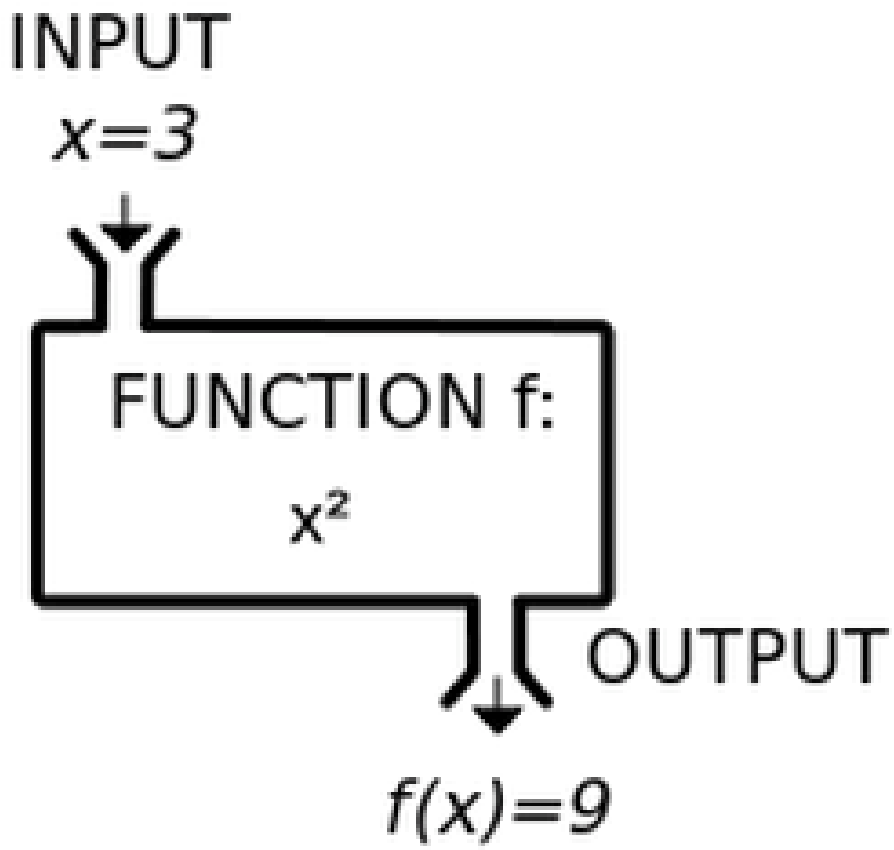
Functions

Functions in Math



Functions in Programming

Programmers love shortcuts that make writing code easier. One of the most common shortcuts is to give a name to a block of code that does an especially useful job.

Then, instead of having to type out the whole block each time you need it, you simply type its name.

These named blocks of code are called **functions.**

Defining a function

Define When you use the **def** keyword and write the code for a function, coders say you “define” the function. You also define a variable when you first set its value.

```
1  def name_of_the_function(argument1, argument2):
2      print("hello" + argument1)
3      print(7+ argument2)
4
5
6
```

How to use a function : Calling a function

Using a function is also known as “calling” it.

To call a function, you just type the function’s name, followed by
a set of brackets that contain any parameters you want
the function to work with.
Parameters are a bit like variables that belong to the function, and
they allow you to pass data between different parts of your program.

When a function doesn’t need any parameters, the
brackets are left empty.

Built in functions

Python has a number of built-in functions that you can use in your code. These are helpful tools that let you do lots of tasks, from inputting information and showing messages on the screen to converting one type of data into another. You've already used some of Python's built-in functions, such as `print()` and `input()`. Have a look at these examples. Why not try them out in the shell?

This asks the user to type in their name.

```
>>> name = input('What is your name?')
What is your name? Sara
>>> greeting = 'Hello' + name
>>> print(greeting)
Hello Sara
```

This shows the content of the variable `greeting` on the screen.

△ `input()` and `print()`

These two functions are like opposites. The `input()` function lets the user give instructions or data to the program by typing them in. The `print()` function sends output to the user by displaying messages or results on the screen.

