PYTHON REVISION TOUR (12 Marks)

Python is a widely used high-level programming language for general-purpose programming, created by Guido Van Rossum and first released in 1991. My first program in Python is:

print("Hello World")

Python shell can be used in two ways, viz., interactive mode and script mode.

TOKENS IN PYTHON (Lexical Unit)

Smallest individual unit in a program.

1. KEYWORDS

Predefined words with special meaning to the language processor; **reserved for special purpose** and must not be used as normal identifiers names. Example - True, for, if, else, elif, from, is, and, or, break, continue, def, import etc.

2. IDENTIFIERS (1 mark)

Names given to different parts of the program viz. variable, objects, classes, functions, lists, dictionaries and so forth. The naming rules:

- Must only be a non-keyword word with no space in between. Example: salary, maxmarks
- Must be made up of only letters, numbers and underscore (_). Ex: _salary, _cs
- Can not start with a number. They can contain numbers. Ex: exam21

Some valid identifiers are:

Myfile	MYFILE	Salary2021	_Chk

Invalid Identifies are:

My.file	break	2021salary	if

3. LITERALS

Data items that have a fixed value.

1. String Literals: are sequence of characters surrounded by quotes (single, double or triple)
   S1="Hello" S2="Hello" S3="""Hello"""
2. Numerical Literals: are numeric values in decimal/octal/hexadecimal form.
   D1=10 D2=0o56 (Octal) D3=0x12 (Hex)
3. Floating point literals: real numbers and may be written in fractional form (0.45) or exponent form (0.17E2).
4. Complex number literal: are of the form a+bJ, where a, b are int/floats and J(or i) represents $\sqrt{-1}$ i.e. imaginary number. Ex: C1=2+3j
5. Boolean Literals: It can have value True or False. Ex: b=False
6. Special Literal None: The None literal is used to indicate absence of value. Ex: c=None

4. OPERATORS (3 marks)

Tokens that trigger some computation/action when applied to variables and other objects in an expression. These are

Arithmetic	: +, -, *, /, %, **, //
Bitwise	: &, ^, |
Identity	: is, is not
Relational	: >, >=, <, <=,!=, ==
Logical	: and, or
Assignment	: =
Membership	: in, not in

Arithmetic assignment: /=, +=, -=, *=, %=, **=, //=

PRECEDENCE OF ARITHMETIC OPERATORS

PE(MD) (AS) = read PEMDAS

P=parenthesis () → E=exponential →M, D = multiplication and division → A, S=addition and subtraction

Ex: 12*(3%4)//2+6 = 24 (Try at your end)

5. PUNCTUATORS

Punctuators are symbols that are used to organize sentence structure. Common punctuators used in Python are: ' " # \ ( ) [ ] { } @ , : .

DATA TYPES

Means to identify type of data and set of valid operations for it. These are

1. NUMBERS

To store and process different types of numeric data: Integer, Boolean, Floating-point, Complex no.

2. STRING

Can hold any type of known characters i.e. letters, numbers, and special characters, of any known scripted language. It is immutable datatype means the value can't be changed at place. Example: "abcd", "12345","This is India", "SCHOOL".

-6	-5	-4	-3	-2	-1
S	C	H	O	O	L
0	1	2	3	4	5

3. LISTS (2 marks)

Represents a group of comma-separated values of any datatype between square brackets. Examples:

```
L1=list()
L2=[1, 2, 3, 4, 5]
L3=[5, 'Neha', 25, 36, 45]
L4=[45, 67, [34, 78, 87], 98]
```

In list, the indexes are numbered from 0 to n-1, (n= total number of elements). List also supports forward and backward traversal. List is **mutable datatype** (its value can be changed at place).

4. TUPLE (1 mark)

Group of comma-separated values of any data type within parenthesis. Tuple are **immutable** i.e. non-changeable; one cannot make change to a tuple.

```
t1=tuple()
t2=(1, 2, 3, 4, 5)
t3=(34, 56, (52, 87, 34), 67, 'a')
```

5. DICTIONARY (1 mark)

The dictionary is an unordered set of comma separated key:value pairs, within { }, with the requirement that within a dictionary, no two keys can be the same. Following are some examples of dictionary:

```
        d1=dict()
        d2={1:'one', 2:'two', 3:'three'}
        d3={'mon':1, 'tue':2, 'wed':3}
        d4={'rno':1, 'name':'lakshay', 'class':11}
```

In dictionary, key is immutable whereas value is mutable.

STRING PROCESSING AND OTHER FUNCTIONS

STRING REPLICATION

When a string is multiplied by a number, string is replicated that number of times.

Note: Other datatype – list and tuple also show the same behavior when multiplied by an integer number.

```
>>> STR='Hi'
>>> print(STR*5)
>>>HiHiHiHiHi
```

```
>>> L=[1, 2]
>>> print(L*5)
>>> [1,2,1,2,1,2,1,2,1,2]
```

```
>>> t=(1, 2)
>>> print(t*5)
>>> (1,2,1,2,1,2,1,2,1,2)
```

STRING SLICE (2 marks)

String Slice refers to part of a string containing some contiguous characters from the string. The syntax is **str_variable[start:end:step]**.

start: from which index number to start

end: upto (end-1) index number characters will be extracted

step: is step value. (optional) By default it is 1

Example : Suppose S='KENDRIYA'

Backward Index no	-8	-7	-6	-5	-4	-3	-2	-1
Character	K	E	N	D	R	I	Y	A
Forward Index no	0	1	2	3	4	5	6	7

Command	Output	Explanation
S	KENDRIYA	Will print entire string.
S[:]	KENDRIYA	Will print entire string.
S[::]	KENDRIYA	Will print entire string.
S[2]	N	Will print element at index no 2.
S[2:6]	NDRI	From index number 2 to (6-1) i.e. 5.
S[1:6:2]	EDI	From index 1 to 5 every 2nd letter.
S[::2]	KNRY	From start to end … every 2nd letter.
S[::3]	KDY	From start to end … every 3rd letter.
S[::-1]	AYIRDNEK	Will print reverse of the string.
S[: -1]	KENDRIY	Will print entire string except last letter
S[-1]	A	Will print only last letter
S[-5::]	DRIYA	From index no -5 to end of the string.
S[-7:-4:]	END	From index no -7 to -5.

OTHER FUNCTIONS

length()	istitle()
startswith()	endswith()
count(word)	isspace()
swapcase()	find()
lstrip()	rstrip()
strip()	

LIST PROCESSING AND OTHER OPERATIONS

CREATING LIST

1) Empty List

```
>>>L=[]        OR      >>> L=list()
```

2) From existing sequence

```
>>>st='HELLO'
>>> L1=list(st)           # from a string
>>> t=(1, 2, 3)
>>> L2=list(t)           # from a tuple
```

3) from Keyboard entering element one by one

```
>>> L=[]
>>> for i in range(5):
        n=int(input('Enter a number :'))
        L.append(n)
```

TRAVERSING A LIST

```
>>> L=[1, 2, 3, 4, 5]
>>> for n in L:
        print(n)
```

JOINING LISTS

```
>>> L1=[1, 2, 3]
>>> L2=[33, 44, 55]
>>> L1 + L2            # output: [1, 2, 3, 33, 44, 55]
```

SLICING THE LIST (1 mark)

```
>>> L=[10, 11, 12, 13, 14, 15, 16, 18]
>>> seq=L[2:6]
```

```
# it will take from index no 2 to (6-1) i.e. 5
>>> seq                # output: [12, 13, 14, 15]
```
General syntax for slicing is L[start : end : step]

APPENDING ELEMENT

```
>>> L=[10, 11, 12]
>>>L.append(20)       # Will append 20 at last
>>> L                  # output : [10, 11, 12, 20]
>>>L.append([4,5])
# Appended data will be treated as a single element
>>> L          # output: [10, 11, 12, 20, [4, 5]]
```

EXPANDING LIST

```
>>> L=[10, 11, 12]
>>>L.extend([44,55])
# Will extend given item as separate elements
>>> L                  # output: [10, 11, 12, 44, 55]
```

UPDATING LIST

```
>>> L=[10, 11, 12]
# Suppose we want to change 11 to 50
>>> L[1]=50
>>> L                  # output: [10, 50, 12]
```

DELETING ELEMENTS

del List[index]# Will delete the item at given index
del List[start : end]# Will delete from index no start to (end-1)

```
>>> L=[x for x in range(1,11)]
>>> L          # list is [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> del L[2:5]
>>> L          # now list is [1, 2, 6, 7, 8, 9, 10]
```

OTHER FUNCTIONS (1 mark)

len()	clear()	sum(List)
pop(index)	remove()	min(List)
insert(pos, value)	sort()	max(List)
index(value)	pop()	count()

TUPLE Vs LISTS

Tuples (is immutable)	List (Mutable)
We can not change value in place.	we can change the value in place

DICTIONARY

Dictionaries are a collection of some unordered key:value pairs; are indexed by keys (keys must be of any non-mutable type).

TRAVERSING A DICTIONARY

```
>>> d={1:'one', 2:'two', 3:'three'}
>>> for key in d:
            print(key, ":", d[key], sep='\t')
1 : one          2 : two          3 : three
```

ADDING ELEMENT TO DICTIONARY

```
>>>d[4]='four'          # will add a new element
>>> print(d)
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

DELETING ELEMENTS FROM DICTIONARY

```
>>> del d[3]
# will delete entire key:value whose key is 3
>>> print(d)    # op: {1: 'one', 2: 'two', 4: 'four'}
```

OTHER FUNCTIONS

Here is a list of functions to recall them which work on dictionaries.

pop(key)	len()	values()
items()	keys()	update()

WORKING WITH FUNCTIONS

Block of statement(s) for doing a specific task.
Advantages of function:

- Cope-up with complexity
- Hiding details – Abstraction
- Fewer lines of code - lessor scope for bugs
- Increase program readability
- Re-usability of the code

DEFINING AND CALLING FUNCTION

In python, a function may be declared as:

```
def function_name([Parameter list]) :
        """Doc-string i.e. documentation of fn"""
        Body of the function
        return value
```

Remember: **Implicitly, Python returns None**, which is also a built-in, immutable data type. Ex:

```
>>> def f(a,b):
        """demo of a function"""        #doc-string
        c=a+b
        return c
>>>f(10, 20)            # calling of function
```

def is a keyword to declare a function
f is the name of the function
a, b – parameter/argument of the function
return is a keyword

Functions can be categorized in three types:
1. Built-in function   2. Modules   3. User-defined

BUILT-IN FUNCTION

- Pre-defined functions available in Python.
- Need not import any module (file).
- Example of some built-in functions are:
  str(), eval(), input(), min(), max(), abs(), type(), len(), round(), range(), dir(), help()

MODULES

- Module in Python is a file (name.py) which contains the definition of variables and functions. A module may be used in other programs. We can use any module in two ways
  1. import <module-name>
  2. from <module-name> import <func-name>

Some well-known modules are: math, random, matplotlib, numpy, pandas, datetime

USER-DEFINED FUNCTIONS

- UDFs are the functions which are created by the user as per requirement.
- After creating them, we can call them in any part of the program.
- Use 'def' keyword for UDF.

ACTUAL AND FORMAL PARAMETER/ ARGUMENT

Formal Parameter appear in function definition
Actual Parameter appear in function call statement.
Example:

```
def findSum(a, b):      # a, b formal parameter
     print('The sum is ',(a+b))
#main-program
X, Y=10,30
findSum(X, Y)           # X, Y are actual parameter
```

VARIABLE SCOPE

- Arguments are local
- Variable inside functions are local
- If variable is not defined , look in parent scope
- Use the global statement to assign to global variables

TYPES OF ARGUMENTS

Python supports four types of arguments:

1. Positional	2 Default
3. Keyword	4. Variable Length

1. POSITIONAL ARGUMENTS

Arguments passed to function in correct positional order. If we change the position of their order, then the result will change.

```
>>> def subtract(a, b):
        print(a - b)
>>> subtract(25, 10)        # output: 15
>>> subtract(10, 25)        # output: - 15
```

2. DEFAULT ARGUMENTS

To provide default values to **positional arguments**; If value is not passed, then default values used.
Remember - If we are passing default arguments, then do not take non-default arguments, otherwise it will result in an error.

