat” has structure [ItemNo, Item_Name, Company, Price]. Write a function CountRec(Company) in Python which accepts the Company name as parameter and count and return number of Items by the given Company are stored in the binary file “Store.dat”.

10. A binary file “Store.dat” has structure [ItemNo, Item_Name, Company, Price]. Write a function AddRecord() which accepts a List of the record [ItemNo, Item_Name, Company, Price] and appends in the binary file “Store.Dat”.

3 Marks Questions

1. A binary file “Book.dat” has structure [BookNo, Book_Name, Author, Price].

i. Write a user defined function CreateFile() to input data for a record and add to “Book.dat” .

ii. Write a function CountRec(Author) in Python which accepts the Author name as parameter and count and return number of books by the given Author are stored in the binary file “Book.dat”

2. A binary file “SCHOOL.DAT” has structure [Roll_Num, Name, Percentage]

i) Write a function Count_Rec() in Python that would read contents of the file “SCHOOL.DAT” and display the details of those students whose percentage is below 33.

ii) Write a function Disp_Rec(alphabet) in Python that would read contents of the file “SCHOOL.DAT” and display the details of those students whose name begins with the alphabet as passed as parameter to the function.

3. A binary file “STOCK.DAT” has structure [ITEMID, ITEMNAME, QUANTITY, PRICE]. Write a user defined function MakeFile( )to input data for a record and add to Book.dat.

4. Write a function GetPrice(ITEMID) in Python which accepts the ITEMID as parameter and returns PRICE of the Item stored in Binary file STOCK.DAT.

5. A binary file “EMPLOYEE.DAT” has structure (EMPID, EMPNAME, SALARY). Write a function CountRec( ) in Python that would read contents of the file “EMPLOYEE.DAT” and display the details of those Employees whose Salary is above 20000.

6. A binary file “EMPLOYEE.DAT” has structure (EMPID, EMPNAME, SALARY). Write a function to display number of employees having Salary more than 20000.

7. A binary file named “EMP.dat” has some records of the structure [EmpNo, EName, Post, Salary], Write a user-defined function named NewEmp() to input the details of a new employee from the user and store it in EMP.dat.

8. Write a user-defined function named SumSalary(Post) that will accept an argument the post of employees & read the contents of EMP.dat and calculate the SUM of salary of all employees of that Post.

9. A binary file named “TEST.dat” has some records of the structure [TestId, Subject, MaxMarks, ScoredMarks] Write a function in Python named DisplayAvgMarks(Sub) that will accept a subject as an argument and read the contents of TEST.dat.

10. Write a python program to search and display the record of the student from a binary file “Student.dat” containing students records (Rollno, Name and Marks). Roll number of the student to be searched will be entered by the user.

5 Marks Questions

1. A binary file “student.dat” has structure [rollno, name, marks].

i. Write a user defined function insertRec() to input data for a student and add to student.dat.

ii. Write a function searchRollNo(r) in Python which accepts the student’s rollno as parameter and searches the record in the file “student.dat” and shows the details of student i.e. rollno, name and marks (if found) otherwise shows the message as ‘No record found’.

2. Write a python program to create binary file dvd.dat and write 10 records in it:

Dvd id,dvd name,qty,price

Display those dvd details whose dvd price is more than 25.

CSV FILES

A CSV (Comma-Separated Values) file is a plain text file format used to store tabular data, where each line represents a row, and each value within a row is separated by a comma or other delimiter.

A CSV file (Comma Separated Values file) is a type of plain text file that uses specific structuring to arrange tabular data.,

CSV File operations in Python 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. •

WHY USE CSV?

• The extensive use of social networking sites and their various associated applications requires the handling of huge data.

But the problem arises as to how to handle and organize this large unstructured data?

• The solution to the above problem is CSV.

Thus, 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 file formats are of plain text format, it makes it very easy for website developers to create applications that implement CSV.

• 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 CSV stands for “comma separated values”.

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.

CSV File operations 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.

• To perform read and write operations 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:–

1)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 from a CSV file

– 2. Writing to a CSV file

2) Opening and closing of CSV File

```
# Open the CSV file in write mode
file = open('data.csv', 'w', newline='')
# Perform operations on the file, such as writing data
file.write("Name, Age, City\n")
file.write("Alice, 25, New York\n")
file.write("Bob, 30, San Francisco\n")

# Close the file to free up system resources
file.close()
```

•Reading from a CSV File

- Reading from a CSV file is done using the reader object.

