

+2

2025
புதிய பதிப்பு

COMPUTER SCIENCE FULL STUDY MATERIAL

PREPARED BY

P.PERIYASAMY M.Sc.,M.PHIL.,M.ED.,
DEPARTMENT OF COMPUTER SCIENCE
SHANTHINIKETHAN HR SEC SCHOOL
KARIMANGALAM - 635111.



Great School For Great Future

1. FUNCTION

SECTION - A

1. What is a subroutine?

- ✓ Subroutines are small sections of code.
- ✓ Used to do a particular task that can be used repeatedly.
- ✓ In Programming languages subroutines are called as Functions

2. Define function with respect to programming language.

- ✓ In Programming languages subroutines are called as Functions
- ✓ A function contains a set of code that works on many kinds of inputs and produces a output.

3. Write the inference you get from $X := (78)$.

- ✓ $X := (78)$ is an expression.
- ✓ (78) is a function definition.
- ✓ Definitions bind values to names,
- ✓ Here ,the value 78 being bound to the name 'X'.

4. Differentiate between interface and implementation.

Interface

- ✓ An interface is a set of action that an object can do ,but won't actually do it.
- ✓ In OOP all classes are the interface.
- ✓ The class template specifies to enable an object to be created and operated properly.
- ✓ Attributes and behavior of an object is controlled by interface(sending function).

Ex. $\text{sum}(5,6)$

Implementation

- ✓ Implementation carries out the instructions defined in the interface.
- ✓ In OOP all objects are processed and executed by the implementation

Ex.

```
let sum x y:  
return x + y
```

5. What is recursive function?

A function definition which call itself is called recursive function

Ex. let rec sum x y:

return x + y

6. Which of the following is a normal function**definition and which is recursive function definition****i) let rec sum x y:****return x + y****ii) let disp :****print 'welcome'****iii) let rec sum num:****if (num!=0) then return num + sum (num-1)****else return num**

1. Recursive function
2. Normal function
3. Recursive function

SECTION - B**1. Mention the characteristics of Interface**

- ✓ An interface is a set of action that an object can do ,but won't actually do it.
- ✓ In OOP all classes are the interface.
- ✓ The class template specifies to enable an object to be created and operated properly.
- ✓ Attributes and behavior of an object is controlled by interface(sending function).

2. Why strlen() is called pure function?

- ✓ strlen() is a pure function
- ✓ because the function takes one variable as a parameter, and accesses it to find its length.
- ✓ This function reads external memory but does not change it,

3. What is the side effect of impure function? Give example.

- ✓ The variables used inside the function may cause side effects though the functions which are not passed with any arguments.
- ✓ When a function depends on variables or functions outside of its definition block, For example random(),date()
- ✓ Here the function random() is impure because it will not get the same result each time we call the function.

4. Differentiate pure and impure function**Pure Function**

- ✓ It will give exact result when the same arguments are called.
- ✓ The return value depends on its arguments passed.
- ✓ Same return values.

- ✓ They do not have any side effects.
- ✓ They do not modify the arguments Ex strlen()

Impure Function

- ✓ It will not give exact result
- ✓ The return value does not depend on its arguments passed.
- ✓ Different return values
- ✓ They have side effects.
- ✓ They may modify the arguments Ex. date()

SECTION - C

1. What are called Parameters and write a note on

(i) Parameter without Type (ii) Parameter with Type

- ❖ Parameters are the variables in a function definition.
- ❖ Arguments are the values which are passed to a function definition.

(i) Parameter without Type

- ✓ Data types are not mentioned with parameter in function definition.

Ex. let add a b:=
 return a+b

Some compiler solves this type algorithmically, but some require the type.

(ii) Parameter with Type

- ✓ Data types are mentioned with parameter in function definition.
- ✓ In parameter with type, the parentheses are mandatory.
Ex . let add (a: int) (b: int) :=
 return a+b
- ✓ here ,explicitly write down Data types.
- ✓ This can help with debugging such an error message.

2.Explain with example Pure and impure functions (or)

Differentiate pure and impure function

Pure Function

- ✓ It will give exact result when the same arguments are called.
- ✓ The return value depends on its arguments passed.
- ✓ Same return values.
- ✓ They do not have any side effects.
- ✓ They do not modify the arguments Ex strlen()

Impure Function

- ✓ It will not give exact result
- ✓ The return value does not depend on its arguments passed.
- ✓ Different return values
- ✓ They have side effects.
- ✓ They may modify the arguments Ex. date()

3. Identify in the following program

```
let rec gcd a b :=
```

```
if b <> 0 then gcd b (a mod b)
```

```
else return a
```

i) Name of the function

Ans: gcd

ii) Identify the statement which tells it is a recursive function

Ans : let rec gcd

iii) Name of the argument variable

Ans: a , b

iv) Statement which invoke the function recursively

Ans: gcd b (a mod b)

v) Statement which terminates the recursion

Ans: return a

4. Explain with an example interface and implementation.**Interface**

- ✓ An interface is a set of action that an object can do ,but won't actually do it.
- ✓ In OOP all classes are the interface.
- ✓ The class template specifies to enable an object to be created and operated properly.
- ✓ Attributes and behavior of an object is controlled by interface(sending function).

Ex.

```
sum(5,6)
```

Implementation

- ✓ Implementation carries out the instructions defined in the interface.
- ✓ In OOP all objects are processed and executed by the implementation

Ex.

```
let sum x y:
return x + y
```

2.DATA ABSTRACTION

SECTION - A

Define abstraction.

- ✓ The process of providing only the essentials and hiding the details is known as abstraction.

1. What is abstract data type?

- ✓ Abstract Data type (ADT) is defined by a set of value and a set of operations.
- ✓ The data representation is unknown
- ✓ Here ,how object can be used.

2. Differentiate constructors and selectors.

Constructors

- ✓ Constructors are functions that build the abstract data type.
- ✓ Constructors create an object, bundling together different pieces of information,
Ex. city = makecity (name, lat, lon) ---- Constructors

Selectors

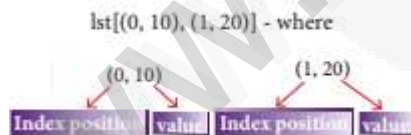
- ✓ Selectors are functions that retrieve information from the data type.
- ✓ Selectors extract individual pieces of information from the object.

Ex. getname(city)
 getlat(city) -----Selectors
 getlon(city)

3. What is a Pair? Give an example.

- ✓ Way of bundling two values together into one can be considered as a pair.
- ✓ Pair is made up of list or Tuple
- ✓ Therefore List can be called as Pairs.

Ex. a= [(0, 10), (1, 20)] or b= (0, 10)



here ,(0, 10) (1, 20) - are pairs

4. What is a Tuple? Give an example.

- ✓ Elements are enclosed by parenthesis (,).
- ✓ The elements are separated by comma(,).
- ✓ Elements are unchangeable.
- ✓ Tuples are similar to List Ex. a= (2,4,6)

5. What is a List? Give an example.

- ✓ It is an ordered collection of values
- ✓ Enclosed within square brackets [].
- ✓ Each value of a list is called as element ,separated by comma(,).
- ✓ It consists of numbers, characters, strings and even the nested lists as well.
- ✓ The elements can be modified or mutable.
- ✓ lists are similar to arrays

ex. m=[10,25.5,"rmk",[4,5,6]]

SECTION - B**1. Differentiate Concrete data type and abstract datatype.**

Concrete data type	Abstract data type.
<ul style="list-style-type: none"> ✓ They are direct implementations of concept ✓ The data representation is known ✓ Object is actually implemented 	<ul style="list-style-type: none"> ✓ Abstract Data type (ADT) is defined by a set of value and a set of operations ✓ The data representation is unknown ✓ Here how object can be used.

2. Which strategy is used for program designing? Define that Strategy. Or What is wishful thinking in program designing?

- ✓ 'wishful thinking' strategy is used for program designing.
- ✓ It is making decisions according to what might be imagine instead of by appealing to reality.

3. Identify Which of the following are constructors and selectors?

- (a) N1=number() (b) acceptnum(n1)
 (c) displaynum(n1) (d) eval(a/b) (e) x,y= makeslope
 (m), makeslope(n) (f) display()

- A. Constructor
 B. Selector
 C. Selector
 D. selector
 E. Constructor
 F. Selector

4. What are the different ways to access the elements of a list. Give example.

The elements of a list can be accessed in two ways.

1. Multiple assignment a := [10, 20]

x, y := a ;

here x will become 10 and y is 20 binds each element to a different name

2. Element selection operator

- ✓ Accessing the elements in a list is by the element selection operator,
- ✓ It is expressed using square brackets

Ex.a=[10,20]

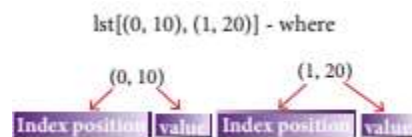
a[0]

10

a[1]

20

Method III



Represent list similar to a

1.) a=[(0, 10), (1, 20)] – where (0,10) (1,20) -- pairs

5. Identify Which of the following are List, Tuple and class ?

- (a) arr [1, 2, 34] (b) arr (1, 2, 34)
 (c) student [rno, name, mark]
 (d) day= ('sun', 'mon', 'tue', 'wed')
 (e) x= [2, 5, 6.5, [5, 6], 8.2]
 (f) employee [eno, ename, esal, eaddress]

- a. List
 b. Tuple
 c. Class
 d. Tuple
 e. List
 f. Class

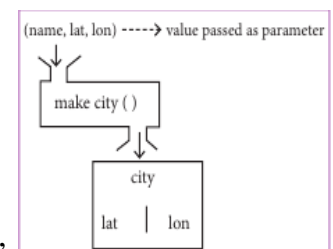
SECTION - C**1.How will you facilitate data abstraction. Explain it with suitable example**

- ❖ To facilitate data abstraction, we need to create two types of functions:
 constructors and selectors.

Constructors

- ✓ Constructors are functions that build the abstract data type.
- ✓ Constructors create an object, bundling together different pieces of information,

Ex. city = makecity (name, lat, lon) ---- This is Constructor which create object namely city.



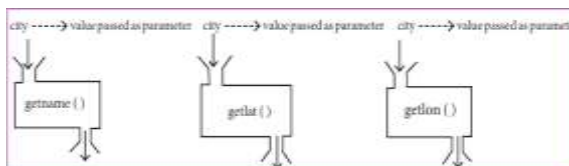
Selectors

- ✓ Selectors are functions that retrieve Information from the data type.
- ✓ Selectors extract individual pieces of information from the object. Ex.

getname(city)

getlat(city) ----- Selectors

getlon(city)



2.What is a List? Why List can be called as Pairs. Explain with example

List

- ✓ It is an ordered collection of values
- ✓ Enclosed within square brackets [].
- ✓ Each value of a list is called as element ,separated by comma(,).
- ✓ It consists of numbers, characters, strings and even the nested lists as well.
- ✓ The elements can be modified or mutable.
- ✓ lists are similar to arrays

Ex. m=["welcome",10,20.5,[30,40]]

Why List can be called as Pairs:

- ✓ Pair is made up of list or Tuple
- ✓ Therefore List can be called as Pairs.

Ex. a= [(0, 10), (1, 20)] , here ,(1, 20) (0, 10) - are pairs

The elements of a list can be accessed in two ways.

1. Multiple assignment

lst := [10, 20]

x, y := lst ; here x will become 10 and y is 20

2. Element selection operator

- ✓ Accessing the elements in a list is by the element selection operator.

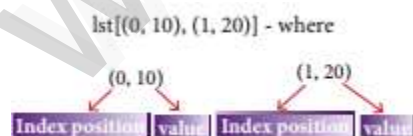
Ex. a:=[10,20]

a[0]

10

3.Method

Represent list similar to a



2.) [(0, 10), (1, 20)] – where (0,10) (1,20) – pairs Therefore List can be called as Pairs.

3.How will you access the multi-item. Explain with example.

- ✓ lists does not allow to name the various parts of a multi- item object.
- ✓ In OOPs class construct used to represent multi-part objects

Ex.

class Person:

creation()

firstName := " "

lastName := " "

id := " "

email := " "

Here,

Person

creation ()

first Name,last Name, id,email

Let main() contains

p1:=Person() - object p1 created

firstName := " Padma "

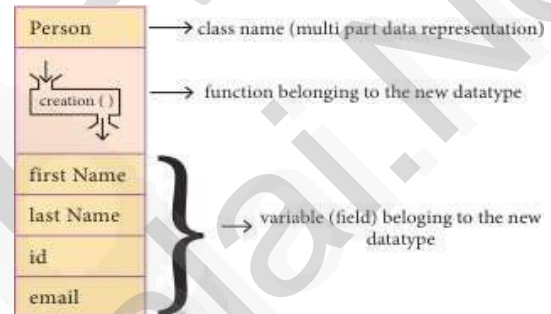
lastName :="Baskar"

id :="994-222-1234"

email="compsci@gmail.com"

output of firstname : Padma

```
class Person:
    creation( )
    firstName := " "
    lastName := " "
    id := " "
    email := " "
```



Let main() contains

p1:=Person()	statement creates the object.
firstName := " Padmashri "	setting a field called firstName with value Padmashri
lastName :="Baskar"	setting a field called lastName with value Baskar
id :="994-222-1234"	setting a field called id value 994-222-1234
email="compsci@gmail.com"	setting a field called email with value compsci@gmail.com
-- output of firstName : Padmashri	

SECTION - C

1. Explain the types of scopes for variable or LEGB rule with example.**Define LEGB rule**

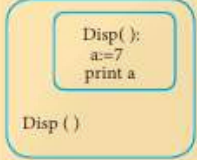
- ✓ The *LEGB* rule is used to decide the order for scope resolution.
- ✓ The scopes are listed below in terms of hierarchy (highest to lowest).

There are 4 types of Variable Scope, They are Local(L) , Enclosed(E), Global(G), Built-in (B)

Local(L) scope

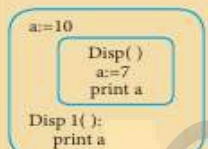
- ✓ Variables defined within current function / class.
- ✓ It cannot be accessed outside of the function.

For example

	Entire program	Output of the Program
1. Disp(): 2. a:=7 3. print a 4. Disp()		7

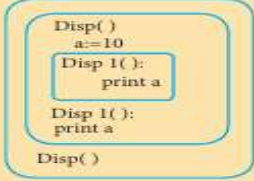
Global(G) Scope

- ✓ A variable is declared outside of all the functions in a program is known as global variable.
- ✓ It can be accessed inside or outside of all the functions in a program.

	Entire program	Output of the Program
1. a:=10 2. Disp(): 3. a:=7 4. print a 5. Disp() 6. print a		7 10

Enclosed(E) scope

- ✓ A variable which is declared inside a function, contains another function definition with in it,
- ✓ The inner function can also access the variable of the outer function.
- ✓ This scope is called enclosed scope.

	Entire program	Output of the Program
1. Disp(): 2. a:=10 3. Disp1(): 4. print a 5. Disp1() 6. print a 7. Disp()		10 10

Built-in(B) Scope

- ✓ Built-in scope has all the names that are pre-loaded into program scope when we start the compiler or Interpreter

2.What are the Characteristics of Modules? Define Module.

- ✓ A module is a part of a program.
- ✓ Programs are composed of one or more independently developed modules.
- ✓ A single module can contain one or several statements closely related each other
- ✓ Modules contain instructions, processing logic, and data.
- ✓ Modules can be separately compiled and stored in a library.
- ✓ Modules can be included in a program.
- ✓ Module segments can be used by invoking a name and some parameters.
- ✓ Module segments can be used by other modules.

