



Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

[Take the 2-minute tour](#)

Is it pythonic for a function to return multiple values?

In python, you can have a function return multiple values. Here's a contrived example:

```
def divide(x, y):  
    quotient = x/y  
    remainder = x % y  
    return quotient, remainder
```


```
(q, r) = divide(22, 7)
```

This seems very useful, but it looks like it can also be abused ("Well...function X already computes what we need as an intermediate value. Let's have X return that value also").

When should you draw the line and define a different method?


[python](#) [function](#) [return-value](#)

edited May 29 '12 at 11:25

 [user](#)
1,523 ●5 ●16 ●33

[add a comment](#)

asked Sep 14 '08 at 20:15

 [Readonly](#)
52.9k ●70 ●152 ●181

9 Answers

Absolutely (for the example you provided).

Tuples are first class citizens in Python

There is a builtin function `divmod()` that does exactly that.

```
q, r = divmod(x, y) # ((x - x%y)/y, x%y) Invariant: div*y + mod == x
```

There are other examples: `zip`, `enumerate`, `dict.items`.

```
for i, e in enumerate([1, 3, 3]):  
    print "index=%d, element=%s" % (i, e)  
  
# reverse keys and values in a dictionary  
d = dict((v, k) for k, v in adict.items()) # or  
d = dict(zip(adict.values(), adict.keys()))
```

BTW, parentheses are not necessary most of the time. Citation from [Python Library Reference](#):

Tuples are constructed by the comma operator (not within square brackets), with or without enclosing parentheses, but an empty tuple must have the enclosing parentheses, such as `a, b, c` or `()`. A single item tuple must have a trailing comma, such as `(d,)`.

Functions should serve single purpose

Therefore they should return a single object. In your case this object is a tuple. Consider tuple as an ad-hoc compound data structure. There are languages where almost every single function returns multiple values (list in Lisp).

Sometimes it is sufficient to return `(x, y)` instead of `Point(x, y)`.

Named tuples

With the introduction of named tuples in Python 2.6 it is preferable in many cases to return named tuples

instead of plain tuples.

```
>>> import collections
>>> Point = collections.namedtuple('Point', 'x y')
>>> x, y = Point(0, 1)
>>> p = Point(x, y)
>>> x, y, p
(0, 1, Point(x=0, y=1))
>>> p.x, p.y, p[0], p[1]
(0, 1, 0, 1)
>>> for i in p:
...     print(i)
...
0
1
```

edited Sep 13 '12 at 0:49



Warren Blanchet

901 ●9 ●12

answered Sep 14 '08 at 20:54



J.F. Sebastian

112k ●21 ●184 ●298

1 The ability to use tuples like that is so handy. I really miss having that ability in Java some times. – [MBCook](#) Feb 25 '09 at 0:03

2 Thanks a lot for mentioning Named tuples. I've been looking for something like this. – [vobject](#) Sep 10 '09 at 8:11

[add a comment](#)

Firstly, note that Python allows for the following (no need for the parenthesis):

```
q, r = divide(22, 7)
```

Regarding your question, there's no hard and fast rule either way. For simple (and usually contrived) examples, it may seem that it's always possible for a given function to have a single purpose, resulting in a single value. However, when using Python for real-world applications, you quickly run into many cases where returning multiple values is necessary, and results in cleaner code.

So, I'd say do whatever makes sense, and don't try to conform to an artificial convention. Python supports multiple return values, so use it when appropriate.

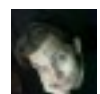
edited Sep 15 '08 at 3:44



Joel Spolsky ♦

23.2k ●14 ●64 ●85

answered Sep 14 '08 at 20:42



Jason Etheridge

3,944 ●4 ●18 ●28

I'd be interested in seeing such a case, could you show one? – [Vinko Vrsalovic](#) Sep 14 '08 at 20:46

3 comma creates tuple, therefore expression: `q, r` is a tuple. – [J.F. Sebastian](#) Sep 14 '08 at 21:27

Vinko, an example is `tempfile.mkstemp()` from the standard library, which returns a tuple containing a file handle and an absolute path. JFS, yes, you're right. [Jason Etheridge](#) Sep 15 '08 at 1:24

But that IS a single purpose... You can have single purpose and multiple return values – [Vinko Vrsalovic](#) Sep 15 '08 at 6:26

If other languages could have multiple return types, I wouldn't be stuck with references and 'out' parameters. – [orip](#) Dec 5 '08 at 1:00

[show 1 more comment](#)

The example you give is actually a python builtin function, called `divmod`. So someone, at some point in time, thought that it was pythonic enough to include in the core functionality.