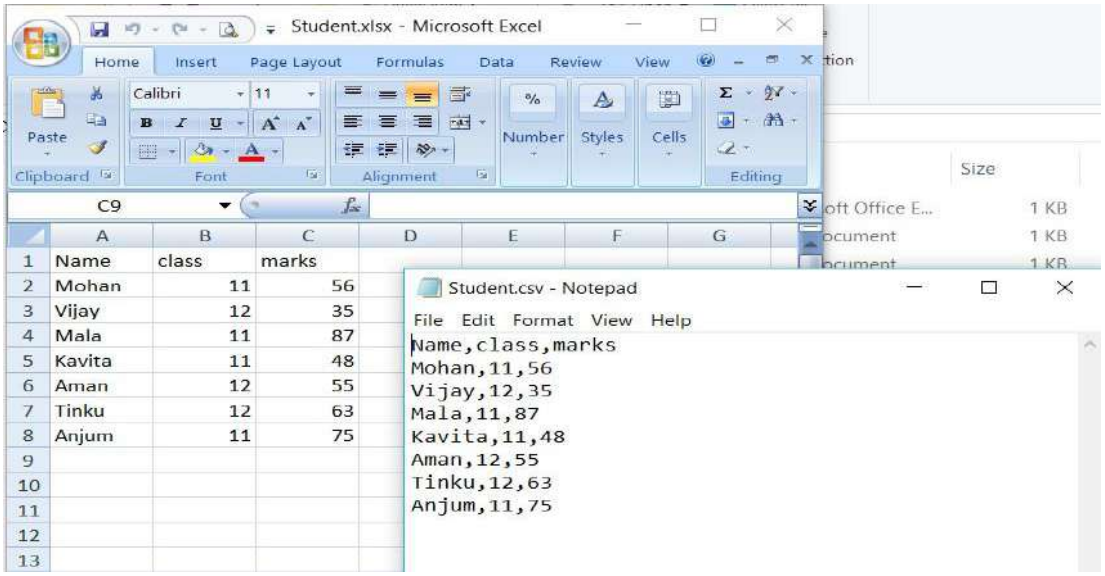
The CSV file is opened as a text file with Python's built-in open()function, which returns a file object.

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. We can also use next() directly on it to read the next line of the CSV file, or we can treat it like a list in a for loop to read all the lines of the file (as lists of the

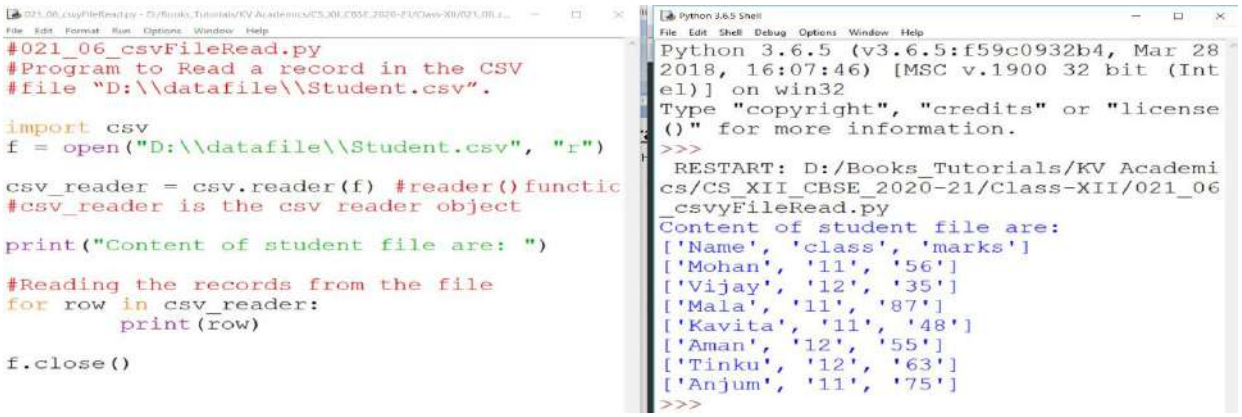
file's fields).

- Let us 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.

