VELAMMAL MAT. HR. SEC. SCHOOL, SURAPET, CHENNAI – 66
XII - COMPUTER SCIENCE
QEX REVISION

CHAPTER – 6 – CONTROL STRUCTURES

PART-II

1. List the control structures in Python.
- There are three important control structures
    1. Sequential
    2. Alternative or Branching
    3. Iterative or Looping

2. Write note on break statement.
- The **break** statement terminates the loop containing it.
- Control flows to the statement immediately after the body of the loop.
- **Syntax:** *break*
- **EXAMPLE:**

```
for word in "Jump Statement":
    if word = = "e":
        break
    print (word, end= ' ')
```

- **Output:**

```
Jump Stat
```

3. Write the syntax of if … else statement.
- **Syntax:**

```
if <condition>:
    statements-block1
```

4. Define control structures?
- A program statement that causes a jump of control from one part of the program to another is called control structure or control statement.

5. Write note on range () in loop
- In Python, **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.**
- **The syntax of range():**
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 :

range (1,30,1)	*will start the range of values from 1 and end at 29*
range (2,30,2)	*will start the range of values from 2 and end at 28*
range (30,3,-3)	*- will start the range of values from 30 and end at 6*

6. Write note on continue statement.
- Continue statement is used to skip the remaining part of a loop and start with next iteration.
- **Syntax:** *continue*
- **Example:**

```
for word in "Jump Statement":
    if word = = "e":
        continue
    print (word, end=")
print ("\n End of the program")
```

- **Output:**

```
Jump Statmnt
End of the program
```

PART - III

1. Write a program to display

```
A
A B
A B C
A B C D
A B C D E
```

- **Coding:**

```
for i in range(65,70):
    for j in range(65, i+1):
        print(chr(j), end=' ')
    print(end='\n')
```

2. Write short note on if … else statement.
- The **if .. else** statement provides control to check the true block as well as the false block.
- **Syntax:**

```
if <condition>:
    statements-block 1
else:
    statements-block 2
```

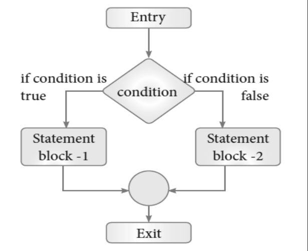

Fig. 6.1 if..else statement execution

- **Example :**

```
a = int(input("Enter any number :"))
if a%2==0:
    print (a, " is an even number")
else:
    print (a, " is an odd number")
```

- **Output1:**

```
Enter any number: 6
6 is an even number
```

- **Output2:**

```
Enter any number: 7
7 is an odd number
```

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

```
a=int (input("Enter first number : "))
```

```
b=int (input("Enter second number : "))
c=int (input("Enter third number : "))
if a>b and a>c:
        print (a, "is largest number")
elif b>c:
        print (b, "is largest number")
else:
        print (c, "is largest number")
```

Output:

```
Enter first number: 8
Enter second number: 3
Enter third number: 12
12 is largest number
```

4. Explain while loop with syntax and example.

- In the **while** loop, the condition is any valid Boolean expression returning True or False.
- The **else** part of while is optional part of **while**.
- The **statements block1** is kept executed till the condition is True.
- If the **else** part is written, it is executed when the condition is tested False.
- Recall **while** loop belongs to entry check loop type, that is it is not executed even once if the condition is tested False in the beginning.
- **Syntax:**

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

- Example :

```
i = 1
while(i<=5)
        print(i, end='\t')
        i=i+1
```

- Output:

```
1        2        3        4        5
```

5. List the differences between break and continue statements.

Break	Continue
• The **break** statement terminates the loop containing it. • Control flows to the statement immediately after the body of the loop.	• Continue statement is used to skip the remaining part of a loop and start with next iteration.
• **Syntax:** *break*	• **Syntax:** *continue*

• **EXAMPLE:**
for word in "Jump Statement":
    if word = = "e":
        break
    print (word, end= ' ')
	Example:
	    for word in "Jump Statement":
	        if word = = "e":
	            continue
	        print (word, end=")

• **Output:** Jump Stat	• **Output:** Jump Statmnt

PART - IV

1. Write a detail note on 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.
- **Syntax:**

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

- The *counter_variable* mentioned in the syntax is similar to the control variable that we used in the **for** loop of C++ and the *sequence* refers to the initial, final and increment value.
- In Python, **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.**
- **The syntax of range():**
  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 range
  Range(1,6,1) will start the range of values from 1 and end at 5.
- Example :

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

- **Output:**

```
2 4 6 8
```

2. Write short note on nested if..elif...else statement.

- When we need to construct a chain of **if** statement(s) then **'elif'** clause can be used instead of **'else'.**
- **Syntax:**

```
if <condition-1>:
        statements-block 1
elif <condition-2>:
        statements-block 2
else:
        statements-block n
```

- In the syntax of **if..elif..else**, **condition-1** is tested if it is **true** then **statements-block1** is executed, otherwise the control checks **condition-2**, if it is **true statements-block2** is executed and even if it **fails statements-block n** mentioned in **else** part is executed.