3. What are the benefits of using modular programming include?

What is Modular programming

- The process of subdividing a computer program into separate sub-programs is called Modular Programming
 - ✓ Less code to be written.
 - ✓ Can be reuse
 - ✓ Programs can be designed more easily
 - ✓ It allows many programmers to collaborate on the same application.
 - ✓ The code is stored across multiple files.
 - ✓ Code is short, simple and easy to understand.
 - ✓ Errors can easily be identified,
 - ✓ The same code can be used in many applications.
 - ✓ The scoping of variables can easily be controlled.

4. ALGORITHMIC STRATEGIES

SECTION - A

1. What is an Algorithm?

- ✓ An algorithm is a **finite set of instructions** to do a particular task.
- ✓ It is a **step-by-step procedure** for solving a given problem.
- ✓ It can be implemented in any suitable programming language.
- ✓ There is **no specific rules** for algorithm

2. Write the phases of performance evaluation of an algorithm.

- ✓ Analysis of algorithms and performance evaluation can be divided into two different phases:

1. A Priori estimates: This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.

2. A Posteriori testing: This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected

3. What is Insertion sort?

- ✓ Insertion sort is a simple sorting algorithm.
- ✓ It works by taking elements from the list one by one and inserting them in their correct position in to a new sorted list.
- ✓ This algorithm builds the final sorted array at the end.

4. What is sorting?

- ✓ Sorting is the **process of arranging data** in ascending or descending order.
- ✓ Sort algorithms are three types, They are

- 1.Selection sort
- 2.Bubble sort
- 3.Insertion sort

5.What is searching ? Write its types?

Searching is a process of finding a **particular element** in a given **set of elements**.

Types of searching are

- ✓ Linear
- ✓ Binary

SECTION - B

1.List the Characteristics of an Algorithm

- ✓ Input
- ✓ Output
- ✓ Finiteness
- ✓ Definiteness
- ✓ Effectiveness
- ✓ Correctness
- ✓ Simplicity
- ✓ Unambiguous
- ✓ Feasibility
- ✓ Portable
- ✓ Independent

2. Discuss about Algorithmic complexity and its types. Or What is Complexity of an Algorithm?

- ✓ The complexity of an algorithm gives the **running time** and/or **the storage space** required by the algorithm .

Time complexity

- ✓ Time complexity is measured by **the number of steps** taken by the algorithm to complete the process.

Space complexity

- ✓ Space complexity is measured by **the maximum memory space required** by the algorithm to complete the process.

3.What are the factors that influence time and space complexity.

The time and the space are the two main factors to measure the efficiency of algorithm.

- ✓ **Time Factor** -Time is measured by **counting the number of key operations** in the algorithm.
- ✓ **Space Factor** - Space is measured by the **maximum memory space** required by the algorithm.

4.Write a short note on Asymptotic Notations.

Asymptotic Notations are **languages** that uses meaningful statements about **time and space** complexity. There are three asymptotic notations,

(i) Big O

- ✓ Big O is used to describe the **worst-case** of an algorithm.

(ii) Big Ω

- ✓ Big Omega is the **reverse** Big O,
- ✓ Big Omega is used to describe **the lower bound**.(best-case).

(iii) Big Θ

- ✓ When an algorithm has a complexity with **lower bound = upper bound**,
- ✓ An algorithm has a complexity **$O(n \log n)$** and **$\Omega(n \log n)$** , it's actually has the complexity **$\Theta(n \log n)$** [Best-case and worst-case]

5. What do you understand by Dynamic programming?

- ✓ Dynamic programming is an algorithmic design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.
- ✓ Dynamic programming approach is similar to divide and conquer. The given problem is divided into smaller and yet smaller possible sub-problems.
- ✓ Dynamic programming is used whenever problems can be divided into similar sub-problems. so that their results can be re-used to complete the process.
- ✓ Dynamic programming approaches are used to find the solution in optimized way. For every inner sub problem, dynamic algorithm will try to check the results of the previously solved sub-problems. The solutions of overlapped sub-problems are combined in order to get the better solution.

SECTION - C**1.Explain the Characteristics of an Algorithm**

Input :

- ✓ Zero or more values to be supplied.

Output:

- ✓ At least one value is produced.

Finiteness :

- ✓ Algorithms must terminate after finite number of steps.

Definiteness:

- ✓ All operations should be well defined.

Effectiveness:

- ✓ Every instruction must be carried out effectively.

Correctness:

- ✓ The algorithms should be error free.

Simplicity :

- ✓ Easy to implement.

Unambiguous :

- ✓ Each of its steps and their inputs/outputs should be clear and unambiguous must lead to only one meaning.

Feasibility:

- ✓ Should be feasible with the available resources.

Portable:

- ✓ An algorithm should be able to handle all range of inputs.

Independent:

- ✓ An algorithm should be independent of any programming code

2. Discuss about Linear search algorithm.

- ✓ Linear search also called sequential search
- ✓ It used to find a particular value in a unordered list.
- ✓ It checks the search element with each element in sequence until the desired element is found.

Pseudo code

1. In the array using for loop
2. In every iteration, compare the search value with the current value of the list.
 - ✓ If the values match, display the current index and value of the array
 - ✓ If the values do not match, move on to the next array element.
- 3.If no match is found, display not found.

Example

Input: values[] = {13,31,28,11,2}

Target = 28

Output:2

3. What is Binary search? Discuss with example.

- ✓ Binary search also called half-interval search algorithm.
- ✓ It used to find a particular value in a ordered list.
- ✓ It can be done as divide-and-conquer search algorithm and executes in logarithmic time.

Procedure for Binary search

1. Start with the middle element:
 - If the search element is equal to the middle element of the array i.e., the middle value = number of elements in array/2, then return the index of the middle element.

- If not, then compare the middle element with the search value,
 - If the search element is greater than the number in the middle index, then select the elements to the right side of the middle index, and go to Step-1.
 - If the search element is less than the number in the middle index, then select the elements to the left side of the middle index, and start with Step-1.
2. When a match is found, display success message with the index of the element matched.
 3. If no match is found for all comparisons, then display unsuccessful message.

Binary Search Working principles

List of elements in an array must be sorted first for Binary search. The following example describes the step by step operation of binary search. Consider the following array of elements, the array is being sorted so it enables to do the binary search algorithm. Let us assume that the search element is 60 and we need to search the location or index of search element 60 using binary search.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

First, we find index of middle element of the array by using this formula :

$$\text{mid} = \text{low} + (\text{high} - \text{low}) / 2$$

Here it is, $0 + (9 - 0) / 2 = 4$ (fractional part ignored). So, 4 is the mid value of the array.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

Now compare the search element with the value stored at mid value location 4. The value stored at location or index 4 is 50, which is not match with search element. As the search value 60 is greater than 50

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

Now we change our low to mid + 1 and find the new mid value again using the formula.

$$\text{low} = \text{mid} + 1$$

$$\text{mid} = \text{low} + (\text{high} - \text{low}) / 2$$

Our new mid is 7 now. We compare the value stored at location 7 with our target value 60.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

The value stored at location or index 7 is not a match with search element, rather it is more than what we are looking for. So, the search element must be in the lower part from the current mid value location

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

The search element still not found. Hence, we calculated the mid again by using the formula.

$high = mid - 1$

$mid = low + (high - low)/2$

Now the mid value is 5.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

Now we compare the value stored at location 5 with our search element. We found that it is a match

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

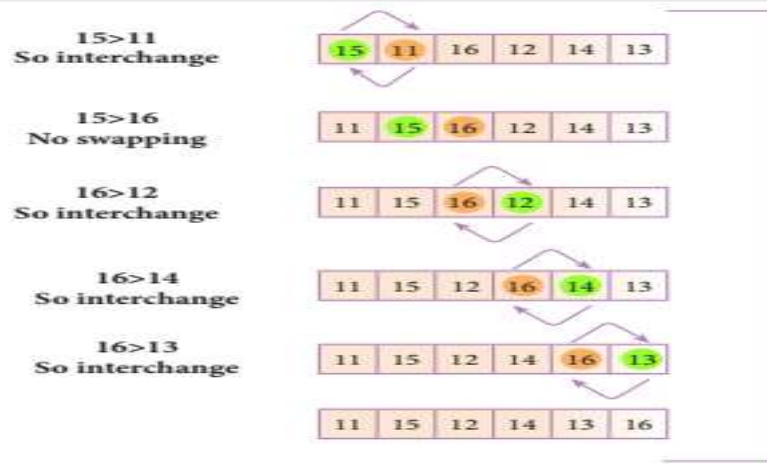
We can conclude that the search element 60 is found at location or index 5. For example if we take the search element as 95, For this value this binary search algorithm return unsuccessful result.

4. Explain the Bubble sort algorithm with example.

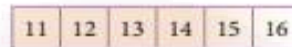
- ✓ Bubble sort is a simple sorting algorithm.
- ✓ It starts at the beginning of the list
- ✓ It compares each pair of adjacent elements and swaps and to be continued until no swaps are needed.

Pseudo code/ Algorithm

- ✓ Start with the first element i.e., index = 0,
- ✓ compare the current element with the next element of the array.
- ✓ If the current element > the next element of the array, swap them.
- ✓ If the current element < the next element, move to the next element.
- ✓ Go to Step 1 and repeat until end of the index is reached.
- ✓ For example, Let's consider an array with values {15, 11, 16, 12, 14, 13}



At the end of all the iterations we will get the sorted values in an array as given below:



5. Explain the concept of Dynamic programming.

- ✓ Dynamic programming approach is similar to divide and conquer.
- ✓ The given problem is divided into smaller and yet smaller possible sub-problems.

Steps to do Dynamic programming

- ✓ The given problem will be divided into smaller overlapping sub-problems.
- ✓ An optimum solution for the given problem can be achieved by using result of smaller sub-problem.
- ✓ Dynamic algorithms uses Memoization.

Example

Fibonacci Iterative Algorithm with Dynamic programming approach

Initialize $f_0=0$, $f_1=1$

step-1: Print f_0 and f_1

step-2: Calculate fibonacci $fib \leftarrow f_0 + f_1$

step-3: Assign $f_0 \leftarrow f_1$, $f_1 \leftarrow fib$

step-4: Print fib

step-5: Goto step-2 and repeat upto number of terms.

5.PYTHON-VARIABLES AND OPERATORS

SECTION - A

1. What are the different modes that can be used to test Python Program?

- ✓ In Python, programs can be written in two ways namely **interactive mode** and **Script mode**.

Interactive mode:

- ✓ It allows us to write codes in **Python command prompt (>>>)** directly and the interpreter displays the **result(s) immediately**.
- ✓ It is used as a **simple calculator**.

Script mode:

- ✓ Script mode is used to **create, edit** python source file and **store** as separate file with **.py** extension.

2. Write short note on Tokens?

- ✓ Python breaks each logical line into a sequence of **elementary lexical** components known as Tokens.

The normal token types are

- 1) Identifiers,
- 2) Keywords,
- 3) Operators,
- 4) Delimiters and
- 5) Literals.

3.What are the different operators that can be used in Python?

- ✓ Operators are **special symbols** which represent arithmetic, conditional matching etc
Operators are categorized as

- Arithmetic operators
- Relational or Comparative operators
- Logical operators
- Assignment operators
- Conditional operator

4.What is a literal? Explain the types of literals ? Literal

- Literal is a raw data given in a variable or constant. In Python, they are

1. Numeric
2. String
3. Boolean.

5. Write short notes on Exponent data?

- An Exponent data contains decimal **digit part**, **decimal point**, **exponent part** followed by one or more digits.

123.34, 456.23, 156.23 # Floating point data

12.E04, 24.e04 # Exponent data

SECTION - B

1. Write short notes on Arithmetic operator with examples.

Arithmetic operators

- ✓ It takes two **operands** and performs a **calculation** on them.
- ✓ They are used for simple arithmetic.

1. *+* (Addition) $\ggg a + b$
2. *-* (Subtraction) $\ggg a - b$
3. *** (Multiplication) $\ggg a * b$
4. */* (Division) $\ggg a / b$
5. *%* (Modulus) $\ggg a \% 30$
6. **** (Exponent) $\ggg a ** 2$
7. *//* (Floor Division) $\ggg a // 30$ (integer division)

2. What are the assignment operators that can be used in Python?

Assignment operators

- ✓ *=* is a simple assignment operator
- ✓ Used to assign values to variable.
- ✓ There are various compound operators like

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

a=10

a+=3

print(a)

output: 13

3. Explain Ternary operator with examples

- ✓ **Ternary operator** is also known as **conditional operator**
- ✓ It is used to evaluate **based on** a condition being **true** or **false**.

Syntax:

- ✓ **Variable Name** = [on_true] if [Test expression] else [on_false]

Example

m= 50 if 50<70 else 70

print(m)

output : 50

2. Explain input() , print() functions with examples

- ✓ The print() is used to display result on the screen.
- ✓ The print() function is used to display result on the screen.

Syntax:

print (“string to be displayed as output ”)

print (variable)

print (“String to be displayed as output ”, variable)

print (“Str1 ”, var1, “Str 2”, var2,)

Example

```
>>>print (“Welcome to Python”)
```

Welcome to Python

```
>>> x = 5
>>> print (x)
5
```

- ✓ Comma (,) is used as a separator in print () to print more than one item
- ✓ The input() helps to enter data at run time by the user .
- ✓ In Python, input() function is used to accept data as input at run time.

1) input() with prompt string

Syntax:

variable =input (“prompt string”)

- ✓ It is used to display message or escape sequence on the monitor.

Example

```
>>> c=input (“Enter Your City: ”)
Enter Your City: chennai
```

2) input() without prompt string

- ✓ No message is displayed on the screen.

Syntax :variable =input()

```
>>> d=input()
```

ELANGO

```
>>> print(d)
```

```
>>> ELANGO
```

- ✓ The input () accepts all data as string or characters.
- ✓ Numerical value should be explicitly converted into numeric data type.

int() - integer

float()- float .

Example:

```
x = int (input())
```

```
print (x)
```

Output: 34

```
4)x,y=int (input("No 1 :")),int(input("No 2:"))
```

3.Discuss in detail about Tokens in Python

✓ Python breaks each logical line into a sequence of **elementary lexical** components known as Tokens.

The normal token types are

- 1) Identifiers,
- 2) Keywords,
- 3) Operators,
- 4) Delimiters and
- 5) Literals.

1) Identifiers,

✓ An Identifier is a **name** used to identify a **variable, function, class, module** or **object**.

Rules

- ✓ It has **alphabet, digits** and **underscore (_)** only.
- ✓ It must **start** with an **alphabet** or **underscore (_)**.
- ✓ It is **case sensitive**
- ✓ Identifiers must **not be** a python **keyword**.

Valid - a,sum,tot_marks,m2

Invalid - 2m,a+b,tot-marks

2) Keywords,

- ✓ Keywords are **special words** used by Python interpreter to recognize the structure of program.
- ✓ They **cannot be used** for any other purpose.

Ex. for,if,is,as,or etc..

3) Operators

- ✓ Operators are **special symbols** which represent arithmetic, conditional matching etc
- ✓ Operators are categorized as

1. Arithmetic operators (+,-,*,/,)
2. Relational or Comparative operators(<,>,>=,<=)
3. Logical operators(and or,not)

4. Assignment operators(=)

5. Conditional operator(

4) Delimiters

- ✓ Python uses the **symbols** and **symbol combinations** as delimiters in **expressions, lists, dictionaries and strings**.

Ex. (,) {}, ' , [,]etc..

5) Literals.

