SRI KRISHNA ACADEMY

NEET,JEE AND BOARD EXAM COACHING CENTRE
SBM SCHOOL CAMPUS,TRICHY MAIN ROAD,NAMAKKAL
CELL:9965531727-9443231727

+2 FIRST MID TERM - JULY - 2019
TENTATIVE ANSWER KEY

STD: XII **TENTATIVE ANSWER KEY** **MARKS : 50**

COMPUTER SCIENCE

	SECTION - I	
Q. NO	**ANSWER KEY**	**MARKS**
	CHOOSE THE CORRECT ANSWER	1X10=10
1	(A) impure function	
2	(c)return	
3	(a)Pair	
4	(b)=	
5	(b) Asymptotic Notations	
6	(d)Ctrl+N	
7	(c)Run → Run Module	
8	b)pass	
9	d)Lambda	
10	c)round()	
11	Part-A • 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 these subroutines are called as Functions.	2
12	• 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.	2
13	➢ Asymptotic Notations are languages that uses 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 used to describe the lower bound (best-case).	2

	(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.	
14	1)Identifiers, 2) Keywords, 3) Operators, 4) Delimiters and 5) Literals.	**2**

15	**Syntax:**
	while <condition>:
	    statements block 1
	[else:
	    statements block2]
		2

16.	• User-defined functions - Functions defined by the users themselves. • Built-in functions - Functions that are inbuilt with in Python. • Lambda functions - Functions that are anonymous un-named function. • Recursion functions - Functions that calls itself is known as recursive.	**2**
17.	**Output:** 2 4 6 8	**2**
18.		**3**
19.	• Searching is designed to check for an element or retrieve an element from any data structure where it is stored.	**2**

18.

List	**Dictionary**
List is an ordered set of elements.	a dictionary is a data structure that is used for matching one element (Key) with another (Value).
The index values can be used to access a particular element.	in dictionary key represents index. Remember that, key may be a number of a string.
Lists are used to look up a value.	a dictionary is used to take one value and look up another value.

	Types: • Linear (or) sequential search • Binary (or) half interval search	1
20.	• 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. **Example: >>>a+b**	2 1
21.	**Syntax:** *if <condition>:* *statements-block 1* *else:* *statements-block 2*	3
22.	• for loop uses the range() function in the sequence to specify the initial, final and increment values. • range() generates a list of values starting from **start** till **stop-1.** **Syntax of range ( ) function:** • *range (start value, end value, step value)* Ex: for x in range (1, 11):	1 1 1
23.	A variable, with global scope can be used anywhere in the program. It can be created by defining a variable outside the scope of any function/block. **Rules of global Keyword :** The basic rules for ***global*** keyword in Python are: When we define a variable outside a function, it's global by default. You don't have to use global keyword. We use global keyword to read and write a global variable inside a function. Use of global keyword outside a function has no effect Example: c = 1 *# global variable* def add(): print(c) add()	1 1 1

24.	a = int(input("Enter any number :"))
	if a%2==0:
	print (a, " is an even number")
	else:
	print (a, " is an odd number")
	Output 1:
	Enter any number :56
	56 is an even number
	Output 2:
	Enter any number :67
	67 is an odd number
		3

25.	**PART-C** **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**. • This means that 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 behavior of that variable. • Let us see an example **let square x** **return: x * x** • The above function square is a pure function because it will not give different results for same input. ***let i: = 0;*** ***if i <strlen (s) then*** ***-- Do something which doesn't affect s*** ***++i*** • This function reads external memory but does not change it, and the value returned derives from the external memory accessed. **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	$2\frac{1}{2}$ $2\frac{1}{2}$

behave the same every time it's called.

- For example the mathematical function random() will give different outputs for the same function call.

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

- Here the function Random is impure as it is not sure what will be the result when we call the function.

b) **Explain the types of scopes for variable or LEGB rule with example.**

Types of Variable Scope:

There are 4 types of Variable Scope, let's discuss them one by one:

- **Local Scope**

Local scope refers to variables defined in current function. Always, a function will first 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()
```

- **Global Scope**

A variable which is declared outside of all the functions in a program is known as global variable. This means, global variable can be accessed inside or outside of all the functions in a program.

- **Example**

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

Output of the Program

```
 7
10
```

- 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

$2\frac{1}{2}$

scope.

- **Enclosed Scope**
    - ➢ All programming languages permit functions to be nested. A function (method) with in another function is called nested function.
    - ➢ 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.

➢ **Example**