```
def findSum(a=30, b=20, c=10):
    print('Sum is ',(a+b+c))
'''
def mySum(a=30, b=20, c):
  print('Sum is ',(a+b+c))
#Error that non-default argument follows default arguments
'''
#main-program
p, q, r =100, 200, 300
findSum(p,q,r)       #output - Sum is 600
findSum(p,q)         #output - Sum is 310
findSum(r)           #output - Sum is 330
findSum(q,r)         #output - Sum is 510
findSum()            #output - Sum is 60
```

3. KEYWORD ARGUMENTS

we can pass arguments value by keyword, i.e. by passing parameter name to change the sequence of arguments, instead of their position. Example:

```
def print_kv(kvnm, ronm):
  print(kvname,'comes under',roname,'region')

#main-program
print_kv(kvnm='Rly Gandhidham', ronm='Ahmd')
print_kv(ronm='Jaipur', kvnm='Bharatpur')
```

4. VARIABLE LENGTH ARGUMENTS

In certain situations, we can pass variable number of arguments to a function. Such types of arguments are called variable length argument. They are declared with * (asterisk) symbol.

```
def findSum(*n):
  sm=0
  for i in n:
    sm=sm+i
  print('Sum is ',sm)
#main-program
findSum()              #output – Sum is 0
findSum(10)            #output – Sum is 10
findSum(10,20,30)      #output – Sum is 60
```

PASSING STRINGS TO FUNCTION

```
def countVowel(s):
  vow='AEIOUaeiou'
  ctr=0
  for ch in s:
    if ch in vow:
      ctr=ctr+1
  print('Number of vowels in', s ,'are', ctr)
#main-program
data=input('Enter a word to count vowels :')
countVowel(data)
```

PASSING LIST TO FUNCTION

```
def calAverage(L):
  sm=0
  for i in range(len(L)):
    sm=sm+L[i]
  av=sm/len(L)
  return av
#main-program
marks=[25, 35, 32, 36, 28]
print('Marks are :',marks)
avg=calAverage(marks)
print('The average is', avg)
```

PASSING TUPLES TO FUNCTION

```
def midPoint(t1, t2):
   m=((t1[0]+t2[0])/2.0,
(t1[1]+t2[1])/2.0)
   return m
#main-program
p1=(2, 4)
p2=(6, 6)
mp=midPoint(p1,p2)
print('The Mid-Point of is', mp)
```

PASSING DICTIONARY TO FUNCTION

```
def printMe(d):
   """Function to print values of given
dictionary"""
   for key in d:
      print(d[key])

#main-program
d={1:'mon', 2:'tue', 3:'wed' }
printMe(d)
```

FUNCTIONS USING LIBRARIES

Math library (import math) (1 mark)

Function	Description	Example
pi	value of pi	3.14159
ceil(x)	integer >=x	ceil(1.2) → 2.0
floor(x)	integer <=x	floor(1.2) → 1.0
pow(x,y)	x raise y	pow(3,2) →9
sqrt(n)	square root	sqrt(2) → 1.414

String library (1 mark)

capitalize()	Convert 1st letter to upper case
index(ch)	Return index-no of 1st occurrence
isalnum()	True, if entire string is (a-z, A-Z, 0-9)
isalpha()	True, if entire string is (a-z, A-Z)
isdigit()	True, if entire string is (0-9)
islower()	True, if entire string is in lower
isupper()	True, if entire string is in upper
len()	Return length of the string

Random library (import random) (2 Marks)

random()	Return floating value between 0 and 1.
randrange(0,N)	This generates integer number from 0 to (N-1).
randint(a, b)	Return any number b/w given number a and b including them
uniform(a, b)	Return floating number b/w given numbers a and b.

EXERCISE TO UNDERSTAND THE CONCEPTS WITH SOLUTION

Sno	Question	Answer
1	Find the invalid identifier(s) from the following: a) MyName b) True c) 2ndName d) My_Name	b) True, as it is a keyword c) 2ndName, Because it is starting with a digit.
2	Given the lists L=[1, 3, 6, 82, 5, 7, 11, 92], write the output of print(L[2:5]).	[6,82,5], as it will start from index no 2 to (5-1) i.e. 4.
3	Identify the valid arithmetic operator in Python from the following: a) ? b) < c) ** d) and	c) ** as it is used to find power
4	Suppose tuple T is T = (10, 12, 43, 39), Find incorrect? a) print(T[1]) b) T[2] = -29 c) print(max(T)) d) print(len(T))	b) T[2]= -29 (as tuple is immutable and we can't change its value)
5	Declare a dictionary Colour, whose keys are 1, 2, 3 and values are Red, Green and Blue respectively.	Colour={1:'Red', 2:'Green', 3:'Blue'}
6	A tuple is declared asT = (2, 5, 6, 9, 8) What will be the value of sum(T)?	It will give the sum of all elements of tuple i.e. 2+5+6+9+8 = 30
7	Name the built-in mathematical function / method that is used to return an absolute value of a number.	abs()
8	Identify declaration of L = ['Mon','23','Bye', '6.5'] a) dictionary b) string c) tuple d) list	d) list, as a list is collection of heterogeneous data.
9	Find the output of the following code? >>>name="ComputerSciencewithPython" >>>print(name[3:10])	It will print string from index no 3 to (10-1) i.e. 9 means 7 characters. So outout will be **puterSc**

10	Write the full form of IDLE.	Integrated Development Learning Environment
11	Find the output: >>>A = [17, 24, 15, 30] >>>A.insert(2, 33) >>>print (A[-4])	A.insert(2,33) will insert 33 at index no 2. Now the list is [17, 24, 33, 15, 30]. So print(A[-4]) will start counting from last element starting from -1, -2… Hence will give 24.
12	Name the Python Library modules which need to be imported to invoke the following functions: (a) ceil() (b) randrange()	(a) math (b) random
13	What will be the result of the following code? >>>d1 = {"abc" : 5, "def" : 6, "ghi" : 7} >>>print (d1[0]) (a) abc (b) 5 (c) {"abc":5} (d) Error	(d) Error, because dictionary works on the principle of key:value. These is no key as 0, so it will produce an error.

14	STR="VIBGYOR"
	colors=list(STR)
	>>>del colors[4]
	>>>colors.remove("B")
	>>>colors.pop(3)
	>>>print(colors)
		It will create a list as colors=['V', 'I', 'B', 'G', 'Y', 'O', 'R']
		del colors[4] will delete the element at index no 4 i.e. so list will be ['V', 'I', 'B', 'G', 'O', 'R']. colors.remove("B") will remove 'B'. so list will be ['V', 'I', 'G', 'O', 'R']. colors.pop(3) will extract 'O'. So finally colors will be ['V', 'I', 'G', 'R'].

15	Suppose list L is declared as L = [5 * i for i in range (0,4)], list L is a) [0, 1, 2, 3,] b) [0, 1, 2, 3, 4] c) [0, 5, 10, 15] d) [0, 5, 10, 15, 20]	It is List Comprehension. Expression L = [i for i in range (0,4)] will generate [0, 1, 2, 3]. Since here we are writing 5*i, so correct answer will be c) [0, 5, 10, 15]

EXERCISE TO UNDERSTAND THE CONCEPTS WITH SOLUTION (3 Marks)

Sno	Question	Answer

1	Find possible o/p (s) at the time of execution of the program from the following code? Also specify the maximum values of variables Lower and Upper.
	import random as r
	AR=[20, 30, 40, 50, 60, 70];
	Lower =r.randint(1,3)
	Upper =r.randint(2,4)
	for K in range(Lower, Upper +1):
	    print (AR[K],end="#")
	(i) 10#40#70# (ii) 30#40#50#
	(iii) 50#60#70# (iv) 40#50#70#
		Lower = r.randint(1,3) means Lower will have value 1,2, or 3
		Upper =r.randint(2,4) means Upper will have value 2, 3, or 4
		So K will be from (1, 2, 3) to (2, 3, 4)
		Means if K=1, then upper limit (2,3,4)
		If K=2, then upper limit (2,3,4)
		If K=3, then upper limit (2,3,4)
		
		So correct answer (ii) 30#40#50#

2	Write a function LShift(Arr,n), which accepts a list Arr of numbers and n is a numeric value by which all elements of the list are shifted to left.
	Sample Input Data of the list
	Arr= [10,20,30,40,12,11], n=2
	Output:
	Arr = [30,40,12,11,10,20]
		def LShift(Arr, n):
		  L=Arr[n:] + Arr[:n]
		  return L
		Arr= [10,20,30,40,12,11]
		n=2
		L1=LShift(Arr,n)
		print(L1)

3	Write a function in python named SwapHalfList(Array), which accepts a list Array of numbers and swaps the elements of 1st Half of the	def SwapHalfList(Array):
		   mid=len(Array)//2
		   if sum(Array[:mid]) > sum(Array[mid:]):
		      print(Array[mid:] + Array[:mid])

	list with the 2nd Half of the list ONLY if the sum of 1st Half is greater than 2nd Half of the list.
	Sample Input Data of the list:
	Array= [100, 200, 300, 40, 50, 60],
	Output Arr = [40, 50, 60, 100, 200, 300]
		`    else:`
		`        print(Array)`
		`Array= [ 100, 200, 300, 40, 50, 60]`
		`SwapHalfList(Array)`

4	Write a function listchange(Arr) in Python, which accepts a list Arr of numbers, the function will replace the even number by value 10 and multiply odd number by 5 .
	Sample Input Data of the list is:
	a=[10, 20, 23, 45]
	listchange(a)
	output : [10, 10, 115, 225]
		`def listchange(Arr):`
		`    for i in range(len(Arr)):`
		`        if Arr[i]%2==0:`
		`            Arr[i]=10`
		`        else:`
		`            Arr[i]=Arr[i]*3`
		`    print(Arr)`
		`a=[10, 20, 23, 45]`
		`listchange(a)`

5	Write a function HowMany(ID, VALUE) to count and display number of times the VALUE is present in the list ID. For example, if the ID contains [115, 25, 65, 59, 74, 25, 110, 250] and the VALUE contains 25, the function should print: **25 found 2 times.**	`def HowMany(ID, VALUE):`
		`    ctr=ID.count(VALUE)`
		`    print(VALUE,'found',ctr,'times')`
		`List=[115, 25, 65, 59, 74, 25, 110, 250]`
		`Value=25`
		`HowMany(List, Value)`

DATA FILE HANDLING: 12 MARKS (TEXT FILE-3,CSV-4,BINARY-5)+PRACTICAL-7

1. **File:** A file is a named location on a secondary storage media like HDD where data are permanently stored for reusability.
   Need of File Handling: To store data in secondary storage for reusability. To access the data fast.
   To perform different operations on file likes: open, read, write, search, update, close etc.
2. **Types data files: (i) Text File (ii) Binary File (iii) csv file**
   Text File:A text file consists of human readable characters. Each line of a text file terminated By a special character called the end of line (EOL). Default EOL used newline (\n).
   Binary file: It made up of non-human readable characters and symbols. They represent the actual content such as image, audio, video, exe file.
   CSV File: A CSV (Comma Separated Value) format is one of the simplest and common way to store tabular data. To represent a csv file, it must be saved with .csv extension. Comma is also a delimiter which separates two files in a row

3 Modes of files:

Text file mode	**Binary File Mode**	**CSV File Mode**	**Description**
'r'	'rb'	'r'	By default Read. Opens a file for reading, error if the file does not exist.
'w'	'wb'	'w'	Write - Opens a file for writing, creates the file if it does not exist
'a'	'ab'	'a'	Append - Opens a file for appending, creates the file if it does not exist
'r+'	'rb+'		Read and Write-File must exist, otherwise error is raised.
'w+'	'wb+'	Read and Write-File is created if does not exist.	
'a+'	'ab+'	Read and write-Append new data	
'x'	'xb'	Create - Creates the specified file, returns an error if the file exists	

Opening and closing a file: To open a file we use open() method and to close file, we use close() method.
Standard syntax for open a file in two way: (i) file_object=open("file name",mode name)
(ii) with open("file name",access mode) as file object:

Close a file: file_object.close()
Example: f = open("book.txt",'r') where f is file object/file handle and open() is method, name of file- book.txt mode-read(by default file open in read mode indicated by 'r') another way to open file is :-
With open("book.txt",'r') as f :
Modules required in files:
(i) os module: we used for remove(), rename() and getcwd() methods.
remove(): To delete a file, **rename():** To rename a file, **getcwd():** To find the current directory
(ii) pickle module: we used in binary file for dump() and load() method respectively to write into a file and read contents from the file.
Pickle module is used for serializing/pickling and de-serializing/unpickling any python object.
Pickling- It will convert python object into byte stream and unpickling convert byte stream into python Object.
dump() method: It used to convert(pickling) python objects for writing data in a binary file.
Syntax: module name.dump(dataobject,file object)
load(): It is used to load/read(unpickling) data from a file. Syntax:
storeobject=modulename.load(file object).
(iii) csv module: - we used in csv file for reader(),writer(),writerow(),writerows() methods.
reader(): function is used to read the file, which returns an iterable reader object.
The reader object is then iterated using for loop to print the content of each row.
csv.reader(file object) function in default mode of csv file having comma delimiter.
writer (): The csv.writer(file object) function returns a writer object that converts the user's data into delimiter string.
Writerow(): The string can later be used to write into CSV File using
storeobeject.writerow( ) function.
Writerows(): If we need to write the content of 2-Dimensional list into csv file, instead of using writerow() function many times, we can write storeobject.**writerows()** func.
Note: syntax for accessing modules and methods respectively: import module name and Objectname=module name.method name()
Text file Methods: write (): writing a single string and returns number of characters being written.
writeline(): writing a sequence of(multiple)strings to a file and does not return the number of characters written in the file.(Both functions use as fileobject.write(single string/multiple string)
read (): It is used to read a specified number of bytes of data from a data file**.** fileobject.read(n) where **n-**no of bytes read, if we do not pass no argument or a negative number is specified in read () then it read entire content of file.
readline([n]): It will read one complete line from a file where line terminate with (\n). if we pass as an argument n then it read a specified number of bytes. If no argument pass or negative number pass it read complete line. To read the entire file line by line using the readline() we use a for loop.
readlines(): It reads all the lines and returns the lines along with newline as a list of strings.
split (): If we want to display each word of a line separately as an element of a list.
splitlines(): It is used instead of split(), then each line is returned as element of a list.
SETTING OFFSETS IN A FILE: To access the data in random fashion then we use seek () and tell () Method. **tell ():** It returns an integer that specifies the current position of the file object in the file. fileobject.tell() and seek(): It is used to position the file object at a particular position in a file. fileobject.seek(offset [, reference point]) where offset is the number of bytes by which the file object is to be moved and reference point indicating the starting position of the file object. Values of reference point as 0- beginning of the file, 1- current position of the file, 2- end of file.
Q1. What is full form of csv? Which delimiter used by default?
Ans: CSV- Comma Separated Value and comma delimiter used by default.
Q2. What is difference between read('r') and write('w') mode of file?
Ans: Read mode: - Opens a file for reading, error if the file does not exist. Write mode: Opens a file for writing, creates the file if it does not exist.
Q3. Which modules used in csv and binary file?
Ans: In csv file- csv module and binary file: pickle module.

Q4. Which method is used to specify current position and particular position of file object respectively?
Ans: tell ()-specify current position of file object and seek ()- specify the particular position of file object.

NOTE: TEXT FILE SOLUTION STEPS

Def of fun -> Open file -> Use method as per Ques -> Loop -> Condition -> Print -> Close File

BINARY FILE SOLUTION STEPS

Pickle Module -> Def of fun -> Open file -> Inputs -> Store Obj -> dump -> Close File

Q1. 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. And display the Total count separately.

For example: if REPORT.TXT consists of
"ENTRY LEVEL OF PROGRAMMING CAN BE LEARNED FROM USEFUL FOR VARIETY OF USERS."
Then, Output will be: No. of Lines with E: 1
No. of Lines with T: 1

Solution:

