

KENDRIYA VIDYALAYA SANGATHAN BHOPAL REGION



**STUDY MATERIAL
COMPUTER SCIENCE (083)
CLASS - XII**

2024-25



OUR PATRONS



Shri. Dr. R. Senthil Kumar

Deputy Commissioner KVS RO, Bhopal



Smt. Rani Dange

Asst. Commissioner KVS RO, Bhopal



Smt. Kiran Mishra

Asst. Commissioner KVS RO, Bhopal



Smt. Nirjala Budania

Asst. Commissioner KVS RO, Bhopal



Shri. Vijay Vir Singh

Asst. Commissioner KVS RO, Bhopal

INDEX

Unit No.	Unit Name	Topic	Page No.
1	Computational Thinking & Programming - 2	Rev. Tour (Python Basics [Data Types + Operators etc], Conditional Statements, Looping)	5-26
		Rev. Tour (String, List, Tuple, Dictionary)	27-40
		Intro to Python Modules + Working with Functions	41-63
		Exception Handling + File Handling	64-75
2	Computer Networks	Computer Networks	
3	Database Management	Database Management	

Content Coordinators

Shri Rajesh Sahu, Principal
PMSHRI Kendriya Vidyalaya Narmadapuram

Smt. Sangeeta M Chauhan
PGT (Comp. Sc.)
PM SHRI Kendriya Vidyalaya
No. 3 Morar Cantt, Gwalior

Smt. Kirti Asrekar
PGT (Comp. Sc.)
PMSHRI Kendriya Vidyalaya
Seoni Malwa

Content Creators

Name of K.V.	Name of P.G.T. (Comp. Sc.)
Tikamgarh	Mrs. Monika Verma
IIT Indore	Mr. Deepak Bhide
Mungaoli	Mr. Vijendra Rathore
Indore No. 1(Shift I)	Mr. Kamal Kishore Sharma
Itarsi CPE	Mrs. Hemlata Soni

Final Design and Compilation

Mhow	Mrs. Manisha Dubey
Mungaoli	Mr. Vijendra Rathore
Dhar	Mrs. Manisha Malviya
Indore No. 1(Shift I)	Mr. Kamal Kishore Sharma

Unit I: Computational Thinking and Programming –

Revision of Python topics covered in Class XII

Python tokens:

(1) keywords :

Keywords are reserved words. Each keyword has a specific meaning to the Python interpreter, and we can use a keyword in our program only for the purpose for which it has been defined. As Python is case sensitive, keywords must be written exactly.

(2) Identifier :

Identifiers are names used to identify a variable, function, or other entities in a program.

The rules for naming an identifier in Python are as follows:

- The name should begin with an uppercase or a lowercase alphabet or an underscore sign (`_`). This may be followed by any combination of characters `a–z`, `A–Z`, `0–9` or underscore (`_`). Thus, an identifier cannot start with a digit.
- It can be of any length. (However, it is preferred to keep it short and meaningful).
- It should not be a keyword or reserved word
- We cannot use special symbols like `!`, `@`, `#`, `$`, `%`, etc., in identifiers.

Variables:

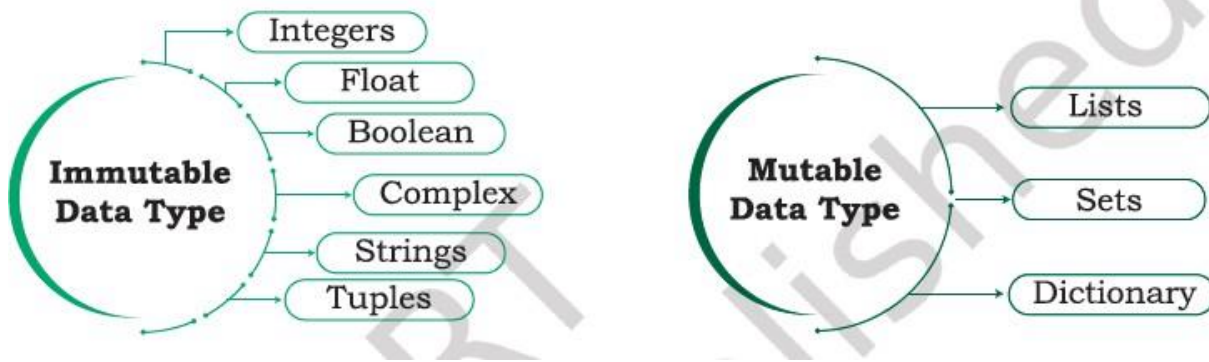
A variable in a program is uniquely identified by a name (identifier). Variable in Python refers to an object — an item or element that is stored in the memory.

Comments:

Comments are used to add a remark or a note in the source code. Comments are not executed by interpreter. a comment starts with `#` (hash sign). Everything following the `#` till the end of that line is treated as a comment and the interpreter simply ignores it while executing the statement.

Mutable and immutable data types :

Variables whose values can be changed after they are created and assigned are called mutable. Variables whose values cannot be changed after they are created and assigned are called immutable.



Operators:

An operator is used to perform specific mathematical or logical operation on values. The values that the operators work on are called operands.

Arithmetic operators :four basic arithmetic operations as well as modular division, floor division and exponentiation. (+, -, *, /) and (% , // , **)

Relational operators :

Relational operator compares the values of the operands on its either side and determines the relationship among them. ==, != , > , < , <= , , >=

Logical operators :

There are three logical operators supported by Python. These operators (and, or, not) are to be written in lower case only. The logical operator evaluates to either True or False based on the logical operands on either side. and , or , not

Assignment operator :

Assignment operator assigns or changes the value of the variable on its left. a=1+2

Augmented assignment operators :

+ = , - = , / = * = , // = % = , ** =

Identity operators : is, is not :

Membership operators : in, not in

● **Expressions :**

An expression is defined as a combination of constants, variables, and operators. An expression always evaluates to a value. A value or a standalone variable is also considered as an expression but a standalone operator is not an expression.

(i) 100

(iv) 3.0 + 3.14

- (ii) num (v) $23/3 - 5 * 7(14 - 2)$
(iii) num – 20.4 (vi) "Global" + "Citizen"

SUMMARY

The if statement is used for selection or decision making.

- The looping constructs while and for allow sections of code to be executed repeatedly under some condition.
- for statement iterates over a range of values or a sequence.
- The statements within the body of for loop are executed till the range of values is exhausted.
- The statements within the body of a while are executed over and over until the condition of the while is false.
- If the condition of the while loop is initially false, the body is not executed even once.
- The statements within the body of the while loop must ensure that the condition eventually becomes false; otherwise, the loop will become an infinite loop, leading to a logical error in the program.
- The break statement immediately exits a loop, skipping the rest of the loop's body.

Execution continues with the statement immediately following the body of the loop. When a continue statement is encountered, the control jumps to the beginning of the loop for the next iteration.

- A loop contained within another loop is called a nested loop.

MCQ QUESTIONS:

1. The numbered position of a letter in a string is called _____.
(a) position
(b) integer position
(c) index
(d) location
2. The operator _____ tells if an element is present in a sequence or not.
(a) exists
(b) in
(c) into
(d) inside
3. The keys of a dictionary must be of _____ types.
(a) Integer

- (b) mutable
- (c) immutable
- (d) any of these

4. Following set of commands is executed in shell, what will be the output?

```
>>>str = "hello"
```

```
>>>str[ : 2]
```

```
>>>
```

- (a) he
- (b) lo
- (c) olleh
- (d) hello

5. What data type is the object below?

```
L = [1, 23, 'hello', 1]
```

- (a) list
- (b) dictionary
- (c) array
- (d) tuple

6. What data type is the object below?

```
L = 1, 23, 'hello', 1
```

- (a) list
- (b) dictionary
- (c) array
- (d) tuple

7. To store values in terms of key and value, what core data type does Python provide?

- (a) list
- (b) tuple
- (c) class
- (d) dictionary

8. What is the value of the following expression?

```
3 + 3.00, 3**3.0
```

- (a) (6.0, 27.0)
- (b) (6.0, 9.00)
- (c) (6, 27)

(d) [6.0, 27.0]

(e) [6, 27]

9. List AL is defined as follows:

```
AL = [1, 2, 3, 4, 5]
```

Which of the following statements removes the middle element 3 from it so that the list AL equal [1, 2, 4, 5]?

(a) `del a[2]`

(b) `a[2 : 3] = []`

(c) `a[2:2] = []`

(d) `a[2] = []`

(e) `a.remove(3)`

10. Which two lines of code are valid strings in Python?

(a) This is a string

(b) 'This is a string'

(c) (This is a string)

(d) "This is a string"

11. You have the following code segment:

```
String1 = "my"
```

```
String2 = "work"
```

```
print(String1 + string2)
```

What is the output of this code?

(a) my work

(b) work

(c) mywork

(d) my

12 You have the following code segment

```
String1 = "my"
```

```
String2 = "work"
```

```
print(String1 + string2.upper())
```

What is the output of this code?

(a) mywork

(b) MY Work

(c) myWORK

(d) My Work

13. Which line of code produces an error?

(a) "one" + 'two'

(b) 1+ 2

(c) "one"+ "2"

(d) '1' + 2

14. What is the output of this code?

```
>>> int("3" + " 4")
```

(a) "7"

(b) "34"

(c) 34

(d) 24

15. Which line of code will cause an error?

1. num = [5, 4, 3, [2], 1]

2. print(num [0])

3. print(num[3][0])

4. print (num[5])

(a) Line 3

(b) Line 2

(c) Line 4

(d) Line 1

16. Which is the correct form of declaration of dictionary?

(a) Day = {1 : 'Monday', 2 : 'Tuesday', 3 : 'wednesday'}

(b) Day = {1 ; 'Monday', 2 ; 'Tuesday', 3 ; 'wednesday'}

(c) Day = [1 : 'Monday', 2 : 'Tuesday', 3 : 'wednesday']

(d) Day = {1 'Monday', 2 'Tuesday', 3 'wednesday'}

17. Identify the valid declaration of L:

```
L = [1, 23, 'hi', 6]
```

(a) list

(b) dictionary

(c) array

(d) tuple

18. Which of the following is not considered a valid identifier in Python?

- (a) two2
- (b) _main
- (c) hello_rsp1
- (d) 2 hundred

19. What will be the output of the following code-`print("100+200")`?

- (a) 300
- (b) 100200
- (c) 100+200
- (d) 200

20. Which amongst the following is a mutable data type in Python?

- (a) int
- (b) string
- (c) tuple
- (d) list

21. `pow()` function belongs to which library?

- (a) math
- (b) string
- (c) random
- (d) maths

22. Which of the following statements converts a tuple into a list?

- (a) `len(string)`
- (b) `list(string)`
- (c) `tup(list)`
- (d) `dict(string)`

23. The statement: `bval = str i > str2` shall return..... As the output if two strings `str1` and `str2` contains "Delhi" and "New Delhi".

- (a) True
- (b) Delhi
- (c) New Delhi
- (d) False

24. What will be the output generated by the following snippet?

```
a = [5, 10, 15, 20, 25]
```

```
k = 1
i = a[1] + 1
j = a[2] + 1
m = a[k+1]
print (i, j, m)
```

- (a) 11 15 16
- (b) 11 16 15
- (c) 11 15 15
- (d) 16 11 15

25. The process of arranging the array elements in a specified order is termed as:

- (a) Indexing
- (b) Slicing
- (c) Sorting
- (d) Traversing

26. Which of the following Python functions is used to iterate over a sequence of number by specifying a numeric end value within its parameters?

- (a) range()
- (b) len()
- (c) substring()
- (d) random()

27. What is the output of the following?

```
d = {0: 'a', 1: 'b', 2: 'c'}
```

```
for i in d:
```

```
    print(i)
```

- (a)
0
1
2
- (b)
a
b
c
- (c)

0
a
1
b
2
c
(d)

2
a
2
b
2
c

28. What is the output of the following?

```
x = 123
```

```
for i in x:
```

```
    print(i)
```

(a) 1 2 3
(b) 123
(c) error
(d) infinite loop

29. Which arithmetic operators cannot be used with strings?

(a) +
(b) *
(c) -
(d) all of the above

30. What will be the output when the following code is executed?

```
>>> str1="helloworld"
```

```
>>> str1[::-1]
```

(a) dlrowolleh
(b) hello
(c) world
(d) helloworld

31. What is the output of the following statement?

```
print ("xyyzxyzxzyy".count ('yy', 1))
```

- (a) 2
- (b) 0
- (c) 1
- (d) Error

32. Suppose list1 [0.5 * x for x in range(0, 4)], list1 is:

- (a) [0, 1, 2, 3]
- (b) [0, 1, 2, 3, 4]
- (c) [0.0, 0.5, 1.0, 1.5]
- (d) [0.0, 0.5, 1.0, 1.5, 20]

33. Which is the correct form of declaration of dictionary?

- (a) Day={1:'monday',2:'tuesday',3:'wednesday'}
- (b) Day=(1;'monday',2;'tuesday',3;'wednesday')
- (c) Day=[1:'monday',2:'tuesday',3:'wednesday']
- (d) Day={1 monday',2 tuesday',3 wednesday'}

34. Identify the valid declaration of L: L = [1, 23, 'hi', 6]

- (a) list
- (b) dictionary
- (c) array
- (d) tuple

Answer =

- 1. c
- 2. b
- 3. c
- 4. a
- 5. a
- 6. d
- 7. d
- 8. a
- 9. a, b, e
- 10. b, d
- 11. c

- 12. c
- 13. d
- 14. c
- 15. c
- 16. a
- 17. a
- 18. d
- 19. b
- 20. d
- 21. a
- 22. b
- 23. d
- 24. b
- 25. c
- 26. a
- 27. a
- 28. c
- 29. c
- 30. a
- 31. a
- 32. c
- 33. a
- 34. a

Output Questions:

1. What is the output of the following? x = ['ab', 'cd']

```
for i in x: i.upper()
```

```
    print(x)
```

- a) ['ab', 'cd'].
- b) ['AB', 'CD'].
- c) [None, None].
- d) none of the mentioned

2. What is the output of the following? x = ['ab', 'cd']

```
for i in x: x.append(i.upper())  
    print(x)
```

- a) ['AB', 'CD'].
- b) ['ab', 'cd', 'AB', 'CD'].
- c) ['ab', 'cd'].
- d) none of the mentioned

3. What is the output of the following? i = 1

```
while True:  
    if i%3 == 0: break
```

```
print(i)
```

```
i += 1
```

- a) 1 2
- b) 1 2 3
- c) error
- d) none of the mentioned

4. What is the output of the following? i = 1

```
while True:
```

```
    if i%007 == 0:
```

```
        break print(i)
```

```
    i += 1
```

- a) 1 2 3 4 5 6
- b) 1 2 3 4 5 6 7
- c) error
- d) none of the mentioned

5. What is the output of the following?

```
    i = 5
```

```
    while True:
```

```
        if i%0011 == 0:
```

```
            break print(i)
```

```
        i += 1
```


a) 5 6 7 8 9 10

b) 5 6 7 8

c) 5 6

d) error

6. What is the output of the following? i = 5

```
while True:
```

```
    if i%009 == 0:
```

```
        break print(i)
```

```
    i += 1
```

a) 5 6 7 8

b) 5 6 7 8 9

c) 5 6 7 8 9 10 11 12 13 14 15

d) error

7. What is the output of the following? $i = 1$

```
while True:
```

```
    if  $i \% 2 == 0$ : break
```

```
print(i) i += 2
```

a) 1

b) 1 2

c) 1 2 3 4 5 6 ...

d) 1 3 5 7 9 11 ...

8. What is the output of the following? $i = 2$

```
while True:
```

```
    if  $i \% 3 == 0$ : break
```

```
print(i) i += 2
```

a) 2 4 6 8 10 ...

b) 2 4

c) 2 3

d) error

9. What is the output of the following? i = 1

```
while False:
```

```
    if i%2 == 0: break
```

```
print(i) i += 2
```

- a) 1
- b) 1 3 5 7 ...
- c) 1 2 3 4 ...
- d) none of the mentioned

10. What is the output of the following?

```
True = False while
```

```
    True:
```

```
print(True) break
```

- a) True
- b) False
- c) None
- d) none of the mentioned

11. What is the output of the following? i = 0

```
while i < 5: print(i)
```

```
    i += 1
```

```
        if i == 3:
```

```
            break
```

```
    else:
```

```
print(0)
```

- a) 0 1 2 0
- b) 0 1 2
- c) error
- d) none of the mentioned

12. What is the output of the following? i = 0

```
while i < 3: print(i)
```

```
    i += 1
```

else:

print(0)

- a) 0 1 2 3 0
- b) 0 1 2 0
- c) 0 1 2
- d) error

13. What is the output of the following?

```
x = "abcdef" while i in
```

```
x:
```

```
print(i, end=" ")
```

- a) a b c d e f
- b) abcdef
- c) i i i i i ...
- d) error

14. What is the output of the following?

```
x = "abcdef" i = "i"
```

```
while i in x: print(i, end=" ")
```

- a) no output
- b) i i i i i ...
- c) a b c d e f
- d) abcdef

15. What is the output of the following? x = 'abcd'

```
for i in x: print(i.upper())
```

- a) a b c d
- b) A B C D
- c) a B C D
- d) error

16. What is the output of the following? x = 'abcd'

```
for i in range(len(x)): i.upper()
```

```
print (x)
```

- a) a b c d

b) 0 1 2 3

c) error

d) none of the mentioned

17. What is the output of the following? x = 'abcd'

```
for i in range(len(x)): x = 'a'
```

```
print(x)
```

a) a

b) abcd abcd abcd

c) a a a a

d) none of the mentioned

18. What is the output of the following? x = 'abcd'

```
for i in range(len(x)): print(x)
```

```
x = 'a'
```

a) a

b) abcd abcd abcd abcd

c) a a a a

d) none of the mentioned

19. What is the output of the following? x = 123

```
for i in x: print(i)
```

a) 1 2 3

b) 123

c) error

d) none of the mentioned

20. What is the output of the following? d = {0:

```
'a', 1: 'b', 2: 'c'}
```

```
for i in d: print(i)
```

a) 0 1 2

b) a b c

c) 0 a 1 b 2 c

d) none of the mentioned

ANSWERS:

Answer 1: a

Explanation: The function upper() does not modify a string in place, it returns a new string which isn't being stored anywhere

Answer 2: d

Explanation: The loop does not terminate as new elements are being added to the list in each iteration.

Answer 3: c

Explanation: SyntaxError, there shouldn't be a space between + and = in +=. Answer 4: a

Explanation: Control exits the loop when i become Answer 5:

b

Explanation: 0011 is an octal number. Answer6:

d

Explanation: 9 isn't allowed in an octal number. Answer 7: d

Explanation: The loop does not terminate since i is never an even number. Answer 8: b

Explanation: The numbers 2 and 4 are printed. The next value of i is 6 which is divisible by 3 and hence control exits the loop

Answer 9: d

Explanation: Control does not enter the loop because of False.. Answer 10 :

d

Explanation: SyntaxError, True is a keyword and it's value cannot be changed. Answer 11: b

Explanation: The else part is not executed if control breaks out of the loop. Answer 12: b

Explanation: The else part is executed when the condition in the while statement is false.

Answer 13: d

Explanation: NameError, i is not defined. Answer 14: a

Explanation: "i" is not in "abcdef".

Answer 15: b

Explanation: The instance of the string returned by upper() is being printed. Answer 16 : c

Explanation: Objects of type int have no attribute upper(). Answer 17: c

Explanation: range() is computed only at the time of entering the loop. Answer 18 : d

Explanation: abcd a a a is the output as x is modified only after 'abcd' has been printed once.

Answer 19: c

Explanation: Objects of type int are not iterable. Answer 20: a

Explanation: Loops over the keys of the dictionary.

3 MARK QUESTIONS

Q1. Differentiate between break and continue statement used in python.

Q2. What is comment in python? Explain its significance.

Q3. Explain the types of errors occurring in python programming language.