Another way of calling

Some of the different types of data we've come across so far, such as integers, strings, and lists, have their own functions. These functions must be called in a special way. You type the data or the name of the variable holding the data, followed by a dot, the function's name, and finally brackets. Test out these code snippets in the shell.

The function has two parameters.

```
>>> message = 'Python makes me happy'
>>> message.replace('happy', ':D')
'Python makes me :D'
```

The new string replaces **happy** with `:D`.

△ `replace()`

Two parameters are needed for this function: the first is the part of a string you want to replace, while the second is the string you want to put in its place. The function returns a new string with the replacements made.

Don't forget the dot.

Empty brackets mean that no parameters are needed.

```
>>> 'bang'.upper()
'BANG'
```

This is the new string, all in capitals.

△ `upper()`

The `upper()` function takes an existing string and returns a new string in which all the lower-case characters are changed to upper-case (capitals).

The list of numbers stored in the variable

```
>>> countdown = [1, 2, 3]
>>> countdown.reverse()
>>> print(countdown)
[3, 2, 1]
```

The list is now reversed.

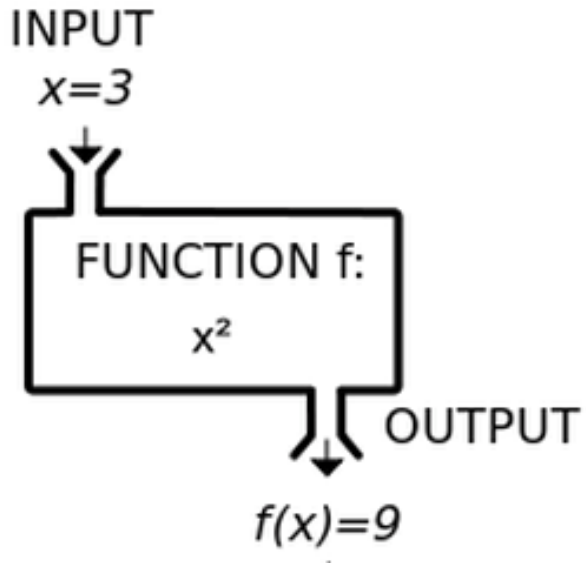
△ `reverse()`

Use this function when you want to reverse the order of the items in a list. Here, it's used to reverse a list of numbers stored in the variable `countdown`. Instead of printing the list as `[1, 2, 3]`, the function makes it print `[3, 2, 1]`.

Functions returning some value

A return is a value that a function returns to the calling script or function when it completes its task.

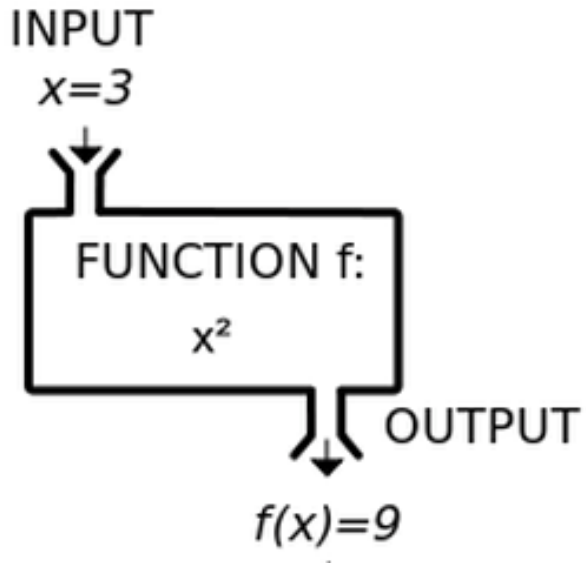
A return value can be of any type. The type of value your function returns depends largely on the task it performs.