```
def countlines_et():
   f=open("report.txt",'r')
   lines=f.readlines()
   linee=0
   linet=0
   for i in lines:
      if i[0]=='E':
         linee+=1
      elif i[0]=='T':
         linet+=1
   print("No.of Lines with E:",linee)
   print("No.of Lines with T:",linet)
countlines_et()
```

OR

Write a method/function show_todo():in python to read contents from a text file abc.txt and display those lines which have occurrence of the word " TO" or "DO"
For example: If the content of the file is
"THIS IS IMPORTANT TO NOTE THAT SUCCESS IS THE RESULT OF HARD WORK WE ALL ARE EXPECTED TO DO HARD WORK. AFTER ALL EXPERIENCE COMES FROM HARDWORK."
The method/function should display"
THIS IS IMPORTANT TO NOTE THAT SUCCESS IS THE RESULT OF HARD WORK.
WE ALL ARE EXPECTED TO DO HARD WORK.

Solutions:

```
def show_todo():
   f=open("abc.txt",'r')
   lines=f.readlines()
   for i in lines:
      if "TO" in i or "DO" in i:
         print(i)
show_todo()
```

Q2. Write a function that counts the number of "Me" or "My" words present in a text file" story1.txt.
If the "story1.txt" content are as follows:
My first book was Me and My Family. It gave me chance to be known to the world.
The output of the function should be:
Count of Me/My in file: 4

Solution:

```
def displayMeMy():
   num=0
   f=open("story1.txt","rt")
   N=f.read()
   M=N.split()
   for x in M:
      if x=="Me" or x== "My":
         print(x)
         num=num+1
   f.close()
   print("Count of Me/My in file:",num)
displayMeMy()
```

OR

Write a function count_A_M(), which should read each character of a text file STORY.TXT, should count and display the occurrence of alphabets A and M (including small cases a and m too).
Example: If the file content is as follow:
Updated information As simplified by official websites.
The count_A_M():function should display the output as:
A or a:4 M or m:2

Solution:

```
def count_A_M():
   f=open("story1.txt","r")
   A,M=0,0
   r=f.read()
   for x in r:
      if x[0]=="A" or x[0]=="a" :
         A=A+1
      elif x[0]=="M" or x[0]=="m":
         M=M+1
   f.close()
   print("A or a: ",A,"\t M or m: ",M)
```

count_A_M()

Q3. Consider a binary file stock.dat that has the following data: OrderId, MedicineName,Qty and Price of all the medicines of wellness medicos, write the following functions:
a)AddOrder() that can input all the medicine orders.
b)DisplayPrice() to display the details of all the medicine that have Price.
Solution:

```
import pickle
def AddOrder():
   f=open("Stock.dat",'ab')
   OrderId=input("Enter Order Id")
   MedicineName=input("Enter Medicine Name")
   Qty=int(input("Enter Quantity:"))
   Price=int(input("Enter Price:"))
   data=[OrderId,MedicineName,Qty,Price]
   pickle.dump(data,f)
   f.close()
AddOrder()
def DisplayPrice():
   f=open("Stock.dat",'rb')
   try:
      while True:
         data=pickle.load(f)
         if data[3]>500:
          print(data[0],data[1],data[2],data[3],sep="\t")
  except:
          f.close()
DisplayPrice()
```

Q5. Create a binary file funandfood.dat that can store details of rides such as Ticketno, Ridename, No_ofpersons, and price with the help of AddRides() function and write another python function display Total to display total amount of each ticket. Also count total number of tickets sold.
Solution:

```
import pickle                    # to use binary file
def AddRides():
   f=open("funandfood.dat",'ab')
   Ticketno=input("Enter Ticket Number")
   RideName=input("Enter The Name of Ride")
   No_ofperson=int(input("Enter no of Persons"))
   Price=int(input("Enter Price:"))
   data=[Ticketno,RideName,No_ofperson,Price]
   pickle.dump(data,f)
   f.close()
AddRides()
def Display Total():
   f=open("funandfood.dat",'rb')
   total=0
   count=0
```

Q4. A binary file "Book.dat" has structure [BookNo, Book_Name, Author, Price].
a.Write a user defined function CreateFile() to input data for a record and add to Book.dat.
b. Write a function CountRec(Author) in Python which accepts the Author name as parameter and count and retum number of books by the given Author are stored in the binary file "Book.dat".

```
import pickle
def CreateFile():
   fobj=open("Book.dat","ab")
   BookNo=int(input("Book Number : "))
   Book_name=input("Name :")
   Author = input("Author: ")
   Price = int(input("Price : "))

rec=[BookNo,Book_Name,Author,Price]
   pickle.dump(rec,fobj)
   fobj.close()
def CountRec(Author):
   fobj=open("Book.dat","rb")
   num = 0
try:
   while True:
      rec=pickle.load(fobj)
      if Author==rec[2]:
         num = num + 1
except:
   fobj.close()
   return num
```

Q6. 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. The function will calculate & display the Average of the ScoredMarks of the passed Subject on screen.
Solution:

```
def DisplayAvgMarks(sub):
   f=open("TEST.dat","rb")
   count=0
   sum=0
   try:
```

```
   try:
      while True:
         data=pickle.load(f)
         total=data[2]*data[3]
print(data[0],data[1],data[2],data[3],total,sep="\t")
         count=count+1
   except:
      f.close()
   print("Total number of Tickets sold are:",count)
DisplayTotal()
```

```
      while True:
         pos=f.tell()
         rec=pickle.load(f)
         if rec[1]==sub:
            sum+=rec[3]
            count+=1
   except:
      f.close()
   print("AVG Marks:",sum/count)
DisplayAvgMarks(sub)
```

Q7. Abhisar is making a software on “Countries & their Capitals” in which various records are to be stored/retrieved in CAPITAL.CSV data file. It consists of some records. He has written the following code. As a programmer, you have to help him to successfully execute the program.

```
import_______           # Statement-1
def AddNewRec(Country,Capital):          # Fn. to add a new record in CSV file
    f=open(“CAPITAL.CSV”,_________)        #  Statement-2  fwriter=csv.writer(f)
    fwriter.writerow([Country,Capital])
    f.__          #Statement-3
defShowRec(): # Fn. to display all records from CSV file with open(“CAPITAL.CSV”,”r”) as NF:
    NewReader=csv._(NF)   #Statement-4 for rec in NewReader:
    print(rec[0],rec[1])
AddNewRec(“INDIA”,”NEW   DELHI”)
AddNewRec(“CHINA”,”BEIJING”)
ShowRec()     #Statement-5
```

1) Name the module to be imported in Statement-1. 2) Write the file mode to be passed to add new record in Statement-2. 3) Fill in the blank in Statement-3 to close the file. 4) Fill in the blank in Statement-4 to read the data from a csv file. 5) Write the output which will come after executing Statement-5.	1) csv module 2) ‘a’ -append mode. 3) f.close() 4) reader() 5) INDIA NEW DELHI CHINA BEIJING

DATA STRUCTURE – STACK (4 Marks) + (7 Marks in Practical)

A stack is a linear list implementation in LIFO – Last In First Out manner where insertions and deletions are restricted to occur only at one end – Stack's top. The technical terms for insertion-in-a-stack and deletion-from-stack are push and pop respectively. Here is a complete program of stack operations:

```
#-------------
maxsize=5
top=-1
#-------------

def pop(stack):
   global top
   if (isEmpty(stack)):
      print("\n Stack is UnderFlow \n")
   else:
      n=stack[top]
      stack.pop()
      print("Removed Element ",n)
      top=top-1


def push(stack):
   global top
   if (isFull(stack)):
      print("\n Stack is OverFlow \n ")
   else:
      n=int(input("\nEnter an element to push :"))
      top=top+1
      stack.append(n)

def traverse(stack):
   if (isEmpty(stack)):
      print("Stack is Empty ")
   else:
      for i in stack:
         print(i,end="  ")


def peak(stack):
   global top
   return stack[top]
```

```
def isFull(stack):
   global maxsize
   if (top==maxsize-1):
      return True
   else:
      return False


def isEmpty(stack):
   if (top==-1):
      return True
   else:
      return False