- ✓ Literal is a raw data given in a variable or constant. In Python,
- ✓ They are 1.Numeric 2. String 3.Boolean

6.

6.CONTROL STATEMENT

SECTION - A

1.What is Control statements? Or Define Control

Structure. (Or)List the control structures in Python

- ✓ A program statement that causes a **jump of control** from **one part** of the program to **another** is called **control structure** or **control statement**.

There are three important control structures

- ✓ Sequential
- ✓ Alternative or Branching
- ✓ Iterative or Looping

2. Write note on break statement.

- ✓ The **break** statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.

3.Write is the syntax of if..else statement

Syntax:

if <condition>:

statements-block 1

else:

statements-block 2

4.Write note on range () in loop

- ✓ sequence refers to the initial, final and increment value.
- ✓ In python, range() function used for the sequence.
- ✓ **range()** generates a list of values starting from **start** till **stop-1**.

Syntax of range()

range (start,stop,step)

Where,

- ✓ start – refers to the **initial** value (**0** is default)
- ✓ stop – refers to the **final** value (stop-1)
- ✓ step – refers to **increment** value(**1** is default)

start and step are **optional** part.

SECTION - B

1. Write a program to display

A

A B

A B C

A B C D

A B C D E

```
for c in range(65,70):
```

```
    for a in range (65,c+1):
```

```
        print (chr(a),end=' ')
```

```
        print(end='\n')
```

```
    c+=1
```

2. Write note on if..else structure.

- ✓ The **if ..else** statement used to check the **true block** as well as the **false block**.

Syntax:

```
if <condition>:
```

```
    statements-block 1
```

```
else:
```

```
    statements-block 2
```

- ✓ if the **condition** is **true** ,True block statements will be executed
- ✓ if the **condition** is **False** ,False block statements will be executed

Example:

```
x=int (input("Enter your Age:"))
```

```
if x>=18:
```

```
    print ("You are eligible for voting")
```

```
else:
```

```
    print("You are not eligible for voting")
```

output 1

Enter your Age:45

You are eligible for voting

Output 2

Enter your Age:12

You are not eligible for voting

3. Using if..else..elif statement write a suitable program to display largest of 3 numbers.

```
a,b,c=int(input()),int(input()),int(input())
if a>b and a>c:
    print("A is greater")
elif b>a and b>c:
    print("B is greater")
else:
    print("C is greater")
```

4. Write the syntax of while loop.

Syntax:

```
initialization
while condition:
    statements block 1
    updation
else:
    statements
```

5. List the differences between break and continue statements

i) break statement

- ✓ The **break** statement **terminates** the **current loop**.
- ✓ The control **comes out** of the loop and **starts** executing **after** the loop structure.
- ✓ In **nested** loop, break will terminate the **innermost loop**.
- ✓ If a loop is left by break, the else part is **not executed**.

Ex.

for w in "Jump Statement":

```
if w=="e":
    break
else:
    print (w, end= ' ')
```

print("END")

output :J u m p S t a t

(ii) continue statement

- ✓ Continue statement is used to **skip** the remaining part of a loop .
- ✓ It starts with **next iteration**.
- ✓

Example:

```
for w in "Jump Statement":
    if w=="e":
        continue
    else:
        print (w, end= ' ')
print("\nEND")
Output :Jump Statmnt
```

SECTION - C

1.Explain for loop with an example or Write note on range () in loop

- ✓ for loop is the most comfortable loop.
- ✓ It is an entry check loop.
- ✓ The statements block1 is executed repeatedly till the condition is True.
- ✓ else part is executed when the condition is False.
- ✓ The else part is optional

Syntax:

```
for counter_variable in sequence:
    statements-block 1
else: - optional
    statements
```

- ✓ counter_variable is similar to the control variable *Sequence*
- ✓ sequence refers to the initial, final and increment value.
- ✓ In python, range() function used for the sequence.
- ✓ range() generates a list of values starting from start till stop-1.

Syntax of range()

```
range (start,stop,step)
```

Where,

- ✓ start – refers to the initial value (0 is default)
- ✓ stop – refers to the final value (stop-1)
- ✓ step – refers to increment value(1 is default) start and step are optional part.

Example :

```
n=int(input('Enter a number : '))
```

```
f=1
```

```
for x in range(1, n+1):
```

f=f * x

print('Factorial = ',f)

OUTPUT:

Enter a number : 5

Factorial = 120

2. Write a detail note on if..else..elif statement with suitable example.

Nested if..elif...else statement:

- ✓ When we need to construct a nested **if** statement(s) then **elif** clause can be used instead of **else**.
- ✓ **elif** can be considered to be **else if**.

Syntax:

If (condition1):

Block statements 1

elif (condition 2):

block statements 2

else :

block statements 3

- ✓ No limit of using '**elif**' clause, but an '**else**' clause should be placed at the **end**. In the syntax of **if..elif..else**,
- ✓ **condition-1** is tested if it is **true** then **block statements 1** is executed,
- ✓ otherwise the control **checks condition-2**, if it is **true** then **block statements 2** is executed
- ✓ otherwise **else** part is executed.

Example:

a,b,c=int(input()),int(input()),int(input())

if a<b and a<c:

print(a," Smallest")

elif b<a and b<c:

print(b," Smallest ")

else:

print(c," Smallest ")

output

5 2 3

2 Smallest

3. Write a program to display all 3 digit odd numbers.

```
for i in range(101,1000,2):
```

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

output:

101,103,105,107....999.

4. Write a program to display multiplication table for a given number.

```
n=int(input())
```

```
for i in range(1,11):
```

```
    print(n,"X",i,"=",n*i)
```

Output:

```
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
10 x 2 = 20
```


7.PYTHON FUNCTIONS

SECTION - A

1. Define function and its advantages.

Functions are named blocks of code that are designed to do one specific job.

Advantage :

- ✓ Functions help us to divide a program into **modules**
- ✓ It avoids **repetition**
- ✓ Function code can be **reuse**.
- ✓ It provides better **modularity** for application.

2. What are the types of function in python?

There are four types of functions in python, they are

- ✓ **User Defined** functions
- ✓ **Anonymous or Lambda** functions
- ✓ **Recursion** functions
- ✓ **Built-in** functions

3.What are the advantages of user defined functions?

Functions are named blocks of code that are designed to do one specific job.

Advantage :

- ✓ It avoids repetition and makes high degree of code reusing.
- ✓ It provides better modularity for your application.

4.What is meant by scope of variable? Mention its types.

- ✓ Scope refers to the **accessibility of a variable**.
- ✓ Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it.
- ❖ There are **local** and **Global** scope

Local Scope

- ✓ A variable declared **inside** the function's body is known as local variable.

Global Scope

- ✓ A global variable, can be used **anywhere in the program**.
- ✓ It is created **outside** the **function/block**.

5. Define global scope.

Global Scope

- ✓ A global variable, can be used **anywhere in the program**.
- ✓ It is created **outside** the **function/block**.

Rules of global Keyword

- ✓ To define a variable **outside a function**, it's global by **default**.
- ✓ **global** keyword used to **modify** the global variable **inside** a function.

Example:

```
c = 10
```

```
def add():
```

```
    c=c+2
```

```
    print(c)
```

```
add()
```

output: local variable 'c' referenced before assignment

6. What is base condition in recursive function?

- ✓ A recursive function calls itself.
- ✓ The **condition** that is applied in any **recursive function** is known as **base condition**.
- ✓ If the **base condition** is True then the **program gives output and exits**.
- ✓ otherwise it will execute like an **infinite loop**

7. How to set the limit for recursive function? Give an example.

- ✓ Python **stops** calling **recursive** function after **1000** calls by **default**.
- ✓ To change the limit, use **sys.setrecursionlimit(limit_value)** function by importing **sys**

Example:

```
import sys
```

```
sys.setrecursionlimit(3000)
```

```
def fact(n):
```

```
    if n == 0:
```

```
        return 1
```

```
    else:
```

```
        return n * fact(n-1)
```

```
print(fact (2000))
```

SECTION - B

1. Write the rules of local variable.

- ✓ Scope refers to the accessibility of a variable.
- ✓ There are local and Global scope

Local Scope

- ✓ A variable declared **inside** the function's body is known as local variable.

Rules of local variable

- ✓ It can be accessed only within the function/block.
- ✓ It is created inside the function/block.
- ✓ A local variable only exists while the function is executing.
- ✓ The formal arguments are also local to function.

Ex

```
def loc():
```

```
    y=4
```

```
    loc()
```

```
print(y)
```

2. Write the basic rules for global keyword in python.**The basic rules for *global* keyword in Python are:**

- ✓ When we define a variable outside a function, it's global by default. You don't have to use global keyword.
- ✓ We use global keyword to modify the value of the global variable inside a function.
- ✓ Use of global keyword outside a function has no effect

Use of global Keyword

```
c = 10
```

```
def add():
```

```
    global c
```

```
    c=c+2
```

```
    print(c)
```

```
add()
```

output: 12

3. What happens when we modify global variable inside the function?

- ✓ Thrown an error because,
- ✓ Without using the global keyword we cannot modify the global variable inside the function.
- ✓ but we can only access the global variable

Example:

```

x = 0                                # global variable
def add():
    global x
    x = x + 5                        # increment by 5
    print ("Inside add() function x value is :", x)
add()
print ("In main x value is :", x)

```

Output:

Inside add() function x value is : 10

In main x value is : 5

#value of x changed outside the function

4. Differentiate ceil() and floor() function?

ceil()	floor()
Returns the smallest integer $\geq x$	Returns the largest integer $\leq x$
math.ceil(x)	math.floor(x)
X=26.7 y=-26.7 z=-23.2	X=26.7 y=-26.7 z=-23.2
Print(math.ceil(x)) = 27	Print(math.floor(x)) = 26
Print(math.ceil(y)) = -26	Print(math.floor(y)) = -27
Print(math.ceil(z)) = -23	Print(math.floor(z)) = -24

5. Write a Python code to check whether a given year is leap year or not.

```

n=int(input("Enter the year"))
if(n%4==0):
    print (" Leap Year")
else:
    print ("Not a Leap Year")

```

Output:

Enter the year 2012

Leap Year

6. What is composition in functions?

The **value** returned by a **function** may be used as an **argument** for **another function** in a nested manner. This is called **composition**.

For example,

- ✓ using the **input()** function and apply **eval()** function to evaluate its value, for example:

```
>>> n1 = eval (input ("Enter a number: "))
      Enter a number: 234
>>> n1
      234
```

7. How recursive function works?

- ✓ A recursive function calls itself.
- ✓ It works **like loop**.
- ✓ The **condition** that is applied in any **recursive function** is known as **base condition**.
- ✓ If the base condition is **True** then the **program gives output** and **exits**.
- ✓ otherwise it will execute like an **infinite loop**.

Example:

```
def fact(n):
    if n == 0: # Base function
        return 1
    else:
        return n * fact(n-1) # calling itself
```

```
print(fact (5))
```

output: 120

8. What are the points to be noted while defining a function?

Syntax:

```
def function_name (parameter1, parameter2... ) :
```

Block of Statements

```
return (expression / None)
```

- ✓ Function begin with **def** keyword followed by **function name** and **parenthesis ()**.
- ✓ **parameters** or **arguments** should be placed within **parenthesis ()**.
- ✓ The code block is **indented** and always comes after a **colon (:)**.
- ✓ The statement "**return [expression]**" **exits** a function,

Example:

```
def hello():
    print("Hello")
    return
hello() ----- □ Calling statement
output : Hello
```


SECTION - C

1. Explain the different types of function with an example.

There are four types of functions in python, they are

- ✓ **User Defined** functions
- ✓ **Anonymous or Lambda** functions
- ✓ **Recursion** functions
- ✓ **Built-in** functions

User defined function

- ✓ Functions defined by the **users**.

Syntax:

def function_name (parameter1, parameter2...) :

Block of Statements

return (expression / None)

- ✓ Function begin with **def** keyword followed by **function name** and **parenthesis ()**.
- ✓ **parameters** or **arguments** should be placed within **parenthesis ()**.
- ✓ The code block is **indented** and always comes after a **colon (:)**.
- ✓ The statement **“return [expression]”** exits a function,

Example:

def hello():

print(“Hello”)

return

hello() ----- □ Calling statement

output : **Hello**

Anonymous or lambda function

- ✓ It is defined **without a name**.
- ✓ Instead of **def** keyword **lambda** keyword is used.
- ✓ It is also called as **lambda functions**.
- ✓ Lambda function **can take any number of arguments**
- ✓ Must **return one value** in the form of an **expression**.

Syntax:

lambda arg1,arg2,arg3,...argn : expression

Example:

s=lambda a,b,c : a+b+c

print(s(10,20,30))

Output : 60

- ✓ A recursive function calls itself.
- ✓ It works **like loop**.
- ✓ The **condition** that is applied in any **recursive function** is known as **base condition**.
- ✓ If the base condition is **True** then the **program gives output and exits**. otherwise it will execute like an **infinite loop**.

Example:

```
def fact(n):  
    if n == 0: # Base function  
        return 1  
    else:  
        return n * fact(n-1) # calling itself  
  
print(fact (5))  
  
output: 120
```

Built – in function

- ✓ Built – in Functions are inbuilt with in Python

Ex. `abs()`,`ord()`,`chr()`,`bin()`,`max()`,`min()`,`sum()` etc...

2. Explain the scope of variables with an example.

- ✓ Scope refers to the **accessibility of a variable**.
- ✓ There are **local** and **Global** scope

Local Scope

- ✓ A variable declared **inside** the function's body is known as local variable.

Rules of local variable

- ✓ It can be accessed **only within the function/block**.
- ✓ It is created **inside the function/block**.
- ✓ A local variable **only exists** while the function is **executing**.
- ✓ The **formal arguments** are also **local to function**.

Ex:

```
def loc():
    y=4
    loc()
    print(y)
```

Output: Name Error: name 'y' is not defined because y is a local variable.

Global Scope

- ✓ A global variable, can be used **anywhere in the program**.
- ✓ It is created **outside the function/block**.

Rules of global Keyword

- ✓ To define a variable **outside a function**, it's global by **default**.
- ✓ **global** keyword used to **modify** the global variable **inside** a function.

Example:

```
c = 10
```

```
def add():
```

```
    c=c+2
```

```
    print(c)
```

```
add()
```

output: local variable 'c' referenced before assignment

3.Explain the following built-in functions.

(a) id() (b) chr() (c) round() (d) type() (e) pow()

Function	Description	Syntax	Example
id ()	<ul style="list-style-type: none"> ✓ Returns address of an object 	id (object)	<pre>x=15 y='a' print ('address of x is :',id (x)) print ('address of y is :',id (y)) Output: address of x is : 1357486752 address of y is : 13480736</pre>
chr ()	<ul style="list-style-type: none"> ✓ Returns Unicode character for the given ASCII value 	chr (i)	<pre>c=65 print (chr (c)) Output:A</pre>
round ()	<ul style="list-style-type: none"> ✓ Returns the nearest integer to its input ✓ First argument is used to specify the value to be rounded ✓ Second argument is used to specify the number of decimal digits 	round(number [,ndigits])	<pre>x= 17.9 print ('x value is rounded to', round (x)) Output: X value is rounded to 18</pre>

type ()	✓ Returns the type of object.	type (object)	<i>x= 15.2</i> <i>print (type (x))</i> <i>Output:</i> <i><class 'float'></i>
pow ()	✓ Return the computation of ab i.e $a^{**}b$	pow (a,b)	<i>a= 5</i> <i>b= 2</i> <i>print (pow (a,b))</i> <i>Output:</i> <i>25</i>

4. Write a Python code to find the L.C.M. of two numbers.

```

x=int(input("Enter first number:"))
y=int(input("Enter second number:"))
if x>y:
    min=x
else:
    min=y
while(1):
    if((min%x == 0) and (min % y == 0)):
        print("LCM is:",min)
        break
    min=min+1

```

OUTPUT:

Enter first number:2

Enter second number:3

LCM is: 6

5. Explain recursive function with an example.

- ✓ A recursive function calls itself.
- ✓ It works like loop.
- ✓ The condition that is applied in any recursive function is known as base condition.
- ✓ If the base condition is True then the program gives output and exits.
- ✓ otherwise it will execute like an infinite loop.

