Functions

Functions

Functions are blocks of reusable code.

Syntax:

def func_name(arguments):
    statements
    return some_value

Example:

```
"""Example : Add two numbers"""
def add(a,b):
    c=a+b
    return c

a=int(input("Enter first value:"))
b=int(input("Enter second value:"))
print("sum={}".format(add(a,b)))
```

Def:

a , b = arguments

add() = function name

c = local variable created to store & return result

Variable Scope

Variable scope: it determines the visibility of the variable

2 types:

- **Local Variable**: Visible only in the function where it's declared
- **Global Variable**: Declared once in program and visible to all the functions

Example:

```
x=300   #Global declaration of x

def test1():
    print("x=",x) #calling global x

def test2():
    x=100  #local variable x
    print("x=",x) #will print the local variable value

test1()
test2()
```

Function Composition

- f(g(x))
- **Syntax:**

```
def f(x):
        statement1
        return something
def g(x):
        statement2
        return something
call f(g(x))  #main function
```

Example:

```
def add(x):
    z=x+20
    return z
def sub(x):
    u=x-10
    return u

x=10
print("Function composition:",add(sub(x)))
```

Recursion

- Function calling itself until it reaches a base condition is recursion.
- **Syntax:**

```
def func1():
        base case statements
        return func1()

print(func())                #main function
```

Example:

```
"""Print 1-n numbers"""

def tennumbers(x):
    if(x==0):              #base case
        return
    if(x):                 #recursion
        tennumbers(x-1)
    print(x)

x=10
tennumbers(x)
```

Lambda

Syntax:

Variable=lambda list_of_arguments: expression

```
"""Lambda Example"""
x=int(input("Enter the value of x:"))
y=int(input("Enter the value of y:"))
z=lambda x,y: x+y
print("z has x+y:{}".format(z(x,y)))
y=lambda x:x*5
print("y*5:{}".format(y(x)))
```

Result:

```
Enter the value of x:5

Enter the value of y:4
z has x+y:9
y*5:25
```

Use Functions

1. Find cube of any number
2. Find diameter, circumference and area of circle .
3. Find maximum and minimum between two numbers using functions.
4. A number is even or odd using functions.
5. A number is prime, Armstrong or perfect number using functions.
6. All prime numbers between given interval using functions.
7. All strong numbers between given interval using functions.
8. **All Armstrong numbers between given interval using functions.**
9. **All perfect numbers between given interval using functions**.

Use Recursion

1. Find sum of digits of the number using Recursive Function.
2. Find power of any number using recursion.
3. Print all natural numbers between 1 to n using recursion.
4. Print all even or odd numbers in given range using recursion.
5. Find sum of digits of a given number using recursion.
6. Write a C program to find factorial of any number using recursion.
7. Write a C program to generate nth Fibonacci term using recursion.
8. Write a C program to find GCD (HCF) of two numbers using recursion.
9. Write a C program to find LCM of two numbers using recursion.
10. Write a C program to check whether a number is palindrome or not using recursion

1. Write a C program to display all array elements using recursion.
2. Write a C program to find sum of elements of array using recursion.
3. Write a C program to find maximum and minimum elements in array using recursion.
4. Write a C program to find sum of all natural numbers between 1 to n using recursion.
5. Write a C program to find sum of all even or odd numbers in given range using recursion.
6. Write a C program to find reverse of any number and string using recursion.
7. Prime or Composite