#********** Main Program ************
stack=[]
a=True
while a:
   print("\n1. Stack Push Operation ")
   print("2. Stack Pop Operation ")
   print("3. Show Peak / Top Position ")
   print("4. Traverse / Show Stack ")
   print("5. Exit ")
   ch=int(input("Enter Choice :"))
   if ch == 1:
      push(stack)
   elif ch == 2:
      pop(stack)
   elif ch == 3:
      print("\n Peak Position ",peak(stack))
      print('Top is ', top)
   elif ch == 4:
      traverse(stack)
   elif ch == 5:
      a=False
   else:
      print('Please enter a valid choice ')
```

UNIT 2 : COMPUTER NETWORKS [10 MARKS]

NETWORK	Interconnection of two more devices / people / things
Social Network Mobile Network	Computer Network Railway Network Airlines Network
COMPUTER NETWORK	Interconnection of two or more computers / computing devices.
EVOLUTION OF NETWORKING	Research project of US Department of Defence to connect academic and research institutes - called ARPANET
In 1980s high capacity network setup by NSF to connect more institutes.	Early 1990s, ARPANET and NSF Net along with some other networks joined to form the INTERNET

$$\begin{tikzcd}\text{1969 ARPANET} \arrow[r] & \text{1980s NSFnet} \arrow[r] & \text{1990s Internet}\end{tikzcd}$$

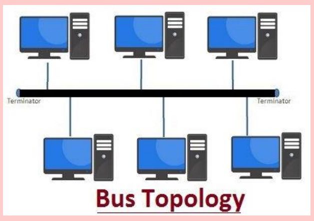
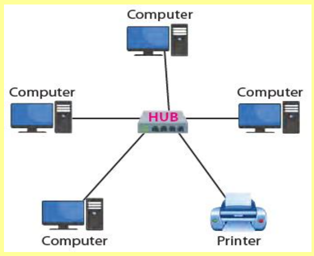
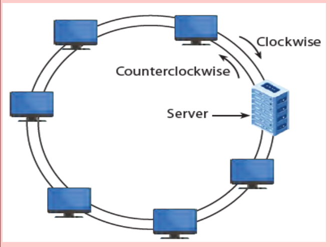
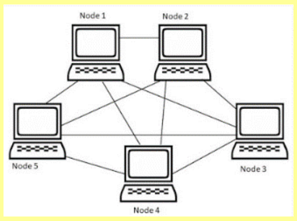
ARPANET – Advanced Research Project Agency Network
NSF – National Science Foundation / Federation

INTERNET	Globally-interconnected network of networks. {WWW is its part}
INTERSPACE {Interspace is what Internet will become}	A client-server software that allows users to communicate with each other in real time to send and receive data of various types such as data files, video, audio ,textual data , 3-D dynamic content.
INTRANET	A local or restricted network within an organization
NEED FOR NETWORKING	Resource Sharing ; Communication Medium : Reliability : Cost Factor

TYPES OF NETWORKS – Based on geographical area and data transfer rate [1 mark]

PAN (Personal Area Network)	LAN (Local Area Network)	MAN (Metropolitan Area Network)	WAN (Wide Area Network)
Interconnecting few personal devices like laptop, mobile etc. Area – 10 meters Bluetooth / USB	Connects devices in limited area, say office, university campus etc. Area – upto 1 Km Ethernet Cable, Fibre Optics, Wi-Fi etc	Extended form of LAN, within the city. Example – CableTV network in a town. Area – 30-40 Km	Connects devices, LANs and WANs across different parts of country or different countries or continents. Example – Internet

NETWORK TOPOLOGIES - pattern of layout or inter-connection between devices (computer nodes , printers etc.) in a network. [1 mark / 2 marks]

	 Computer Computer Computer HUB Computer Printer		
BUS topology Easy to setup ; Less cable length ; Fault diagnosis difficult; Not suitable for large networks	**STAR topology** Centrally controlled ; Fault diagnosis easy; Expensive to setup; If central hub fails, network disrupts.	**RING topology** Easy to setup ; Higher rate of data transmission; Troubleshooting difficult ;	**MESH topology** Network can be expanded without affecting existing LAN ; Robust topology; Complex setup

SWITCHING TECHNIQUES	How data is transmitted over a network [1/2 Mark]
CIRCUIT SWITCHING	Physical connection is setup between source and destination computers to send data. (Dedicated data connection required, Used for phone calls)
MESSAGE SWITCHING	Data divided as data-packets is passed from one switching office to another till it reaches destination. (Store and forward)
PACKET SWITCHING	Data is divided in equal-sized packets at source, transmitted via store and forward way and re-assembled at destination. (Used to send/receive data over a network, **more efficient**)

TERMS USED IN DATA COMMUNICATION	[1 mark – application]
CHANNEL	medium of data transmission from one device/point to another. { Example - you view different TV Channels (broadcast on different frequencies) , YouTube Channels}
BAND WIDTH	Difference between the highest and lowest frequencies (measured in terms of "Hertz" like Hz, KHz, MHz etc)
DATA TRANSFER RATE	amount of data transferred per second
BAUD RATE	measuring unit for data carrying capacity of communcation channel
words used	bps (Bits Per Second), Bps (Bytes per second) , kbps , Kbps etc.
{Note the CAPS 'B' for Bytes, 'b' for bits, you can guess for kbps / Kbps, mbps / Mbps etc }	

TRANSMISSION MEDIA	[1mark – case study based]
WIRED (Guided)	**WIRELESS (Unguided)**
Twisted Pair Cable (Ethernet Cable) Economical and Easy to use stp (shielded twisted pair) , utp (un- shielded twisted pair)	**Infrared** – Are electromagnetic radiation for line-of-sight; Frequency 300 GHz - 400 THz; Range 10-30 meters **Bluetooth** - standard wireless (radio wave) communication protocol uses 2.4 GHz frequency; max range 100 meter
Co-axial Cable Example = cable TV wire	**Radio wave** (frequency range 30 Hz – 300 GHz)
Optical Fiber Cable Most reliable, fast transmission, expensive	**Satellite** (downlink frequency 1.5 – 20 GHz) (Uplink frequency 1.6 GHz – 30 GHz) VERY FAST, EXPENSIVE
	Microwave (frequency range 300 MHz – 300 GHz)
	All unguided media = transmitter, receiver and atmosphere

NETWORK DEVICES	[1 mark – case study]
MODEM (MODulator DEModulator) External modem , Internal modem	enables a computer to transmit data over telephone lines ; Used to convert digital signals into analog signals and vice versa.
RJ45 connector (Registered Jack - 45)	Eight-wire connector used to connect computers on LANs, especially Ethernets.
ETHERNET CARD (**NIC** – Network Interface Card) (**NIU** – Network Interface Unit)	Hardware device that helps in the connection of nodes within a network. Physical address of a NIC is known as MAC address
MAC Address = Media Access Control Address	(6-bytes long → Example 10 : B5 :03 :63:2E:FC)

GATEWAY →	establishes intelligent connection between a local network and external networks that are completely different.
BRIDGE →	connects local networks with same standard but having different types of cables
ROUTER →	connects multiple networks with different protocols
ROUTER v/s	**BRIDGE**
Can handle multiple protocols and works using IP addresses	Cannot manage multiple protocols and works using MAC addresses
GATEWAY →	establishes intelligent connection between a local and external network that are completely different
REPEATER →	used to re-generate received signal and re-transmit towards destination
TIP - When to suggest use of **Repeater**?	When distance between devices is **more than 90 meter**
SWITCH v/s	**HUB**
An intelligent device that connects several nodes for form a network.	An electronic device which connects several nodes to form a network.
Sends information only to intended nodes	Redirects the information to all the nodes in broadcast form.
WiFi Card →	For wireless communication to send and receive signals between devices

Tips for CASE STUDY BASED questions

Question	Hint for Answering
Layout	Draw block diagram interconnecting blocks, prefer the block or unit with maximum devices as main to connect other blocks
Topology	Write name of topology – Star / Bus / Ring etc.
Placement of Server	In the unit/block with maximum number of computers
Placement of Hub/Switch	In every block / unit
Placement of Repeater	As per layout diagram, if distance between two blocks is above 90 meter
Cost-effective medium for internet	Broadband / connection over telephone lines
Communication media for LAN	Ethernet (upto 100 meter) / Co-axial cable for high speed within LAN
Cost/Budget NOT an issue in LAN	Optical Fiber
Communication media for Hills	Radio wave / Microwave
Communication media for Desert	Radio wave
Very fast communication between two cities / countries	Satellite (avoid it in case economical / budget is mentioned)
Device / software to prevent unauthorized access	Firewall (Hardware and/or Software)

NETWORK PROTOCOLS : Set of rules for communication among networked devices. These include how and when a device can send and receive data, how it is packaged, how it reaches its destination.

TCP/IP – Transmission Control Protocol/Internet Protocol. A two-layer protocol.

TCP - divides the data into packets for transmitting and re-assembling received packets at the destination.

IP - responsible for routing the data packets (to find route/way)

PPP Point-to-Point Protocol - Used for direct communication between two devices, like a computer connected via phone line to a server (other examples - cellular telephone, fibre optic links etc)

FTP File Transfer Protocol -Used for transfer of files (upload/download) to or from a remote server.

HTTP HyperText Transfer Protocol- transfer data from one device to another on the world wide web.
HTTP defines how the data is formatted and transmitted over the network.

{ HTTPS - Hypertext Transfer Protocol Secure: advanced and secure version of HTTP. **}**

Wireless / Mobile Communication protocol: [1 mark case study/ full forms]

GSM = Global System for Mobile Communication. GSM technology is used for transmitting mobile voice and data services.

{With GSM, all subscriber and wireless provider information is stored on interchangeable modules known as **SIM (Subscriber Identification Module)** cards. }

GPRS = General Packet Radio Service - transmission of IP packets over existing cellular networks. **Applications** = Multi-media Message Service (MMS), Internet Access via Mobiles and Data Communication

WLL= Wireless Local Loop is a generic term for an access system that uses wireless links rather than conventional copper wires to connect subscribers to the local telephone company's switch.

MOBILE TELECOMMUNICATION TECHNOLOGIES: [1 mark full form / feature]

Mobile is a device which is portable. Mobile communication is based on cellular networks.

{A cellular network is radio network - land is divided into areas called cells. The network of cells enables the mobile devices to communicate even if they are moving from one cell to another via base stations.}

Mobile Systems (G = Generation)

1 G	2 G	2.5 G	3 G
introduced in late 1970s and early 1980s; **analog cellular** technology	introduced in early 1990s; based on **GSM technology**; by swapping out the **SIM card**, users can switch phones or providers.	using **packet switched domain**	Adds multi-media facility to 2G - **allowing video, audio, and graphics** applications ; {Year 2000 – 2010 }
Only **voice facility** available ; based on **circuit-switched** technology	used **circuit switching** ; Both **voice and data conversations** were digitally encrypted	used **GPRS** (General Packet Radio Service) **in addition to GSM.**	Watching **streaming video** or **video telephony** became a reality (Mobile TV) ;
Low capacity , poor voice links and **no security**	Known for **paging, SMS, voicemail and fax services**	Services like **MMS**, sending **pictures through e-mail** possible	**Data rates up to 2 Mbps**; Technologies used – UMTS, EDGE, CDMA

Some terms we need to be familiar with –
FDMA - Frequency Division Multiple Access. **CDMA - Code Division Multiple Access.**
TDMA - Time Division Multiple Access.

4G Mobile Systems = Based on **packet switching only (IP based).** { Year 2010 -2020 };
Bandwidth – 100Mhz ; Term used for 4G is **MAGIC**

Mobile multimedia	**A**nytime, anywhere Fast transmission 100Mbps – 1Gbps	**G**lobal mobile support	**I**ntegrated wireless solutions (uses LTE and Wi-Max)	**C**ustomized personal service

{4G LTE = Fourth Generation Long Term Evolution} 4G can provide **better-than-TV quality images and video-links , supports interactive multimedia, voice and video**

5G Mobile Systems = uses **orthogonal frequency-division multiplexing (OFDM) framework**; radio millimeter bands in the 30 GHz to 300 GHz range. **More faster data transmission than 4G, data rate from 1 Gb and above** { From year 2020 onwards }. Highly interactive multi-media, voice streaming, **more efficient**.

Mobile processors =
Like CPU in a computer system, mobile processor receives and executes every command, performing billions of calculations per second.

Components of Mobile Processors - Mainly the following three -
1. Application Processing Unit = Has the Control Unit of the mobile's CPU (Central Processing Unit)
2. GPU (Graphics Processing Unit) = Assists the CPU for handling the graphics.
3. Communications Processing Unit = for calling and call receiving via the phone's middleware

A few more components in smartphone's processors -

a. Camera ISP (Image Signal Processing)	b. Radio and 3G / 4G Modem
c. Memory Controller	d. Audio / Video Engine

e-Mail – [1 mark – case study / application]
e-Mail or email, short for "electronic mail," is the transmission of messages electronically over computer networks.
e-Mail PROTOCOLS : Email uses multiple protocols within the TCP/IP suite. Some common e-mail protocols are -
SMTP - Simple Mail Transfer Protocol – used to **send emails** on the internet
POP3 - Post Office Protocol Version 3 – to **receive emails** from a remote server to a local email client.
IMAP - Internet Message Access Protocol (IMAP) is a mail protocol used for accessing email on a remote web server from a local client. *{ Example - We use MS-Outlook}*
Telnet – Used to connect to remote computers over a TCP/IP network (interactive , text- based)

PROTOCOLS FOR CHAT AND VIDEO CONFERENCING:
Online conversations in which you are immediately able to send messages back and forth to one another is called **"chat"**.
A **video conference** is a telecommunication technology, which permits two or more people in different locations to interact via mutual video or audio transmission simultaneously.
VoIP - voice over Internet Protocol, which is a base for all internet communications.

WIRELESS TECHNOLOGIES -
Wi-Fi *{ "**WiFi** is a **short** name for **Wireless Fidelity**" }*
Wi-Fi is a wireless networking technology that uses radio waves to allow computers and other devices to communicate over a wireless signal.
WiMax **WiMAX** (**Worldwide Interoperability for Microwave Access**) is a family of wireless broadband communication standards. WiMAX systems are expected to deliver **broadband access** services to **residential and enterprise customers** in an **economical way**.
{ WiFi's range is approx. 30 m. WiMAX range is a radius of 50 km }

NETWORK SECURITY CONCEPTS –

Threats and prevention from –
Viruses – **V**ital **I**nformation **R**esource **U**nder **S**iege; Viruses are small programs that are written intentionally to damage the data and files on a system; computer slows down; programs malfunction; files destroyed
Worms - a self-replicating program that runs independently and travels across network connections. Worms cause more damage.
Trojan horse - a kind of virus that looks safe but has hidden effects.
Spam - unwanted bulk mail which is sent by an unauthorized or unidentified person in order to eat the entire disk space.
PREVENTION – Use anti-virus software; keep computer software updated ; use firewall ; follow safe browsing practices – using authorization , authentication, keeping passwords safe.
COOKIE = A cookie is a small text file sent by web server to a web browser when a user browses a website.
FIREWALL = A hardware or software or both that is used to prevent unauthorized access to or from a computer network;

CYBER LAW	**CYBER CRIME**
Legal system of laws and regulatory aspects of issues of the internet and World Wide Web	**Cybercrime**, or **computer-oriented crime**, is a crime in which a computer and internet is used. Cybercrimes can be against persons or against property or against the government

India IT Act -
"INFORMATION TECHNOLOGY ACT, 2000" [ITA- 2000] - to protect the field of e-commerce, e-governance, e-banking as well as penalties and punishments in the field of Cyber Crimes.
The above Act was further amended in the form of **IT Amendment Act, 2008 [ITAA-2008]** in December 2008. Major aspects covered in IT AA-2008 include **new sections** on offences like **cyber terrorism, data protection, digital signatures, e-Documents (e-governance)** etc.
INTELLECTUAL PROPERTY RIGHTS –
Intellectual property rights are the **rights** given to persons over the **creations of their minds**.
Intellectual Property can be – Industrial Property (Patents , Trademarks) and Copyright.
For example, an invention and an original work of authorship are intellectual property and protected by the intellectual property right called "patent" and "copyright".
Other examples of **Intellectual Property** with a view of IPR –
Patents , Trademarks, Plant Varieties, Copyrights, Trade secrets, Industrial Design rights etc.

HACKING	**CRACKING**
Engaging in **harmless** technical experiments and fun learning activities, using computer programming skills to intentionally **access a computer without authorization.**	A method by which a person **gains unauthorized access** to a computer with the **intention of causing damage**.

Types of Hackers – Black Hat Hackers (also known as crackers), White Hat Hackers, Grey Hat Hackers

WEB SERVICES :
WWW : **World Wide Web** is a combination of all resources and users on the Internet that are using the Hypertext Transfer Protocol (HTTP) ;
Sir **Tim Berners -Lee** (Born in London, UK) is the inventor of WWW.

Website	**Webpage**
Collection of webpages	Webpage is part of website
Each website has specific internet address (URL) by which we can access the website	Webpages have hyperlinks to connect one web page to another in the website
Example - http://cbseacademic.nic.in/ More examples - amazon.com, flipkart.com, google.com	Example – curriculum_2021.html is a webpage of the CBSE website. http://cbseacademic.nic.in/curriculum_2021.html
All publicly accessible websites collectively constitute the **World Wide Web**	

WEB BROWSER = Web browser is software program to navigate the web pages on the internet.
Examples - Google Chrome , Mozilla Firefox, Internet Explorer etc.

WEB HOSTING	**WEB SERVER**
Web hosting is the process of uploading/saving the web content on a web server to make it available on WWW (World Wide Web).	A web server is a computer or a group of computers that hosts or stores content of website. Examples – Apache Tomcat , IIS

Web 2.0 =Web 2.0 refers to new generation of dynamic and interactive websites.

HTML (Hyper Text Markup Language)	**XML**(eXtensible Markup Language)
HTML is used to display the data, text and images of a webpage on web browser and focus is on the format of data displayed.	XML is used to describe the data and focus is on the content of the data. XML is recommended by the World Wide Web Consortium (W3C). It is a free open standard.
HTML tags are predefined	XML tags are not predefined. We can create our own tags. XML code can also contain HTML tags.
HTML tags are not case sensitive. Example - <HTML> or <html> are the same	XML tags are case-sensitive

URL (Uniform Resource Locator)	**DOMAIN NAME**
URL is unique identifier of a web page	Domain name is your website name
The most general form of a URL syntax is as follows: Protocol:// <domain name> / <directory path>/<object name> For example - **https://www.example-site.com/sql/sql_intro.asp**	

(**Domain Name System / Domain Name Resolution** - when the user types a domain name, the domain names are translated into Internet Protocol (IP) addresses. The computers or machines, access websites based on IP addresses)

UNIT III – DATABASE MANGEMENT (20 Marks)

A DBMS or **database management system** is a software used to create and manage databases. {manage = insert new records, display, modify, delete , provide access control etc. }	Database = Collection of inter-related tables and other objects. DBMS = DB + MS , DataBase + software to handle the DB

RDBMS = **Relational Database Management System**	RDBMS = database is organised in the form of relations (i.e. tables)	Examples of popular RDBMS – MySQL, Oracle, Sybase, DB2

TERMINOLGY (RDBMS) [1 mark / 2marks]

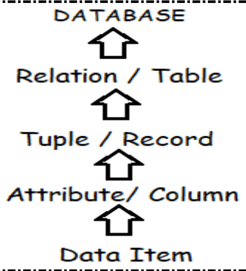
Table : **CUSTOMER**

Cust_Id	**CNAME**	**AADHAR_NO**	**ADDRESS**	**PHONE**
C01	ANUJ KUMAR	345612348912	GAUTAM VIHAR	8765432190
CO3	NEHA SINGH	367823458904	RANI BAGH	7722334673
CO4	NAREN GARG	453627980423	MAHESH NAGAR	6345612566
CO7	BHARAT VERMA	516782245679	MALVIYA NAGAR	9818624567

Primary Key - An attribute or set of attributes that uniquely identify a record in a table/relation e.g. - in table customer(cust_id, cname, aadhar_no, address, phone) , the attribute cust_id is primary key	**Candidate Key**- An attribute or set of attributes that can become primary key of the table/relation e.g. - customer(cust_id, cname, aadhar_no, address, phone) , the attribute cust_id and aadhar_no are candidate keys	**Alternate Key -** The candidate key which is not chosen as primary key e.g. in table customer(cust_id, cname, aadhar_no, address, phone) , the attribute cust_id is chosen as primary key, then aadhar_no will be alternate key

Foreign Key - A non-key attribute of a table whose values are derived from primary key of some other table is known as foreign key in current table.	e.g – Table Customer(cust_id, cname , address, phone)
	
	Table Orders (Order_id, order_dt , cust_id , amount)
		Customer.cust_id = primary key
		Orders.cust_id = foreign key

Database = Collection of inter-related tables and other objects.
Relation /Table = collection of inter-related records
Tuple /Row/ Record = collection of attributes
Attribute/Column / Field = descriptive property of an entity
Data item = value



SQL = Structured Query Language (pronounced as = SEEQUEL)	SQL = Open industry standard language used to query (create and manage) databases	MySQL = Open Source RDBMS (Michael Widenius aka Monty = Chief Inventor)

DATATYPES COMMONLY USED IN **SQL** –

For Text CHAR (n) VARCHAR (n)	Numeric Data INT (n) or INTEGER (n) DECIMAL(n, d) or FLOAT (n)	Date Date format 'yyyy-dd-mm'	Boolean values tinyint(1) (value = 0 means False , 1 means True)

decimal(n,d) = n is total number of digits and d is no of digits after decimal
example - decimal(7,2) => total 7 digits of number (5+ 2 decimal part)

Advantages of SQL [1 mark / 2 marks]

Faster Query Processing ; User-friendly language , no coding skills necessary ; Standardised Language with uniform platform ; Interactive ; Portable

Categories of SQL Commands [1 mark / 2 marks]

DDL = Data Definition Language Used to create/modify table structure (CREATE , ALTER , DROP etc)	DML = Data Manipulation Language Used to change table data (INSERT, UPDATE, SELECT, DELETE etc)	DCL = Data Control Language (GRANT, REVOKE)	TCL = Transaction Control Language (COMMIT, ROLLBACK, SAVEPOINT)

SQL Commands at a glance [1 mark / 2 marks]

1 - CREATE DATABASE = to create tables and work with them	create database <database-name> ;
2 – VIEW LIST OF EXISTING DATABASES ON YOUR MACHINE =	show databases ;
3 – OPEN A DATABASE FOR WORK =	use <database-name> ;
4 – VIEW LIST OF EXISTING TABLES AND / OR VIEWS =	show tables ;
5 – VIEW THE STRUCTURE OF AN EXISTING TABLE =	desc <table-name> ; OR describe <table-name> ;
6 - CREATE TABLE (DDL command)= CREATE TABLE <table-name> (<col1-name> datatype [(size)] [constraint] , <col2-name> datatype[(size)] [constraint] ,	 CREATE TABLE PUPIL (admno integer(4) primary key , name varchar(18) not null,

- - - - -); dob date , gender char(6) DEFAULT 'MALE', fees decimal (7, 2) , avgmark decimal(5,2)) ;

7 - INSERT Records (DML Command)= Two ways of using insert command (A and B) :

A = INSERT INTO <table-name> VALUES (<value-1> , <value-2> , - - -) ;
- - order of data-values same as table structure i.e. columns order

B = INSERT INTO <table-name> (<col1-name> , <col2-name> , - - -)
VALUES (<col1-value> , <col2-value> , - - -) ;
- - useful when inserting partial record or change the order of insertion

- - 'string' or "string" , date as 'yyyy-mm-dd'

- - non-numeric data in 'quotes'

as per A=
insert into pupil values(114, 'ANITA MATHUR', '2002-06-20' , 'FEMALE', 3150.0 , 91.2) **;**

as per B =
insert into pupil(name, admno, dob)
values('DEV SHARMA', 112, '2003-01-03') **;**

8 - ALTER TABLE (DDL command)-

- to add a column
ALTER TABLE <table-name> ADD <col-name> <datatype>[(<size>)] [constraint] **;**

e.g. - ALTER TABLE pupil ADD grade char(2)**;**

- to add integrity constraint
ALTER TABLE <table-name> ADD <constraint> (<col-name>);

- to redefine a column (datatype , size, default-value)
ALTER TABLE <TABLE-NAME>
MODIFY (<COL-NAME> NEWdatatype [(<size>)]) [FIRST | AFTER colname] ;

Example - ALTER TABLE PUPIL Modify name varchar(20);

ALTER TABLE <TABLE-NAME>
MODIFY <old_col_name> <new_col_name > < new_col_definition> ;

9- DROP COMMAND – (DDL command) To delete a table as well as its structure from database.

DROP TABLE <table-name> ; OR DROP TABLE [IF EXISTS] <table-name> ;	DROP TABLE FLIGHT ;
DROP is also used as a CLAUSE in ALTER TABLE command	ALTER TABLE book DROP disc_amt ; ALTER TABLE flight DROP PRIMARY KEY ;

TABLE : **PUPIL**

Admno	**Name**	**DOB**	**Gender**	**Fees**	**Avgmark**	**Grade**
104	RAVINDER	2004-02-24	MALE	3150.0	85.6	B
107	PARTH GUPTA	2003-07-15	MALE	2850.0	90.3	A
112	DEV SHARMA	2003-09-03	MALE	300.0	NULL	C
114	ANITA MATHUR	2003-06-20	FEMALE	3150.0	92.7	NULL
122	NAVNEET	2004-03-10	MALE	2850.0	87.5	B
126	GEETU VERMA	2003-11-16	FEMALE	2700.0	91.4	A
128	PREETI	2004-01-13	FEMALE	3000.0	93.6	A

10- UPDATE Query (DML Command) - To modify existing record(s)

UPDATE <table-name>
SET <col-name> = <value> [, <col2-name> = <value> , - - -]
[WHERE <condition>] ;

update pupil
set avgmark = 89.7 , grade = 'B'
where admno = 107 or admno = 112 ;

11- DELETE Query (DML Command)- To remove a record(S)
DELETE FROM <table-name> [WHERE <condition>] ;
Example - delete from pupil where admno = 126 ;

12- SELECT Query (DML Command) - to view data (content) of a table / view

General syntax - SELECT <col-list> FROM <table-name> [,<table2-name> , - - -] [WHERE <condition>] [ORDER BY ASC | DESC] [GROUP BY <col-name> [HAVING <condition-based-on-group-col>]] ;	(In the commands the keywords are written in CAPITALS so that they are easy to identify. Otherwise SQL commands are NOT CASE SENSITIVE. One can type in small-case or upper-case)
Examples -	There are many ways of using SELECT Command
select admno , dob , name from pupil ; select * from pupil ;	select name , 'was born on' , dob from pupil ; select name , dob AS "Date of Birth" from pupil ;
Column ALIAS NAME – keyword AS used. <col_name> AS <alias-name>	select admno, name, dob AS BIRTHDATE from pupil ;
USE " " or ' ' (quotes) if alias name is more than one word long	(in above examples, column alias BIRTHDATE , "Date of Birth" have been used in order by clause)

Following are the clauses/operators which can be used with SELECT command:

DISTINCT - Used to display distinct values from a column of a table.	select DISTINCT name from pupil; To view data of names (without repetition) of the students
WHERE - Used to specify the condition based on which rows of a table are displayed	**OPERATORS USED IN WHERE CLAUSE -** > , < , < = , = , != , AND (&&) , OR (||), NOT
To view name, dob, grade of students with 'B' grade.	select name, dob, grade from pupil WHERE grade = 'B' ;
To view data of admission number 126	SELECT * FROM pupil WHERE admno = 126 ;
To view name, admission number and date_of_birth of those students who have fees more than 3000	select name , admno , dob from pupil where fees > 3000 ;
BETWEEN - Used to define the range of values within which the column values must fall to make a condition true.	Range includes both the upper and the lower values. select * from pupil where fees BETWEEN 3000 AND 3500 ;
Same command using AND , relational operators	select * from pupil where fees >= 3000 AND fees <= 3500 ;
IN - Used to select values that match any value in a list of Specified values	select admno, name from pupil where name IN ('RAVINDER' , 'NAVNEET ') ;
Same command using OR operator	select admno, name from pupil where name = 'RAVINDER' name = 'NAVNEET ' ;
LIKE - Used for pattern matching of string data using wildcard characters % and _	% = zero, one or many characters _ = single character (underscore symbol)
To view data from pupil for names begin with letter 'P'	select * from pupil where name LIKE 'P%' ;
To view details of those students whose fees is more than 3000 and name ends with 'R'	select * from pupil where fees > 3000 AND name LIKE '%R' **;**
'A%' = the string begins with character 'A' '_ _a%' = the third character is 'a' '_ _ a' = any three letter string that ends in 'a' '_ _ _ _' = Any four letter string	(I have typed space in between underscore character to show clarity, it is typed in continuity in the actual command)

select * from pupil where name LIKE 'A%' ORDER BY name **;**	select empno , ename, salary+comm AS "TOTAL_PAY" from employee where name LIKE 'R%' ORDER BY total_pay ;
IS NULL / IS NOT NULL -	Used to select rows in which the specified column is NULL (or IS NOT NULL)
To view the details of those students whose dob is not entered / dob datavalue is not available.	select * from pupil where dob IS NULL ;
ORDER BY - Used to display the selected rows in ascending or in descending order of the specified column/expression	by default the ORDER is ASC i.e. ascending order. For descending order, must specify DESC
select * from pupil ORDER BY name ; OR select * from pupil ORDER BY name ASC ;	select admno, dob, name, grade from pupil ORDER BY name DESC ; select * from pupil ORDER BY grade , name DESC ; (NOTICE two columns in order by, here Col1 – ASC, Col2-DESC)
GROUP BY – To apply a SQL SELECT query on a group of records instead of whole table. GROUP BY <column-name> is used	A group column is generally that column of the table which has repeating values.
For example, columns fees, gender , grade in table pupil have repeating values – you can group the values of these columns.	select gender, count(*) from pupil GROUP BY gender ; select max(fees) from pupil GROUP By grade ;

Avoid non-group function or non-group column in SELECT clause.

Group functions ignore NULL values.

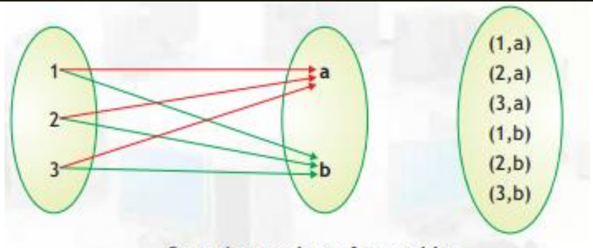
Look at ORDER BY AND GROUP BY again – Order by	Group by
HAVING - To add condition to GROUP BY column. (i.e. Use only with Group By)	select grade, avg(marks) from pupil group by grade HAVING count(*) > 1 ;

AGGREGATE FUNCTIONS (Also known as Group/ Multi-Row Functions) [1 / 2 marks – output)

SUM()	Returns the sum of the column
MIN()	Returns the minimum value in the given column or set of values
MAX()	Returns the maximum in the given column or set of values
AVG()	Returns the average value of data in the given column or set of values
COUNT()	Returns the total number of values / records as per the given column

WORKING WITH MORE THAN ONE TABLE - [1 / 2 mark – query]

CARTESIAN PRODUCT OR CROSS JOIN	Cross Join of two tables is obtained by pairing up each row of one table with each row of the other table.



Cartesian product of two tables

Table A (3 rows, 4 columns)
Table B (2 rows, 3 columns)
A X B = 6 rows, 7 columns (3 x2 , 4 + 3)
(degree = total columns , cardinality = total rows in a table)

TIPS - memorise Degree / Cardinality C **not** C { Column is **not** Cardinality}
Column (6 Letters) = Degree (6 letters)
Cardinality (longer word) = Rows (horizontally long)

Table : BOOKS

Book_Id	Book_Name	Publishers	Type
C01	Fast Cook	EPB	Cookery
F01	The Tears	First	Fiction
T01	My C++	TDH	Text
T02	C++ Brain	TDH	Text
F02	Thuderbolts	First	Fiction

Table : ISSUED

Book_Id	Price	Qty_Issued
T01	400	4
C01	350	5
F01	280	2
C01	300	6

EQUI_JOIN – joining based on common column (use it in WHERE clause, <table1>.<common-column> = table2.<common-column>)

select * from books ,issued where books.book_id = issued.book_id ;

NATURAL_JOIN – like equi-join, difference is that the common column of tables appears only once in the result

select * from books NATURAL JOIN issued;

Example , To display the Book Name, Quantity_Issued and Price for all books of TDH publishers-
select book_name , qty_issued , price from books B , issued S
where B.book_id = S.book_id and publishers = 'TDH' ;
(Here B , S are table ALIAS NAMES)

INTERFACE OF PYTHON WITH AN SQL DATABASE: -2 MARKS (5 MARKS PRACTICAL EXAM)

INTERFACE OF PYTHON WITH AN SQL DATABASE

When we want to design real life applications to manipulate
Data stored in database we need interface python with MySQL. The steps are

(i) We use pip install MySQL.Connector :This command we use to install library of MySQL with python.
(ii)import MySQL.connector: This statement run on python to access the module of MySQL if we don't get
Any error means this module working properly.
(iii) mydb=MySql.connector.connect(host="localhost",user="root",passwd="tiger",database="school") : To make the connection with MySQL database using connect() function where user, password and database as per our system which we assign during installing of MySQL. Mydb is connection object.
(iv)cursor=mydb.cursor()-a database cursor is useful control structure for row by row processing of records
(v) cursor.execute("select * from stud"):It will execute the sql query and store the retrieved records.
(vi) data=cursor.fetchall():Extract data from result set using fetch() functions.

fetchall()	:It will return all the records retrieved in tuple form.
fetchone()	:It will return one record from the result set.
fetchmany(n)	:It will return number of records as per value of n and by-default only one record.

(vii) count=coursor.rowcount
It is the property of cursor object that return number of rows retrieved.

Q1. Which command is use to install MySQL library in python?
Ans: pip install MySQL. Connector with path of python

Q2. Which method we use to establish the connection?
Ans: connect() method with connection object.

Q3. Which statement we use to access the MySQL module?
Ans: import mysql.connector

Q4. What are the difference between fetchone(),fetchmany(),fetchall()? Hint- Above given

Q5. Mr.Harsh want to interface python with mysql and write some code help him to write the code

