What is a function in Python? Functions in Python

Functions in Python have many different types. You need to clearly understand Python functions to use them properly.

Functions in Python have many different types. You need to clearly understand **Python functions** to use them properly.

What is a function in Python?

A function is an organized and reusable block of code used to implement a related action. Functions in Python provide better modularity to your application, while also allowing for a higher level of code reuse.

As you know, Python has many built-in functions like print(). but you can also create your own. These functions are called user-defined functions.

For example, if you need to write an application that draws hundreds of triangles to create a kaleidoscopic effect, you can do it two ways:

1. **Don't use functions** : You repeat the code to draw each triangle one by one
2. **Using a function** : You create a series of coordinates and feed them into the triangle drawing function

The second method is more efficient, requires less code to be written, and is often the preferred method of programmers. Not only that, with this method, if you want to change from triangle to square, you only need to change a few lines of code.

Another benefit of using functions is modularity and compactness. If you write another application that has triangles in it, you can copy and paste the triangle function you just wrote above.

Define a function

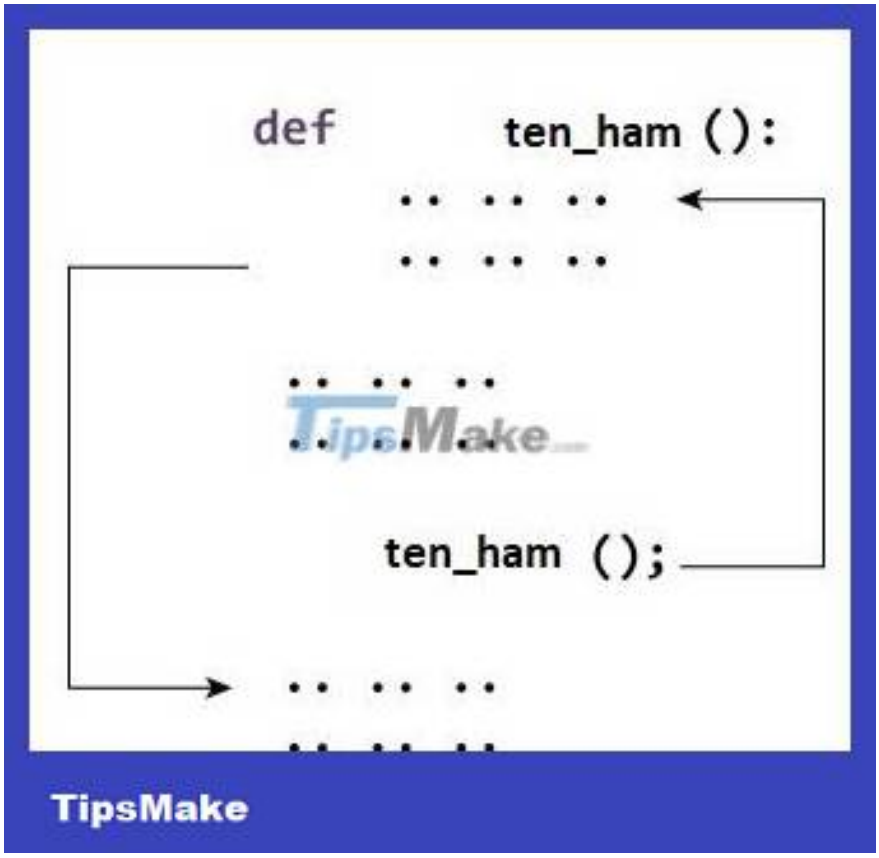
You can define functions to provide the required functionality. Below are simple rules for defining a function in Python.

1. Function blocks begin with the def keyword followed by the function name and parentheses (( )).
2. Any input parameters or arguments are placed inside parentheses. You can also specify parameters inside parentheses.
3. The first command of a function can be an optional command - this function's docstring or docstring.
4. The code block inside each function begins with a symbol (:) and is indented.
5. This command returns [expression] exiting a function, optionally passing an expression back to the caller. A statement that returns no arguments is the same as a statement that returns None.

Syntax of Python functions

```
def ten_ham(các tham s?/??i s?): Kh?i l?nh bên trong hàm L?nh ngoài hàm ho?c l?nh g?i hàm
```

How functions work in Python:



Function example in Python

Below is a simple function definition, including the function name, function parameters, function description, and a statement:

```
def chao(ten): """Hàm này dùng ?? chào m?t ng??i, tên ???c truy?n vào nh? m?t tham s?""" print("Chào b?n " + ten + ". Chúc m?t ngày vui v?!")
```

Call functions in Python

Once a function has been defined (like the example above), 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 function `chao()`just defined above, we type the following command right at the prompt:

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

The results obtained are:

```
Chào b?n TipsMake.com.com. Chúc m?t ngày vui v?!
```

Note : In Python, the function definition must always come before the function call. Otherwise, they will report an error as follows:

```
NameError: name 'chao' is not defined
```

Some extremely simple examples of Functions in Python.

Example 1: Print out the greeting on the screen:

```
def QtmHello(): print("QuanTriMang xin chào các b?n!") return; QtmHello()
```

Here's how to define a function in Python and call it. When running the program, the results obtained are:

```
QuanTriMang xin chào các b?n!
```

Here, we define the **QtmHello** function and call it. First, we need to define the function using the statement `def`, then we can add any line of code inside the function. Commands `return`are often used to exit a function and return to the place where the function was called.

You should capitalize each letter in your function name. This is a common and useful way of writing code, it helps distinguish functions in Python from statements.

