

## Functions

What is a function? Before we start making functions, let us learn what a function is and why we need them?

### Defining a Function

A function is a reusable block of code or programming statements designed to perform a certain task. To define or declare a function, Python provides the `def` keyword. The function block of code is executed only if the function is called or invoked.

### Declaring and Calling a Function

When we make a function, we call it declaring a function. When we start using the it, we call it calling or invoking a function. Function can be declared with or without parameters.

```
# syntax
# (Declaring a function)
# def function_name():
#     codes
#     codes
#(Calling a function)
#function_name()
```

## Function without Parameters

Function can be declared without parameters.

Example:

```
#creating function

def add_two_numbers():
    n1 = 2
    n2 = 3
    sum = n1 + n2
    print(sum)
```

```
# calling a function
add_two_numbers()
```

## Function Returning a Value - Part 1

Function can also return values, if a function does not have a return statement, the value of the function is `None`. Let us rewrite the above functions using `return`. From now on, we get a value from a function when we call the function and print it.

```
def add_two_numbers():
    num_one = 2
    num_two = 3
    total = num_one + num_two
    return total
```

```
print(add_two_numbers())
```

```
5
None
```



```
def generate_full_name():
    first_name = 'Abc'
    last_name = 'Def'
    space = ' '
    full_name = first_name + space + last_name
    return full_name
```

```
a=generate_full_name()
print(a)
```

Abc Def

## Function with Parameters

In a function we can pass different data types(number, string, boolean, list, tuple, dictionary or set) as a parameter

- **Single Parameter:** If our function takes a parameter we should call our function with an argument

```
# syntax:
# (Declaring a function)
# def function_name(parameter):
#     codes
#     codes
# (Calling function)
# print(function_name(argument))
```

Example:

```
#function definition
def greetings (name):
    message = name + ', welcome to Python for Everyone!'
    return message
```

```
print(greetings('Riya')) #function calling
```

```
#function defintion
def add_ten(num):
    ten = 10
    return num + ten
```

```
a= add_ten() # storing the returned value in a variable
print(a) # display the value
```

```
-----  
TypeError                                 Traceback (most recent call last)  
/tmp/ipython-input-1579638416.py in <cell line: 0>()  
----> 1 a= add_ten() # storing the returned value in a variable  
      2 print(a) # display the value
```

```
TypeError: add_ten() missing 1 required positional argument: 'num'
```

Next steps: [Explain error](#)

```
# function generating square of a number
def square_number(x):
    return x * x
```

```
n=int(input("enter the number ")) #taking number as input
sq=square_number(n) # storing the returned value in a variable
print(sq) #display
```

```
enter the number 66
4356
```

```
print(square_number(7)) #function calling and display the value
```

```
#area of circle
def area_of_circle (r):
    PI = 3.14
    area = PI * r ** 2
    return area
```

```
print(area_of_circle(10))
```

- **Two Parameter:** A function may or may not have a parameter or parameters. A function may also have two or more parameters. If our function takes parameters we should call it with arguments. Let us check a function with two parameters:

```
# syntax:
#(Declaring a function)
# def function_name(para1, para2):
#     codes
#     codes
# (Calling function)
# print(function_name(arg1, arg2))
```

Example:

```
#function Defintion-2 parameters
def weight_of_object (mass, gravity):
    weight = str(mass * gravity)+ ' N' # the value has to be changed to a string first
    return weight
```

```
print('Weight of an object in Newtons: ', weight_of_object(100, 9.81))
```

- If we do not return a value with a function, then our function is returning None by default.

## Function Returning a Value - Part 2

To return a value with a function we use the keyword `return` followed by the variable we are returning. We can return any kind of data types from a function.

- **Returning a string:** Example:

```
def print_name(firstname):
    return firstname
print_name('Asa') # Asa

def print_full_name(firstname, lastname):
    space = ' '
    full_name = firstname + space + lastname
```

```
return full name
```

- **Returning a number:** Example:

```
def add_two_numbers (num1, num2):  
    total = num1 + num2  
    return total  
print(add_two_numbers(2, 3))  
  
def calculate_age (current_year, birth_year):  
    age = current_year - birth_year  
    return age;  
print('Age: ', calculate_age(2019, 1819))
```

- **Returning a boolean:** Example:

```
def is_even (n):  
    if n % 2 == 0:  
        print('even')  
        return True    # return stops further execution of the function, similar to break  
    return False
```

```
print(is_even(10)) # True
```

```
even  
True
```

```
print(is_even(7)) # False
```

## ▼ Function as a Parameter of Another Function

```
#You can pass functions around as parameters  
def square_number (n):  
    return n * n  
def do_something(f, x):  
    return f(x)  
print(do_something(square_number, 3)) # 27
```

```
Start coding or generate with AI.
```