ANSWER OF 3 MARK QUESTIONS

1) The break statement terminates the current loop, i.e. the loop in which it appears, and resumes execution at the next statement immediately after the end of that loop. If the break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.

When a continue statement is encountered, the control jumps to the beginning of the loop for the next iteration, thus skipping the execution of statements inside the body of the loop for the current iteration. As usual, the loop condition is checked to see if the loop should continue further or terminate. If the condition of the loop is entered again, else the control is transferred to the statement immediately following the loop.

2) Comments in Python are identified with a hash symbol, #, and extend to the end of the line. Hash characters in a string are not considered comments, however. There are three ways to write a comment - as a separate line, beside the corresponding statement of code, or as a multi-line comment block.

here are multiple uses of writing comments in Python. Some significant uses include:

- Increasing readability
- Explaining the code to others
- Understanding the code easily after a long-term
- Including resources
- Re-using the existing code

3) There are three types of Python errors.

1. Syntax errors

Syntax errors are the most basic type of error. They arise when the Python parser is unable to understand a line of code. Syntax errors are almost always fatal, i.e. there is almost never a way to successfully execute a piece of code containing syntax errors.

2. Logical errors

These are the most difficult type of error to find, because they will give unpredictable results and may crash your program. A lot of different things can happen if you have a logic error.

3. Run time errors

Run time errors arise when the python knows what to do with a piece of code but is unable to perform the action. Since Python is an interpreted language, these errors will not occur until the flow of control in your program reaches the line with the problem. Common example of runtime errors are using an undefined variable or mistyped the variable name.

4 MARK QUESTIONS

Q1. Differentiate between type conversion and type casting in python with examples.

Q2. Explain mutable and immutable objects in python with examples.

Q3. What is the use of else statement in for loop and in while loop ? Explain.

ANSWER OF 4 MARK QUESTIONS

1. Type Conversion

In type conversion, the python interpreter automatically converts one data type to another. Since Python handles the implicit data type conversion, the programmer does not have to convert the data type into another type explicitly.

The data type to which the conversion happens is called the destination data type, and the data type from which the conversion happens is called the source data type.

In type conversion, the destination data of a smaller size is converted to the source data type of larger size. This avoids the loss of data and makes the conversion safe to use.

x = 20

y = 25.5

Z = x + y

Here value in z, int type is converted to float type

Type Casting

n type casting, the programmer has to change the data type as per their requirement manually. In this, the programmer explicitly converts the data type using predefined functions like int(), float(), str(), etc. There is a chance of data loss in this case if a particular data type is converted to another data type of a smaller size.

```
x = 25
```

```
float(x)
```

It converts into float type

2. Mutable in Python can be defined as the object that can change or be regarded as something changeable in nature. Mutable means the ability to modify or edit a value.

Mutable objects in Python enable the programmers to have objects that can change their values. They generally are utilized to store a collection of data. It can be regarded as something that has mutated, and the internal state applicable within an object has changed. Immutable objects in Python can be defined as objects that do not change their values and attributes over time.

These objects become permanent once created and initialized, and they form a critical part of data structures used in Python.

Python is used in numbers, tuples, strings, frozen sets, and user-defined classes with some exceptions. They cannot change, and their values and it remains permanent once they are initialized and hence called immutable.

3. Else with loop is used with both while and for loop. The else block is executed at the end of loop means when the given loop condition is false then the else block is executed.

```
i = 0
```

```
while i<5:
```

```
    i+=1
```

```
    print("i =",i)
```

```
else:
```

```
    print("else block is executed")
```

Explanation

- declare i=0
- we know then while loop is active until the given condition is true. and we check i<5 it's true till the value of i is 4.
- i+=1 increment of i because we don't want to execute the while loop infinite times.
- print the value of i
- else block execute when the value of i is 5.

```
l = [1, 2, 3, 4, 5]
```

```
for a in l:
```

```
    print(a)
```

```
else:
```

```
    print("else block is executed")
```

Explanation

- declare a list l=[1,2,3,4,5]
- for loop print a.
- else block is execute when the for loop is read last element of list.

List

- A Python list is a built-in data structure that can hold an ordered collection of items.
- Lists are mutable, meaning their elements can be changed after the list is created.
- We can create a list in Python by enclosing comma-separated values within square brackets ([]).
- For example:

```
my_list = [1, 2, 3, 4, 5]
```

- create lists and lists with different data types:

```
empty_list = []
```

```
mixed_list = [1, "two", 3.0, True]
```

- Lists support various operations like indexing, slicing, appending, inserting, removing, and more.

Here are some common list operations:

- a) Indexing:

```
print(my_list[0]) # Output: 1
```

- b) Slicing:

```
print(my_list[1:3]) # Output: [2, 3]
```

- c) Appending:

```
my_list.append(6) # Adds 6 to the end of the list
```

- d) Inserting:

```
my_list.insert(0, 0) # Inserts 0 at index 0
```

- e) Removing:

```
my_list.remove(3) # Removes the first occurrence of 3
```

6. Length of a list:

```
print(len(my_list)) # Output: 6
```

List Methods

some common methods associated with Python lists along with examples:

1. `append ()` : Adds an element to the end of the list.

```
my_list = [1, 2, 3]
```

```
my_list.append(4)
```

```
print(my_list) # Output: [1, 2, 3, 4]
```

2. `extend()` : Appends the elements of another list to the end of the current list.

```
my_list = [1, 2, 3]
another_list = [4, 5, 6]
my_list.extend(another_list)
print(my_list) # Output: [1, 2, 3, 4, 5, 6]
```

3. `insert()` : Inserts an element at the specified index.

```
my_list = [1, 2, 3]
my_list.insert(1, 5)
print(my_list) # Output: [1, 5, 2, 3]
```

4. `remove()` : Removes the first occurrence of a specified element.

```
my_list = [1, 2, 3, 2]
my_list.remove(2)
print(my_list) # Output: [1, 3, 2]
```

5. `pop()` : Removes the element at the specified position and returns it. If no index is specified, it removes and returns the last item in the list.

```
my_list = [1, 2, 3]
popped_element = my_list.pop(1)
print(popped_element) # Output: 2
print(my_list) # Output: [1, 3]
```

6. `index()` : Returns the index of the first occurrence of a specified element.

```
my_list = [1, 2, 3, 2]
index = my_list.index(2)
print(index) # Output: 1
```

7. `count()` : Returns the number of occurrences of a specified element.

```
my_list = [1, 2, 3, 2]
count = my_list.count(2)
print(count) # Output: 2
```

8. `sort()` : Sorts the list in ascending order.

```
my_list = [3, 1, 2]
my_list.sort()
print(my_list) # Output: [1, 2, 3]
```

9. `reverse()` : Reverses the elements of the list in place.

```
my_list = [1, 2, 3]
```

```
my_list.reverse()
print(my_list) # Output: [3, 2, 1]
```

10. `copy()` : Returns a shallow copy of the list.

```
my_list = [1, 2, 3]
copy_list = my_list.copy()
print(copy_list) # Output: [1, 2, 3]
```

These are some of the commonly used methods associated with Python lists.

Tuples

Tuples in Python are immutable sequences, which means their elements cannot be changed once assigned. Unlike lists, tuples have only a few built-in methods. Here are the main methods available for tuples along with examples:

1. `count()` : Returns the number of times a specified value occurs in a tuple.

```
my_tuple = (1, 2, 3, 2, 2)
count = my_tuple.count(2)
print(count) # Output: 3
```

2. `index()` : Returns the index of the first occurrence of a specified value. If the value is not found, it raises a `ValueError``.

```
my_tuple = (1, 2, 3, 2)
index = my_tuple.index(2)
print(index) # Output: 1
```

Besides these methods, tuples support various operations and functions due to their immutable nature:

3. `len()` : Returns the number of items in a tuple.

```
my_tuple = (1, 2, 3)
length = len(my_tuple)
print(length) # Output: 3
```

4. `max()` : Returns the largest item in the tuple.

```
my_tuple = (1, 2, 3)
maximum = max(my_tuple)
print(maximum) # Output: 3
```

5. `min()` : Returns the smallest item in the tuple.

```
my_tuple = (1, 2, 3)
minimum = min(my_tuple)
print(minimum) # Output: 1
```

6. `sum()` : Returns the sum of all items in the tuple.

```
my_tuple = (1, 2, 3)
total = sum(my_tuple)
print(total) # Output: 6
```

7. `in` : Checks if an element exists in the tuple.

```
my_tuple = (1, 2, 3)
exists = 2 in my_tuple
print(exists) # Output: True
```

8. `tuple()` : Converts another data type into a tuple.

```
my_list = [1, 2, 3]
my_tuple = tuple(my_list)
print(my_tuple) # Output: (1, 2, 3)
```

9. Slicing : Accesses a range of elements in the tuple.

```
my_tuple = (1, 2, 3, 4, 5)
sliced = my_tuple[1:4]
print(sliced) # Output: (2, 3, 4)
```

10. Concatenation : Combines two or more tuples.

```
tuple1 = (1, 2)
tuple2 = (3, 4)
combined = tuple1 + tuple2
print(combined) # Output: (1, 2, 3, 4)
```

Tuples are especially useful for representing fixed collections of items and are often used in situations where data integrity is important and should not be altered.

- ✓ In Python, tuples are immutable sequences, typically used to store collections of heterogeneous data. They are similar to lists but have a few distinct methods. Here's a description of the primary methods available for tuples, along with examples:

1. count() method

The `count()` method returns the number of times a specified value appears in the tuple.

Syntax:

```
tuple.count(value)
```

Example:

```
# Creating a tuple
```

```
my_tuple = (1, 2, 3, 1, 4, 1, 5)
```

```
# Counting the number of times 1 appears in the tuple
```

```
count_ones = my_tuple.count(1)
```

```
print(count_ones) # Output: 3
```

2. index() method

The `index()` method returns the index of the first occurrence of the specified value. If the value is not found, it raises a `ValueError`.

Syntax:

```
tuple.index(value, start, end)
```

- `value`: The item to search for.

- `start` (optional): The position to start the search.

- `end` (optional): The position to end the search.

Example:

```
# Creating a tuple
```

```
my_tuple = (1, 2, 3, 1, 4, 1, 5)
```

```
# Finding the index of the first occurrence of 1
```

```
index_one = my_tuple.index(1)
```

```
print(index_one) # Output: 0
```

```
# Finding the index of the first occurrence of 1, starting from position 2
```

```
index_one_from_2 = my_tuple.index(1, 2)
```

```
print(index_one_from_2) # Output: 3
```

3. Tuple concatenation

Tuples can be concatenated using the `+` operator to form a new tuple.

Example:

```
# Creating two tuples
```

```
tuple1 = (1, 2, 3)
```



```
tuple2 = (4, 5, 6)
```

```
# Concatenating tuples
```

```
concatenated_tuple = tuple1 + tuple2
```

```
print(concatenated_tuple) # Output: (1, 2, 3, 4, 5, 6)
```

4. Tuple repetition

Tuples can be repeated using the `*` operator to form a new tuple.

Example:

```
# Creating a tuple
```

```
my_tuple = (1, 2, 3)
```

```
# Repeating the tuple
```

```
repeated_tuple = my_tuple * 3
```

```
print(repeated_tuple) # Output: (1, 2, 3, 1, 2, 3, 1, 2, 3)
```

5. Tuple slicing

Tuples can be sliced to extract a portion of the tuple.

Example:

```
# Creating a tuple
```

```
my_tuple = (1, 2, 3, 4, 5, 6)
```

```
# Slicing the tuple from index 1 to 4 (excluding 4)
```

```
sliced_tuple = my_tuple[1:4]
```

```
print(sliced_tuple) # Output: (2, 3, 4)
```

```
# Slicing the tuple with a step
```

```
step_sliced_tuple = my_tuple[0:6:2]
```

```
print(step_sliced_tuple) # Output: (1, 3, 5)
```

6. Tuple unpacking

Tuple unpacking allows assigning the values of a tuple to multiple variables in a single statement.

Example:

```
# Creating a tuple
```

```
my_tuple = (1, 2, 3)
```

```
# Unpacking the tuple into variables
```

```
a, b, c = my_tuple
```

```
print(a) # Output: 1
```

```
print(b) # Output: 2
```

```
print(c) # Output: 3
```

Summary

Tuples in Python have fewer methods than lists due to their immutable nature. The primary methods, `count` and `index`, provide basic functionality for tuple manipulation. Additional operations like concatenation, repetition, slicing, and unpacking enhance their usability. Here's a quick reference to the methods discussed:

- `count()`: Counts occurrences of a value.
- `index()`: Finds the first occurrence index of a value.
- Concatenation (`+`): Combines tuples.
- Repetition (`*`): Repeats tuples.
- Slicing: Extracts portions of a tuple.
- Unpacking: Assigns tuple values to variables.

These methods and operations cover the majority of use cases for working with tuples in Python.

String methods

Python provides a wide array of string methods to perform various operations on strings. Here are some of the most used string methods with examples:

1. `str.upper()`

- Converts all characters in the string to uppercase.

```
text = "hello world"  
print(text.upper()) # Output: HELLO WORLD
```

2. `str.lower()`

- Converts all characters in the string to lowercase.

```
text = "HELLO WORLD"  
print(text.lower()) # Output: hello world
```

3. `str.capitalize()`

- Capitalizes the first character of the string and converts the rest to lowercase.

```
text = "hello world"  
print(text.capitalize()) # Output: Hello world
```

4. `str.title()`

- Capitalizes the first character of each word.

```
text = "hello world"  
print(text.title()) # Output: Hello World
```

5. `str.strip()`

- Removes any leading and trailing whitespace.

```
text = " hello world "  
print(text.strip()) # Output: "hello world"
```

6. str.lstrip() and str.rstrip()

- `lstrip()` removes leading whitespace.

- `rstrip()` removes trailing whitespace.

```
text = " hello world "  
print(text.lstrip()) # Output: "hello world "  
print(text.rstrip()) # Output: " hello world"
```

7. str.replace(old, new)

- Replaces occurrences of a substring with another substring

```
text = "hello world"  
print(text.replace("world", "Python")) # Output: hello Python
```

8. str.find(sub)

- Returns the lowest index of the substring if found, otherwise returns -1.

```
text = "hello world"  
print(text.find("world")) # Output: 6
```

9. str.startswith(prefix) and str.endswith(suffix)

- `startswith(prefix)` checks if the string starts with the specified prefix.

- `endswith(suffix)` checks if the string ends with the specified suffix.

```
text = "hello world"  
print(text.startswith("hello")) # Output: True  
print(text.endswith("world")) # Output: True
```

10. str.split(separator)

- Splits the string into a list of substrings based on the specified separator.

```
text = "hello world"  
print(text.split()) # Output: ['hello', 'world']  
text = "apple,banana,cherry"  
print(text.split(",")) # Output: ['apple', 'banana', 'cherry']
```

11. str.join(iterable)

- Joins the elements of an iterable (e.g., list) into a single string, with the string acting as a separator.

```
words = ["hello", "world"]
```

```
print(" ".join(words)) # Output: hello world
```

12. str.isdigit()

- Returns True if all characters in the string are digits.

```
text = "12345"
```

```
print(text.isdigit()) # Output: True
```

```
text = "123abc"
```

```
print(text.isdigit()) # Output: False
```

13. str.isalpha()

- Returns True if all characters in the string are alphabetic.

```
text = "hello"
```

```
print(text.isalpha()) # Output: True
```

```
text = "hello123"
```

```
print(text.isalpha()) # Output: False
```

14. str.isalnum()

- Returns True if all characters in the string are alphanumeric (either alphabets or digits).

```
text = "hello123"
```

```
print(text.isalnum()) # Output: True
```

```
text = "hello 123"
```

```
print(text.isalnum()) # Output: False
```

15. str.count(sub)

- Returns the number of non-overlapping occurrences of the substring.

```
text = "hello world"
```

```
print(text.count("l")) # Output: 3
```

These methods allow you to manipulate and query strings efficiently in Python.

Dictionary Basics

- Definition: A dictionary is an unordered collection of key-value pairs.
- Syntax: `{key1: value1, key2: value2, ...}`

Creating a Dictionary

```
my_dict = {'name': 'Alice', 'age': 25}
```

Accessing Values

```
name = my_dict['name'] # Access value by key
```

```
age = my_dict.get('age')
```

```
# Access value by key, returns None if key doesn't exist
```

Adding and Updating Entries

```
my_dict['gender'] = 'Female' # Add new entry
```

```
my_dict['age'] = 26 # Update existing entry
```

Removing Entries

```
del my_dict['gender'] # Remove entry by key
```

```
age = my_dict.pop('age') # Remove entry by key and return its value
```

```
item = my_dict.popitem()
```

```
# Remove and return the last inserted key-value pair
```

Python Dictionary methods

1. `dict.get(key[, default])`

Returns the value for the specified key if the key is in the dictionary. If not, it returns the default value (None if not specified).

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
```

```
print(my_dict.get('name')) # Output: Alice
print(my_dict.get('gender', 'Not specified')) # Output: Not specified
```

2. dict.keys()

Returns a view object that displays a list of all the keys in the dictionary.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
print(my_dict.keys()) # Output: dict_keys(['name', 'age'])
```

3. dict.values()

Returns a view object that displays a list of all the values in the dictionary.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
print(my_dict.values()) # Output: dict_values(['Alice', 25])
```

4. dict.items()

Returns a view object that displays a list of dictionary's key-value tuple pairs.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
print(my_dict.items()) # Output: dict_items([('name', 'Alice'), ('age', 25)])
```

5. dict.update([other])

Updates the dictionary with the elements from another dictionary object or from an iterable of key-value pairs.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
my_dict.update({'age': 26, 'gender': 'Female'})
print(my_dict) # Output: {'name': 'Alice', 'age': 26, 'gender': 'Female'}
```

6. dict.pop(key[, default])

Removes the specified key and returns the corresponding value. If the key is not found, the default value is returned if provided; otherwise, a `KeyError` is raised.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
age = my_dict.pop('age')
print(age) # Output: 25
print(my_dict) # Output: {'name': 'Alice'}
```

7. dict.popitem()

Removes and returns a (key, value) pair from the dictionary. Pairs are returned in LIFO (last-in, first-out) order.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
item = my_dict.popitem()
print(item) # Output: ('age', 25)
print(my_dict) # Output: {'name': 'Alice'}
```

8. dict.setdefault(key[, default])

If the key is in the dictionary, it returns the value. If not, inserts the key with a specified default value and returns that value.

Example:

```
my_dict = {'name': 'Alice'}
age = my_dict.setdefault('age', 25)
print(age) # Output: 25
print(my_dict) # Output: {'name': 'Alice', 'age': 25}
```

9. dict.clear()

Removes all items from the dictionary.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
my_dict.clear()
print(my_dict) # Output: {}
```

10. dict.copy()

Returns a shallow copy of the dictionary.

Example:

```
my_dict = {'name': 'Alice', 'age': 25}
new_dict = my_dict.copy()
print(new_dict) # Output: {'name': 'Alice', 'age': 25}
```

11. dict.fromkeys(iterable, value=None)

Creates a new dictionary with keys from the specified iterable and values set to a specified value (default is None).

Example:

```
keys = ['name', 'age', 'gender']  
default_dict = dict.fromkeys(keys, 'Unknown')  
print(default_dict)  
  
# Output: {'name': 'Unknown', 'age': 'Unknown', 'gender': 'Unknown'}
```

These are some of the most commonly used dictionary methods in Python along with examples of how to use them.

Python modules:

- The act of partitioning a program into individual components is called "Modularity".
- In Python, a module is a file containing Python code that defines variables, functions, and classes.
- Modules allow you to organize and reuse code by breaking it into separate files, making it easier to maintain and understand complex programs.
- Python's standard library comes with a vast collection of built-in modules that cover various functionalities
- If needed, you can also create your own custom modules.
- A module is a separately saved unit whose functionality can be reused.
- A Python module has the .py extension.
- A Python module can contain objects like docstrings, variables, constants, classes, objects, statements, functions etc.
- The Python modules that come preloaded with Python are called "standard library modules".
- Python module can be imported in a program using import statement.
- There are two forms of import statements:

(i) `import <module name>`

(ii) `from <module name> import <object>`

- To use a module in your Python code, you need to import it using the import statement.

math module:

- The math module in Python is a built-in module that provides various mathematical functions and constants.
- It is part of the Python Standard Library i.e. it does not require any additional installation to use.
- To use the math module, you need to import it at the beginning of your Python script.

```
import math
```

- Once you've imported the module, you can access its functions and constants using the math prefix.

