PYTHON FUNCTIONS

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

As you already know, Python gives you many built-in functions like print, etc. but you can also create your own functions. These functions are called *user-defined functions*.

Defining a Function

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

Function blocks begin with the keyword def followed by the function name and parentheses ().

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.

The code block within every function starts with a colon : and is indented.

The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
def functionname( prameters):
   "function_docstring"
    function_body
  [ return [expression]]
```

By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

Example
The following function takes a string as input parameter and prints it on standard screen.

```
def printme( str ):
    "This prints a passed
    string into this
    function"
    Print(str)
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.
Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme function −

```
 # Now you can call printme function
printme("I'm first call to user definedfunction!")
printme("Again second call to the samefunction")
```

Function Arguments
You can call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments
- 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.
To call the function *printme*, you definitely need to pass one argument, otherwise it gives a syntax error asfollows –

```
def printme( str ):
"This prints a passed string into this function"
   print( str)
# Now you can call printme function
```

```
printme()
```
When the above code is executed, it produces the following result:
```
Traceback (most recent call last): File "test.py", line 11, in <module>
printme();
TypeError: printme() takes exactly 1 argument (0 given)
```

Keyword arguments

Keyword arguments are related to the function calls. When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name. This allows you to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters. You can also make keyword calls to the *printme* function in the following ways –

```
# Function definition is here
def printme( str ):
#"This prints a passed string into this function"
    print( str)
# Now you can call printme function
printme( str = "My string")
```

The following example gives more clear picture. Note that the order of parameters does not matter.

```
def printinfo( name, age ):
   print ("Name: ", name)
   print ("Age ", age )
printinfo( age=50, name="miki" )
```

Default arguments

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument. The following example gives an idea on default arguments, it prints default age if it is not passed –

```
def printinfo( name, age = 35 ):
  print("Name",name);
  print( "Age ", age)
printinfo( age=50, name="miki" )
printinfo( name="miki" )
```

Note : default arguments should follow non-default arguments. Failing to do so results in syntax error as shown in the example below :

```
def multiplication(num1=2,num2):
```

```
  return (num1*num2)
```

SyntaxError: non-default argument follows default argument

Mixing Positional and Keyword Arguments

It is possible to mix positional arguments and Keyword arguments, but for this positional argument must appear before any Keyword arguments. Let's see this through an example.

```
def my_func(a, b, c):
    print(a, b, c)
# using positional arguments only
my_func(12, 13, 14)

# here first argument is passed as positional arguments while other two as keyword argument
my_func(12, b=13, c=14)

# same as above
my_func(12, c=13, b=14)

# this is wrong as positional argument must appear before any keyword argument
# my_func(12, b=13, 14)
```

Illustrating the working of functions

```
def fun(a,b=1,c=5):
    print(a,b,c)

fun(3)   # function 1
fun(3,7,10)  # function2
fun(25,c=20) # function 3
fun(c=20,a=10) # function 4
#fun(29,b=10,20)  # function 5   error positional argument follows keyword argument
#fun(37,a=47) # function 6 error multiple values assign for a
# fun(d=56) # function 7  error unexpected keyword argument d
fun(b=56) # function 8 error required argument to a is missing correction fun(56) or fun(a=56)

'''correction for function 5
 fun(29,b=10,c=20)
  correction for function 6
```

```
fun(37)
fun(a=47)

correcttion for function 7
fun(56)
fun(b=56)

'''
```

Variable-length arguments (Not in syllabus)

If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

This way the function will receive a *tuple* of arguments, and can access the items accordingly:

```
# example 1 of variable length arguments
def add(*N):
    for I in N:
        print(I,end='  ')
    print()
add(10,20,30,40)
add(2,4)
add()
# example 2 of variable length arguments
def my_function(*kids):
  print("The youngest child is " + kids[2])

my_function("Emil", "Tobias", "Linus")
```

The *return* Statement

The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.
All the above examples are not returning any value. You can return a value from a function as follows –

```
def Add( arg1, arg2 ):
    total = arg1 + arg2
```

```
    print ("Inside the function : ", total_
    return total;

total = sum( 10, 20 );
print ("Outside the function : ", total)
```

Returning multiple values from Function

We can return multiple values from function using the return statement by separating them with a comma (,). Multiple values are returned as tuples.

```
def bigger(a, b):
    if a > b:
        return a, b
    else:
        return b, a

s = bigger(12, 100)
print(s)
print(type(s))
```

```
Expected Output:
(100, 12)
<class 'tuple'>
```

Scope of Variables

All variables in a program may not be accessible at all locations in that program. This depends on where you have declareda variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python −

Global ariables Local variables

Global vs. Local variables

In Python, a variable that is defined outside any function or any block is known as a global variable. It can be accessed in any functions defined onwards. Any change made to the global variable will impact all the functions in the program where that variable can be accessed.

