

# Notes - Friday 01/15/21

## Friday 01/15/21

Today I wanted to work on the test and comprehension part of the course [which can be seen here](#). They threw in something called `*args` and `**kwargs` in the middle of my coding practice. So let's learn about args.

### `*args` and `**kwargs`

#### `*args`

These are functional parameters. Arguments are `*args` and pronounced "star args" and keyword arguments are `**kwargs` and pronounced "double star kw-ar-gs". These are things in Python functions that allow for the code to accept an arbitrary number of arguments and keyword arguments without pre-defining parameters in the function calls.

So let's say we want to setup a function that will return 5% of the sum of 2 numbers. But what if I then want to change the function to take a variable or arbitrary amount of arguments. So then I don't have to define the number of numbers the function will do. Note that the `*args` will become a tuple. The `*args` could technically be anything; it could be `*urwordshere` or `*spam`. Best practice is to call it `*args`.

CBAkwarg01.png

### `**kwargs`

Python offers a way to handle an arbitrary number of key word arguments. It creates a dictionary of key value pairs. So it does the same thing that `*args` does where that returns a tuple, but instead returns a dictionary. Then the user can define values in the dictionary that can be messed with inside the function. Again `**kwargs` could be anything after `**urusernamehere`. Best practice is to call it `**kwargs` so it's easy to recognize.

# Why \*args and \*\*kwargs?

They're useful to use when you pull in outside libraries. They might not be useful now but will be useful later. Note that they can be combined

CBAkwarg02.png

## Things I learned from the Test - Skyline question

The question being asked was:

"Define a function called **myfunc** that takes in a string, and returns a matching string where every even letter is uppercase and every odd letter is lower case. Assume that the incoming string only contains letters and don't worry about the numbers, spaces, or punctuation. The output string can start with either upper or lower case. The letters should alternate throughout the string. Just provide the definition for the function. You don't need to run it. Remember you need to use the *return* function and not *print*."

The answer was:

```
def myfunc(x):
    out = []
    for i in range(len(x)):
        if i%2==0:
            out.append(x[i].lower())
        else:
            out.append(x[i].upper())
    return ''.join(out)
```

Things I learned:

- I didn't realize that you could combine range and length to count the number of letters.
- .join - Joins all items in a tuple into a string with a hash character as a separator
- Remember that .append is what modifies the list that variable out calls out
- An explanation on what is going on in this code [can be found here](#)

---

Revision #2

Created 12 January 2021 23:31:06 by cba88

Updated 16 January 2021 05:15:14 by cba88