Here are some commonly used functions and constants provided by the math module:

Mathematical Constants:

- `math.pi`: Represents the mathematical constant π (pi).
 - `math.e`: Represents the mathematical constant e (Euler's number).
- Mathematical Functions:
- `math.sqrt(x)`: Returns the square root of x .
 - `math.pow(x, y)`: Returns x raised to the power y .
 - `math.exp(x)`: Returns the exponential of x (e^x).
 - `math.log(x, base)`: Returns the logarithm of x to the specified base (default base is e).
- Trigonometric Functions (all angles are in radians):
- `math.sin(x)`, `math.cos(x)`, `math.tan(x)`: Sine, cosine, and tangent of x , respectively.
 - `math.asin(x)`, `math.acos(x)`, `math.atan(x)`: Arcsine, arccosine, and arctangent of x , respectively.
- Hyperbolic Functions:
- `math.sinh(x)`, `math.cosh(x)`, `math.tanh(x)`: Hyperbolic sine, cosine, and tangent of x , respectively.
- Angular Conversion:
- `math.degrees(x)`: Converts x from radians to degrees.
 - `math.radians(x)`: Converts x from degrees to radians.
- Miscellaneous:
- `math.ceil(x)`: Returns the smallest integer greater than or equal to x .
 - `math.floor(x)`: Returns the largest integer less than or equal to x .
 - `math.factorial(x)`: Returns the factorial of x . Study the following examples:

Example 1:

```
import math

print(math.sqrt(25))          # Output: 5.0
print(math.sin(math.pi/2))  # Output: 1.0
print(math.degrees(math.atan(1))) # Output: 45.0 (angle in degrees)
```

Example 2:

```
x = 3.7
rounded_up = math.ceil(x)
rounded_down = math.floor(x)

print("Rounded up:", rounded_up) # Output: 4
print("Rounded down:", rounded_down) # Output: 3
```

Example 3:

```
import math

print("Value of π (pi):", math.pi) # Output: 3.141592653589793
print("Value of e (Euler's number):", math.e) # Output: 2.718281828459045
```

Statistics module:

- The statistics module in Python is another built-in module that provides functions for working with statistical data.
- It offers a variety of statistical functions to compute measures like mean, median, standard deviation, variance, etc.
- The statistics module is part of the Python Standard Library, so there's no need to install any additional packages to use it.

Here are some commonly used functions provided by the statistics module:

- `statistics.mean(data)`: Calculates the arithmetic mean (average) of the data.
- `statistics.median(data)`: Computes the median value of the data.
- `statistics.mode(data)`: Finds the mode (most common value) in the data.

Example 1:

```
import statistics

data = [2, 4, 6, 8, 10]

median_high = statistics.median_high(data)
median_low = statistics.median_low(data)

print("Median High:", median_high) # Output: 6
print("Median Low:", median_low) # Output: 6
```

Example 2:

```
import statistics
data = [2, 3, 3, 4, 4, 4, 5, 5, 5, 5]

mean_value = statistics.mean(data)
mode_value = statistics.mode(data)

print("Mean:", mean_value) # Output: 4.0
print("Mode:", mode_value) # Output: 5 (Note: 5 is the most common value, it appears 4 times)
```

random module

- The random module in Python is another built-in module that provides functions for generating random numbers, sequences, and making random choices.
- It is commonly used for tasks such as random number generation, random shuffling, and random sampling.

```
import random
|
```

Here are some commonly used functions provided by the random module:

- `random.random()`: Generates a random float number in the range [0.0, 1.0).
- `random.uniform(a, b)`: Generates a random float number in the range [a, b).
- `random.randint(a, b)`: Generates a random integer in the range [a, b] (inclusive).
- `random.choice(sequence)`: Picks a random element from a sequence (list, tuple, string, etc.).
- `random.shuffle(sequence)`: Shuffles the elements of a sequence randomly (in-place).

Example 1:

What is the possible outcome/s of following code?

```
import random

colors = ['red', 'green', 'blue', 'yellow', 'purple', 'orange']
random_sample = random.randint(3,5)
print(colors[random_sample])
```

Possible options:

- a) green
- b) yellow
- c) blue
- d) orange

Solution:

Here, the possible values for variable random-sample are 3, 4 and 5. Hence, the possible Outputs of above code are b) Yellow and d) orange.

Example 2:

What are the possible output/s for the following code?

```
import random
low=25
point =5
for i in range (1,5):
    Number=low + random.randint(0,point)
    print(Number,end=" : ")
    point-=1;
print()
```

Output Options:

- i. 29: 26:25 :28 :
- ii. 24: 28:25:26:
- iii. 29: 26:24 :28 :
- iv. 29: 26:25:26:

Solution: Option iv

Example 3:

What are the possible outcome/s for the following code:

```
import random
LIMIT = 4
Points = 100 +random.randint(0,LIMIT);
for p in range(Points,99,-1):
    print(p,end="#")
print()
```

Output Options:

i. 103#102#101#100#

ii. 100#101#102#103#

iii. 100#101#102#103#104#

iv. 4#103#102#101#100#

Solution:

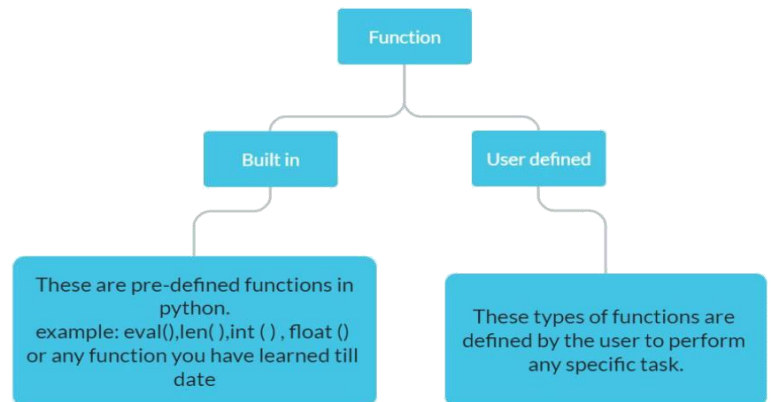
Option i and option iv

Functions

- A function is a named block of statements that can be invoked by its name.

OR

Reusable block of code that performs a specific task. For example: len(), print(), min(), max(), sorted () , type() etc.



- A function is a block of organized and reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusability.
- Function blocks begin with the keyword def followed by the function name and parentheses ().
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement-the documentation string of the function or docstring.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exit a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.
- Defining a function only gives a name, specifies the parameters that are to be included in the function, a structure the blocks of code.
- The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python :
 1. Global variables
 2. Local variables
- Variables that are defined inside a function body have a local scope, and those defined outside

have a global scope.

Types of Functions

- Examples of Some Built-in Functions
 - (i) `print()`: It prints objects to the text stream file.
 - (ii) `input()`: It reads the input, converts it to a string and returns that.
 - (iii) `sorted()`: Returns a new sorted list from the items in iterable..
 - (iv) `bool()`: Returns a boolean value i.e., True or False..
 - (v) `min()`: Returns the smallest of two or more arguments.
 - (vi) `any()`: Returns True if any element of the iterable is True.
- String Functions
 - (i) `partition()`: It splits the string at the first occurrence of the given argument and returns a tuple containing three parts.
 - (ii) `join()`: It takes a list of string and joins them as a regular string.
 - (iii) `split()`: It splits the whole string into the items with separator as a delimiter.
 - (iv) User-Defined Functions: User defined functions are those that we define ourselves in our program and then call them wherever we want.

Defining a function in Python:

Name the function and specifies what to do when the function is called. Python interpreter ignores the function definition until the function is called.

Calling a function:

Calling the function actually performs the specified actions with the indicated parameters

Function Definition in Python

In Python a function is defined using the `def` keyword

```
def func():  
  
    print("Hello")  
  
func()
```

- Arguments: Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
- Actual Parameters (Arguments) are values supplied to the function when it is

invoked/called

- Formal Parameters are variables declared by the function that get values when the function is called.

Example 1:

Observe the following code:

```
def func1(x):# Funtion Definition

    print(x)

func1("First call") #calling function

func1("Second call") #calling function
```

Output:

First call

Second call

In the above example, a user defined function "function1" has been defined that receives one argument. Once the function is defined, it can be called any number of times with different arguments.

Formal argument: x

Actual argument:

"first call to function " passed in first call "second call to function" passed in second call

Example 2: Write a function ADD(A,B) that receives two integer arguments and prints their sum.

```
def ADD(A,B): # function definition
    print(A+B)
ADD(7,9) # function call
ADD(10,15)#function call
ADD(20,25)#function call
```

Output:

```
-----
16
25
45
```

return keyword:

In Python, the `return` keyword is used in functions to specify the value that the function

will return when it is called. When a function is executed, it may perform some computations or operations, and the result can be sent back to the caller using the `return` statement.

The basic syntax for using the `return` statement is as follows:

```
def my_function(arguments):  
    # Code inside the function  
    # ...  
    return value_to_be_returned
```

Here's what you need to know about the `return` statement:

1. Returning a Value:

When you want to return a specific value from the function, you can use the `return` statement followed by the value you want to return.

Note: The function will stop executing immediately after the `return` statement is encountered, and the value will be passed back to the caller.

```
def add(a, b):  
    return a + b  
  
result = add(5, 3) # The function returns 8, and the value is assigned to the "result" variable.
```

2. Returning Multiple Values:

Python allows you to return multiple values from a function as a tuple. You can simply separate the values with commas after the `return` statement.

```
def cord():  
  
    a=10  
  
    b=20  
  
    return a, b  
  
x, y = cord()
```

3. Returning None:

If a function doesn't have a `return` statement or has a `return` statement without any value, it implicitly returns `None`.

`None` is a special constant in Python that represents the absence of a value.

```
def task():  
  
    print("Do task")
```

```
result=task()
```

```
print(result) #This will print "None"
```

Early Exit with Return:

You can use the `return` statement to exit a function early if certain conditions are met.

This is useful when you want to terminate the function before reaching the end.

```
def div(a,b):
```

```
    if b==0:
```

```
        return "Cannot divide by zero!"
```

```
    return a/b
```

```
result1=div(10,2) # returns 5.0
```

```
result2=div(10,2) #returns "Cannot divide by zero!"
```

The `return` statement is a powerful tool that enables functions to produce results and pass data back to the calling code. Understanding how to use it correctly will help you design and implement effective functions in Python.

Scope of a variable:

In Python, the scope of a variable refers to the region of the program where the variable is accessible. The scope determines where a variable is created, modified, and used.

Global Scope:

- Variables defined outside of any function or block have a global scope.
- They are accessible from anywhere in the code, including inside functions.
- To create a global variable, you define it at the top level of your Python script or module.

```
X=10 #Global variable
```

```
def f():
```

```
    print(x) #accessing the global variable inside the function.
```

```
def f1():
```

Print(x) #accessing the global variable inside the function.

f() #Output : 10

f1() #Output : 10

Local Scope:

- Variables defined inside a function have a local scope.
- They are accessible only within the function where they are defined.
- Local variables are created when the function is called and destroyed when the function returns.

def f():

```
    y=5    #Local variable
```

```
    print(y)
```

f() #output: 5

#y is not accessible outside the function

print(y) will result in an error

Points to be noted:

- When local variable and global variable have different names: global variable can be accessed inside the function

```
a=10 #global variable
def fun():
    x=20 #local variable
    y=x+a # global copy of variable a is accessed
    return y
print(fun()) # 30
```

When local and global variable have same name : priority is given to local copy of variable

```
a=10 #global variable
def fun():
    a=20 #local variable
    y=20+a # local copy of variable a is accessed
    return y
print(fun()) # 40
```

Lifetime of a variable:

The lifetime of a variable in Python depends on its scope. Global variables persist

throughout the program's execution, local variables within functions exist only during the function's execution.

Study the following programs:

Example 1:

```
g=100 # global variable

def f1():
    '''any changes made to local copy of variable g will
    not be reflected on global copy of variable g'''

    g=200 # local copy of g will be created, accessible only in f1()
    g=g+10 # local g will be updated
    print(g)
print(g) # global g is accessible
f1()
print(g) # global variable is accessible
```

Example 2:

```
var=40 # global var created that is visible throughout the program

def fun1():
    var=20 #local copy of var is created
    var+=1 # local copy of var is increased by 1
    print(var)
def fun2():
    print(var) #global copy of var will be increased

print(var) # 40
fun1() # 21
print(var) # 40
fun2() # 40
print(var) #40
```

Passing list as argument to the function:

Please note that when a list is passed as arguments, the original copy of list is passed to the function i.e if any change is made at any index in the list inside the function, it is reflected in original list. That is because list is a mutable datatype and in Python, when you pass a list as an argument to a function, you are actually passing a reference to the list rather than a copy of the list. This means that the function parameter will point to the same memory location as the original list. As a result, any changes made to the list within the function will be reflected in the original list outside the function.

```
def modify(list):

    list.append(4)
```

```
list[0]="modified"

my=[1,2,3]

modify(list)

print(list) #output : ['modified', 2,3,4]
```

However, if you assign a different list to a variable inside a function in Python, it will create a new local variable that is separate from any variables outside the function. This local variable will only exist within the scope of the function, and changes made to it won't affect the original list outside the function.

```
def modify_list(some_list):
    some_list = [10, 20, 30] # Assign a new list to the local variable
    print("Inside the function:", some_list)

my_list = [1, 2, 3]
modify_list(my_list)

print("Outside the function:", my_list)
```

Output:

```
=====
['modified', 2, 3, 4]
```

global keyword

In Python, the global keyword is used to indicate that a variable declared inside a function should be treated as a global variable, rather than a local variable. When you assign a value to a variable inside a function, Python, by default, creates a local variable within that function's scope. However, if you need to modify a global variable from within a function, you must use the global keyword to specify that you want to work with the global variable instead.

Here's the basic syntax for using the global keyword:

```
global variable_name
```

For example:

```
g=10

def modify():
```

global g

g=20 #global copy of variable will be modified

modify()

print (g) #output will be 20

Types of arguments passed to a function:

Positional Arguments:

- These are the most common type of arguments and are matched to the function parameters based on their positions. The first argument corresponds to the first parameter, the second argument corresponds to the second parameter, and so on.
- The number and order of positional arguments must match the function's parameter list.

```
def add(a, b):  
    return a + b  
  
result = add(2, 3) # Here, 2 and 3 are positional arguments.
```

Default

Arguments:

- Default arguments are used when a function is called with fewer arguments than there are parameters.
- The default values are specified in the function definition.
- If a value is not provided for a parameter during the function call, the default value is used.

```
def power(base, exponent=2):  
    return base ** exponent  
  
result1 = power(3) # Using the default exponent (2)  
result2 = power(3, 4) # Providing a specific exponent (4)
```

Keyword Arguments:

- In this type, each argument is preceded by a keyword (parameter name) followed by an equal sign.

- The order of keyword arguments does not matter, as they are matched to the function parameters based on their names.
- These arguments provide flexibility to call a function with arguments passed in any order.

```
def add(a, b, c=0):
    return a + b + c

result1 = add(1, 2)
result2 = add(a=1, b=2, c=3)
result3=add(b=10,a=2) # default value of c will be used

print(result1) # Output: 3
print(result2) # Output: 6
print(result3) #output :12
```

OBJECTIVE TYPE QUESTIONS	
1)	<p>Identify the incorrect statement?</p> <p>A) The variables used inside function are called local variables.</p> <p>B) The local variables of a particular function can be used inside other functions, but these cannot be used in global space</p> <p>C) The variables used outside function are called global variables</p> <p>D) In order to change the value of global variable inside function, keyword global is used.</p> <p>Answer:B</p>
2)	<p>When you create your own functions, they are called?</p> <p>A) built-in functions</p> <p>B) user-defined functions</p> <p>C) control functions</p> <p>D) None of the above</p> <p>Answer:B</p>
3)	<p>Which of the following function headers is correct?</p> <p>A) def fun(a = 2, b = 3, c) B) def fun(a = 2, b, c = 3)</p> <p>C) def fun(a, b = 2, c = 3) D) def fun(a, b, c = 3, d)</p> <p>Answer:C</p>
4)	<p>What is a variable defined outside a function referred to as?</p> <p>A) local variable B) global variable C) static Variable D) automatic variable</p> <p>Answer:B</p>

7)	<p>What will be the output of the following code?</p> <pre>def compute_average(numbers): total = 0 count = 0 for num in numbers: total += num count += 1 return total / count marks = [80, 90, 75, 95, 85] average = compute_average(marks) print("Average:", average)</pre> <p>a) Average: 85 b) Average: 85.5 c) Average: 85.0 d) Average: 80, 90, 75, 95, 85</p> <p>Answer:B</p>
----	---

8)	<p>What will be the output of the following code?</p> <pre>def func(x, y=2, z=3): return x + y + z result = func(1, z=4) print(result)</pre> <p>a) 6 b) 8 c) 10 d) 11</p> <p>Answer:C</p>
----	---

9)	<p>What will be the output of the following code?</p> <pre>a = 5 def foo(): global a a += 1 foo() print(a)</pre> <p>a)5 b)6 c) 10 d) NameError: name 'a' is not defined</p> <p>Answer:A</p>
10)	<p>What is the output of the following code snippet?</p> <pre>def foo(x=0): x += 1 return x result = foo(5) + foo(3) print(result)</pre> <p>a) 10 b) 9 c) 8 e) 7</p> <p>Answer:A</p>

Short Answer Type Questions – (1 mark for correct answer)

- Q.2. What do you mean by modularity?
- Q.3. What is a function?
- Q.4. What is function header?
- Q.5. What is `_main_`?
- Q.6. What is a function call?
- Q.7. What are parameters?
- Q.8. What is an argument?
- Q.9. Name the constant available in math module.
- Q.10. Name the categories of functions.
- Q.11. What are docstring conventions?
- Q.12. What is the role of an argument of a function?
- Q.13. What is the general syntax for defining a function in Python?
- Q.14. What are docstrings?
- Q.15. Find the error in the following codes.

```
def plus(total_decrement)
    output=total_decrement
```

Short Long Answer Type Questions – (2 mark for correct answer)

Q.1. What are default arguments?

Ans. Python allows function arguments to have default values; if the function is called without the argument, the argument gets its default value.

Q.2. What do you understand by local and global scope of variables? How can you access a global variable inside the function, if function has a variable with same name.

Ans. A global variable is a variable that is accessible globally. A local variable is one that is only accessible to the current scope, such as temporary variables used in a single function definition. A variable declared outside of the function or in global scope is known as global variable. This means, global variable can be accessed inside or outside of the function where as local variable can be used only inside the function. We can access by declaring variable as global A.

Q. 3. What are the differences between parameters and arguments? Ans.

S.No.	Parameters	Arguments
1	Values provided in function header	Values provided in function call.

2	(eg) def area (r): →r is the parameter	(eg) def main() radius = 5.0 area (radius)
---	---	--

Q. 4. What are keyword arguments?

Ans. If there is a function with many parameters and we want to specify only some of them in function call, then value for such parameters can be provided by using their names instead of the positions.

These are called keyword arguments.

(e.g.) def simpleinterest(p, n=2, r=0.6): def
 simpleinterest(p, r=0.2, n=3):

Q.5. What are the advantages of keyword arguments?

Ans. It is easier to use since we need not to remember the order of the arguments. We can specify the values for only those parameters which we want, and others will have default values.

Q. 6. Differentiate between Built-in functions and user defined functions.

Ans. Built in functions are predefined functions that are already defined in Python and can be used anytime.

e.g. len(), type(), int(), etc.

