**Introduction to User-Defined Functions**

**Definition:**

A user-defined function in Python is a block of reusable code designed to perform a specific task. It allows you to encapsulate functionality into a single named entity, making code more modular, readable, and maintainable.

**Basics:**

**Declaration:**

Functions are declared using the def keyword, followed by the function name and parameters enclosed in parentheses.

**Syntax:-**

* def function\_name(parameter1, parameter2, ...):

# Function body

**Calling a Function:**

Functions are called by using their name followed by parentheses and passing arguments, if any.

* function\_name(argument1, argument2, ...)

**Return Statement:**

Functions can optionally return a value using the return statement. If no return statement is provided, the function returns None by default.

**Example-1:-**

def greet(name):

return "Hello, " + name

print(greet("Alice")) # Output: Hello, Alice

**Example-2:-**

def evenOdd( x ):

if (x % 2 == 0): print("even")

else: print("odd")

# Driver code

evenOdd(2)

evenOdd(3)

**Python Default arguments**

A default arguments is a parameter that assumes a default value if a value is not provided in the function call for that argument. The following example illustrates Default arguments.

# Python program to demonstrate

# default arguments

def myFun(x, y = 50):

print("x: ", x)

print("y: ", y)

# Driver code

myFun(10)

**Output:**

x: 10  
y: 50

## ****Python****Function with return value

Sometimes we might need the result of the function to be used in further processes. Hence, a function should also return a value when it finishes its execution. This can be achieved by a return statement.   
A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller. The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned.

**Syntax:**

def fun():  
 statements  
 .  
 return [expression]

**Example :-**

def add(a, b):

# returning sum of a and b

return a + b

def is\_true(a):

# returning boolean of a

return bool(a)

# calling function

res = add(2, 3)

print("Result of add function is {}".format(res))

res = is\_true(2 < 5)

print("\nResult of is\_true function is {}".format(res))

**Decorators:**

Decorators are functions that modify the behavior of other functions. They are applied using the @decorator\_name syntax.

**Example :-**

def my\_decorator(func):

def wrapper():

print("Before function call")

func()

print("After function call")

return wrapper

@my\_decorator

def say\_hello():

print("Hello!")

say\_hello() # Output: Before function call, Hello!, After function call

**Variable-Length Arguments:**

Functions can accept a variable number of arguments using \*args and \*\*kwargs syntax.

**Example :-**

def add(\*args):

total = 0

for num in args:

total += num

return total

print(add(1, 2, 3, 4)) # Output: 10