VELAMMAL MAT. HR. SEC. SCHOOL, SURAPET, CHENNAI- 66
STD: XII (COMPUTER SCIENCE)
QUARTERLY EXAM – CH 1 TO 5 - REVISION

CHAPTER – 1 FUNCTION

PART - II

1. What are Subroutines?

- Subroutines are the basic building blocks of computer programs.
- Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly.
- In Programming languages subroutines are called as Functions.

2. Define function with respect to Programming language?

- A function is a unit of code that is defined within a greater code structure.
- A function contains a set of code that works on many kinds of inputs, like
  - Variants
  - Expressions and
  - Produces a concrete output.

All functions are static definitions.

There is no dynamic function definitions.

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

X:=(78) .

- It has an expression in it.
- But (78) is not itself an expression, It is a function definition.
- Definitions bind values to names, in this case the value 78 is bound to the name 'X'.

4. Difference between interface and implementation.

Interface	Implementation
Interface just defines what an object can do, but won't actually do it	Implementation carries out the instructions defined in the interface

5. **Which of the following is a normal function definition and which is recursive function definition**.

```
i) let rec sum x y :
        return x + y
```
Ans : Recursive Function

```
ii) let disp :
        print 'welcome'
```
Ans : Normal Function

```
iii) let rec sum num :
        if (num!=0) then
                return num + sum  (num-1)
        else
                return num
```
Ans : Recursive Function

PART-III

1. Characteristics of interface

1. The class template specifies the interfaces to enable an object to be created and operated properly.
2. An object's attributes and behaviour is controlled by sending functions to the object.

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, and the value returned derives from the external memory accessed.

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.
- In such cases the function is called impure function.
- When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called.
- For example the mathematical function **random ()** will give different outputs for the same function call.

4. Differentiate pure and impure function

Pure Function	**Impure Function**
The return value of the pure functions depends on its arguments passed.	The return value of the impure functions does not depend on its arguments passed.
Pure Function do not have any side effects	Impure Function have side effects
If you call the pure functions with the same set of arguments, you will always get the same return values.	if you call the impure functions with the same set of arguments, you might get the different return values
They do not modify the arguments which are passed to them	They may modify the arguments which are passed to them
Example: strlen()	Example : random(), Date()

5. What happens if you modify a variable outside the function? Give an example.

In the example the value of y get changed inside the function definition due to which the result will change each time.
The side effect of the inc () function is it is changing the data of the external visible variable **'y'**.
Eg:

```
let y: = 0
let (int) inc (int) x:
        y: = y + x;
        return (y)
```

PART-IV

1. What are called Parameters and write a note on
i) Parameter without Type   ii) Parameter with Type

Parameters (and arguments)

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

i. Parameter without Type

- Parameters are the variables in a function definition
  (requires: b>=0)

```
(returns: a to the power of b)
let rec pow a b:=
        if b=0 then 1
        else a * pow a (b-1)
```

- Variable ‘ b’ is the parameter
- The value which is passed to the variable ‘ b’ is the argument.
- The pre-condition **(requires)** and post condition **(returns)** of the function is given.
- We have not mentioned any types: (data types)
- if expression can return 1 in the then branch, the entire if expression has type int.
- if expression has type ‘int’, the function's return type also be ‘int’.
- ‘b’ is compared to 0 with the equality operator, so ‘b’ is also a type of ‘int’.
- ‘a’ is multiplied with an expression using the * operator, ‘a’ must be an int.

ii. Parameter with Type

```
(requires: b> 0 )
(returns: a to the power of b )
let rec pow (a: int) (b: int) : int :=
        if b=0 then 1
        else a * pow a (b-1)
```

- When we write the type annotations for **‘a’** and **‘b’,** the parentheses are mandatory.
- Generally we can leave out these annotations, because it's simpler to let the compiler infer them.
- There are times we may want to explicitly write down types.
- This is useful when you get a type error from the compiler that doesn't make sense.
- Explicitly annotating the types can help with debugging such an error message

2. 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 - **gcd**

ii) Identify the statement which tells it is a recursive function **let rec gcd a b :=**

iii) Name of the argument variable – **a and b**

iv) Statement which invoke the function recursively **gcd b (a mod b)**

v) Statement which terminates the recursion **b <> 0**

3. Explain with example pure and impure function

Pure functions

- Pure functions are functions which will give exact result when the same arguments are passed.
- For example the mathematical function sin (0) always results **0**.
- Every time you call the function with the same arguments, you will always get the same result.
- A function can be a pure function provided it should not have any external variable, which will alter the behaviour of that variable
- **Eg:**

let square x
return: x * x
	let i: = 0;
	if i <strlen (s) then
	--Do something which doesn't affect s

	++i

Impure functions

- The variables used inside the function may cause side effects, though the functions which are not passed with any arguments.
- In such cases the function is called impure function.
- When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it’s called.
- For example the mathematical function random() will give different outputs for the same function call.
- Eg:

```
--let Random number
let a := random()
        if a > 10 then
                return a
        else
                return 10
```

4. Explain with an example interface and implementation

- An interface is a set of action that an object can do.
- In Object Oriented Programming language, an Interface is a description of all functions that a class must have in order to be a new **interface**.
- The purpose of interfaces is to allow the computer to enforce the properties of the class
- A class declaration combines the external interface (its local state) with an implementation of that interface (the code that carries out the behaviour).
- An object is an instance created from the class.
- The interface defines an object’s visibility to the outside world.
- Implementation carries out the instructions defined in the interface
- In object oriented programs classes are the interface and how the object is processed and executed is the implementation.

For example

When you press a light switch, the light goes on, you may not have cared how it splashed the light.
Anything that **"ACTS LIKE"** a light, should have function definitions like turn_on () and a turn_off().

Extra question and answer:

1. Function Specification

a:= (24)
Definitions bind values to names, in this case the value 24 is bound to the name ‘a’.

2. The syntax to define functions

- The definition is introduced by the keyword let,
- Followed by the name of the function and its arguments;
- The formula that computes the image of the argument is written after an = sign.
- If you want to define a recursive function: use “let rec” instead of “let”.
  let rec fn a1 a2 ... an := k
- ‘fn’ → variable indicating an identifier being used as a function name.
- ‘a1’ to ‘an’ → variables indicating the identifiers used as parameters.
- The keyword ‘rec’→ is required if ‘fn’ is to be a recursive function

3. Define Recursive function:

- A function definition which call itself is called recursive function

4. The syntax for function types:

$x \rightarrow y$
$x1 \rightarrow x2 \rightarrow y$
$x1 \rightarrow \ldots \rightarrow xn \rightarrow y$

The type $x \rightarrow y$. A function that gets an input of type ‘x’ and returns an output of type ‘y’
$x1 \rightarrow x2 \rightarrow y$. A function takes two inputs ‘x1’ and ‘x2’, and returns an output of type ‘y’.
$x1 \rightarrow \ldots \rightarrow xn \rightarrow y$ has type ‘x’ as input of n arguments and ‘y’ type as output.

