Solved Problem

1. *What is the significance of having functions in a program ?*

Solution. Creating functions in programs is very useful. It offers following advantages :

(*i*) *The program is easier to understand.*

Main block of program becomes compact as the code of functions is not part of it, thus is easier to read and understand.

(*ii*) *Redundant code is at one place, so making changes is easier.*

Instead of writing code again when we need to use it more than once, we can write the code in the form of a function and call it more than once. If we later need to change the code, we change it in one place only. Thus it saves our time also.

(*iii*) *Reusable functions can be put in a library in modules.*

We can store the reusable functions in the form of modules. These modules can be imported and used when needed in other programs.

2. *From the program code given below, identify the parts mentioned below :*

```
def processNumber(x):
    x = 72
    return x + 3

y = 54
res = processNumber(y)
```

Identify these parts : *function header, function call, arguments, parameters, function body, main program*

Solution.

Function header	:	`def processNumber(x) :`
Function call	:	`processNumber(y)`
Arguments	:	`y`
Parameters	:	`x`
Function body	:	`x = 72` `return x + 3`
Main program	:	`y = 54` `res = processNumber(y)`

3. *Trace the following code and predict output produced by it.*

```
1.  def power(b, p) :
2.      y = b ** p
3.      return y
4.
5.  def calcSquare(x) :
6.      a = power(x, 2)
7.      return a
8.
```

```
9.  n = 5
10. result = calcSquare(n) + power(3, 3)
11. print(result)
```

Solution. Flow of execution for above code will be :

$1 \rightarrow 5 \rightarrow 9 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 10 \rightarrow 11$

The output produced by above code will be :

```
52
```

4. *Trace the flow of execution for following programs* :

(*a*)
```
1   def power(b, p):
2       r = b ** p
3       return r
4
5   def calcSquare(a):
6       a = power(a, 2)
7       return a
8
9   n = 5
10  result = calcSquare(n)
11  print (result)
```

(*b*)
```
1. def increment(x) :
2.     x = x + 1
3.
4. # main program
5. x = 3
6. print(x)
7. increment(x)
8. print(x)
```

(*c*)
```
1.  def increment(x):
2.      z = 45
3.      x = x + 1
4.      return x
5.
6.  # main
7.  y = 3
8.  print(y)
9.  y = increment(y)
10. print(y)
11. q = 77
12. print(q)
13. increment(q)
14. print(q)
15. print(x)
16. print(z)
```

Solution.

(*a*) $1 \rightarrow 5 \rightarrow 9 \rightarrow 10 \rightarrow 5 \rightarrow 6 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 7 \rightarrow 10 \rightarrow 11$

(*b*) $1 \rightarrow 5 \rightarrow 6 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 8$

Control didnot return to function call statement (7) as nothing is being returned by increment( )

(*c*) $1 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 13 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 14 \rightarrow 15 \rightarrow 16$

5. *What is the difference between the formal parameters and actual parameters? What are their alternative names? Also, give a suitable Python code to illustrate both.*

Solution. ***Actual Parameter*** is a parameter, which is used in *function call* statement to send the value from *calling function* to the *called function*. It is also known as ***Argument***.

Formal Parameter is a parameter, which is used in *function header* of the *called function* to receive the value from *actual parameter*. It is also known as ***Parameter***.

For example,

```
def addEm(x, y, z):
    print(x + y + z)

addEm(6, 16, 26)
```

In the above code, *actual parameters* are ***6***, ***16*** and ***26*** ; and *formal parameters* are ***x***, ***y*** and ***z***.

6. *Consider a function with following header :*

```
def info(object, spacing = 10, collapse = 1):
```

Here are some function calls given below. Find out which of these are correct and which of these are incorrect stating reasons :

```
a. info( obj1 )
b. info(spacing = 20)
c. info( obj2, 12 )
d. info( obj11, object = obj12)
e. info( obj3, collapse = 0)
f. info( )
g. info(collapse = 0, obj3 )
e. info( spacing = 15, object = obj4 )
```

Solution.

