11. Python - Functions - Part - 1

Table of Contents

1. Function 2
2. Types of function 2
2.1. Pre-defined or built-in functions 2
2.2. User Defined Functions: 3
3. Function related terminology 3
4. Function definition 4
4.1. Creating a function 4
4.2. Calling a function 6
6. A Function can call other function 10
5. Based on Parameters: Functions are two types 12
5.1. Function without parameters 12
5.2. Function with parameters 13
7. return keyword in python 17
7.1. Function without return 18
7.2. Function with return 19
7.3. return vs None 25
7.4. A function can return multiple values 26

11. Python - Functions - Part - 1

1. Function

- ✓ A function can contain group of statements which performs the task.

Advantages

- ✓ Maintaining the code is an easy way.
- ✓ Code reusability.

Make a note

- ✓ print() is predefined function in python which prints output on the console.

2. Types of function

- ✓ There are two types of functions,
  - Pre-defined or built-in functions
  - User-defined functions

2.1. Pre-defined or built-in functions

- ✓ The functions which are already existing in python are called as predefined function

Examples

- print(p)
- type(p)
- input(p)

2.2. User Defined Functions:

- ✓ Based on requirement a programmer can create his own function, these functions are called as user defined functions.
- ✓ So, practically we will see how to define and use, user defined functions.

3. Function related terminology

- ✓ If we want to understand function concept in better way then we need to focus on function related terminology,
    - def keyword
    - name of the function
    - parenthesis **()**
    - parameters (if required)
    - colon symbol**:**
    - function body
    - return type (optional)

4. Function definition

- ✓ A function can contains group of statements.
- ✓ The purpose of function is to perform an operation.
- ✓ A function can contain mainly two parts,
    1. Creating a function
    2. Calling a function

4.1. Creating a function

- ✓ Very first step is we need to create a function.
- ✓ We need to use **def** keyword to create a function.
- ✓ After **def** keyword we should write name of the function.
    - After function name, we should write parenthesis **()**
        - This parenthesis may contain parameters.
            - Parameters are just like variables which receive the values
            - If function having parameters, then we need to provide the values while calling.
            - We will learn more in parameterized function
    - After parenthesis we should write colon **: symbol**
    - After : symbol in next line we should provide indentation
- ✓ Function body.
    - Actual logic contains by function body
    - This function body helps to perform the operation.
- ✓ Before closing the function, function may contain return type.

Syntax

```
def functionname():
        """ doc string"""
        Body of the function to perform operation
```

A naming convention to define a function

- ✓ As discussed in Naming convention chapter, function name should be in lower case.
- ✓ If name having multiple words, then separating words with underscore (_) symbol is a good practice.

Program Name: Creating a function
demo1.py

```
# function creation
def display():
        print("Welcome to function")
```

output

Make a note

- ✓ When we execute above program, then function body not executed.
- ✓ To execute function body, we need to call the function.

4.2. Calling a function

- ✓ After function is created then we need to call that function to execute the function body.
- ✓ While calling the function, function name should be match otherwise we will get error.

Program Name: Create and call user defined function
demo2.py

```
# function creation
def display():
        print("Welcome to function concept")

# function calling
display()
```

output

```
Welcome to function concept
```

Program Name: Create and call user defined function
demo3.py

```
# function creation
def display():
        print("Welcome to function concept")

# function calling
display()
display()
```

output

```
Welcome to function concept
Welcome to function concept
```

Program Name: Create and call user defined function
demo4.py

```
# function creation
def display():
        print("Welcome to function concept")

# function calling
details()
```

output

```
NameError: name 'details' is not defined
```

Question

- Can i create more than one function in a single python program?

Answer

- Yes, we can
- Based on requirement we can create any number of functions.

Program Name: Creating two functions and calling those functions

demo5.py

```
def first():
    print("This is first function")

def second():
    print("This is second function")

first()
second()
```

output

```
This is first function
This is second function
```

Program Name: Creating two functions and calling those functions
demo6.py

```
def first():
        print("This is first function")

def second():
        print("This is second function")

second()
first()
```

output

```
This is second function
This is first function
```

6. A Function can call other function

- ✓ Based on requirement a function can call another function as well.
- ✓ We can call a function inside another function.

Syntax

```
def first_function():
    body of the first function
    we can call the secondfunction


def second_function():
    body of the second function

first function calling
```

Program Name: Creating two functions
demo7.py

```
def m1():
    print("first function")

def m2():
    print("second function")

m1()
m2()
```

output

```
first function
second function
```

Program Name: One function can call another function

demo8.py

```
def m1():
        print("first function")
        m2()

def m2():
        print("second function")

m1()
```

output

```
first function
second function
```

5. Based on Parameters: Functions are two types

- ✓ Based on parameters, functions can be divided into two types,
  - Function without parameters **(or)** No parameterised function
  - Function with parameters **(or)** Parameterised function

5.1. Function without parameters

- ✓ If a function having no parameters then that function is called as, No parameterized function

Syntax

```
def nameofthefunction():
        body of the function to perform operations

function calling
```

Program Name: Function which having no parameters
demo9.py

```
# defining a function
def display():
        print("Welcome to function which having no parameters")

# calling function
display()
```

output

```
Welcome to function which having no parameters
```

5.2. Function with parameters

- ✓ If a function having parameters then that function called as parameterised function

Why function having parameters?

- ✓ Function parameters help to process the function operation.
- ✓ When we pass parameters then,
  - Function capture parameters values
  - These values perform the operations.
  - Finally it brings the result.

