Chapter 6

A.

1. C
2. A, B
3. C
4. D
5. B
6. D
7. B
8. A
9. D
10. D
11. B
12. C
13. B

B.

1. True
2. True
3. False
4. True
5. False
6. False
7. True
8. True
9. True
10. True

C.

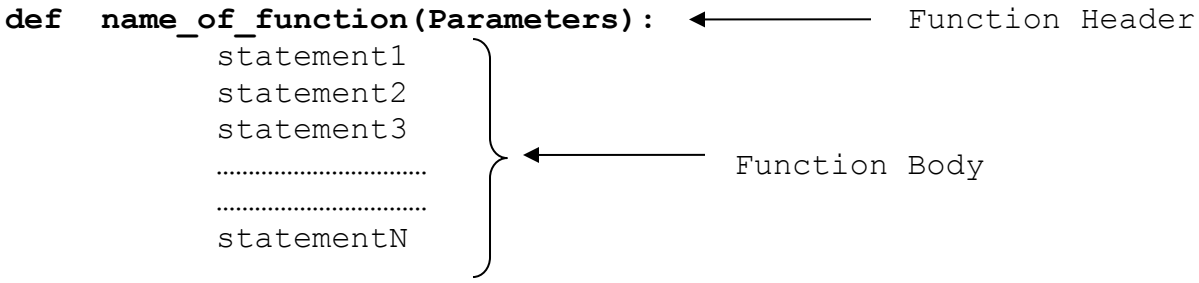
1. Advantages of using function are as follows
   a. Complex problems can be decomposed into smaller modules/pieces.
   b. Reduce duplication of code
   c. Reusability of code

2. If we want to perform the task repetitively, then it is not necessary to re-write the particular block of program again and again. This task can be done by function by

shifting the particular block of statements in user defined functions.

3. A function is a self-contained block or sub program of one or more statements that perform a special task when called.

4. The syntax for function as follows



A function in python consists of header and body. The header part of the function begins with keyword **'def'**. The name of the function is followed by **def** keyword. Function definition may contain zero or one or more than one parameters. If it contains multiple parameters, all the parameters are separated by commas. Whereas, a function body is a block of statement/s. The block of statement within the function defines the action that function has to be performed.

5. User defined function helps to decompose a large program into small parts which makes program easy to understand, maintain and debug. User defined functions can be modified. Whereas build in functions come with specific feature thus we cannot modify the contents of build in functions.

6. The following program demonstrates how arguments are passed and results are returned from the function

```
def calc(r):
    return 3.14*r*r
x = (int(input('Enter the radius:')))
print(' The  are  of circle is',calc(x)) #X passed as argument to calc()

Output
Enter the radius:5
78.5
```

In the above program, we have defined calc() as function which perform its task when called. Thus before calling we have read the radius of circle as input from the user and the same has been passed as argument to function calc(). Once the function is called it takes value of x as radius, calculate area and return back area of circle to the print statement.

7.

Values passed to a function when calling a function referred as arguments. Following program gives clear idea of arguments.

```
def area(p,q):
    print(p*q)

l = 10
b = 20
area(l,b)   #Arguments  l and b  passed to a function
```

8.

The return statement is used to return from a function. Even programmer can make use of it very effectively to return a value from it. The following program demonstrates the use of return statement within a function.

```
def sq(x):
    return x*x

print(sq(x=20))
```

9. Yes, programmer can return multiple values from a function. The following program illustrates simple example we can written multiple values.

```
def swap(a, b):
     a , b =  b, a
     return a, b
print(swap(10,20))
```

10.

Local Variables:

Variables declared within functions are referred to be as local variable and all those variables are local to the function. The following program illustrate the concept of local variable as follows

```
def Scope_Demo(x):
    x = x +10
    print(' The value of x(within local scope) is :',x)

x = 50
Scope_Demo(x)
print(' Still value of x is:',x)
```

Output

```
The value of x(within local scope) is : 60
Still value of x is: 50
```

From the above program, we can conclude the variable defined inside function is local to the function and the value of the local variable remains unaffected outside the function.