Choose the best answer

1. The small sections of code that are used to perform a particular task is called

   (A) **Subroutines** (B) Files (C) Pseudo code (D) Modules

2. Which of the following is a unit of code that is often defined within a greater code structure?

   (A) Subroutines (B) **Function** (C) Files (D) Modules

3. Which of the following is a distinct syntactic block?

   (A) Subroutines (B) Function (C) **Definition** (D) Modules

4. The variables in a function definition are called as

   (A) Subroutines (B) Function (C) Definition (D) **Parameters**

5. The values which are passed to a function definition are called

   (A) **Arguments** (B) Subroutines (C) Function (D) Definition

6. Which of the following are mandatory to write the type annotations in the function definition?

   (A) Curly braces (B) **Parentheses** (C) Square brackets (D) indentations

7. Which of the following defines what an object can do?

   (A) Operating System (B) Compiler (C) **Interface** (D) Interpreter

8. Which of the following carries out the instructions defined in the interface?

   (A) Operating System (B) Compiler (C) **Implementation** (D) Interpreter

9. The functions which will give exact result when same arguments are passed are called

   (A) Impure functions (B) Partial Functions
   (C) Dynamic Functions (D) **Pure functions**

10. The functions which cause side effects to the arguments passed are called

    (A) **impure function** (B) Partial Functions
    (C) Dynamic Functions (D) Pure functions

Hands on Practice

1. Write algorithmic function definition to find the minimum among 3 numbers.

```
let min 3 x y z :=
if x < y then
if x < z then x else z
else
if y < z then y else z
```

2. Write algorithmic recursive function definition to find the sum of n natural numbers.

```
let rec sum n:
if (n!=0) then return n + sum (n-1)
else
return n
```

CHAPTER – 2 - DATA ABSTRACTION

PART-II

1. What is Abstract Data Type (ADT)

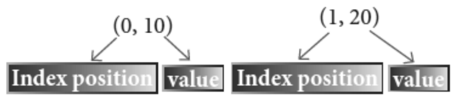
- It is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.
- The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.

2. Differentiate Constructors and Selectors

Constructors	Selectors
Constructors are functions that build the abstract data type.	Selectors are functions that retrieve information from the data type.
Constructors create an object, bundling together different pieces of information.	Selectors extract individual pieces of information from the object
Example : city = makecity (name, lat, lon)	Example: getname(city), getlat(city), getlon(city)

3. What is Pair? Give an example.

- Python provides a compound structure called Pair which is made up of list or Tuple.
- The first way to implement pairs is with the List construct.
- Any way of bundling two values together into one can be considered as a pair.
- Lists are a common method to do so.
- Therefore List can be called as Pairs.
- Example: lst := [10, 20]
- lst[(0, 10), (1, 20)] - where



4. What is List? Give an Example.

- List is constructed by placing expressions within square brackets separated by commas.
- Such an expression is called a list literal.
- Example :

  lst := [10, 20]

5. What is Tuple? Give an example.

- A tuple is a comma-separated sequence of values surrounded with parentheses.
- Tuple is similar to a list.
- Example colour= ('red', 'blue', 'Green')

PART-II

1. Differentiate Concrete Data Type and Abstract Data Type.

Concrete Data Type	Abstract Data Type
It is a Data type whose representation is known	It is a Data type whose representation is unknown
Concrete data types or structures (CDT's) are direct implementations of a relatively simple concept	It is a type (or class) for objects whose behavior is defined by a set of value and a set of operations
Concrete data representation is defined as an independent part of the program	The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented

2. Which strategy is used for program designing? Define that strategy.

- A powerful strategy for designing programs: **'wishful thinking'**
- **Wishful Thinking** is the formation of beliefs and making decisions according to what might be pleasing to imagine instead of by appealing to reality.

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

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

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

- The elements of a list can be accessed in two ways.
- **Method I**
  - Method of multiple assignments
  - Which unpacks a list into its elements and binds each element to a different name.
  - **Example:** lst := [10, 20]
    x, y := lst
  - In the above example ***x*** will become10 and ***y*** will become 20.
- o **Method II**
  - o Accessing the elements in a list is by the element selection operator, also expressed using square brackets.
  - o Unlike a list literal, a square brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.
  - o Example: lst := [10, 20]

    lst[0]
    10
    lst[1]
    20
- In both the example mentioned above mathematically we can represent list similar to a set

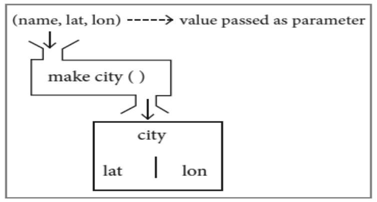
5. Identify Which of the following are List, Tuple and class ?

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

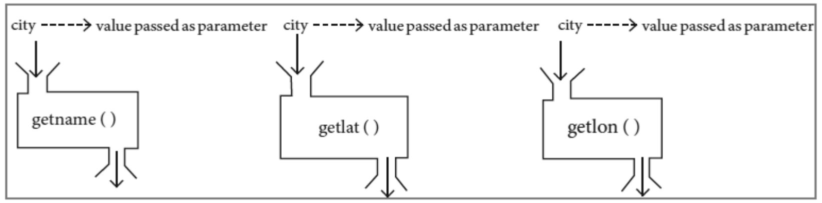
PART - IV

1. How will you facilitate data abstraction? Explain it with suitable example

- To facilitate data abstraction, you will need to create two types of functions:
  1. Constructors
  2. Selectors
- **Constructors:**
  - Constructors are functions that build the abstract data type.
  - Constructors create an object, bundling together different pieces of information.
  - Example : city = makecity (name, lat, lon)
  - Here makecity (name, lat, lon) is the constructor which creates the object city.