(*a*)	Correct	***obj1*** is for positional parameter **object** ; **spacing** gets its default value of *10* and **collapse** gets its default value of *1*.
(*b*)	Incorrect	Required positional argument (object) missing ; required arguments cannot be missed.
(*c*)	Correct	Required parameter **object** gets its value as ***obj2*** ; **spacing** gets value *12* and for skipped argument **collapse**, default value *1* is taken.
(*d*)	Incorrect	Same parameter object is given multiple values – one through positional argument and one through keyword(named) argument.
(*e*)	Correct	Required parameter **object** gets its value as ***obj3*** ; **collapse** gets value *0* and for skipped argument **spacing**, default value *10* is taken.
(*f*)	Incorrect	Required parameter **object's** value cannot be skipped.
(*g*)	Incorrect	Positional arguments should be before keyword arguments.
(*h*)	Correct	Required argument **object** gets its value through a keyword argument.

7. *What is the default return value for a function that does not return any value explicitly ?*

(*a*) None (*b*) int (*c*) double (*d*) null

Solution. (*a*)

8. *What will following code print ?*

```
def addEm(x, y, z):
    print(x + y + z)

def prod (x, y, z) :
    return x * y * z

a = addEm(6, 16, 26)
b = prod ( 2, 3, 6)
print(a, b)
```

Solution.

```
None 36
```

9. *In the previous question's code, identify the void and non-void functions. The previous code stores the return values of both void and non-void functions in variables. Why did Python not report an error when void functions do not return a value?*

Solution.

Void function : addEm( )

Non-void function : prod( )

In Python, void functions do not return a value; rather they report the absence of returning value by returning None, which is legal empty value in Python. Thus, variable ***a*** stores **None** and it is not any error.

10. *Consider below given function headers. Identify which of these will cause error and why ?*

(*i*) `def func(a = 1, b):`

(*ii*) `def func(a = 1, b, c = 2):`

(*iii*) `def func(a = 1, b = 1, c = 2):`

(*iv*) `def func(a = 1, b = 1, c = 2, d):`

Solution. Function headers (*i*), (*ii*) and (*iv*) will cause error because **non-default arguments cannot follow default arguments**.

Only function header (*ii*) will not cause any error.

11. *What is the difference between a local variable and a global variable ? Also, give a suitable Python code to illustrate both.*

Solution. The differences between a local variable and global variable are as given below :

	Local Variable	Global Variable
1.	It is a variable which is declared within a function or within a block	It is variable which is declared outside all the functions
2.	It is accessible only within a function/block in which it is declared	It is accessible throughout the program

For example, in the following code, ***x, xCubed*** are global variables and ***n*** and ***cn*** are local variables.

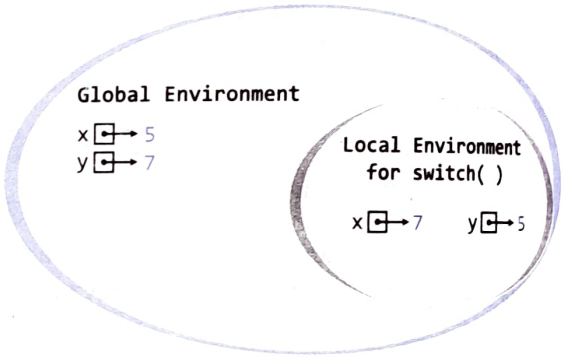
```
def cube(n):
    cn = n * n * n
    return cn

x = 10
xCubed = cube(x)
print(x,"cubed is", xCubed)
```

12. *Following code intends to swap the values of two variables through a function, but upon running the code, the output shows that the values are swapped inside the switch() function but back again in main program, the variables remain un-swapped. What could be the reason? Suggest a remedy.*

```
def switch(x, y):
    x, y = y, x
    print("inside switch :", end = ' ')
    print("x =", x, "y =", y)

x = 5
y = 7
print("x =", x, "y =", y)
switch(x, y)
print("x =", x, "y =", y)
```



Solution. The reason for un-reflected changes in the main program is that although both main program and *switch( )* have variables with same names *i.e., x* and *y*, but their scopes are different as shown in the adjacent figure.

The scope of ***x*** and ***y*** of *switch( )* is local. Though they are swapped in the namespace of *switch( )* but their namespace is removed as soon as control returns to main program. The global variables ***x*** and ***y*** remain unchanged as *switch( )* worked with a different copy of values not with original values.

