***\*args and \*\*kwargs in python explained***

[AUGUST 4, 2013](http://freepythontips.wordpress.com/2013/08/04/args-and-kwargs-in-python-explained/)[YASOOB](http://freepythontips.wordpress.com/author/yasoob008/)[ARGS](http://freepythontips.wordpress.com/tag/args/), [ARGS AND KWARGS](http://freepythontips.wordpress.com/tag/args-and-kwargs/),[KWARGS](http://freepythontips.wordpress.com/tag/kwargs/), [PYTHON](http://freepythontips.wordpress.com/tag/python/), [PYTHON DECORATOR ARGS KWARGS](http://freepythontips.wordpress.com/tag/python-decorator-args-kwargs/),[PYTHON FUNCTION ARGS KWARGS](http://freepythontips.wordpress.com/tag/python-function-args-kwargs/), [PYTHON SUPER ARGS KWARGS](http://freepythontips.wordpress.com/tag/python-super-args-kwargs/)[17 COMMENTS](http://freepythontips.wordpress.com/2013/08/04/args-and-kwargs-in-python-explained/#comments)

Hi there folks. I have come to see that most new python programmers have a hard time figuring out the \*args and \*\*kwargs magic variables. So what are they ? First of all let me tell you that it is not necessary to write \*args or \*\*kwargs. Only the \* (aesteric) is necessary. You could have also written \*var and \*\*vars. Writing \*args and \*\*kwargs is just a convention. So now lets take a look at \*args first.

**Usage of \*args**  
\*args and \*\*kwargs are mostly used in function definitions. \*args and \*\*kwargs allow you to pass a variable number of arguments to a function. What does variable mean here is that you do not know before hand that how many arguments can be passed to your function by the user so in this case you use these two keywords. \*args is used to send a **non-keyworded** variable length argument list to the function. Here’s an example to help you get a clear idea:

def test\_var\_args(f\_arg, \*argv):

print "first normal arg:", f\_arg

for arg in argv:

print "another arg through \*argv :", arg

test\_var\_args('yasoob','python','eggs','test')

This produces the following result:

first normal arg: yasoob

another arg through \*argv : python

another arg through \*argv : eggs

another arg through \*argv : test

I hope this cleared away any confusion that you had. So now lets talk about \*\*kwargs

**Usage of \*\*kwargs**  
\*\*kwargs allows you to pass **keyworded** variable length of arguments to a function. You should use \*\*kwargs if you want to handle **named arguments** in a function. Here is an example to get you going with it:

def greet\_me(\*\*kwargs):

if kwargs is not None:

for key, value in kwargs.iteritems():

print "%s == %s" %(key,value)

>>> greet\_me(name="yasoob")

name == yasoob

So can you see how we handled a keyworded argument list in our function. This is just the basics of \*\*kwargs and you can see how useful it is. Now lets talk about how you can use \*args and \*\*kwargs to call a function with a list or dictionary of arguments.

**Using \*args and \*\*kwargs to call a function**  
So here we will see how to call a function using \*args and \*\*kwargs. Just consider that you have this little function:

def test\_args\_kwargs(arg1, arg2, arg3):

print "arg1:", arg1

print "arg2:", arg2

print "arg3:", arg3

Now you can use \*args or \*\*kwargs to pass arguments to this little function. Here’s how to do it:

# first with \*args

>>> args = ("two", 3,5)

>>> test\_args\_kwargs(\*args)

arg1: two

arg2: 3

arg3: 5

# now with \*\*kwargs:

>>> kwargs = {"arg3": 3, "arg2": "two","arg1":5}

>>> test\_args\_kwargs(\*\*kwargs)

arg1: 5

arg2: two

arg3: 3

**Order of using \*args \*\*kwargs and formal args**  
So if you want to use all three of these in functions then the order is

some\_func(fargs,\*args,\*\*kwargs)

I hope you have understood the usage of \*args and \*\*kwargs. If you have got any problems or confusions with this then feel free to comment below. For further study i suggest the official [python docs on defining functions](http://docs.python.org/tutorial/controlflow.html#more-on-defining-functions) and [\*args and \*\*kwargs on stackoverflow](http://stackoverflow.com/questions/3394835/args-and-kwargs).