```
import___________.connector                                #Line1
mydb=mysql.connector._______(host="localhost",user="root",
passwd="tiger",database="school")                           #Line2
cursor=mydb.__________()                                   #Line3
cursor.______________("select * from stud")                #Line4
data=cursor._________()                                    # Line 5 To retrieved all records
count=cursor._________                                     #Line6 To count total rows
```

Ans: Line1:-mysql, Line2:-connect, Line3:cursor ,Line4: execute, Line5: fetchall, Line6: rowcount

12. CSV FILE

CSV stands for Comma Separated values.
e.g.

rno, name, marks
11, Kush, 55
12, Abhijeet, 59

Rno	Name	Marks
11	Kush	55
12	Abhijeet	59

A CSV (Comma Sepparated Value) format is one of the simplest and common way to store tabular data. To represent a csv file, it must be saved with .csv extension.

csv file is used to store tabular data in a comma separated values.
Each line of file is a data record
Each record consists of one or more field separated by comma. (comma is separator). Comma is also a delimiter which separates two files in a row

csv module is used to perform read/ write operation in csv file.

import csv module	# import csv
csv.reader( )	# Read from csv file
csv.writerow( )	# Write to csv file – single row
csv.writerows( )	# Write to csv file – multiple rows
open( )	# Functions to open csv file
close( )	# Functions to close csv file

ACCESS MODE

r	-	read mode	used to read the content of csv file.
w	-	write mode	used to write to csv file, previous data is first deleted and write new data. Only newly added data will be shown to csv file.
a	-	append mode	data will be added at the last previous data is not deleted, only inserted after previous data

OPEN CSV FILE

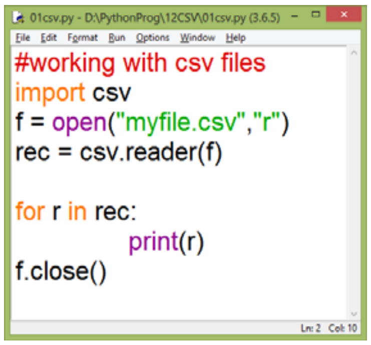
There are two ways to open a csv file

a. file variable/ file handler = open(csv_file_name, access_mode)
b. using with function
   with open(csv_file_name, access_mode) as file variable/ file handler :

CLOSE CSV FILE

file variable/ file handler.close( )

