Why would you use the return statement in Python?

What is the simple basic explanation of what the return statement is, how to use it in Python?

And what is the difference between it and the `print` statement?

python printing return

edited Feb 11 '14 at 8:13 — Bleeding Fingers — 3,808 7 21 53

asked Aug 20 '11 at 2:44 — Hitesh Kumar — 170 1 3 4

10 Answers

The `print()` function writes, i.e., "prints", a string in the console. The `return` statement causes your function to exit and hand back a value to its caller. The point of functions in general is to take in inputs and return something. The `return` statement is used when a function is ready to return a value to its caller.

For example, here's a function utilizing both `print()` and `return`:

```
def foo():
    print("hello from inside of foo")
    return 1
```

Now you can run code that calls foo, like so:

```
if __name__ == '__main__':
    print("going to call foo")
    x = foo()
    print("called foo")
    print("foo returned " + str(x))
```

If you run this as a script (e.g. a `.py` file) as opposed to in the Python interpreter, you will get the following output:

```
going to call foo
hello from inside foo
called foo
foo returned 1
```

I hope this makes it clearer. The interpreter writes return values to the console so I can see why somebody could be confused.

Here's another example from the interpreter that demonstrates that:

```
>>> def foo():
...     print("hello from within foo")
...     return 1
```

```
...
>>> foo()
hello from within foo
1
>>> def bar():
...   return 10 * foo()
...
>>> bar()
hello from within foo
10
```

You can see that when `foo()` is called from `bar()` , 1 isn't written to the console. Instead it is used to calculate the value returned from `bar()` .

`print()` is a function that causes a side effect (it writes a string in the console), but execution resumes with the next statement. `return` causes the function to stop executing and hand a value back to whatever called it.

edited Jun 2 '16 at 22:04

user2357112
92.6k 8 70 142

answered Aug 20 '11 at 2:47

Nathan Hughes
59.4k 13 102 155

You need to use `"foo returned " + str(x)` or else you'll get `TypeError: cannot concatenate 'str' and 'int' objects` . – icktoofay Aug 20 '11 at 2:54

@icktoofay: fixed, thank you – Nathan Hughes Aug 20 '11 at 2:56

I think the *dictionary* is your best reference here

Return and Print

In short:

return *gives something back* or *replies* to the caller of the function while **print** *produces text*

answered Aug 23 '11 at 4:47

Kimvais
20.1k 7 71 106

4 Why the downvote? – Kimvais Aug 24 '11 at 12:22

Although it may have been unintentional, the tone of your answer may have been interpreted as follows: "This is so obvious because it's in the dictionary." To me, that tone comes across because you ended with the actual answer (rather than starting with it), and you italicized the word "dictionary." Again, I recognize that the tone I "heard" may not have been the tone you intended. – Kevin Markham Oct 13 '16 at 14:26

Think of the print statement as causing a ***side-effect***, it makes your function write some text out to the user, but it can't be **used by another function.**

I'll attempt to explain this better with some examples, and a couple definitions from Wikipedia.

Here is the definition of a function from Wikipedia

A function, in mathematics, associates one quantity, the argument of the function, also known as the input, with another quantity, the value of the function, also known as the output..

Think about that for a second. What does it mean when you say the function has a value?

What it means is that you can actually substitute the value of a function with a normal value! (Assuming the two values are the same type of value)

Why would you want that you ask?

What about other functions that may accept the same type of value as an *input*?

```
def square(n):
    return n * n

def add_one(n):
    return n + 1
```

```
print square(12)

# square(12) is the same as writing 144

print add_one(square(12))
print add_one(144)
#These both have the same output
```

There is a fancy mathematical term for functions that only depend on their inputs to produce their outputs: Referential Transparency. Again, a definition from Wikipedia.

Referential transparency and referential opaqueness are properties of parts of computer programs. An expression is said to be referentially transparent if it can be replaced with its value without changing the behavior of a program

It might be a bit hard to grasp what this means if you're just new to programming, but I think you will get it after some experimentation. In general though, you can do things like print in a function, and you can also have a return statement at the end.

Just remember that when you use return you are basically saying "A call to this function is the same as writing the value that gets returned"

Python will actually insert a return value for you if you decline to put in your own, it's called "None", and it's a special type that simply means nothing, or null.

answered Aug 23 '11 at 5:12

Wes
1,591 1 9 23

`return` means, "output this value from this function".

`print` means, "send this value to (generally) stdout"

In the Python REPL, a function return will be output to the screen by default (this isn't quite the same as print).

