Python Keyword Arguments

Summary: in this tutorial, you'll learn about the Python keyword arguments, and how to use them to make function calls more obvious.

Introduction to the Python keyword arguments

Let's start with a simple function (https://www.pythontutorial.net/python-basics/python-functions/) that calculates the net price from the selling price and discount:

```
def get_net_price(price, discount):
    return price * (1-discount)
```

The `get_net_price()` function has two parameters: `price` and `discount` .

The following shows how to call the `get_net_price()` function to calculate the net price from the price `100` and discount `10%` :

```
net_price = get_net_price(100, 0.1)
print(net_price)
```

Output:

```
90.0
```

In the `get_net_price(100, 0.1)` function call, we pass each argument as a positional argument. In other words, we pass the `price` argument first and the `discount` argument second.

However, the function call `get_net_price(100, 0.1)` has a readability issue. Because by looking at that function call only, you don't know which argument is `price` and which one is the `discount` .

On top of that, when you call the `get_net_price()` function, you need to know the position of each argument.

If you don't, the function will calculate the `net_price` incorrectly. For example:

```
net_price = get_net_price(0.1, 100)
print(net_price)
```

Output:

```
-9.9
```

To improve the readability, Python introduces the keyword arguments.

The following shows the keyword argument syntax:

```
fn(parameter1=value1,parameter2=value2)
```

By using the keyword argument syntax, you don't need to specify the arguments in the same order as defined in the function.

Therefore, you can call a function by swapping the argument positions like this:

```
fn(parameter2=value2,parameter1=value1)
```

The following shows how to use the keyword argument syntax to call the `get_net_price()` function:

```
net_price = get_net_price(price=100, discount=0.1)
```

Or:

```
net_price = get_net_price(discount=0.1, price=100)
```

Both of them returns the same result.

When you use the keyword arguments, their names that matter, not their positions.

Note that you can call a function by mixing positional and keyword arguments. For example:

```
net_price = get_net_price(100, discount=0.1)
```

Keyword arguments and default parameters

Suppose that you have the following `get_net_price()` function that calculates the net price from the selling price, tax, and discount.

```
def get_net_price(price, tax=0.07, discount=0.05):
    return price * (1 + tax - discount)
```

In the `get_net_price()` function, the tax and discount parameters have default values (https://www.pythontutorial.net/python-basics/python-default-parameters/) of `7%` and `5%` respectively.

The following calls the `get_net_price()` function and uses the default values for `tax` and `discount` parameters:

```
net_price = get_net_price(100)
print(net_price)
```

Ouput:

```
102.0
```

Suppose that you want to use the default value for the `tax` parameter but not `discount`. The following function call doesn't work correctly.

```
net_price = get_net_price(100, 0.06)
```

… because Python will assign `100` to price and `0.1` to tax, not discount.

To fix this, you must use keyword arguments:

```
net_price = get_net_price(price=100, discount=0.06)
print(net_price)
```

Output:

```
101.0
```

Or you can mix the positional and keyword arguments:

```
net_price = get_net_price(100, discount=0.06)
print(net_price)
```

Output:

```
101.0
```

Python keyword argument requirements

Once you use a keyword argument, you need to use keyword arguments for the remaining parameters.

The following will result in an error because it uses the positional argument after a keyword argument:

```
net_price = get_net_price(100, tax=0.08, 0.06)
```

Error:

```
SyntaxError: positional argument follows keyword argument
```

To fix this, you need to use the keyword argument for the third argument like this:

```
net_price = get_net_price(100, tax=0.08, discount=0.06)
print(net_price)
```

Summary

- Use the Python keyword arguments to make your function call more readable and obvious, especially for functions that accept many arguments.
- All the arguments after the first keyword argument must also be keyword arguments too.