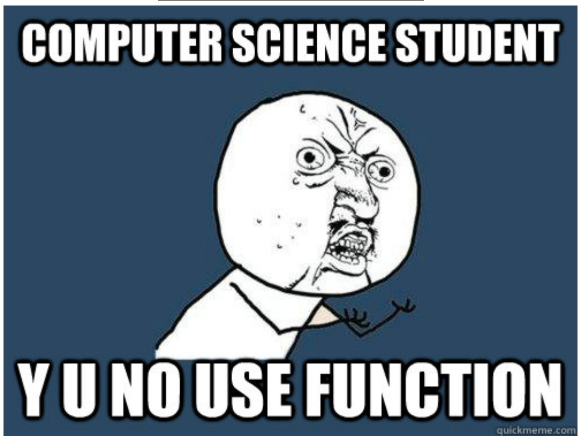
WEEK 5 REVIEW



Functions

- EXACTLY like in algebra/calculus, except you get to design it
- Terminology
  - Function definition/implementation: this is declaring what the function actually DOES
  - Function call: this is USING the function. The function must be defined/implemented in order for another chunk of code to call the function
  - Parameters/Input values/Arguments: what you pass INTO the function. It's whatever is between the parenthesis. A function can take no input, one input, and multiple inputs. Can also set default values for a parameter so if the user wants to pass in a value, they can but they're not required to
    - myFunc(): no input/null paramater/void parameter
    - myFunc(someInteger): Takes one input called someInteger
    - myFunc(someInteger, someString): Takes two inputs
    - myFunc(someInteger=18, someString): Can take in two parameters. The user can choose to enter an integer and they must enter a string. So examples of function calls are: myFunc(23, "Brandon"), myFunc(someString="Brandon")
  - Output/Return Value: This can be null/void (ie there is no return value) or you can return a value. You can also return multiple values (more on this later). A return value, if there is one, is signified by the key word "return." If there is no return value, then as soon as the code block ends, the function automatically returns to its caller
  - Caller = the code that called the function
- If there is a return statement, YOU MUST ALWAYS CATCH THE VALUE in a variable. So say the function sumNumbers(myListOfNums) has a line that looks something like return total, then when I call sumNumbers, my call line should look something like: totalSum = sumNumbers(myListOfNumbers)
- If a variable is declared inside a function, it "dies" at the end of the function
- If you pass a mutable object into a function (like a list) you don't have to return a new list, any changes you make to the list will persist beyond the life of the function