Function quick ref

```
def functionname(parameter1, parameter2): # define
    result = parameter1 + parameter2 # do stuff
    return result # return only one thing

hw_result = functionname("hello", "world") # call the function
print(hw_result)
```

1. Design phase (remember that these are thinking tasks here, so you can answer phase 1 items in your own natural lanugage)
    1. what do you want to name the function?
        - follow normal variable naming protocols
    2. what should the function take as input, if anything?
        - You can have 0-many parameter (inputs)
    3. what does the function need to do? (this will be your do stuff section)
    4. what should the function return? or: what should the function give back to you
        - While you can choose to not return something, we won't be doing that.
        - You must pick one and only one object to return. You can get away with returning multiple things by returning a data collection object, but we'll be sticking to strings and numbers for now.
2. Defining phase (remember that this phase only teaches Python about your function. None of the code in the body of the function has actually been executed after this phase).
    1. Add the `def` block line in, providing the function name you chose.
        - So you'll have `def whatever():` thus far
    2. Add the parameter variables into the `()` of the def block
        - Give these sensible names that you'll remember later, do not just ay x and y.
        - You will provide these in the order that they should be provided when calling the function, and the variable names separated bf commas. Leave the `()` empty if you aren't requiring that the function take any input.
        - So you'll have `def whatever(thing1, thing2):` at this point
    3. Inside the function definition block (so indented under the `def` line), add your do stuff items.

- So you'll have

```
def whatever(thing1, thing2):
    new = thing1 + thing2
```

4. Add your return statement

    - this is a separate step because it will always come last in your function.
    - it should always come last because your function execution will immediately stop as soon as Python hits a return statement.
    - `return` is a keyword and not a function, so there aren't any `()`. Provide the single object that you want to return after the keyword. Do not add multiple items after this separated by commas. It'll look like it's working, but not working as you are intending. So yes you can return back data collections.
    - So you'll have

```
def whatever(thing1, thing2):
    new = thing1 + thing2
    return new # see? no ()!
```

3. Calling phase (now is the time to make the code actually work. When you call the function you are finally asking Python to actually execute the code now).

    1. Somewhere later in your code, you'll call the function with the function's name and `()`. You'll add this in your main code indent area, and not inside your function definition.
    2. Provide the input values in the `()` according to how you defined it. So if you said `def student(name, age):`, then you must provide the parameter inputs in that order.
    3. Choose where you want the returned results to go. You essentially have two choices: print it or save it to a variable. I suggest saving it to a variable, that way you have the value to mess with later.

    - So you'll have

```
def whatever(thing1, thing2):
    new = thing1 + thing2
    return new # see? no ()!

myresult = whatever("fizzy", "pop")
```