Python Default Parameters

Summary: in this tutorial, you'll learn about the Python default parameters to simplify function calls.

Introduction to Python default parameters

When you define a function (https://www.pythontutorial.net/python-basics/python-functions/) , you can specify a default value for each parameter.

To specify default values for parameters, you use the following syntax:

```
def function_name(param1, param2=value2, param3=value3, ...):
```

In this syntax, you specify default values (`value2` , `value3` , ...) for each parameter using the assignment operator (`=)` .

When you call a function and pass an argument to the parameter that has a default value, the function will use that argument instead of the default value.

However, if you don't pass the argument, the function will use the default value.

To use default parameters, you need to place parameters with the default values after other parameters. Otherwise, you'll get a syntax error.

For example, you cannot do something like this:

```
def function_name(param1=value1, param2, param3):
```

This causes a syntax error.

Python default parameters example

The following example defines the `greet()` function that returns a greeting message:

```
def greet(name, message='Hi'):
    return f"{message} {name}"
```

The `greet()` function has two parameters: `name` and `message`. And the `message` parameter has a default value of `'Hi'`.

The following calls the `greet()` function and passes the two arguments:

```
def greet(name, message='Hi'):
    return f"{message} {name}"


greeting = greet('John', 'Hello')
print(greeting)
```

Output:

```
Hello John
```

Since we pass the second argument to the `greet()` function, the function uses the argument instead of the default value.

The following example calls the `greet()` function without passing the second argument:

```
def greet(name, message='Hi'):
    return f"{message} {name}"


greeting = greet('John')
print(greeting)
```

Output:

```
Hi John
```

In this case, the `greet()` function uses the default value of the `message` parameter.

Multiple default parameters

The following redefines the `greet()` function with the two parameters that have default values:

```
def greet(name='there', message='Hi'):
    return f"{message} {name}"
```

In this example, you can call the `greet()` function without passing any parameters:

```
def greet(name='there', message='Hi'):
    return f"{message} {name}"


greeting = greet()
print(greeting)
```

Output:

```
Hi there
```

Suppose that you want the `greet()` function to return a greeting like `Hello there` . You may come up with the following function call:

```
def greet(name='there', message='Hi'):
    return f"{message} {name}"


greeting = greet('Hello')
print(greeting)
```

Unfortuntely, it returns an unexpected value:

```
Hi Hello
```

Because when you pass the `'Hello'` argument, the `greet()` function treats it as the first argument, not the second one.

To resolve this, you need to call the `greet()` function using keyword arguments (https://www.pythontutorial.net/python-basics/python-keyword-arguments/) like this:

```
def greet(name='there', message='Hi'):
    return f"{message} {name}"


greeting = greet(message='Hello')
print(greeting)
```

Output:

```
Hello there
```

Summary

- Use Python default parameters to simplify the function calls.
- Place default parameters after the non-default parameters.