```
#working with csv files
import csv
f = open("myfile.csv", "r")
rec = csv.reader(f)

for r in rec:
    print(r)
f.close()
```


01csv.py - D:\PythonProg\12CSV\01csv.py (3.6.5)
File Edit Format Run Options Window Help
Ln: 2 Col: 10

```
#working with csv files
import csv
f = open("myfile.csv","r")
rec = csv.reader(f)

for r in rec:
        print(r)
f.close()
```

READING CSV FILE

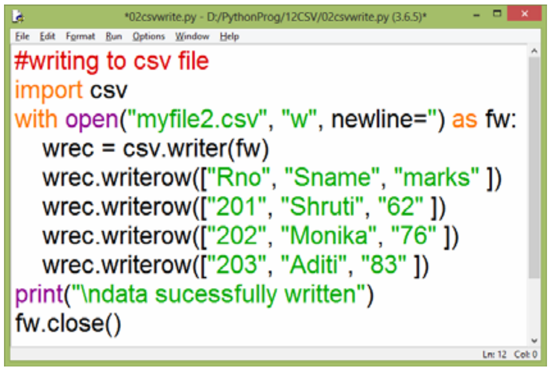
The csvreader( ) function is used to read the file, which returns an iterable reader object.
The reader object is then iterated using for loop to print the content of each row.
csv.reader() function in default mode of csv file having comma delimiter.
If our csv file is a tab delimiter, to read such files, we need to pass optional parameter to csv.reader( ) fnction.

```
f = open("myfile.csv", "r", “\t”)          f = open(csvfile, mode,delimiter)
                                            f = open("myfile.csv", "r", “,”)
```

WRITING TO CSV FILE

To write csv file in python, we can use csv.writer() function.
The csv.writer() function returns a writer object that converts the user’s data into delimiter string.
The string can later be used to write into CSV File using writerow( ) function.