Global variables:

Variables that are assigned outside functions are said to exist in global scope. Therefore variable existing in global scope are said to be **global variable**. The following program gives clear idea between local and global scope of a variable.

```
x = 30  #Global Variable
def Demo():
    y = 10   # Local Variable
    print(' The value of local variable y is :',y)
    print(' The value of global variable x is :',x)
Demo()
```

Output

```
The value of local variable y is : 10
The value of global variable x is : 30
```

1. Write a function **eval_Quadratic_Equa(a, b, c, x)** which returns the value of any quadratic equation of form
   $ax^2 + bx + c$

```
import math
def eval_Quadratic_Equa(a, b ,c, x):
    d = (b**2) - 4*a*c
    if d < 0:
        print('The equation has no real solution')
    elif d == 0:
        x = (- b + math.sqrt(d))/ (2*a)
        print(' This equation has one solution:',x)
```

```
    else:
        x1 = (-b - math.sqrt(d))/(2*a)
        x2 = (-b  + math.sqrt(d))/(2*a)
        print('  Equation  has  two  solutions:  ',format(x1,'.2f'),'  and
',format(x2,'.2f'))

a =  float(input('Enter the value of a:'))
b = float(input('Enter the value of b:'))
c =  float(input('Enter the value of c:'))
eval_Quadratic_Equa(a,b,c,x = -1)
```

Output

```
Enter the value of a:1
Enter the value of b:0
Enter the value of c:-48
Equation has two solutions:  -6.93  and  6.93
```

2. Write function **calc_exp(base, exp)**, which computes the exponent of any number i.e. $\mathbf{base}^{\mathbf{exp}}$. The function should take two values as base which can be float or integer. Exp will be an integer greater than 0.

```
def calc_exp(base,exp):
    print(base**exp)
base = float(input('Enter the value of base:'))
exp = int(input('Enter the value of exponent:'))
calc_exp(base,exp)
```

Output

```
Enter the value of base:5
Enter the value of exponent:3
125.0
```

3. Write a function **Calc_GCD_Recurr(a, b)** which calculates the GCD recursively of two numbers. The function should take two positive integers and should return one integer as GCD.

Note: The greatest common divisor (GCD) of two positive integers is the largest integer that divides each of them without a remainder.

Example:

```
gcd(12 , 2) = 2
gcd(6 , 12) = 6
gcd(9 , 12) = 3
```

```
def Calc_GCD_Recurr(a, b):
    if b > a:
        return Calc_GCD_Recurr(b, a)
```

```
    if a%b == 0:
        return b
    return Calc_GCD_Recurr(b, a % b)
a = int (input('Enter the value of a:'))
b = int (input('Enter the value of b:'))
print(Calc_GCD_Recurr( a, b))
```

Output

```
Enter the value of a:9
Enter the value of b:12
3
```

4. Write a function **reverse_number()** to return the reverse of the number entered.

Example:
Reverse_number(1234) displays **4321**

```
def reverse_number(num):
    rev = 0
    while(num > 0):
        rev = (10*rev)+num%10
        num //= 10
    return rev

num = int(input('Enter the number:'))
print(reverse_number(num))
```

Output

```
Enter the number:234
432
```

5. A four-digit integer is entered through the keyboard. Write a function to calculate the sum of the four-digit number both without recursion and using recursion.

Without Recursion

```
def Calc_Sum_Digits(num):
    rem = 0
    sum1 = 0
    while(num > 0):
        rem = num%10
        sum1 = rem + sum1
        num //= 10
    return sum1

num = int(input('Enter the number:'))
print(Calc_Sum_Digits(num))
```

Output

```
Enter the number:5433
```

```
15
```

With Recursion

```
def Calc_Sum_Digits(num):
    rem = 0
    sum1 = 0
    if num == 0:
        return  sum1
    else:
        return sum1 + num%10 + Calc_Sum_Digits(num//10)

num = int(input('Enter the number:'))
print(Calc_Sum_Digits(num))
```

Output

```
Enter the number:123456789
45
```

6. A positive integer is entered through the keyboard. Write a function factors(num) to obtain the factors of the given numbers.

```
def factors(num):
    print('Factors of ',num,' are as follows: ')
    for x in range(1,num+1):
        if  num % x == 0:
            print(x, end = ' ')
        else:
            pass
num = int(input('Please Enter the number:'))
factors(num)
```

Output

```
Please Enter the number:120
Factors of  120  are as follows:
   1  2 3 4 5 6 8 10 12 15 20 24 30 40 60
```

7. Write a program to define function **dec_bin(num)** to convert the existing decimal number into its equivalent binary number.

```
def Dec_to_Bin(n):
   if n > 1:
       Dec_to_Bin(n//2)
   print(n % 2,end = '')

dec = int(input('Enter the Decimal Number:'))
Dec_to_Bin(dec)
```