- **Example :**

```
a=int (input("Enter first number : "))
b=int (input("Enter second number : "))
c=int (input("Enter third number : "))
if a>b and a>c:
        print (a, "is largest number")
elif b>c:
        print (b, "is largest number")
else:
        print (c, "is largest number")
```

- **Output:**

```
Enter first number: 8
Enter second number: 3
Enter third number: 12
12 is largest number
```

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

```
for n in range(101,1000,2):
        print(n,end="\n")
```

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

```
n = input(int("Enter the table number:"))
for a in range(1,6):
        print(n, " X ", a , " = " , n*a)
          print(end='\n')
```

Output:

```
Enter the table number: 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
```

<u>EXTRA QUESTION ANSWERS:</u>

1. What are sequential statements?
   - A sequential statement is composed of a sequence of statements which are executed one after another.

2.What is alternative or branching statement?
   - There may be situations, where we need to skip a segment or set of statements and execute another segment based on the test of a condition. This is called *alternative* or *branching*

3.What are iterative or looping statement?
   - To execute a set of statements multiple times, called ***iteration*** or ***looping***.

4.Write short note on simple if statement.
   - **Simple if** is the simplest of all decision making statements.
   - Condition should be in the form of relational or logical expression.
   - **Syntax:**

```
if <condition>:
        statements-block1
```

- **Example :**

```
x=int (input("Enter your age :"))
if x>=18:
        print ("You are eligible for voting")
```

- **Output:**

```
Enter your age: 19
You are eligible for voting
```

5. Explain alternate method to use of if .. else statement.
   - An alternate method of **if..else statement** can also written as:
   - **Syntax:**
     variable = variable1 ***if*** *condition* ***else*** *variable 2*
   - The condition specified in the **if** is checked, if it is **true**, the value of **variable1** is stored in variable on the left side of the assignment, otherwise **variable2** is taken as the value.

- **Example:**

```
a = int (input("Enter any number :"))
x="even" if a%2==0 else "odd"
print (a, " is ",x)
```

- **Output1:**

```
Enter any number: 6
6 is even
```

- **Output2:**

```
Enter any number: 7
7 is odd
```

6. Give example how to use "in" and "not in" in if statement.

```
ch=input ("Enter a character :")
if ch in ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'):
        print (ch,' is a vowel')
if ch not in ('a', 'b', 'c'):
        print (ch,' the letter is not a/b/c')
```

Output 1:

```
Enter a character : e
e is a vowel
```

7. Write a program to check if the year is leap year or not.

```
year=int (input("Enter  year : "))
if year%4==0:
        print(year, " is a leap year")
else:
        print(year, " is not a leap year")
```

Output1:

```
Enter year: 2000
2000 is a leap year
```

Output2:

```
Enter year: 2001
2001 is not a leap year
```

8. Write a program to check whether the given character is a vowel or not.

```
ch=input (“Enter a character :”)
if ch in (‘a’, ‘A’, ‘e’, ‘E’, ‘i’, ‘I’, ‘o’, ‘O’, ‘u’, ‘U’):
        print (ch, “ is a vowel”)
else:
        print(ch, “ is not a vowel”)
```

Output1:
```
Enter a character : e
e is a vowel
```
Output2:
```
Enter a character: b
B is not a vowel
```

9. Using if..else..elif statement check smallest of three numbers.

```
a=int (input(“Enter first number : ”))
b=int (input(“Enter second number : ”))
c=int (input(“Enter third number : ”))
if a<b and a<c:
        print (a, “is smallest number”)
elif b<c:
        print (b, “is smallest number”)
else:
        print (c, “is smallest number”)
```

Output:
```
Enter first number: 8
Enter second number: 3
Enter third number: 12
3 is smallest number
```

10. Write a program to check if a number is Positive, Negative or zero.

```
n=int (input(“Enter first number : ”))
if n>0:
        print (n, “is Positive number”)
elif n==0:
        print (n, “is Zero”)
else:
        print (n, “is Negative number”)
```

Output:
```
Enter first number: 8
8 is Positive number
```

11.What is loop or iteration? When we use loop? What are the types of loop?

- A **loop** statement allows to execute a statement or group of statements multiple times.
- Iteration or loop are used in situation when the user need to execute a block of code several of times or till the condition is satisfied.
- **Python provides two types of looping constructs:**
    1. while loop
    2. for loop

12. What is nested loop? Explain with example.

- A loop placed within another loop is called as nested loop structure.
- One can place a while within another while; for within another for;
- for within while and while within for to construct such nested loops.
- Following is an example to illustrate the use of for loop to print the following pattern
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```
- **Program code:**
```
i=1
while (i<=6):
        for j in range (1,i):
                print (j,end='\t')
        print (end='\n')
        i +=1
```

13. What are jump statements in python?

- The jump statement in Python, is used to unconditionally transfer the control from one part of the program to another.
- There are three keywords to achieve jump statements in Python :
  1. break
  2. continue
  3. pass

14. Write a note on pass statement.

- **pass** statement in Python programming is a null statement.
- pass statement when executed by the interpreter it is completely ignored.
- Nothing happens when pass is executed, it results in no operation.
- **Syntax:** *pass*
- **<u>Program to illustrate the use of pass statement</u>**
```
a=int (input(“Enter any number :”))
if (a==0):
        pass
else:
        print (“non zero value is accepted”)
```
- **Output:**
```
Enter any number :3
non zero value is accepted
```
- When the above code is executed if the input value is 0 (zero) then no action will be performed.