In student.csv (notepad) file, the first line is the header and remaining lines are the data/ records. The fields are separated by comma. In general, the separator character is called a delimiter, and the comma is



```
#021_06_csvFileRead.py
#Program to Read a record in the CSV
#file "D:\\datafile\\Student.csv".

import csv
f = open("D:\\datafile\\Student.csv", "r")

csv_reader = csv.reader(f) #reader()functic
#csv_reader is the csv reader object

print("Content of student file are: ")

#Reading the records from the file
for row in csv_reader:
        print(row)

f.close()
```

```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28
2018, 16:07:46) [MSC v.1900 32 bit (Int
el)] on win32
Type "copyright", "credits" or "license
()" for more information.
>>>
 RESTART: D:/Books_Tutorials/KV Academi
cs/CS_XII_CBSE_2020-21/Class-XII/021_06
_csvyFileRead.py
Content of student file are:
['Name', 'class', 'marks']
['Mohan', '11', '56']
['Vijay', '12', '35']
['Mala', '11', '87']
['Kavita', '11', '48']
['Aman', '12', '55']
['Tinku', '12', '63']
['Anjum', '11', '75']
>>>
```

not the only one used. Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters.

Program to read the contents of "student.csv" file

Every record is stored in reader object in the form of a List. We first open the CSV file in READ mode. The file object is named f. The file object is converted to csv.reader object. The reader object is used to read records as lists from a csv file. Iterate through all the rows using a for loop. row is nothing but a list containing all the field values

Writing to CSV FILE

STEPS:

1. import csv library.
2. Define a filename and Open the file using open().
3. Create a csvwriter object using csv.writer().
4. Write the header.
5. Write the rest of the data.

Writer Objects:

csvwriter.writerow(row)

Write the row parameter to the writer's file object

csvwriter.writerows(rows)

Example code to demonstrate use of writer objects:

```
import csv
# Data to be written to the CSV file
data_single_row = ['Name', 'Age', 'Grade']
data_multiple_rows = [ ['Alice', 20, 'A'],['Bob', 22, 'B']  ['Charlie', 21, 'C']]
```

```
# Writing data using writerow()
with open('students.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(data_single_row)                # Write a single row
    writer.writerow(['David', 23, 'B'])             # Write another single row
# Writing data using writerows()
with open('students.csv', 'a', newline='') as file:       # Use 'a' to append to the existing file
    writer = csv.writer(file)
    writer.writerows(data_multiple_rows)            # Write multiple rows at once
```

- We first define the data to be written to the CSV file as a list of lists (**data**).
- Using **writerow()**, we write each row of data to the CSV file one by one.
- Then, using **writerows()**, we append additional rows of data to the CSV file at once.

PRACTICE QUESTIONS

MCQ

1. Which Python module is used to work with CSV files?

A) csv

B) pandas

C) json

D) os

2. What is the purpose of the csv.writer() object in Python?

A) To read data from a CSV file

B) To write data into a CSV file

C) To perform mathematical operations

D) To create directories

3. To open a CSV file for writing, which mode should you use in the open() function?

A) 'r'

B) 'w'

C) 'a'

D) 'rb'

4. Which method is used to write a single row into a CSV file using the csv.writer() object?

A) write()

B) writerows()

C) writerow()

D) row()

5. How do you close a CSV file after you finish working with it in Python?

A) close_file()

B) close()

C) end()

D) stop()

6. Which method is used to write multiple rows into a CSV file using the csv.writer() object?

A) write()

B) writerows()

C) writerow()

D) row()

7. What parameter should be used in the open() function to ensure correct newline handling when working with CSV files?

A) newline='ignore'

B) newline=''

C) newline='skip'

D) newline=None

8. To read data from a CSV file in Python, which method of the csv.reader() object is used to iterate through each row?

A) readline()

B) next()

C) readrows()

D) for loop

9. What does the newline='' parameter in the open() function ensure when working with CSV files?

A) It skips writing empty lines.

B) It converts newlines to spaces.

C) It ensures universal newline support.

D) It prevents reading blank rows.

10. Which of the following is NOT a valid mode for opening a CSV file in Python?

A) 'r' (read mode)

B) 'w' (write mode)

C) 'a' (append mode)

D) 'x' (exclusive creation mode)

11. What type of data format is a CSV file?

A) Binary

B) Text-based

C) Image

D) Executable

12. Which method is used to read the entire contents of a CSV file into a list of lists using the csv.reader() object?

A) read()

B) readline()

C) next()

D) list()

13.When using the csv.writer() object, which method is used to write a list of data into a CSV file as a single row?