From now on, whenever you want to say hello to everyone, you just need to write `QtmHello ()`.

For example:

```
def QtmHello(): print("QuanTriMang xin chào các b?
n!") return; QtmHello() print("B?n ?ang h?c v?
 Python trên QuanTriMang.com") QtmHello()
```

Run this program and you will see the words " *TipsMake hello everyone!* " displayed twice.

```
QuanTriMang xin chào các b?n! B?n ?ang h?c v?
 Python trên QuanTriMang.com QuanTriMang xin chào các b?n!
```

Because the function definition and function call code are separate, the program will not run until you use the function call function in Python. You can also call a function from another function, for example:

```
def QtmHello(): print("QuanTriMang xin chào các b?
n!") HomNayTotNgay() return; def HomNayTotNgay(): print("Hôm nay là m?
t ngày t?t, ?úng không nh?!") return; QtmHello()
```

When running the program, the results obtained are:

```
QuanTriMang xin chào các b?n! Hôm nay là m?t ngày t?t, ?úng không nh?!
```

Now you know how to define and call functions in Python. However, now we are going to explore the true power of functions in Python.

How to pass data to functions in Python

While functions are useful for performing repetitive tasks, their real power lies in their ability to give and receive data. Python allows us to call a function while passing data to it.

Example 1:

```
def XinChao(Name): print("Xin chào " + Name) return; XinChao("QuanTriMang")
```

When running the program, the results obtained are:

```
Xin chào QuanTriMang
```

This means that the same function can perform slightly different functions, depending on the variables we include.

Example 2: Enter user-entered data into the function

```
def XinChao(Name): print("Xin chào " + Name) return; Name = input("Nh?
p tên c?a b?n: ") XinChao(Name)
```

The result when running the command will be greeted by the name the user enters:

```
Nh?p tên c?a b?n: QTM Xin chào QTM
```

How to manipulate data in functions

Functions in Python can even convert data. To do this, we need to pass information into the function, perform actions, and then return the information.

Here is an example:

```
def PhepNhan(Number): return Number * 10; print(PhepNhan(5))
```

Here, the returned result is **50** because we added the value 5 by calling the function in Python and the result is 5 multiplied by 10. Notice that we can call the Python function as the name of a integer. This allows us to code faster and more flexibly.

There are countless ways we can use this feature. Here is another example, just 3 lines of code:

```
def DemTen(Name): return len(Name); DienTen = "QuanTriMang.com" print(DemTen(Dien
```

This small application is a name length counter. By using `len()` Python's built-in function that returns an integer based on the length of the string. Note, the function `len()` will include spaces.

Docstring in Python

The first string immediately after the function title is called **docstring** (documentation string), it is used to explain the function of the function. Although the docstring is not required, giving a brief explanation of the function's function will help future users, even you, when calling the function, be able to immediately understand what the function will do without having to look up the definition again. function to consider.

Adding notes to your code is a good habit. There is no guarantee that after a few months of returning, you will remember the details and clearly of the previously written code without any errors.

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 an attribute `__doc__`of the function.

To check the docstring of the function `chao()`above, enter the following code and run it:

```
def chao(ten): """Hàm này dùng ?? chào m?t ng??i, tên ???c truy?n vào nh? m
?t tham s?""" print("Chào b?n " + ten + ". Chúc m?t ngày vui v?
!") print(chao.__doc__)
```

Here are the results:

```
Hàm này dùng ?? chào m?t ng??i, tên ???c truy?n vào nh? m?t tham s?
```

Return statement in Python function

Commands `return`are often used to exit a function and return to the place where the function was called.

Syntax of return command :

```
return [danh_sach_bieu_thuc]
```

This command can contain a calculated expression and a return value. If there is no expression in the statement or no return statement in the function, the function will return None.

Return command example:

```
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
```

Scope and lifetime of variables

The scope of a variable is the section of the program in which the variable is recognized. Parameters and variables defined inside a function are not "visible" from the outside. Therefore, these variables and parameters only have scope within the function.

The lifetime of a 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 from previous function calls.

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

In the above program, we use the same variable x, one variable inside the function `ham_in()`, one variable x outside and print these two values ??so you can see the scope of the variable. The value of x we ??initialize is 30, although the function `ham_in()`has changed the value of x to 15, it has no effect on the value of x outside the function. When running the program we get the following results:

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

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

Function types in Python

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

Some Python functions you need to know

`reduce()`

Python's reduce() function iterates over each item in a list or any other iterable data type, and returns a single value. It is one of the class methods `functools`available in Python. For example:

```
from functools import reduce def add_num(a, b): ???
return a+b a = [1, 2, 3, 10] print(reduce(add_num, a)) Output: 16
```

You can also format a list of strings using the function `reduce()`:

```
from functools import reduce def add_str(a,b): return a+' '+b a = ['TipsMake', '
?t', 'web', 'th??ng m?i ?i?n t?'] print(reduce(add_str, a)) Output:
 TipsMake.com là m?t m?t web th??ng m?i ?i?n t?
```

`split()`

The split() function breaks a string based on set criteria. You can use it to strip a string value from a web form. For example:

```
words = "column1 column2 column3" words = words.split(" ") print(words)
Output: ['column1', 'column2', 'column3']
```

`enumerate()`

The enumerate() function returns the length of an iterator and can iterate over its items simultaneously. For example:

```
fruits = ["grape", "apple", "mango"] for i, j in enumerate(fruits): print(i, j)
Output: 0 grape 1 apple 2 mango
```

You finished reading the article "**What is a function in Python? Functions in Python**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com