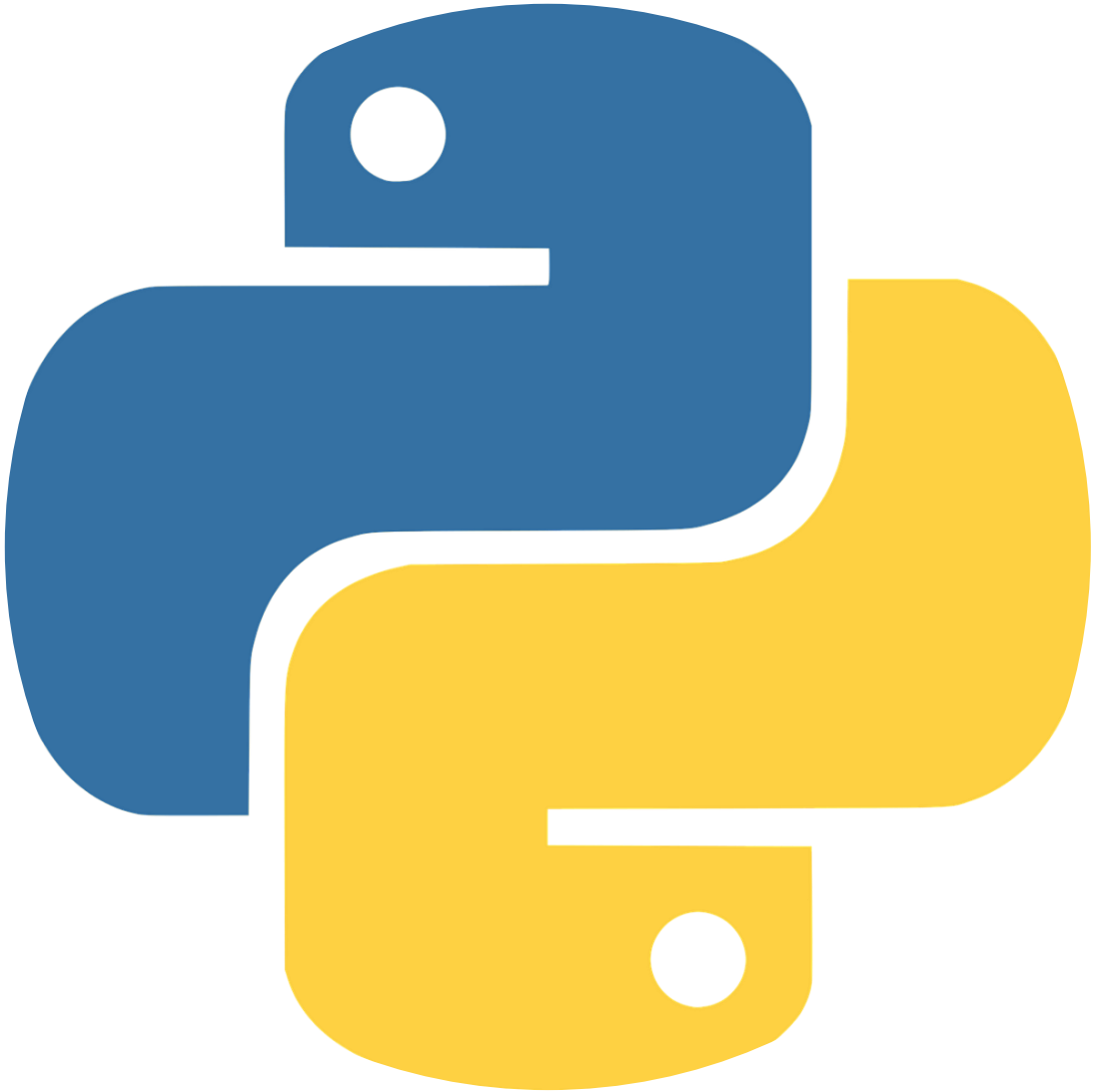


Functions returning some value

A return is a value that a function returns to the calling script or function when it completes its task.

A return value can be of any type. The type of value your function returns depends largely on the task it performs.





Let's make some functions

LINGO

Function terms

There are a number of special words that coders use when talking about functions.

Call To use a function.

Define When you use the `def` keyword and write the code for a function, coders say you "define" the function. You also define a variable when you first set its value.