A) write()

B) writerow()

C) writeall()

D) row()

14,In Python's CSV module, which of the following is true about delimiter characters?

A) The delimiter character cannot be customized.

B) The delimiter character must always be a comma.

C) The delimiter character separates values within a CSV file.

D) The delimiter character is used for comments.

15. How does the csv.writerows() method differ from the csv.writerow() method?

A) writerows() writes a single row, while writerow() writes multiple rows.

B) writerows() writes multiple rows at once, while writerow() writes one row at a time.

C) writerows() converts data to CSV format, while writerow() writes data as-is.

D) writerows() automatically adds headers, while writerow() does not.

16.Which of the following is a benefit of using the csv module in Python for CSV file operations?

A) It requires less memory compared to other modules.

B) It automatically converts CSV files to Excel format.

C) It provides advanced data visualization features.

D) It supports various data formats other than CSV.

17.What does the newline='' parameter in the open() function prevent when writing to CSV files?

A) It prevents empty lines from being written.

B) It prevents writing data as binary.

C) It prevents newlines from being converted to spaces.

D) It prevents duplicate rows from being written.

18. When using the csv.reader() object to read from a CSV file, what type of data structure is each row of data represented as?

A) String

B) Dictionary

C) List

D) Tuple

19. How does the csv.DictReader() class differ from the csv.reader() class in Python?

A) DictReader() reads data as lists, while reader() reads data as dictionaries.

B) DictReader() reads data with column headers, while reader() does not.

C) DictReader() reads data as tuples, while reader() reads data as dictionaries.

D) DictReader() reads data as dictionaries, while reader() reads data as lists.

20. What is the purpose of using the newline='' parameter when opening a CSV file in Python?

A) To convert newlines to spaces.

B) To ensure cross-platform compatibility for newline characters.

C) To skip writing empty lines to the CSV file.

D) To automatically add headers to the CSV file.

2 marks questions

1. What is the purpose of using the csv module in Python?
2. Discuss the importance of newline handling when working with CSV files.
3. Differentiate between writerow() and writerows() methods in the csv.writer() object.
4. Describe the data format of a CSV file.
5. Explain the concept of a delimiter character in CSV files.
6. Write a Python code snippet to open a CSV file named "data.csv" in write mode and write a single row of data into it using the csv.writer() object.Include the necessary import statement and ensure proper closing of the file after writing.
7. Create a Python script that reads data from a CSV file named "input.csv" using the csv.reader() object and prints each row of data to the console.Handle any exceptions that may occur during file handling.
8. Write a Python program that generates a CSV file named "output.csv" and writes multiple rows of data into it using the csv.writer() object.The data can be generated randomly or from predefined lists.
9. Modify the previous code to append additional rows of data to the "output.csv" file using the csv.writer() object.
10. Implement a Python function that takes a CSV file path as input and returns the total number of rows in the CSV file using the csv.reader() object.

3 marks questions

1. write a Python program that performs the following tasks:
    - Opens a CSV file named "inventory.csv" in read mode using the **csv.reader()** object.
    - Iterates through each row in the CSV file.
    - Checks if the quantity (second column) of each item is less than 10. If so, appends the item's name (first column) to a list named **low_stock_items**.
    - Finally, prints the list **low_stock_items** containing the names of items with low stock.

2. Fill in the blanks in the following code snippet to open a CSV file named "sales.csv" in read mode using the **csv.reader()** object and calculate the total revenue by summing up the values in the third column (Price) of each row.

```
import csv
total_revenue = 0 with open('sales.csv', 'r') as file:
   reader = csv.reader(file) next(reader)    # Skip header row
 for row in reader:
     total_revenue += _______
     print('Total revenue:', total_revenue)
```

3. Given the CSV file "employees.csv":

```
Name,Department,Salary Alice,HR,50000 Bob,Engineering,60000 Charlie,Sales,45000
```

Identify and correct any errors in the following Python code snippet, then determine the output of the corrected code:

```
import csv
 total_salary = 0
with open('employees.csv', 'r') as file:
     reader = csv.reader(file)
              for row in reader:
              total_salary += row[2]
              print('Total salary:', total_salary)
```