```
1. Disp():
2. a:=10
3. Disp1():
4. print a
5. Disp1()
6. print a
7. Disp()
```

Output of the Program

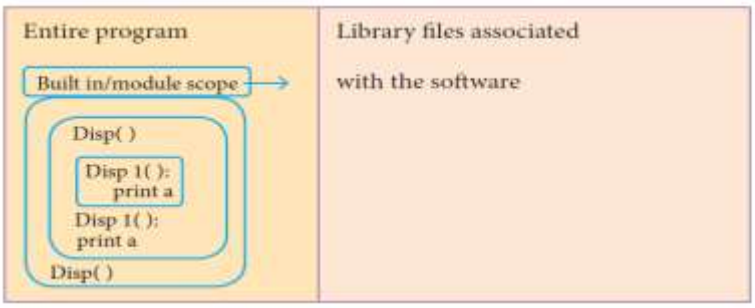
```
10
10
```

In the above example Disp1() is defined with in Disp(). The variable 'a' defined in Disp() can be even used by Disp1() because it is also a member of Disp().

- **Built-in Scope**

➢ Finally, we discuss about the widest 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. They are loaded as soon as the library files are imported to the program.



➢ Normally only Functions or modules come along with the software, as packages. Therefore they will come under Built in scope.

(OR)

LEGB rule:

- Scope also defines the order in which variables have to be mapped to the object in order to obtain the value. Let us take a simple example as shown below:

```
1. x:= 'outer x variable'
2. display():
3. x:= 'inner x variable'
4. print x
5. display()
```

- When the above statements are executed the statement (4) and (5) display the result as

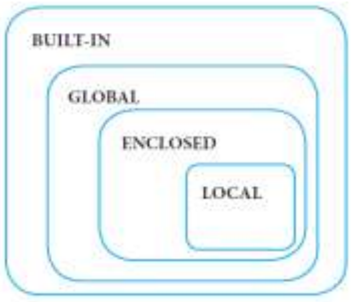
Output

```
outer x variable
inner x variable
```

- Above statements give different outputs because the same variable name x resides in different scopes, one inside the function display() and the other in the upper level.
- The value 'outer x variable' is printed when x is referenced outside the function definition. Whereas when display() gets executed, 'inner x variable' is printed which is the x value inside the function definition.
- From the above example, we can guess that there is a rule followed, in order to decide from which scope a variable has to be picked.
- 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)

$2\frac{1}{2}$



26.	**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 ")
	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 1:
	input( ) with prompt string
	>>> city=input ("Enter Your City: ")
	Enter Your City: Madurai
	>>> print ("I am from ", city)
	I am from Madurai.
	The **input ( )** accepts all data as string or characters but not as numbers. If a numerical value is entered, the input values should be explicitly converted into numeric data type. The **int( )** function is used to convert string data as integer data explicitly
		$2\frac{1}{2}$
		
		$2\frac{1}{2}$

b)	**(b) chr ()** • **Description**: Returns the Unicode character for the given ASCII value. • This function is inverse of ord() function. • **Syntax:** chr (i) • **Example:** c=65 d=43 print (chr (c)) print(chr (d)) **Output:** A + **(c) round()** • **Description:** Returns the nearest integer to its input. 1. First argument (number) is used to specify the value to be rounded 2. Second argument (n digits) is used to specify the number of decimal digits desired after rounding. • **Syntax:** round (number [,n digits])	**1**

- **Example:** 1

```
x= 17.9
y= 22.2
z= -18.3
print ('x value is rounded to', round (x))
print ('y value is rounded to', round (y))
print ('z value is rounded to', round (z))
```

Output:1

```
x value is rounded to 18
y value is rounded to 22
z value is rounded to -18
n1=17.89
print (round (n1,0))
print (round (n1,1))
print (round (n1,2))
```

Output:2

```
18.0
17.9
17.89
```

(d) type()

- **Description:** Returns the type of object for the given single object.
  Note: This function used with single object parameter.
- **Syntax:** type (object)
- **Example:** 1

```
x= 15.2
y= 'a'
s= True
print (type (x))
print (type (y))
print (type (s))
```

Output:

```
<class 'float'>
<class 'str'>
<class 'bool'>
```

(e) pow()

- **Description:** Returns the computation of ab i.e. (a**b) a raised to the power of b. 1
- **Syntax:** pow (a,b)
- **Example:** a= 5

	b= 2 c= 3.0 print (pow (a,b)) print (pow (a,c)) print (pow (a+b,3)) **Output:** 25 125.0 343 **MIN( ):** The MIN() function returns the smallest value of the selected column. **Example** import sqlite3 connection = sqlite3.connect("Organization.db") cursor = connection.cursor() print("Displaying the name of the Highest Average") cursor.execute("SELECT sname,max(AVERAGE) FROM student ") result = cursor.fetchall() print(result) print("Displaying the name of the Least Average") cursor.execute("SELECT sname,min(AVERAGE) FROM student ") result = cursor.fetchall() print(result) **OUTPUT** Displaying the name of the Highest Average [('PRIYA', 98.6)] Displaying the name of the Least Average [('TARUN', 62.3)]	**1**
27.	**for loop** **for** loop is the most comfortable loop. It is also an entry check loop. The condition is checked in the beginning and the body of the loop(statements-block 1) is executed if it is only True otherwise the loop is not executed.	**2**

Syntax:

for counter_variable in sequence:
statements-block 1
[else: # optional block
statements-block 2]

1

The syntax of range() is as follows:

range (start,stop,[step])

Where,

start – refers to the initial value

stop – refers to the final value

step – refers to increment value, this is optional part.

Example:

