Scope of variable:-

----->> All variables in a program may not be access at all locations in that program.

----->> They are acceble depends on where you have declared the variables.

Variables are divided in to 2-types. They are ,

1 . Global variable

2 . Local Variable.

1.Global variables:

------>> The variables which are declared outside of all the functions are known as global variables.

------>> Global variables of one module can be used through out module.

Example:

```
a=100
def f1():
    print(a)
f1()
```

Output:

```
100
```

2. Local variables:

----->> the variables declared with in a function are known as local variables.

------>> local variables of one function can not be aceessed outside of that function.

Example:

```
def f1():
    a=100     # local
    print(a)
def f2() :
    print(a)
a=200    # global
```

```
f1()

f2()
```

Reccursive function call:-

---->> the concept of calling one function by the same function name is known as a "recursive" function calling.

Note : Infinite Recursion is not supported by the python:-

```
def f1():
    print("in f1")
    f1()
f1()
```

LAMBDA FUNCTIONS

1. Lambda function is a small anonymous functions, i.e. functions without a name.

2. These functions are throw-away functions, i.e. they are just needed where they have been created.

3. We use a lambda function when we require a nameless function for short time.

The general syntax of a lambda function is quite simple:

lambda argument_list : expression

The argument list consists of a comma separated list of arguments and the expression is an arithmetic expression using these arguments.

Example:

```
a =  lambda x , y :  x if x > y else y

print(a(20,5))
```

Output : 20

In the above script lambda x,y: x if x>y else y is lambda function whereas x,y are the arguments and x if x>y else y is expression.

The above function has no name and it returns a function object which is assigned to a identifier ‘a’.

Generally we use "def" keyword to create a function definition but in this case we don't use def keyword instead we use "lambda" keyword.

We can use either def or lambda methods to create a function.

------>> after executing the lambda function,it will returns the expression value

Q) Write a function to add two numbers ?

By using def

```
def    add_nums(x , y):
        return x + y
print(add_nums(1,2))                # 3
```

by using Lambda:

```
a = lambda x,y:x+y
print(a(1,2))                    # 3
```

Q) Write a function to find maximum value of given two numbers ?

By using def

```
def max_nums(x,y):
        if x > y:
            return x
        else :
            return y
print(max_nums(20,5))            # 20
```

By using lambda

```
a =  lambda x,y: x if x>y else y
print(a(20,5))                   # 20
```

Q) Write a function to calculate square for given number ?

By using def

```
def square_val(x):
        return x*x
print(square_val(10))              # 100
```

By using lambda

```
a  = lambda x : x * x
print(a(10))                  # 100
```

Example:

```
double  =  lambda  x  :  x*2
print double(5)
```

--->> It is equal to like below ,

```
def double(x):
    return x*2
double(10)
```

NOTE: generally we use the lambda function, when ever we want to pass one function as a parameter to the another function

Q. Write a Python program to print the only Even numbers from given list elements ?

```
lst =[1,5,4,6,8,11]
even_list =list(filter( lambda  x  :  ( x%2 == 0,  lst)))
print(even_list)
```

output : [4,6,8]

---->> we can also use regular function as a parameter to another function.

For example,

```
lst = [1,2,3,4,5,6,7,8,9,10]
l2 = []
```

```
def evenOrOdd(l):
        if l%2==0:
          return l
x = list(filter(evenOrOdd, lst))
print(x)
```

Note: We can also do same thing using normal functions like below,

```
lst = [1,2,3,4,5,6,7,8,9,10]
l2 = []
def evenOrOdd(l):
    for i in l:
        if i%2==0:
            l2.append(i)
    return l2
x = evenOrOdd(lst)
print(x)
```

Output:

[2,4,6,8]

Example:

```
def f1(a):
    return a
print(f1(10))->10
```

f1(10) ---> no output

Example:

```
def f1(a):
        return a
b=f1(10)
print(b)   ----->> # 10
```

Example:

```
def f1(a):
    print(a)
f1(10)          # 10
print(f1(10))     # 10
                    # None
b = f1(10)
print(b)      #  10
                # None
```

Practice :

Q 1) Write a python function to add two numbers ?

```
def add(x,y):
      return x+y
add(10,20)    #  30
```

Q 2) Write a python function to find maximum value of two given values ?

```
def max(x,y) :
    if x > y :
        return x
    else :
         return y
max(1,2)       # 2
max(10,2)     # 10
```

Q 3)Write a python function to find max value of three given values ?

```
def maxofthree(x,y,z):
        if x > y  and  x > z :
            return x
        elif  y  >  z :
            return y
        else  :
            return z
maxofthree(100,3,20)      # 100
maxofthree(10,3,20)       # 20
maxofthree(10,30,20)      # 30
```

Python filter() Function

Python filter() function is used to get filtered elements.