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

```
a = 0
b = 1
n = int(input(“Enter number of terms”))
print(a, end=’ ‘)
print(b, end=' ')
for i in range(3, n+1):
    c=a+b
    print(c, end=' ')
    a = b
    b = c
Output:
Enter number of terms 8
0 1 1 2 3 5 8 13
```

16. Write a program to display sum of natural numbers, up to n.

```
s = 0
n = int(input("Enter value for n"))
for i in range(1, n+1):
    s = s + i
print("sum = ", s)

output:
Enter value for n 5
sum=15
```

17. Write a program to check if the given number is a palindrome or not.

```
n = int(input("Enter the number"))
s, d, x = 0,0,n
while(n!=0):
    d = n%10
    s = (s*10)+d
    n = n//10
if s ==x:
    print("The given number is palindrome")
else:
    print("The given number is not palindrome")
```

18. Write a program to print the following pattern

```
* * * * *
* * * *
* * *
* *
*
```

- Coding:

```
for i in range(5,0, -1):
        for j in range(1, i+1):
                print('*', end=' ')
        print(end='\n')
```

Choose the best answer 1 Marks

1. How many important control structures are there in Python?

A) 3 B) 4 C) 5 D) 6

2. elif can be considered to be abbreviation of

A) nested if B) if..else **C) else if** D) if..elif

3. What plays a vital role in Python programming?

A) Statements B) Control C) Structure **D) Indentation**

4. Which statement is generally used as a placeholder?

A) continue B) break **C) pass** D) goto

5. The condition in the if statement should be in the form of

A) Arithmetic or Relational expression B) Arithmetic or Logical expression
C) Relational or Logical expression D) Arithmetic

6. Which is the most comfortable loop?

A) do..while B) while **C) for** D) if..elif

7. What is the output of the following snippet?

```
i=1
while True:
        if i%3 ==0:
                break
        print(i,end='')
        i +=1
```

A) 12 B) 123 C) 1234 D) 124

8. What is the output of the following snippet?

```
   T=1
      while T:
    print(True)
      break
```

A) False B) True C) 0 **D) no output**

9. What is the output of the following snippet?

```
   T=1
   while T:
      print(True)
      break
```

A) False **B) True** C) 0 D) no output

10. Which amongst this is not a jump statement ?

A) for B) goto C) continue D) break

11. Which punctuation should be used in the blank?

if <condition>_
statements-block 1
else:
statements-block 2

A) ; **B) :** C) :: D) !

12. Find odd one out.

A) Keywords B) Operator C) Identifiers **D) Programs**

13. Which of the following is not a control structures?

A) Sequential B) Branching **C) Operator** D) Looping

14. To construct a chain of if statement, else can be replaced by

A) while B) ifel C) else if **D) elif**

15. Which of the following statement is incorrect?

A) In a if statement there is no limit of elif clause that can be used

B) else clause if used should be placed at the end

C) python provides three types of if structure

D) if .. elif .. else not similar to C++ nested if

16. The following statements is an example of

```
print("ONE")
print("TWO")
```

A)Iterative B) Branching **C) Sequential** D) Looping

17. Which statement in python used to transfer the control from one part of the program to another unconditionally?

A) jump B) loop C) alternative D) iterative

18. range(20), the range count from

A) 1 to 20 **B) 0 to 19** C) 1 to 19 D) 0 to 20

19. Python provides _______ types of looping constructs.

A) 3 B) 4 C) 5 **D) 2**

20. Write the output for the following program:

```
for i in range(1, 10, 2):
        print(i, end = ' ')
```

A)1 3 5 7 9 B) 1 3 5 7 C) 1 3 5 D) 1 2 3 4 5 6 7 8 9 10

CHAPTER – 7 – PYTHON FUNCTIONS

PART - II

1. What are functions? What are its types?

- Functions are nothing but a group of related statements that perform a specific task.
  1. User-defined Functions
  2. Built-in Functions
  3. Lambda Functions
  4. Recursion Functions

2. Explain the different types of functions?

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

3. What are the main advantages of function?

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

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

- Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it.
- Two types of scopes - **local scope** and **global scope**.

5. Define Global scope.

- **Global Scope**
- 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.

6. What is base condition in recursive function?

- The condition that is applied in any recursive function is known as base condition.
- A base condition is must in every recursive function otherwise it will continue to execute like an infinite loop.

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

- print(fact (2000)) will give Runtime Error maximum recursion depth exceeded in comparison.
- This happens because python stops calling recursive function after 1000 calls by default.
- 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))
```

PART - III

1. Write the rules of local variable?

- **Rules of local variable**
- A variable with local scope can be accessed only within the function/block that it is created in.
- When a variable is created inside the function/block, the variable becomes local to it.
- A local variable only exists while the function is executing.
- The formal arguments are also local to function.

2. Write the basic rules for global keyword in python?

- **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

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

```
c = 1 # global variable
def add():
        c = c + 2 # increment c by 2
        print(c)
add()
```

Output:

```
Unbound Local Error: local variable 'c' referenced before assignment
```

- Without using the global keyword we cannot modify the global variable inside the function but we can only access the global variable.

4. Differentiate floor() and ceil() functions.

Function	Description	Syntax	Example
floor ( )	Returns the largest integer less than or equal to x	math.floor (x)	x=26.7 y=-26.7 z=-23.2 print (math.floor (x)) print (math.floor (y)) print (math.floor (z)) **Output:** 26 -27 -24
ceil ( )	Returns the smallest integer greater than or equal to x	math.ceil (x)	x= 26.7 y= -26.7 z= -23.2 print (math.ceil (x)) print (math.ceil (y)) print (math.ceil (z)) **Output:** 27 -26 -23

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