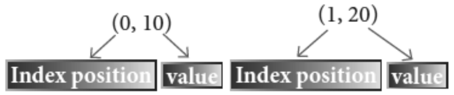


- **<u>Selectors:</u>**
  - Selectors are functions that retrieve information from the data type.
  - Selectors extract individual pieces of information from the object.
  - Example:
  - getname(city)
  - getlat(city)
  - getlon(city)
- These are selectors because these functions extract the information of the city object.


city -----> value passed as parameter
city -----> value passed as parameter
city -----> value passed as parameter
getname ( )
getlat ( )
getlon ( )

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

- List is constructed by placing expressions within square brackets separated by commas.
- Such an expression is called a list literal.
- Example :
  lst := [10, 20]
- The elements of a list can be accessed in two ways.
- **Method I**
  - Method of multiple assignments
  - Which unpacks a list into its elements and binds each element to a different name.
  - **<u>Example:</u>** lst := [10, 20]
    x, y := lst
  - In the above example ***x*** will become10 and ***y*** will become 20.
- **Method II**
  - Accessing the elements in a list is by the element selection operator, also expressed using square brackets.
  - Unlike a list literal, a square brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.
  - Example: lst := [10, 20]
    lst[0]
    10
    lst[1]
    20
- In both the example mentioned above mathematically we can represent list similar to a set
- Example: lst := [10, 20]
- lst[(0, 10), (1, 20)] - where



- Any way of bundling two values together into one can be considered as a pair.
- Lists are a common method to do so.
- Therefore List can be called as Pairs.

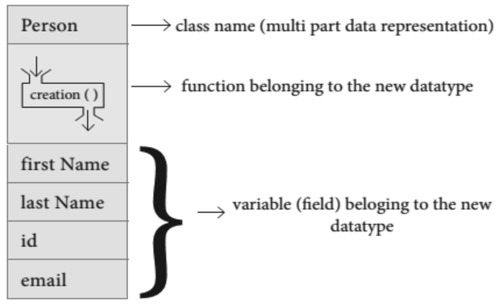
3. How will you access the multi-item. Explain with example.

- The structure construct (class construct) to represent multi-part objects where each part is named (given a name).
- A class as bundled data and the functions that work on that data.
- The class (structure) construct defines the form for multi-part objects that represent a person.
- Its definition adds a new data type, in this case a type named Person.
- Once defined, we can create new variables (instances) of the type.
  Consider the following pseudo code:

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

- **Person** is referred to as a class or a type

The new data type Person is pictorially represented as



Person
class name (multi part data representation)
creation ( )
function belonging to the new datatype
first Name
last Name
id
email
variable (field) beloging to the new datatype

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	

<u>EXTRA QUESTION AND ANSWER:</u>

1. What are different ways to implement ADT?

- The List ADT can be implemented using singly linked list or doubly linked list.
- Stack ADT and Queue ADT can be implemented using lists.

2. What is Concrete data type (CDT)

- Concrete data types or structures (CDT's) are direct implementations of a relatively simple concept.
- A concrete data type is a data type whose representation is known

3. What is Abstraction?

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

4. Pseudo code to compute the distance between two city objects:

```
distance(city1, city2):
lt1, lg1 := getlat(city1), getlon(city1)
lt2, lg2 := getlat(city2), getlon(city2)
return ((lt1 - lt2)**2 + (lg1 - lg2)**2)) ½
```

5. How will you represent Abstract data type using Rational numbers?

- Any program consist of two parts
  1. The part that operates on abstract data and
  2. The part that defines a concrete representation,
- It is connected by a small set of functions that implement abstract data in terms of the concrete representation
- **Example:**
- A rational number is a ratio of integers, and rational numbers constitute an important sub-class of real numbers.
- A rational number such as 8/3 or 19/23 is typically written as: <numerator>/<denominator>

Solution:

```
- - constructor
- - constructs a rational number with numerator n, denominator d
rational(n, d)
- - selector
numer(x) → returns the numerator of rational number x
denom(y) → returns the denominator of rational number y
```

- The pseudo code for the representation of the rational number using the above constructor and selector is

```
x, y :=8,3
rational(n,d)
numer(x)/numer(y)
- - output : 2.6666666666666665
```

6. Write the pseudo code to represent Rational Numbers Using List.

```
rational(n, d):
    return [n, d]
numer(x):
    return x[0]
denom(x):
    return x[1]
```

7. What are the Difference between List and Tuple?

LIST	TUPLE
L ist is constructed by placing expressions within square brackets separated by commas.	A tuple is a comma-separated sequence of values surrounded with parentheses.
The elements of a list can be changed	That elements of a tuple cannot be changed
Example: lst := [10, 20]	Example: colour= ('red', 'blue', 'Green')

8. Write the Representation of Tuple as a Pair.

nums := (1, 2)

```
nums[0]
1
nums[1]
2
```

- The square bracket notation is used to access the data you stored in the pair.
- The data is zero indexed, meaning you access the first element with nums[0] and the second with nums[1].

Choose the best answer (1 Mark)

1. Which of the following functions that build the abstract data type?
A) **Constructors** B) Destructors C) recursive D)Nested

2. Which of the following functions that retrieve information from the data type?
A) Constructors B) **Selectors** C) recursive D)Nested

3. The data structure which is a mutable ordered sequence of elements is called
A) Built in B) **List** C) Tuple D) Derived data

4. A sequence of immutable objects is called
A) Built in B) List C) **Tuple** D) Derived data

5. The data type whose representation is known are called
A) Built in data type B) Derived data type
C) **Concrete data type** D) Abstract data type

6. The data type whose representation is unknown are called
A) Built in data type B) Derived data type
C) Concrete data type **D) Abstract data type**

7. Which of the following is a compound structure?
A) **Pair** B) Triplet C) single D) quadrat

8. Bundling two values together into one can be considered as
A) **Pair** B) Triplet C) single D) quadrat

9. Which of the following allow to name the various parts of a multi-item object?
A) Tuples B) Lists C) **Classes** D) quadrats

10.Which of the following is constructed by placing expressions within square brackets?
A) Tuples B) **Lists** C) Classes D) quadrats

Chapter – 3 - SCOPING

PART - II

1. What is Scope?

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- The scope of a variable is that part of the code where it is visible.

2. Why scope should be used for variable. State the reason

- To limit a variable's scope to a single definition.
- Changes inside the function can't affect the variable on the outside of the function in unexpected ways

3. What is Mapping?

- The process of binding a variable name with an object is called mapping.
- := (equal to sign) is used in programming languages to map the variable and object.
- **Example: a:=5**

4. What do you mean by Namespaces?

- Programming languages keeps track of all the mappings with namespaces.
- Namespaces are containers for mapping names of variables to objects.

5. How Python represents the private and protected Access specifiers?

- Python doesn't have any mechanism that effectively restricts access to any instance variable or method.
- Python prescribes a convention of prefixing the name of the variable or method with single or double underscores to emulate the behaviour of protected and private access specifiers.
- All members in a Python class are public by default

Note:

- All members in C++ and java they are private by default.
- Any member can be accessed from outside the class environment in Python which is not possible in C++ and java.