This function takes two arguments, first is a function and the second is iterable object.

The filter function returns a sequence from those elements of iterable for which function returns True.

The first argument can be None if the function is not available and returns only elements that are True.

For example:

```
lst = [1,2,3,4,5,6,7,8,9,10]
x = list(filter( None, lst))
print(x)
```

Output: [1,2,3,4,5,6,7,8,9,10]

Signature :filter (function, iterable)

Parameters Explanation:

function: It is a function. It returns only condition matched elements.

Iterable: Any iterable sequence like list, tuple, and string.

Both the parameters are required.

return Statement:

It returns the same as returned by the function.

Let's see some examples of **filter()** function to understand it's functionality.

Python filter() Function Examples:

Q. Write a program to returns values higher than 5 using filter function?

```
# Python filter() function example
def filterdata(x):
    if x > 5:
        return x
# Calling function
result = filter(filterdata,(1,2,6))
# Displaying result
print(list(result))
```

Output: [6]

Q. Write a program to find numbers which are divisible by 3 from given seaquence?

```
# Python filter() function example
def multiplicationof3(val):
    if (val % 3) == 0:
        return val
# Calling function
result = filter(multiplicationof3,(1,3,5,6,8,9,12,14))
# Displaying result
result = list(result)
```

```
print(result) # multiples of 3
```

Output: [3, 6, 9, 12]

Python map() Function

The python map() function is used to return a list of results after applying a given function to each item of an iterable(list, tuple etc.).

Example:

```
def calculateAddition(n):
    return n+n
numbers = (1, 2, 3, 4)
result = map(calculateAddition, numbers)
print(result)
# converting map object to set
numbersAddition = set(result)
print(numbersAddition)
```

Output:

```
<map object at 0x7fb04a6bec18>
{8, 2, 4, 6}
```

Q. Add 5 value to every list item and returns result ?

```
lst = [1,4,7,9,6,11,16,15]
result = list(map( lambda  x : x + 5, lst))
print(result)
```

Output: [6, 9, 12, 14, 11, 16, 21, 20]

reduce() in Python:

The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.

This function is defined in “functools” module.

Explanation:

At first step, first two elements of sequence are picked and the result is obtained.

Next step is to apply the same function to the previously attained result and the number just succeeding the second element and the result is again stored.

This process continues till no more elements are left in the container.

The final returned result is returned and printed on console.

For Example:

```
# python code to demonstrate working of reduce()
# importing functools for reduce()
import functools
# initializing list
lis = [1, 3, 5, 6, 2, ]
# using reduce to compute sum of list
print("The sum of the list elements is : ", end="")
print(functools.reduce(lambda a, b: a+b, lis))

# using reduce to compute maximum element from list
print("The maximum element of the list is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b, lis))
```

Output:

```
The sum of the list elements is : 17
The maximum element of the list is : 6
```

Output:

```
The sum of the list elements is : 17
The maximum element of the list is : 6
```

Examples:

Q. How to find sum of given list values ?

---->> using sum() function

```
lst = [1,4,7,9,6,11,16,15]
#print(sum(lst))
```

---->> using reduce() with lambda expression

```
import functools
lst = [1,4,7,9,6,11,16,15]
total_sum = functools.reduce(lambda x,y : x + y , lst)
#print(total_sum)
```

---->> using userdefined function with reduce() function

```
lst = [1,4,7,9,6,11,16,15]
def total_sum(x,y):
    return x + y
total_sum = functools.reduce(total_sum, lst)
print(total_sum)
```

---->> using userdefined function logics

```
lst = [1,4,7,9,6,11,16,15]
def  total_sum(obj):
    global a
    a = 0
    for i in obj:
        a = a + i
total_sum(lst)
print("total sum of lst is : ",a)
```

Q. How to find maximum value of given list values ?

---->> using predefined max() function

```
print(max(lst))
```

---->> using userdefined function with reduce() function

```
def find_gigNumber(x,y):
```

```
    if x > y:
     return x
  else:
     return y
biggest_num = functools.reduce(find_gigNumber, lst)
print(biggest_num)
```

operator functions:

reduce() can also be combined with operator functions to achieve the similar functionality as with lambda functions and

makes the code more readable.

