VIRUDHUNAGAR DISTRICT
SECOND REVISION EXAM, FEBRUARY 2025
STANDARD 12
COMPUTER SCIENCE

PART – I

I. Answer all and choose the best answer:

1. d. pure function
2. a. constructor
3. a. public member
4. b. Time and Space
5. a. >>>
6. a. 3
7. a. Lambda
8. d. Either (a) or (b)
9. c. len
10. b. .
11. c. relation
12. a. SELECT
13. a. line terminator
14. c. Python file name
15. a. Sqlite_master

PART – II

II. Answer ANY 6 of the following and question no. 21 is compulsory: 6 X 2 = 12

16. What do you mean by Namespaces?

Namespaces:

- Namespaces are containers for mapping names of variables to objects (name : = object).

Example:

a:=5

- Here the variable „a" is mapped to the value „5".

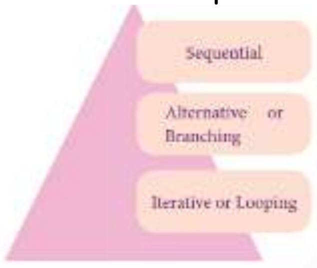
17. What is Insertion sort?

- 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.

18. List the control structures in Python.

Control structures in Python:

- Three important control structures are,



19. Write short notes on Tokens.

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.

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

- Python stops calling recursive function after 1000 calls by default.
- So, It also allows you to change the limit using sys.setrecursionlimit (limit_value).

Example:

```
import sys
sys.setrecursionlimit(3000)
def fact(n):
        if n == 0:
                return 1
        else:
                return n * fact(n-1)
print(fact (2000))
```

21. **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
```

22. List some examples of RDBMS.

- SQL Server
- Oracle
- MySQL
- MariaDB
- SQLite

23. What is use of next() function?

next() function:

- **"next()"command** is used to avoid or skip the first row or row heading.
- **Example:** While sorting the row heading is also get sorted, to avoid that the first is skipped using next().
- Then the list is sorted and displayed.

24. Differentiate compiler and interpreter.

Compiler	Interpreter
Compiler generates an Intermediate Code.	Interpreter generates Machine Code.
Compiler reads entire program for compilation.	Interpreter reads single statement at a time for interpretation.
Error deduction is difficult	Error deduction is easy
Comparatively faster	Slower
Example: gcc, g++, Borland TurboC	**Example:** Python, Basic, Java

PART – III

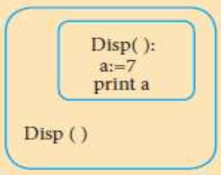
III. Answer ANY 6 of the following and question no. 33 is compulsory: 6 X 3 = 18

25. Define local scope with example.

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

- On execution of the above code the variable **a** displays the value 7, because it is defined and available in the local scope.

26. Discuss about Algorithmic complexity and its types.

ALGORITHMIC COMPLEXITY:

- 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.

TYPES OF COMPLEXITY:

1. Time Complexity

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

2. 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 **fixed part and variable part**.

27. What are string literals? Explain.

String literals:

- 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."'
print (strings)
print (char)
print (multiline_str)
```

Output:

```
This is Python
C
This is a multiline string with more than one line code.
```

28. Write the syntax of while loop.

Syntax:

```
while <condition>:
        statements block 1
[else:
        statements block2]
```

29. List out the set operations supported by python.

Set Operations:

(i) Union: It includes all elements from two or more sets.

(ii) Intersection: It includes the common elements in two sets.

(iii) Difference: It includes all elements that are in first set (say set A) but not in the second set (say set B).

iv) Symmetric difference: It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.

30. Write a python program to find total and average marks of 3 subjects using class

```
 a = int(input("Enter the marks of first subject: "))
b = int(input("Enter the marks of second subject: "))
c = int(input("Enter the marks of third subject: "))
total = a+b+c
avg = total/3
print("Total marks: ",total)
print("Average marks: ",avg)
```

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

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

Syntax:

```
SAVEPOINT savepoint_name;
```

Example: SAVEPOINT A;

32. What are the applications of scripting language?

Applications of scripting language:

- To automate certain tasks in a program
- Extracting information from a data set

- ☯ Less code intensive as compared to traditional programming language
- ☯ can bring new functions to applications and glue complex systems together

33. . 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**

PYTHON SCRIPT:

```
import sqlite3
connection = sqlite3.connect("organization.db")
c=conn.execute("SELECT * FROM Employee GROUP BY Dept")
for row in c:
print(row)
conn.close()
```

PART - IV

34. . **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.