PART - III

1. Explain Types of scopes for variable or LEGB rule with example.

SCOPE:

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- The scope of a variable is that part of the code where it is visible.

TYPES OF VARIABLE SCOPE:

- Local Scope
- Enclosed Scope
- Global Scope
- Built-in Scope

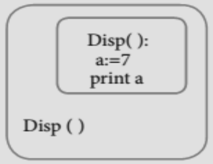
LEGB RULE:

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

Local(**L**)	Defined inside function/class
Enclosed(**E**)	Defined inside enclosing functions (Nested function concept)
Global(**G**)	Defined at the uppermost level
Built-in (**B**)	Reserved names in built-in functions (modules)

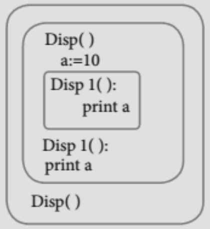
i) LOCAL SCOPE:

- Local scope refers to variables defined in current function.
- A function will always look up for a variable name in its local scope.
- Only if it does not find it there, the outer scopes are checked.
- Example:

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

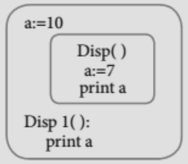
ii) ENCLOSED SCOPE:

- A variable which is declared inside a function which 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.
- When a compiler or interpreter searches for a variable in a program, it first search Local, and then search Enclosing scopes.
- Example:

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

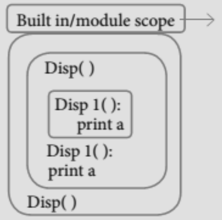
iii) GLOBAL SCOPE:

- A variable which is declared outside of all the functions in a program is known as global variable.
- Global variable can be accessed inside or outside of all the functions in a program.
- Example:

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

iv) BUILT-IN-SCOPE:

- The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter.
- Any variable or module which is defined in the library functions of a programming language has Built-in or module scope.
- Normally only Functions or modules come along with the software, as packages. Therefore they will come under Built in scope.
- Example:

Entire program	Library files associated
 Built in/module scope; Disp( ); Disp 1( ): print a; Disp 1( ): print a; Disp( )	with the software

4. What is Access Control? Why access control is required?

- Access control is a security technique that regulates who or what can view or use resources in a computing environment.
- It is a fundamental concept in security that minimizes risk to the object.
- It is a selective restriction of access to data.
- In Object oriented programming languages it is implemented through access modifiers.

- Classical object-oriented languages, such as C++ and Java, control the access to class members by public, private and protected keywords

5. Identify the scope of the variables in the following pseudo code and write its output

```
color:= Red
mycolor():
   b:=Blue
   myfavcolor():
       g:=Green
       print color, b, g
   myfavcolor()
   print color, b
mycolor()
print color
```

Scope of the variables:

color – Global
b – Enclosed
g - Local

Output:

```
Red Blue Green
Red Blue
Red
```

PART - IV

2. Write ANY 5 Characteristics of Modules.

1. Modules contain instructions, processing logic, and data.
2. Modules can be separately compiled and stored in a library.
3. Modules can be included in a program.
4. Module segments can be used by invoking a name and some parameters.
5. Module segments can be used by other modules.

3. Explain ANY 5 benefits IN USING modular programming

- Less code to be written.
- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming 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, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

EXTRA QUESTION ANSWER:

1. What is Life time?

- The duration for which a variable is alive is called its 'life time'

2. Write a short note on 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 work perfectly on individual level and can be integrated with other modules. A software program can be divided into modules to ease the job of programming and debugging as well.
- A program can be divided into small functional modules that work together to get the output.

3. Write a short note on Modular programming.

- The process of subdividing a computer program into separate sub-programs is called Modular programming.
- Modular programming enables programmers to divide up the work and debug pieces of the program independently.
- Examples: procedures, subroutines, and functions.

4. Write a short note on different access Specifiers?

There are three access specifiers

1.	**Private**
2.	**Public**
3.	**Protected**

- **Private members** of a class are denied access from the outside the class. They can be handled only from within the class.
- **Public members** (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method.
- **This arrangement of private instance variables and public methods ensures the principle of data encapsulation**.
- **Protected members** of a class are accessible from within the class and are also available to its sub-classes. No other process is permitted access to it. This enables specific resources of the parent class to be inherited by the child class.

Choose the best answer (1 Mark)

1. Which of the following refers to the visibility of variables in one part of a program to another part of the same program?
(A) **Scope** (B) Memory (C) Address (D) Accessibility
2. The process of binding a variable name with an object is called
(A) Scope (B) **Mapping** (C) late binding (D) early binding
3. Which of the following is used in programming languages to map the variable and object?
(A) :: **(B) :=** (C) = (D) ==
4. Containers for mapping names of variables to objects is called
(A) Scope (B) Mapping (C) Binding (D) **Namespaces**
5. Which scope refers to variables defied in current function?
(A) Local Scope (B) Global scope (C) Module scope (D) Function Scope
6. The process of subdividing a computer program into separate sub-programs is called
(A) Procedural Programming (B) **Modular programming**
(C)Event Driven Programming (D) Object oriented Programming
7. Which of the following security technique that regulates who can use resources in a computing environment?
(A) Password (B) Authentication
(C) **Access control** (D) Certification
8. Which of the following members of a class can be handled only from within the class?
(A) Public members (B) Protected members
(C) Secured members (D) **Private members**
9. Which members are accessible from outside the class?
(A) **Public members** (B) Protected members
(C) Secured members (D) Private members

10. The members that are accessible from within the class and are also available to its subclasses is called
(A) Public members (B) **Protected members**
(C) Secured members (D) Private members

CHAPTER - 4 - ALGORITHMIC STRATEGIES

Part – II

Answer the following questions (2 Marks)

1. What is an Algorithm?

- An algorithm is a finite set of instructions to accomplish a particular task.
- It is a step-by-step procedure for solving a given problem.

2. Define Pseudo code.

- An informal high-level description of the operating principle of a computer program or other algorithm
- It is an implementation of an algorithm in the form of annotations and informative text written in plain English

3. Who is an Algorist?

- An Algorist is a person skilled in designing of an algorithm.