A variable that is defined inside any function or a block is known as a local variable. It can be accessed only in the function or a block where it is defined. It exists only till the function executes.

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

Following is a simple example −

```
total = 0  # This is global variable. # Function definition is here
def sum( arg1, arg2 ):
# Add both the parameters and return them."
   total = arg1 + arg2; # Here total is local variable.
   print ("Inside the function local total : ", total)
   return total;

# Now you can call sum function
sum( 10, 20 );
print ("Outside the function global total : ", total)
```

When the above code is executed, it produces the following result −

```
Inside the function local total :        30
Outside the function global total :      0
```

Example 1:

```
global_var = 12         # a global variable

def func():
    local_var = 100     # this is local variable
    print(global_var)   # you can access global variables in side function

func()                  # calling function func()

#print(local_var)       # you can't access local_var outside the function, because as soon as function ends local_var is destroyed
```

Example 2:

```
xy = 100

def cool():
    xy = 200    # xy inside the function is totally different from xy outside the function
    print(xy)   # this will print local xy variable i.e 200

cool()

print(xy)       # this will print global xy variable i.e 100
```

You can bind local variable in the global scope by using the global keyword followed by the names of variables separated by comma (,).

Example 3:

```
t = 1

def increment():
    global t   # now t inside the function is same as t outside the function
    t = t + 1
    print(t) # Displays 2


increment()
print(t) # Displays 2
```

Note that you can't assign a value to variable while declaring them glob al .

Example 4:

```
t = 1

def increment():
    #global t = 1   # this is error
    global t
    t = 100        # this is okay
    t = t + 1
    print(t) # Displays 101

increment()
print(t) # Displays 101
```

Note: In fact there is no need to declare global variables outside the function. You can declare them global inside the function.

Example 5:

```
def foo():
    global x   # x is declared as global so it is available outside the functi
on
    x = 100

foo()
print(x)
```

Note:

• Any modification to global variable is permanent and affects all the functions where it is used.

• If a variable with the same name as the global variable is defined inside a function, then it is considered local to that function and hides the global variable.

• If the modified value of a global variable is to be used outside the function, then the keyword global should be prefixed to the variable name in the function

Flow of Execution :

Flow of execution can be defined as the order in which the statements in

a program are executed. The Python interpreter starts executing the instructions in a program from the first statement. The statements are executed one by one, in the order of appearance from top to bottom. When the interpreter encounters a function definition, the statements inside the function are not executed until the function is called. Later, when the interpreter encounters a function call, there is a little deviation in the flow of execution. In that case, instead of going to the next statement, the control jumps to the called function and executes the statement of that function. After that, the control comes back the point of function call so that the remaining statements in the program can be executed. Therefore, when we read a program, we should not simply read from top to bottom. Instead, we should follow the flow of control or execution. It is also important to note that a function must be defined before its call within a program.

Example 1:

```
 [2] def Greetings(Name):
 [3]      print("Hello "+Name)
 [1] Greetings("John")
 [4] print("Thanks")
```

Example 2:

```
 [4]          def RectangleArea(l,b):
 [5]               return l*b
 [1]          l = input("Length: ")
 [2]          b = input("Breadth: ")
 [3][6]       Area = RectangleArea(l,b)
 [7]          print(Area)
[8]           print("thanks")
```

```
helloPython()                    #Function Call
def helloPython():                 #Function definition
    print("I love Programming")
```

On executing the above code the following error is produced:

```
Traceback (most recent call last):
File "C:\Users\\Prog 7-11.py", line 3, in <module>
   helloPython()                    #Function Call
 NameError: name 'helloPython' is not defined
```

The error ‘function not defined’ is produced even though the function has been defined. When a function call is encountered, the control has to

jump to the function definition and execute it. In the above program, since the function call precedes the function definition, the interpreter does not find the function definition and hence an error is raised.