Example:

```

def fact(n):
    if n == 0: # Base function
        return 1
    else:
        return n * fact(n-1) # calling itself
print(fact (5))
output: 120

```

8.STRINGS AND STRING MANIPULATION

SECTION - A

1. Define String. Or What is String?

- ✓ String is a **data type** in **python**.
- ✓ It is **array of characters**.
- ✓ String is a set of **Unicode characters** consists of **letters, numbers, or special symbols** .
- ✓ enclosed within **single, double** or even **triple** quotes.
- ✓ Strings are **immutable** in 'Python',
- ✓ It **can not be changed** during execution.

Ex. "ELANGO"

2.Do you modify a string in Python?

- ✓ Strings are **immutable** in 'Python',
- ✓ It **can not be changed** during execution
- ✓ If you want to modify the string, completely **overwrite** a new string value on the **existing** string variable.

a="RMK"

a="ELANGO"

3.How will you delete a string in python?

- ✓ We cannot delete a **particular character** in a string.
- ✓ We can **remove entire** string variable using **del** command.

Syntax: *del(string variable)*

Ex.>>>a="ELANGO"

>>>del(a)

4.What will be the output of the following python code?

str1="School" print(str1*3)

OUTPUT:

School School School

5.What is slicing?

- ✓ Slice is a **substring** of a main string.
- ✓ **Slicing operator [:]** with **index or subscript value** is used to take slice (substring).

Syntax: *variable[start:end]*

start - beginning index (default is 0)

end - last index value (n-1) n: length of string


```
>>> s="ELANGO"
>>> print (s[0]) #output : E
>>> print (s [1:5]) #output :LANG
>>> print (s[:5]) #output: ELANG
>>> print (s [3:]) #output: GO
```

SECTION – B

1. Write a Python program to display the given pattern

```
COMPUTER
COMPUTE
COMPUT
COMPU
COMP
COM
CO
C
```

```
s="COMPUTER"
i=len(s)
for x in s:
    print(s[ : i])
    i-=1
```

[OR]

```
str="COMPUTER"
index=len(str)
for i in str:
    print(str[:index])
    index-=1
```

2. Write a short about the followings with suitable example: (a) capitalize() (b) swapcase()

FUNCTION	PURPOSE	EXAMPLE
capitalize()	Used to capitalize the first character of the string	<pre>>>> city="chennai" >>> print(city.capitalize())</pre> <p>Output: Chennai</p>
swapcase()	It will change case of every character to its opposite case vice-versa.	<pre>>>> str1="tAmiL NaDu" >>> print(str1.swapcase())</pre> <p>Output: <i>TaMIL nAdU</i></p>

CODE:

```
str1 = "welcome"
str2 = "to school"
str3=str1[:2]+str2[len(str2)-2:]
print(str3)
```

OUTPUT:

Weol

4.What is the use of format()? Give an example.

- ✓ The **format()** function is used for **formatting strings**.
- ✓ The curly braces { } are used as **placeholders** or **replacement fields** which get replaced along with format() function

Ex.

a,b=10,5

```
print("The sum of {} and {} is {}".format(a, b, a+b))
```

#output: The sum of 10 and 5 is 15

5. Write a note about count() function in python.

- ✓ Returns the number of substrings occurs within the given string.
- ✓ Substring may be **a character**.
- ✓ Range is **optional**.
- ✓ Search is **case sensitive**.

Syntax: *Variable.count(string,start,end)*

```
>>> a="ELANGOVAN"
```

```
>>> print(a.count('E'))
```

1

```
>>> print(a.count('AN'))
```

2

```
>>> print(a.count('A',0,5))
```

1

SECTION – C

1.Explain about string operators in python with suitable example**(i) Concatenation (+)**

- ✓ **Joining of two or more** strings is called as **Concatenation**.
- ✓ The plus (+) operator is used .

Example

```
>>> "welcome" + "Python"
```

'welcomePython'

(ii) Append (+ =)

- ✓ **Adding a new** string with an existing string.
- ✓ The operator += is used.

Example:

```
>>>x="Welcome to "
```

```
>>>x+="Learn Python"
```

```
>>>print (x)
```

Welcome to Learn Python

(iii) Repeating (*)

- ✓ The multiplication operator (*) is used to display a string in **multiple number of times**.

Example

```
>>> str1="Welcome "
```

```
>>> print (str1*4)
```

Welcome WelcomeWelcomeWelcome

(iv) String slicing

- ✓ Slice is a **substring** of a main string.
- ✓ **Slicing operator [:]** with **index or subscript value** is used to take slice (substring).

Syntax:

variable[start:end]

start - beginning index(default is 0)

end -last index value (n-1) n: length of string

```
>>> s="ELANGO"
```

```
>>> print (s[0]) #output : E
```

```
>>> print (s [1:5]) #output :LANG
```

```
>>> print (s[:5]) #output: ELANG
```

```
>>> print (s [3:]) #output: GO
```

(v) Stride when slicing string

- ✓ In the slicing operation, a third argument is **stride**,
- ✓ which refers to the **number of characters** to move forward **after the first character** is retrieved from the string.
- ✓ The default value of stride is **1**.

Example

```
>>> s = "Welcome to learn Python"
```

```
>>> print (s [10:16])
```

learn

```
>>> print (s [10:16:4])
```

r

```
>>> print (s [10:16:2])
```

er

```
>>> print (s [::-3])
```

Wceoenyo

- ✓ *If you specify a negative value, it prints in reverse order*

```
>>> a="ELANGO"
```

```
>>> print(a[::-1])
```

OGNALE

9.LISTS, TUPLES, SETS, AND DICTIONARY

SECTION – A

1.What is List in python?

- ✓ A list in Python is known as a “**sequence data type**” like strings.
- ✓ It is an **ordered** collection of values
- ✓ Enclosed within **square brackets []**.
- ✓ Each value of a list is called as **element** separated by ,(comma).
- ✓ It consists of **numbers, characters, strings** and even the **nested lists** as well.
- ✓ The elements can be **modified** or **mutable**.
- ✓ lists are similar to **arrays**

ex. m=[10,25.5,”rmk”,[4,5,6]]

2. How will you access list element in python?

How will you access the list elements in reverse order

- ✓ **Index value** is used to access an element in a list
- ✓ Index value is an integer number which can be **positive** or **negative**.
- ✓ **Positive** value of index counts from the beginning of the list(**0 to n-1**)
- ✓ **Negative** value counts **backward** from end of the list (i.e. in **reverse order**) (**-n to -1**)
- ✓ A **negative index** can be used to access an element in reverse order.
- ✓ This is called as **Reverse Indexing**.

Syntax:

*List_Var= [e1,e2,e3,.....en]
print(list_var[index])*

Example:

```
>>>m=[10,20,30,40]
>>>print(m[2])
30
>>>print(m[-1]) # reverse order
40
```

3.What will be the value of x in following python code?

```
List1=[2,4,6,[1,3,5]]
x=len(List1) print(x)
```

OUTPUT:

4

4. Differentiate del with remove() function of List.

1)del statement

- ✓ It is used to delete elements if the **index value** is **known** .
- ✓ Used to delete **particular** element ,**multiple** elements and **entire** list

Syntax:

- ✓ To delete particular element in a list **del list_var[index of an element]**

Ex. **del m[2]**

- ✓ To delete multiple elements in a list **del list_var[from:to]**

Ex. **del m[1:3]**

- ✓ To delete entire list **del list_var**

ex. **del m**

2) remove() function

- ✓ It is used to delete a element if the **index value** is **not known**.
- ✓ Used to **delete** only elements **not list**.

Syntax: List_var.remove(element)

Ex. m=[10,20,30,40,50]

m.remove(30)

>>> print(a)

output: [10,20,40,50]

5. Write the syntax of creating a Tuple with n number of elements.

To create empty tuple **Tuple_Var =()**

- ✓ To create tuple with n number elements **Tuple_Var =(E1,E2,E3.....En)**
- ✓ To create tuple with n number elements **without** parenthesis **Tuple_Var = E1,E2,E3.....En**
- ✓ The tuple() function is used to create tuples from a list.

Syntax: tuplename=tuple([list element])

Ex.

>>>a=tuple([1,2,3])

6. What is set in Python?

- ✓ In python, a set is another type of collection data type.
- ✓ A Set is a mutable and an unordered collection of elements without duplicates or repeated element.
- ✓ This feature used to include membership testing and eliminating duplicate elements.

SECTION – B

1.What are the difference between list and Tuples?

list	Tuples
✓ List Tuple Elements are enclosed by square brackets []	✓ Elements are enclosed by parenthesis () .
✓ Elements are changeable	✓ Elements are unchangeable
✓ Slower than tuples	✓ Faster then list
Ex[1,2,3]	Ex.(1,2,3)

2. Write a short note about sort() in python.

sort()

- ✓ Sort the elements in a list
- ✓ It will **affect** the **original**
- ✓ It has two arguments, they are **reverse** and **key**.
- ✓ If **reverse** is True, list sorting is in **descending** order
- ✓ **Key** specify the **name** of **user defined function**
- ✓ Both are **optional**
- ✓ Ascending is **default**

Syntax: list_var.sort([reverse=True],[key=function])

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

```
list=[2**x for x in range(5)]
```

```
print(list)
```

OUTPUT:

```
[1, 2, 4, 8, 16]
```

4.Explain the difference between del() and clear() in dictionary with an example

- ✓ **del** statement used to **delete a particular element**.
- ✓ **del dictionary_name[key]**

```
d={'name ':'ELANGO','age':25}
```

```
del d['age']
```

```
print(d)
```

output: {'name: "ELANGO"}

To delete an entire dictionary

```
del dictionary_name
```

```
d={'name ':'ELANGO','age':25}
```

```
del d
```

```
print(d)
```

output: NameError:name 'd' is not defined

✓ **Clear()** is used to delete all the elements in a dictionary.

```
dictionary_name.clear( )
```

```
d={'name ':'ELANGO','age':25}
```

```
d.clear()
```

```
print(d)
```

output: { }

List	Dictionary
✓ It is an ordered set of elements	✓ It is a data structure .
✓ Index values used to access a particular element	✓ used for matching one element (Key) with another (Value).
✓ Enclosed with []	✓ Key ,value used to access a particular element.
	✓ Enclosed with { }

SECTION – C

1.What are the different ways to insert an element in a list. Explain with suitable example.

Or How to add elements in list? Differentiate between append() and extend() append()

✓ **append()** function is used to add a **single element** to the list as **last element**.

Syntax: list_var.append(element)

ex.

```
m=[10,20,30,40]
```

```
m.append(50)
```

```
print(a)
```

```
[10,20,30,40,50]
```

extend()

✓ **extend()** function is used to add **more than one element** to the list as **last elements**.

✓ **Syntax: list_var.extend([element])**

ex.

```
m=[10,20,30,40]
m.extend([50,60])
print(a)
[10,20,30,40,50,60]
```

insert()

✓ The insert() function is used to insert an element at **any position** of a list.

Syntax: **list_var.insert(index position ,element)**

ex.

```
m=[10,20,30,40]
m.insert(2,50)
print(a)
[10,20,50,30,40]
```

2.What is the purpose of range()? Explain with an example.

✓ In Python, for loop uses the **range()** function to generate the **sequence of value**.

Syntax of range()

range (start,stop,step)

Where,

- ✓ start – refers to the **initial** value (**0** is default)
- ✓ stop – refers to the **final** value (stop-1)
- ✓ step – refers to **increment** value(1 is default)

start and step are **optional** part.

Example :

```
for x in range(2,11,2) :
    print(x,end = ' ')
Output: 2 4 6 8 10
```

3.What is nested tuple? Explain with an example

- ✓ A tuple can be defined **inside** another tuple; called Nested tuple.
- ✓ **Each tuple** is considered as an **element**.
- ✓ The **for loop** will be useful to access all the elements in a nested tuple.

```
T = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H",
97.5), ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8))
for i in T:
    print(i)
```

Output:

('Vinodini', 'XII-F', 98.7)

('Soundarya', 'XII-H', 97.5)

('Tharani', 'XII-F', 95.3)

('Saisri', 'XII-G', 93.8)

4.Explain the different set operations supported by python with suitable example.

List out the set operations supported by python

- ✓ Python supports **four** types of **set operations**. They are , 1.Union, 2.Intersection 3.difference
4.Symmetric difference

1.Union

- ✓ It **includes all** elements **from two or more** sets
- ✓ **|**-operator or **union()** function is used to **join** two sets in python.

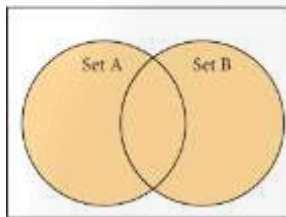
a={1,2,3}

b={4,5}

c=a|b

print(c)

{1,2,3,4,5}



2.Intersection

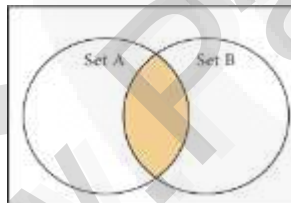
- ✓ It includes the **common** elements in two sets
- ✓ **&**-operator or **intersection()** function is used to **intersect** two sets in python

a={1,2,3}

b={4,3}

print(a & b)

{3}



3.Difference

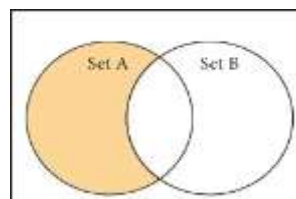
- ✓ It **includes all** elements that are in **first** set but **not in** the **second** set .
- ✓ **-**operator or **difference()** function is used to difference operation in python

a={1,2,3}

b={4,3}

print(a-b)

{1,2}



4.Symmetric difference

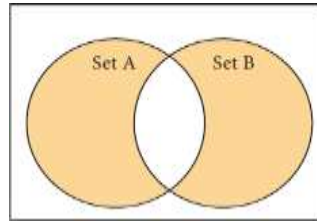
- ✓ It **includes all** the elements that are in two sets but **not the one** that are **common to two sets**.
- ✓ **^** operator or **symmetric_difference()** function is used to symmetric difference set operation in python

```
a={1,2,3}
```

```
b={4,3}
```

```
print(a^b)
```

```
{1,2,4}
```



10.PYTHON CLASSES AND OBJECTS

SECTION – A

1.What is class. How to define class in python

- ✓ Class is the **main building block** in Python
- ✓ Class is a **template** for the object.
- ✓ A class is a way of binding **Class variable** and **functions** together.

Defining classes

- ✓ In Python, a class is defined by using the keyword **class**.
- ✓ Every class has a **unique name** followed by a **colon (:)**.

Syntax:

class class_name:

statements

Example:

class Sample:

x, y = 10, 20

2. What is instantiation ? How to create object or instance in class?

- ✓ The **process of creating object** is called as “Class Instantiation.

Syntax: object_name = class_name()

Ex.

Class rmk:

.....