4. What is sorting?

- Arranging the data in ascending or descending order is called as sorting
- Example: Bubble sort, Selection sort, Insertion sort

5. What is searching? Write its types.

- It is the process of finding a particular data in a collection of data
  Types:
  1. Linear Search or Sequential search
  2. Binary Search

Part - III

Answer the following questions (3 Marks)

1. List the characteristics of an algorithm.

1. Input
2. Output
3. Finiteness
4. Definiteness
5. Effectiveness
6. Correctness
7. Simplicity
8. Unambiguous
9. Feasibility
10. Portable
11. Independent

2. Discuss about Algorithmic complexity and its types.

- The complexity of an algorithm f (n) gives the running time and/or the storage space required by the algorithm in terms of n as the size of input data.

Time Complexity:

- The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.

Space Complexity:

- Space complexity of an algorithm is the amount of memory required to run to its completion.
- The space required by an algorithm is equal to the sum of the following two components:
- **A fixed part:**
  - It is defined as the total space required to store certain data and variables for an algorithm.
  - For example, simple variables and constants used in an algorithm.
- **A variable part:**
  - It is defined as the total space required by variables, which sizes depends on the problem and its iteration.
  - For example: recursion used to calculate factorial of a given value n

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

Time Factor:

Time is measured by counting the number of key operations like comparisons in the sorting algorithm.

Space Factor:

Space is measured by the maximum memory space required by the algorithm

4. Write a note on asymptotic notation.

- Asymptotic Notations are languages that use meaningful statements about time and space complexity.
- The following three asymptotic notations are mostly used *to represent time complexity of algorithms*:

(i) Big O:

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

(ii) Big Ω:

Big Omega is the reverse Big O.

If Big O is used to describe the upper bound (worst - case) of an asymptotic function, 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, say that an algorithm has a complexity O (n log n) and Ω (n log n), it's actually has the complexity Θ (n log n), which means the running time of that algorithm always falls in n log n in the 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.

Part - IV

Answer the following questions (5Marks)

1. Explain the characteristics of an algorithm.

1. **Input:** Zero or more quantities to be supplied.

2. **Output:** At least one quantity is produced.
3. **Finiteness:** Algorithms must terminate after finite number of steps.
4. **Definiteness:** All operations should be well defined.
   For Example operations involving division by zero or taking square root for negative number are unacceptable.
5. **Effectiveness:** Every instruction must be carried out effectively.
6. **Correctness:** The algorithms should be error free.
7. **Simplicity:** Easy to implement.
8. **Unambiguous:** Algorithm should be clear and unambiguous.
   Each of its steps and their inputs/outputs should be clear and must lead to only one meaning.
9. **Feasibility:** Should be feasible with the available resources.
10. **Portable:** An algorithm should be generic, independent of any programming language or an operating system able to handle all range of inputs.
11. **Independent:** An algorithm should have step-by-step directions, which should be independent of any programming code.

2. Discuss about Linear search algorithm.

Linear Search

- Linear search also called sequential search is a sequential method for finding a particular value in a list.
- This method checks the search element with each element in sequence until the desired element is found or the list is exhausted.
- In this searching algorithm, list need not be ordered.

Pseudo code

```
1. Traverse the array using for loop
2. In every iteration, compare the target search key 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 the search element not found.
```

Example 1:
Input: values[] = {5, 34, 65, 12, 77, 35}
target = 77
Output: 4

Example 2:
Input: values[] = {101, 392, 1, 54, 32, 22, 90, 93}
target = 200
Output: -1 (not found)

3. What is Binary search? Discuss with example.

- Binary search also called half-interval search algorithm.
- It finds the position of a search element within a sorted array.
- The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.

Pseudo code 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 lef
     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.
```

For Example:

- Let us assume that the search element is 50 and we need to search the location or index of search element 50 using binary search.

Values	10	20	30	40	50	60	70	80	90	100

Index	0	1	2	3	4	5	6	7	8	9

- First, we find index of middle element of the array by using this formula :
- mid = low + (high - low) / 2
- Here it is, 0 + (9 - 0) / 2 = 4 (fractional part ignored). So, 4 is the mid value of the array.
- 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 match with search element.
- The search element 50 is found at location or index 4.

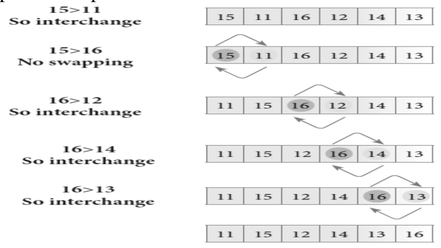
4. Explain the Bubble sort algorithm with example.

- Bubble sort is a simple sorting algorithm.
- The algorithm starts at the beginning of the list of values stored in an array.
- It compares each pair of adjacent elements and swaps them if they are in the unsorted order.
- This comparison and passed to be continued until no swaps are needed, which indicates that the list of values stored in an array is sorted.
- The algorithm is a comparison sort, is named for the way smaller elements "bubble" to the top of the list.
- It is too slow and less efficient when compared to insertion sort and other sorting methods.

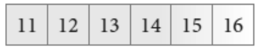
Pseudo code

1. Start with the fist element i.e., index = 0, compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next or right side of the 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} Below, we have a pictorial representation of how bubble sort will sort the given array.



- The above pictorial example is for iteration-1. Similarly, remaining iteration can be done. The final iteration will give the sorted array.
- At the end of all the iterations we will get the sorted values in an array as given below:



11	12	13	14	15	16

5. Explain the concept of Dynamic programming with suitable example.

- 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.

<u>Steps to do Dynamic programming</u>:

• 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.

<u>Fibonacci Iterative Algorithm with Dynamic programming approach</u>

The following example shows a simple Dynamic programming approach for the generation of Fibonacci series.