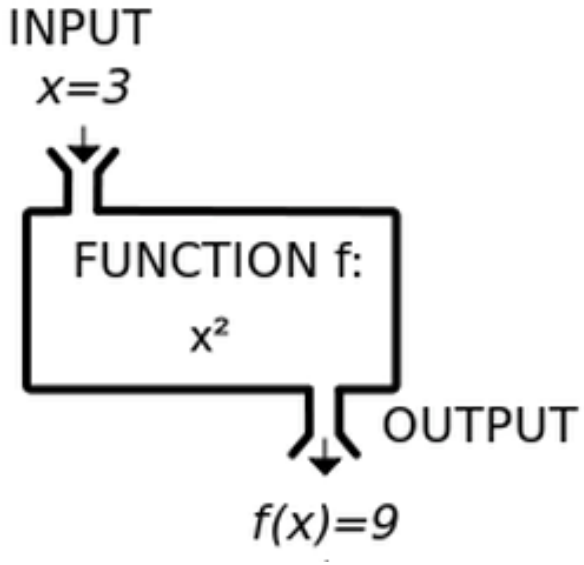
Parameter A piece of data (information) that you give to a function to use.

Return value Data that you pass from a function back to the main code. You get it using the keyword `return`.

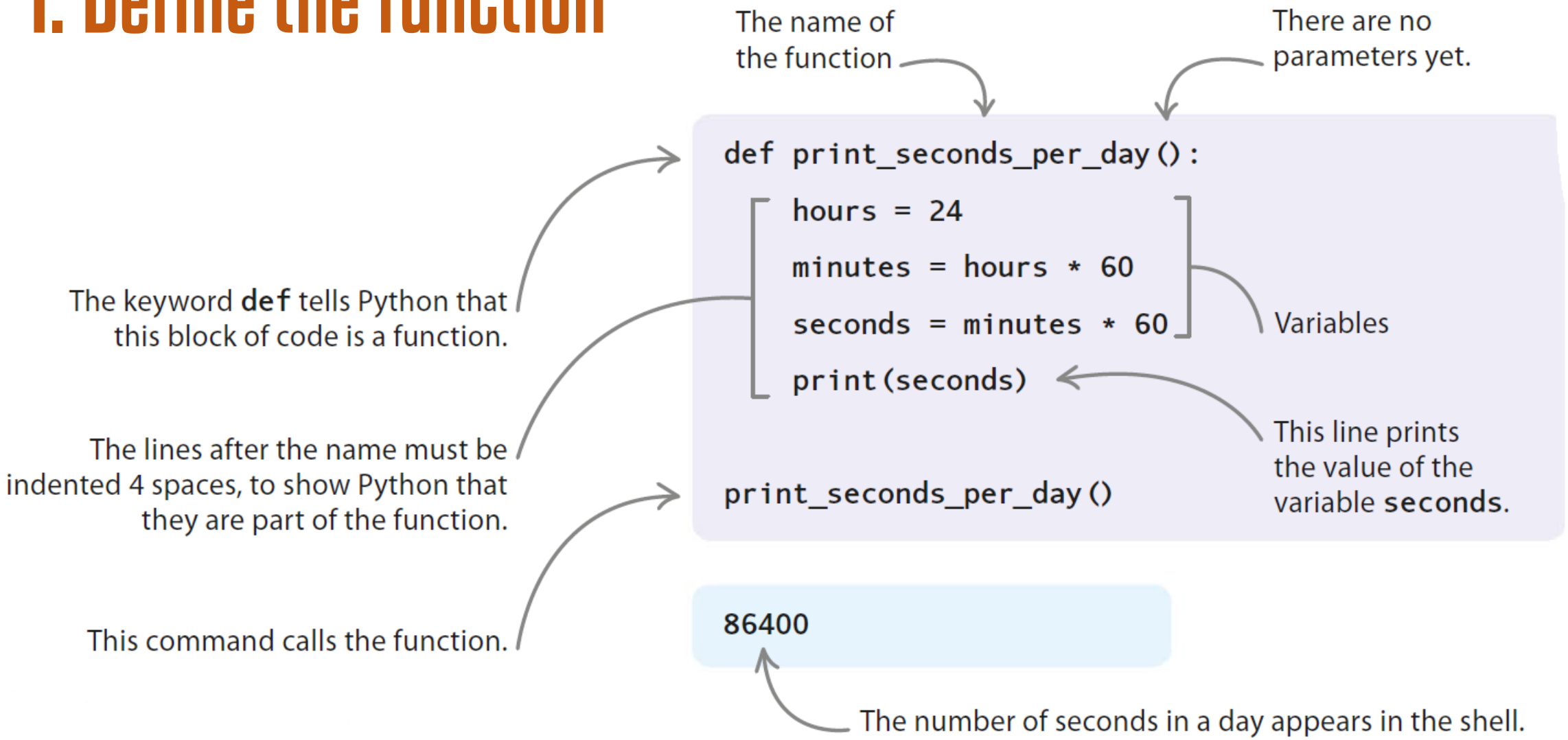
Making a function

The best functions have a clear purpose and a good name that explains what they do.

Follow these instructions to create, or “define”, a function that calculates the number of seconds in a day and then prints the answer on the screen.