The remedy of above problem is that the *switch( )* gets to work with global variables so that changes are made in the global variables. This can be done with the help of global statement as shown below:

```
def switch(x, y):
    global x, y
    x, y = y, x
    print("inside switch :", end = ' ')
    print("x =", x, "y =", y)

x = 5
y = 7
print("x =", x, "y =", y)
switch (x, y)
print("x =", x, "y=", y)
```

Now the above program will be able to swap the values of variables through switch( ). (Though, now passing parameter is redundant.)

13. *Following code intends to add a given value to global variable a. What will the following code produce?*

```
1. def increase(x) :
2.     a = a + x
3.     return
4.
5. a = 20
6. b = 5
7. increase(b)
8. print(a)
```

Solution. The above code will produce an error.

The reason being whenever we assign something to a variable inside a function, the variable is created as a local variable. Assignment creates a variable in Python.

Thus at line 2, when variable ***a*** is incremented with the passed value ***x***, Python tries to create a local variable ***a*** by assigning to it the value of expression on the right hand side of assignment. But variable ***a*** also appears in the right hand side of assignment, which results in error because **a** is undeclared so far in function.

To assign some value to a global variable from within a function, without creating a local variable with the same name, global statement can be used. So, if we add

```
global a
```

in the first line of function body, the above error will be corrected. Python won't create a local variable ***a***, rather will work with global variable ***a***.

14. *Which names are local, which are global and which are built-in in the following code fragment?*

```
invaders = 'Big names'
pos = 200
level = 1

def play( ) :
    max_level = level + 10
    print(len (invaders ) == 0)
    return max_level


res = play()
print(res)
```

Solution.

Global names	:	invaders, pos, level, res
Local names	:	max_level
Built in	:	len

15. *Predict the output of the following code fragment ?*

```
def func(message, num = 1):
    print(message * num)

func('Python')
func('Easy', 3)
```

Solution.

```
Python
EasyEasyEasy
```

16. *Predict the output of the following code fragment ?*

```
def check(n1 = 1, n2 = 2):
    n1 = n1 + n2
    n2 += 1
    print(n1, n2)

check()
check(2, 1)
check(3)
```

Solution.

```
3 3
3 2
5 3
```

17. *What is the output of the following code?*

```
a = 1
def f ( ) :
    a = 10
print(a)
```

Solution. The code will print 1 to the console.

18. *What will be the output of following code?*

```
def interest (prnc, time =2 , rate = 0.10) :
    return (prnc * time * rate)

print(interest (6100, 1))
print(interest (5000, rate = 0.05))
print(interest (5000, 3, 0.12 ))
print(interest (time = 4, prnc = 5000))
```

Solution.

```
610.0
500.0
1800.0
2000.0
```

19. *Is return statement optional ? Compare and comment on the following two return statements :*

```
return
return val
```

Solution. The return statement is optional ONLY WHEN the function is *void* or we can say that when the function does not return a value. A function that returns a value, must have at least one *return* statement.

From given two return statements, statement

```
return
```

is not returning any value, rather it returns the control to caller along with empty value *None*. And the statement

```
return val
```

is returning the control to caller along with the value contained in variable *val*.

20. *Write a function that takes a positive integer and returns the one's position digit of the integer.*

Solution.

```
def getOnes(num):
    # return the ones digit of the integer num
    onesDigit = num % 10
    return onesDigit
```

21. *Write a function that receives an octal number and prints the equivalent number in other number bases i.e., in decimal, binary and hexadecimal equivalents.*

Solution.

```
def oct2others( n) :
    print("Passed octal number :", n)
    numString = str(n)
    decNum = int( numString, 8)
    print("Number in Decimal :", decNum)
    print("Number in Binary :", bin(decNum))
    print("Number in Hexadecimal :", hex(decNum))

num = int(input("Enter an octal number :"))
oct2others(num)
```

Please recall that bin( ) and hex( ) do not return numbers but return the string-representations of equivalent numbers in binary and hexadecimal number systems respectively.