User defined functions are defined by the programmer.

Q. 7. Differentiate between Built-in functions and functions defined in modules.

Ans. Built-in functions are predefined functions that are already defined in Python and can be used anytime e.g. len(), type(), int() etc. Functions int defined in modules are predefined in modules and can be used only when the corresponding module is imported e.g. to use predefined function sqrt() the math module needs to be imported as import math.

[Long Answer Type Questions - \(3 mark for correct answer\)](#)

Q.1. List a type of arguments and explain any 2 type of arguments.

Ans. Type of arguments:

- Positional arguments
- Default arguments
- Keyword arguments
- Variable length arguments

Keyword arguments :

If there is a function with many parameters and we want to specify only some of them in

function call, then value for such parameters can be provided by using their names instead of the positions.

These are called keyword arguments.

e.g.

```
def simpleinterest(p, n=2, r=0.6):  
    def simpleinterest(p, r=0.2, n=3):
```

Default arguments

Python allows function arguments to have default values; if the function is called without the argument, the argument gets its default value.

e.g.

```
def add(a,b=0):  
    def mul(a=1,b=1):
```

(Note: Student may explain any 2 type of argument)

Q. 2. Write a method in Python to find and display the prime number between 2 to

N. Pass N as argument to the method. Ans. def

prime(N):

```
    for a in range (2, N):  
        prime = 1  
        for i in range (2, a):  
            if a%i ==0:  
                prime = 0  
        if prime == 1:  
            print (a)
```

OR

def prime(N):

```
    for a in range (2, N):  
        for i in range (2, a):  
            if a%i == 0:  
                break  
        else:  
            print (a) break
```

Q. 3. Write definition of a function

1. OddSum(Numbers) to add Odd values in the list Numbers.
2. EvenSum(Numbers) to add Even values in the list Numbers.

Ans. def OddSum(Numbers):

```
Sum=0
for i in range(len(Numbers)):
    if(Numbers[i]%2!=0):
        Sum+=Numbers[i]
print(Sum)
```

def EvenSum(Numbers):

```
Sum=0
for i in range(len(Numbers)):
    if(Numbers[i]%2==0):
        Sum+=Numbers[i]
print(Sum)
```

Q.4. Define a function overlapping () that takes two lists and returns true if they have at least one member in common, False otherwise.

Ans. def overlapping (list1, list2):

```
l1= len(list1) l2=
len(list2)
flag=False
for i in range (l1):
    for j in range (l2):
        if list1[i]==list2[j]:
            flag=True
return flag
```

Q. 5. Write a Python program to reverse a string. Ans. def

string_reverse(str1):

```
rstr1= ""
index = len(str1)
while index > 0:
    rstr1+= str1[index-1] index =
    index-1
return rstr1 print(string_reverse("1234abcd"))
```

Q. 6. Write a python program to find simple interest using a user defined function with parameters and with return value.

Ans. #Python program to calculate simple interest using function def

```
simpleInterest(P, N, R):  
    SI = (P * N * R)/100  
    return SI  
  
P = float(input("Enter the principal amount : ")) N =  
    float(input("Enter the number of years : ")) R =  
    float(input("Enter the rate of interest : "))  
  
#calculate simple interest by using this formula  
SI = simpleInterest(P, N, R)  
  
#print  
print("Simple interest : ",SI)
```

Q. 7. Explain any three string functions with example?

Ans. 1) **isupper()**:The isupper() method returns True if all the characters are in upper case, otherwise False.

```
txt = "THIS IS NOW!"  
x = txt.isupper() print(x)
```

2) **upper()**: The upper() method returns a string where all characters are in upper case.

```
txt = "Hello my friends" x =  
    txt.upper()  
print(x)
```

3) **isdigit()**: The isdigit() method returns True if all the characters are digits, otherwise False.

```
txt = "50800"  
x = txt.isdigit() print(x)
```

Some Exercise to Students:

1. Write a function myfunc() which accept two numbers as parameter and return the sum of square of first number and square root of second number.

Example: if a=4, b=16, then myfunc(a,b) will return 20

2. What do you mean by default argument in a function? Explain giving example.
3. Explain the two types of functions depending on their return value with example.
4. Write two forms of import statement.
5. What is the difference between Actual Parameter and Formal Parameter? Explain giving example.
6. Write a function which accept a list and multiply its all elements and returns it. If any number is

zero, then it leaves that zero. Example if the list is $L=[1,2,3,5,0,-6,4,5]$, then it returns -3600. Write main function to test the sample list.

7. Write advantages of using functions.
8. Name the python library modules which need to be imported to invoke the following functions:
(i) floor() (ii) randint() (iii) sin()
9. Generate twin prime numbers between 1 and 100. Twin prime numbers are the two numbers whose difference is 2 and both are prime numbers. Like (3,5), (11,13) etc.
10. Write a function midPoint() which accept two points of 2-D and return the midpoint of them. For example mid-point of (2, 4) and (6, 6) is (4, 5). Use the function in main-program to test your Python code on the above values.

Exception Handling

Exceptions are run-time errors which are expected to arrive when some specific situation occur in a program. E.g. ZeroDivisionError, IndexError, FileNotFoundError etc.

The program stops abnormally whenever it encounters a run-time-error, however it is possible to handle this error with some valid codes, and the program can be allowed to run after handling the error. Handling this situation is called as exception handling.

In python we can handle the errors using try-except-finally block as follows:

1. try block: A try block contains the code which is expected to produce the run-time error. Such code may be written inside a try block.
2. After try block, we have one or more except blocks to handle multiple errors distinctly. Error handling code is written inside except blocks. Whenever any error occurs in try block it runs the particular except block.
3. A finally block which is optional is run every time irrespective of whether exception occurs or not.

Exception handling in python:

It involves the use of try and except clause as shown in below example:

```
try:
    print('Below line will throw an exception')
    a= 5/0 #dividing by zero
    print('This code is not printed')
except:
    print('Thrown exception is caught')
    print('Divide by zero error')
finally:
    print('finally runs always')
```

Output:

```
Below line will throw an exception
Thrown exception is caught
Divide by zero error
finally runs always
```

We can have multiple except blocks for each type of error separately:

```

try:
    print('1. Divide by zero')
    print('2. Index error.')
    print('3. Value error.')
    print('4. Any other error.')
    a=int(input('Which type of error you want to try: '))
    if a==1:
        a= 5/0 #zero division error
    elif a==2:
        d=[1,2,3]
        d[3]=10 #index error
    elif a==3:
        a=int('XII') #value error
    else:
        print(b) #b is not defined, name error
except ZeroDivisionError:
    print('ZeroDivisionError')
except IndexError:
    print('IndexError')
except ValueError:
    print('ValueError')
except:
    print('Any other error than above')
finally:
    print('finally runs always')

```

Output:

```

1. Divide by zero
2. Index error.
3. Value error.
4. Any other error.
Which type of error you want to try: 3
ValueError
finally runs always

```

```

===== RESTART: C:/Users/91948/AppData/1
1. Divide by zero
2. Index error.
3. Value error.
4. Any other error.
Which type of error you want to try: 4
Any other error than above
finally runs always

```

Points to remember:

1. A try clause can be followed with multiple except clause having different error names. (named except)

```
except ZeroDivisionError:
    print('ZeroDivisionError')
except IndexError:
    print('IndexError')
```

2. Error name is optional to write in except which is common for all type of errors. (unnamed except)

```
except:
    print('Any other error than above')
```

3. A finally block is always executed.

```
finally:
    print('finally runs always')
```

4. A try must have at least one except block.
5. finally is optional.
6. All exceptions are sub-classes of Exception class.
7. The assert keyword in python is used to detect problems early.
8. We can also have exception argument (object) which is the reference of the raised exception object which describes the exception. It can be converted to str to print the error message.

```
except ZeroDivisionError as e:
    print(str(e))
```

Raising or forcing an exception: we can raise our own exception and also pass a custom message to the except block using raise keyword.

```
try:
    a=int(input('Enter a: '))
    b=int(input('Enter b: '))
    if b==0:
        raise ZeroDivisionError('Value of b could not be Zero')
    print(a/b)
except ZeroDivisionError as e:
    print(str(e))
```

Assert statement: python assert statement is a debugging aid that tests a condition. If the condition is true it does nothing and if the condition is false it raises an AssertionError with an optional error message.

```
a=int(input('Enter a: '))
b=int(input('Enter b: '))
assert b!=0, 'denominator must not be zero'
print(a/b)
```

Output:

```
Enter a: 2
Enter b: 0
Traceback (most recent call last):
  File "C:/Users/91948/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe", line 1, in <module>
    a=int(input('Enter a: '))
    b=int(input('Enter b: '))
    assert b!=0, 'denominator must not be zero'
AssertionError: denominator must not be zero
```

Benefits of exception handling:

1. It separates error handling code from normal code.
2. It clarifies the code and enhance readability.
3. It makes for clear, robust, fault tolerant programs.

Some built-in exceptions:

Sno.	Exception Name	Description
1	EOFError	When reading attempt at the end of a file
2	IOError	When I/O operation fails
3	NameError	When a variable is used without creating/defining
4	IndexError	When invalid index is accessed
5	ImportError	When an import statement fails
6	TypeError	When an operation or function applied to objects of inappropriate types.
7	ValueError	When a function receives inappropriate values
8	ZeroDivisionError	When divided by zero
9	OverflowError	When an arithmetic operation is too large
10	KeyError	When invalid key is accessed in dictionary

File Handling

Text file vs Binary file:

Text file	Binary file
It stores text or strings.	It stores all types of objects.
Stores information in the form of stream of ASCII or UNICODE characters	Stores information in the form of stream of bytes which is in the same format as it is in memory.
Each line is terminated with a special character known as EOL character.(\r\n)	There is no delimiter in binary file.
Some internal translation takes place	No translation takes place.

Regular text files: stores the text in the same form as typed. Newline character ends a line, translation takes place. Have .txt extension.

Delimited text files: a specific character is stored to separate the values e.g. a tab or comma.

TSV: tab separated value. Extension is .txt or .csv

CSV: comma separated values. .csv extension.

.INF files .RTF (rich text format) files are also text files.

File object/ file handle: It is a file object or variable created using open function using which a file is processed.

File opening modes:

Mode	Purpose	Description
w	Write in a text file	Creates a text file (if not existing) and writes into it. If the file exists its old content is replaced with the new one.
r	Reading a text file	Read from a text file, if the file doesn't exist it gives error.
a	Append in a text file	Creates a text file (if not existing) and writes into it. If the file exists its old content is retained and the new content is added at the end.

wb	Write in a binary file	Creates a binary file (if not existing) and writes into it. If the file exists its old content is replaced with the new one.
rb	Reading a binary file	Read from a binary file, if the file doesn't exist it gives error.
ab	Append in a binary file	Creates a binary file (if not existing) and writes into it. If the file exists its old content is retained and the new content is added at the end.
w+	Write as well as read from a text file	Creates a text file (if not existing) and write/read it. If the file exists its old content is replaced with the new one.
r+	read as well as write in a text file	The file is opened only if it exists and read/write can be performed.
a+	Write as well as read from a text file	Creates a text file (if not existing) and write/read it. If the file exists its old content is retained and the new one is added at the end.
wb+	Same as w+ but for binary files	
rb+	Same as w+ but for binary files	
ab+	Same as a+ but for binary files	

Opening a file:

A file can be opened using open function as follows:

```
f=open( 'filename.txt', 'r')
```

Closing a file:

A file is saved only when it is closed.

```
f.close()
```

Using with block to open a file:

A file can be opened using with keyword. It creates a block for the file accessing. The file gets closed automatically when the with block ends.

```
with open('filename.txt','r') as f:
```

```
data=f.read()
print(data)
```

Writing in a text file:

1. Using write() function a single string can be written.
2. Using writelines() function multiple strings can be written. Multiple strings are written in a list.

The delimiter character '\n' is required to be added if the data is required in multiple lines.

```
f=open('myfile.txt','w')
f.write('single string\n')
f.write('single string again\n')
f.writelines(['string1\n','string2\n','string3\n'])
f.close()
```

Reading from a text file:

1. Using read(n) function n characters can be read at a time. If nothing is passed it will read the whole file at a time.

```
with open('myfile.txt','r') as f:
    data=f.read()#full file is read
    print(data)
```

2. Using readline() function a single line can be read at a time.

```
with open('myfile.txt','r') as f:
    firstline=f.readline()
    secondline=f.readline()
    print(firstline)
    print(secondline)
```

3. Using readlines() function all the lines can be read at a time. It returns list of strings/lines.

```
with open('myfile.txt','r') as f:
    alllines=f.readlines()
    print(alllines)
```

4. Using for loop it can read line by line automatically.

```
with open('myfile.txt','r') as f:
    for line in f:
```



```
print(line)
```

End of the file/ EOF: when the pointer reaches to the end of the file while reading in a text file, any further read attempt will give empty string. But no error is given as in case of binary file.

File pointer/cursor: read/write operation is performed at the current position of the file pointer. A file pointer keeps track of the current position in the file where the next read or write operation will be performed. File pointer moves automatically when read or write operation is performed.

Seek() and tell() function:

File.seek(n) sets the cursor position to the nth byte number. Byte number starts from 0.

File.tell() returns the current position of the cursor.

Pickle module: Reading and writing in binary files is performed using pickle.load(object,File) and pickle.dump(File) function.

Creating binary files: program to create binary file:

```
import pickle
f=open('myfile.dat','wb')
n=int(input('Enter the number of records: '))
for i in range(n):
    rno=int(input('Enter rollno: '))
    name=input('Enter name: ')
    clas=input('Enter Class: ')
    record=[rno,name,clas]
    pickle.dump(record,f)
f.close()
```

Reading from a binary file: When it reaches to the end of the file, further read attempt will raise an error / exception. A try-except block is used to handle the error.

Program to read from a binary file.

```
import pickle
f=open('myfile.dat','rb')
```

```

try:
    while True:
        record=pickle.load(f)
        print(record)
except:
    pass
f.close()

```

Searching in a binary file:

```

import pickle
f=open('myfile.dat','rb')
rno=int(input('Enter the rollno to be searched: '))
try:
    found=False
    while True:
        record=pickle.load(f)
        if record[0]==rno:
            print('Search successfull')
            print(record)
            found=True
            break
except:
    pass
if not found:
    print('Roll number not found')
f.close()

```

Modifying a binary file: In order to modify a binary file we open the given file in read mode to read from it. And we open another temp file in which we copy all the records from given file, but the record which is to be modified is first modified and then written in the binary file. Now the original/given file is the one which contains the old data and the temp file is the one which contains the updated data. So we delete the original file and rename the temp file to the original file.

Program to modify a binary file:

```
import pickle
import os
f1=open('myfile.dat','rb')
f2=open('temp.dat','wb')
rno=int(input('Enter the rollno to be modified: '))
try:
    found=False
    while True:
        record=pickle.load(f1)
        if record[0]==rno:
            print('Record found please modify')
            newclass=input('Enter the new class of the student:')
            record[2]=newclass
            found=True
        pickle.dump(record,f2)
except:
    pass
if not found:
    print('Roll number not found')
f1.close()
f2.close()
os.remove('myfile.dat')
os.rename('temp.dat','myfile.dat')
```

Deleting a record from a binary file: We copy all the data from the given file to a temp file except the one which is to be deleted. And then we delete the given file and rename the temp file to the given file.

```
import pickle
import os
f1=open('myfile.dat','rb')
```

```

f2=open('temp.dat','wb')
rno=int(input('Enter the rollno to be deleted: '))
try:
    found=False
    while True:
        record=pickle.load(f1)
        if record[0]==rno:
            print('Record found and deleted.')
            found=True
        else:
            pickle.dump(record,f2)
except:
    pass
if not found:
    print('Roll number not found')
f1.close()
f2.close()
os.remove('myfile.dat')
os.rename('temp.dat','myfile.dat')

```

CSV file: comma separated values file is a text file only in which stores the values separated by comma/delimiter.

Reading/writing in a csv file is used with special objects called as reader and writer and these objects are created for a specific file using the csv module.

Writing in a csv file is done using writer.writerow(record) function.

Reading from a csv file is done using the for loop applied on the reader object which reads each record from the csv file one by one.

Program to create a csv file:

```

import csv
f=open('myfile.csv', 'w',newline='')
w=csv.writer(f)

```

```

n=int(input('Enter the number of students: '))
for i in range(n):
    rno=int(input('Enter roll number:'))
    name=input('Enter name: ')
    clas=input('Enter class: ')
    record=[rno,name,clas]
    w.writerow(record)
f.close()

```

Writing multiple records using writerows() function

```

import csv
f=open('myfile.csv', 'w',newline='')
w=csv.writer(f)
n=int(input('Enter the number of students: '))
allrecords=[]
for i in range(n):
    rno=int(input('Enter roll number:'))
    name=input('Enter name: ')
    clas=input('Enter class: ')
    record=[rno,name,clas]
    allrecords.append(record)
w.writerows(allrecords)
f.close()

```

Reading from a csv file:

```

import csv
f=open('myfile.csv', 'r')
r=csv.reader(f)
for record in r:
    print(record)
f.close()

```

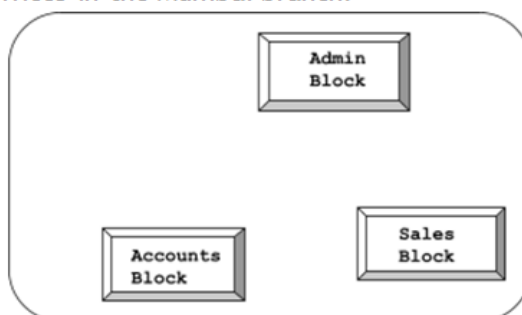
case study based questions

Transmission media +network device

1. Galaxy Provider Ltd. is planning to connect its office in Texas, USA with its branch at Mumbai. The Mumbai branch has 3 Offices in three blocks located at some distance from each other for different operations -ADMIN, SALES and ACCOUNTS.

As a network consultant, you have to suggest the best network related solutions for the issues/problems raised in (a) to (d), keeping in mind the distances between various locations and other given parameters.

Layout of the Offices in the Mumbai branch:



Shortest distances between various locations:

ADMIN Block to SALES Block	300 m
SALES Block to ACCOUNTS Block	175 m
ADMIN Block to ACCOUNTS Block	350 m
MUMBAI Branch to TEXAS Head Office	14000 km

Number of Computers installed at various locations are as follows:

ADMIN Block	255
ACCOUNTS Block	75
SALES Block	30
TEXAS Head Office	90

Q 1 It is observed that there is a huge data loss during the process of data transfer from one block to another. Suggest the most appropriate networking device out of the following, which needs to be placed along the path of the wire connecting one block office with another to refresh the signal and forward it ahead.

- (i) MODEM
- (ii) ETHERNETCARD
- (iii) REPEATER
- (iv) HUB

Ans:- iii

2. Which hardware networking device out of the following, will you suggest to connect all the computers within each block ?

- (i) SWITCH
- (ii) MODEM
- (iii) REPEATER
- (iv) ROUTER

Ans:- (i)

3. Which service/protocol out of the following will be most helpful to conduct live interactions of employees from Mumbai Branch and their counterparts in Texas?