1. Define the function

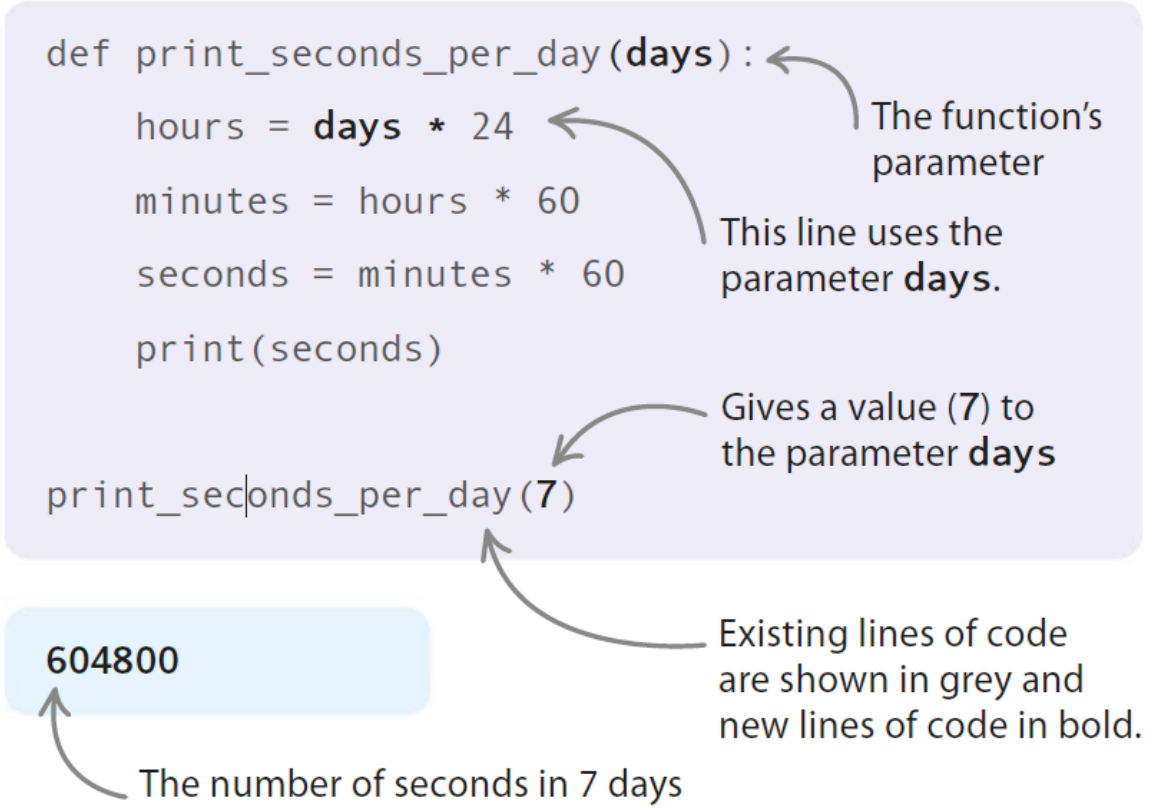


2. Add parameters

If you want to give your function any values to work with, you put them inside the brackets as parameters.

For example, to find out the total number of seconds in a particular number of days, change your code to look like this.

The function now has the parameter **days**. You can specify the number of days when you call the function. Try it out yourself.