22. *Write a program that generates 4 terms of an AP by providing initial and step values to a function that returns first four terms of the series.*

Solution.

```
def retSeries(init, step ):
    return init, init+step, init+2*step, init+3*step

ini = int(input("Enter initial value of the AP series :"))
st = int(input("Enter step value of the AP series :"))
print("Series with initial value", ini, "& step value", st, "goes as:")
t1, t2, t3, t4 = retSeries(ini, st)
print(t1, t2, t3, t4)
```

GLOSSARY

Argument	A value provided to a function in the function call statement.
Flow of execution	The order of execution of statements during a program run.
Parameter	A name used inside a function to refer to the value which was passed to it as an argument.
Function	Named subprogram that acts on data and often returns a value.
Actual Argument	Argument
Actual Parameter	Argument
Formal Parameter	Parameter
Formal Argument	Parameter
Scope	Program part(s) in which a particular piece of code or a data value (e.g., variable) can be accessed.

Assignment

Type A : Short Answer Questions/Conceptual Questions

1. A program having multiple functions is considered better designed than a program without any functions. Why ?
2. What all information does a function header give you about the function ?
3. What do you understand by flow of execution ?
4. What are arguments ? What are parameters ? How are these two terms different yet related ? Give example.
5. What is the utility of :
   (*i*) default arguments,
   (*ii*) keyword arguments ?
6. Explain with a code example the usage of default arguments and keyword arguments.
7. Describe the different styles of functions in Python using appropriate examples.
8. Differentiate between fruitful functions and non-fruitful functions.
9. Can a function return multiple values ? How ?
10. What is scope ? What is the scope resolving rule of Python ?
11. What is the difference between local and global variables ?
12. When is *global* statement used ? Why is its use not recommended ?
13. Write the term suitable for following descriptions :
    (*a*) A name inside the parentheses of a function header that can receive a value.
    (*b*) An argument passed to a specific parameter using the parameter name.
    (*c*) A value passed to a function parameter.
    (*d*) A value assigned to a parameter name in the function header.
    (*e*) A value assigned to a parameter name in the function call.
    (*f*) A name defined outside all function definitions.
    (*g*) A variable created inside a function body.

Type B : Application Based Questions

1. What are the errors in following codes ? Correct the code and predict output :

(a)
```
total = 0;
def sum( arg1, arg2 ):
total = arg1 + arg2;
print("Total :", total)
return total;
sum( 10, 20 );
print("Total :", total)
```

(b)
```
def Tot(Number)      #Method to find Total
    Sum = 0
    for C in Range (1, Number + 1) :
        Sum += C
    RETURN Sum
print (Tot[3])       #Function Calls
print (Tot[6])
```

[CBSE D 2015]

2. Consider the following code and write the flow of execution for this. Line numbers have been given for your reference.

```
1    def power(b, p):
2        y = b ** p
3        return y
4
5    def calcSquare(x):
6        a = power(x, 2)
7        return a
8
9    n = 5
10   result = calcSquare(n)
11   print(result)
```

3. What will the following function return ?

```
def addEm(x, y, z):
    print(x + y + z)
```

4. What will the following function print when called ?

```
def addEm(x, y, z):
  return x + y + z
  print(x + y + z)
```

5. What will be the output of following programs ?

(i)
```
num = 1
def myfunc():
    return num
print(num)
print(myfunc())
print(num)
```

(ii)
```
num = 1
def myfunc():
    num = 10
    return num
print(num)
print(myfunc())
print(num)
```

(iii)
```
num = 1
def myfunc():
    global num
    num = 10
    return num
print(num)
print(myfunc())
print(num)
```

(iv)
```
def display():
    print("Hello", end = ' ')
display()
print("there!")
```

6. Predict the output of the following code :

```
a = 10
y = 5

def myfunc():
    y = a
    a = 2
    print("y =", y, "a =", a)
    print("a + y =", a + y)
    return a + y

print("y =", y, "a =", a)
print(myfunc())
print("y =", y, "a =", a)
```

7. What is wrong with the following function definition ?

```
def addEm(x, y, z):
  return x + y + z
  print("the answer is", x + y + z)
```

8. Write a function namely fun that takes no parameters and always returns *None*.
9. Consider the code below and answer the questions that follow :

```
def multiply(number1,number2):
    answer = number1*number2
    print(number1, 'times',number2, '=', answer)

    return(answer)
output = multiply(5,5)
```

(*i*) When the code above is executed, what prints out ?