This is an example of print:

```
>>> n = "foo\nbar" #just assigning a variable. No output
>>> n #the value is output, but it is in a "raw form"
'foo\nbar'
>>> print n #the \n is now a newline
foo
bar
>>>
```

This is an example of return:

```
>>> def getN():
...     return "foo\nbar"
...
>>> getN() #When this isn't assigned to something, it is just output
'foo\nbar'
>>> n = getN() # assigning a variable to the return value. No output
>>> n #the value is output, but it is in a "raw form"
'foo\nbar'
>>> print n #the \n is now a newline
foo
bar
>>>
```

answered Aug 20 '11 at 2:59

cwallenpoole
48.6k 13 80 123

In python, we start defining a function with "def" and end the function with "return".

A function of variable x is denoted as f(x). What this function does? Suppose, this function adds 2 to x. So, f(x)=x+2

Now, the code of this function will be:

```
def A_function (x):
    return x + 2
```

After defining the function, you can use that for any variable and get result. Such as:

```
print A_function (2)
>>> 4
```

We could just write the code slightly differently, such as:

```
def A_function (x):
    y = x + 2
    return y
print A_function (2)
```

That would also give "4".

Now, we can even use this code:

```
def A_function (x):
    x = x + 2
    return x
print A_function (2)
```

That would also give 4. See, that the "x" beside return actually means (x+2), not x of "A_function(x)".

I guess from this simple example, you would understand the meaning of return command.

answered May 18 '15 at 7:22

Sheikh Ahmad Shah
96 1 2 10

Just to add to @Nathan Hughes's excellent answer:

The `return` statement can be used as a kind of control flow. By putting one (or more) `return` statements in the middle of a function, we can say: "stop executing this function. We've either got what we wanted or something's gone wrong!"

Here's an example:

```
>>> def make_3_characters_long(some_string):
...     if len(some_string) == 3:
...         return False
...     if str(some_string) != some_string:
...         return "Not a string!"
...     if len(some_string) < 3:
...         return ''.join(some_string,'x')[:,3]
...     return some_string[:,3]
...
>>> threechars = make_3_characters_long('xyz')
>>> if threechars:
...     print threechars
... else:
...     print "threechars is already 3 characters long!"
...
threechars is already 3 characters long!
```

See the Code Style section of the Python Guide for more advice on this way of using `return`.

edited Oct 2 '14 at 18:06

Christa
112 1 1 8

answered Aug 6 '13 at 14:52

LondonRob
16.1k 7 39 79

Difference between "return" and "print" can also be found in the following example:

RETURN:

```
def bigger(a, b):
    if a > b:
        return a
    elif a <b:
        return b
    else:
        return a
```

The above code will give correct results for all inputs.

PRINT:

```
def bigger(a, b):
    if a > b:
        print a
```

```
    elif a <b:
        print b
    else:
        print a
```

NOTE: This will fail for many test cases.

ERROR:

```
----

FAILURE  : Test case input: 3, 8.

            Expected result: 8

FAILURE  : Test case input: 4, 3.

            Expected result: 4

FAILURE  : Test case input: 3, 3.

            Expected result: 3


You passed 0 out of 3 test cases
```

edited May 11 '14 at 16:15

Ashwini Chaudhary
141k 22 225 302

answered May 11 '14 at 16:11

Pj_
673 4 11

Here is my understanding. (hope it will help someone and it's correct).

```
def count_number_of(x):
    count = 0
    for item in x:
        if item == "what_you_look_for":
        count = count + 1
    return count
```

So this simple piece of code counts number of occurrences of something. The placement of return is significant. It tells your program where do you need the value. So when you print, you send output to the screen. When you return you tell the value to go somewhere. In this case you can see that count = 0 is indented with return - we want the value (count + 1) to replace 0. If you try to follow logic of the code when you indent the return command further the output will always be 1, because we would never tell the initial count to change. I hope I got it right. Oh, and return is always inside a function.

edited Nov 18 '13 at 0:21

answered Nov 18 '13 at 0:15

piotrektt
56 5

`return` should be used for **recursive** functions/methods or you want to use the returned value for later applications in your algorithm.

`print` should be used when you want to display a meaningful and desired output to the user and you don't want to clutter the screen with intermediate results that the user is not interested in, although they are helpful for debugging your code.

The following code shows how to use `return` and `print` properly:

```
def fact(x):
    if x < 2:
        return 1
    return x * fact(x - 1)

print(fact(5))
```

This explanation is true for all of the programming languages not just **python**.

answered Mar 17 '16 at 22:00

Habib Karbasian
19 1 6

`return` is part of a function definition, while `print` outputs text to the standard output (usually the

console).

A function is a procedure accepting parameters and returning a value. `return` is how the latter, while the former is done with `def` .

Example:

```
def timestwo(x):
    return x*2
```

answered Sep 19 '16 at 13:38

icarito
53 6

---

protected by Tunaki Jan 31 '16 at 15:40

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?