4. Write a Python program that opens a CSV file named "students.csv" in write mode using the **csv.writer()** object. The program should prompt the user to enter student names and their respective grades, and then write this data into the CSV file. Ensure appropriate error handling for incorrect input.
5. Explain the concept of delimiter characters in CSV files and discuss their significance when working with the **csv** module in Python. Provide examples of different delimiter characters and explain how they affect the organization and interpretation of data within a CSV file.
6. Write a Python program that reads data from a CSV file named "data.csv" using the csv.reader() object. For each row, if the value in the second column (index 1) is greater than 50, write that row into a new CSV file named "high_scores.csv" using the csv.writer() object.
7. Fill in the blanks in the following code snippet to open a CSV file named "output.csv" in append mode and add a new row containing the student's name, age, and grade using the csv.writer() object.

```
import csv
student_data = ['Alice', 25, 'A']
with open('output.csv', 'a', newline='') as file:
   writer = csv.writer(_____)
```

```
    writer._____(student_data)
```

Output and Error Handling:

8. Given the following CSV file named "inventory.csv":

What will be the output of the following Python program? If there is any error, identify and correct it.

```
total_cost = 0
with open('inventory.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        quantity = int(row[1])
        price = float(row[2])
        total_cost += quantity * price
print('Total inventory cost:', total_cost)
```

9. Write a Python program that reads data from a CSV file named "students.csv" using the csv.reader() object. Create a dictionary where the keys are the student names and the values are lists containing their age and grade. Print the dictionary.
10. Fill in the blanks in the following code snippet to read data from a CSV file named "sales.csv" using the csv.reader() object and calculate the total sales amount. Print the total sales amount.

```
import csv
total_sales = 0
with open('sales.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        sales_amount = float(_____[2])
        total_sales += sales_amount
print('Total sales amount:', total_sales)
```

5 mark questions

1. Write a Python program that reads data from a CSV file named "inventory.csv" using the **csv.DictReader()** class. The CSV file contains columns for "Product", "Quantity", and "Price". Your program should calculate the total value of each product in inventory (quantity * price) and print a summary report showing each product's name and total value.
2. Identify and correct the error in the following Python code that attempts to open a CSV file named "data.csv" for writing using the **csv.writer()** object.

```
import csv
data = [['Name', 'Age', 'City'], ['Alice', 25, 'New York'], ['Bob', 30, 'San Francisco']]
```

with open('data.csv', 'r') as file:

writer = csv.writer(file) writer.writerows(data)

3. Fill in the blanks in the following code snippet to open a CSV file named "output.csv" in write mode and write multiple rows of data into it using the **csv.writer()** object. The data to be written includes product information (name, quantity, price) stored in a list of dictionaries.

import csv

product_data = [{'Name': 'Laptop', 'Quantity': 10, 'Price': 1200}, {'Name': 'Mouse', 'Quantity': 50, 'Price': 20}, {'Name': 'Keyboard', 'Quantity': 20, 'Price': 50}]

with open('output.csv', 'w', newline='') as file:

fieldnames = ['Name', 'Quantity', 'Price']

writer = csv.DictWriter(file, fieldnames=_____)

writer.writeheader()

writer.writerows(_____)

4. Write a Python program that reads data from a CSV file named "sales.csv" using the **csv.reader()** object. The CSV file contains columns for "Date", "Product", and "Revenue". Your program should calculate and print the total revenue earned for each product across all dates.
5. Write a Python program that reads data from two CSV files, "sales.csv" and "inventory.csv", using appropriate methods like **csv.reader()** or **csv.DictReader()**.
   The "sales.csv" file contains columns for "Date", "Product", and "Revenue", while the "inventory.csv" file contains columns for "Product" and "Quantity". Your program should combine these datasets to create a new CSV file named "combined_data.csv" that includes the columns "Date", "Product", "Revenue", and "Available Quantity". Ensure to handle missing or mismatched data appropriately.

Case study based questions-4 marks each

1. Imagine you work for a retail company that stores its daily sales data in a CSV file named "sales_data.csv". Develop a Python script using the csv module to read this file and generate a daily sales report. The report should include total sales revenue, the number of items sold, and a breakdown of sales by product category.
2. You are managing a student gradebook for a school, and the grade data is stored in a CSV file named "gradebook.csv". Design a Python program using the csv module to read and update student grades. Implement functionalities such as adding new grades, calculating average grades for each subject, and generating individual progress reports.
3. In a warehouse setting, you have an inventory CSV file named "inventory.csv" containing product details like name, quantity, and reorder level. Create a Python application using the csv module to track inventory levels, identify low-stock items (below reorder level), and generate a restocking list with recommended quantities.