(*ii*) What is variable output equal to after the code is executed ?

10. Consider the code below and answer the questions that follow :

```
def multiply(number1, number2):
    answer = number1 * number2
    return(answer)
    print(number1,'times', number2, '=',answer )

output = multiply(5,5)
```

(*i*) When the code above is executed, what gets printed ?

(*ii*) What is variable output equal to after the code is executed ?

11. Find the errors in code given below :

(*a*)
```
def minus(total, decrement)
    output = total - decrement
    print(output)
    return (output)
```

(*b*)
```
define check()
    N = input ('Enter N: ')
    i = 3
    answer = 1 + i ** 4 / N
    Return answer
```

(c)
```
def alpha (n, string = 'xyz', k = 10) :
    return beta(string)
    return n

def beta (string)
    return  string == str(n)

print(alpha("Valentine's Day"):)
print(beta (string = ' true '))
print(alpha(n = 5, "Good-bye") :)
```

12. Draw the entire environment, including all user-defined variables at the time line 10 is being executed

```
1.   def sum(a, b, c, d ):
2.       result = 0
3.       result = result + a + b + c + d
4.       return result
5.
6.   def length():
7.       return 4
8.
9.   def mean(a, b, c, d):
10       return float(sum (a, b, c, d))/length()
11.
12.  print(sum(a, b, c,d), length(), mean(a, b, c, d))
```

13. Draw flow of execution for above program.
14. In the following code, which variables are in the same scope ?

```
def func1():
  a = 1
  b = 2
def func2():
  c = 3
  d = 4

e = 5
```

15. Write a program with a function that takes an integer and prints the number that follows after it. Call the function with these arguments :

    4, 6, 8, 2 + 1, 4 – 3 * 2, –3 –2

16. Write a program with non-void version of above function and then write flow of execution for both the programs.
17. What is the output of following code fragments ?

(i)
```
def increment(n):
    n.append([4])
    return n
L = [1, 2, 3]
M = increment(L)
print(L, M)
```

(*ii*)
```
def increment(n):
    n.append([49])
    return n[0], n[1], n[2], n[3]
L = [23, 35, 47]
m1, m2, m3, m4 = increment(L)
print(L)
print(m1, m2, m3, m4)
print(L[3] == m4)
```

Type C : Programming Practice/Knowledge based Questions

1. Write a function that takes amount-in-dollars and dollar-to-rupee conversion price; it then returns the amount converted to rupees. Create the function in both void and non-void forms.
2. Write a function to calculate volume of a box with appropriate default values for its parameters. Your function should have the following input parameters :

   (*a*) length of box ; (*b*) width of box ; (*c*) height of box.

   Test it by writing complete program to invoke it.
3. Write a program to have following functions :

   (*i*) a function that takes a number as argument and calculates cube for it. The function does not return a value. If there is no value passed to the function in function call, the function should calculate cube of 2.

   (*ii*) a function that takes two char arguments and returns True if both the arguments are equal otherwise False.

   Test both these functions by giving appropriate function call statements.
4. Write a function that receives two numbers and generates a random number from that range. Using this function, the main program should be able to print three numbers randomly.
5. Write a function that receives two string arguments and checks whether they are same-length strings (returns True in this case otherwise false).
6. Write a function namely *n*thRoot( ) that receives two parameters x and n and returns nth root of x *i.e.*, $x^{\frac{1}{n}}$.

   The default value of n is 2.
7. Write a function that takes a number n and then returns a randomly generated number having exactly n digits (not starting with zero) *e.g.*, if n is 2 then function can randomly return a number 10-99 but 07, 02 etc. are not valid two digit numbers.
8. Write a function that takes two numbers and returns the number that has minimum one's digit.

   [*For example,* if numbers passed are 491 and 278, then the function will return 491 because it has got minimum one's digit out of two given numbers (491's **1** is < 278's **8**)].
9. Write a program that generates a series using a function which takes first and last values of the series and then generates four terms that are equidistant *e.g.*, if two numbers passed are 1 and 7 then function returns 1 3 5 7.