```
Initialize f0=0, f1 =1
step-1: Print the initial values of Fibonacci f0 and f1
step-2: Calculate fibonacci fi ← f0 + f1
step-3: Assign f0← f1, f1← fi
step-4: Print the next consecutive value of fibonacci fi
step-5: Goto step-2 and repeat until the specified number of terms generated
```

For example if we generate fibonacci series upto 10 digits, the algorithm will generate the series as shown below:

The Fibonacci series is : 0 1 1 2 3 5 8 13 21 34 55

Additional Questions

Algorithm used in real time activities:

- Packing books in school bag
- Finding shortest path to search a place
- Scheduling day-to-day activities
- Preparation for examination, etc.

Algorithmic solution:

An algorithm that yields expected output for a valid input is called an algorithmic solution.

Algorithm Analysis:

An estimation of the time and space complexities of an algorithm for varying input sizes is called algorithm analysis.

Algorithmic strategy:

A way of designing algorithm is called algorithmic strategy.

Which algorithm can be called as best algorithm?

The best algorithm to solve a given problem is one that requires less space in memory and takes less time to execute its instructions to generate output.

Algorithm Vs Program

Algorithm	**Program**
Algorithm helps to solve a given problem logically and it can be contrasted with the program	Program is an expression of algorithm in a programming language
Algorithm can be categorized based on their implementation methods, design techniques etc	Algorithm can be implemented by structured or object oriented programming approach
There is no specific rules for algorithm writing but some guidelines should be followed	Program should be written for the selected language with specific syntax
Algorithm resembles a pseudo code which can be implemented in any language	Program is more specific to a programming language

Analysis of Algorithm and performance evaluation 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.

Factors which decide the efficiency of an Algorithm

Time Factor –

- Time is measured by counting the number of key operations like comparisons in the sorting algorithm.

Space Factor –

- Space is measured by the maximum memory space required by the algorithm.

Efficiency of an algorithm:

- The efficiency of an algorithm is defined as the number of computational resources used by the algorithm.
- An algorithm must be analyzed to determine its resource usage.
- It is measured based on the usage of different resources.
- For maximum efficiency of algorithm we wish to minimize resource usage.
- The important resources such as time and space complexity cannot be compared directly.
- So time and space complexity could be considered for an algorithmic efficiency.

Method for determining Efficiency

The efficiency of an algorithm depends on how efficiently it uses time and memory space.
The execution time depends on the factors such as:

1. Speed of the machine
2. Compiler and other system Software tools
3. Operating System
4. Programming language used
5. Volume of data required

Selection sort

- The selection sort is a simple sorting algorithm that improves on the performance of bubble sort by making only one exchange for every pass through the list.
- This algorithm will fist fid the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue until the entire array is sorted in respective order.
- This algorithm repeatedly selects the next-smallest element and swaps in into the right place for every pass. Hence it is called selection sort.

Pseudo code

1. Start from the fist element i.e., index-0, we search the smallest element in the array, and
replace it with the element in the first position.
2. Now we move on to the second element position, and look for smallest element present in the sub-array, from starting index to till the last index of sub - array.
3. Now replace the second smallest identified in step-2 at the second position in the or original array, or also called fist position in the sub array.
4. This is repeated, until the array is completely sorted.

Insertion sort

- Insertion sort is a simple sorting algorithm. It works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list.
- This algorithm builds the final sorted array at the end.
- This algorithm uses n-1 number of passes to get the final sorted list

Pseudo for Insertion sort

```
Step 1 − If it is the first element, it is already sorted.
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted
```

Choose the best answer: (1 Marks)

1. The word comes from the name of a Persian mathematician Abu Ja’far Mohammed ibn-i Musa al Khowarizmi is called?

(A) Flowchart (B) Flow **(C) Algorithm** (D) Syntax

2. From the following sorting algorithms which algorithm needs the minimum number of swaps?

(A) Bubble sort (B) Quick sort (C) Merge sort **(D) Selection sort**

3. Two main measures for the efficiency of an algorithm are

(A) Processor and memory (B) Complexity and capacity

(C) Time and space (D) Data and space

4. The complexity of linear search algorithm is

(A) O(n) (B) O(log n) (C) O(n2) (D) O(n log n)

5. From the following sorting algorithms which has the lowest worst case complexity?

(A) Bubble sort (B) Quick sort **(C) Merge sort** (D) Selection sort

6. Which of the following is not a stable sorting algorithm?

(A) Insertion sort **(B) Selection sort** (C) Bubble sort (D) Merge sort

7. Time complexity of bubble sort in best case is

(A) θ (n) (B) θ (n log n) (C) θ (n2) (D) θ (n (log n) 2)

8. The Θ notation in asymptotic evaluation represents

(A) Base case **(B) Average case** (C) Worst case (D) NULL case

9. If a problem can be broken into sub problems which are reused several times, the problem possesses which property?

(A) Overlapping sub problems (B) Optimal substructure (C) Memoization (D) Greedy

10. In dynamic programming, the technique of storing the previously calculated values is called ?

(A) Saving value property (B) Storing value property **(C) Memoization** (D) Mapping

CHAPTER – 5 - PYTHON - VARIABLES AND OPERATORS

PART - II

1. What are the different modes that can be used to test Python Program?
- There are two different modes
- Interactive mode
- Script mode

2. Write short notes 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.
- Whitespace separation is necessary between tokens, identifiers or keywords.

3. What are the different operators that can be used in Python?
- There are 5 different types Operators
  1. Arithmetic Operators
  2. Relational Operators
  3. Logical Operators
  4. Assignment Operators
  5. Conditional Operator

4. What is a literal? Explain the types of literals?
- Literal is a raw data given in a variable or constant.
- There are various types of literals.
  1) Numeric Literals
     - Integer
     - Float
     - Complex
  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.
- Example: 12.34E04, 24.65e-04

PART - III

1. Write short notes on Arithmetic operator with examples.
- An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them.

Operators – Operations	Examples	Result
Let a = 10, b = 3		
+ (Addition)	>>> a + b	13
- (Subtraction)	>>>a – b	7
* (Multiplication)	>>> a*b	30
/ (Divisioin)	>>> a / b	3.333
% (Modulus)	>>> a % 3	1
** (Exponent)	>>> a ** 2	100
// (Floor Division)	>>> a//3	3

2. What are the assignment operators that can be used in Python?
- = is a simple assignment operator to assign values to variable.
- There are various compound operators like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	Description	Example
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b="Computer"
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5

*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2
%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
**=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3 # x = x//3

3. Explain Ternary operator with examples.

- Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false.
- It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.
- The Syntax conditional operator is,
  Variable Name = [on_true] if [Test expression] else [on_false]
- **Example :**
  min= 50 if 49<50 else 70

4. Write short notes on Escape sequences with examples.

- The backslash **"\"** is called the **"escape"** character.
- It is used in representing certain whitespace characters: **"\t"** is a tab, **"\n"** is a newline, and **"\r"** is a carriage return.
- For example to print the message "It's raining", the Python command is
  >>> print ("It\'s rainning")
  It's rainning

5. What are string literals? Explain.

- **String literal** is a sequence of characters surrounded by quotes.
- It supports single, double and triple quotes for a string.
- A character literal is a single character surrounded by single or double quotes.
- The value with triple-quote "' '" is used to give multi-line string literal.
- **Example:**