.....

a=rmk() # a is an object of the class

3.What is the output of the following program.?

class Sample:

__num=10

def disp(self):

print(self.__num)

S=Sample()

S.disp()

print(S.__num)

output: 10

error

2. Write a class with two private class variables and print the sum using a method.*class add:**s=0 # public variable**def __init__(self,a,b):**self.__a=a**self.__b=b # a,b, - private variables**def display(self):**s=self.__a+self.__b**return s**x=add(10,5)**print(x.display())**output : 15***3. Find the error in the following program to get the given output?***class Fruits:**def __init__(self, f1, f2):**self.f1=f1**self.f2=f2**def display(self):**print("Fruit 1 = %s, Fruit 2 = %s"%(self.f1, self.f2))**F = Fruits ('Apple', 'Mango')**del F.display**F.display()**Output**Fruit 1 = Apple, Fruit 2 = Mango***Correction:**✓ The statement *del F.display* Should not be used to display the output**4. What is the output of the following program?***class Greeting:**def __init__(self, name):**self.__name = name**def display(self):**print("Good Morning ", self.__name)**obj=Greeting('Bindu Madhavan')**obj.display()***output:** *Good Morning Bindu Madhavan*

5.How will you create constructor and Destructor in Python?

- ✓ In Python, “**init**” function is used as a **Constructor**.
- ✓ It must **begin** and **end** with **double underscore** (**__**).
- ✓ It is **automatically executed** when an object of a class is **created**
- ✓ This constructor function can be defined **with** or **without** arguments.
- ✓ Constructor is used to **initialize** the class variables

Syntax:

```
def __init__(self,[args]):
    statements
```

example:

```
class add:
    def __init__(self):
        a=10
```

Destructor:

- ✓ In Python, **del** function is used as **destructor**.
- ✓ It must **begin** and **end** with **double under score** (**__**).
- ✓ It is **automatically executed** when an object **exit** from the scope.
- ✓ Destructor **removes** the memory of an object

Syntax:

```
def __del__(self):
    statements
```

SECTION – C

1.How will you create constructor and Destructor in Python?

- ✓ In Python, “**init**” function is used as a **Constructor**.
- ✓ It must **begin** and **end** with **double underscore**(**__**).
- ✓ It is **automatically executed** when an object of a class is **created**
- ✓ This constructor function can be defined **with** or **without** arguments.
- ✓ Constructor is used to **initialize** the class variables

Syntax:

```
def __init__(self,[args]):
    statements
```

example:

```
class add:
    def __init__(self):
        a=10
```

Destructor:

- ✓ In Python, **del** function is used as **destructor**.
- ✓ It must **begin** and **end** with **double underscore**(**__**).
- ✓ It is **automatically executed** when an object **exit** from the scope.
- ✓ Destructor **removes** the memory of an object

Syntax:

```
def __del__(self):
```

```
statements
```

11.DATABASE CONCEPTS

SECTION – A

1.Mention few examples of a database.

DBMS:

- ✓ Fox pro
- ✓ dbase.
- ✓ IBM DB2.
- ✓ Microsoft Access.
- ✓ Microsoft Excel.
- ✓ MySQL.

2.List some examples of RDBMS

RDBMS :

- ✓ SQL Server
- ✓ Oracle
- ✓ MySQL
- ✓ Maria DB
- ✓ SQLite

3.What is data consistency?

- ✓ Data Consistency means that data values are the **same at** all instances of a database

4.What is the difference between Hierarchical and Network data model?

Hierarchical Model

- ✓ It was developed by **IBM**(Information Management System).
- ✓ Data is represented as a simple **tree like structure** form.
- ✓ Relationship is **one-to many** ie **parent-child** relationship.
- ✓ **One child** can have only **one parent** but **one parent** can have **many children**.
- ✓ Used in **IBM Main Frame computers**.

Network Database Model

- ✓ It is similar to the **hierarchical** data model.
- ✓ Relationship is **many-to many** relationships.
- ✓ A **child** may have **many parent** nodes.
- ✓ This model is easier and faster to access the data.

5..What is normalization?

- ✓ Normalization is an integral part of RDBMS in order to **reduce data redundancy** and **improve data integrity**.

SECTION – B

1.What is the difference between Select and Project command

Select Command	Project Command
<ul style="list-style-type: none"> The SELECT operation is used for selecting a subset with tuples according to a given Condition C. 	<ul style="list-style-type: none"> The projection method defines a relation that contains a vertical subset of Relation.
<ul style="list-style-type: none"> Select filters out all tuples that do not satisfy C. 	<ul style="list-style-type: none"> The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Symbol : σ	Symbol : Π
<u>General Form:</u> $\sigma (R)$ <u>Example:</u> $\sigma = \text{"Big Data"} (STUDENT)$ course	<u>Example:</u> $\Pi (STUDENT)$ course

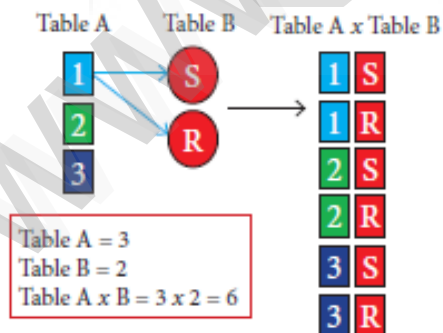
2.What is the role of DBA?

Database Administrators (DBA)

- ✓ DBA manages the complete database management system.
- ✓ DBA managing the license keys, user accounts ,security and access etc.

3. Explain Cartesian Product with a suitable example.

- ✓ Cross product is a way of combining two relations.
- ✓ $A \times B$ means A times B,
- ✓ It is used to merge columns from two relations



4.Explain Object Model with example.

Object Model

Object model stores the data in the form of **objects**, **attributes** and **methods**, **classes** and

Inheritance.

- ✓ This model handles more complex applications, such as **Geographic information System(GIS)**, **scientific experiments**, **engineering design** and **manufacturing**.
- ✓ It is used in **file Management System**.
- ✓ It provides a clear **modular structure**.
- ✓ It is **easy to maintain** and **modify** the existing code.

5.Write a note on different types of DBMS users.

Database Administrators

Database Administrator or DBA is the one who manages the complete database management system.

Application Programmers or Software Developers

This user group is involved in developing and designing the parts of DBMS.

End User

End users are the one who store, retrieve, update and delete data.

Database designers:

They are responsible for identifying the data to be stored in the database for choosing appropriate structures to represent and store the data.

SECTION – C

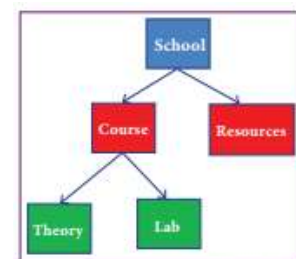
1.Explain the Types of Data Model

A data model describes **how** the **data** can be **represented** and **accessed** from a software after complete **implementation**

1. Hierarchical Model
2. Relational Model
3. Network Database Model
4. Entity Relationship Model
5. Object Model

Hierarchical Model

- ✓ It was developed by **IBM**(Information Management System).
- ✓ Data is represented as a simple **tree like structure** form.
- ✓ Relationship is **one-to many** ie **parent-child** relationship.
- ✓ **One child** can have only **one parent** but **one parent** can have **many children**.



Hierarchical Model Fig. 11.3

- ✓ Used in **IBM Main Frame computers**.

Relational Model

- ✓ It was developed by **E.F. Codd**.
- ✓ Data is represented as **tables (relations)**.
- ✓ **Rows** in the table stored all the information related to a **particular type**.
- ✓ A **relation key** is an attribute which **uniquely** identifies a **particular tuple (record)**

Stu_id	Name	Age
1	Malar	17
2	Sarav	16
3	Vidu	16

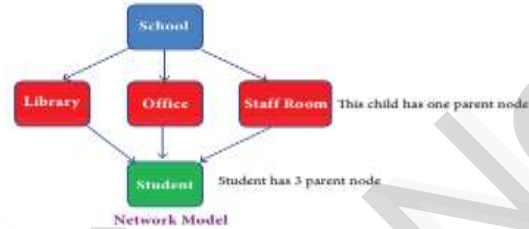
Tea_id	Name	Teacher
1	C++	Karan
2	Php	Ramakrishnan
3	Python	Vidya

Stu_id	Tea_id	Mark
1	1	92
1	2	89
1	3	96

Relational Model

Network Database Model

- ✓ It is similar to the **hierarchical** data model.
- ✓ Relationship is **many-to many** relationships.
- ✓ A **child** may **have many parent** nodes.
- ✓ This model is easier and faster to access the data.



Entity Relationship Model. (ER model)

- ✓ It was developed by **Chen**.
- ✓ It is **conceptual design** for the database.
- ✓ The developer can **easily understand** the system by looking at ER model.



Rectangle represents the **entities**. E.g. Doctor and Patient

Ellipse represents the **attributes** E.g. D-id, D-name, P-id, P-name.

Diamond represents the **relationship** in ER diagrams

Object Model

- ✓ Object model stores the data in the form of **objects, attributes and methods, classes and Inheritance**.
- ✓ This model handles more complex applications, such as **Geographic information System (GIS), scientific experiments, engineering design and manufacturing**.
- ✓ It is used in **file Management System**.
- ✓ It provides a clear **modular structure**.
- ✓ It is **easy to maintain and modify** the existing code.

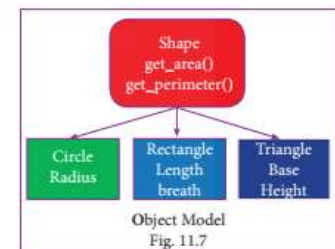
Example

Shape, Circle, Rectangle and Triangle --**objects**.

Circle has the attribute **radius**.

Rectangle has the attributes **length** and **breadth**.

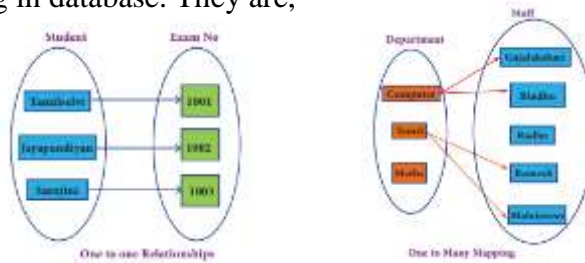
Triangle has the attributes **base** and **height**. The objects Circle, Rectangle and Triangle **inherit** from the object Shape.



2.Explain the different types of relationship mapping.

There are four types of relationship mapping in database. They are,

1. **One-to-One** Relationship
2. **One-to-Many** Relationship
3. **Many-to-One** Relationship
4. **Many-to-Many** Relationship



One-to-One Relationship

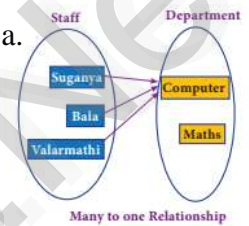
- ✓ **One entity** is related with **only one** other entity.
- ✓ **One row** in a table is linked with **only one row** in another table and vice versa.

For example: A student can have only one exam number

One-to-Many Relationship

- ✓ One entity is related to **many other** entities.
- ✓ **One row** in a table **A** is linked to **many rows** in a table **B**, but **one row** in a table **B** is linked to **only one row** in table **A**.

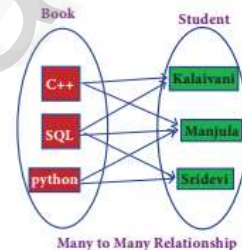
For example: One Department has many staff



Many-to-One Relationship

- ✓ **Many** entities can be related with **only one** in the other entity.

For example: A number of staff members working in one Department.



Many-to-Many Relationship

- ✓ Multiple records in a table are associated with multiple records in **another table**.

Example: Many Books in a Library are issued to many students

3.Differentiate between DBMS and RDBMS

Basis of Comparison	DBMS	RDBMS
Expansion	Database Management System	Relational Data Base Management System
Data storage	Navigational model ie data by linked records	Relational model (in tables). ie data in tables as row and column
Data redundancy	Present	Not Present
Normalization	Not performed	RDBMS uses normalization to reduce redundancy
Data access	Consumes more time	Faster, compared to DBMS.

Keys and indexes	Does not use.	used to establish relationship. Keys are used in RDBMS.
Transaction management	Inefficient, Error prone and insecure.	Efficient and secure.
Distributed Databases	Not supported	Supported by RDBMS.
Example	Dbase, FoxPro.	SQL server, Oracle, mysql, Maria DB, SQLite, MS Access.

4.Explain the different operators in Relational algebra with suitable examples.

- Relational Algebra is used for modeling data stored in relational databases and for defining queries on it.
- Relational Algebra is divided into various groups.

1) Unary Relational Operations

- SELECT (symbol : σ)
- PROJECT (symbol : Π)

2) Relational Algebra Operations from Set Theory

- UNION (\cup)
- INTERSECTION (\cap)
- DIFFERENCE ($-$)
- CARTESIAN PRODUCT (\times)

SELECT (symbol : σ)

General form $\sigma_c (R)$

R – relation c - condition

- ✓ The SELECT operation is used for selecting a subset with tuples according to a given condition.
- ✓ Select filters out all tuples that do not satisfy C.

Studno	Name	Course	Year
cs1	Kannan	Big Data	II
cs2	Gowri Shankar	R language	I
cs3	Lenin	Big Data	I
cs4	Padmaja	Python Programming	I

Example: $\sigma_{\text{course}} = \text{"Big Data"} (\text{STUDENT})$

Studno	Name	Course	Year
cs1	Kannan	Big Data	II
cs3	Lenin	Big Data	I

PROJECT (symbol : Π)

- ✓ The projection eliminates all attributes except those mentioned in the projection list.
- ✓ It contains a **vertical** subset of Relation.
- ✓ **Example:** $\Pi_{\text{course}} (\text{STUDENT})$

Course
Big Data
R language
Python Programming

UNION (Symbol : \cup) A \cup B

- ✓ It includes **all tuples** that are in tables **A** or in **B**.
- ✓ It also **eliminates duplicates**.
- ✓ **Expressed as A \cup B**

Table A		Table B	
Studno	Name	Studno	Name
cs1	Kannan	cs1	Kannan
cs3	Lenin	cs2	GowriShankar
cs4	Padmaja	cs3	Lenin

RESULT:

Table A \cup B	
Studno	Name
cs1	Kannan
cs2	GowriShankar
cs3	Lenin
cs4	Padmaja

SET DIFFERENCE (Symbol : $-$)

- ✓ It includes **all tuples** that are **in A** but **not in B**.
- ✓ The attribute name of A has to match with the attribute name in B.
- ✓ Example: using table A and B

Table A - B	
cs4	Padmaja

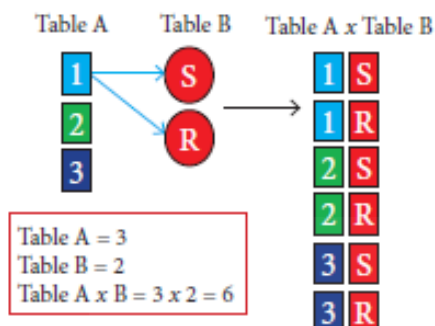
INTERSECTION (symbol : \cap) A \cap B

- ✓ Defines a relation consisting of a set of all tuple that are in both in A and B.
- ✓ However, A and B must be union-compatible.