That is why, the function definition should be made before the function call as shown below:

```
def helloPython(): #Function definition
        print("I love Programming")
helloPython()      #Function Call
```

Name Resolution (Scope Resolution)

- For every name used within program python follows name resolution rules known as **LEGB** rule.
- **(i) LOCAL : first check whether name is in local environment, if yes Python uses its value otherwise moves to (ii)**
- **(ii) ENCLOSING ENVIRONMENT: if not in local, Python checks whether name is in Enclosing Environment, if yes Python uses its value otherwise moves to (iii)**
- **GLOBAL ENVIRONMENT: if not in above scope Python checks it in Global environment, if yes Python uses it otherwise moves to (iv)**
- **BUILT-IN ENVIRONMENT: if not in above scope, Python checks it in built-in environment, if yes, Python uses its value otherwise Python would report the error:**
- name <variable> not defined

Mutability and Immutability of arguments/parameters and function call

```
def check(num):
    print('Value in check() is ',num)
    print('Id of num in check() is ', id(num))
    num+=10
    print('Value in check() is ',num)
    print('Id of num in check() is ', id(num))
myvalue=100
print('Value of myvalue is ',myvalue)
print('Id of myvalue  is ', id(myvalue))
check(myvalue)
print('Value of myvalue is ',myvalue)
print('Id of myvalue  is ', id(myvalue))
```

PROGRAM USING FUNCTIONS

1. WAF to check a number is even or odd, if it is even return true, else return false
2. WAF to display the smallest of 3 numbers
3. WAF to display first 'n' even numbers in ascending order
4. WAF to display first 'n' odd numbers in descending order.
5. WAF to convert a binary number to decimal number

PRACTICE QUESTIONS

	STATE TRUE OR FALSE
1.	Function makes a program more readable and reduces the program size?
2.	In python functions can return only one value.
3.	Actual parameters are the parameters specified within a pair of parentheses in the function definition'
4.	Python passes parameters by value.
5.	Value returning functions should be generally called from inside an expression.
6.	Function header and function definition is same thing.
7.	In Python, Keyword arguments are available in function definition.
8.	You can call a function only once after defining .
9.	The following is a valid function definition (T/F)?_____ def Disp(x=0,y): print(x,y)
10.	User can change the functionality of a built in functions.
11.	The variable declared outside a function is called a global variable.
12.	The following code is a valid code (T/F)_____? def Disp(sub1,sub2): print(sub1,sub2) Disp(sub=100,sub2=89) #Calling

13.	The default valued parameter specified in the function header becomes optional in the function calling statement.
14.	The following Python code is a example of Positional argument (T/F) def Swap(x,y): x,y=y,x p=90 q=78 Swap(p,q)#Calling
15.	Default parameters can be skipped in function call?
16.	Variable defined inside functions cannot have global scope?
17.	A python function may return multiple values?
18.	Positional arguments can follow keyword arguments?

	ASSERTION & REASONING
1.	**A:** The function header „def read (a=2, b=5, c):‟ is not correct. **R:** Non default arguments can‟t follow default arguments.
2.	A function code is given as follows: def study (num = 5): print(num + 5) **A:** We can call the above function either by statement 'study(7)' or 'study( )'. **R:** As the function contains default arguments, it depends on the caller that the above function can be called with or without the value.
3.	**A:** len( ), type( ), int( ), input( ) are the functions that are always available for use. **R:** Built in functions are predefined functions that are always available for use. For using them we don‟t need to import any module.
4.	**A:** def Disp(x,y,z): print(x+y+z) z=10 Disp(x=67,y=78,z) **R:** The above code depicts the concept of keyword argument. But, during execution it will show an error.
5.	**A:** Every function returns a value if the function does not explicitly return a value, then it will return „Zero‟. **R:** Zero is equivalent to None.
6.	**A:** A function is a block of organized and reusable code that is used to perform a single related action. **R:** Function provides better modularity for your application and a high degree of code reusability.
7.	Consider the following code: **x = 100** **def study( ):** **global x** **x = 50** **print(x)** **study()** **A:** 50 will be the output of above code. **R:** Because the x used inside the function study( ) is of local scope.

8.	**A:** The number of actual parameters in a function call may not be equal to the number of formal parameters of the function. **R:** During a function call, it is optional to pass the values to default parameters
9.	**A:** The function definition calculate (a, b, c=1, d) will give error. **R:** All the non-default arguments must precede the default arguments.
10.	**A:** Key word arguments are related to the function calls **R:** When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.
11.	**A:** The default arguments can be skipped in the function call. **R:** The function argument will take the default valueseven if the values are supplied in the function call
12.	**A:** global keyword is used inside a function to enable the function alter the value of a global variable. **R:** A global variable cannot be accessed by a function without the use of global keyword.
13.	**A:** Keyword arguments were the named arguments with assigned values being passed in the function call. **R:** Default arguments cannot be skipped while calling the function while keyword arguments are in function call statement.
14.	**A:** Keyword arguments are related to function calls. **R:** When you use keyword arguments in a function call, the caller identifies the arguments by the values passed.
15.	**A:** If the arguments in function call statement match the number and order of arguments as defined in the function definition, such arguments are called positional arguments. **R:** During a function call, the argument list first contains default argument(s) followed by positional argument(s).
16.	**A:** A function header has been declared as **def MYFUN(A, B=30)**: The function call MYFUN(20, 60, B=80) will give an error. **R:** During a function call, the same parameter should not be provided two values.