```
for i in range (2,10,2):
print (i, end=' ')
```

1

Output:

```
2 4 6 8
```

b) **Explain the different types of function with an example.**

Basically, we can divide functions into the following types:

- User-defined functions
- Built-in functions
- Lambda functions
- Recursion functions

1

Functions	Description
User-defined functions	Functions defined by the users themselves.
Built-in functions	Functions that are inbuilt with in Python.
Lambda functions	Functions that are anonymous un-named function.
Recursion functions	Functions that calls itself is known as recursive.

1

Syntax for User defined function

```
def <function_name ([parameter1, parameter2…] )> :
<Block of Statements>
return <expression / None>
```

Block:

- A block is one or more lines of code, grouped together so that they are treated as one big sequence of statements while execution. In Python, statements in a block are written with indentation.

- Usually, a block begins when a line is indented (by four spaces) and all the statements of the block should be at same indent level.

Nested Block:

- A block within a block is called nested block. When the first block statement is indented by a single tab space, the second block of statement is indented by double tab spaces.
- Example of defining a function:

```
Do_Something( ):
value =1          #Assignment Statement
return value      #Return Statement
```

3

Example:

```
def hello():
print ("hello - Python")          return
```

Anonymous Functions:

- Lambda function is mostly used for creating small and one-time anonymous function.
- Lambda functions are mainly used in combination with the functions like filter(), map() and reduce().

Syntax of Anonymous Functions:

lambda [argument(s)] :expression

Example:

```
sum = lambda arg1, arg2: arg1 + arg2
print ('The Sum is :', sum(30,40))
print ('The Sum is :', sum(-30,40))
```

Output:

```
The Sum is : 70
The Sum is : 10
```

- The above lambda function that adds argument **arg1** with argument **arg2** and stores the result in the variable sum. The result is displayed using the print().

Python recursive functions:

- When a function calls itself is known as recursion. Recursion works like loop but sometimes it makes more sense to use recursion than loop. You can convert any loop to recursion.
- A recursive function calls itself. Imagine a process would iterate indefinitely if not stopped by some condition. Such a

process is known as infinite iteration.

Overview of how recursive function works

- Recursive function is called by some external code.
- If the base condition is met then the program gives meaningful output and exits.
- Otherwise, function does some required processing and then calls itself to continue recursion.
- Here is an example of recursive function used to calculate factorial.

