Python Tips

Your daily dose of bite sized python tips

Advertisements

*args and **kwargs in python explained

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 and *args and **kwargs on stackoverflow.

You might also like :
*) Making a url shortener in python
*) 20 Python libraries you can't live without
*) Targeting python 2 and 3 at the same time.

SHOW YOUR LOVE BY SHARING THIS:

More

Like

31 bloggers like this.

RELATED:

All About Decorators in Python
In "python"

The self variable in python explained
In "python"

Python socket network programming
In "python"

August 4, 2013 Yasoob python args, args and kwargs, kwargs, python, python decorator args kwargs, python function args kwargs, python super args kwargs

73 thoughts on "*args and **kwargs in python explained"

Pingback: Lambda Functions (And Friends!) – ewoodworth

Pingback: ESP32 / ESP8266 MicroPython: HTTP POST Requests | techtutorialsx

Pingback: Stepping through the Code – Part 1 – Pathfinder

Rommel Rodriguez Perez

October 21, 2017 at 2:44 am

Great article, thanks!.

airwavves

November 5, 2017 at 12:49 am

Cleared things up for and didn't waste my time.

Pingback: Final Summative Project | Seong Won C's Blog

Vanen Dallas

January 24, 2018 at 4:04 pm

Awesome! Thank you for explaining this with such clarity

Vicente

November 18, 2018 at 6:23 am

¡Magnificent!

nkem

November 20, 2018 at 7:05 pm

thanks. I appreciate your clear explanations

do.OJi (@beltreg)

December 11, 2018 at 8:56 am

Thank you. Great post.

Robery Davis

December 31, 2018 at 5:07 pm

Very clear and well written article. It cleared up the mystery for me even though I have been dabbling in python for 17 years. Much better written then the official documentation explanation. And that was because it was written in Plain English and NOT geekese.

mmagic

March 15, 2019 at 1:06 pm

I like your explanation style, makes a convoluted but simple features of Python, simple to understand for normal human. This should be inside the official python documentation.