A \cap B	
cs1	Kannan
cs3	Lenin

PRODUCT OR CARTESIAN PRODUCT (Symbol : X)

- ✓ Cross product is a way of combining two relations.
- ✓ **A x B means A times B,**
- ✓ It is used to merge columns from two relations

**5.Explain the characteristics of RDBMS.**

1. Data Stored in a Tables	<ul style="list-style-type: none"> • Data is stored into tables, created inside the database. • DBMS also allows to have relationship between tables.
2. Reduced Redundancy	<ul style="list-style-type: none"> • Unnecessary repetition of data in database was a big problem. • DBMS follows Normalisation which divides the data in such a way that repetition is minimum.
3.Data Consistency	<ul style="list-style-type: none"> • Data Consistency means that data values are the same at all instances of a database.
4.Support Multiple user and Concurrent Access	<ul style="list-style-type: none"> • DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
5.Query Language	<ul style="list-style-type: none"> • DBMS provides users with a simple query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. Security	<ul style="list-style-type: none"> • The DBMS also takes care of the security of data, protecting the data from unauthorized access. • Creating user accounts with different access permissions we can easily secure our data.
7.DBMS Supports Transactions	<ul style="list-style-type: none"> • It allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

12.STRUCTURED QUERY LANGUAGE

SECTION – A

1. Write a query that selects all students whose age is less than 18 in order wise.

Query:

SELECT * FROM Student WHERE Age<=18 ORDER BY Name;

2. Differentiate Unique and Primary Key constraint.

Unique Key Constraint	Primary Key Constraint
<ul style="list-style-type: none"> This constraint ensures that no two rows have the same value in the specified columns. 	<ul style="list-style-type: none"> This constraint declares a field as a Primary key which helps to uniquely identify a record.
<ul style="list-style-type: none"> The UNIQUE constraint can be applied only to fields that have also been declared as NOT NULL. 	<ul style="list-style-type: none"> The primary key does not allow NULL values and therefore a primary key field must have the NOT NULL constraint.

3. Write the difference between table constraint and column constraint?

Table Constraint	Column Constraint
<ul style="list-style-type: none"> Table constraints apply to a group of one or more columns. 	<ul style="list-style-type: none"> Column constraints apply only to individual column.

4. Which component of SQL lets insert values in tables and which lets to create a table?

- ✓ Insert values in tables: DML – *Data Manipulation Language*.
- ✓ Create a table: DDL – *Data Definition language*.

5. What is the difference between SQL and MySQL?

SQL	MySQL
<ul style="list-style-type: none"> Structured Query Language is a language used for accessing databases. 	<ul style="list-style-type: none"> MySQL is a database management system, like SQL Server, Oracle, Informix, Postgres, etc.
<ul style="list-style-type: none"> ✓ SQL is a DBMS 	<ul style="list-style-type: none"> ✓ MySQL is a RDBMS.

SECTION – B

1.What is a constraint? Write short note on Primary key constraint.

- ✓ Constraint is a condition applicable on a field or set of fields.
- ✓ Primary constraint declares a field as a Primary key which helps to uniquely identify a record.
- ✓ It is similar to unique constraint except that only one field of a table can be set as primary key.
- ✓ The primary key does not allow **NULL** values and therefore a primary key field must have the **NOT NULL** constraint.

2.Write a SQL statement to modify the student table structure by adding a new field.

Syntax :

ALTER TABLE <table-name> ADD <column-name><data type><size>;

- ✓ To add a new column “Address” of type „char“ to the Student table, the command is used as

Statement:

ALTER TABLE Student ADD Address char;

3.Write any three DDL commands. Data Definition Language:

Create Command: To create tables in the database.

CREATE TABLE Student (Admno integer, Name char(20), Gender char(1), Age integer);

Alter Command: Alters the structure of the database.

ALTER TABLE Student ADD Address char;

Drop Command: Delete tables from database.

DROP TABLE Student;

4.Write the use of Savepoint command with an example.

- ✓ The SAVEPOINT command is used to temporarily save a transaction so that you can roll back to the point whenever required.

Syntax: SAVEPOINT savepoint_name;

Example: SAVEPOINT A;

5.Write a SQL statement using DISTINCT keyword.

- ✓ The **DISTINCT** keyword is used along with the **SELECT** command to eliminate duplicate rows in the table.
- ✓ This helps to eliminate redundant data.
- ✓ ***For Example: SELECT DISTINCT Place FROM Student;***

SECTION – C

1.What is a constraint? What are the types of constraint? Explain.

Constraint is a **condition** applicable on a field or set of fields.

- ✓ Tables can be created with constraints.

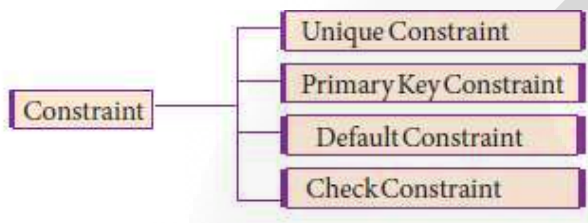
There are two types of constraints .they are.

Column Constraint

- ✓ Constraint is applied to **individual** field.
- ✓ It is given in **any field** of the table

Table Constraint

- ✓ It is applied to a **group of field**.
- ✓ It is given at the **end of the table** definition.



Types:

(i) Unique Constraint

- ✓ The **UNIQUE** constraint do not allow **duplicate** entries.
- ✓ It is **NOT NULL** constraint.

Example:

```

CREATE TABLE Student
(Admno integer NOT NULL UNIQUE,
Name char (20) NOT NULL,
Age integer, );
  
```

Here **Admno** field do not allow **duplicate** entries and cannot be **NULL**.

ii)Primary Key Constraint

- ✓ It helps to **uniquely identify** a record.
- ✓ Only **one field** of a table can be **set as primary key**.
- ✓ It is **NOT NULL** constraint.

Example :

```

CREATE TABLE Student
(Admno integer NOT NULL PRIMARY KEY,
Name char (20) NOT NULL,
Age integer );
  
```

(iii) DEFAULT Constraint

- ✓ It is used to assign a **default value** for the field.
- ✓ When **no value** is given for the specified field **automatically** the default value will be **assigned** to the field.

Example:

```
CREATE TABLE Student
(Admno integer NOT NULL,
Name char (20) NOT NULL,
Age integer DEFAULT = "17");
```

- ✓ In the above example, the "Age" field is assigned a default value of 17, therefore when no value is entered in age by the user, it automatically assigns 17 to Age.

(iv) Check Constraint

- ✓ It helps to set a **limit value** placed for a field.
- ✓ It allows only the **restricted values** on that field.
- ✓ It may use **relational and logical operators** for **condition**.

Example :

```
CREATE TABLE Student
(Admno integer NOT NULL,
Name char (20) NOT NULL,
Age integer (CHECK<=19));
```

TABLE CONSTRAINT

- ✓ apply to more than one columns
- ✓ It is normally **given** at the **end of the table** definition.

Example:

```
CREATE TABLE Student
(Admno integer NOT NULL,
Name char (20) NOT NULL,
Age integer,
PRIMARY KEY (Admno,Name) );
```

2. Consider the following employee table. Write SQL commands for the qtns.(i) to (v).

EMP CODE	NAME	DESIG	PAY	ALLO WANCE
S1001	Hariharan	Supervisor	29000	12000
P1002	Shaji	Operator	10000	5500
P1003	Prasad	Operator	12000	6500
C1004	Manjima	Clerk	8000	4500
M1005	Ratheesh	Mechanic	20000	7000

- ✓ To display the details of all employees in descending order of pay.

*SELECT * FROM employee ORDER BY DESC;*

- ✓ To display all employees whose allowance is between 5000 and 7000.

*SELECT * FROM employee WHERE allowance BETWEEN 5000 AND 7000;*

- ✓ To remove the employees who are mechanic.

DELETE FROM employee WHERE design="Mechanic";

- ✓ To add a new row.

INSERT INTO employee

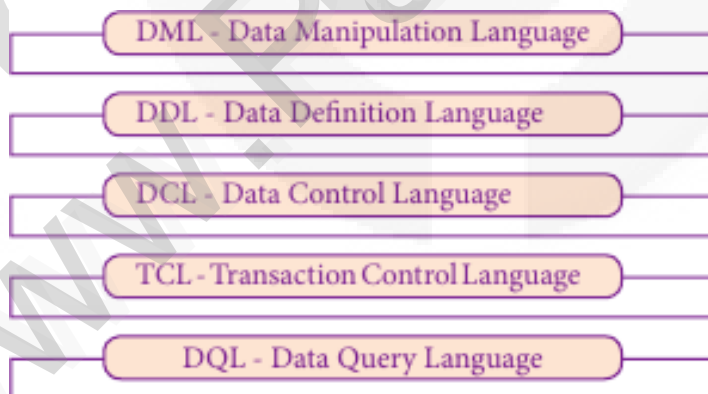
(empcode,name,design,pay,allowance)VALUES(S1002,Baskaran,Supervisor, 29000,12000);

- ✓ To display the details of all employees who are operators.

*SELECT * FROM employee WHERE design="Operator";*

3. What are the components of SQL? Write the commands in each.

Components of SQL are:

**D) DATA MANIPULATION LANGUAGE(DML)**

A Data Manipulation Language (DML) is a computer programming language used for adding (inserting), removing (deleting), modifying (updating), retrieving data in a database.

Two types of DML

- ✓ **Procedural DML** –to specify what data is needed.
- ✓ **Non-Procedural DML**-without specify what data is needed .

DML commands :

Insert: Inserts data into a table

Update : Updates the existing data within a table

Delete: Deletes all records from a table, but not the field.

II) DATA DEFINITION LANGUAGE (DDL)

- ✓ It used to **define, create and modify** the **structure** of **database objects**.

Functions of DDL:

- ✓ It **gives** a unique **name** to each data item type, record type, file type and database.
- ✓ Used to define the **size** of the data item and the **range** of values.
- ✓ It gives **privacy locks**.

DDL commands

Create : To create **tables** in the database

Alter : Alters the **structure** of the database

Drop : Delete **tables** from database

Truncate : Remove all **records** from a table

III) DATA CONTROL LANGUAGE(DCL)

- ✓ A Data Control Language (DCL) is used to control the **access of data** stored in a database.

DCL commands:

Grant :Grants **permission** to one or more users to perform specific tasks

Revoke : **Withdraws** the access **permission** given by the GRANT statement

IV) TRANSACTIONAL CONTROL LANGUAGE(TCL)

- ✓ Transactional control language (TCL)is used to manage transactions in the database.

TCL commands:

Commit: Saves any transaction into the database **permanently**

Rollback: **Restores** the database to last **commit** state

Save point: Saves any transaction into the database **temporarily** .

V)DATA QUERY LANGUAGE(DQL)

- ✓ The Data Query Language is used to query or retrieve data from a database.

DQL command:

Select: It displays the records from the table

4. Construct the following SQL statements in the student table-

- (i) SELECT statement using GROUP BY clause.
- (ii) SELECT statement using ORDER BY clause.

1.SELECT statement using GROUP BY clause.

SELECT Gender FROM Student GROUP BY Gender;

Output:

Gender
Male
Female

SELECT Gender, count(*) FROM Student GROUP BY male;

Output:

Gender	Count(*)
Male	5
Female	3

2.SELECT statement using ORDER BY clause.

SELECT * FROM student WHERE Age>=18 ORDER BY Name DESC;

Output:

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai

5. Write a SQL statement to create a table for employee having any five fields and create a table constraint for the employee table.

CREATE TABLE employee

(Admno integer NOT NULL ,

Name char (20) NOT NULL,

ecode integer NOT NULL,

Desig char(10),

Salary(CHECK <=20000),

PRIMARY KEY(Admno,ecode));

13.PYTHON AND CSV FILES

SECTION – A

1. What is CSV File?

- ✓ CSV - Comma Separated Values.
- ✓ CSV format is a **plain text format** with a series of values separated by commas.
- ✓ CSV can be **opened** with any text editor.
- ✓ extension is **.csv**
- ✓ CSV files can be much faster, and it also consumes less memory

2.Mention the two ways to read a CSV file using Python.

There are two ways to read a CSV file.

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

3.Mention the default modes of the File.

- ✓ The default is reading (,r") in text mode.
- ✓ In this mode, while reading from the file the data would be in the format of **strings**.

4. What is use of next() function?

- ✓ **next()** – used to **skip the first row** to sort a csv file.
- ✓ **Example:** While sorting the row heading is also get sorted, to avoid that the first is skipped using next().

5.How will you sort more than one column from a csv file? Give an example statement.

- ✓ To sort by more than one column you can use **itemgetter** with multiple indices.

Syntax: *operator.itemgetter(col_no)*

Example: *sortedlist = sorted (data, key=operator.itemgetter(1))*

SECTION – B

1.Write a note on open() function of python. What is the difference between the two methods?

- ✓ **open()** function used to **open a file**.
- ✓ This function returns a **file object**, also called a **handle**, as
- ✓ it is used to **read** or **modify** the **file** .

Example:

```
>>>f=open("sample.csv")
>>> with open("test.txt",'r') as f: (f – file object)
```

2. Write a Python program to modify an existing file.

- ✓ Making **changes** in the data of **existing file** or add more data is called **modification**

```
import csv
a=['3','Meena','chennai']
with open('ela.csv','r') as f:
    r=csv.reader(f)
    x=list(r)
    x[3]=a
with open('els.csv','w') as wf:
    write=csv.writer(wf)
    write.writerows(x)
f.close()
cw.close
```

In the above program,

- ✓ The third row of “ela.csv” is modified and saved.
- ✓ First the “ela.csv” file is read by using csv.reader() function.
- ✓ list() stores each row of the file.
- ✓ The statement “x[3] = a”, changed the third row of the file with the new content in “a”.

3. How to read csv file with default delimiter comma from python?

```
import csv
with open('D://ela.csv','r') as f:
    read = csv.reader(f)
    for x in read:
        print(x)
f.close()
```

4. What is the difference between the write mode and append mode.

Write mode: ‘w’

- ✓ It writes the data from the **beginning** of the **new file**.
- ✓ It **overwrites** if the file **already exists**.

Append Mode: ‘r’

- ✓ It adds the data at the **end** of the **existing file**

5. What is the difference between reader() and DictReader() function?

csv module’s reader function (csv.reader()):

- ✓ **csv.reader()** - to read CSV file into **list/tuple**..
- ✓ It will take each line of the file and make a list of all columns.
- ✓ It can **read data** from **csv files** of different formats like quotes (" "), pipe (|) and comma (,).

syntax :

csv.reader(fileobject,delimiter,fmtparams)

file object :- contain path and mode of the file

delimiter :- an optional , (,) is default .

fmtparams : optional ,It is used to override the default values like skipinitialspace, quoting etc.

DictReader class(DictReader()):

- ✓ **DictReader()** - To read a CSV file into a **dictionary**
- ✓ It works **similar** to the **reader()** class .
- ✓ It creates an object which maps **data to a dictionary**.
- ✓ The keys are given by the **fieldnames** as **parameter**.

SECTION – C

1.Differentiate Excel file and CSV file.

Excel	CSV
<ul style="list-style-type: none"> Excel is a binary file that holds information about all the worksheets in a file, including both content and formatting. 	<ul style="list-style-type: none"> CSV format is a plain text format with a series of values separated by commas.
<ul style="list-style-type: none"> XLS files can only be read by applications that have been especially written to read their format, and can only be written in the same way. 	<ul style="list-style-type: none"> CSV can be opened with any text editor in Windows like notepad, MS Excel, OpenOffice, etc.
<ul style="list-style-type: none"> Excel is a spreadsheet that saves files into its own proprietary format viz. xls orxlsx 	<ul style="list-style-type: none"> CSV is a format for saving tabular information into a delimited text file with extension .csv
<ul style="list-style-type: none"> Excel consumes more memory while importing data 	<ul style="list-style-type: none"> Importing CSV files can be much faster, and it also consumes less memory

2.Tabulate the different mode with its meaning.