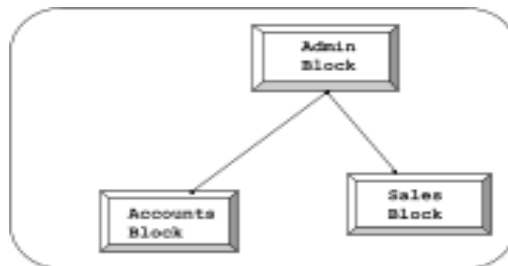
- (i) FTP
- (iii) SMTP

- (ii) PPP
- (iv) VoIP

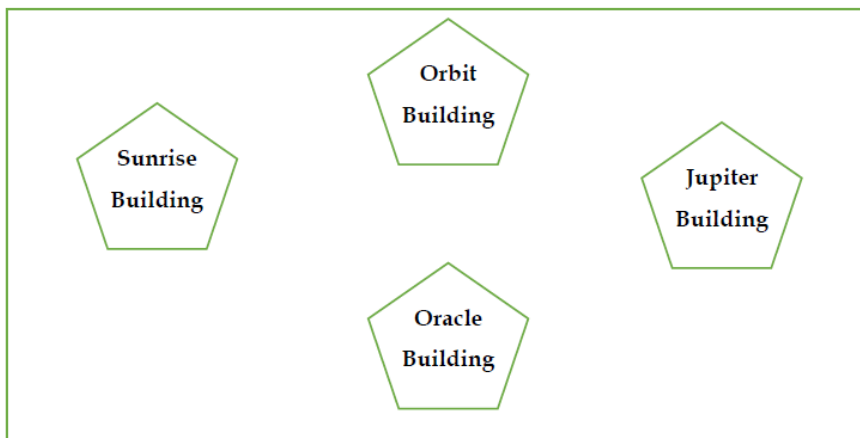
Ans (iv)

4. Draw the cable layout (blocktoblock) to efficiently connect the three offices of the Mumbai branch.

Ans:



2. Aryan Infotech Solutions has set up its new center at Kamla Nagar for its office and web based activities. The company compound has 4 buildings as shown in the diagram below:



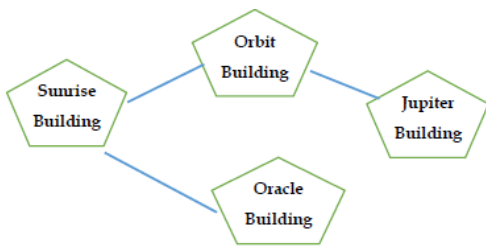
Building to Orbit Building.	50 Mtrs
Orbit Building to Oracle Building	85 Mtrs
Oracle Building to Sunrise Building	25 Mtrs.
Sunrise Building to Jupiter Building	170 Mtrs.
Jupiter Building to Oracle Building	125 Mtrs.
Orbit Building to Sunrise Building	45 Mtrs.
Number of Computers in each of the buildings is follows:	
Jupiter Building	30
Orbit Building	150
Oracle Building	15
Sunrise Building	35

i) Suggest a cable layout of connections between the buildings.

- ii) Suggest the most suitable place (i.e. building) to house the server of this organisation with a suitable reason
- iii) Suggest the placement of the following devices with justification:
 - a. Internet Connecting Device/Modem
 - b. Switch
- iv) 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.
- v) What do you mean by PAN? Explain giving example.

Ans:-

i) suggest a cable layout of connections between the buildings:-



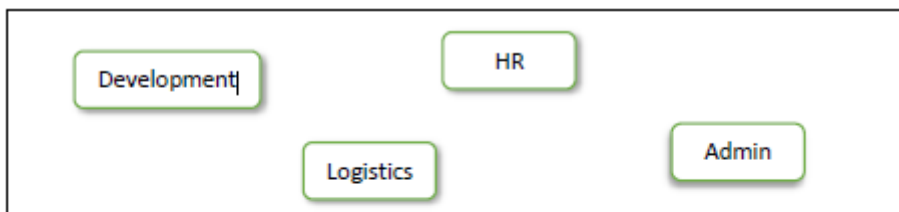
ii) Orbit Building

iii)

- a. Internet Connecting Device/Modem- Orbit Building
- b. Switch- Each Building

iv) MAN, it is formed to connect various locations of the city via various communication media.

3. Magnolia Infotech wants to set up their computer network in the Bangalore based campus having four buildings. Each block has a number of computers that are required to be connected for ease of communication, resource sharing and data security. You are required to suggest the best answers to the questions i) to v) keeping in mind the building layout on the campus.



Number of Computers.

Block	Number of computers
Development	100
HR	120
Admin	200
Logistics	110

Distance Between the various blocks	
Block	Distance
Development to HR	50m
Development to Admin	75m
Development to Logistics	120m

HR to Admin	110m
HR to Logistics	50m
Admin to Logistics	140m

- i) Suggest the most appropriate block to host the Server. Also justify your choice.
- ii) Suggest the device that should be placed in the Server building so that they can connect to Internet Service Provider to avail Internet Services.
- iii) Suggest the wired medium and draw the cable block to block layout to economically connect the various blocks.
- iv) Suggest the placement of Switches and Repeaters in the network with justification.
- v) Suggest the high-speed wired communication medium between Bangalore Campus and Mysore campus to establish a data network

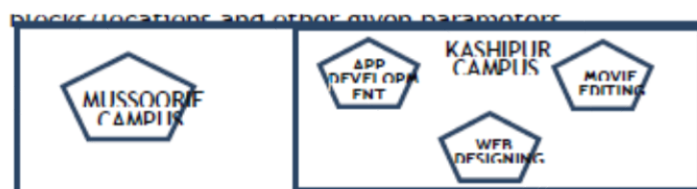
Ans:-

- i) Admin Block since it has maximum number of computers.
- ii) Modem should be placed in the Server building
- iii) The wired medium is UTP/STP cables.



- iv) Switches in all the blocks since the computers need to be connected to the network. Repeaters between Admin and HR block & Admin and Logistics block. The reason being the distance is more than 100m.
- v) Optical Fiber cable connection

4. MakeInIndia Corporation, an Uttarakhand based IT training company, is planning to set up training centres in various cities in next 2 years. Their first campus is coming up in Kashipur district. At Kashipur campus, they are planning to have 3 different blocks for App development, Web designing and Movie editing. Each block has number of computers, which are required to be connected in a network for communication, data and resource sharing. As a network consultant of this company, you have to suggest the best network related solutions for them for issues/problems raised in question nos. (i) to (v), keeping in mind the distances between various blocks/locations:



Block	Distance
App development to Web designing	28 m
App development to Movie editing	55 m
Web designing to Movie editing	32 m
Kashipur Campus to Mussoorie Campus	232 km

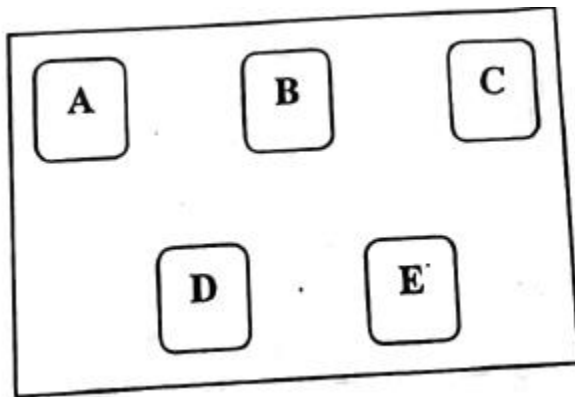
Number of computers	
Block	Number of Computers
App development	75
Web designing	50
Movie editing	80

- (i) Suggest the most appropriate block/location to house the SERVER in the Kashipur campus (out of the 3 blocks) to get the best and effective connectivity. Justify your answer.
- (ii) Suggest a device/software to be installed in the Kashipur Campus to take care of data security.
- (iii) Suggest the best wired medium and draw the cable layout (Block to Block) to economically connect various blocks within the Kashipur Campus.
- (iv) Suggest the placement of the following devices with appropriate reasons:
- Switch / Hub
 - Repeater
- (v) Suggest a protocol that shall be needed to provide Video Conferencing solution between Kashipur Campus and Mussoorie Campus.

Ans:-

- i) Movie editing block is the most appropriate to house the server as it has the maximum number of computers.
- ii) Firewall
- iii) (iii) Ethernet Cable
- (iv) Repeater is not required between the blocks as the distances are less than 100 mts.
- (v) Protocol: VoIP

5. An International Bank has to set up its new data center in Delhi, India. It has 5 five blocks of buildings - A, B, C, D and E.



Distance between the blocks and number of computers in each block areas given below:-

Distance Between Blocks		No of Computers	
Block B to Block C	30m	Block A	55
Block C to Block D	30m	Block B	180
Block D to Block E	35m	Block C	60
Block E to Block C	40m	Block D	55
Block D to Block A	120m	Block E	70
	45m		

Block D to Block B				
Block E to Block B	65m			

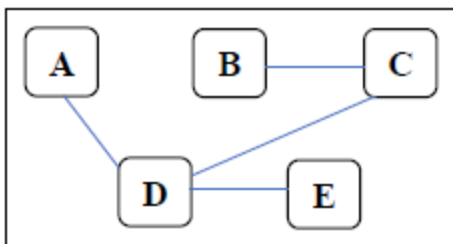
- (i) Suggest the most suitable block to host the server. Justify your answer
- (ii) Draw the cable layout (block to block) to economically connect various blocks within the Delhi campus of International Bank.
- (iii) Suggest the placement of the following devices with justification:
- Repeater
 - Hub/Switch
- (iv) The bank is planning to connect its head office in London. Which type of network out of LAN, MAN, or WAN will be formed? Justify your answer.
- (v) Suggest a device/software to be installed in the Delhi Campus to take care of data security.

Ans:

(i) Block B

Justification- Block B has maximum number of computers. Reduce traffic.

(ii)



(iii) (a) between D and A blocks (b) in all the blocks

(iv) WAN

(v) Firewall

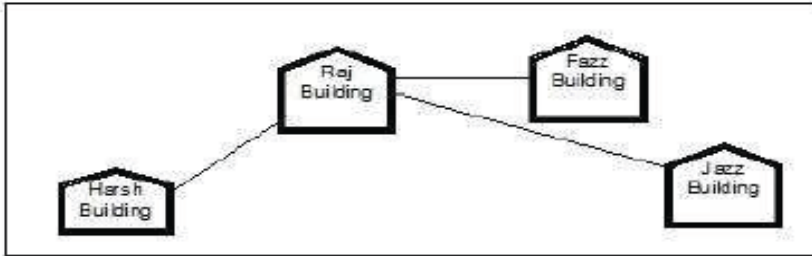
6. Ravya Industries has set up its new center at Kaka Nagar for its office and web based activities. The company compound has 4 buildings as shown in the diagram below:

- (i) Suggest a cable layout of connections between the buildings.
- (ii) Suggest the most suitable place (i.e. building) to house the of this organization with a suitable reason.
- (iii) Suggest the placement of the following devices with appropriate reasons:
- Hub/Switch
 - Repeater
- (iv) 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?

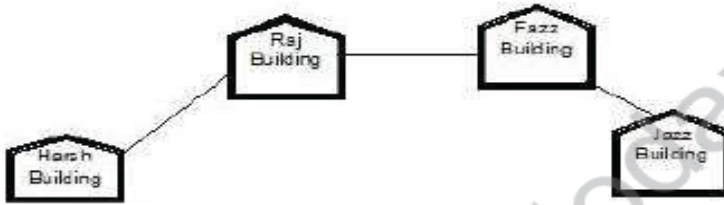
(v) Suggest a device/software to be installed in the Campus to take care of data security.

Ans:-

(i) Any one of the following



Layout option 2



(ii) The most suitable place / block to house the server of this organisation would be Raj Building, as this block contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.

(iii)

- (a) Switch/hub will be placed in all blocks to have connectivity within the block.
- (b) Repeater is not required between the blocks as the distances are less than 100 mts.

(iv) MAN, because MAN (Metropolitan Area Networks) are the networks that link computer facilities within a city.

(v) Firewall

7. Tech Corporation (TTC) is a professional consultancy company. The company is planning to set up their new offices in India with its hub at Hyderabad. As a network adviser, you have to understand their requirement and suggest them the best available solutions. Their queries are mentioned as (i) to (v) below.

Physical locations of the blocks of TTC

Block to block distance (in m)

Block (From)	Block (To)	Distance
Human Resource	Conference	110
Human Resource	Finance	40
Conference	Finance	50

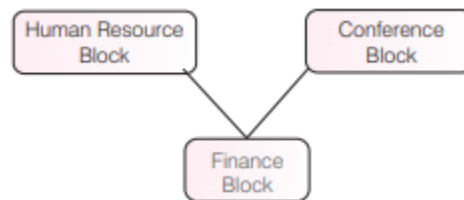
Expected number of computers

Block	Computers
Human Resource	25
Finance	120
Conference	90

- a) Which will be the most appropriate block, where TTC should plan to install their server?
- b) Draw a block to block cable layout to connect all the buildings in the most appropriate manner for efficient communication.
- c) What will be the best possible connectivity out of the following, you will suggest to connect the new setup of offices in Bangalore with its London based office.
- Satellite Link
 - Infrared
 - Ethernet
- d) Which of the following device will be suggested by you to connect each computer in each of the buildings?
- Switch
 - Modem
 - Gateway
- e) Company is planning to connect its offices in Hyderabad which is less than 1 km. Which type of network will be formed?

Ans:-

- (i) TTC should install its server in finance block as it is having maximum number of computers.
- (ii) The layout is based on minimum cable length required, which is 120 metres in the above case.



- (iii) Satellite Link.
- (iv) Switch.
- (v) LAN

8. A company ABC Enterprises has four blocks of buildings as shown

B1

B2

B3

B4

Centre to center distance between various blocks:		Number of computers in each block:	
B3 TO B1	50M	B1	150
B1 TO B2	60M	B2	15
B2 TO B4	25M	B3	15
B4 TO B3	170M	B4	25
B3 TO B2	125M		
B1 TO B4	90M		

Computer sin each block are net worked but blocks are not networked. The company has now decided to connect the blocks also.

- (i) Suggest the most appropriate topology for the connections between the blocks.
- (ii) The company wants internet accessibility in all the blocks. The suitable and cost-effective technology for that would be?
- (iii) Which devices will you suggest for connecting all the computers with in each of their blocks.
- (iv) The company is planning to link its head office situated in New Delhi with the offices in hilly areas. Suggest a way to connect it economically.
- (v) Suggest the most appropriate location of the server, to get the best connectivity for maximum number of computers.

Ans:-

(i) star

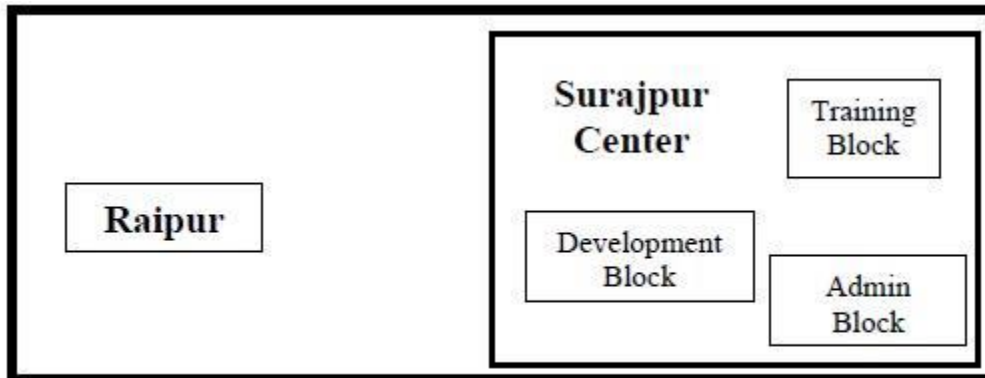
(ii) Broadband

(iii) Switch/Hub

(iv) RadioWave

(v) Block B1

9. FutureTech Corporation, a Bihar based IT training and development company, is planning to set up training centers in various cities in the coming year. Their first center is coming up in Surajpur district. At Surajpur center, they are planning to have 3 different blocks - one for Admin, one for Training and one for Development. Each block has number of computers, which are required to be connected in a network for communication, data and resource sharing. As a network consultant of this company, you have to suggest the best network related solutions for them for issues/problems raised in question nos. (i) to (v), keeping in mind the distances between various blocks/locations and other given parameters.



Distance between various blocks/locations:

Block	Distance
Development to Admin	28 m
Development to Training	105 m
Admin to Training	32 m
Surajpur Campus to Coimbatore Campus	340 km

Number of computers:

Block	Number of Computers
Development	90
Admin	40
Training	50

(i) Suggest the most appropriate block/location to house the SERVER in the Surajpur center (out of the 3 blocks) to get the best and effective connectivity. Justify your answer.

(ii) Suggest why should a firewall be installed at the Surajpur Center?

(iii) Suggest the best wired medium and draw the cable layout(Block to Block) to most efficiently connect various blocks within the Surajpur Center.

(iv) Suggest the placement of the following devices with appropriate reasons:-

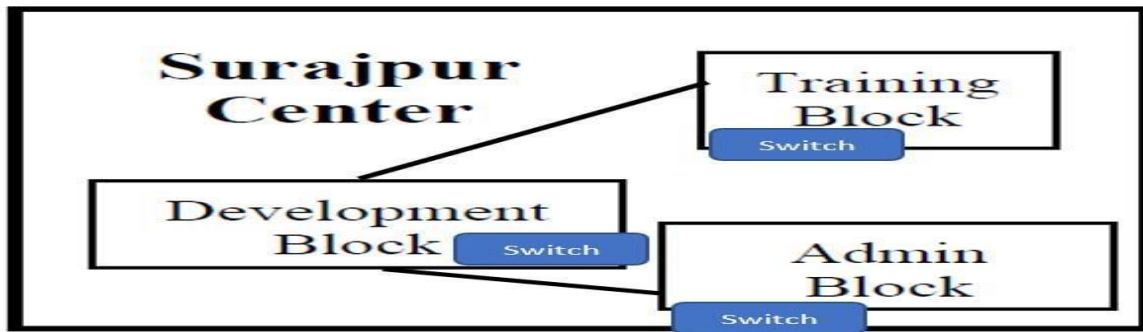
- (a) Switch/Hub
- (b) Router

(v) Suggest the best possible way to provide wireless connectivity between Surajpur Center and Raipur Center.

Ans:-

i) Development because it contains more number of computers

ii) Surajpur centre has multiple blocks and firewall ensures security. So it is required. It allows or block unwanted attacks.



iii)

iv) a) Switch/Hub- In every block to interconnect the devices within every block

b) Router-In development block because server is going to be placed here

v) Satellite

10. Perfect Edu Services Ltd. is an educational organization. It is planning to setup its India campus at Chennai with its head office at Delhi. The Chennai campus has 4 main buildings – ADMIN, ENGINEERING, BUSINESS and MEDIA. 5

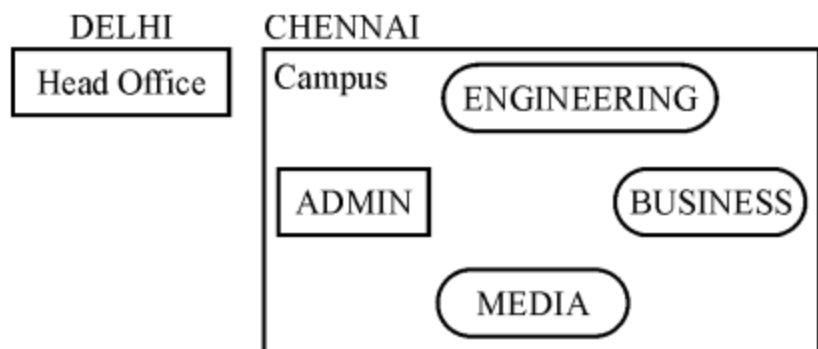
You as a network expert have to suggest the best network related solutions for their problems raised in (i) to (v), keeping in mind the distances between the buildings and other given parameters.

Shortest distances between various buildings :

ADMIN to ENGINEERING	55 m
ADMIN to BUSINESS	90 m
ADMIN to MEDIA	50 m
ENGINEERING to BUSINESS	55 m
ENGINEERING to MEDIA	50 m
BUSINESS to MEDIA	45 m
DELHI Head Office to CHENNAI Campus	2175 km

Number of Computers installed at various buildings are as follows :

ADMIN	110
ENGINEERING	75
BUSINESS	40
MEDIA	12
DELHI Head Office	20



- (i) Suggest the most appropriate location of the server inside the CHENNAI campus (out of the 4 buildings), to get the best connectivity for maximum no. of computers. Justify your answer.
- (ii) Suggest and draw the cable layout to efficiently connect various buildings within the CHENNAI campus for connecting the computers.
- (iii) Which hardware device will you suggest to be procured by the company to be installed to protect and control the internet uses within the campus ?
- (iv) Which of the following will you suggest to establish the online face-to-face communication between the people in the Admin Office of CHENNAI campus and DELHI Head Office ?