```
year  = int(input("Enter year"))
if year% 4 == 0:
        print(year. "is a leap year")
else:
        print(year, "is not a leap year")
```

- **Output 1:**
  Enter year 2020
  2020 is a leap year
- **Output 2:**
  Enter year 2019
  2019 is not a leap year

6. **What is Composition in functions?**

- The value returned by a function may be used as an argument for another function in a nested manner. This is called **composition**.
- For example, if we wish to take a numeric value or an expression as a input from the user, we take the input string from the user using the function **input()** and apply **eval()** function to evaluate its value, for example:
- *# This program explains composition*

```
>>> n1 = eval (input ("Enter a number: "))
Enter a number: 234
>>> n1
234
>>> n2 = eval (input ("Enter an arithmetic expression: "))
Enter an arithmetic expression: 12.0+13.0 * 2
>>> n2
38.0
```

7. H**ow recursive function works? Explain with an example?**

- 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.
- **Example:**

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

Output:

```
1
120
```

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

- Any input parameters or arguments should be placed within these parentheses when you define a function.
- The code block always comes after a colon (:) and is indented.
- The statement **"return [expression]"** exits a function, optionally passing back an expression to the caller.
- A **"return"** with no arguments is the same as return **None**.
- Python keywords should not be used as function name.

PART -IV

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

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

User defined functions:

- Functions must be defined, to create and use certain functionality.

- Function blocks begin with the keyword **“def”** followed by function name and parenthesis ().
- **Syntax for User defined function**

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

- **Example:**

```
def hello():
    print (“hello - Python”)
    return
```

Built-in function:

1. **id( )** - Return the “identity” of an object. i.e. the address of the object in memory.
   Note: the address of x and y may differ in your system.
   - **Syntax:** id (object)
   - **Example:**

```
x=15
y='a'
print ('address of x is :',id (x))
print ('address of y is :',id (y))
```

   - **Output:**

```
address of x is : 1357486752
address of y is : 13480736
```

2. **chr() 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
+
```

Lambda Function:

- In Python, anonymous function is a function that is defined without a name.
- While normal functions are defined using the **def** keyword, in Python anonymous functions are defined using the **lambda** keyword.
- Hence, anonymous functions are also called as **lambda** functions.

Use of lambda or anonymous function?

- 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**
- The syntax for anonymous functions is as follows:

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
```

- **RECURSIVE FUNCTION:**
- 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.

Recursive Function – Base Condition:

- The condition that is applied in any recursive function is known as base condition.
- A base condition is must in every recursive function otherwise it will continue to execute like an infinite loop.

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.
- Example of recursive function:

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

Output:

```
1
120
```

2. Explain the scope of variables with example in Python?

- Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it.
- Two types of scopes - **local scope** and **global scope**.

- **Local Scope**
- A variable declared inside the function's body or in the local scope is known as local variable.
- **Rules of local variable**
- A variable with local scope can be accessed only within the function/block that it is created in.
- When a variable is created inside the function/block, the variable becomes local to it.
- A local variable only exists while the function is executing.
- The formal arguments are also local to function.
- **Example**

```
def loc():
        y=0 # local scope
        print(y)
loc()
```

- **Output:**
  0

- **Global Scope**
- 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()
```

- **Output:**

```
1
```

3. Explain the following built-in functions.
(a) id() (b) chr() (c) round() (d) type() (e) pow()

2. id( ) - Return the “identity” of an object. i.e. the address of the object in memory.
Note: the address of x and y may differ in your system.

- **Syntax:** id (object)
- **Example:**

```
x=15
y='a'
print ('address of x is :',id (x))
print ('address of y is :',id (y))
```

- **Output:**

```
address of x is : 1357486752
address of y is : 13480736
```

2. chr() 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
+
```

3. round() Returns the nearest integer to its input.
First argument (number) is used to specify the value to be rounded.

- **Syntax:** round(number[,digits])
- **Example:**

```
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
```

4. type() Returns the type of object for the given single object.
Note: This function used with single object parameter.

- **Syntax:** type(object)
- **Example:**

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

- **Output:**

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

5. pow( ) Returns the computation of ab i.e. (ab) a raised to the power of b.**
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
```

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

```
a = int(input(“ Enter first number”))
b = int(input(“ Enter second number”))
```

```
g  = a if a>b else b
while (True):
        if g%a == 0 and g%b == 0:
                print("LCM is ", g)
                break
        g+=1
```

- **Output:**

```
Enter first number 2
Enter second number 4
LCM is 4
```

5. What is recursive function? Explain with example.

- 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.

Recursive Function – Base Condition:

- The condition that is applied in any recursive function is known as base condition.
- A base condition is must in every recursive function otherwise it will continue to execute like an infinite loop.

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.
- Example of recursive function:

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

Output:

```
1
120
```

EXTRA QUESTION ANSWER:

1. How will you modify Global Variable inside a function using global keyword?

- **Modifying Global Variable From Inside the Function**
- Without using the global keyword we cannot modify the global variable inside the function but we can only access the global variable.
- **Example :**

```
x = 0 # global variable
def add():
        global x
        x = x + 5 # increment by 2
        print ("Inside add() function x value is :", x)
add()
print ("In main x value is :", x)
```

- **Output:**

```
Inside add() function x value is : 5
In main x value is : 5
```

2. What is defining a function?

- Functions must be defined, to create and use certain functionality.
- Function blocks begin with the keyword **"def"** followed by function name and parenthesis ().
- **Syntax for User defined function**

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

- **Example:**

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

3. What is a 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*.

4. What are nested blocks?

- 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.

