Python function parameter

In the previous article we learned about the built-in Python function and the user-defined Python function with customizable number of parameters. You will know how to define a function using the default parameters, keyword and custom parameters in this article.

In the previous article we learned about the built-in Python function and the user-defined Python function with customizable number of parameters. You will know how to define a function using the default parameters, keyword and custom parameters in this article.

Depending on how you define the Python function in any way we can call different functions, of course, no matter how you call the function, the final result is still reasonable and the same.

Parameters of Python functions

In the user-defined Python function, we already know how to define and call a function, like the example below:

```
def Xin_chao(ten,loi_chao):
 """Hàm Xin_chao chào m?t ng??i
v?i thông ?i?p cho tr??c """
 print("Xin chào",ten + ', ' + loi_chao)

Xin_chao("Nam","??c bài trên TipsMake.com vui v? nha!")
```

When running the above code, we will get the text on the screen as follows:

```
Xin chào Nam, ??c bài trên TipsMake.com vui v? nha!
```

As you can see, this `Xin_chao()` function has 2 parameters. Therefore, if we call the function with two parameters, it will run "well" and will not encounter any error messages.

If the function is called with a different parameter number 2, the interpreter will display an error message. If you want, you can try calling the `Xin_chao()` function with one parameter and no parameters.

```
# Error message when calling Xin_chao () function with 1 parameter
TypeError: Xin_chao () missing 1 argument positional required: 'loi_chao'

# Error message when calling Xin_chao () function without parameters
TypeError: Xin_chao () missing 2 required positional arguments: 'ten' and 'loi_
```

Besides defining functions with a fixed number of parameters, you can use the methods we introduce below to create custom parameters for the function:

Default parameter in Python function

In Python, the parameter of the function can have a default value. We can provide this default value for a parameter by using the assignment operator (=) as the example below:

```
def Xin_chao(ten, loi_chao = "??c bài TipsMake.com vui nha!"):
 """
 Hàm này chào m?t ng??i v?i thông ?i?p cho tr??c.

  N?u thông ?i?p không ???c cung c?p,
 nó s? ???c m?c ??nh là "??c bài TipsMake.com vui nha!"
 """
print("Xin chào",ten + ', ' + loi_chao)

Xin_chao("H?i")
Xin_chao("D?ng","b?n kh?e không?")
```

When running the above code, we have the following output:

```
Xin chào H?i, ??c bài TipsMake.com vui nha!
Xin chào D?ng, b?n kh?e không?
```

In this function, the `ten` parameter has no default value and is required when calling the function. The parameter `loi_chao` has a default value of `"??c bài TipsMake.com vui nha!"` . Therefore, when calling the function you can provide this parameter or not. If the parameter `loi_chao` , it will override the default value.

In a function, the number of parameters with default values ??is not limited, but once there are default parameters, all parameters on the right of the default parameter must also have default values. That is, non-default parameters cannot be behind default parameters. For example, if you define the Xin_chao () function as follows:

```
def Xin_chao(loi_chao = "??c bài TipsMake.com vui nha!",ten):
```

Then will receive an error message:

```
SyntaxError: non-default argument follows default argument
```

The keyword parameter in Python

When calling a function with several values, these values ??will be assigned to the parameters according to their location. For example, in the Xin_chao () function above, when calling `Xin_chao("D?ng","b?n kh?e không?")` , The `"D?ng"` value is assigned to the `ten` parameter and the value `"b?n kh?e không?"` assigned to `loi_chao` .

Python allows calling functions by keyword parameters. When calling a function this way, the order (position) of the parameter may change, but the result is still valid and similar to other function calls.

```
# 2 tham s? keyword
Xin_chao(ten = "D?ng",loi_chao = "b?n kh?e không?")
```

```
# 2 tham s? keyword, th? t? tham s? thay ??i
Xin_chao(loi_chao = "b?n kh?e không?",ten = "D?ng")
# 1 tham s? keyword, 1 theo v? trí
Xin_chao("D?ng",loi_chao = "b?n kh?e không?")
```

As you can see above, it is possible to combine parameters by keyword or location when calling a function. But remember that the parameters keyword must go after the parameter by location. If the parameter according to the position after the keyword parameter you will receive an error message:

```
Xin_chao(ten = "D?ng","b?n kh?e không?")
```

Error message:

```
SyntaxError: non-keyword argument after keyword argument
```

Custom parameters in Python functions

Sometimes, we do not know in advance the number of parameters that will be passed to the function, it is time to use the function call with the number of custom parameters.

In the function definition, we use an asterisk * before the parameter name to denote the parameter type, as the example below:

```
def Xin_chao(*ten_chao):
  """
  Hàm này s? chào m?t danh sách ng??i cho tr??c
  """
  for ten in ten_chao:
    print("Xin chào",ten)

Xin_chao("H?i","Hoa","Công","S?n")
```

When running the above code we get the following result:

```
Xin chào H?i
Xin chào Hoa
Xin chào Công
Xin chào S?n
```

Here, we call the function with many parameters. This parameter is "wrapped" into a tuple before being passed into the function. Inside the function, we use the `for` loop to get all the parameters in tuple.

In the next lesson, we will learn about Python recursion.

Next lesson: Recursive function in Python

Last lesson: Python function is user defined

Don't forget to do your Python homework.

guides. Thank you for reading and for following us regularly.