4. A company receives customer feedback through an online survey, and the feedback data is stored in a CSV file named "feedback_data.csv". Develop a Python script using the csv module to read and analyze the feedback. Implement sentiment analysis to categorize feedback as positive, neutral, or negative and generate a summary report highlighting key customer sentiments.

5. You are responsible for managing personal finances and have a CSV file named "expenses.csv" containing daily expense records. Build a Python application using the csv module to read and analyze the expense data. Implement functionalities such as calculating total expenses, categorizing expenses (e.g., food, transportation), and generating a budget overview with spending trends.

ANSWERS

1.A) csv

2.B) To write data into a CSV file

3.B) 'w'

4. C) writerow()

5. B) close()

6. B) writerows()

7. D) newline=None

8. D) for loop

9. C) It ensures universal newline support.

10.D) 'x' (exclusive creation mode)

11.B) Text-based

12.D) list()

13. B) writerow()

14. C) The delimiter character separates values within a CSV file.

15. B) writerows() writes multiple rows at once, while writerow() writes one row at a time.

16.A) It requires less memory compared to other modules.

17.A) It prevents empty lines from being written.

18.C) List

19.D) DictReader() reads data as dictionaries, while reader() reads data as lists.

20. B) To ensure cross-platform compatibility for newline characters.

2 marks questions

1.The csv module in Python is used to work with CSV (Comma Separated Values) files. It provides functions to read, write, and manipulate data in CSV format, making it easier to handle tabular data.

2.

Newline handling is crucial when working with CSV files because different operating systems use different newline characters (such as '\n' for Unix/Linux and '\r\n' for Windows). If newline

handling is not done correctly, it can lead to issues like extra blank lines or improper row parsing. Using newline='' in the open() function ensures universal newline support, preventing such issues.

3.

- writerow(): Writes a single row of data into the CSV file.
- writerows(): Writes multiple rows of data into the CSV file. It takes an iterable of rows (e.g., a list of lists) and writes each row as a separate line in the CSV file.

4.

A CSV file is a text-based file format used to store tabular data. Each line in a CSV file represents a row, and values within each row are separated by a delimiter character, commonly a comma (','), although other characters like tabs or semicolons can also be used.

5.

The delimiter character is used to separate values within a CSV file. It signifies where one value ends and the next one begins within a row. Common delimiter characters include commas (','), tabs ('\t'), semicolons (';'), and pipes ('|'). The choice of delimiter depends on the data and the requirements of the CSV file.

6.

```
import csv
data = ['Name', 'Age', 'City']
with open('data.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(data)
```

7.

```
import csv
try:
    with open('input.csv', 'r') as file:
        reader = csv.reader(file)
        for row in reader:
            print(row)
except FileNotFoundError:
    print("File not found.")
except Exception as e:
    print("Error:", e)
```

8.

```
import csv
import random
data = [['Name', 'Age', 'City'],
        ['Alice', 25, 'New York'],
```

```
        ['Bob', 30, 'San Francisco'],
        ['Charlie', 28, 'Los Angeles']]
with open('output.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(data)
```

9

```
import csv
new_data = [['David', 35, 'Chicago'],
            ['Emma', 29, 'Houston']]
with open('output.csv', 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(new_data)
```

10.

```
import csv
def count_rows(csv_file):
    try:
        with open(csv_file, 'r') as file:
            reader = csv.reader(file)
            row_count = sum(1 for row in reader)
            return row_count
    except FileNotFoundError:
        return 0
    except Exception as e:
        print("Error:", e)
        return 0
file_path = 'data.csv'
total_rows = count_rows(file_path)
print("Total rows in", file_path, ":", total_rows)
import csv

low_stock_items = []
with open('inventory.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        if int(row[1]) < 10:  # Assuming quantity is in the second column
            low_stock_items.append(row[0])  # Assuming item name is in the first column
```

```
print("Low stock items:", low_stock_items)
```