- r - open for read (default)
- w - open for write, To create new file
- x - open for exclusive creation
- a - open for appending at the end of the file.
- t - open for text mode.(default)
- b - open for binary mode.
- +
- open for updating

3. Write the different methods to read a File in Python.

There are two ways to read a CSV file.

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

1.csv module's reader function (csv.reader())

- ✓ **csv.reader()** - to read CSV file into **list/tuple..**
- ✓ It will take each line of the file and make a list of all columns.
- ✓ It can **read data** from **csv files** of different formats like quotes (" "), pipe (|) and comma (,).

syntax : `csv.reader(fileobject,delimiter,fmtparams)`

file object :- contain path and mode of the file

delimiter :- an optional , (,) is default .

fmtparams: optional ,It is used to override the default values like skipinitialspace, quoting etc.

2.DictReader class (DictReader())

- ✓ DictReader() - To read a CSV file into a dictionary
- ✓ It works similar to the reader() class .
- ✓ It creates an object which maps data to a dictionary.
- ✓ The keys are given by the fieldnames as parameter.

4. Write a Python program to write a CSV File with custom quotes.

```
import csv
info = [
    ["SNO", "Person", "DOB"],
    ["1", "Madhu", "18/12/2001"],
    ["2", "Sowmya", "19/2/1998"],
    ["3", "Sangeetha", "20/3/1999"],
    ["4", "Eshwar", "21/4/2000"],
    ["5", "Anand", "22/5/2001"]
]
csv.register_dialect("myDialect", quoting=csv.QUOTE_ALL)
with open("c:\\pyprg\\ch13\\person.csv", "w") as f:
    writer = csv.writer(f, dialect="myDialect")
    for row in info:
        writer.writerow(row)
f.close()
```

OUTPUT:

```
"SNO","Person","DOB" "1","Madhu","18/12/2001" "2","Sowmya","19/2/1998"
"3","Sangeetha","20/3/1999" "4","Eshwar","21/4/2000" "5","Anand","22/5/2001"
```

5. Write the rules to be followed to format the data in a

CSV file.

- ✓ Each record (row of data) is to be located on a separate line,
- ✓ **A line break** by pressing **enter key**.
- ✓ The last record in the file line **break** is optional
- ✓ The first line of the **file is header**.
- ✓ The header will contain names .
- ✓ Header line is **optional**
- ✓ Within the header and each record, there may be one or more fields, separated by **commas**.
- ✓ Spaces are considered part of a field.
- ✓ The last field in the record **must not be** followed by a comma.
- ✓ Each field may or may not be enclosed in double quotes.
- ✓ Fields containing **line breaks** (CRLF), **double quotes**, and **commas** should be enclosed in double-quotes

4.What is modules ? What is the use of Modules?

- ✓ Modules used to break down large programs into small manageable files.
- ✓ Modules provide reusability of code
- ✓ We can define our most used functions in a module and import it.
- ✓ Example : matplotlib, csv, pip

5.What is the use of CD command? Give an example

- ✓ “cd” command used to change directory and path.
- ✓ Example : `c:\> CD D:\python program`

SECTION – B**1.Differentiate between Python and C++.**

PYTHON	C++
<ul style="list-style-type: none"> ✓ An interpreter is used to compile ✓ Dynamic-typed language ✓ Data type is not required while declaring variable ✓ It can act both as scripting and programming language ✓ Python uses Automatic Garbage Collection ✓ Code is shorter than C++ ✓ Return multiple values 	<ul style="list-style-type: none"> ✓ Compiler is used to compile ✓ Statically typed language ✓ Data type is required while declaring variable ✓ It is a programming language only ✓ It does not ✓ Code is lengthier than python ✓ Return only one value.

2.What are the Applications of Scripting Languages?

- ✓ To **automate** certain tasks in a program
- ✓ **Extracting information** from a data set
- ✓ **Less code** intensive
- ✓ It can bring **new functions** to applications.

3.Define MinGE.(or) What is MinGW? What is its use?

- ✓ MinGW refers to a **set of runtime header files**,
- ✓ MinGW allows to **compile** and **execute** C++ program through Python program using **g++**.

4. Identify the module ,operator, definition name for the following:welcome.display()

welcome - Module name

. - Dot operator

display() - function call

5. What is sys.argv? What does it contain?sys.argv :

- ✓ It's basically an **array**.
- ✓ Holding the **command-line arguments** of the program.
- ✓ It contains all the items **via** the command-line input

To use **sys.argv**, you will first have to **import sys**.Example: *main(sys.argv[1])*Accepts the **program file** (Python program) and the input file (C++ file) as a list(array).

- ✓ **sys.argv[0]**: It is first argument ,contains the python program.
- ✓ **sys.argv[1]**: It is the next argument, contains the name of C++ file.

SECTION – C**1. Write any 5 features of Python.**

- ✓ Python uses Automatic Garbage Collection.
- ✓ Python is a dynamically typed language.
- ✓ Python runs through an interpreter.
- ✓ Python code tends to be 5 to 10 times shorter than that written in C++.
- ✓ In Python, there is no need to declare types explicitly.

In Python, a function may accept an argument of any type, and return multiple values without any kind of declaration beforehand.

2. Explain each word of the following command.**COMMAND:**Python <filename.py> -<i> <C++ filename without cpp extension> Where ,

Python	Keyword to execute the Python program from command-line
<filename.py >	Name of the Python program to executed
-< i >	Input mode
<C++ filename without cpp extension>	Name of C++ file to be compiled and executed

3.What is the purpose of sys,os,getopt module in Python

(i) Python's sys module

sys.argv :

- ✓ It's basically an **array**.
- ✓ Holding the **command-line arguments** of the program.
- ✓ It contains all the items **via** the command-line input

To use **sys.argv**, you will first have to **import sys**.

Example: **main(sys.argv[1])**

Accepts the **program file** (Python program) and the input file (C++ file) as a list(array).

- ✓ **sys.argv[0]**: It is first argument ,contains the python program.
- ✓ **sys.argv[1]**: It is the next argument, contains the name of C++ file.

(ii) Python's OS Module

- ✓ The OS module allows to **interface** with the **Windows operating system** where Python is running on.

Example: **os.system()**

- ✓ Used to execute the C++ compiling in the **shell**(command window)

Syntax:

os.system ('g++' + <variable_name1> '- <mode>' + <variable_name2>)

Example:

os.system('g++' + cpp_file + '-o' + exe_file)

- ✓ **g++** compiler **compiles** the file **cpp_file** and **-o** (output) send to **exe_file**

(iii) Python getopt module

- ✓ The getopt module of Python helps to **split command-line options** and **arguments**.
- ✓ **getopt()** method returns value consisting of two elements, they are **opts** ,**args**.

opts and args .

- ✓ **Opts** contains list of **splitted** strings like **mode**, **path**.
- ✓ **args** contains any string if at all **not splitted** .
- ✓ **args** will be an **empty array** if there is **no error** .

Syntax:

<opts>,<args> =getopt.getopt(argv, options, [long_options])

- ✓ **argv** – This is the argument list of values to be splitted.
- ✓ **options** –it is for input or for output, with options (like 'i' or 'o') that followed by a colon (:).
- ✓ **long_options** –It is passed with a list of strings. It should be followed by an equal sign ('=').

Example.

opts, args = getopt.getopt (argv, "i:",['ifile=']

4. Write the syntax for getopt() and explain its arguments and return values.

Python getopt Module:

- ✓ The **getopt** module of Python helps you to parse (split) command-line options and arguments.
- ✓ This module provides two functions to enable command-line argument parsing.
- ✓ **getopt.getopt method:**
- ✓ This method parses command-line options and parameter list.

Syntax of getopt method:

`<opts>,<args>=getopt.getopt(argv, options, [long_options])`

- ✓ Here is the detail of the parameters –

- argv** -- This is the argument list of values to be parsed (splited). In our program the complete command will be passed as a list.
- options** -- This is string of option letters that the Python program recognize as, for input or for output, with options (like „i“ or „o“) that followed by a colon (:). Here colon is used to denote the mode.
- long_options**-- This parameter is passed with a list of strings. Argument of Long options should be followed by an equal sign ('=').

In our program the C++ file name will be passed as string and „i“ also will be passed along with to indicate it as the input file.

- ✓ **getopt()** method returns value consisting of two elements.
- ✓ Each of these values are stored separately in two different list (arrays) **opts and args** .
- ✓ **Opts** contains list of splitted strings like mode, path and args contains any string if at all not splitted because of wrong path or mode.
- ✓ **args** will be an empty array if there is no error in splitting strings by getopt().

Example:

❖ `opts, args = getopt.getopt (argv, "i:",['ifile='])`

where opts contains	('i', 'c:\\pyprg\\p4')
-i:	option nothing but mode should be followed by :
'c:\\pyprg\\p4'	value nothing but the absolute path of C++ file .

- ✓ In our examples since the entire command line commands are parsed and no leftover argument, the second argument args will be empty [].
- ✓ If args is displayed using print() command it displays the output as [].

Example: `>>>print(args)`

5. Write a Python program to execute the following

c++ coding

```
#include <iostream>
using namespace std;
int main()
{
cout<<"WELCOME";
return(0);
}
```

The above C++ program is saved in a file welcome.cpp

Now select **File->New** in **Notepad** and type the

following python program

```
import sys, os, getopt
def main(argv):
cpp_file = ""
exe_file = ""
opts, args = getopt.getopt(argv, "i:", ['ifile='])
for o, a in opts:
if o in ("-i", "--ifile"):
cpp_file = a + '.cpp'
exe_file = a + '.exe'
run(cpp_file, exe_file)
def run(cpp_file, exe_file):
print("Compiling " + cpp_file)
os.system('g++ ' + cpp_file + ' -o ' + exe_file)
print("Running " + exe_file)
print("-----")
print
os.system(exe_file)
print
if __name__ == '__main__':
main(sys.argv[1:])
```

✓ Save the File as **mypython.py**

✓ To compile and execute **welcome.cpp** file

python mypython.py -i welcome

OUTPUT: WELCOME

15.DATA MANIPULATION THROUGH SQL

SECTION – A

1.Mention the users who uses the Database.

- a)Humans
- b) other programs and applications

2.What are the steps to connect to database using sqlite3 in python?

Step 1: **import sqlite3**

Step 2: create a connection using **connect ()** method and **pass** the **name** of the database File

Step 3: Set the cursor object

```
cur = connection. cursor ()
```

Example:

```
import sqlite3
c=sqlite3.connect("D:\ela.db")
cur=c.cursor()
```

3.What is the advantage of declaring a column as “INTEGER PRIMARY KEY”

- ✓ If a column of a table is declared to be an INTEGER PRIMARY KEY,
- ✓ The NULL will be automatically **converted** into an **integer**
- ✓ For NULL ,it stores **1**

```
sql_command = """INSERT INTO Student (Rollno, Sname, Grade, gender, Average,
birth_date) VALUES (NULL, "Akshay", "B", "M", "87.8", "2001-12-12");"""
```

```
cursor.execute(sql_command))
```

4.Write the command to populate (add) record in a table. Give an example

- ✓ To populate (**add record**) the table "**INSERT**" command is passed to SQLite.
- ✓ Ex. cur.execute(“INSERT INTO stu(rno,name)VALUES(201,'ELANGO’)”)

5.Which method is used to fetch all rows from the database table?

The **fetchall()** method is used to fetch all rows from the database table

Example: *result = cursor.fetchall()*

SECTION – B

1.What is SQLite? What is it advantage?

- ✓ SQLite is a simple **relational database** system,
- ✓ It **saves** data in **files** or in **the internal memory** of the **computer**.
- ✓ It is designed to be **embedded** in applications,

Advantages: SQLite is fast, rigorously tested, and flexible, making it easier to work

2.Mention the difference between fetchone() and fetchmany()

fetchone()

- ✓ It returns the **next row** of a **result set** or None in case there is no row left.
- ✓ Using loop we can display all the records

Ex

```
import sqlite3
c = sqlite3.connect("Academy.db")
cur = c.cursor()
cur.execute("SELECT * FROM student")
result = cur.fetchone()
for r in result:
    print(r)
```

fetchmany()

- ✓ It returns the **next number of rows (n)** of the result set
- ✓ specified number of records is done by using fetchmany().

Ex .

```
import sqlite3
c = sqlite3.connect("Academy.db")
cur = c.cursor()
cur.execute("SELECT * FROM student")
r = cur.fetchmany(3)
print( r)
```

3.What is the use of Where Clause. Give a python statement Using the where clause.

- ✓ The **WHERE** clause is used to **filter data** based on a specified **condition** WHERE clause can be combined with **AND,OR ,NOT**

Example:

```
import sqlite3
c=sqlite3.connect("D:\ela.db")
cur=c.cursor()
cursor.execute("SELECT * FROM student where total >400")
a=cur.fetchall()
print(x)
```

4.Read the following details.Based on that write a python script to display department wise records

database name :- organization.db

Table name :- Employee

Columns in the table :- Eno, EmpName, Esal, Dept

```
import sqlite3  
c = sqlite3.connect("organization.db")  
cur = c.cursor()  
cur.execute("create table employee(eno  
integer,empname varchar(20),  
cur.execute("SELECT * FROM Employee GROUP BY dept")  
result = cur.fetchall()  
for r in result:  
    print(r)
```

5.Read the following details.Based on that write a python script to display records in desending order of Eno

database name :- organization.db

Table name :- Employee

Columns in the table :- Eno, EmpName, Esal, Dept.

```
import sqlite3  
c = sqlite3.connect("organization.db")  
cur = c.cursor()  
cur.execute("SELECT * FROM Employee SORT BY Eno DESC")  
result = cur.fetchall()  
for r in result:  
    print(r)
```

SECTION – C

1. Write in brief about SQLite and the steps used to use it.

- ✓ SQLite is a simple **relational database** system,
- ✓ It saves data in **files** or in **the internal memory** of the **computer**.
- ✓ It is designed to be **embedded** in applications,

Advantages:

- ✓ SQLite is fast, rigorously tested, and flexible, making it easier to work

Steps to connect to database using sqlite3 in python

Step 1: **import sqlite3**

Step 2: create a connection using **connect ()** method and **pass** the **name** of the database File

Step 3: Set the cursor object

```
cursor = connection. cursor ()
```

Example:

```
import sqlite3
c=sqlite3.connect("D:\ela.db")
cur=c.cursor()
cur.execute("create TABLE stu(no integer not null primary key,name varchar(20);")
cur.execute("insert into stu(no,name)values(101,'elango')")
c.commit()
cur.execute("select * from stu;")
a=cur.fetchone()
for x in a:
print(x)
```

2. Write the Python script to display all the records of the following table using fetchmany()

Icode	ItemName	Rate
1003	Scanner	10500
1004	Speaker	3000
1005	Printer	8000
1008	Monitor	15000
1010	Mouse	700

CODING:

```

import sqlite3
c=sqlite3.connect("D:\ela.db")
cur=c.cursor()
cur.execute("create table item(icode integer not null primary key,itemName
varchar(20),Rate integer;")
cur.execute("insert into item (icode,itemName,Rate)values(1003,Scanner,10500)")
cur.execute("insert into item (icode,itemName,Rate)values(1004,Speaker,3000)")
cur.execute("insert into item (icode,itemName,Rate)values(1005,printer,8000)")
cur.execute("insert into item (icode,itemName,Rate)values(1008,monitot,15000)")
cur.execute("insert into item (icode,itemName,Rate)values(1010,mouse,700)")
c.commit()
cur.execute("select * from item;")
x=cur.fetchmany(7)
print(x)

```

3.What is the use of HAVING clause. Give an example python script.