Two types of parameter passing are,
1. Parameter Without Type
2. Parameter With Type

1. Parameter Without Type:

- ☯ Lets see an example of a function definition of Parameter Without Type:

(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)

- ☯ In the above function definition **variable „ b"** is the **parameter** and the **value** passed **to** the variable ***„b"*** is the **argument**.
- ☯ The precondition ***(requires)*** and postcondition ***(returns)*** of the function is given.
- ☯ We have not mentioned any types: ***(data types)***. This is called parameter without type.
- ☯ In the above function definition the expression has type ***„int"***, so the function's return type also be ***„int"*** *by implicit.*

OR

Explain the concept of Dynamic programming with suitable example.

Concept of Dynamic programming:

- Dynamic programming is 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 (i.e) the problem can be divided into smaller sub-problems.
- Results of the sub-problems can be re-used to complete the process.
- Dynamic programming approaches are used to find the solution in optimized way.

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.

Fibonacci Iterative Algorithm with Dynamic Programming Approach

- 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 fibanocci fib ← f0 + f1
- step-3: Assign f0← f1, f1← fib
- step-4: Print the next consecutive value of fibanocci fib
- 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

35. Tabulate with examples the various types of operators used in python.

- Operators are special symbols which represent computations, conditional matching etc.
- The value of an operator used is called operands.

Operators are categorized as

- ✓ Arithmetic
- ✓ Relational
- ✓ Logical
- ✓ Assignment etc.

Arithmetic operators:

- An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them.
- They are used for simple arithmetic. Most computer languages contain a set of such operators that can be used within equations to perform different types of sequential calculations.

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.

Logical operators:

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

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.

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.

OR

Write a Python program to display Fibonacci series 0,1,1,2,3,5, …(upto n terms)

```
n_terms = int(input ("How many terms the user wants to print? "))
n_1 = 0
n_2 = 1
count = 0
 if n_terms <= 0:
        print ("Please enter a positive integer, the given number is not valid")
```

```
elif n_terms == 1:
  print ("The Fibonacci sequence of the numbers up to", n_terms, ": ")
    print(n_1)
else:
   print ("The fibonacci sequence of the numbers is:")
   while count < n_terms:
      print(n_1)
    nth = n_1 + n_2
        n_1 = n_2
       n_2 = nth
       count += 1
```

36. Explain the 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.

TYPES OF VARIABLE SCOPE:

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

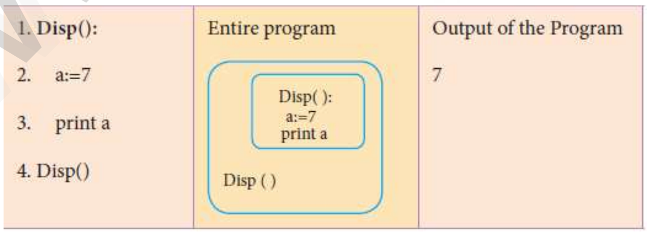
LEGB RULE:

❖ 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).

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

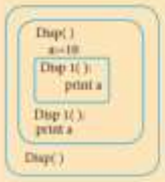
2. a:=7
3. print a
4. Disp()
	Disp( ):
	a:=7
	print a
	Disp ( )
		7

❖ On execution of the above code the variable **a** displays the value 7, because it is defined and available in the local scope.

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.

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()		

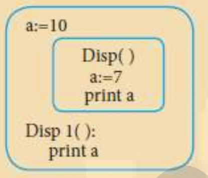
- ❖ In the above example Disp1() is defined within Disp(). The variable „a" defined in Disp() can be even used by Disp1() because it is also a member of Disp().