```
strings = "This is Python"
char = "C"
multiline_str = '''This is a multiline string with more than one line code.'''
```

PART - III

1. Describe in detail the procedure Script mode programming.

- Choose **File → New File** or press **Ctrl + N** in Python shell window.
- An **untitled** blank script text editor will be displayed on screen
- Type the following code in Script editor
- a =100
- b = 350
- c = a+b
- print ("The Sum=", c)
- Choose **File → Save** or Press **Ctrl + S**
- In the **Save As** dialog box, select the location where you want to save your Python code, and type the file name in **File Name** box.
- Python files are by default saved with extension **.py.**
- Thus, while creating Python scripts using Python Script editor, no need to specify the file extension.
- Finally, click **Save** button to save your Python script.
- Choose **Run → Run Module** or Press **F5**

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

- A program needs to interact with the user to accomplish the desired task; this can be achieved using **Input-Output functions**.

input() function :

- In Python, **input( )** function is used to accept data as input at run time.
- The syntax for **input()** function is,
- Variable = input (“prompt string”)
- Example :

```
>>> city=input (“Enter Your City: ”)
Enter Your City: Madurai
>>> print (“I am from “, city)
I am from Madurai
```

The print() function :

- In Python, the **print()** function is used to display result on the screen.
- The syntax for **print()** is as follows:
- **Example :**
- print (“string to be displayed as output ”)
- print (variable)
- print (“String to be displayed as output ”, variable)
- print (“String1 ”, variable, “String 2”, variable, “String 3” ……)
- The **print** ( ) evaluates the expression before printing it on the monitor.
- The print () displays an entire statement which is specified within print ( ).
- **Comma (,)** is used as a separator in **print ( )** to print more than one item.
- **Example:**

```
>>> print (“Welcome to Python Programming”)
Welcome to Python Programming
>>> x = 5
>>> y = 6
>>> z = x + y
>>> print (z)
11
>>> print (“The sum = ”, z)
The sum = 11
>>> print (“The sum of ”, x, “ and ”, y, “ is ”, z)
The sum of 5 and 6 is 11
```

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.
- An identifier must start with an alphabet (A..Z or a..z) or underscore (_).
- Identifiers may contain digits (0 .. 9)
- Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.
- Identifiers must not be a **python** keyword.
- Python does not allow punctuation character such as %,$, @ etc., within identifiers.
- **Example of valid identifiers :** Sum, total_marks, regno, num1
- **Example of invalid identifiers :** 12Name, name$, total-mark, continue

2) Keywords

- **Key**words are special words used by Python interpreter to recognize the structure of program.
- As these words have specific meaning for interpreter, they cannot be used for any other purpose.
- Example :

false	class	finally	is	return
none	continue	for	lambda	try

3) Operators

- Operators are special symbols which represent computations, conditional matching etc.
- Value and variables when used with operator are known as **operands**.
- Operators are categorized as
    - Arithmetic Operators
    - Relational Operators
    - Logical Operators
    - Assignment Operators
    - Ternary Operator

(i) Arithmetic operators

- An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them.

Operators – Operations	**Examples**	**Result**
Let a = 100, b = 10		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30	3

(ii) Relational or Comparative operators

- A Relational operator is also called as **Comparative** operator which checks the relationship between two operands.
- If the relation is true, it returns **True**; otherwise it returns **False**.

Operator - Operation	**Examples**	**Result**
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

(iii) Logical operators

- Logical operators are used to perform logical operations on the given relational expressions.
- There are three logical operators they are **and, or** and **not**.

Operator	**Example**	**Result**
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

(iv) Assignment operators

- In Python, = is a simple assignment operator to assign values to variable.
- There are various compound operators in Python like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	**Example**
=	>>> x=10 >>> b="Computer"
+=	>>> x+=20 # x=x+20
-=	>>> x-=5 # x=x-5
*=	>>> x*=5 # x=x*5
/=	>>> x/=2 # x=x/2
%=	>>> x%=3 # x=x%3
**=	>>> x**=2 # x=x**2
//=	>>> x//=3

(v) Conditional operator

- Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false.
- It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.
- The Syntax conditional operator is,
  Variable Name = [on_true] if [Test expression] else [on_false]
- **Example :**
  min= 50 if 49<50 else 70
  min= 50 if 49>50 else 70

4) Delimiters

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

(	)	[	]	{	}
,	:	.	‘	=	;
+=	-=	*=	/=	//=	%=
&=	|=	^=	>>=	<<=	**=

5) Literals.

- Literal is a raw data given in a variable or constant. In Python, there are various types of literals.
  1) Numeric
     1. Integer
     2. Float
     3. Complex
  2) String
  3) Boolean

<u>EXTRA QUESTION ANSWER:</u>

1. What are the Key features of Python?
   - It is a general purpose programming language which can be used for both scientific and non-scientific programming.
   - It is a platform independent programming language.
   - The programs written in Python are easily readable and understandable.
2. What are the two modes available in Python?
   - In Python, programs can be written in two ways namely **Interactive mode** and **Script mode.**
   - The Interactive mode allows us to write codes in Python command prompt **(>>>).**
   - The interactive mode can also be used as a **simple calculator.**
   - Where as in script mode programs can be written and stored as separate file with the extension **.py** and executed.
   - Script mode is used to create and edit python source file.

5. Write short notes on comments in Python.
   - In Python, comments begin with hash symbol (**#**).
   - The lines that begins with **#** are considered as comments and ignored by the Python interpreter. Comments may be single line or no multi-lines.
   - Example :
     # It is Single line Comment
     ‘“It is multiline comment
     which contains more than one line ”’

6. What are the uses of indentation in Python?
   - Python uses whitespace such as **spaces** and **tabs** to define program blocks whereas other languages like C, C++, java use curly braces { } to indicate blocks of codes for class, functions or body of the loops and block of selection command.
   - The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be indented with same amount spaces.

1. What are tokens? What are its types?
   - 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.
   - Whitespace separation is necessary between tokens, identifiers or keywords.

2. What are identifiers?
   - An Identifier is a name used to identify a variable, function, class, module or object.
   - An identifier must start with an alphabet (A..Z or a..z) or underscore (_).
   - Identifiers may contain digits (0 .. 9)
   - Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.
   - Identifiers must not be a **python** keyword.
   - Python does not allow punctuation character such as %,$, @ etc., within identifiers.
   - **Example of valid identifiers :** Sum, total_marks, regno, num1
   - **Example of invalid identifiers :** 12Name, name$, total-mark, continue

3. What are Keywords?
   - **Key**words are special words used by Python interpreter to recognize the structure of program.
   - As these words have specific meaning for interpreter, they cannot be used for any other purpose.
   - Example :

false	class	finally	is	return
none	continue	for	lambda	try

4. What are operators? What are its types? Explain with example.
   - Operators are special symbols which represent computations, conditional matching etc.
   - Value and variables when used with operator are known as **operands**.
   - Operators are categorized as
     1. Arithmetic Operators
     2. Relational Operators
     3. Logical Operators
     4. Assignment Operators

(i) Arithmetic operators

- An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them.

Operators – Operations	**Examples**	**Result**
Let a = 100, b = 10		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30	3

(ii) Relational or Comparative operators

- A Relational operator is also called as **Comparative** operator which checks the relationship between two operands.
- If the relation is true, it returns **True**; otherwise it returns **False**.
- Python supports following relational operators

Operator - Operation	**Examples**	**Result**
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

(iii) Logical operators

- Logical operators are used to perform logical operations on the given relational expressions.
- There are three logical operators they are **and, or** and **not**.

Operator	**Example**	**Result**
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

(iv) Assignment operators

- In Python, = is a simple assignment operator to assign values to variable.
- Let **a** = 5 and **b** = 10 assigns the value 5 to **a** and 10 to **b** these two assignment statement can also be given as **a,b=5,10** that assigns the value 5 and 10 on the right to the variables a and b respectively.
- There are various compound operators in Python like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	**Description**	**Example**
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b="Computer"
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5
*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2
%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
**=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3

(v) Conditional operator

- Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false.
- It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.
- The Syntax conditional operator is,
  Variable Name = [on_true] if [Test expression] else [on_false]
- **Example :**
  min= 50 if 49<50 else 70
  min= 50 if 49>50 else 70

5. What are delimeters?

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

(	)	[	]	{	}
,	:	.	‘	=	;
+=	-=	*=	/=	//=	%=
&=	|=	^=	>>=	<<=	**=

6. What are literals?

- Literal is a raw data given in a variable or constant. In Python, there are various types of literals.
  1) Numeric
  2) String
  3) Boolean
- **Numeric Literals** consists of digits and are immutable (unchangeable).
- Numeric literals can belong to 3 different numerical types
  4. Integer
  5. Float
  6. Complex

  1. Integer - Example:
  a = 0b1010 *#Binary Literals*
  b = 100 *#Decimal Literal*
  c = 0o310 *#Octal Literal*
  d = 0x12c *#Hexadecimal Literal*

  2. Float - Example:
  float_1 = 10.5
  float_2 = 1.5e2

  3. Complex – Example:
  x = 1 + 3.14 j

- In Python a **string literal** is a sequence of characters surrounded by quotes.
- Python supports single, double and triple quotes for a string.
- A character literal is a single character surrounded by single or double quotes.
- The value with triple-quote "' '" is used to give multi-line string literal.
- **Example:**
  strings = "This is Python"

char = "C"
multiline_str = '''This is a multiline string with more than one line code.'''

- A **Boolean literal** can have any of the two values: True or False.
  boolean_1 = True
  boolean_2 = False

7. Write short notes on escape sequences.

- In Python strings, the backslash **"\"** is a special character, also called the **"escape"** character.
- It is used in representing certain whitespace characters: **"\t"** is a tab, **"\n"** is a newline, and **"\r"** is a carriage return.
- For example to print the message "It's raining", the Python command is
  >>> print ("It\'s rainning")
  It's rainning
- Python supports the following escape sequence characters.

Escape sequence character	**Description**	**Example**	**Output**
\\	Backslash	>>> print("\\test")	\test
\’	Single-quote	>>> print("Doesn\'t")	Doesn't
\”	Double-quote	>>> print("\"Python\"")	"Python"
\n	New line	print("Python","\n","Lang..")	Python Lang..
\t	Tab	print("Python","\t","Lang..")	Python Lang

8. Write short note on data types in Python.

- All data values in Python are objects and each object or value has type.
- Python has Built-in or Fundamental data types such as
  1. Number
  2. String
  3. Boolean
  4. tuples
  5. lists and
  6. Dictionaries.

1. **<u>Number Data type:</u>**
   - The built-in number objects in Python supports integers, floating point numbers and complex numbers.
   - Integer Data can be decimal, octal or hexadecimal.
   - **Example :**

