Python Function Arguments

In computer programming, an argumentÂ is a value that is accepted by a function.

Before we learn about function arguments, make sure to know about Python Functions.

Example 1: Python Function Arguments

```
def add_numbers(a, b):
    sum = a + b
    print('Sum:', sum)

add_numbers(2, 3)

# Output: Sum: 5
```

In the above example, the function `add_numbers()` takes two parameters: `a` and `b`. Notice the line,

```
add_numbers(2, 3)
```

Here, `add_numbers(2, 3)` specifies that parameters `a` and `b` will get values **2** and **3** respectively.

Function Argument with Default Values

In Python, we can provide default values to function arguments.

We use the `=` operator to provide default values. For example,

```
def add_numbers( a = 7,  b = 8):
    sum = a + b
    print('Sum:', sum)


# function call with two arguments
add_numbers(2, 3)

#  function call with one argument
add_numbers(a = 2)

# function call with no arguments
add_numbers()
```

Output

```
Sum: 5
Sum: 10
Sum: 15
```

In the above example, notice the function definition

```
def add_numbers(a = 7, b = 8):
    ...
```

Here, we have provided default values **7** and **8** for parameters a and b respectively. Here's how this program works

1. add_number(2, 3)

Both values are passed during the function call. Hence, these values are used instead of the default values.

2. add_number(2)

Only one value is passed during the function call. So, according to the positional argument **2** is assigned to argument `a`, and the default value is used for parameter `b`.

3. add_number()

No value is passed during the function call. Hence, default value is used for both parameters `a` and `b`.

Python Keyword Argument

In keyword arguments, arguments are assigned based on the name of the arguments. For example,

```
def display_info(first_name, last_name):
    print('First Name:', first_name)
    print('Last Name:', last_name)

display_info(last_name = 'Cartman', first_name = 'Eric')
```

Output

```
First Name: Eric
Last Name: Cartman
```

Here, notice the function call,

```
display_info(last_name = 'Cartman', first_name = 'Eric')
```

Here, we have assigned names to arguments during the function call.

Hence, `first_name` in the function call is assigned to `first_name` in the function definition. Similarly, `last_name` in the function call is assigned to `last_name` in the function definition.

In such scenarios, the position of arguments doesn't matter.

Python Function With Arbitrary Arguments

Sometimes, we do not know in advance the number of arguments that will be passed into a function. To handle this kind of situation, we can use arbitrary arguments in Python.

Arbitrary arguments allow us to pass a varying number of values during a function call.

We use an asterisk (*) before the parameter name to denote this kind of argument. For example,

```
# program to find sum of multiple numbers

def find_sum(*numbers):
    result = 0

    for num in numbers:
        result = result + num

    print("Sum = ", result)

# function call with 3 arguments
find_sum(1, 2, 3)

# function call with 2 arguments
find_sum(4, 9)
```

Output

```
Sum =  6
Sum =  13
```

In the above example, we have created the function `find_sum()` that accepts arbitrary arguments. Notice the lines,

```
find_sum(1, 2, 3)
find_sum(4, 9)
```

Here, we are able to call the same function with different arguments.

Note: After getting multiple values, `numbers` behave as an array so we are able to use the for loop to access each value.

Table of Contents