Functions in Python

Python function is a block of statement that return the specific task.
The idea is to put some commonly or repeatedly done task together and make a function ,so instead of writing same code again and again for different inputs, we can do the function call to reuse code contained in it over and over again.

Syntax- python function

```
Def function_name (parameters):          #def is a keyword
  #statement
Return expression
```

```
Example: def fun();
            print("welcome to python")
            fun()

   Output: welcome to python.
```

Defining and calling a function with parameters

```
Syntax-    def function-name(parameter:data_type)
           Return_type:
           #body of the function
           Return expression
```

```
 Example: #some more funtions
           Def is-prime(n):
           If n is [2,3]:
           Return true
           if(n==1) or (n%2==0):
           Return false
           r=3
           While r*r<=n:
           If n% r==0:
           Return False
           r+=2
           Return true
           print (is_prime(78),is_prime(79))
 Output: False True
```

Arguments of a python Function:

Arguments are the values passed inside the parenthesis of the function. A function can be have any number of arguments separated by a comma.

Example: #simple python function to check whether x is even or odd

```
Def evenodd(x):
print(“even”)
Else:
print (“odd)
#driver code to call the function
evenodd(2)
evenodd(3)
```

Output: even
Odd

Types of arguments:

1. **Default argumen**t: it is a parameter that assume a default value is not provided in the function call for the argument.
   Example: def myfun(x,y=50):
   ```
   print(“x:”,x)
   print(“y:”,y)
   #drivercide (we call myfun() with only argument)
   myfun(10)
   ```
   Output: x=10
   y=50

2. **Keyword arguments**: The idea is to allow the caller to specify the argument name with value so that caller does not need to remember the order of parameters.
   Example: def student(firstname, lastname):
   ```
   print(firstname,lastname)
   #keyword arguments
   student(firstname=’pavan’,lastname=’kumar’)
   student(lastname=’kumar’,firstname=’pavan’)
   ```
   Output: pavan kumar
   Kumar pavan

3. **Variable length arguments**: We can pass a variable number of arguments to a function using special symbol. There are two special symbols.
   3.1 *args(non-keyword arguments)
   **kwargs(keyword arguments)

Example:

```
Def myfun(*argsv):
For arg in argv:
print(arg)
myfun('hello','welcome','to','pyhton')
```

Output:

```
Hello
Welcome
to
Python
```

<u>Docstring</u>: The first string after the function is called the document string or docstring in short. This is used to describe the functionality of the function. The use of docstring in function is optional but it is considered a good practice.

Syntax: print(function_name_doc_)

Example:

```
def evenodd(x):
    “”Function to check whether x is even or odd””
    If (x%2==2):
    print(“even”)
    else:
    print(“odd”)
    #Driver code to call the function
    print(evenodd_doc_)
```

Output: function to check if the number is even or odd

<u>Return statement in python function-</u>

The function return statement is used to exit from a function and go to the function caller and return the specified value or data item to the caller.

Syntax: return[expression_list]

The return statement can consist of a variable,an expression, or a constant which is returned to the end of the function execution. If none of the above is present with the return statement a none object is returned.

Example:

```
def Square_value(num):
“”This function returns the square value of the entered number””
Return num**2
print(square_value(2))
print (Square_value(4))
```

Output:

```
4
16
```

Pass by reference Or Pass by value:

One important thing to note is,In python every variable name is a reference. When we pass a variable to a function, a new reference to the object is created.

Example:
```
def myfun(x):
            x[0]=20
      #driver code(note the list is modified after function call
      list=[10,11,12,13,14,15]
      myfun(list)
      print(list)
```
Output: [20,10,11,12,13,14,15]

Solved Examples:

1.Define a function that returns the maximum of two numbers

```
def max_of_two(x, y):
                         # Check if x is greater than y
 if x > y:
                          # If x is greater, return x
return x
                          If y is greater or equal to x, return y
return y
```

2.Define a function that returns the maximum of three numbers

```
def max_of_three(x, y, z):
                             # Call max_of_two function to find the maximum of y and z,
                             # then compare it with x to find the overall maximum return
max_of_two(x, max_of_two(y, z))
                              # Print the result of calling max_of_three function with arguments 3 6 -5
print(max_of_three(3, 6, -5))
```

Output: 6

3.write a program to multiply all number in a list

```
                             # Define a function named 'multiply' that takes a list of numbers as input
def multiply(numbers):
                            # Initialize a variable 'total' to store the multiplication result, starting at 1
total = 1

                              # Iterate through each element 'x' in the 'numbers' list
```

```
 for x in numbers:
                                                    # Multiply the current element 'x' with the 'total'
 total *= x

                                                            # Return the final multiplication result stored in the 'total' variable
 return total

                          # Print the result of calling the 'multiply' function with a tuple of numbers (8, 2, 3, -1, 7)
print(multiply((8, 2, 3, -1, 7)))

 Output: 329
```