3 MARKS QUESTIONS ANSWERS

1.

```
import csv
low_stock_items = []
with open('inventory.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        if int(row[1]) < 10:  # Assuming quantity is in the second column
            low_stock_items.append(row[0])  # Assuming item name is in the first column
print("Low stock items:", low_stock_items)
```

2.

```
import csv
total_revenue = 0
with open('sales.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        total_revenue += float(row[2])  # Assuming Price is in the third column
print('Total revenue:', total_revenue)
```

3.

```
import csv
total_salary = 0
with open('employees.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        total_salary += int(row[2])  # Assuming Salary is in the third column
print('Total salary:', total_salary)
```

4.

```
import csv
def write_student_data(file_name):
    try:
        with open(file_name, 'w', newline='') as file:
            writer = csv.writer(file)
            writer.writerow(['Name', 'Grade'])  # Write header row
```

```
        while True:
            name = input("Enter student name (or type 'exit' to quit): ")
            if name.lower() == 'exit':
                break
            grade = input("Enter student grade: ")
            writer.writerow([name, grade])
    except Exception as e:
        print("Error:", e)
write_student_data('students.csv')
```

5.

Delimiter characters in CSV files are used to separate individual fields (data values) within a row. The most common delimiter character is a comma (,), but other characters like tabs (\t), semicolons (;), and pipes (|) can also be used.

The significance of delimiter characters when working with the **csv** module in Python is that they determine how data is organized and interpreted within a CSV file. When reading a CSV file, Python uses the specified delimiter character to split each row into individual fields, making it possible to access and process the data accordingly. Similarly, when writing data to a CSV file, the delimiter character is used to separate different values within each row. Using the correct delimiter ensures that the data is correctly formatted and can be read or written without errors.

6.

```
import csv
# Define the input and output file names
input_file = 'data.csv'
output_file = 'high_scores.csv'
# Open the input and output CSV files
with open(input_file, 'r') as file:
    reader = csv.reader(file)
    with open(output_file, 'w', newline='') as output_file:
        writer = csv.writer(output_file                    # Write header row to the output file
        header = next(reader)
        writer.writerow(header)
              # Iterate through each row in the input file
        for row in reader:
            # Check if the value in the second column is greater than 50
            if int(row[1]) > 50:  # Assuming the second column contains integers
                writer.writerow(row)
```

7.import csv

```
student_data = ['Alice', 25, 'A']
with open('output.csv', 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(student_data)
```

8.Correct code:

```
import csv
student_data = ['Alice', 25, 'A']
with open('output.csv', 'a', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(student_data)
```

output:

Total inventory cost: 500.0

9.

```
import csv
student_dict = {}
with open('students.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        name, age, grade = row
        student_dict[name] = [int(age), grade]
print(student_dict)
```

10.

```
import csv
total_sales = 0
with open('sales.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        sales_amount = float(row[2])  # Assuming sales amount is in the third column
        total_sales += sales_amount
print('Total sales amount:', total_sales)
```

5 MARKS QUESTIONS ANSWERS

1.
```
import csv
product_values = {}
with open('inventory.csv', 'r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        product = row['Product']
        quantity = int(row['Quantity'])
        price = float(row['Price'])
        total_value = quantity * price
        if product in product_values:
            product_values[product] += total_value
        else:
            product_values[product] = total_value
print("Summary Report - Total Value of Each Product:")
for product, total_value in product_values.items():
    print(f"{product}: ${total_value:.2f}")
```

2.
```
import csv
data = [['Name', 'Age', 'City'], ['Alice', 25, 'New York'], ['Bob', 30, 'San Francisco']]
with open('data.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(data)
```

3
```
import csv
product_data = [
    {'Name': 'Laptop', 'Quantity': 10, 'Price': 1200},
    {'Name': 'Mouse', 'Quantity': 50, 'Price': 20},
    {'Name': 'Keyboard', 'Quantity': 20, 'Price': 50}
]
fieldnames = ['Name', 'Quantity', 'Price']
with open('output.csv', 'w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=fieldnames)
    writer.writeheader()
    writer.writerows(product_data)
```

4.

```
import csv
product_revenue = {}
with open('sales.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)  # Skip header row
    for row in reader:
        product = row[1]  # Assuming Product is in the second column
        revenue = float(row[2])  # Assuming Revenue is in the third column
        if product in product_revenue:
            product_revenue[product] += revenue
        else:
            product_revenue[product] = revenue
print("Total Revenue Earned for Each Product:")
for product, total_revenue in product_revenue.items():
    print(f"{product}: ${total_revenue:.2f}")
```