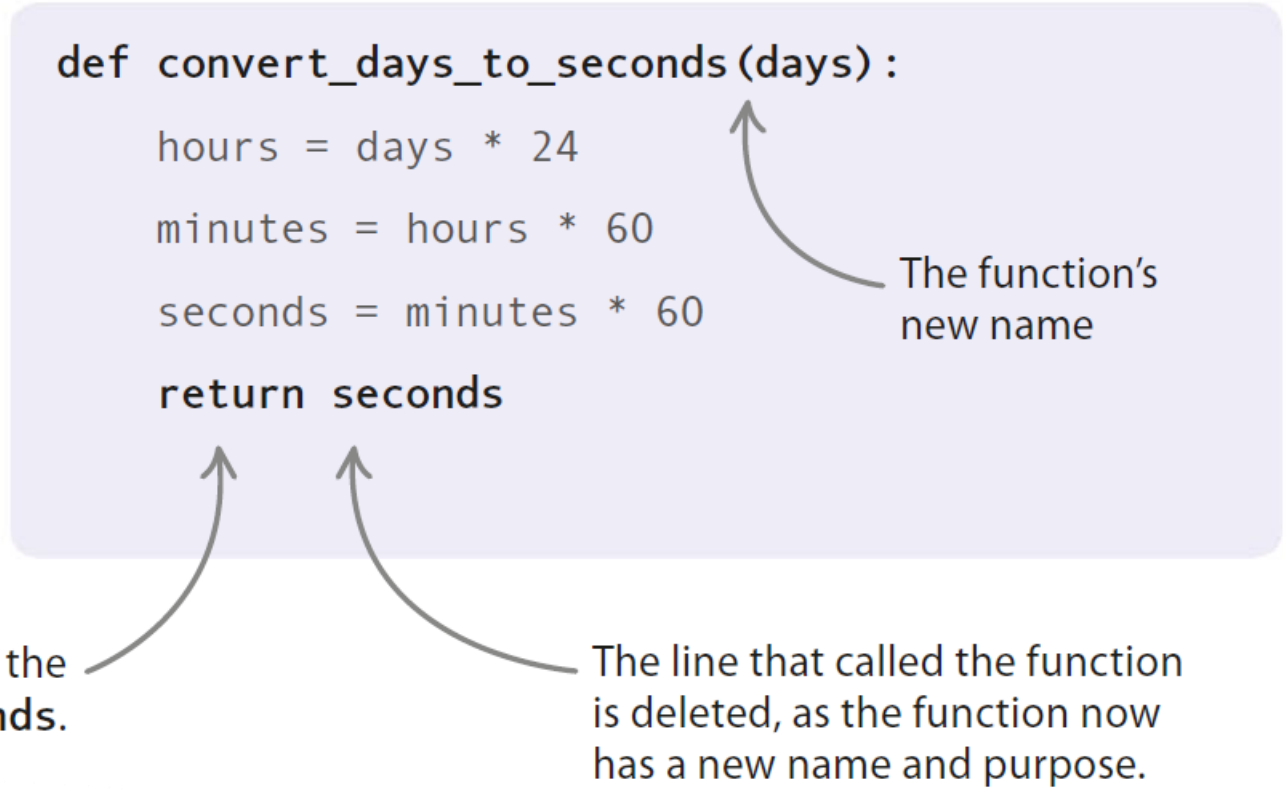


3. Add parameters

Once you have a function that does something useful, you'll want to use the results from that function in the rest of your code.

You can get values out of a function by "returning" them.