```
#writing to csv file
import csv
with open("myfile2.csv", "w", newline='') as fw:
    wrec = csv.writer(fw)
    wrec.writerow(["Rno", "Sname", "marks" ])
    wrec.writerow(["201", "Shruti", "62" ])
    wrec.writerow(["202", "Monika", "76" ])
    wrec.writerow(["203", "Aditi", "83" ])
print("\ndata sucessfully written")
fw.close()
```

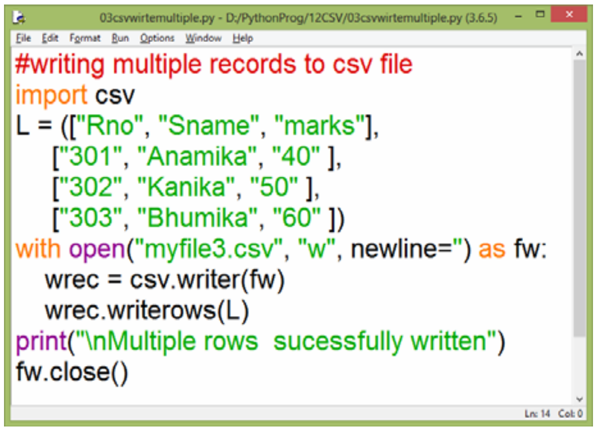

02csvwrite.py - D:/PythonProg/12CSV/02csvwrite.py (3.6.5)
File Edit Format Run Options Window Help
Ln: 12 Col: 0

```
#writing to csv file
import csv
with open("myfile2.csv", "w", newline='') as fw:
    wrec = csv.writer(fw)
    wrec.writerow(["Rno", "Sname", "marks" ])
    wrec.writerow(["201", "Shruti", "62" ])
    wrec.writerow(["202", "Monika", "76" ])
    wrec.writerow(["203", "Aditi", "83" ])
print("\ndata sucessfully written")
fw.close()
```

WRITE MULTIPLE ROWS WITH WRITEROWS( ) FUNCTION

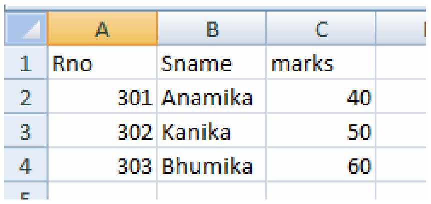
If we need to write the content of 2-Dimensional list into csv file, instead of using writerow( ) function many times, we can write **writerows()** function.

```
#writing Multiple records to csv file
import csv
L = (["Rno", "Sname", "marks"],
    ["301", "Anamika", "40" ],
    ["302", "Kanika", "50" ],
    ["303", "Bhumika", "60" ])
with open("myfile3.csv", "w", newline='') as fw:
   wrec = csv.writer(fw)
   wrec.writerows(L)
print("\nMultiple rows  sucessfully written")
fw.close()
```


03csvwirtemultiple.py - D:/PythonProg/12CSV/03csvwirtemultiple.py (3.6.5)
File Edit Format Run Options Window Help
Ln: 14 Col: 0

```
#writing multiple records to csv file
import csv
L = (["Rno", "Sname", "marks"],
    ["301", "Anamika", "40" ],
    ["302", "Kanika", "50" ],
    ["303", "Bhumika", "60" ])
with open("myfile3.csv", "w", newline='') as fw:
   wrec = csv.writer(fw)
   wrec.writerows(L)
print("\nMultiple rows  sucessfully written")
fw.close()
```

CONTENT OF CSV FILE : MYFILE3.CSV

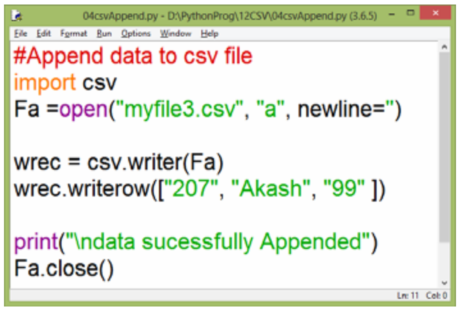


	A	B	C
1	Rno	Sname	marks
2	301	Anamika	40
3	302	Kanika	50
4	303	Bhumika	60

APPEND DATA TO CSV FILE

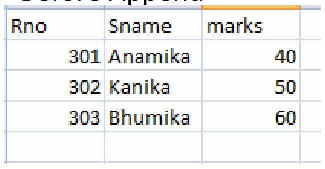
Append data means, previous data will exists. After all previous data, new data will be inserted after last record. ACCESS MODE will be “a”

```
#Append data to csv file
import csv
Fa =open("myfile3.csv", "a", newline=''):
   wrec = csv.writer(fw)
   wrec.writerow(["207", "Akash", "99" ])
print("\ndata sucessfully Appended")
fw.close()
```



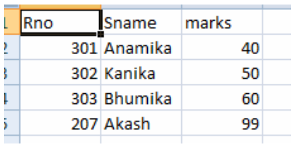
```
#Append data to csv file
import csv
Fa =open("myfile3.csv", "a", newline='')

wrec = csv.writer(Fa)
wrec.writerow(["207", "Akash", "99" ])

print("\ndata sucessfully Appended")
Fa.close()
```

Before Append



Rno	Sname	marks
301	Anamika	40
302	Kanika	50
303	Bhumika	60

After append

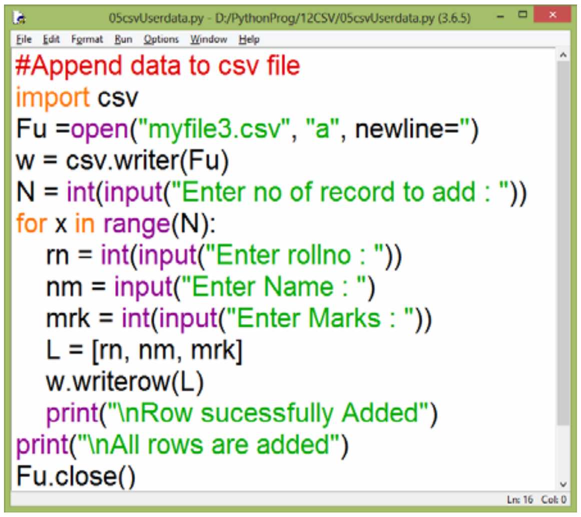


Rno	Sname	marks
301	Anamika	40
302	Kanika	50
303	Bhumika	60
207	Akash	99

APPEND DATA TO CSV FILE FROM USER

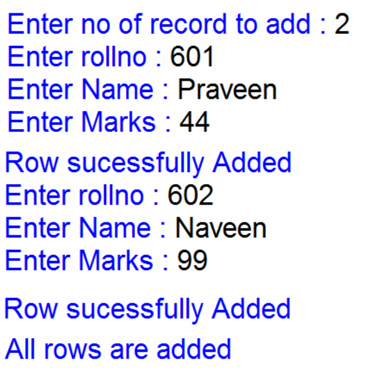
Append data means, previous data will exists. After all previous data, new data will be inserted after last record. ACCESS MODE will be “a”
Record will be given by user when program is run and will be added to csv file.

```
#Append data to csv file
import csv
Fu =open("myfile3.csv", "a", newline='')
w = csv.writer(Fu)
N = int(input("Enter no of record to add : "))
for x in range(N):
   rn = int(input("Enter rollno : "))
   nm = input("Enter Name : ")
   mrk = int(input("Enter Marks : "))
   L = [rn, nm, mrk]
   w.writerow(L)
   print("\nRow sucessfully Added")
print("\nAll rows are added")
Fu.close()
```



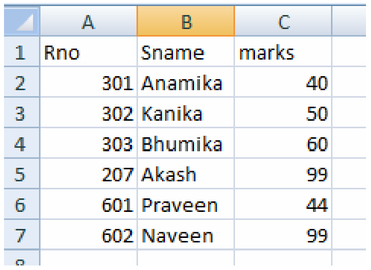
```
#Append data to csv file
import csv
Fu =open("myfile3.csv", "a", newline='')
w = csv.writer(Fu)
N = int(input("Enter no of record to add : "))
for x in range(N):
   rn = int(input("Enter rollno : "))
   nm = input("Enter Name : ")
   mrk = int(input("Enter Marks : "))
   L = [rn, nm, mrk]
   w.writerow(L)
   print("\nRow sucessfully Added")
print("\nAll rows are added")
Fu.close()
```

OUTPUT



```
Enter no of record to add : 2
Enter rollno : 601
Enter Name : Praveen
Enter Marks : 44

Row sucessfully Added
Enter rollno : 602
Enter Name : Naveen
Enter Marks : 99

Row sucessfully Added

All rows are added
```

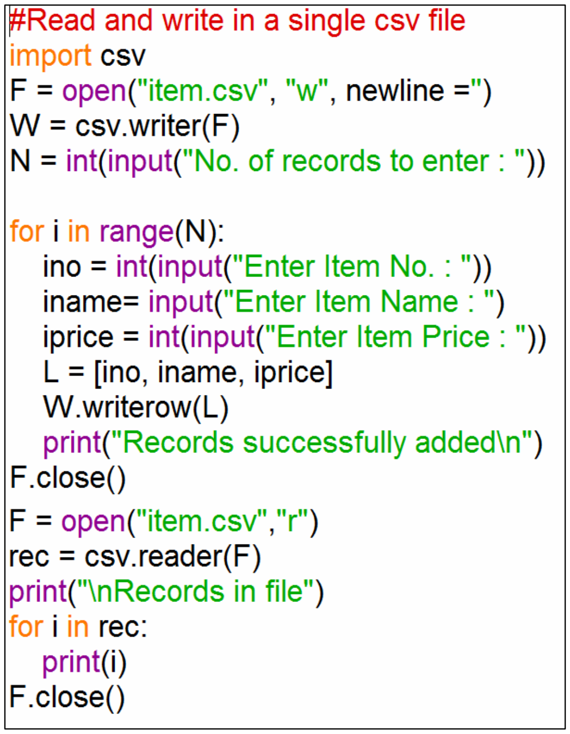
CSV FILE



	A	B	C
1	Rno	Sname	marks
2	301	Anamika	40
3	302	Kanika	50
4	303	Bhumika	60
5	207	Akash	99
6	601	Praveen	44
7	602	Naveen	99

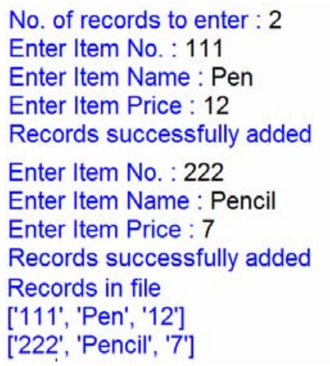
SINGLE FILE TO FIRST CREATE FILE USING WRITE AND THEN READ RECORD

```
#Read and write in a single csv file
import csv
F = open("item.csv", "w", newline ='')
W = csv.writer(F)
N = int(input("No. of records to enter : "))

for i in range(N):
    ino = int(input("Enter Item No. : "))
    iname= input("Enter Item Name : ")
    iprice = int(input("Enter Item Price : "))
    L = [ino, iname, iprice]
    W.writerow(L)
    print("Records successfully added\n")
F.close()

F = open("item.csv","r")
rec = csv.reader(F)
print("\nRecords in file")
for i in rec:
    print(i)
F.close()
```



```
#Read and write in a single csv file
import csv
F = open("item.csv", "w", newline ='')
W = csv.writer(F)
N = int(input("No. of records to enter : "))

for i in range(N):
    ino = int(input("Enter Item No. : "))
    iname= input("Enter Item Name : ")
    iprice = int(input("Enter Item Price : "))
    L = [ino, iname, iprice]
    W.writerow(L)
    print("Records successfully added\n")
F.close()
F = open("item.csv","r")
rec = csv.reader(F)
print("\nRecords in file")
for i in rec:
    print(i)
F.close()
```

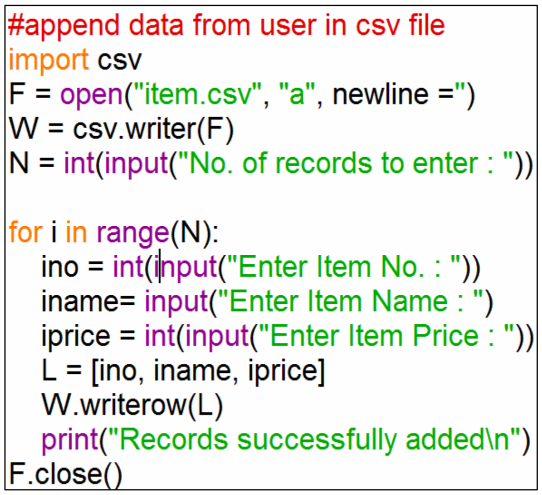
OUTPUT



```
No. of records to enter : 2
Enter Item No. : 111
Enter Item Name : Pen
Enter Item Price : 12
Records successfully added

Enter Item No. : 222
Enter Item Name : Pencil
Enter Item Price : 7
Records successfully added
Records in file
['111', 'Pen', '12']
['222', 'Pencil', '7']
```

APPEND RECORD

```
#append data from user in csv file
import csv
F = open("item.csv", "a", newline ='')
W = csv.writer(F)
N = int(input("No. of records to enter : "))

for i in range(N):
    ino = int(input("Enter Item No. : "))
    iname= input("Enter Item Name : ")
    iprice = int(input("Enter Item Price : "))
    L = [ino, iname, iprice]
    W.writerow(L)
    print("Records successfully added\n")
F.close()
```



```
#append data from user in csv file
import csv
F = open("item.csv", "a", newline ='')
W = csv.writer(F)
N = int(input("No. of records to enter : "))

for i in range(N):
    ino = int(input("Enter Item No. : "))
    iname= input("Enter Item Name : ")
    iprice = int(input("Enter Item Price : "))
    L = [ino, iname, iprice]
    W.writerow(L)
    print("Records successfully added\n")
F.close()
```

* * * * * * * * * * * * * * *

SAMPLE QUSTIONS

1

What possible output(s) are expected to be displayed on screen at the time of execution of the program from the following code?