17.	**A:** The writerow function of csv module takes a list having length equal to the number of columns in CSV file. **R:** The data inside a CSV file is stored in the form of a string.
18.	**A:** Python allows function arguments to have default values; if the function is called without the argument, the argument gets its default value. **R:** During a function call, the argument list first contains default argument(s) followed by positional argument(s).
19.	**A:** A CSV file stores data in rows and the values in each row is separated with a separator, also known as a delimiter. **R:** You cannot change the by default comma as a value separator.
20.	**A:** Global variable is declared outside the all the functions. **R:** It is accessible through out all the functions.
21.	**A:** If the arguments in function call statement match the number and order of arguments as defined in the function definition, such arguments are called positional arguments. **R:** During a function call, the argument list first contains default argument(s) followed by positional argument(s).
22.	**A:** A variable is still valid if it is not defined inside the function. The values defined in global scope can be used. **R:** Python used LEGB rule to resolve the scope of a variable.
23.	**A:** When passing a mutable sequence as an argument, function modifies the original copy of the sequence. **R:** Function can alter mutable sequences passes to it.
24.	**A:** Key word arguments are related to function calls. **R:** When you use keyword arguments in funation call, the caller identifies the arguments by the parameter name.
25.	**A:** Keyword arguments are related to the function calls **R:** When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.

	OBJECTIVE TYPE QUESTIONS (MCQ)
1.	When large programs are broken down into smaller units known as… (a) sub program (b) functions (c) class (d) None of the above
2.	Select which of the following is NOT TRUE for Python function: (a) A function only executes when it is called and we can reuse it in a program. (b) In a python function a keyword argument can not be followed by a positional argument. (c) A Python function can return multiple values. (d) Python function doesn"t return anything unless and until you add a return statement.
3.	Which of the following is a valid function name? (a) start_function( ) (b) start function( ) (c) start-function( ) (d) All of the above
4.	Which of the following functions header is correct? (a) def study(a=2, b=5, c) : (b) def study(a=2, b, c=5) : (c) def study(a, b=2, c=5): (d) None of the above
5.	The values being passed through function-call statement are called_____. (a) Arguments (b) Parameter (c) Values (d) None of these
6.	These are predefined functions that are always available for use. For using them we don"t need to import any module. (a) Built in function (b) Predefined function (c) User defined (d) None of the above
7.	Aman wants to write a function in python. But he doesn"t know how to start with it! Select the keyword used to start a function out of the following: (a) function (b) start (c) def (d) fun
8.	How many values can be returned by a function in python? (a) 0 (b) 1 (c) more than one (d) 2
9.	Python passes arguments to function by (a) Reference (b) Value (c) Both (d) None of the above
10.	Which of the following is not a part of the python function? (a) function header (b) return statement (c) parameter list (d) function keyword
11.	A variable created or defined in a function body is known as… (a) local (b) global (c) built-in (d) instance

12.	What will be output of following code: X = 50 def funct(X): X = 2 funct (X) print("X is now:", X) a) X is now: 50 b) X is now: 2 c) Error d) None of the above
13.	For a function header as follows: def cal(x,y=20): Which of the following function calls will give an error? (a) cal(15,25) (b) cal(x=15,y=25) (c) cal(y=25) (d) cal(x=25)
14.	The values received in function definition/header statement are called_____. (a) Arguments (b) Parameter (c) Values (d) None of these
15.	What will be the output of the following code: def calculate(a, b): return a+b, a-b res = calculate(7, 7) print(res) (a) (14,0) (b) [14, 0] (c) 14 (d) 0
16.	The_____of a variable is the area of a program where it may be referenced (a) External (b) Global (c) Scope (d) Local
17.	In which part of memory does the system stores the parameter and local variables of function call. (a) Heap (b) Stack (c) Both a and b (d) None of the above
18.	When you use multiple type argument in function then default argument take place_____ (a) At beginning (b) At end (c) Anywhere (d) None of the above
19.	A Function that does not have any return value is known as…….. (a) Library function (b) Void function (c) Fruitful function (d) None of the above
20.	Which of the following is not a part of the python function? (a) Function header (b) Return statement (c) Parameter list (d) Function keyword
21.	A variable created or defined within a function body is (a) local (b) global (c) built in (d) instance

22.	In_____arguments we can skip the default argument, but all the arguments should match the parameters in the function definitions. (a) keyword (b) required (c) default (d) None of the above.