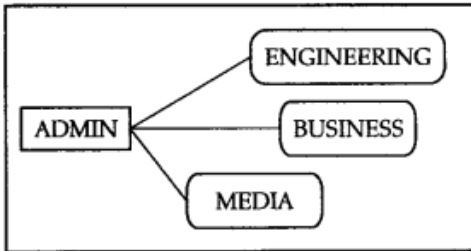
- (a) Cable TV
- (b) Email
- (c) Video Conferencing
- (d) Text Chat

(v) Name protocols used to send and receive emails between CHENNAI and DELHI office?

Ans:-

(i) admin; it contains the max number of systems. to reduce traffic

(ii)



(iii) firewall

(iv) (c) Video Conferencing

(v) POP and SMTP

11. Quick Learn University is setting up its academic blocks at Prayag Nagar and Planning to set up network. The university has 3 academic blocks and one human resource Centre as shown in the diagram given below:



Centre-to-Centre distance between various blocks is as follows:

Law block to business block	40 m
Law block to technology block	80 m
Law block to HR block	105 m
Business block to technology block	30 m
Business block to HR block	35 m
Technology block to HR block	15 m

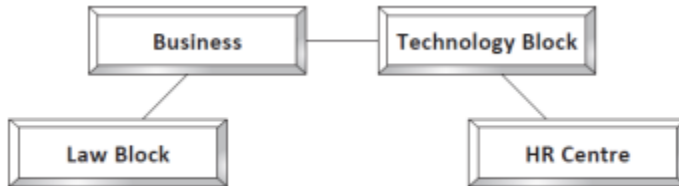
Number of computers in each of the buildings is as follows:

Law block	15
Technology block	40
HR Centre	115
Business block	25

- (a) Suggest a cable layout of connection between the blocks.
- (b) Suggest the most suitable place to house the server of the organization with suitable reason.
- (c) Which device should be placed/installed in each of these blocks to efficiently connect all the computers within these blocks?
- (d) The university is planning to link its sales counters situated in various parts of the CITY. Which type of network out of LAN, MAN or WAN will be formed?
- (e) Which network topology may be preferred between these blocks?

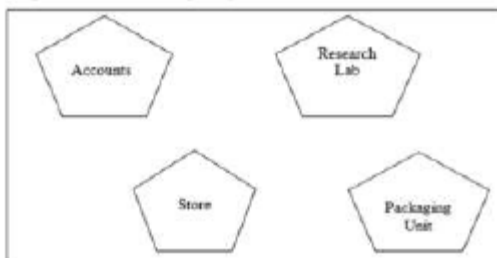
Ans:-

- (a) Suggest a cable layout of connection between the blocks.



- (b) HR centre because it consists of the maximum number of computers to house the server.
- (c) Switch/ Hub should be placed in each of these blocks.
- (d) MAN
- (e) Bus

12. Rehaana Medicos Center has set up its new center in Dubai. It has four buildings as shown in the diagram given below:



Distances between various buildings are as follows:

Accounts to Research Lab	55 m
Accounts to Store	150 m
Store to Packaging Unit	160 m
Packaging Unit to Research Lab	60 m
Accounts to Packaging Unit	125 m
Store to Research Lab	180 m

No of Computers

Accounts	25
Research Lab	100
Store	15
Packaging Unit	60

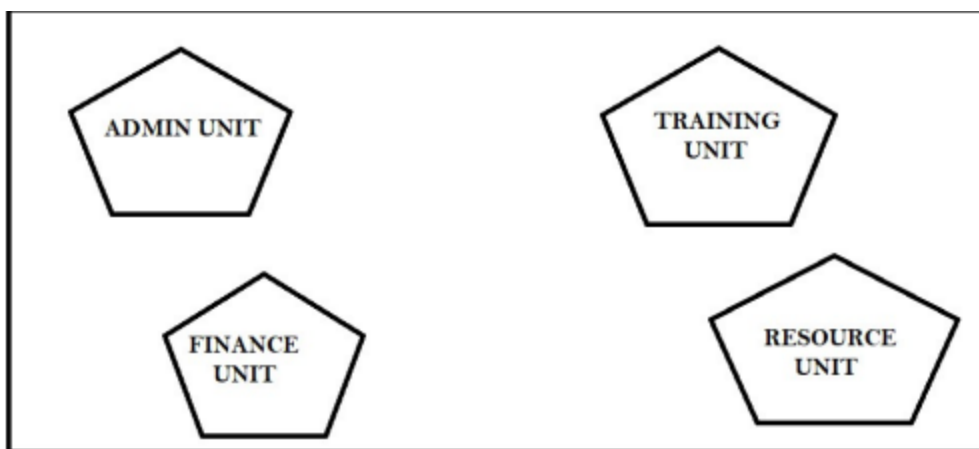
As a network expert, provide the best possible answer for the following queries:

- i) Suggest a cable layout of connections between the buildings.
- ii) Suggest the most suitable place (i.e. buildings) to house the server of this organization.
- iii) Suggest the placement of the Repeater device with justification.
- iv) Suggest a system (hardware/software) to prevent unauthorized access to or from the network.
- v) Suggest the placement of the Hub/ Switch with justification.

Ans:-

- (i) 1 Mark for correct Layout.
- (ii) Research Lab (1 Mark)
- (iii) 1 Mark for correct Justification.
- (iv) Antivirus/ Firewall (1 Mark for Correct Answer)
- (v) 1 Mark for correct Justification

13. "VidyaDaan" an NGO is planning to setup its new campus at Nagpur for its web-based activities. The campus has four(04) UNITS as shown below:



→ Distances between above UNITS are given here s under:

UNIT-1	UNIT-2	DISTANCE(In mtrs.)
ADMIN	TRAINING	65
ADMIN	RESOURCE	120
ADMIN	FINANCE	100
FINANCE	TRAINING	60
FINANCE	RESOURCE	40
TRAINING	RESOURCE	50

→ No. of Computers in various UNITS are:

UNIT	NO. OF COMPUTERS
ADMIN	150
FINANCE	25
TRAINING	90
RESOURCE	75

- i) Suggest an ideal cable layout for connecting the above UNITS.
- ii) Suggest the most suitable place i.e. UNIT to install the server for the above
- iii) Which network device is used to connect the computers in all UNITS?

- iv) Suggest the placement of Repeater in the UNITS of above network.
 - v) NGO is planning to connect its Regional Office at Kota, Rajasthan. Which out of the following wired communication, will you suggest for a very high-speed Connectivity ?
- (a) Twisted Pair cable (b) Ethernet cable (c) Optical Fiber

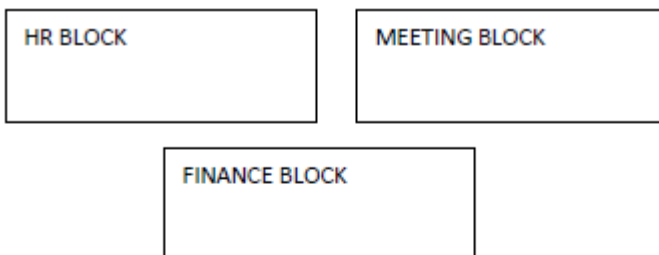
Ans:- i. Layout



- ii. Admin
- iii. SWITCH/HUB
- iv. ADMIN & FINANCE
- v. (c) Optical Fiber

14. India Tech Solutions (ITS) is a professional consultancy company. The company is planning to set up their new offices in India with its hub at Hyderabad. As a network adviser, you have to understand their requirement and suggest them the best available solutions. Their queries are mentioned as (i) to (v) below.

Physical locations of the blocks of TTC:-



Block to block distance (in m)

Block (From)	Block (To)	Distance
HR Block	MEETING	110
HR Block	Finance	40
MEETING	Finance	80

Expected number of computers

Block	Computers
HR	25
Finance	120
MEETING	90

- (i) Which will be the most appropriate block, where TTC should plan to install their server?
- (ii) Draw a block to block cable layout to connect all the buildings in the most appropriate manner for efficient communication.

(iii) What will be the best possible connectivity out of the following, you will suggest to connect the new set up of offices in Bangalore with its London based office.

- Satellite Link
- Infrared
- Ethernet

(iv) Which of the following device will be suggested by you to connect each computer in each of the buildings?

- Switch
- Modem
- Gateway

(v) Company is planning to connect its offices in Hyderabad which is less than 1 km. Which type of network will be formed?

Ans:

(i) TTC should install its server in finance block as it is having maximum number of computers.

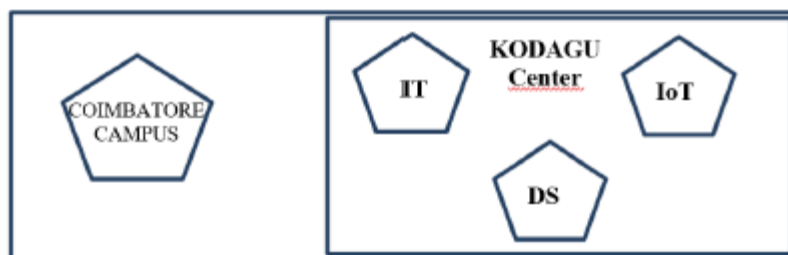
(ii) Any suitable layout

(iii) Satellite Link.

(iv) Switch.

(v) LAN

15. Total-IT Corporation, a Karnataka based IT training company, is planning to set up training centers in various cities in next 2 years. Their first campus is coming up in Kodagu district. At Kodagu campus, they are planning to have 3 different blocks, one for AI, IoT and DS (Data Sciences) each. Each block has number of computers, which are required to be connected in a network for communication, data and resource sharing. As a network consultant of this company, you have to suggest the best network related solutions for them for issues/problems raised in question nos. (i) to (v), keeping in mind the distances between various blocks/locations and other



Distance between various blocks/locations:

Block	Distance
IT to DS	28 m
IT to IoT	55 m
DS to IoT	32 m
Kodagu Campus to Coimbatore Campus	304 km

Number of computers:

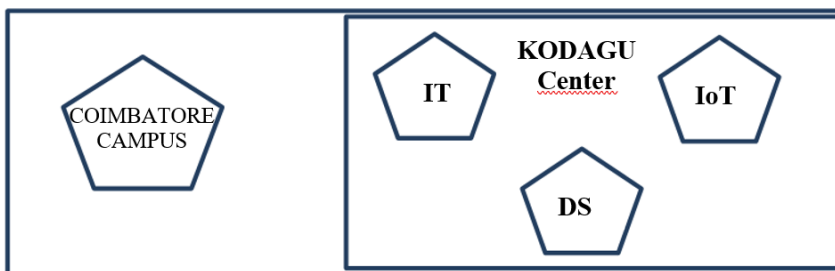
Block	Number of Computers
IT	75
DS	50
IoT	80

given parameters.

- (i) Suggest the most appropriate block/location to house the SERVER in the Kodagu campus (out of the 3 blocks) to get the best and effective connectivity. Justify your answer.
- (ii) Suggest a device/software to be installed in the Kodagu Campus to take care of data security.
- (iii) Suggest the best wired medium and draw the cable layout (Block to Block) to most efficiently connect various blocks within the Kodagu Campus.
- (iv) Suggest the placement of the following devices with appropriate reasons:
 - a) Switch/Hub b) Router
- (v) Suggest a protocol that shall be needed to provide Video Conferencing solution between Kodagu Campus and Coimbatore Campus.

Ans:-

- i) **IoT block, as it has the maximum number of computers**
- ii) **Firewall**
- iii) **Optical fiber**



- iv) a) **Switch/Hub: In each block to interconnect the computers in that block.**
 b) **Router: In IoT block (with the server) to interconnect all the three blocks.**
- v) **VOIP(Voice over internet protocol)**

UNIT 3: DATABASE MANAGEMENT

DATABASE: Database is a recordkeeping system which holds interrelated data that can be easily accessed, managed and updated.

RDBMS: A relational database management system (RDBMS) is a SOFTWARE that allows you to create, update, and administer a relational database (Data is organized in the form of table which contains rows and columns)

Need for Database:

- Minimize Data Redundancy: Database system minimizes redundancy(duplicate data) by data normalization.
- Reduce Data Inconsistency: When multiple copies of same data do not match with one another it is known as data inconsistency.
- Data Security: Database allows only the authorized user to access data in the database
- Data Sharing: Database allow its users to share data among themselves.
- Data Integrity: Data in the database are accurate and consistent.

Data Model: A data model is the way data is organised in a database.

There are different types data models controls the representation of data in a database :=

- Relational Data model
- NetworkData Model
- Hierarchical Data Model
- Object Oriented Data Model
- The Relational data Model

The Relational data Model is far being used by most of the popular Database Management Systems. In his course we limit our discussion on Relational data model.

IMPORTANT TERMINOLOGIES : (Parts of MySQL Table)

let's consider data regarding Doctors

DOCTOR ← Table name known as Relation

	Name	Age	Department	Charges	Gender
0	Arprit	62	Surgery	300	M
1	Zarina	22	ENT	250	F
2	Kareem	32	Orthopaedic	200	M
3	Arun	12	Surgery	300	M
4	Zubin	30	ENT	250	M
5	Kettaki	16	ENT	250	F
6	Ankita	29	Cardiology	800	F
7	Zareen	45	Cardiology	300	F
8	Kush	19	Cardiology	800	M
9	Shilpa	23	Nuclear	400	F

Name, Age, Department etc are Column Name Or Atributes

Data Values

Known as Row/Tuple/Record

<https://www.Pythonclassroomdiary.wordpress.com>

Relational Database Terminologies

- ✓ **Tuple or Record:** In Relational Database, rows in the Relation(Table) are referred as records
- ✓ **Attributes or Fields:** Columns are referred as Attributes or Fields.
- ✓ **Cardinality:** Total number of records in the relation is called as its Cardinality.
- ✓ **Degree:** Total number of Attributes/Fields in the relation is called as its Degree.
- ✓ **Domain :** A domain is a set of values that can be stored in a column of a database table.

KEYS in Database :

- **Primary Key :** A column or group of columns in a table which helps us to identify unique tuple (Record) in a table.
- **Candidate Keys :** All the keys which may be selected as Primary Key are Candidate keys
- **Alternate Keys :** All the Candidate keys which are not selected as primary key are called an alternate key
- **Foreign Key :** A foreign key is a column which is added to create a relationship with another table. Foreign keys help us to maintain data integrity and also allow navigation between two different instances of an entity.

SQL (Structured Query Language). SQL is a programming language designed to manage data stored in relational database and is used to query, insert, update and modify data. SQL commands are grouped into four major categories

- i. **Data Definition Language (DDL)** - are used for creating, modifying, and dropping the physical structure of database objects.
Examples: CREATE, ALTER, DROP, RENAME
- ii. **Data Manipulation Language (DML)** - are used for storing, retrieving, modifying, and deleting data.
Examples: SELECT , INSERT, UPDATE, DELETE
- iii. **Transaction Control Language (TCL)** - These commands are used to managing changes done through DML commands
Examples: ROLLBACK, COMMIT, SAVEPOINT
- iv. **Data Control Language (DCL)** - These SQL commands are used for providing security to database objects
Examples: GRANT, REVOKE

Data types in SQL

Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n)	Character string of Variable length. Maximum length n
INTEGER	Integer numerical (no decimal). Precision 10
DECIMAL (p,s) NUMERIC(p,s)	Precision p, Scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
FLOAT	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values

Operators in SQL:

The following are the commonly used operators in SQL

1. Arithmetic Operators +, -, *, /
2. Relational Operators =, <, >, <=, >=, <>
3. Logical Operators OR, AND, NOT

- **Arithmetic operators** are used to perform simple arithmetic operations.
- **Relational Operators** are used when two values are to be compared and
- **Logical operators** are used to connect search conditions in the WHERE Clause in SQL

Constraints:

Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play when ever we try to insert, delete or update a record in a relation.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

Not null ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

Unique constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

Primary key constraint means that a column can not have duplicate values and not even a null value.

e.g. Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

Foreign key is used to enforce referential integrity and is declared as a primary key in some other table.

e.g `cust_id varchar(5) references master(cust_id);`

it declares `cust_id` column as a foreign key that refers to `cust_id` field of table `master`. That means we cannot insert that value in `cust_id` field whose corresponding value is not present in `cust_id` field of `master` table.

Check constraint limits the values that can be inserted into a column of a table.

e.g `marks number(3) check(marks>=0);`

The above statement declares `marks` to be of type number and while inserting or updating the value in `marks` it is ensured that its value is always greater than or equal to zero.

Default constraint is used to specify a default value to a column of a table automatically. This default value will be used when user does not enter any value for that column.

e.g `balance number(5) default = 0;`

SQL COMMANDS

1. CREATE TABLE is used to create table structure

```
CREATE TABLE student (  
Roll_no    number(3) primary key,  
Name       varchar(25) not null,  
Class      varchar(10),  
Marks      number(3) check(marks>0),  
City       varchar(25) );  
Data Modifications in SQL
```

After a table has been created using the create table command, tuples can be inserted into the table, or tuples can be deleted or modified.


2. INSERT Statement

The simplest way to insert a tuple into a table is to use the insert statement

```
insert into <table> [(<column i, . . . , column j>)] values (<value i, . . . , value j>);
```

```
INSERT INTO student VALUES(101,'Rohan','XI',400,'Jammu');
```

While inserting the record it should be checked that the values passed are of same data types as the one which is specified for that particular column.

 For inserting a row interactively (from keyboard) & operator can be used.

e.g INSERT INTO student VALUES(&Roll_no','&Name','&Class','&Marks','&City');

In the above command the values for all the columns are read from keyboard and inserted into the table student.

NOTE:- In SQL we can repeat or re-execute the last command typed at SQL prompt by typing “/” key and pressing enter.

Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Queries:

To retrieve information from a database we can query the databases. SQL SELECT statement is used to select rows and columns from a database/relation.

3. SELECT Command

This command can perform selection as well as projection.

Selection: This capability of SQL can return you the tuples form a relation with all the attributes.

Projection: This is the capability of SQL to return only specific attributes in the relation.


* **SELECT * FROM student;** command will display all the tuples in the relation student

* **SELECT * FROM student WHERE Roll_no <=102;** command display only those records whose Roll_no less than or equal to 102.

 **SELECT command can also display specific attributes from a relation.**

* **SELECT name, class FROM student;** The above command displays only name and class attributes from student table.

* **SELECT count(*) AS “Total Number of Records” FROM student;** Display the total number of records with title as “Total Number of Records” i.e an alias

 We can also use arithmetic operators in select statement, like

* **SELECT Roll_no, name, marks+20 FROM student;**

* **SELECT name, (marks/500)*100 FROM student WHERE Roll_no > 103;**

 **Eliminating Duplicate/Redundant data**

DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement.

 **Conditions based on a range**

SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.

e.g. `SELECT Roll_no, name FROM student WHERE Roll_no BETWEEN 100 AND 103;`

The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

Conditions based on a list

To specify a list of values, IN operator is used. This operator select values that match any value in the given list.

e.g. `SELECT * FROM student WHERE city IN ('Jammu','Amritsar','Gurdaspur');`

The above command displays all those records whose city is either Jammu or Amritsar or Gurdaspur.

Conditions based on Pattern

SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

- a. percent(%) Matches any string
- b. Underscore(_) Matches any one character

e.g. `SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";`

displays those records where last digit of Roll_no is 3 and may have any number of characters in front.

e.g. `SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";`

displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

4. DELETE Command

To delete the record from a table SQL provides a delete statement. General syntax is:-

`DELETE FROM <table_name> [WHERE <condition>];`

e.g. `DELETE FROM student WHERE city = 'Bhopal';`

This command deletes all those records whose city is Bhopal.