5. What are the advantages of user defined functions?

- Functions help us to divide a program into modules. This makes the code easier to manage.
- It implements code reuse. Every time you need to execute a sequence of statements, all you need to do is to call the function.
- Functions, allows us to change functionality easily, and different programmers can work on different functions.

6. Explain the different types of function arguments?

- Arguments are used to call a function and there are primarily 4 types of functions that one can use:

1. Required arguments,
2. Keyword arguments,
3. Default arguments and
4. Variable-length arguments.

- **Required Arguments**
- "Required Arguments" are the arguments passed to a function in correct positional order.
- Here, the number of arguments in the function call should match exactly with the function definition. You need atleast one parameter to prevent syntax errors to get the required output.
- Example :

```
def printstring(str):
        print ("Example - Required arguments ")
        print (str)
        return
printstring("Welcome")
```

- **Keyword Arguments**
- Keyword arguments will invoke the function after the parameters are recognized by their parameter names.
- The value of the keyword argument is matched with the parameter name and so, one can also put arguments in improper order (not in order).
- **Example**

```
def printdata (name, age):
        print ("Example-3 Keyword arguments")
        print ("Name :",name)
        print ("Age :",age)
        return
printdata (age=25, name="Gshan")
```

- **Default Arguments**
- In Python the default argument is an argument that takes a default value if no value is provided in the function call.
- The following example uses default arguments, that prints default salary when no argument is passed.
- Example :

```
def printinfo( name, salary = 3500):
        print ("Name: ", name)
        print ("Salary: ", salary)
        return
printinfo("Mani")
```

- **Output:**

```
Name: Mani
Salary: 3500
```

- **Variable-Length Arguments:**
- In some instances you might need to pass more arguments than have already been specified.
- Going back to the function to redefine it can be a tedious process.
- Variable-Length arguments can be used instead.
- These are not specified in the function's definition and an asterisk (*) is used to define such arguments.

- **Syntax - Variable-Length Arguments**

```
def function_name(*args):
        function_body
        return_statement
```

- **Example :**

```
def printnos (*nos):
    for n in nos:
        print(n)
    return
print ('Printing two values')
printnos (1,2)
print ('Printing three values')
printnos (10,20,30)
```

Output:

```
Printing two values
1
2
Printing three values
10
20
30
```

7. What is anonymous function?
- In Python, anonymous function is a function that is defined without a name.
- While normal functions are defined using the **def** keyword, in Python anonymous functions are defined using the **lambda** keyword.
- Hence, anonymous functions are also called as **lambda** functions.

8. What is the use of lambda or anonymous function?
- 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().

9. Write short notes on return statement.
- The return statement causes your function to exit and returns a value to its caller.
- The point of functions in general is to take inputs and return something.
- The return statement is used when a function is ready to return a value to its caller.
- Any number of 'return' statements are allowed in a function definition but only one of them is executed at run time.
- Syntax of return
  return [expression list]

Hands on Experience

1. **Try the following code in the given program**

```
def printinfo( name, salary = 3500):
        print ("Name: ", name)
        print ("Salary: ", salary)
        return
```

Sl.No	Code	Result
1	printinfo("3500")	Name: 3500 Salary:3500
2	printinfo("3500", "Sri")	Name: 3500 Salary:Sri
3	printinfo(name="balu")	Name: balu Salary:3500
4	printinfor("jose", 1234)	Name: jose Salary:1234
5	printinfo(" ", salary=1234)	Name: Salary:1234

2. **Evaluate the following functions and write the output**

Sl.No	Function	Output
1	eval('25*2-5*4')	30

2	**math.sqrt(abs(-81))**	**9.0**
3	**math.ceil(3.5+4.6)**	**9**
4	**math.floor(3.5+4.6)**	**8**

3. Evaluate the following functions and write the output

Sl.No	**Function**	**Output**
1	**1. abs(-25+12.0)** **2. abs(-3.2)**	**13.0** **3.2**
2	**1. ord('2')** **2. ord('$')**	**50** **36**
3	**type('s')**	**<class 'str'>**
4	**bin(16)**	**0b10000**
5	**1. chr(13)** **2. print(chr(13))**	**'\r'**
6	**1. round(18.2,1)** **2. round(18.2,0)** **3. round(0.5100,3)** **4. round(0.5120,3)**	**18.2** **18.0** **0.510** **0.512**
7	**1. format(66,'c')** **2. format(10,'x')** **3. format(10,'X')** **4. format(0b110,'d')** **5. format(0xa,'d')**	**B** **a** **A** **6** **10**
8	**1. pow(2, -3)** **2. pow(2,3.0)** **3. pow(2,0)** **4. pow((1+2),2)** **5. pow(-3,2)** **6. pow(2*2,2)**	**0.125** **8.0** **1** **9** **9** **16**

Choose the best answer: (1 Mark)

1. A named blocks of code that are designed to do one specific job is called as

(a) Loop (b) Branching **(c) Function** (d) Block

2. Which of the following keyword is used to begin the function block?

(a) define (b) for (c) finally **(d) def**

3. Which of the following keyword is used to exit a function block?

(a) define **(b) return** (c) finally (d) def

4. While defining a function which of the following symbol is used.

(a) ; (semicolon) (b) . (dot) **(c) : (colon)** (d) $ (dollar)

5. In which arguments the correct positional order is passed to a function?

(a) Required (b) Keyword (c) Default (d) Variable-length

6. Read the following statement and choose the correct statement(s).

(I) In Python, you don't have to mention the specific data types while defining function.