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

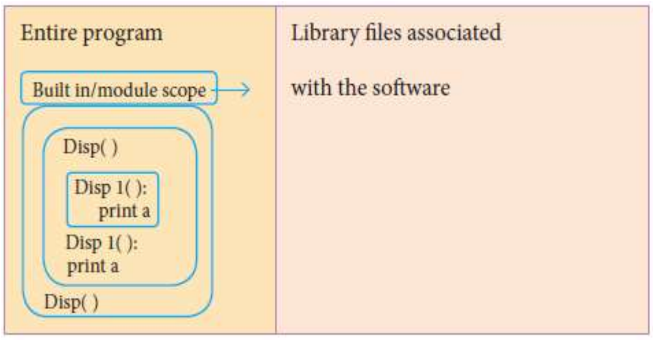
2. Disp():	 a:=10
	Disp( )
	a:=7
	print a
	Disp 1( ):
	print a
		7

3. a:=7		10
4. print a		
5. Disp()		
6. print a		

- ❖ On execution of the above code the variable **a** which is defined inside the function displays the value 7 for the function call Disp() and then it displays 10, because **a** is defined in global scope.

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.



OR

Write a Python program to count the occurrence of each word in a given string.

```
def count(s, c):
c1=0
for i in s:
if i == c:
c1+=1
return c1
str1=input ("Enter a String: ")
ch=input ("Enter a character to be searched: ")
cnt=count (str1, ch)
print ("The given character {} is occurs {} times in the given string".format(ch,cnt))
```

Out Put

```
Enter a String: Software Engineering
Enter a character to be searched: e
The given character e is occurs 3 times in the given string
```

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

. 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 (∪)
- INTERSECTION (∩)
- DIFFERENCE (−)

- CARTESIAN PRODUCT (X)

SELECT (symbol : σ)

General form σ_c (R) with a relation R and a condition C on the attributes of R.

- 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.

Example:

σ_{course} = "Big Data" **(STUDENT)**

PROJECT (symbol : Π)

- The projection eliminates all attributes of the input relation but those mentioned in the projection list.
- The projection method defines a relation that contains a vertical subset of Relation.

Example:

Π_{course} (STUDENT)

UNION (Symbol :∪) A ∪ B

- It includes all tuples that are in tables A or in B.
- It also eliminates duplicates.
- Set A Union Set B would be expressed as A ∪ B

SET DIFFERENCE (Symbol : -)

- The result of A – B, is a relation which 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.

INTERSECTION (symbol : ∩) A ∩ 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.

PRODUCT OR CARTESIAN PRODUCT (Symbol : X)

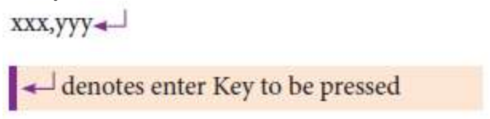
- Cross product is a way of combining two relations.
- The resulting relation contains, both relations being combined.
- This type of operation is helpful to merge columns from two relations.
- A x B means A times B, where the relation A and B have different attributes.

OR

Write the rules to be followed to format the data in a CSV file.

1. Each record (row of data) is to be located on a separate line, delimited by a line break by pressing enter key.

For example:



xxx,yyy↵

↵ denotes enter Key to be pressed

2. The last record in the file may or may not have an ending line break.

For example:

```
ppp, qqq ↵
yyy, xxx
```

3.There may be an optional header line appearing as the first line of the file with the same format as normal record lines.

- The header will contain names corresponding to the fields in the file and should contain the same number of fields as the records in the rest of the file.

For example: field_name1,field_name2,field_name3

```
aaa,bbb,ccc ↵
zzz,yyy,xxx CRLF( Carriage Return and Line Feed)
```

4.Within the header and each record, there may be one or more fields, separated by commas.

- Spaces are considered part of a field and should not be ignored.
- The last field in the record must not be followed by a comma.

For example: Red , Blue

5. Each field may or may not be enclosed in double quotes.

- If fields are not enclosed with double quotes, then double quotes may not appear inside the fields.

For example:

```
"Red","Blue","Green"↵     #Field data with doule quotes
Black,White,Yellow        #Field data without doule quotes
```

6.Fields containing line breaks (CRLF), double quotes, and commas should be enclosed in doublequotes.