23.	defmax_of_two(x,y):
	if x > y:
	return x
	else:
	return y
	defmax_of_three(x,y,z):
	return max_of_two(x, max_of_two(y,z))
	
	print(max_of_three(3, 6, -5))
	
	(a) 3 (b) 6 (c) -5 (d) error

24.	By default if you return multiple value separated by comma, then it is returned as (a) List (b) Tuples (c) String (d) None of the above.
25.	Find the output of following code : x=100 def study(x): global x x=50 print("Value of x is :", x) a. 100 b. 50 c. Error d. None of the above
26.	What will be the output of the following Python code? **def change(i = 1, j = 2):** **i = i + j** **j = j + 1** **print(i, j)** **change(j = 1, i = 2)** (a) An exception is thrown because of conflicting values (b) 1 2 (c) 3 3 (d) 3 2

27.	def Findoutput():
	  L="earn"
	  X=" "
	  L1=[]
	  Count=1
	  for i in L:
	  if i in ['a','e','I','o','u']:
	  X=X+i.swapcase()
	  else:
	  if(Count%2!=0):
	  X=X+str(len(L[:Count]))
	  else:
	  X=X+i
	  Count=Count+1
	  print(X)
	
	Findoutput()
	
	(a)EArn (b)EA3n
	(c)EA34 (d)EARN

28.	If a function doesn't have a return statement, which of the following does the function return? (a) int (b) null (c) None (d) An exception is thrown without the return statement
29.	The values being passed through a function call statements are called (a) Actual parameter (b) Formal parameter (c) default parameter (d) None of these
30.	How many types of arguments are there in function? (a) 1 (b) 2 (c) 3 (d) 4

	2 MARKS/3 MARKS
1.	(a) Differentiate between positional parameters and default Parameters with suitable example program for each. (b) How can a function return multiple values? Illustrate with an example program.

2.	**Rewrite the following program after finding the error(s)**
	DEF execmain():
	x = input("Enter a number:")
	if (abs(x)=x):
	print ("You entered a positive number")
	else:
	x=*-1
	print "Number made positive:"x
	execmain()

3.	Uma is a student of class XII. During examination, she has been assigned an incomplete python code (shown below). The code shows the variable scope in a program. Help her in completing the assigned code.
	……. fun(x, y): # Statement-1
	……. a # Statement-2
	a = 10
	x, y = ……. # Statement-3
	b = 20
	b = 30
	c = 30
	print (a, b, x, y) # Statement-4
	a, b, x, y = 1, 2, 3,4
	…….(50, 100) # Statement-5
	fun()
	print(a, b, x, y) # Statement-6
	1. Write the suitable keyword for blank space in the line marked as Statement-1.
	2. Write the suitable keyword to access variable a globally for blank space in the line marked as Statement-2.
	3. Write down the python to swap the values of x and y for the blank space in the line marked as Statement-3.
	4. Mention the output for the line marked as Statement-4.
	5. the missing code for the blank space in the line marked as Statement-5.

4.	**Give output of the following code:** def func(a, b=5, c=10): print('a is', a, 'and b is', b, 'and c is', c) func(3, 7) func(25, c = 24) func(c = 50, a = 100)

5. **Predict the output of the Python code given below:**

```
value = 50
def display(N):
    global value
    value = 25
    if N%7==0:
      value = value + N
    else:
       value = value - N
    print(value, end="#")

display(20)
print(value)
```

6. Consider the program given below and justify the output.

```
c = 10
def add():
    global c
    c = c + 2
    print("Inside add():", c)

add()
c=15
print("In main:", c)
```

Output:
Inside add() : 12
In main: 15
What is the output if "global c" is not written in the function add()?

7. **Predict the output of the following code:**

```
def L1(D):
   CNT = 3
   TOTAL = 0
   for C in [7,5,4,6]:
      T = D[CNT]
      TOTAL = float (T) + C
      print(TOTAL)
      CNT-=1
TXT = ["20","50","30","40"]
L1(TXT)
```

8. (a) Differentiate between default parameter(s) and keyword parameter(s) with a suitable example for each.
(b) What is the difference between Local Variable and Global Variable?