```
                    import random
                    POINTS=[20,40,10,30,15]
                    BEGIN=random.randint(1,3)
                    LAST=random.randint(2,4)
                    for C in range (BEGIN,LAST+1):
                              print(POINTS[C],'#')

(i)       30# 15#                  (ii)      10# 30 # 15 #
iii)      20 # 40# 15 #            (iv)      40 # 10# 30 # 15 #

             (a) i,ii        (b) i, ii, ii,      (c) i,ii,iv        (d) ii, iii,iii
```

```
STRING = ‘CBSEONLINE’
NUMBER= random.randint(0,3)
N=9
while   STRING[N] !=’L’ :
          print( STRING[N] + STRING[NUMBER] + “#”, end=’  ‘)
           NUMBER+=1
           N=N-1

     (i)       ES#NE#IO#
     (ii)      LE$NO#ON#
     (iii)     NS#IE#LO#
     (iv)      EC#NB#IS#
```

2

```
x=[1,2,3]
counter=0
while counter < len(x):
          print( x[counter]*% )
          for y in x :
                    print(y * ’*’)
          counter+=1
==================================
```

Find Output

```
                    Msg1= “WeLcOME “
                    Msg2= “GUeSTs”
                    Msg3= “”
                    for I in range (0,   len(Msg2) + 1 ) :
                              if Msg1[ I ] >=”A” and Msg1 [ I ] <=”M” :
                                        Msg3=Msg3+Msg1[ I ]
                              elif Msg1[ I ] >=”N” and Msg1 [ I ] <=”Z” :
                                        Msg3=Msg3+Msg2[ I ]
                              else :
                                        Msg3=Msg3 + “*”
print(Msg3)
```

Find the output of the following

```
 Text=”gmail@com”
  L=len(Text)
        ntext=” “
        For i in range (0,L):
              If    text[i].isupper():
                     ntext=ntext+text[i].lower()
             elif text[i].isalpha():
  ntext=ntext+text[i].upper()
              else:
                ntext=ntext+’bb’
========================================
```

3

Predict the output   :

```
def  check(n1=1, n2=3):
          n1 = n1+n2:
          n2+=1
          print(n1,n2)

check()
check(2,1)
check(4)
```

Trace the flow of execution for following program :

```
        (a)   1      def power(b, p) :
               2             r = b ** p
               3             return r
               4
               5       def calcSquare(a) :
               6              a = power(a, 2)
               7              return a
               8
               9         n = 5
               10        result = calSquare(n)
               11         print(result)
```

4

Consider the following code :
```
string = input ( “Enter a string:” )
 count = 3
          while True :
                 if string[0]== ‘a’ :
                                  string = string[2 :]
                 elif string[-1] == ‘b’ :
                                 string = string [ : 2]
                 else :
                                 count += 1
                                 break
          print (string)
          print (count)
```
what will be the output produced if the input is
(i)   aabbcc   (ii)   aaccbb   (c) abcc

```
Inp = input (“Enter a String”)
while len(Inp) <=4 :
        if Inp[-1]==’z’ :                    # cond. 1
                Inp=Inp[0:3]+’c’
        elif ‘a’ in Inp:                      # cond 2
                Inp=Inp[0]+’bb’
        elif not int (Inp[0]) :            # Cond. 3
                Inp = ‘1’ Inp[1:]+’z’
        else :
                Inp=Inp + ’*’
```
What will be the output if the input is
(a) 1bzz
(b) xyz
(c) 0xy

5

Find the order of line number in which this program will execute
```
1. def power (b,p)  :
2.       y = b ** p
3.       return y
4.
5. def calcSquare(x) :
6.       a=power(x,2)
7.       return a
8. n=5
9. result = calcSquare(n) + power(3,3)
10. print(result)
```

```
x="hello world"
print(x[:2],x[:-2],x[-2:])
print(x[6],x[2:4])
print(x[2:-3],x[-4:-2])
```

6

(a) Evaluate the following expression
12, 7, 3, -   /, 2, 1, 5, + , * , +
(b) Convert the following expression in postfix form :
A * (B + D) / E – F - (G+H / K)
(c) True and False OR not True and False or Not True

29 (a) Evaluate the following expression
12, 7, 3, -   /, 2, 1, 5, + , * , +
29 (b) Convert the following expression in postfix form :
A * (B + D) / E – F - (G+H / K)
29c)   Convert to Post Fix Operation
True and False OR not True and False or Not True

7

Write a function in python, **AddQ(Arr)** and **RemoveQ(Arr)** for performing insertion and deletion operations in a Queue. **Arr** is the list used for implementing queue and **data** is the value to be inserted.
OR.

Write a function in python, S**Push(package)** and S**Pop(package)** to add a new Package and delete a Package from a List of Package Description, considering them to act as push and pop operations of the Stack data structure.

8

```
def AddQ(Arr):
    data=int(input("enter data to be inserted: "))
    Arr.append(data)
-------------------------
def RemoveQ(Arr):
    if (Arr==[]):
        print( "Queue empty")
   else:
       print ("Deleted element is: ",Arr[0])
       del(Arr[0])
```

```
def SPush(Package):
      a=int(input("enter package title : "))
      Package.append(a)
--------------------------------
def SPop(Package):
    if (Package==[]):
       print( "Stack empty")
   else:
       print ("Deleted element:",Package.pop())
```

9

```
        def isFull(stack):
        if (top==maxsize-1):
                return True
        else:
                return False
```

```
def isEmpty(stack):
        if (top==-1):
                return True
        else:
                return False
```

10

```
def Traverse(stack):
if (isEmpty(stack)):
print("Stack is Empty ")
else:
for i in stack:
print(i,end=" ")
```

```
def Push(stack):
global top
if (isFull(stack)):
print("\n Stack is OverFlow \n ")
else:
n=int(input("Enter An Element to Push "))
top=top+1
stack.append(n)
```

```
def Pop(stack):
global top
if (isEmpty(stack)):
print("\n Stack is UnderFlow \n")
else:
n=stack[top]
print("Removed Element ",n)
top=top-1
stack.pop()
```

```
## Write MAIN FUNCTION FOR IT
```

12

Center to center distances between various blocks

Harsh Building to Raj Building	50 m
Raz Building to Fazz Building	60 m
Fazz Building to Jazz Building	25 m
Jazz Building to Harsh Building	170 m
Harsh Building to Fazz Building	125 m
Raj Building to Jazz Building	90 m

Number of Computers

Harsh Building	25
Raj Building	50
Fazz Building	125
Jazz Bulding	10

e1) Suggest a suitable cable layout of connections between the buildings.
e2) Suggest the most suitable place (i.e. building) to house the server of this organisation with a suitable , reason.
e3) Suggest the placement of the following devices with justification:
(i) Internet Connecting Device/Modem
(ii) Switch
e4) The organisation is planning to link its sale counter situated in various parts of the same city, which type of network out of LAN, MAN or WAN will be formed? Justify your answer.
e5) What type of channel (Medium) will you use to form this network

6.INDIAN PUBLIC SCHOOL in Darjeeling is setting up the network between its different wings. There are 4 wings named as SENIOR(S), JUNIOR (J), ADMIN (A) and HOSTEL (H).

Distance between various Wings

Wing A to Wing S	100 m
Wing A to Wing J	200 m
Wing A to Wing H	400 m
Wing S to Wing J	300m
Wing S to Wing H	100 m
Wing J to Wing H	450 m

Number of Computers

Wing A	10
Wing S	200
Wing J	100
Wing H	50

==========================

(i) Suggest a suitable Topology for networking the computer of all wings.
(ii) Name the wing where the server is to be installed. Justify your answer
(iii) Suggest the placement of Hub/Switch in the network.
(iv) Mention the economic technology to provide internet accessibility to all wings.

13	Write definition of a Method MSEARCH(STATES) to display all the state names from a list of STATES, which are starting with alphabet M. For example: If the list STATES contains [“MP’,”UP”,”MH”,”DL”,”MZ”,”WB”] The following should get displayed MP MH MZ	1. def Changer(P,Q=10): a. P=P/Q b. Q=P%Q c. print P,"#",Q d. return P 2. -------------------- 3. A=200 4. B=20 5. A=Changer(A,B) 6. print A,"$",B 7. B=Changer(B) 8. print (A,"$",B) 9. A=Changer(A) 10. print (A,"$",B)

14	def count H ():
	    f = open (“para.txt” , “r”)
	    lines =0
	    l=f. readlines ()
	    for i in L:
	        if i [0]== ‘H’:
	            lines +=1
	    print (“No. of lines are: “ , lines)
		def countmy ():
		    f=open (“DATA.txt” ,”r”)
		    count=0
		    x= f.read()
		    word =x.split ()
		    for i in word:
		        if (i == “my”):
		            count =count + 1
		    print (“my occurs” ,count, “times”)

15	def display ():
	    file=open(MYNOTES.TXT’ , ‘r’)
	    lines=file.readlines()
	    while line:
	        if line[0]==’K’ :
	        print(line)
	        line=file.readline()
	        file.close()
	------------------------------------ 	
	def Readfile():
	    i=open(“Employee.dat” , “rb+”)
	    x=i .readline()
	    while(x):
	        I= x.split(‘:’)
	        if ((float (I[2]) >=20000) and (float I[2])<=40000):
	            print(x)
	        x= i.readline()
		def Readfile():
		    i=open(“Employee.dat” , “rb+”)
		    x=i .readline()
		    while(x):
		        I= x.split(‘:’)
		        if ((float (I[2]) >=20000) and (float
		    I[2])<=40000):
		            print(x)
		        x= i.readline()
		-- 	
		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”)

Q-16 Write SQL queries for (i) to (iv) and find outputs for SQL queries (v) to(viii), which are based on the tables

TRAINER

TID	TNAME	CITY	HIREDATE	SALARY
101	SUNAINA	MUMBAI	1998-10-15	90000
102	ANAMIKA	DELHI	1994-12-24	80000
103	DEEPTI	CHANDIGARG	2001-12-21	82000
104	MEENAKSHI	DELHI	2002-12-25	78000
105	RICHA	MUMBAI	1996-01-12	95000
106	MANIPRABHA	CHENNAI	2001-12-12	69000

COURSE

CID	CNAME	FEES	STARTDATE	TID
C201	AGDCA	12000	2018-07-02	101
C202	ADCA	15000	2018-07-15	103
C203	DCA	10000	2018-10-01	102
C204	DDTP	9000	2018-09-15	104
C205	DHN	20000	2018-08-01	101
C206	O LEVEL	18000	2018-07-25	105

(i) Display the Trainer Name, City & Salary in descending order of their Hiredate.

(ii) To display the TNAME and CITY of Trainer who joined the Institute in the month of December 2001.

(iii) To display TNAME, HIREDATE, CNAME, STARTDATE from tables TRAINER and COURSE of all those courses whose FEES is less than or equal to 10000.

(iv) To display number of Trainers from each city.

(v) SELECT TID, TNAME, FROM TRAINER WHERE CITY NOT IN('DELHI', 'MUMBAI');

(vi) SELECT DISTINCT TID FROM COURSE;

(vii) SELECT TID, COUNT(*), MIN(FEES) FROM COURSE GROUP BY TID HAVING COUNT(*)>1;

(viii) SELECT COUNT(*), SUM(FEES) FROM COURSE WHERE STARTDATE< '2018-09-15';