```
def fact(n):
if n == 0:
        return 1
else:
        return n * fact (n-1)
print (fact (0))
print (fact (5))
```

Output: 1
120

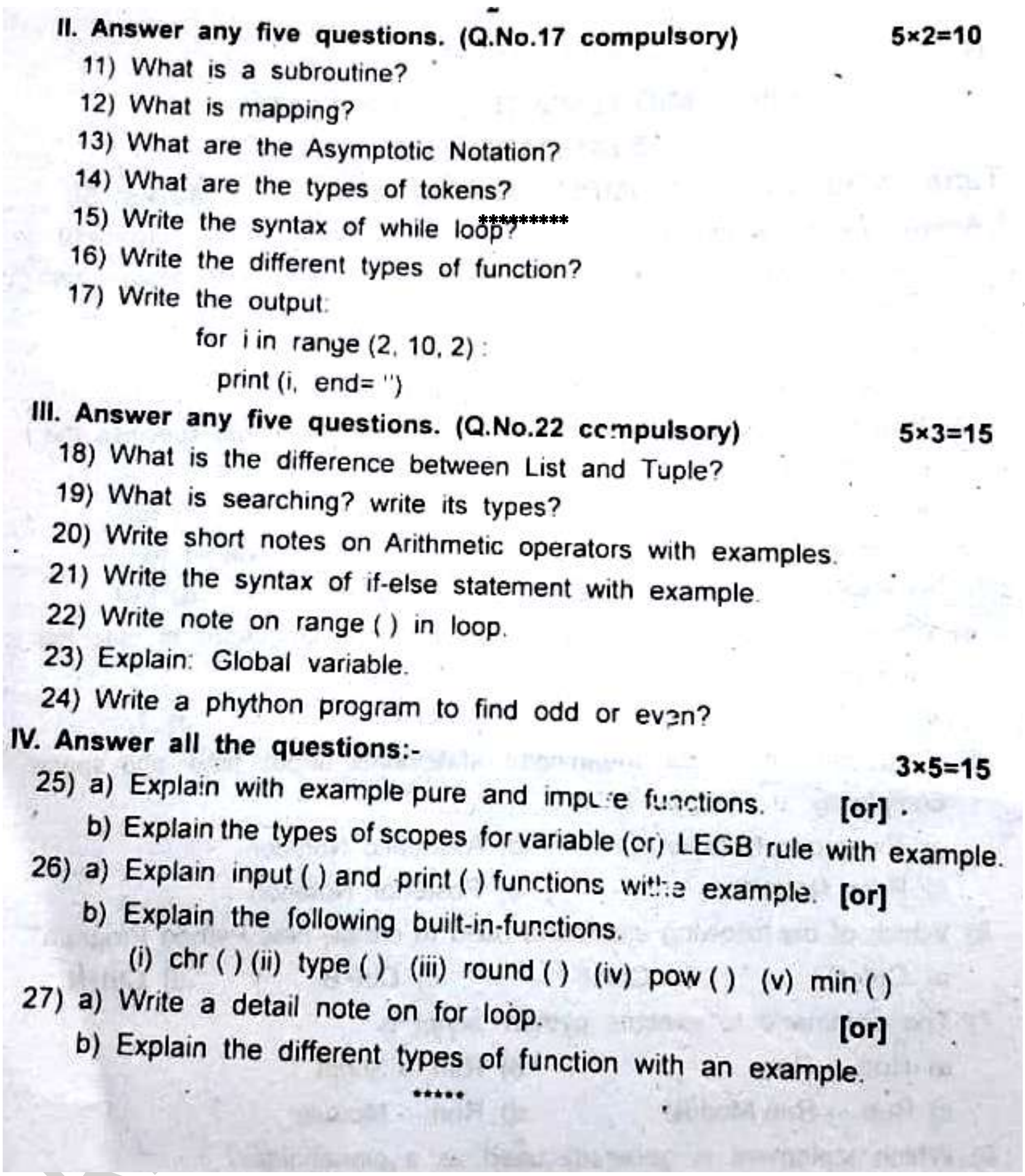
N

FIRST MID TERM TEST - JULY 2019
STANDARD - XII

Time : 1.30 hrs COMPUTER SCIENCE Marks: 50

I. Answer all the questions:- 10×1=10

1) The functions which cause side effects to the arguments passed are called
   a) Impure functions b) Partial functions
   c) Dynamic functions d) Pure functions
2) In a function definition, which of the following keywords specifies the post condition?
   a) let b) requires c) return d) rec
3) Bundling two values together into one can be considered as
   a) Pair b) Triplet c) Single d) List
4) Which of the following is used in programming languages to map the variable and object
   a) [illegible] b) = c) = d) ?
5) Languages that use meaningful statements about time and space complexity are known as
   a) Symptotic Notation b) Asymtotic Notation
   c) Prior Notation d) Posterior Notation
6) Which of the following shortcut is used to create new Python Program?
   a) Ctrl+C b) Ctrl+F c) Ctrl+B d) Ctrl+N
7) The command to execute python script is
   a) Run → Run b) Run → Script
   c) Run → Run Module d) Run → Module
8) Which statement is generally used as a placeholder?
   a) continue b) pass c) break d) gots
9) Which function is called anonymous un-named function?
   a) Recursion b) function c) define d) Lambda
10) Which of the following function returns the nearest integer to its input?
   a) ceil ( ) b) floor ( ) c) round ( ) d) ord ( )



II. Answer any five questions. (Q.No.17 compulsory) **5×2=10**

11) What is a subroutine?
12) What is mapping?
13) What are the Asymptotic Notation?
14) What are the types of tokens?
15) Write the syntax of while loop?
16) Write the different types of function?
17) Write the output:

```
for i in range (2, 10, 2) :
    print (i, end= '')
```

****

III. Answer any five questions. (Q.No.22 compulsory) **5×3=15**

18) What is the difference between List and Tuple?
19) What is searching? write its types?
20) Write short notes on Arithmetic operators with examples.
21) Write the syntax of if-else statement with example.
22) Write note on range ( ) in loop.
23) Explain: Global variable.
24) Write a phython program to find odd or even?

IV. Answer all the questions:- **3×5=15**

25) a) Explain with example pure and impure functions. **[or]**
    b) Explain the types of scopes for variable (or) LEGB rule with example.
26) a) Explain input ( ) and print ( ) functions with example. **[or]**
    b) Explain the following built-in-functions.
       (i) chr ( ) (ii) type ( ) (iii) round ( ) (iv) pow ( ) (v) min ( )
27) a) Write a detail note on for loop. **[or]**
    b) Explain the different types of function with an example.

*****

****