9.	L = 5
	B = 3
	def getValue():
	global L, B
	L = 10
	B = 6
	def findArea():
	Area = L * B
	print("Area = ", Area)
	getValue()
	findArea()

10.	**Rewrite the code after correcting it and underline the corrections.**
	def sum(arg1,arg2):
	total=arg1+arg2;
	print(”Total:”,total)
	return total;
	sum(10,20)
	print("Total:",total)

11.	Write a function DIVI_LIST() where NUM_LST is a list of numbers passed as argument to the function. The function returns two list D_2 and D_5 which stores the numbers that are divisible by 2 and 5 respectively from the NUM_LST. Example: NUM_LST=[2,4,6,10,15,12,20] D_2=[2,4,6,10,12,20] D_5=[10,15,20]

12.	**Predict the output of the following:**
	def Facto(x):
	a=None
	b=None
	for i in range(2,x//2):
	if x%i==0:
	if a is None:
	a = i
	else:
	b = i
	break
	return a,b
	S=Facto(4)
	print(S)

13.	**Rewrite the code after correcting it and underline the corrections.** Def swap(d): n = { } values = d.values() keys = list(d.keys[]) k = 0 for i in values n(i) = keys[k] k=+1 return n result = swap({„a":1,"b":2,"c":3}) print(result)

14.	**Predict the output of the following:**
	def fun(s):
	k=len(s)
	m=""
	for i in range(0,k):
	if(s[i].isupper()):
	m=m+s[i].lower()
	elif s[i].isalpha():
	m=m+s[i].upper()
	else:
	m=m+'bb'
	print(m)
	fun('school2@com')

15.	**Predict the output of the following code:** def runme(x=1, y=2): x = x+y y+=1 print(x, '$', y) return x,y a,b = runme() print(a, '#', b) runme(a,b) print(a+b)
16.	Write the difference between Actual Parameter and formal parameter. Give an example.

17.	**Predict the output of the following:**
	def ChangeVal(M,N):
	for i in range(N):
	if M[i]%5 == 0:
	M[i] //= 5
	if M[i]%3 == 0:
	M[i] //= 3
	L=[25,8,75,12]
	ChangeVal(L,4)
	for i in L :
	print(i, end='#')

18.	**Predict the output of the following:** R = 0 def change(A , B) : global R A += B R +=3 print(R , end='%') change(10 , 2) change(B=3 , A=2)

19.	**Predict the output of the following code:**
	def result(s):
	n = len(s)
	m=''
	for i in range(0, n):
	if (s[i] >= 'a' and s[i] <= 'm'):
	m = m + s[i].upper()
	elif (s[i] >= 'n' and s[i] <= 'z'):
	m = m + s[i-1]
	elif (s[i].isupper()):
	m = m + s[i].lower()
	else:
	m = m + '#'
	print(m)
	result('Cricket') #Calling

20.	Write a function Interchange (num) in Python, which accepts a list num of integers, and interchange the adjacent elements of the list and print the modified list as shown below: (Number of elements in the list is assumed as even) Original List: num = [5,7,9,11,13,15] After Rearrangement num = [7,5,11,9,15,13]

21.	**Rewrite the corrected code and underline each correction.**
	def Tot (Number):
	Sum=0
	for C in RANGE (1, Number + 1):
	Sum + = C
	return Sum
	print(Tot [3])

22.	**Give Output of :** def Change (P, Q = 30) : P = P + Q Q = P - Q print (P,"@",Q) return P R =150 S= 100 R=Change(R, S) print(R,"@",S) S=Change (S)
23.	Write a function **sumcube(L)** to test if an element from list L is equal to the sum of the cubes of its digits i.e. it is an "Armstrong number". Print such numbers in the list. If L contains [67,153,311,96,370,405,371,955,407] The function should print 153,370,371,407

24.	**Predict the output of the following:**
	def Bigger(N1,N2):
	if N1>N2:
	return N1
	else:
	return N2
	L=[32,10,21,54,43]
	for c in range (4,0,-1):
	a=L[c]
	b=L[c-1]
	print(Bigger(a,b),'@', end=' ')'

25.	**What will be the output of following Python Code:**
	def change(num):
	for x in range(0,len(num),2):
	num[x], num[x+1]=num[x+1], num[x]
	data=[10,20,30,40,50,60]
	change(data)
	print(data)