(II) Python keywords can be used as function name.

(a) I is correct and II is wrong (b) Both are correct

(c) I is wrong and II is correct (d) Both are wrong

7. Pick the correct one to execute the given statement successfully.

if ____ : print(x, " is a leap year")

(a) x%2=0 **(b) x%4==0** (c) x/4=0 (d) x%4=0

8. Which of the following keyword is used to define the function testpython(): ?

(a) define (b) pass **(c) def** (d) while

9. Which symbol is used to define variable length arguments?
(a) * (b) $ (c) # (d) @
10. Which of the following function is an example that supports variable length arguments?
(a) while (b) if **(c) print()** (d) input()
11. A Function which calls itself is called as

(a) Built-in **(b) Recursion** (c) Lambda (d) return

12. Which Keyword is used to define anonymous function?

(a) lambda (b) recursion (c) function (d) def

13. Which of the following keyword is used to exit a function block?

(a) define **(b) return** (c) finally (d) def

14. Which function is mostly used for creating small and one time anonymous function?
a) Synchronous **b) Lambda** c) User – defined d) Built – in
15. How many types of scopes in Python?
a) 2 b) 3 c) 4 d) 5
16. Defining a variable outside a function is by default
a) local **b) global** c) function d) operands
17. The output of one function used as an argument for another function is called
a) Recursion b) Built-in **c) Composition** d) Decomposition
18. By default, python stops calling recursive function after
a) 2000 b) 100 c) 5000 **d) 1000**
19. If there is no return statement present inside the function, then the function will return
a) No b) Nothing **c) None** d) def
20. Choose the incorrect statement from the following
i) A variable defined within a block can be accessed outside a function also
ii) Local variable only exits while the function is executing.
iii) A variable defined outside a function is global by default.
iv) local keyword is used to read and write local variable inside a function.
a)i and ii b) iii and iv c) ii and iii **d) i and iv**
21.print(abs(-23.2)) displays
a) -23 b) -23.2 **c) 23.2** d) 0.2
22. print(ord('A')) displays
a) 65 b) A c) a d) 97
23. Which of the following function return the address of the object in memory?
a) address() b) object() **c) id()** d) format()
24. print(format(25,'o') displays
a) 25 b) 025 c) 031 **d) 31**
25. Which function is an alternative to bin()?
a) id() **b) format()** c) type() d) chr()
26. Which function returns the ascii value for the Unicode character?
a) ord() b) type() c) ascii() d) chr()
27. Which of the following function returns the smallest integer greater than or equal to given value?
a) floor() b) round() c) pow() **d) ceil()**
28. print(bin(5)) displays

a) 5	b) 101	**c) 0b101**	d) 1010b

29. print(math.ceil(-23.2)) displays

a) -23.2	b) -24	**c) -23**	d) 23.2

30. print(type(True)) displays

a) <class ‘bool’>	b) <class ‘yes’>	c) <class ‘str’>	d) <class ‘boolean’>

CHAPTER – 8 – STRINGS AND STRING MANIPULATIONS

1. What is a string?
String is a data type in python, which is used to handle array of characters. String is a sequence of Unicode characters that may be a combination of letters, numbers, or special symbols enclosed within single, double or even triple quotes.
Example
‘Welcome to learning Python’
“Welcome to learning Python”
‘‘‘ “Welcome to learning Python” ’’’

2. Write short notes on replace ( ) function.
- A function replace() to change all occurrences of a particular character in a string.
- **Syntax:**
  replace(“char1”, “char2”)
- **The replace() function replaces all occurrences of char1 with char2.**

```
>>> str1="How are you"
>>> print (str1)
How are you
>>> print (str1.replace("o", "e"))
Hew are yeu
```

3. How will you delete a string in Python?
- Python will not allow deleting a particular character in a string.
- Whereas you can remove entire string variable using del command.
- **Example:**
  ```
  >>> str1="How about you"
  >>> print (str1)
  How about you
  >>> del str1
  ```
- This command deletes the string variable str1.

4. Explain the string operators with example?
- Python provides the following operators for string operations. These operators are useful to manipulate string.
- **Concatenation (+)**
- Joining of two or more strings is called as Concatenation. The plus (+) operator is used to concatenate strings in python.
- **Example**
- >>> "welcome" + "Python"
- ***welcomePython***

- **(ii) Append (+ =)**

- Adding more strings at the end of an existing string is known as append. The operator += is used to append a new string with an existing string.
- Example
  ```
  >>> str1="Welcome to "
  >>> str1+="Learn Python"
  >>> print (str1)
  Welcome to Learn Python
  ```

- (iii) Repeating (*)
- The multiplication operator (*) is used to display a string in multiple number of times.
- Example
  ```
  >>> str1="Welcome "
  >>> print (str1*4)
  Welcome Welcome Welcome Welcome
  ```

- (iv) String slicing
- Slice is a substring of a main string. A substring can be taken from the original string by using [ ] operator and index or subscript values. Thus, [ ] is also known as slicing operator.
- **General format of slice operation:**
  str[start:end]
  Where ***start*** is the beginning index
  end is the last index value of a character in the string (n-1)
- Python takes the end value less than one (n-1)from the actual index specified.