```
102, 4567, 567          # Decimal integers
O102, o676, O432        # Octal integers
OX102, oX676, OX432     # Hexadecimal integers
34L, 523L               # Long decimal integers
123.34, 456.23, 156.23  # Floating point data
12.E04,  24.e04         # Exponent data
1 + 3.14 j              # Complex data type
```

2. **String Data type**
   - String data can be enclosed with single quote or double quote or triple quote.
   - **Example :**
     Char_data = ‘A’
     String_data= "Computer Science"
     Multiline_data= ‘‘‘String data can be enclosed with single quote or double quote or triple quote.’’’

3. **Boolean Data type**
   - A Boolean data can have any of the two values: True or False.

   Example :
   Bool_var1=True
   Bool_var2=False

Choose the best answer (1 Marks)

1. Who developed Python?
A) Ritche **B) Guido Van Rossum** C) Bill Gates D) Sunder Pitchai

2. The Python prompt indicates that Interpreter is ready to accept instruction.
A) >>> B) <<< C) # D) <<

3. Which of the following shortcut is used to create new Python Program ?
A) Ctrl + C B) Ctrl + F C) Ctrl + B **D) Ctrl + N**

4. Which of the following character is used to give comments in Python Program ?
A) # B) & C) @ D) $

5. This symbol is used to print more than one item on a single line.
A) Semicolon(;) B) Dollor($) **C) comma(,)** D) Colon(:)

6. Which of the following is not a token?
A) Interpreter B) Identifiers C) Keyword D) Operators

7. Which of the following is not a Keyword in Python?
A) break B) while C) continue **D) operators**

8. Which operator is also called as Comparative operator?
A) Arithmetic **B) Relational** C) Logical D) Assignment

9. Which of the following is not Logical operator?
A) and B) or C) not **D) Assignment**

10. Which operator is also called as Conditional operator?
A) Ternary B) Relational C) Logical D) Assignment

11. Value and variables used in operator are known as
A) Keywords **B) Operands** C) Identifiers D) Functions

12. In python, Which of the following is a simple assignment operator?
A) == B) # C) != **D) =**

13. Which of the following is an incorrect statement?
i) Python identifiers must start with an alphabet
ii) Python identifiers may contain digits
iii) Python identifiers are not case sensitive

iv) Python allows %, $, @ characters with in identifiers

A) i,ii B) ii, iii **C) iii, iv** D) ii, iv

14. Which data can be decimal, octal or hexadecimal?

A) Character **B) Integer** C) Escape sequence D) Symbols

15. Python uses the symbols and symbol combinations as ________ in expressions

A) literals B) keywords C) identifiers **D) delimiters**