Syntax

```
def functionname(parameter1, parameter2, …):
        body of the function
function calling
```

Program Name: One parameterized function

demo10.py

```
def testing(a):
        print("one parameterised function:", a)

testing(10)
```

output

```
one parameterised function: 10
```

Program Name: One parameterized function
demo11.py

```
def testing(a):
        print("one parameterised function:", a)

testing(10.56)
```

output

```
one parameterised function: 10.56
```

Program Name: One parameterized function
demo12.py

```
def testing(a):
        print("one parameterised function:", a)

testing("Daniel")
```

output

```
one parameterised function: Daniel
```

Program Name: One parameterized function
demo13.py

```
def testing(a):
        print("one parameterised function:", a)


x = input("Enter a value:")
testing(x)
```

output

```
Enter a value: 10
one parameterised function: 10
```

Program Name: Two parameterized function
demo14.py

```
def testing(a, b):
        print("two parameterised function:", a, b)

testing(10, 20)
```

output

```
two parameterised function: 10 20
```

Program Name: Function performing addition operation
demo15.py

```
def addition(a, b):
        print("Addition of two values=", (a+b))

addition(10, 20)
```

output

```
Addition of two values =30
```

7. return keyword in python

- ✓ Based on return statement, functions can be divided into two types,
  - Function without return statement
  - Function with return statement

- ✓ return is a keyword in python.
- ✓ We should use return statement with function or method, otherwise we will get error.

Program Name	return outside of function which is invalid demo16.py
	print('Hello') return 100
output	
	SyntaxError: 'return' outside function

7.1. Function without return

- ✓ If a function cannot contains return statement then that function is called as a function without return statement.
- ✓ It's not mandatory to write return statement to a function.
- ✓ A function without return statement is valid.

Program Name: Function displaying information
demo17.py

```
def balance():
        print("My bank balance is: ")

balance()
```

output

```
My bank balance is:
```

7.2. Function with return

- ✓ Based on requirement we can write return statement to a function.
- ✓ A function with return statement is valid.

Syntax

```
def nameofthefunction():
    body of the function
    return result
```

Program Name: Function with return statement displaying information
demo18.py

```
def balance():
    print("My bank balance is: ")
    return 100

balance()
```

output

```
My bank balance is:
```

Note

- ✓ If a function contains return statement then that function calling we need to assign to a variable.
- ✓ Daniel why we need to assign to a variable?
- ✓ Yes, i will explain please wait in another five minutes, then you can understand.

Program Name: Function with return statement
demo19.py

```
def balance():
        print("My bank balance is: ")
        return 100

b = balance()
print(b)
```

output

```
My bank balance is:
100
```

Why we need to assign a function calling to a variable?

- ✓ If we assign function calling to a variable then that variable holding the variable value.
- ✓ That variable we can use further in our program.

Program Name: Function with return statement
demo20.py

```
def balance():
    return 100

b = balance()

if b==0:
    print("Balance is: ", b)

elif b<=0:
    print("Balance is: ", b, " negative please deposit")

else:
    print("Balance is: ", b)
```

output

```
Balance is: 100
```

Program Name: Function with return statement
demo21.py

```
def balance():
        return 0

b = balance()

if b == 0:
        print("Balance is: ", b)

elif b<=0:
        print("Balance is: ", b, " negative please deposit")

else:
        print("Balance is: ", b)
```

output

```
Balance is: 0
```

Program Name: Function with return statement
demo22.py

```
def balance():
        return -50

b = balance()

if b == 0:
        print("Balance is: ", b)

elif b <= 0:
        print("Balance is: ", b, "it is negative please deposit")

else:
        print("Balance is: ", b)
```

output

```
Balance is: -50 it is negative please deposit
```

Note

- ✓ Below program also valid but not recommended.

Program Name: Function with return statement
demo23.py

```
def balance():
    print("My bank balance is: ")
    return 100

print(balance())
```

output

```
My bank balance is:
100
```

Note

- ✓ A method can return a value as well.
- ✓ We will learn this concept again in OOPS chapter.

7.3. return vs None

- ✓ If any function is not return anything, then by default that function returns None data type.
- ✓ We can also say as, if we are not writing return statement, then default return value is None

Program Name: function returning the None value
demo24.py

```
def m1():
        print("This function is returning nothing")

x = m1()
print(x)
```

output

```
This function is returning nothing
None
```

7.4. A function can return multiple values

- ✓ In python, a function can return multiple values.
- ✓ Based on requirement a function can return multiple values.
  - If function is returning two values then while function calling we need to assign to two variables
  - If function is returning three values then while function calling we need to assign to three variables.
  - If function is returning more than one values, while calling function if we assign with one variable then all values will be stored in tuple.

Syntax

```
def name_of_the_function():
    body of the function
    return value1, value2, value3,...,valueN
```

Program Name: Define a function which can return multiple values
demo25.py

```
def m1():
        a = 10
        b = 11
        return a, b

x, y = m1()

print("first value is:", x)
print("second value is:", y)
```

output

```
first value is: 10
second value is: 11
```

Program Name: Define a function which can return multiple values
demo26.py

```
def m1():
        a = 10
        b = 11
        c = 12

        return a, b, c

x, y, z = m1()

print("first value is:", x)
print("second value is:", y)
print("third value is:", z)
```

output

```
first value is: 10
second value is: 11
third value is: 12
```

Program Name: Define a function which can return multiple values
demo27.py

```
def m1():
        a = 10
        b = 11
        c = 12
        return a, b, c

x = m1()
print("all values:", x)
```

output

```
all values: (10, 20, 30)
```

Program Name: A function with parameters and return type.
demo28.py

```
def add(x, y, z):
        result = x+y+z
        return result

r = add(10, 20, 30)

print("Addition of values:", r)
```

output

```
Addition of values: 60
```