- **Example I : slice a single character from a string**
  ```
  >>> str1="THIRUKKURAL"
  >>> print (str1[0])
  T
  ```

- **Example II : slice a substring from index 0 to 4**
  ```
  >>> print (str1[0:5])
  THIRU
  ```

- **Example III : slice a substring using index 0 to 4 but without specifying the beginning index.**
  ```
  >>> print (str1[:5])
  THIRU
  ```

  Example IV : slice a substring using index 6 to 10 but without specifying the end index.
  ```
  >>> print (str1[6:])
  KURAL
  ```

- **(v) Stride when slicing string**
- When the slicing operation, you can specify a third argument as the stride, which refers to the number of characters to move forward after the first character is retrieved from the string. The default value of stride is 1.
- Example
  ```
  >>> str1 = "Welcome to learn Python"
  >>> print (str1[10:16])
  learn
  >>> print (str1[10:16:4])
  r
  ```

```
>>> print (str1[10:16:2])
er
>>> print (str1[::3])
Wceoenyo
```

- You can also use negative value as stride (third argument). If you specify a negative value, it prints in reverse order.
- Example:

```
>>> str1 = "Welcome to learn Python"
>>> print(str1[::-2])
```
nhy re teolW

5. What are string formatting operators?

- The string formatting operator is one of the most exciting feature of python. The formatting operator % is used to construct strings, replacing parts of the strings with the data stored in variables.
- Syntax:

(“String to be display with %val1 and %val2” %(val1, val2))
```
name = "Rajarajan"
mark = 98
print ("Name: %s and Marks: %d" %(name,mark))
```
Output
```
Name: Rajarajan and Marks: 98
```

6. Tabulate the formatting characters?

Format characters	USAGE
%c	Character
%d (or) %i	Signed decimal integer
%s	String
%u	Unsigned decimal integer
%o	Octal integer
%x or %X	Hexadecimal integer (lower case x refers a-f; upper case X refers A-F)
%e or %E	Exponential notation
%f	Floating point numbers
%g or %G	Short numbers in floating point or exponential notation.

7. Write short notes on Escape sequence in python

- Escape sequences starts with a backslash and it can be interpreted differently. When you have use single quote to represent a string, all the single quotes inside the string must be escaped. Similar is the case with double quotes

Example :
```
# String within triple quotes to display a string with single quote
>>> print ('''They said, "What's there?"''')
They said, "What's there?"
# String within single quotes to display a string with single quote using escape sequence
>>> print ('They said, "What\'s there?"')
They said, "What's there?"
# String within double quotes to display a string with single quote using escape sequence
>>> print ("They said, \"What's there?\"")
He said, "What's there?"
```

8. Tabulate the escape sequence characters

Escape Sequence	DESCRIPTION
\newline	Backslash and newline ignored
\\	Backslash
\'	Single quote
\"	Double quote
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Form feed
\n	ASCII Linefeed
\r	ASCII Carriage Return
\t	ASCII Horizontal Tab
\v	ASCII Vertical Tab
\ooo	Character with octal value ooo
\xHH	Character with hexadecimal value HH

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

```
str1 = “School”
print(str1*3)
Output :  School  School  School
```

10. What is slicing?

- Slice is a substring of a main string. A substring can be taken from the original string by using [ ] operator and index or subscript values. Thus, [ ] is also known as slicing operator.

General format of slice operation:

str[start:end]

Where ***start*** is the beginning index

end is the last index value of a character in the string.

Python takes the end value less than one from the actual index specified.

For example, if you want to slice first 5 characters from a string, you have to specify it as 0 to 5. Because, python consider only the end value as n-1.

- **Example I : slice a single character from a string**

```
>>> str1="THIRUKKURAL"
>>> print (str1[0])
```
T

Example II : slice a substring from index 0 to 4
```
>>> print (str1[0:5])
```
THIRU

Example III : slice a substring using index 0 to 4 but without specifying the beginning index.
```
>>> print (str1[:5])
```
THIRU

Example IV : slice a substring using index 0 to 4 but without specifying the end index.
```
>>> print (str1[6:])
```
KURAL

11. Write a Python program to display the given pattern

```
C O M P U T E R
C O M P U T E
C O M P U T
C O M P U
```

```
C O M P
C O M
C O
C
str1="COMPUTER"
index=8
for i in str1:
print (str1[:index+1])
index -= 1
Output :
C O M P U T E R
C O M P U T E
C O M P U T
C O M P U
C O M P
C O M
C O
C
```

12. What will be the output of the given python program?

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

13. Write a Python program to print the pattern.

```
*
* *
* * *
* * * *
* * * * *
        str1=' * '
        i=1
        while i<=5:
                print (str1*i)
                i+=1
Output
*
* *
* * *
* * * *
* * * * *
```

14. What is format( ) function?

- The format( ) function used with strings is very versatile and powerful function used for formatting strings.
- The curly braces { } are used as placeholders or replacement fields which get replaced along with format( ) function.
- Example :

```
num1=int (input("Number 1: "))
num2=int (input("Number 2: "))
print ("The sum of { } and { } is { }".format(num1, num2,(num1+num2)))
Out Put
Number 1: 34
Number 2: 54
The sum of 34 and 54 is 88
```

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

capitalize( )	Used to capitalize the first character of the string	>>> city="chennai" >>> print(city.capitalize()) Chennai

swapcase( )	It will change case of every character to its opposite case vice-versa.	>>> str1="tAmiL NaDu"
		>>> print(str1.swapcase())
		TaMIl nAdU