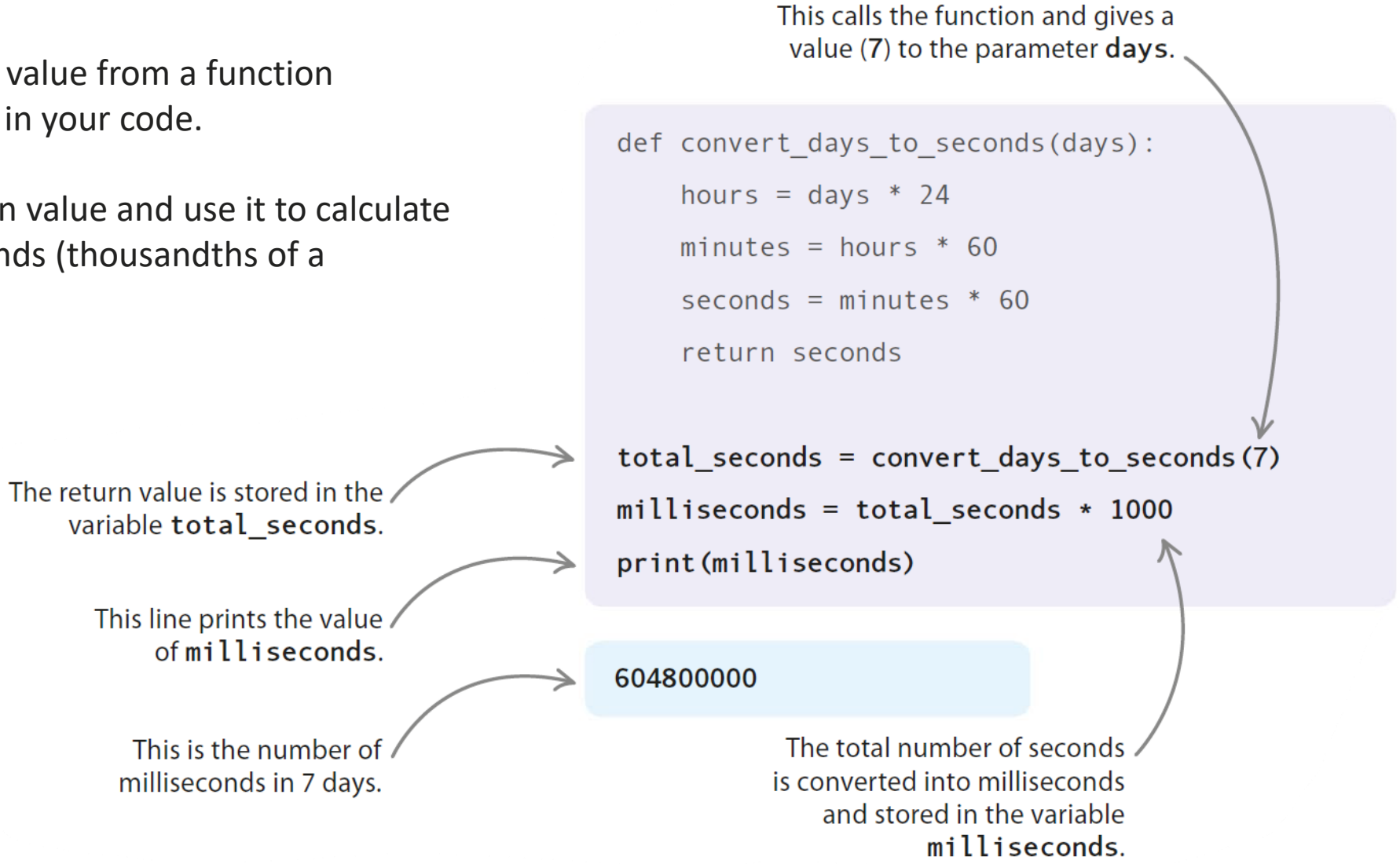
You should rename the function to match its new purpose(we are not printing anymore)



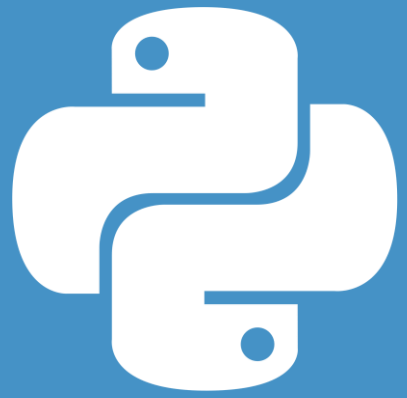
3. Store and use the return value

You can store the return value from a function in a variable to use later in your code.

Here, we store the return value and use it to calculate the number of milliseconds (thousandths of a second).



Time to code



Guess the number using functions