5.

```
import csv
# Read data from sales.csv
sales_data = {}
with open('sales.csv', 'r') as sales_file:
    reader = csv.DictReader(sales_file)
    for row in reader:
        date = row['Date']
        product = row['Product']
        revenue = float(row['Revenue'])
        if product not in sales_data:
            sales_data[product] = {'Date': date, 'Revenue': revenue}
        else:
            sales_data[product]['Revenue'] += revenue
# Read data from inventory.csv
inventory_data = {}
with open('inventory.csv', 'r') as inventory_file:
    reader = csv.DictReader(inventory_file)
    for row in reader:
        product = row['Product']
        quantity = int(row['Quantity'])
```

```
        inventory_data[product] = quantity

# Combine datasets and write to combined_data.csv
fieldnames = ['Date', 'Product', 'Revenue', 'Available Quantity']
with open('combined_data.csv', 'w', newline='') as combined_file:
    writer = csv.DictWriter(combined_file, fieldnames=fieldnames)
    writer.writeheader()
    for product, data in sales_data.items():
        if product in inventory_data:
            data['Available Quantity'] = inventory_data[product]
            writer.writerow(data)
```

4 marks questions

1,

```
import csv
def generate_sales_report(csv_file):
    total_revenue = 0
    total_items_sold = 0
    sales_by_category = {}
    with open(csv_file, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            revenue = float(row['Revenue'])
            total_revenue += revenue
            items_sold = int(row['Items Sold'])
            total_items_sold += items_sold
            category = row['Product Category']
            if category in sales_by_category:
                sales_by_category[category] += revenue
            else:
                sales_by_category[category] = revenue
    print("Daily Sales Report:")
    print(f"Total Revenue: ${total_revenue:.2f}")
    print(f"Total Items Sold: {total_items_sold}")
    print("Sales by Category:")
    for category, revenue in sales_by_category.items():
        print(f"{category}: ${revenue:.2f}")
generate_sales_report('sales_data.csv')
```

2.

```
import csv
def update_student_grades(csv_file, student_name, subject, grade):
    # Read existing grades
    grades = {}
    with open(csv_file, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            name = row['Name']
            if name not in grades:
                grades[name] = {}
            grades[name][row['Subject']] = float(row['Grade'])
    # Update or add new grade
    if student_name in grades:
        grades[student_name][subject] = grade
    else:
        grades[student_name] = {subject: grade}
    # Write updated grades to file
    fieldnames = ['Name', 'Subject', 'Grade']
    with open(csv_file, 'w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=fieldnames)
        writer.writeheader()
        for name, subjects in grades.items():
            for subject, grade in subjects.items():
                writer.writerow({'Name': name, 'Subject': subject, 'Grade': grade})
def calculate_average_grade(csv_file, subject):
    total_grade = 0
    total_students = 0
    with open(csv_file, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['Subject'] == subject:
                total_grade += float(row['Grade'])
                total_students += 1
    if total_students > 0:
        average_grade = total_grade / total_students
        print(f"Average Grade in {subject}: {average_grade:.2f}")
```

```
    else:
        print("No data for this subject.")
def generate_progress_report(csv_file, student_name):
    with open(csv_file, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['Name'] == student_name:
                print(f"Progress Report for {student_name}:")
                for subject, grade in row.items():
                    if subject != 'Name':
                        print(f"{subject}: {grade}")
# Example usage:
update_student_grades('gradebook.csv', 'Alice', 'Math', 85)
calculate_average_grade('gradebook.csv', 'Math')
generate_progress_report('gradebook.csv', 'Alice')
```

3.

```
import csv
def track_inventory(csv_file, reorder_level):
    low_stock_items = []
    restocking_list = {}
    with open(csv_file, 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            name = row['Name']
            quantity = int(row['Quantity'])
            if quantity < reorder_level:
                low_stock_items.append(name)
                restocking_list[name] = reorder_level - quantity

    print("Low Stock Items:")
    for item in low_stock_items:
        print(item)
    print("Restocking List:")
    for item, quantity in restocking_list.items():
        print(f"{item}: {quantity}")
# Example usage:
track_inventory('inventory.csv', 10)
```