To me, if it makes the code cleaner, it is pythonic. Compare these two code blocks:

```
seconds = 1234
minutes, seconds = divmod(seconds, 60)
hours, minutes = divmod(minutes, 60)
```

```
seconds = 1234
minutes = seconds / 60
seconds = seconds % 60
hours = minutes / 60
minutes = minutes % 60
```

answered Sep 14 '08 at 20:55



Nathan Jones

1,610 ●1 ●9 ●7

[add a comment](#)

Yes, returning multiple values (i.e., a tuple) is definitely pythonic. As others have pointed out, there are plenty of examples in the Python standard library, as well as in well-respected Python projects. Two additional comments:

1. Returning multiple values is sometimes very, very useful. Take, for example, a method that optionally handles an event (returning some value in doing so) and also returns success or failure. This might arise in a chain of responsibility pattern. In other cases, you want to return multiple, closely linked pieces of data---as in the example given. In this setting, returning multiple values is akin to returning a single instance of an anonymous class with several member variables.
2. Python's handling of method arguments necessitates the ability to directly return multiple values. In C++, for example, method arguments can be passed by reference, so you can assign output values to them, in addition to the formal return value. In Python, arguments are passed "by reference" (but in the sense of Java, not C++). You can't assign new values to method arguments and have it reflected outside method scope. For example:

```
// C++
void test(int& arg)
{
    arg = 1;
}

int foo = 0;
test(foo); // foo is now 1!
```

Compare with:

```
# Python
def test(arg):
    arg = 1

foo = 0
test(foo) # foo is still 0
```

edited Oct 21 '12 at 8:27



badp

5,270 ●1 ●27 ●54

answered Sep 15 '08 at 15:47



zweiterlinde

6,174 ●1 ●17 ●23

"a method that optionally handles an event and also returns success or failure" - That's what exceptions are for. – [Nathan](#) Oct 19 '10 at 19:52

It's very widely accepted that exceptions are for "exceptional" conditions only, not for mere success or failure. Google "error codes vs exceptions" for some of the discussion. – [zweiterlinde](#) Nov 6 '10 at 18:43

[add a comment](#)

OT: RSRE's Algol68 has the curious "[:=" operator. eg.

```
INT quotient:=355, remainder;
remainder := (quotient [:= 113);
```

Giving a quotient of 3, and a remainder of 16.

Note: typically the value of "(x[:=y)" is discarded as quotient "x" is assigned by reference, but in RSRE's case the returned value is the remainder.

c.f. [Integer Arithmetic - Algol68](#)

answered Mar 12 '09 at 21:48



NevilleDNZ

458 ●3 ●16

[add a comment](#)

It's definitely pythonic. The fact that you can return multiple values from a function the boilerplate you would have in a language like C where you need to define a struct for every combination of types you return somewhere.

However, if you reach the point where you are returning something crazy like 10 values from a single function, you should seriously consider bundling them in a class because at that point it gets unwieldy.

answered Sep 15 '08 at 14:46




indentation

2,333 ●5 ●12 ●12

[add a comment](#)

Returning a tuple is cool. Also note the new namedtuple which was added in python 2.6 which may make this more palatable for you: <http://docs.python.org/dev/library/collections.html#collections.namedtuple>

answered Sep 15 '08 at 15:14



pixelbeat

12.7k 6 25 35

add a comment

I'm fairly new to Python, but the tuple technique seems very pythonic to me. However, I've had another idea that may enhance readability. Using a dictionary allows access to the different values by name rather than position. For example:

```
def divide(x, y):  
    return {'quotient': x/y, 'remainder':x%y }  
  
answer = divide(22, 7)  
print answer['quotient']  
print answer['remainder']
```

answered Sep 15 '08 at 21:05



Fred Larson

28.7k 5 65 99


- 1
- Don't do this. You can't assign the result in a one-liner, it's very clunky. (Unless you want to store the result, in which case it's better to put this into a method.) If your intent with the dict is to self-document the return values, then a) give the fn a reasonably explanatory name (like "divmod") and b) put the rest of the explanation into a docstring (which is the Pythonic place to put them); If that docstring requires longer than two lines, that suggests the function is being thematically overloaded and may be a bad idea. – smci May 23 '11 at 21:18

add a comment

It's fine to return multiple values using a tuple for simple functions such as `divmod` . If it makes the code readable, it's Pythonic.

If the return value starts to become confusing, check whether the function is doing too much and split it if it is. If a big tuple is being used like an object, make it an object. Also, consider using [named tuples](#), which will be part of the standard library in Python 2.6.

answered Sep 15 '08 at 14:43



Will Harris

17.2k 9 43 60

add a comment

Not the answer you're looking for? Browse other questions tagged [python](#) [function](#) [return-value](#) or [ask your own question](#).