For example:

```
Red, “,”, Blue CRLF    # comma itself is a field value.so it is enclosed with double  quotes
Red, Blue , Green
```

7. If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be preceded with another double quote.

For example:

```
“Red, ” “Blue”, “Green”,    # since double quotes is a field value it is enclosed with another double  quotes
,, White
```

38. Write the syntax for getopt() and explain its argumentss and return values.

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, forinput 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)
[]
```

OR

Explain in detail the types of pyplots using Matplotlib.

Line Chart:

- A Line Chart or Line Graph is a type of chart which displays information as a series of data points called „markers" connected by straight line segments.
- A Line Chart is often used to visualize a trend in data over intervals of time – a time series – thus the line is often drawn chronologically.

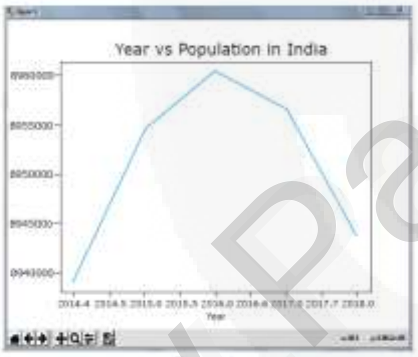
Example:

```
import matplotlib.pyplot as plt
years = [2014, 2015, 2016, 2017, 2018]
total_populations = [8939007, 8954518, 8960387, 8956741, 8943721]
plt.plot (years, total_populations)
plt.title ("Year vs Population in India")
plt.xlabel ("Year")
plt.ylabel ("Total Population")
plt.show()
```

In this program,

Plt.title()	→	specifies title to the graph
Plt.xlabel()	→	specifies label for X-axis
Plt.ylabel()	→	specifies label for Y-axis

Output:



Bar Chart:

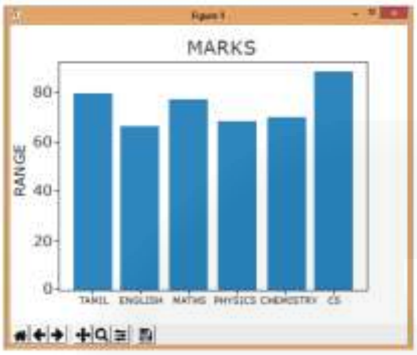
- A BarPlot (or BarChart) is one of the most common type of plot.
- It shows the relationship between a numerical variable and a categorical variable.
- Bar chart represents categorical data with rectangular bars.
- Each bar has a height corresponds to the value it represents.
- The bars can be plotted vertically or horizontally.
- It"s useful when we want to compare a given numeric value on different categories.
- To make a bar chart with Matplotlib, we can use the plt.bar() function

Example:

```
import matplotlib.pyplot as plt
labels = ["TAMIL", "ENGLISH", "MATHS", "PHYSICS", "CHEMISTRY", "CS"]
```

```
usage = [79.8, 67.3, 77.8, 68.4, 70.2, 88.5]
y_positions = range (len(labels))
plt.bar (y_positions, usage)
plt.xticks (y_positions, labels)
plt.ylabel ("RANGE")
plt.title ("MARKS")
plt.show()
```

Output:



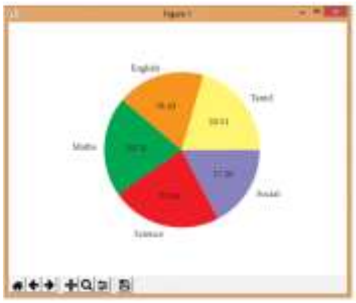
Labels → Specifies labels for the bars.
Usgae → Assign values to the labels specified.
Xticks → Display the tick marks along the x-axis at the values represented. Then specify the label for each tick mark.
Range → Create sequence of numbers.

Pie Chart:

- Pie Chart is probably one of the most common type of chart.
- It is a circular graphic which is divided into slices to illustrate numerical proportion.
- The point of a pie chart is to show the relationship of parts out of a whole.
- To make a Pie Chart with Matplotlib, we can use the plt.pie() function.
- The autopct parameter allows us to display the percentage value using the Python string formatting.

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()
```



****

Mrs. GEETHAMARIMUTHU, M.Sc.,B.Ed.

P.G. Computer Science

VMG RR SRI SARADA SAKTHI MHSS

VIRUDHUNAGAR