For example:

```
# python code to demonstrate working of reduce() using operator functions
# importing functools for reduce()
import functools
# importing operator for operator functions
import operator
# initializing list
lis = [1, 3, 5, 6, 2]
# using reduce to compute sum of list using operator functions
print("The sum of the list elements is : ", end="")
print(functools.reduce(operator.add, lis))
# using reduce to compute product  using operator functions
print("The product of list elements is : ", end="")
print(functools.reduce(operator.mul, lis))
```

Output:

```
The sum of the list elements is : 17
```

The product of list elements is : 180

 # reduce() with lambda expression

'''

```
import functools
lst = [1,2,3,4,5]
result = functools.reduce(lambda x,y : x + y, lst)
print(result)
```

'''

```
# reduce() with operator module  functions
import functools
import operator
lst = [1,2,3,4,5]
lst2 = [30,2,5]
lst3 = [2,3,2]
result1 = functools.reduce(operator.add, lst)
result2 = functools.reduce(operator.mul, lst)
result3 = functools.reduce(operator.sub, lst)
result4 = functools.reduce(operator.itruediv, lst2)
result5 = functools.reduce(operator.pow, lst3)

print("Multiplication result is :",result1)
print("Multiplication result is :",result2)
print("Subtraction result is :",result3)
print("Division result is :",result4)
print("Power values result is :",result5)
```

Output:

Multiplication result is : 15

Multiplication result is : 120

Subtraction result is : -13

Division result is : 3.0

Power values result is : 64

Examples:

Q. How to find a Factorial value of given number using reccursive function?

```
def factorial(n):
    if n == 0:
        return 0
    if n == 1:
        return 1
    else:
        return  n * factorial(n-1)
n = int(input('Enter any number :'))
print('Factorial value of',n ,'is :',factorial(n))
```

Output:

Enter any number :5

Factorial value of 7 is : 120

Q. How to find a factorial value of given number using normal function ?

```
def factorial(n):
    fact = 1
    i = 1
    while i <= n: # 1 <= 5,  2 <= 5, 3 <=5, 4 <= 5, 5 <= 5, 6 <= 5
        fact = fact * i # 1, 2, 6, 24, 120
        i = i + 1 # 2, 3, 4, 5, 6
    return fact
n = int(input('Enter any number :')) # 5
```

```
print('The factorial value of',n,'is :',factorial(n))
```

Output:

```
Enter any number :5
The factorial value of 5 is : 120
```

Q. Write a function to find given value is a palindrom or not ?

```
def is_palondrom(n):
    if n == n[::-1]:
        print(n,'is a palindrom value')
    else:
        print(n,'is  not a palindrom value')
n = input('Enter any number :') # 5
is_palondrom(n)
```

Output:

```
Enter any number :121
121 is a palondrom value
Enter any number :madam
madam is a palindrom value
```

Q. How to create local scope variable as a global scope variable and use them inside anothore function?

```
a = 10 # global scope
def addition():
    global b
    b = 20  # global scope
    print('Addition of',a,'and',b,'is :', a + b)
addition()


def multiplication():
```

```
    print('Multiplication of',a,'and',b,'is :', a * b)
multiplication()
```

Output:

```
Addition of 10 and 20 is : 30
Multiplication of 10 and 20 is : 200
```

Q. How to find the fibonacci sequence numbers and its sum value?

```
Number = int(input("Please Enter the Fibonacci Number Range = "))
First = 0
Second = 1
Sum = 0
for Num in range(0, Number):
    print(First, end = '  ')
    Sum = Sum + First
    Next = First + Second
    First = Second
    Second = Next
print("\nThe Sum of Fibonacci Series Numbers = %d" %Sum)
```

Output:

```
Please Enter the Fibonacci Number Range = 8
0  1  1  2  3  5  8  13
The Sum of Fibonacci Series Numbers = 33
```

Q. Python program to find the sum of all the Fibonacci Series numbers using recursion or recursive functions.

```
def fibonacci(Number):
    if(Number == 0):
        return 0
```

```
    elif Number == 1:
        return 1
    else:
        return fibonacci(Number - 2) + fibonacci(Number - 1)
Number = int(input("Please Enter the Fibonacci Number Range = "))
Sum = 0
for Num in range(Number):
    print(fibonacci(Num), end = '  ')
    Sum = Sum + fibonacci(Num)
print("\nThe Sum of Fibonacci Series Numbers = %d" %Sum)
```

Output:

```
Please Enter the Fibonacci Number Range = 30
0  1  1  2  3  5  8  13  21  34  55  89  144  233  377  610  987  1597  2584  4181  6765  10946
17711  28657  46368  75025  121393  196418  317811  514229
The Sum of Fibonacci Series Numbers = 1346268
```