26. **Predict the output of the following code:**

```
def OUTER(Y,ch):
    global  X,NUM
    Y=Y+X
    X=X+Y
    print(X,"@",Y)
    if ch==1:
       X=inner_1(X,Y)
       print(X,"@",Y)
    elif ch==2:
       NUM=inner_2(X,Y)

 def inner_1(a,b):
    X=a+b
    b=b+a
    print(a,"@",b)
    return a

 def inner_2(a,b):
    X=100
    X=a+b
    a=a+b
    b=a-b
    print(a,"@",b)
    return b

X,NUM=100,1
OUTER(NUM,1)
OUTER(NUM,2)
print(NUM,"@",X)
```

27. **Predict the output of the following:**

```
def Compy(N1,N2=10):
     return N1 > N2

NUM= [10,23,14,54,32]
for VAR in range (4,0,-1):
     A=NUM[VAR]
     B=NUM[VAR-1]
     if VAR >len(NUM)//2:
          print(Compy(A,B),'#', end=' ')
    else:
       print(Compy(B),'%',end='   ')
```

28.	**Predict the output of the following**:
	p=8
	def sum(q,r=5):
	  global p
	  p=(r+q)**2
	  print(p, end= '#')
	a=2;
	b=5;
	sum(b,a)
	sum(r=3,q=2)

29.	Mr.Raja wants to print the city he is going to visit and the distance to reach that place from his native. But his coding is not showing the correct: output debug the code to get the correct output and state what type of argument he tried to implement in his coding. **def Travel(c,d)** ** print("Destination city is ",city)** ** print("Distance from native is ",distance)** **Travel(distance="18 KM",city="Tiruchi")**

30.	**Predict the output of the following:**
	value = 50
	def display(N):
	  global value
	  value = 25
	  if N%7==0:
	  value = value + N
	  else:
	  value = value - N
	  print(value, end="#")
	
	display(20)#Calling
	print(value)

31.	**Predict the output of the following code?**
	def my_func(a=10,b=30):
	  a+=20
	  b-=10
	  return a+b,a-b
	print(my_func(a=60)[0],my_func(b=40)[1])

32.	**Predict the output of the following code:** def f(): global s s += ' Is Great' print(s) s = "Python is funny" s = "Python" f() print(s)
33.	Write a function listchange(Arr,n)in Python, which accepts a list Arr of numbers and n is an numeric value depicting length of the list. Modify the list so that all even numbers doubled and odd number multiply by 3 Sample Input Data of the list: Arr= [10,20,30,40,12,11], n=6 Output: Arr = [20,40,60,80,24,33]
34.	**Predict the output of the following:** def func(b): global x print('Global x=', x) y=x + b x=7 z=x-b print('Local x =',x) print('y=',y) print('z=',z) x=3 func(5)
35.	Write a function LShift(Arr,n) in Python, which accepts a list Arr of numbers and n is a numeric value by which all elements of the list are shifted to left. Sample Input Data of the list Arr= [10,20,30,40,12,11], n=2 Output Arr = [30,40,12,11,10,20]

36.	**Predict the output of the following:**
	def change(A):
	S=0
	for i in range(len(A)//2):
	S+=(A[i]*2)
	return S
	B = [10,11,12,30,32,34,35,38,40,2]
	C = change(B)
	print('Output is',C)

37.	**Write the output for the following python code:**
	def Change_text(Text):
	  T=" "
	  for K in range(len(Text)):
	  if Text[K].isupper():
	  T=T+Text[K].lower();
	  elif K%2==0:
	  T=T+Text[K].upper()
	  else:
	  T=T+T[K-1]
	  print(T)
	
	Text="Good go Head"
	Change_text(Text)

38.	Write a function called letter_freq(my_list) that takes one parameter, a list of strings(mylist) and returns a dictionary where the keys are the letters from mylist and the values are the number of times that letter appears in the mylist, e.g.,if the passed list is as: **wlist=list("aaaaabbbbcccdde")** then it should return a dictionary as{„a":5,"b":4,"c":3,"d":2,"e":1}

39.	**Write the output for the following python code:**
	def Quo_Mod (L1):
	  L1.extend([33,52])
	  for i in range(len(L1)):
	  if L1[i]%2==0:
	  L1[i]=L1[i] /5
	  else:
	
	  L1[i]=L1[i]%10
	L=[100,212,310]
	print(L)
	Quo_Mod(L)
	print(L)