- ✓ It is used along with **GROUP BY** clause in the **SELECT** statement to **filter data** based on the **group** functions

```

import sqlite3
c=sqlite3.connect("D:\ela.db")
cur=c.cursor()
cur.execute("SELECT GENDER,COUNT(GENDER) FROM Stu GROUP BY
GENDER HAVING COUNT(GENDER)>3")
a=cur.fetchall()
print(x)

```

4.Write a Python script to create a table called ITEM with following specification.

Add one record to the table.

Name of the database :- ABC

Name of the table :- Item

Column name and specification :-

Icode :-	integer and act as primary key
Item Name :-	Item Name :-
Rate :-	Integer
Record to be added :-	1008, Monitor,15000

CODING:

```

import sqlite3

c=sqlite3.connect("D:\ela.db")

cur=c.cursor()

cur.execute("create TABLE ITEM(icode integer not null primary key,Itemname
varchar(25),rate integer;")

cur.execute("insert into item (no,name)values(1008,Monitor,15000)")

c.commit()

cur.execute("select * from item;")

a=cur.fetchall()

for x in cur:

    print(x)

```

5. Consider the following table Supplier and item .Write a python script for (i) to (ii)

SUPPLIER				
Suppno	Name	City	Icode	SuppQty
S001	Prasad	Delhi	1008	100
S002	Anu	Bangalore	1010	200
S003	Shahid	Bangalore	1008	175
S004	Akila	Hydrabad	1005	195
S005	Girish	Hydrabad	1003	25
S006	Shylaja	Chennai	1008	180
S007	Lavanya	Mumbai	1005	325

PYTHON SCRIPT:

i) Display Name, City and Itemname of suppliers who do not reside in Delhi.

```

import sqlite3

connection = sqlite3.connect("ABC.db")

cursor.execute("SELECT Supplier.Name, Supplier.City,Item.ItemName FROM
Supplier,Item WHERE Supplier.Icode = Item.Icode AND Supplier.City NOT In Delhi ")

s = [i[0]
for I in cursor.description]

print(s)

result = cursor.fetchall()

for r in result:

    print r

```

OUTPUT:

```
[„Name“,      „City“,      „ItemName“]
[„Anu“,      „Bangalore“,  „Scanner“]
[„Shahid“,   „Bangalore“,  „Speaker“]
[„Akila“,    „Hydrabad“,   „Printer“]
[„Girish“,   „Hydrabad“,   „Monitor“]
[„Shylaja“,  „Chennai“,    „Mouse“]
[„Lavanya“,  „Mumbai“,     „CPU“]
```

Increment the SuppQty of Akila by 40

```
import sqlite3
connection = sqlite3.connect("ABC.db")
cursor.execute("UPDATE Supplier ST SuppQty = SuppQty +40
WHERE Name = „ Akila“ ")
cursor.commit()
result = cursor.fetchall()
print (result)
connection.close()
```

OUTPUT:

(S004, „Akila“, „Hydrabad“, 1005, 235)

16.DATA VISUALIZATION USING PYPLOT: LINE CHART, PIECHARTAND BAR CHART

SECTION – A

1.Define Visualization .

- ✓ Data Visualization is the **graphical representation** of **information** and **data**.
- ✓ Used to communicate information **visually** to users.

2.List the general types of data visualization

- ✓ Charts
- ✓ Tables
- ✓ Graphs
- ✓ Maps
- ✓ Infographics
- ✓ Dashboards

3.List the types of Visualizations in Matplotlib.

- ✓ Line plot
- ✓ Scatter plot
- ✓ Histogram
- ✓ Box plot
- ✓ Bar chart and
- ✓ Pie chart

4.How will you install Matplotlib?

- ✓ **Matplotlib** can be installed using pip software.
- ✓ Importing Matplotlib using the command:**import matplotlib.pyplot as plt**
- ✓ Matplotlib can be imported in the workspace.
- ✓ Pip is a management software for installing python packages.
- ✓ Type the following in your command prompt: **python -m pip install -U matplotlib**
- ✓ This command will download **matplotlib** from the source library

5. Write the difference between the following functions: `plt.plot([1,2,3,4])`, `plt. plot([1,2,3,4], [1,4,9,16])`.

`plt.plot([1,2,3,4])`

- ✓ x-values are 0,1,2,3 (automatically generates)
- ✓ y-values are 1,2,3,4
- ✓ plots using the point (0,1),(1,2),(2,3),(3,4)

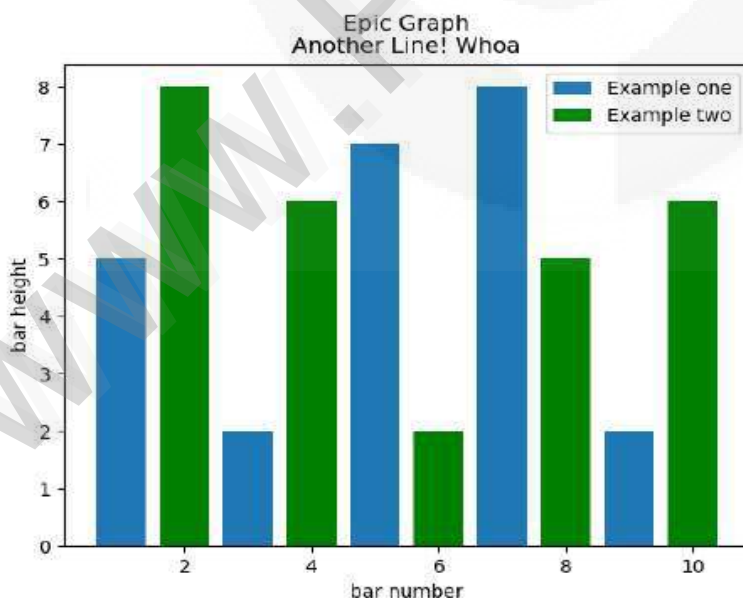
`plt. plot([1,2,3,4], [1,4,9,16])`

- ✓ x- values are [1,2,3,4]
- ✓ y- values are [1,4,9,16]
- ✓ plots using the point (1,1), (2,4), (3,9) and (4,16)

SECTION – B

1. Draw the output for the following data visualization

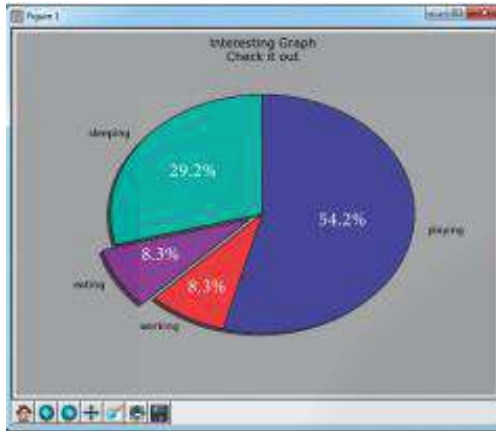
```
import matplotlib.pyplot as plt
plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Example one")
plt.bar([2,4,6,8,10],[8,6,2,5,6], label="Example two", color='g')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title('Epic Graph\nAnother Line! Whoa')
plt.show()
```



2.How will you install Matplotlib?

- ✓ **Matplotlib** can be installed using pip software.
- ✓ Importing Matplotlib using the command: ***import matplotlib.pyplot as plt***
- ✓ Matplotlib can be imported in the workspace.
- ✓ Pip is a management software for installing python packages.

3.Write the plot for the following pie chart output.



```
import matplotlib.pyplot as plt slices = [7,2,2,13]
activities = [„sleeping“, „eating“, „working“, „playing“]
plt.pie(slices, labels=activities, atopct = „y.1.1 f%%“)
plt.title(„Interesting Graph Ceck It Out“)
plt.show()
```

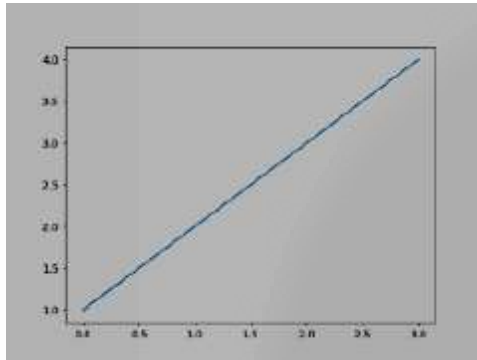
SECTION – C

1.Explain in detail the types of pyplots using Matplotlib.**Line Chart**

- ✓ It displays information as a **series of data points** called '**markers**' connected by **straight line segments**.
- ✓ Used to visualize a data over **intervals of time**

Example

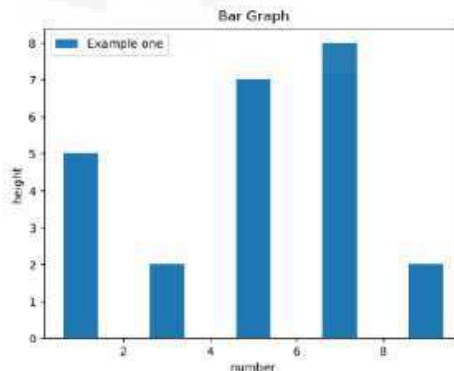
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.show()
```

**Bar Chart**

- ✓ It shows the relationship between a **numerical** variable and a **categorical** variable.
- ✓ Bar chart represents data with **rectangular bars**.
- ✓ **Height** of each bar represents to the **value**.
- ✓ The bars can be plotted **vertically** or **horizontally**.
- ✓ **plt.bar()** function used to make a bar chart

Example:

```
import matplotlib.pyplot as plt
plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Example one")
plt.legend()
plt.xlabel('number')
plt.ylabel('height')
plt.title('Bar Graph')
plt.show()
```

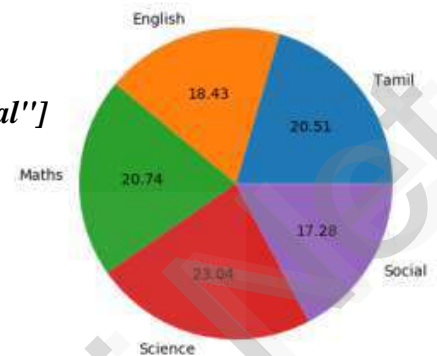


Pie Chart

- ✓ It is a **circular** graphic which is divided into **slices** to represents numerical **proportion**.
- ✓ It shows the relationship of parts **out of a whole**.

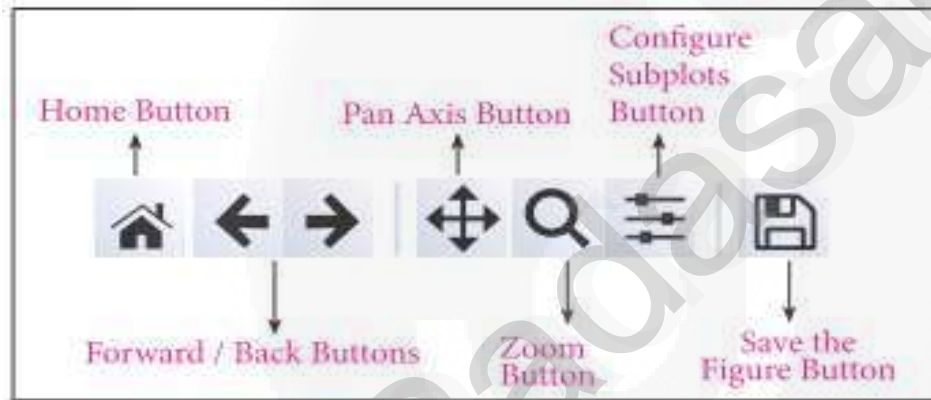
Example:

```
import matplotlib.pyplot as plt
sizes = [89, 80, 90, 100, 75]
labels = ["Tamil", "English", "Maths", "Science", "Social"]
plt.pie(sizes, labels = labels, autopct = "%.2f")
plt.axes().set_aspect("equal")plt.show()
```



- ✓ **plt.pie()** function used to make a Pie
- ✓ The **autopct** parameter allows us to display the percentage value using the Python string formatting

2. Explain the various buttons in a matplotlib window.



Home Button :

- ✓ It helps to begun navigating your chart.
- ✓ To return back to the original view

Forward/Back buttons :

- ✓ Used to move back and previous

Pan Axis:

- ✓ Used to click and drag your graph move around.

Zoom:

- ✓ Used to zoom into specifically.
- ✓ Zooming in - a left click and drag.
- ✓ Zoom out - a right click and drag.

Configure Subplots:

- ✓ This button allows you to configure various spacing options with your figure and plot.

Save Figure:

- ✓ To save your figure in various forms.

3.Explain the purpose of the following functions:

a. plt.xlabel b. plt.ylabel c. plt.title d. plt.legend() e. plt.show()

plt.xlabel() - Specifies label for X axis.

plt.ylabel() - Specifies label for Y axis.

plt.title() - Specifies Title to the graph.

plt.legend() - Displays legend on the graph.

plt.show() - Display the chart or graph

4. Difference between the Histogram AND Bar Graph

Histogram	Bar Graph
✓ shows the frequency of numerical data.	✓ Shows comparisons of different categories of data.
✓ Represents continuous variables.	✓ Represents discrete variables.
✓ Represents numerical data	✓ Represents categorical data.
✓ No gap between the bars	✓ proper spacing between bars
✓ Items are numbers Items are individual entities	✓ Bars cannot be changed
✓ Bars can be changed Width of bars may or may not be always same	✓ Width of bars always same

MOST IMPORTANT QUESTIONS (5 MARKS):

1. Explain pure function with example.
2. Explain impure function with example.
3. Discuss about Linear search.
4. Explain Binary search
5. How will you facilitate data abstraction ? Explain with example.
6. Explain Types of scope.
7. Write any five characteristics of Modulus.
8. Write any five benefits of using modular programming.
9. Write a simple example using LEGB rule
10. Characteristics of algorithm
11. Describe in detail procedure script mode programming.
12. Explain input(),&print()
13. Write a program to display multiplication table.
14. Explain for loop.
15. Explain the scope of variables with example.
16. Explain string operators in python
17. Write are the different ways to insert an element in a list with example
18. What is the use of range () ? Explain.
19. Explain the set operations with example.
20. Differentiate DBMS&RDBMS.
21. Explain the characteristics of DBMS.
22. Explain the components of DBMS
23. What are the different types constraints and their functions.
24. What are the components of SQL. write commands in each
25. Write SQL statements to create a table for employee having any five fields and create a table constraint for the employee table.
26. Differentiate Excel file & CSV file.
27. Write any five features of python.
28. Explain each word of the following command.
29. Python <filename.py>-<i> < c++ filename without cpp extension>
30. What is the purpose of sysmos, getopt module in python. Explain
31. Write a note on aggregate functions of SQL.
32. What is the use of HAVING clause. Give an example python script.

33. Explain the various buttons in a matplotlib window.
34. Explain the purpose of the following function: a. plt.xlabel b. plt.ylabel c. plt.title d. plt.legend() e. plt.show()
35. Explain the different types of data model.

**பொது தேர்வில் 100 க்கு 100 மதிப்பெண் பெற
வாழ்த்துக்கள்**

PREPARED BY

P.PERIYASAMY.,M.PHIL.,M.ED.,

DEPARTMENT OF COMPUTER SCIENCE ,

SHANTHINIKEETHAN HR SEC SCHOOL,

KARIMANGALAM-635111.

CELL:9095346634.

**மேலும் ஒரு மதிப்பெண், CENTUM MATERIAL
COMPULSORY QUESTIONS**

தேவைக்கு தொடர்பு கொள்ளவும்.

E-MAIL: periyasamyps1@gmail.com

Whatsapp: 9095346634.