5. UPDATE Command

To update the data stored in the data base, UOPDATE command is used.

e. g. `UPDATE student SET marks = marks + 100; (Increase marks of all the students by 100.)`

e. g. `UPDATE student SET City = 'Indore ' WHERE city = 'Ujjain';`
changes the city of those students to Indore whose city is Ujjain.

We can also update multiple columns with update command, like

e. g. `UPDATE student set marks = marks + 20, city = 'Jalandhar'`

WHERE city NOT IN ('Jammu','Udhampur');

7. ALTER TABLE Command

In SQL if we ever need to change the structure of the database then ALTER TABLE command is used. By using this command we can add a column in the existing table, delete a column from a table or modify columns in a table.

Adding a column

The syntax to add a column is:-

```
ALTER TABLE table_name  
ADD column_name datatype;
```

e.g ALTER TABLE student ADD(Address varchar(30));

The above command add a column Address to the table atudent.

If we give command

```
SELECT * FROM student;
```

The following data gets displayed on screen:

Roll_no	Name	Class	Marks	City	Address
101	Rohan	XI	400	Jammu	
102	Aneeta Chopra	XII	390	Udhampur	
103	Pawan Kumar	IX	298	Amritsar	
104	Rohan	IX	376	Jammu	
105	Sanjay	VII	240	Gurdaspur	
113	Anju MAhajan	VIII	432	Pathankot	

Note that we have just added a column and there will be no data under this attribute. UPDATE command can be used to supply values / data to this column.

Removing a column

```
ALTER TABLE table_name  
DROP COLUMN column_name;
```

e.g ALTER TABLE Student
DROP COLUMN Address;

The column Address will be removed from the table student.

8. DROP TABLE Command

Sometimes you may need to drop a table which is not in use. DROP TABLE command is used to Delete / drop a table permanently. It should be kept in mind that we can not drop a table if it contains

records. That is first all the rows of the table have to be deleted and only then the table can be dropped. The general syntax of this command is:-

```
DROP TABLE <table_name>;
```

e.g DROP TABLE student;
This command will remove the table student

EXCERCISES

Queries using Create database, Show databases, Use, Create table, Show Tables, Describe, Rename, Alter, Select,

From, Where, Insert, Update commands

i. Creating database the query will be-

Create database school;

ii. For showing list of database-

Show databases;

iii. For accessing database-

Use school;

iv. For creating table student with rollno , name, phone,age,city and percentage columns. Age must be greater than or equals to 16. Use all possible constraints-

```
CREATE TABLE STUDENT (ROLLNO INT(3) PRIMARY KEY, NAME VARCHAR (15) NOT NULL,  
PHONE CHAR(10)UNIQUE, AGE INT (2) DEFAULT 16, CITY VARCHAR (15),PERCENTAGE  
FLOAT(5,2),CHECK (AGE>=16));
```

v. For showing list of tables-

Show Tables;

vi. For showing structure of table-

Describe STUDENT;

OR

DESC STUDENT;

vii. For Renaming the table STUDENT TO XII_CS_STUDENTS-

```
ALTER TABLE STUDENT RENAME TO XII_CS_STUDENTS;
```

viii. Alter Command

- ALTER TABLE - ADD Column
Query to add email id in table student-

```
ALTER TABLE Student  
ADD Email varchar (255);
```

- ALTER TABLE - DROP COLUMN

Query to remove email id from table student-

- ALTER TABLE – CHANGE COLUMN DATATYPE OR SIZE-

Query to change datatype of column ROLLNO from INT to CHAR and size from 3 to 5 in table student-

```
ALTER TABLE Student MODIFY ROLLNO CHAR (5);
```

- ALTER TABLE – DROP a PRIMARY KEY Constraint

Query to remove PRIMARY KEY Constraint from table student-

```
ALTER TABLE Student
DROP PRIMARY KEY (ROLLNO);
```

- ALTER TABLE – ADD a PRIMARY KEY Constraint

Query to add PRIMARY KEY Constraint to column ROLLNO in table student-

```
ALTER TABLE Student
ADD PRIMARY KEY (ROLLNO);
```

- ALTER TABLE - RENAME Column

Query to RENAME column ROLLNO TO RNO in table student-

```
ALTER TABLE Student
CHANGE ROLLNO RNO INT(3);
```

ix. Select, From and Where

- Showing all columns and rows from table student-SELECT * FROM STUDENT;
- Showing only RNO and NAME columns and all rows from table student-SELECT RNO,NAME FROM STUDENT;
- Showing all columns and rows having PERCENTAGE greater than 60 from table student-SELECT * FROM STUDENT WHERE PERCENTAGE>60;
- Showing all columns and rows having PERCENTAGE equal to 60 from table student-SELECT * FROM STUDENT WHERE PERCENTAGE=60;
- Showing all columns and rows having PERCENTAGE equal to 60 or 80 from table student-SELECT * FROM STUDENT WHERE PERCENTAGE=60 OR PERCENTAGE=80;
- Showing all columns and rows having PERCENTAGE between 60 and 80 from table student(60 and 80 must be involved in output) -
SELECT * FROM STUDENT WHERE PERCENTAGE BETWEEN 60 AND 80;OR
SELECT * FROM STUDENT WHERE PERCENTAGE >= 60 AND PERCENTAGE <=80;
- Showing all columns and rows NOT having THE PERCENTAGE 60 from table student-SELECT * FROM STUDENT WHERE PERCENTAGE !=60;
- Showing ONLY UNIQUE NAMES with heading (column alias) 'UNIQUE NAMES' from table student-SELECT DISTINCT NAME AS 'UNIQUE NAMES' FROM STUDENT;

x. The SQL INSERT INTO Statement

The **INSERT INTO** statement is used to insert new records in a table.

- Query to insert values in all the columns of table student-
INSERT INTO STUDENT VALUES(101,'RAMAN','1234567892',18,'JODHPUR',99.98);
- Query to insert values only in RNO and NAME columns of table student-INSERT INTO STUDENT (RNO,NAME')
VALUES(102,'JON');

xi. The SQL UPDATE Statement

The **UPDATE** statement is used to modify the existing records in a table.

- Query to change percentage of all students to 100-
UPDATE STUDENT SET
PERCENTAGE=100;
- Query to change percentage of Raman to 90-
UPDATE STUDENT SET PERCENTAGE=90 WHERE NAME='RAMAN';

xii. The SQL DELETE Statement

The **DELETE** statement is used to delete existing records in a table.

- Query to delete students with rno 101- **DELETE FROM STUDENT WHERE RNO=101**;Query to delete ALL the students
DELETE FROM STUDENT;

2. Queries using DISTINCT, BETWEEN, IN, LIKE, IS NULL, ORDER BY, GROUP BY, HAVING

* **The SQL SELECT DISTINCT Statement**-The **SELECT DISTINCT** statement is used to return only distinct (different) values. Inside a table, a column often contains many duplicate values; and sometimes we only want to list the different (distinct) values.

- Showing ONLY UNIQUE NAMES from table student-SELECT DISTINCT NAME FROM STUDENT;

* **The SQL BETWEEN Operator**- the **BETWEEN** operator selects values within a given range. The values can be numbers, text, or dates. The **BETWEEN** operators is

inclusive: begin and end values are included.

- Showing all columns and rows having PERCENTAGE between 60 and 80 from table student(60 and 80 must be involved in output) -
SELECT * FROM STUDENT WHERE PERCENTAGE BETWEEN 60 AND 80;

* **The SQL IN Operator**-the **IN** operator allows us to specify multiple values in a **WHERE** clause. The **IN** operator is a shorthand for multiple **OR** conditions.

- Showing all columns and rows having PERCENTAGE equal to 60,70,80 or 90 from table student-
SELECT * FROM STUDENT WHERE PERCENTAGE IN(60,70,80,90);

The SQL LIKE Operator -The **LIKE** operator is used in a **WHERE** clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the **LIKE** operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character
- Showing only NAME columns and rows where values of name that start with "a"
SELECT NAME FROM STUDENT WHERE NAME LIKE 'A%';
- Showing only NAME columns and rows where values of name that ends with "a"
SELECT NAME FROM STUDENT WHERE NAME LIKE '%A';
- Showing only NAME columns and rows where values of name that contains "a" in any of the position from table student-
SELECT NAME FROM STUDENT WHERE NAME LIKE '%A%';
- Showing only NAME columns and rows where values of name contains "a" at second position from table student-
SELECT NAME FROM STUDENT WHERE NAME LIKE '_A%';
- Showing only NAME columns and rows where values of name that start with "a" and are at least 2 characters in length from table student-
SELECT NAME FROM STUDENT WHERE NAME LIKE 'A_';
- Showing only NAME columns and rows where values of name that contains only 5 characters in length from table student-
SELECT NAME FROM STUDENT WHERE NAME LIKE '_____''(5 times underscore sign is used)';
- Showing only NAME columns and rows where values of name that start with "a" and end with "o" from table student-
SELECT NAME FROM STUDENT WHERE NAME LIKE 'A%O';

% and _ are known as wildcard characters in SQL

HANDELLING NULL VALUES

- Showing all details from table student where phone number is not given-`SELECT * FROM STUDENT WHERE PHONE IS NULL;`
- Showing only those rows from table student where phone number is given -`SELECT * FROM STUDENT WHERE PHONE IS NOT NULL;`

The SQL ORDER BY Keyword

The **ORDER BY** keyword is used to sort the result-set in ascending or descending order.

The **ORDER BY** keyword sorts the records in ascending order by default. To sort the records indescending order, use the **DESC** keyword.

- Showing NAMES IN ASCENDING ORDER from table student-`SELECT NAME FROM STUDENT ORDER BY NAME;`
OR
`SELECT NAME FROM STUDENT ORDER BY NAME ASC;`
- Showing NAMES IN DESCENDING ORDER from table student-`SELECT NAME FROM STUDENT ORDER BY NAME DESC;`

The SQL GROUP BY Statement

The **GROUP BY** statement groups rows that have the same values into summary rows, like "findthe number of customers in each country".

The **GROUP BY** statement is often used with aggregate functions (**COUNT()**, **MAX()**, **MIN()**, **SUM()**, **AVG()**) to group the result-set by one or more columns.

- SQL statement to lists the number of students in each city:-

```
SELECT CITY,COUNT(CITY)FROM STUDENT
GROUP BY CITY;
```

The SQL HAVING Clause

The **HAVING** clause was added to SQL because the **WHERE** keyword cannot be used with aggregatefunctions.

- SQL statement to lists the number of students in JODHPUR city:-
`SELECT CITY,COUNT(CITY)FROM STUDENT`

```
GROUP BY CITY HAVING CITY='JODHPUR';
```

3. Queries for Aggregate functions- SUM(), AVG(), MIN(), MAX(), COUNT() –All are known as aggregate functions and used with multiple rows but return single answer-
The **SUM()** function returns the total sum of a numeric column.

- SQL statement to print total percentage of students in each city :-

```
SELECT CITY, SUM(PERCENTAGE) FROM STUDENT  
GROUP BY CITY;
```

The **AVG()** function returns the average value of a numeric column.

- SQL statement to print average percentage of students in each city :-

```
SELECT CITY, AVG(PERCENTAGE) FROM STUDENT  
GROUP BY CITY;
```

The **MIN()** function returns the smallest value of the selected column.

- SQL statement to print minimum percentage of students :-

```
SELECT CITY, MIN(PERCENTAGE) FROM STUDENT;
```

The **MAX()** function returns the largest value of the selected column.

- SQL statement to print maximum percentage of students :-

```
SELECT CITY, MAX(PERCENTAGE) FROM STUDENT;
```

The **COUNT()** function returns the number of rows that matches a specified criterion.

- SQL statement to lists the number of students in each city:-

```
SELECT CITY, COUNT(CITY) FROM STUDENT  
GROUP BY CITY;
```

LETS PRACTICE 1

Question 1: Match the SQL Commands with their Descriptions

SQL Command	Description
A. SELECT	1. Removes one or more records from a table.
B. INSERT	2. Retrieves data from one or more tables.
C. DELETE	3. Adds new records to a table.
D. UPDATE	4. Modifies existing records in a table.

Question 2: Match the Database Concepts with their Definitions

Concept	Definition
A. Primary Key	1. Ensures all database transactions are processed reliably.
B. Foreign Key	2. A field in a table that uniquely identifies each row.
C. Normalization	3. A field in a table that links to a primary key in another table.

Question 3:- Fill in the Blanks

1. The _____ clause is used in SQL to filter records.
2. A _____ is a set of rules that defines how data is organized, accessed, and manipulated in a database.
3. The _____ clause in SQL is used to group rows that have the same values in specified columns into summary rows.
4. The process of ensuring data accuracy and consistency within a database is known as data _____.
5. The _____ operation in SQL combines rows from two or more tables based on a related column between them.

Question 4 True /False

1. In SQL, the 'DELETE' statement is used to remove a table from the database.

False

2. A foreign key in a database table must always reference a primary key in another table.

True

3. In a relational database, data is organized into tables, which consist of rows and columns.

True

4. The 'JOIN' clause in SQL is used to combine rows from two or more tables based on a related column.

True

5. The 'ALTER TABLE' statement in SQL is used to modify the data within a table.

False

Question 5 Choose the correct Option

1. Consider the following SQL statement. What type of statement is this?

```
CREATE TABLE employee (name VARCHAR, id INTEGER)
```

- (a) DML (b) DDL
(c) DCL (d) Integrity constraint

2. In the given query which keyword has to be inserted?

```
INSERT INTO student _____ (1002, "Kumar", 2000);
```

- (a) Table (b) Values
(c) Relation (d) Field

3. Which of the following group functions ignore NULL values?

- (a) MAX (b) COUNT
(c) SUM (d) All of the above

4. Where and Having clauses can be used interchangeably in SELECT queries?

- (a) True (b) False
(c) Only in views (d) With order by

5. Which of the following is true about the SQL AS clause?

- A. The AS clause in SQL is used to change the column name in the output or assign a name to a derived column.**
- B. The SQL AS clause can only be used with the JOIN clause.**
- C. The AS clause in SQL is used to defines a search condition.**
- D. All of the mentioned.**

Question 6 : Assertion & Reasoning

1. Assertion (A) : The 'GROUP BY' clause in SQL is used to aggregate data based on specified columns.

Reasoning (R) : The 'GROUP BY' clause is used to group rows that have the same values in specified columns, often used in conjunction with aggregate functions like SUM, COUNT, AVG, etc., to perform operations on grouped data.

Answer R is the correct explanation of A

2. Assertion(A): The primary key uniquely identifies each record in a table.

Reasoning(A): The primary key constraint ensures that each row in a table is uniquely identified, preventing duplicate records and ensuring data integrity.

Answer R is the correct explanation of A

3. Assertion(A) : Natural join can lead to ambiguous column names in the result.

Reason (R) : Natural join automatically joins tables using columns with the same names.

Answer A is True but R is False

4. Assertion (A): ACID properties in database transactions ensure reliability.

Reason (R): ACID stands for Atomicity, Consistency, Isolation, and Durability.

Answer R is the correct explanation of A

Q7. How would you create a table in SQL to store information about students, including their ID, name, and date of birth? Write the SQL statement.

Answer

```
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    Name VARCHAR(100),  
    DateOfBirth DATE  
);
```

Q8. How would you retrieve the names and dates of birth of all students from a Students table who were born after January 1, 2000? Write the SQL query.

Answer

```
SELECT Name, DateOfBirth  
FROM Students  
WHERE DateOfBirth > '2000-01-01';
```

Q9. Question: Write an SQL query to retrieve the names of students and the names of the courses they are enrolled in. Assume there are two tables: Students (StudentID, Name) and Enrollments (StudentID, CourseID), and Courses (CourseID, CourseName).

Answer

```
SELECT Students.Name, Courses.CourseName  
FROM Students  
JOIN Enrollments ON Students.StudentID = Enrollments.StudentID  
JOIN Courses ON Enrollments.CourseID = Courses.CourseID;
```

Q10. Write an SQL query to update the date of birth of a student with StudentID 101 to '2001-05-15'.

Answer

```
UPDATE Students  
SET DateOfBirth = '2001-05-15'
```


WHERE StudentID = 101;

LETS PRACTICE 2

DATABASE MANAGEMENT SYSTEM CBT (10 QcUESTIONS)

- 1 What is the primary purpose of database management?
 - a) To design user interfaces for applications
 - b) To efficiently store, organize, and manage data
 - c) To develop computer networks
 - d) To optimize website performanceAnswer: b) To efficiently store, organize, and manage data
- 2 Which aspect of database management ensures that data is accurate, consistent, and reliable?
 - a) Data Modeling
 - b) Data Security
 - c) Data Integrity
 - d) Data RetrievalAnswer: c) Data Integrity
- 3 Which type of database management system stores data in tables with rows and columns?
 - a) Hierarchical Database
 - b) Relational Database
 - c) NoSQL Database
 - d) Object-Oriented DatabaseAnswer: b) Relational Database
- 4 Which statement is true about primary keys in a database?
 - a) Primary keys are used for data encryption
 - b) Primary keys are used for data indexing
 - c) Primary keys uniquely identify each record in a table
 - d) Primary keys are used for data backupAnswer: c) Primary keys uniquely identify each record in a table
- 5 In the context of database management, what is the purpose of a foreign key?
 - a) To store binary data such as images or videos
 - b) To establish a link between two or more database tables
 - c) To prevent unauthorized access to the database
 - d) To improve data retrieval speedAnswer: b) To establish a link between two or more database tables
- 6 A Database Management System (DBMS) is
 - a) Collection of interrelated data
 - b) Collection of programs to access data
 - c) Collection of data describing one particular enterprise
 - d) All of the aboveAns: - a) Collection of interrelated data
- 7 In mathematical term Table is referred as
 - a) Relation

- b) Attribute
- c) Tuple
- d) Domain

Ans: - a) Relation

8 Which of the following is true regarding Referential Integrity?

- a) Every primary-key value must match a primary-key value in an associated table
- b) Every primary-key value must match a foreign-key value in an associated table
- c) Every foreign-key value must match a primary-key value in an associated table
- d) Every foreign-key value must match a foreign-key value in an associated table

Ans: - c) Every foreign-key value must match a primary-key value in an associated table

9 Which of the following places the common data elements in order from smallest to largest

- a) character, file, record, field, database
- b) character, record, field, database, file
- c) character, field, record, file, database
- d) Bit, Byte, character, record, field, file, database

Ans: - c) character, field, record, file, database

10 When data changes in multiple lists and all lists are not updated, this causes

- a) data redundancy
- b) information overload
- c) duplicate data
- d) data inconsistency

Ans: - d) data inconsistency

SQL JOIN

A join is a query that combines rows from two or more tables. In a join-query, more than one table are listed in FROM clause. The function of combining data from multiple tables is called joining.

SQL can produce data from several related tables by performing either a physical or virtual join of the tables.

The WHERE clause is most often used to perform the JOIN function where two or more tables have common columns.

Consider the following example:-

```
SELECT patient_no,description,normal_charge,charge FROM billed,item
WHERE billed.item_code=item.item_code;
```

1. Cartesian Product:-

Consider the following query:-

```
SELECT *
FROM EMP,DEPT;
```

This query will give the Cartesian product ie all possible concatenations are formed of all rows of both the tables EMP and DEPT. That is, when no particular rows(using WHERE clause) and columns are selected. Such an operation is also known as unrestricted join. It returns $n_1 \times n_2$ rows where n_1 is no of rows in first table and n_2 is no of rows in second table.

2. Equi-join and Natural Join:-

The join, in which columns are compared for equality, is called Equi-join.

A non-equi-join is a query that specifies some relationship other than equality between the columns.

The join in which only one of the identical columns (coming from joined tables) exists is called Natural join.

The Equi-join and Natural join are equivalent except that duplicate columns are eliminated in the Natural join that would otherwise appear in the Equi-Join.

3. What is Natural Join?

Ans:- A NATURAL JOIN is a join operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables.

