Functions in Python

What is Python function? How is the syntax, components, and function types in Python? How to create functions in Python? These questions will be answered in the Python lesson below.

What is Python function? How is the syntax, components, and function types in Python? How to create functions in Python? These questions will be answered in the Python lesson below.

What is Python function?

In Python, a function is a group of interrelated commands that are used to perform a specific task or task. The function helps divide the Python program into smaller blocks / sections / modules. When Python programs are too big, or need to expand, functions that help the program are more organized and manageable.

The function also has a very important effect: to avoid having to repeat the code to execute similar tasks, making the code more compact and reusable.

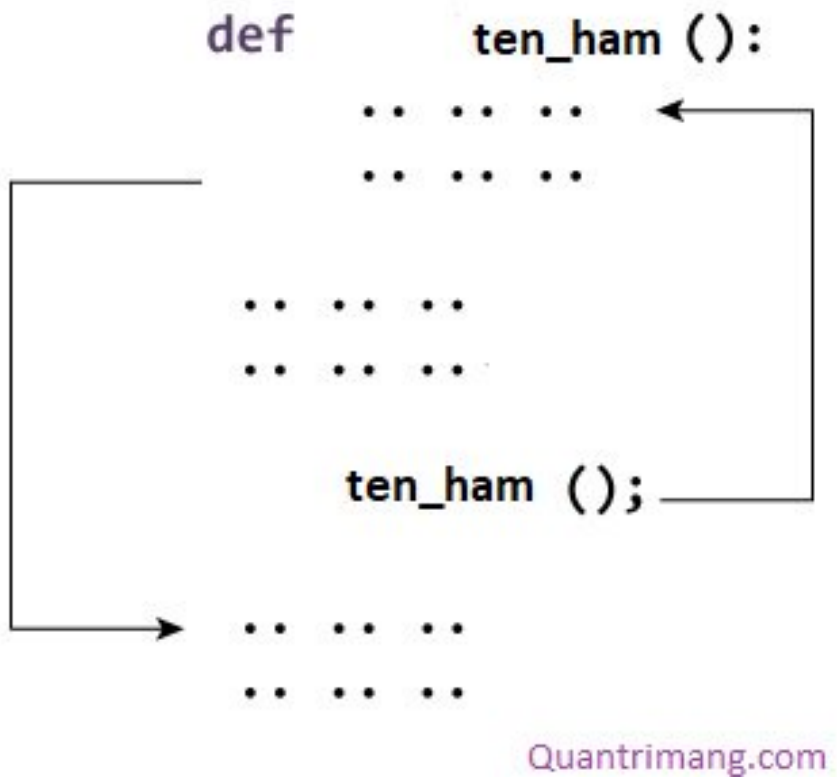
The syntax of Python functions

```
def ten_ham(các tham s?/??i s?):
 """Chu?i v?n b?n ?? mô t? cho hàm (docstring)"""
 Các câu l?nh
```

Basically, a Python function definition will include the following components:

1. Keyword def : Type in the beginning of the function title.
2. ten_ham : The unique identifier for the function. The naming of functions must comply with Python's name and identifier rules.
3. Parameters / arguments : We pass the value to the function through these parameters. They are optional.
4. Colons (:) : Mark the end of the function title.
5. docstring : Optional text string to describe the function of the function.
6. Statements : One or more valid Python commands form a block of commands. These commands must have the same indentation level (usually 4 spaces).
7. Return : This command is optional, used when the value from the function needs to be returned.

How the function works in Python:



Example of Python function

Here is a simple function definition, including the function name, function parameter, function description and a statement:

```
def chao(ten):
 """Hàm này dùng ??
chào m?t ng??i ???c truy?n
vào nh? m?t tham s?"""
 print("Xin chào, " + ten + "!")
```

Call the function in Python

When a function has been defined, you can call it from another function, another program, or even at the command prompt. To call the function we just need to enter the function name with the appropriate parameters.

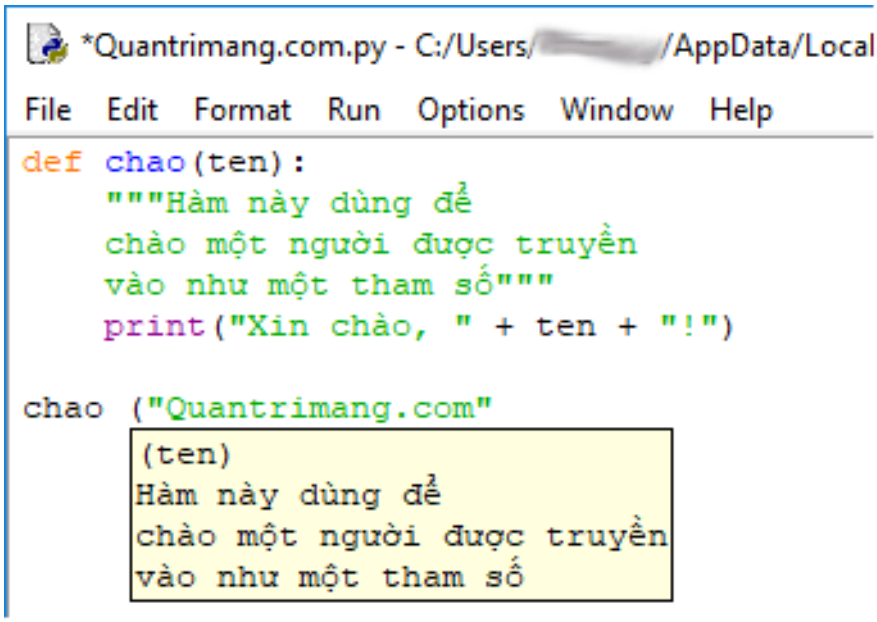
For example, to call the chao () function just defined above, we type the following command at the prompt:

```
>>> chao ("TipsMake.com")
```

