In [4]:
```
#UN-Named statemetns
def named_state(a,b):
    print("Hello from a function")
    print("Statement two")
    print("Statement Three")
    return a+b
```

In [2]:
```
def add(a,b):
  return a+b

add(1,2)
```

Out[2]:
```
3
```

In [5]:
```
lambda_ref = lambda a,b : a+b
```

In [6]:
```
lambda_ref(1,2)
```

Out[6]:
```
3
```

In [7]:
```
named_state(4,5)
```

```
Hello from a function
Statement two
Statement Three
```

Out[7]:
```
9
```

In [13]:
```
def add_values(a,b=12):
    print('A Value is : ',a)
    print('B Value is :',b)
    c = a+b
    print('C value is : ',c)
    return c
```

In [14]:
```
add_values(10,40)
```

```
A Value is :  10
B Value is : 40
C value is :  50
```

Out[14]:
```
50
```

Creating Function

In [16]:
```
#Creating Function
def my_func():
    print("Hello from a function")
    print("Statement two")
    print("Statement Three")
#Calling or Executing function
my_func()
```

```
Hello from a function
Statement two
Statement Three
```

In [17]:
```
#Creating a function with two arguments  a and b
def my_sum(a,b):
    print('Sum of a and b value is : ',a+b)
    return a+b
```

Loading [MathJax]/extensions/Safe.js

```
#calling or executing a function
my_sum(90,20)
```

```
Sum of a and b value is :  110
```

Out[17]:
```
110
```

In [18]:
```
my_func()
```

```
Hello from a function
Statement two
Statement Three
```

In [20]:
```
def my_sum(a,b):
    print('A value is : ',a)
    print('B value is : ',b)
```

In [21]:
```
my_sum(5,4)
```

```
A value is :  5
B value is :  4
```

Calling Or Executing Funciton

In [22]:
```
my_func()
```

```
Hello from a function
Statement two
Statement Three
```

Arguments (parameters)

- Information can be passed into functions as arguments.

In [23]:
```
def my_func(fname,age):
    print("My Name is : ",fname)
    print("MY Age is : ",age)
```

Positional Arguments are processed in order

In [24]:
```
#Positional Arguments are processed in order
my_func(age=23,fname="Govardhan")
```

```
My Name is :  Govardhan
MY Age is :  23
```

In [25]:
```
def addition(a,b):
     print(f" Sum Of {a} + {b}  is  {a+b}")
```

In [26]:
```
addition(10,20)
```

```
 Sum Of 10 + 20  is  30
```

In [27]:
```
def check_even_list(num_list):
    # Go through each number
    # Declare empty list
    even=[]
    for number in num_list:
        # Once we get a "hit" on an even number, we return True
        if number % 2 == 0:
          # append if number is even number using append method
          even.append(number)
```

Loading [MathJax]/extensions/Safe.js

```
        # Don't do anything if its not even
        else:
            pass
    # Notice the indentation! This ensures we run through the entire for loop
    return even

check_even_list([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
```

Out[27]:
```
[2, 4, 6, 8, 10, 12, 14, 16]
```

In [28]:
```
def check_odd_list(num_list):
    # Go through each number
    # Declare empty list
    odd=[]
    for number in num_list:
        # Once we get a "hit" on an even number, we return True
        if number % 2 == 1:
          # append number if its odd number
          odd.append(number)

        # Don't do anything if its not even
        else:
            pass
    # Notice the indentation! This ensures we run through the entire for loop
    return odd

check_odd_list([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16])
```

Out[28]:
```
[1, 3, 5, 7, 9, 11, 13, 15]
```

Default Parameter Value

- The following example shows how to use a default parameter value.
- If we call the function without argument, it uses the default value:

In [29]:
```
def my_function(name,age,loc = "India"):
    print("My Name is  : ",name)
    print("My Age is : ",age)
    print("I am from : ",loc)
# Calling a function without 3rd argument. it will considar default value.
my_function("Govardhan",23)
```

```
My Name is  :  Govardhan
My Age is :  23
I am from :  India
```

In [30]:
```
my_function("Govardhan",23,'Hyderabad')
```

```
My Name is  :  Govardhan
My Age is :  23
I am from :  Hyderabad
```

In [31]:
```
def squareroot(x):
    return x * x
```

In [32]:
```
squareroot(100)
```

Out[32]:
```
10000
```

In [33]:
```
print(squareroot(3))
oot(5))
```

Loading [MathJax]/extensions/Safe.js

```
9
25
```

Variable Number of Arguments (`*args`)

- In cases where you don’t know the exact number of arguments that you want to pass to a function,
- you can use the following syntax with *args:

In [34]:
```
def my_sum(*args):
   return sum(args)
```

In [35]:
```
my_sum(1,2,3,4,5,6,9)
```

Out[35]:
```
30
```