4. What is Equi-Join?

Ans:-An Equi-join is a sql join where we use the equal sign as the comparison operator ie '=' between two tables while specifying the join condition e.g

```
SELECT * FROM EMP E,DEPT D WHERE E.DEPTNO=D.DEPTNO.
```

EXCERCISES

1. The operation whose result contains all pairs of tuples from the two relations, regardless of whether their attribute values match.

- a. Join b. Cartesian Product c. Intersection d. Set difference

Ans:- [b]

2. The following SQL in which type of join :-

```
SELECT CUSTOMER.CUST_ID,ORDER.CUST_ID,ORDER_ID
FROM CUSTOMER,ORDER
WHERE CUSTOMER.CUST_ID=ORDER.ORDER_ID:
```

- a. Equi-Join b. Natural Join c. Outer join d. Cartesian product

Ans:- [a]

3. The following SQL in which type of join:

```
SELECT CUSTOMER.CUST_ID,ORDER.CUST_ID,ORDER_ID
FROM CUSTOMER,ORDER;
```

- a. Equi-join b. Natural-join c. Outer join d. Cartesian Product

Ans:- [d]

7. Which product is returned in a join query have no join condition?

- a. Equi-join b. Cartesian Product c. Both (a) and (b) d. None of the mentioned

Ans:-[b]

8. Which is a join condition an equality operator?

- a. Equi-join b. Cartesian Product c. Both (a) (b) d. None of the mentioned

Ans:- [a]

9. To get data from two or more tables having some common fields, ----- query is created. [join]

10. In qui-join, the join condition joins the two tables using ----- operator. [=]

CASE STUDY BASED QUESTIONS ON SQL JOINS

Case Study 1: Students and Courses

Question 1: You have two tables: Students and Courses. You need to list all students along with the courses they are enrolled in. Each student has a CourseID in the Students table that matches the CourseID in the Courses table.

**Answer :- SELECT Students.StudentName, Courses.CourseName
FROM Students, Courses WHERE Students.CourseID = Courses.CourseID;**

Case Study 2: Employees and Departments

Question 2: You have two tables: Employees and Departments. You need to list all employees along with their department names. The DepartmentID in the Employees table matches the DepartmentID in the Departments table.

Answer :- SELECT Employees.EmployeeName, Departments.DepartmentName

FROM Employees

JOIN Departments ON Employees.DepartmentID = Departments.DepartmentID;

Or

SELECT Employees.EmployeeName, Departments.DepartmentName

FROM Employees, Departments WHERE Employees.DepartmentID = Departments.DepartmentID;

Case Study 3: Movies and Directors

Question 3: You have two tables: Movies and Directors. You need to list all movies along with the names of their directors. The DirectorID in the Movies table matches the DirectorID in the Directors table.

Answer:- SELECT Movies.MovieTitle, Directors.DirectorName

FROM Movies, Directors WHERE Movies.DirectorID = Directors.DirectorID;

Case Study 4 : Consider three tables in a database: employees, departments, and projects. Each table contains information as follows:

The employees table includes columns for employee_id, employee_name, and department_id.

The departments table includes columns for department_id and department_name.

The projects table includes columns for project_id, project_name, and department_id.

- a) Write an SQL query to perform a Cartesian product between the employees and projects tables, displaying all possible combinations of employees and projects.
- b) Write an SQL query to retrieve the names of employees along with the names of their corresponding departments using an equijoin between the employees and departments tables.
- c) Write an SQL query to retrieve the names of projects along with the names of their corresponding departments using a natural join between the projects and departments tables.

Answer a) `SELECT employees.employee_id, employees.employee_name, projects.project_id, projects.project_name FROM employees CROSS JOIN projects;`

b) `SELECT employees.employee_name, departments.department_name FROM employees ,departments WHERE employees.department_id = departments.department_id;`

c) `SELECT projects.project_name, departments.department_name FROM projects NATURAL JOIN departments;`

Case Study 5: Consider the Following tables and write sql query .

Table Customers

CustomerID	CustomerName	ContactName	Country
1	Alfreds	Maria	Germany
2	Ana	Anil	Mexico
3	Antonio	Antonio Moreno	Mexico
4	Rajsons	Thomas	UK
5	Benzeer	Christina	Sweden

Table : Orders

OrderID	CustomerID	OrderDate
10308	2	2024-03-01
10309	37	2024-03-03
10310	77	2024-03-04
10311	3	2024-03-05
10312	5	2024-03-06

- a) Write an SQL query to find the orders along with the customer names for only those orders that have matching customer records.
- b) Write an SQL query to find all customers and their orders, including customers who do not have any orders.
- c) Write an SQL query to find all orders and their customer details, including orders that do not have a matching customer.
- d) Write an SQL query to find all customers and all orders, matching them when possible.

Answer a) `SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate`

FROM Orders ,Customers WHERE Orders.CustomerID = Customers.CustomerID;

b) SELECT Customers.CustomerName, Orders.OrderID, Orders.OrderDate

FROM Customers LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

c) SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

d) SELECT Customers.CustomerName, Orders.OrderID, Orders.OrderDate

FROM Customers FULL OUTER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;

Case Study 6:- Let's say we have a database for a small online bookstore having following tables

books: (book_id, title, author_id, genre_id, price).

authors: (author_id, author_name, birth_year)

genres: (genre_id , genre_name).

customers: (customer_id, customer_name, email)

orders: (order_id, customer_id, order_date, total_amount)

order_details: (order_id, book_id, quantity, unit_price).

a) List all books along with their corresponding authors.

b) Display the total number of orders placed by each customer.

c) List all authors who have written books in the 'Science Fiction' genre.

d) Find the total revenue generated from each genre

Answer

a) SELECT books.title, authors.author_name, genres.genre_name

FROM books ,authors WHERE books.author_id = authors.author_id ;

b) SELECT customers.customer_name, COUNT(orders.order_id) AS total_orders

FROM customers

LEFT JOIN orders ON customers.customer_id = orders.customer_id

GROUP BY customers.customer_name;

c) SELECT DISTINCT authors.author_name

FROM authors

JOIN books ON authors.author_id = books.author_id

JOIN genres ON books.genre_id = genres.genre_id

WHERE genres.genre_name = 'Science Fiction';

d) SELECT genres.genre_name, SUM(order_details.quantity * order_details.unit_price) AS total_revenue FROM genres

JOIN books ON genres.genre_id = books.genre_id

JOIN order_details ON books.book_id = order_details.book_id

GROUP BY genres.genre_name;

Case Study 7: Bank Database

Consider a bank named "SafeBank" that manages accounts, customers, and transactions. The bank maintains data about accounts, customers, transactions, and branches in its database.

Here are the tables in the SecureBank database:

1. **accounts:** Contains information about bank accounts, including account_number, customer_id, balance, and branch_id.
2. **customers:** Stores details about bank customers, such as customer_id, customer_name, email, and phone_number.
3. **transactions:** Contains information about transactions, including transaction_id, account_number, transaction_type, amount, and transaction_date.
4. **branches:** Stores details about bank branches, including branch_id, branch_name, location, and manager.

a) List all accounts along with the names of their account holders and their current balances.

b) Find the total balance of all accounts in each branch.

c) Display the names of branches along with the names of their managers.

d) Find the average balance of accounts for each customer.

Answer

a) SELECT a.account_number, c.customer_name, a.balance

FROM accounts a , customers c WHERE a.customer_id = c.customer_id;

b) SELECT b.branch_id, b.branch_name, SUM(a.balance) AS total_balance

FROM branches b, accounts a WHERE b.branch_id = a.branch_id

GROUP BY b.branch_id, b.branch_name;

c) SELECT b.branch_name, m.manager_name

FROM branches b ,managers m WHERE b.manager_id = m.manager_id;

d) SELECT c.customer_id, c.customer_name, AVG(a.balance) AS avg_balance

FROM customers c, accounts a WHERE c.customer_id = a.customer_id

GROUP BY c.customer_id, c.customer_name;

Case Study 8 : Student and Teacher

Let's consider two tables, "students" and "teachers", and generate SQL join queries based on them.

Table Structures:

students: Contains information about students. Columns(student_id, student_name, class_id, age)

teachers: Stores details about teachers. Columns: (teacher_id, teacher_name, subject_taught)

- a) List all students along with their corresponding class names.
- b) Display the names of teachers along with the subjects they teach.
- c) Show the names of students who are enrolled in the 'Mathematics' class.
- d) List all subjects along with the names of classes they are taught in.
- e) Find the total number of students in each class.

Answer

a) SELECT s.student_id, s.student_name, c.class_name

FROM students s ,classes c WHERE s.class_id = c.class_id;

b) SELECT t.teacher_id, t.teacher_name, t.subject_taught FROM teachers t;

c) SELECT s.student_name FROM students s, classes c WHERE s.class_id = c.class_id

and c.class_name = 'Mathematics';

d) SELECT c.class_name, t.subject_taught

FROM classes c ,teachers t WHERE c.class_id = t.class_id;

e) SELECT c.class_name, COUNT(s.student_id) AS total_students

FROM classes c , students s WHERE c.class_id = s.class_id

GROUP BY c.class_name;

Case Study 9 Let's consider two tables, "doctors" and "patients", and generate SQL join queries based on them.

Table Structures:

doctors: Contains information about doctors. Columns(doctor_id, doctor_name, specialization, hospital_id)

patients: Stores details about patients. Columns: (patient_id, patient_name, doctor_id, admission_date)

a) List all patients along with the names of their treating doctors.

b) Show the names of patients who are treated by doctors specializing in 'Cardiology'.

c) List all doctors along with the number of patients they are treating.

d) Find the total number of patients treated in each hospital.

e) Display the names of patients along with their admission dates and the names of their treating doctors.

Answer

a) SELECT p.patient_id, p.patient_name, d.doctor_name

FROM patients p, doctors d WHERE p.doctor_id = d.doctor_id;

b) SELECT p.patient_name

FROM patients p, doctors d WHERE p.doctor_id = d.doctor_id and d.specialization = 'Cardiology';

c) SELECT d.doctor_name, COUNT(p.patient_id) AS num_patients

FROM doctors d LEFT JOIN patients p ON d.doctor_id = p.doctor_id

GROUP BY d.doctor_name;

```
d) SELECT d.hospital_id, COUNT(p.patient_id) AS num_patients
FROM doctors d ,patients p WHERE d.doctor_id = p.doctor_id
GROUP BY d.hospital_id;
```

```
e) SELECT p.patient_name, p.admission_date, d.doctor_name
FROM patients p, doctors d WHERE p.doctor_id = d.doctor_id;
```

Case Study 10 :- let's consider two tables, "railway" and "passenger", and generate SQL join queries based on them.

Table Structures:

railway: Contains information about railway trips. Columns: (trip_id, source_station, destination_station, departure_time, arrival_time)

passenger: Stores details about passengers on railway trips. Columns(passenger_id, trip_id, passenger_name, ticket_number, seat_number)

- 1) List all passengers along with the details of their corresponding railway trips.
- 2) Display the source and destination stations for all railway trips along with the names of passengers on each trip.
- 3) Show the names of passengers who traveled between 'Station A' and 'Station B'.
- 4) List all railway trips along with the total number of passengers on each trip.
- 5) Find the total number of passengers who traveled from each source station.

Answer

```
1) SELECT p.passenger_id, p.passenger_name, r.source_station, r.destination_station,
r.departure_time, r.arrival_time FROM passenger p, railway r WHERE p.trip_id = r.trip_id;
```

```
2) SELECT r.trip_id, r.source_station, r.destination_station, p.passenger_name
FROM railway r , passenger p WHERE r.trip_id = p.trip_id;
```

```
3) SELECT p.passenger_name
FROM passenger p, railway r WHERE p.trip_id = r.trip_id
```

```
And r.source_station = 'Station A' AND r.destination_station = 'Station B';
```

```
4) SELECT r.trip_id, COUNT(p.passenger_id) AS num_passengers
```

```
FROM railway r, passenger p WHERE r.trip_id = p.trip_id
```

```
GROUP BY r.trip_id;
```

```
5) SELECT r.source_station, COUNT(p.passenger_id) AS num_passengers
```

```
FROM railway r, passenger p WHERE r.trip_id = p.trip_id
```

```
GROUP BY r.source_station;
```

PYTHON-MYSQL CONNECTIVITY:

When we connect a database with Python, we need a library (mysql connector) that provides connectivity functionality. The Steps for Creating Database Connectivity Applications, there are mainly seven steps that must be followed in order to create a database connectivity application. These steps are:

Step 1 : Start Python.

Step 2 : Import the packages required for database programming. Step 3 : Open a connection to database.

Step 4 : Create a cursor instance. Step 5 : Execute a query.

Step 6 : Extract data from result set.

Step 7 : Clean up the environment.

CODE FOR CREATING A MYSQL DATABASE THROUGH PYTHON

```
import mysql.connector
```

```
mydb = mysql.connector.connect( host="localhost", user="kvbhopal", password=" kvbhopal ")
```

```
mycursor = mydb.cursor() mycursor.execute("CREATE DATABASE mydatabase")
```

CODE FOR CREATING A TABLE IN MYSQL THROUGH PYTHON

```
import mysql.connector
```

```
mydb = mysql.connector.connect(host="localhost",user=" kvbhopal",password=" kvbhopal ", database="mydatabase")
```

```
mycursor = mydb.cursor()
```

```
mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

CODE FOR INSERTING DATA IN A MYSQL TABLE THROUGH PYTHON

```
import mysql.connector

mydb = mysql.connector.connect(host="localhost", user="kvbhopal ",password=" kvbhopal ",
    database="mydatabase")

mycursor =mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s,%s)" val = ("Tom", "ABCD")

mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount, "record inserted.")
```

CODE FOR SELECTING AND PRINTING DATA FROM A MYSQL TABLE THROUGH PYTHON

```
import mysql.connector

mydb = mysql.connector.connect(host="localhost",user="kvbhopal", password="
    kvbhopal ", database="mydatabase")
mycursor = mydb.cursor()
mycursor.execute("SELECT * FROM
customers") myresult =
mycursor.fetchall()
for x in myresult:

    print(x)
```

CODE FOR DELETING A RECORD FROM MYSQL TABLE USING PYTHON

```
import
mysql.connec
```

```

tor mydb =
    mysql.connector.connect(host="localhost",user="yourusername",password="yourpassword",
        database="mydatabase")
mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE name = 'XYZ'"
    mycursor.execute(sql)
mydb.commit()

print(mycursor.rowcount, "record(s) deleted")

```

CODE FOR UPDATING A RECORD FROM MYSQL TABLE USING PYTHON

```

import
    mysql.connec
tor mydb =
    mysql.connector.connect(host="localhost",user="yourusername",password="yourpassword",
        database="mydatabase")
mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"
    mycursor.execute(sql)
mydb.commit()

print(mycursor.rowcount, "record(s) affected")

```

IMPORTANT CONNECTIVITY METHODS :

Connectivity Basics: Important functions and its purpose :

- connect(): Establishes a connection to the database.
- cursor(): Creates a cursor object to execute SQL queries.
- execute(): Executes SQL commands.
- commit(): Saves changes made during the transaction.

1. Fetching Data:

- fetchone(): Retrieves the next row of a query result set.
- fetchall(): Fetches all rows of a query result set.
- rowcount: Returns the number of rows affected by the last executed command.

2. Creating Database Applications:

Developing applications involves connecting to the database, executing queries, and processing results.

Utilize `connect()`, `cursor()`, and `execute()` for database operations.

3. Query Formatting:

- `'%s'` Format Specifier or `format()` : Used to dynamically insert values into SQL queries.
- Enhances query flexibility and security by preventing SQL injection attacks.

Example python Code :

for connecting a database named 'mydb' having user name as 'User'

, password as 'password' , host as 'localhost' and database port as 5432

```
# import the mysql-python connector
import mysql.connector
# Establishing a connection to the database

conn = psycopg2.connect(database="mydb", user="user",
    password="password", host="localhost", port="5432")

# Creating a cursor object to execute
queries cur = conn.cursor()
# Executing a query using %s format specifier

cur.execute("INSERT INTO table_name (column1, column2) VALUES (%s, %s)", (value1,
    value2))
```

Fetching data using fetchone() or

```
fetchall() data = cur.fetchone()
```

```
# Committing changes and closing the  
connection conn.commit()  
conn.close()
```

15 MLL (Minimum Learning Level) Questions:

1. What is database connectivity?

Ans: Database connectivity refers to connection and communication between an application and a database system.

2. What is connection? What is its role?

Ans: A Connection (represented through a connection object) is the session between the application program and the database. To do anything with database, one must have a connection object.

3. What is a result set?

Ans: A **result set** refers to a logical set of records that are fetched from the database by executing a query and made available to the application-program.

4. Which package must be imported in Python to create a database connectivity application?

Ans: There are multiple packages available through which database connectivity applications can be created in Python. One such package is **mysql.connector**.

Q5. Explain the following 'results' retrieval methods

A. `fetchone()` B. `rowcount()` C. `fetchall()`

Ans: (A) **fetchone()** :- The `fetchone()` method will return only one row from the result set in the form of tuple containing a record.

(B) **rowcount()** :- `cursor.rowcount()` that always return how many records have been retrieved so for using any of the `fetch..()` methods.

(C) **fetchall()** :- The `fetchall()` method return all the rows from the result set in the

form of a tuple congaing the records.

Q6.How can you use Python with MySQL?

ANS. Python can be used with MySQL in a number of ways. One way is to use the mysql-connector-python library, which is a MySQL driver written in Python. This library can be used to connect to a MySQL database and perform various operations, such as creating and executing SQL queries.

Q7.What is a cursor in the context of MySQL?

ANS. A cursor is a pointer that points to a specific location in a database table. In MySQL, cursors are used to iterate through the rows of a table and retrieve data from them.

Q8.What's the difference between autocommit and commit?

ANS. Autocommit is a database feature that automatically commits changes to the database as soon as they are made. This means that changes are immediately visible to other users and there is no need to explicitly call the commit() method. Commit, on the other hand, is a database feature that allows changes to be made to the database and then explicitly committed by the user. This allows the user to control when changes are made visible to other users.

Q9. How can you check if a table exists in MySQL?

ANS. You can check if a table exists in MySQL by using the SHOW TABLES command. This will show you a list of all the tables in the database. If the table you are looking for is not in the list, then it does not exist.

Q 10.How do you disconnect from the database?

ANS. .Use the close() method. db.close() closes the connection from the database.

Q11. How to Install MySQL Connector Library for Python?

Ans: For Python 2.7 or lower install using pip as:

```
pip install mysql-connector
```

For Python 3 or higher version install using pip3 as:

```
pip3 install mysql-connector
```

Q12. How to create Database in MySQL using Python ?

Syntax to Create new database in SQL is

```
CREATE DATABASE "database_name";
```

Now we create database using database programming in Python

```
import mysql.connector

db_connection = mysql.connector.connect( host=
    "localhost",
    user= "root", passwd=
    "root"
)

# creating database_cursor to perform SQL operation db_cursor =
    db_connection.cursor()
# executing cursor with execute method and pass SQL query db_cursor.execute("CREATE
    DATABASE my_first_db")
# get list of all databases
    db_cursor.execute("SHOW DATABASES")
    #print all databases
for db in db_cursor:
    print(db)
```

Q13. How do we create a cursor object?

The connection object returned by the connect() method is used to create a cursor object. You can create a cursor object using the cursor() method of the connection object/class. The cursor object is used to execute statements to perform database operations.

For example: `cursor1 = conn1.cursor()`

Here, `cursor1` is a cursor object created using connection object `conn1` using `cursor()` method.

Q14. What is the difference between `fetchall()` and `fetchnone()` methods?

Ans: The `fetchall()` method fetches all rows of a result set and returns a list of tuples. The `fetchnone()` method returns a single record as a list/ tuple and `None` if no more rows are available. Q15. How do we execute SQL query through Python?

Ans: To execute SQL queries through Python, cursor object is used along with `execute()` method. For example:

```
cursor1.execute("select * from emp;")
```

Here, `cursor1` is a cursor object which uses `execute` method to run the SQL query. The sql query is given in the form of a string. The output of the SQL query is stored in the cursor object in the form of a result set.