We will get the following result:

```
>>> chao("Quantrimang.com")
Xin chào, Quantrimang.com!
>>>
```

You can also call this chao () function right in the code you are writing as shown below and the results will be the same as above:



Docstring in Python

The first string immediately after the function title is called docstring (documentation string), which is used to explain the function for the function. Although docstring is not required, the short explanation of the function of the function will help the user later, even you, to call the function to understand immediately what the function will do without having to find the definition again. function to consider.

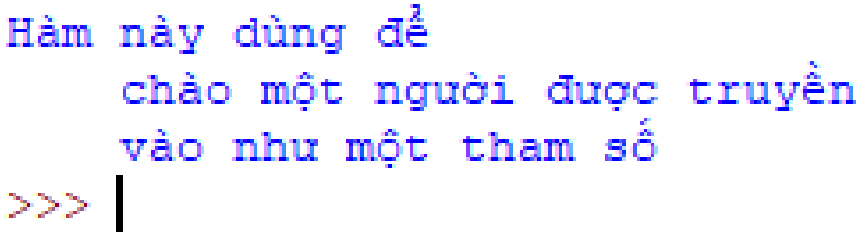
Adding documents to code is a good habit. There is no guarantee that after a few months back you can remember the details, clear the code you wrote earlier without error.

In the example above we have a docstring right below the function title. Docstring is usually written in pairs of 3 quotes. This string will appear as a __doc__ attribute of the function.

To check the chao docstring () above, enter the following code and test it:

```
print (chao.__doc__)
```

Here are the results:



Return command in Python function

The return command is usually used to exit the function and return to the place where the function is called.

Syntax of return :

```
return [danh_sach_bieu_thuc]
```

This command may contain calculated expressions and return values. If there is no expression in the statement or no return command in the function, the function returns None.

Example of return:

```
def gia_tri_tuyet_doi(so):
 """Hàm này tr? v? giá tr? tuy?t ??i
 c?a m?t s? nh?p vào"""
 if so >= 0:
 return so
 else:
 return -so
# ??u ra: 5
print(gia_tri_tuyet_doi(5))
# ??u ra: 8
print(gia_tri_tuyet_doi(-8))
# ??u ra: Giá tr? tuy?t ??i c?a s? nh?p vào
num=int(input("Nh?p s? c?n l?y giá tr? tuy?t ??i: "))
print (gia_tri_tuyet_doi(num))
```

When running the above code, we get the following result:

```
5
8
Nh?p s? c?n l?y giá tr? tuy?t ??i: -7
7
```

Range and duration of variables

The scope of the variable is the program segment where the variable is recognized. Parameters and variables are defined inside a function that cannot be "seen" from outside. Therefore, these variables and parameters only have scope in the function.

The lifetime of the variable is the amount of time that variable appears in memory. When the function is executed, the variable will exist.

The variable is destroyed when we exit the function. The function does not remember the value of the variable in the previous function calls.

```
def ham_in():
x = 15
print("Giá tr? bên trong hàm:",x)
x = 30
ham_in()
print("Giá tr? bên ngoài hàm:",x)
```

In the above program, we use the same variable x, a variable inside the ham_in () function, an external variable x and executes the command to print these two values ??so that you realize the scope of the variable. The value of x we ??initialize is 30, although ham_in () has changed the value of x to 15, but it does not affect the value of x outside the function. When running the program, we get the result:

```
Giá tr? bên trong hàm: 15
Giá tr? bên ngoài hàm: 30
```

This is because the variable x inside the function is different from the variable x outside the function. Although they have the same name, they are actually two different variables with different ranges. The variable x in the function is a local variable, only effective in that function. The variable x outside the function is visible from within the function and has scope across the entire program.

With the variable outside the function, we can read the value of the variable in the function, but it cannot change its value. To change values ??for variables of this type, they must be declared as global variables using the global keyword.

Types of functions in Python

Basically, Python has two main types of functions which are built-in functions in Python and user-defined functions. In the next articles, we will learn more about these two types of functions.

Python exercises have a prize

Previous article: Python loop techniques