16. What will be the output of the given python program?

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

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

Syntax	Description	Example

count(str, beg, end)	Returns the number of substrings occurs within the given range. Remember that substring may be a single character. Range (beg and end) arguments are optional. If it is not given, python searched in whole string. Search is case sensitive.	>>> str1="Raja Raja Chozhan"
		>>>
		print(str1.count('Raja'))
		2
		>>> print(str1.count('r'))
		0
		>>> print(str1.count('R'))
		2
		>>> print(str1.count('a'))
		5
		>>>
		print(str1.count('a',0,5))
		2
		>>> print(str1.count('a',11))
		1

18. Program to check whether the given string is palindrome or not

```
str1 = input ("Enter a string: ")
str2 = ' '
index=-1
for i in str1:
    str2 += str1[index]
    index -= 1
print ("The given string = { } \n The Reversed string = { }".format(str1, str2))
if (str1==str2):
    print ("Hence, the given string is Palindrome")
else:
    print ("Hence, the given is not a palindrome")
```

Output : 1

```
Enter a string: malayalam
The given string = malayalam
The Reversed string = malayalam
Hence, the given string is Palindrome
```

Output : 2

```
Enter a string: welcome
The given string = welcome
The Reversed string = emoclew
Hence, the given string is not a palindrome
```

19. Program to create an Abecedarian series. (Abecedarian refers list of elements appear in alphabetical order)

```
str1="ABCDEFGH"
str2="ate"
for i in str1:
    print ((i+str2),end='\t')
```

Output

```
Aate Bate Cate Date Eate Fate Gate Hate
```

20. Program that accept a string from the user and display the same after removing vowels from it

```
def rem_vowels(s):
    temp_str=''
    for i in s:
        if i in "aAeEiIoOuU":
            pass
        else:
            temp_str+=i
print ("The string without vowels: ", temp_str)
str1= input ("Enter a String: ")
rem_vowels (str1)
```

Output

```
Enter a String: Mathematical fundations of Computer Science
The string without vowels: Mthmtcl fndtns f Cmptr Scnc
```

21. Program that count the occurrences of a character in a 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
```

Choose the best answer

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

str1 = "Chennai Schools"

str1[7] = "-"

(a) Chennai-Schools (b) Chenna-School **(c) Type error** (d) Chennai

2. Defining strings within triple quotes allows creating:

(a) Single line Strings **(b) Multiline Strings** (c) Double line Strings (d) Multiple Strings

3. Strings in python:

(a) Changeable (b) Mutable **(c) Immutable** (d) flexible

4. The subscript of a string may be:

(a) Positive (b) Negative (c) Both (a) and (b) **(d) Either (a) or (b)**

5. Which of the following used to handle array of characters in python?

a) Function b) Composition **c) String** d) Arguments

6. Strings are enclosed with

a) Single quote b) double quote c) triple quote d) All the above

7. String index values are also called as

a) class b) function **c) subscript** d) arguments

8. The positive subscript always starts with

a) 0 b) 1 c) -1 d) 1.0

9. The negative subscript always starts with

a) 0 b) 1 **c) -1** d) 1.0

10. Which command is used to delete the string variable

a) rem **b) del** c) delete d) remove

11. Which of the following is the output of the following python code?

str1="TamilNadu"

print(str1[::-1])

(a) Tamilnadu (b) Tmlau (c) udanlimaT **d) udaNlimaT**

12. Which of the following operator is used for concatenation?

(a) + (b) & (c) * d) =

13. Which of the following is the slicing operator?

(a) { } **(b) [ ]** (c) < > (d) ( )

14. What is stride?

(a) index value of slide operation (b) first argument of slice operation

(c) second argument of slice operation **(d) third argument of slice operation**

15. What is the output from the following statement?

```
str1 = “welcome”
print(str1[::3])
```

a)come b) ome **c) wce** d) wel

16. Adding more strings at the end of the existing string is known as

a) concat b) concatenation c) join **d) append**

17. Which of the following operator is used to construct strings?

a) : **b) %** c) :: d) #

18. Escape sequence starts with a

a) / **b) ** c) // d) \\\\

19. Which is a substring of a main string?

a) stride **b) slice** c) concat d) append

20. What is the output from the following statement?

```
str1 = “python”
print(str1[::-2])
```

a)nhy **b) pyt** c) hy d) on

---

21. Which of the following formatting character is used to print exponential notation in upper case?

(a) %e **(b) %E** (c) %g (d) %n

22. Which of the following is used as placeholders or replacement fields which get replaced along with format( ) function?

(a) { } (b) < > (c) ++ (d) ^^

23. The function returns the length of the string in python is

a) length() b) leng() **c) len()** d) strlen()

24. The function used to search the first occurrence of the substring in the given string is

a) search() **b) find()** c) findstring() d) searchstring()

25. Which function is a powerful function used for formatting strings.

a) format() b) string() c) slice() d) formatstring()

26. The ‘in’ and ‘not in’ operators are called as

a) string operators b) string formatting operators **c) membership operators** d) reference operators

27. What is output of the following.

‘mammals’.find(‘ma’,2)

a)0 b) 1 c) -1 **d) 3**

28. What is the output of the following : print(len(“corporation”))

a) 10 **b) 11** c) 12 d) 9

29. Which function is used to returns the exact copy of the string with all the letters in lowercase?

a) lower() b) tolower() c) lowercase() d) tolowercase()

30. Which function is used to return ASCII code of the character.

a) chr() **b) ord()** c) ascii() d) asc()

*** ALL THE BEST ***