40.	**Predict the output for the following code:**
	def test(i, a =[]):
	  a.append(i)
	  return a
	
	test(25)
	test(32)
	s = test(17)
	print(s)

41.	**Write the output given by following Python code.**
	x=1
	def fun1():
	  x=3
	  x=x+1
	  print(x)
	
	def fun2():
	  global x
	  x=x+2
	  print(x)
	fun1()
	fun2()

42.	Write a function in Shift(Lst), Which accept a List „Lst" as argument and swaps the elements of every even location with its odd location and store in different list eg. if the array initially contains 2, 4, 1, 6, 5, 7, 9, 2, 3, 10 then it should contain 4, 2, 6, 1, 7, 5, 2, 9, 10, 3

43.	**Predict the output of the following:**
	def changer(p, q=10):
	  p=p/q
	  q=p%q
	  print(p,'#',q)
	  return p
	a=200
	b=20
	a=changer(a,b)
	print(a,'$',b)
	a=changer(a)
	print(a,'$',b)

44. **Determine the output of the following code fragments:**

```
def determine(s):
    d={"UPPER":0,"LOWER":0}
    for c in s:
        if c.isupper( ):
            d["UPPER"]+=1
        elif c.islower( ):
            d["LOWER"]+=1
        else:
            pass
print("Original String:",s)
print("Upper case count:", d["UPPER"])
print("Lower case count:", d["LOWER"])
determine("These are HAPPY Times")
```

45. Write a function in Display which accepts a list of integers and its size as arguments and replaces elements having even values with its half and elements having odd values with twice its value . eg: if the list contains 5, 6, 7, 16, 9 then the function should rearranged list as 10, 3,14,8, 18

46. **Predict the output of the code given below:**

```
def Convert(Old):
    l=len(Old)
    New=" "
    for i in range(0,1):
        if Old[i].isupper():
            New=New+Old[i].lower()
elif Old[i].islower():
            New=New+Old[i].upper()
elif Old[i].isdigit():
            New=New+"*"
        else:
            New=New+"%"
    return New

Older="InDIa@2022"
Newer=Convert(Older)
print("New String is: ", Newer)
```

47. Define a function **ZeroEnding(SCORES)** to add all those values in the list of SCORES, which are ending with zero (0) and display the sum.

For example :

If the SCORES contain [200, 456, 300, 100, 234, 678]
The sum should be displayed as 600

48.	Write a Python function SwitchOver(Val) to swap the even and odd positions of the values in the list Val. Note : Assuming that the list has even number of values in it. For example : If the list Numbers contain [25,17,19,13,12,15] After swapping the list content should be displayed as [17,25,13,19,15,12]

49.	**Give output of the following program:**
	def div5(n):
	if n%5==0:
	return n*5
	else:
	return n+5
	def output(m=5):
	for i in range(0,m):
	print(div5(i),'@',end=" ")
	print('\n')
	output(7)
	output()
	output(3)

50.	Write a function INDEX_LIST(L), where L is the list of elements passed as argumentto the function. The function returns another list named „indexList" that stores theindices of all Non-Zero Elements of L. For example: If L contains [12,4,0,11,0,56] The index List will have - [0,1,3,5]

51.	**Predict the output of the Python code given below:**
	def Alpha(N1,N2):
	if N1>N2:
	print(N1%N2)
	else:
	print(N2//N1,'#',end=' ')
	
	NUM=[10,23,14,54,32]
	for C in range (4,0,-1):
	A=NUM[C]
	B=NUM[C-1]
	Alpha(A,B)

52.	Write a function INDEX_LIST(S), where S is a string. The function returnsa list named indexList„ that stores the indices of all vowels of S. For example: If S is "Computer", then indexList should be [1,4,6]

53.	Write a function INDEX_LIST(L), where L is the list of elements passed as argument to thefunction. The function returns another list named „indexList" that stores the indices of allElements of L which has a even unit place digit. For example: If L contains [12,4,15,11,9,56] The indexList will have - [0,1,5]
54.	Rewrite the code after removing allthe syntactical errors, underlining each correction: checkval def(): x = input("Enter a number") if x % 2 == 0 print (x, "is even") else; print (x, "is odd")
55.	Write a function in python named SwapHalfList(Array), which acceptsa list Array of numbers and swaps the elements of 1st Half of the listwith the 2nd Half of the list, ONLY if the sum of 1st Half is greaterthan 2nd Half of the list. Sample Input Data of the list Array= [100, 200, 300, 40, 50, 60], Output Array = [40, 50, 60, 100, 200, 300]

****