Classes

Steven Walton
PS 232 Computational Methods
Department of Physics
Embry-Riddle Aeronautical University
Prescott, AZ 86301

December 1, 2014

Why do we code?

Seriously, why do we code? It's because we're lazy. No, I'm not joking. Look at the things we have done in class so far. Do you want to do that by hand? Pen and paper? Could you do it by pen and paper? I'm guessing you're saying to yourself that the answer is no, and rightly so. Already we have learned about loops and definitions. They make our life drastically easier. For repetitive tasks computers do things much faster than we could ever imagine to do by hand. And this is why we code. Because we are lazy.
So you may have started to think to yourself at this by the time you may have started asking yourself what we would do if we wanted to use a bunch of definitions over and over again. What if we wanted to create a database of lectures, astronomical data, friends, or whatever. How would we do this? The answer is actually in classes. Let's do a simple example.

```
class Drink:
    def __init__(self,t,c):
        self.t = t
        self.c = c

    description = "Nobody described this wonderful drink, we'll never
                   know..."
    creator = "Nobody staked their claim to this drink, I guess it's
               mine now."

    def drinkType(self):
        return self.t

    def cost(self):
        return self.c

    def describe(self,text):
        self.description = text

    def creatorName(self,text):
        self.creator = text
```

What we have done now is create a class for drinks. This is great if we want to have a bunch of different drinks in our code. No longer do we need to write these definitions over and over again, but we can store

them with classes. So let's play with these classes a little. We have created the code, but we have yet to actually make a drink.

```
espresso = Drink("Coffee",1.25)
few = Drink("Whiskey",4.00)
```

We have now created a drink, but all we have is what type of drink it is and the cost of it. If we ask for "espresso.creator" we get back the default value, and the same thing happens if we ask for the description. So let's modify this a little bit.

```
espresso.describe("Elixir of life")
espresso.creatorName("Becca Tobin")
few.describe("An amazing rye whiskey)
few.creatorName("Steven")
```

Now if we print "few.description" or "espresso.creator" we get back the values that we gave. Note that we have a default value and that the definitions are not the same as these values. So that we have a definition to define the value and the value itself. If we did not do it this way we would not be able to correctly create and store our drinks.
Let's say that we want to store these. If you remember back to an earlier lecture we talked about dictionaries. So let's store them that way.

```
dictionary = {}
dictionary["Few Whiskey"] = Drink("Whiskey",4.00)
dictionary["Few Whiskey"].authorName("Steven")
dictionary["Espresso"] = Drink ("Coffee",1.25)
dictionary["Espresso"].describe("The think that keeps Becca alive")
```

Now we have these drinks stored in a dictionary. We can access them like "dictionary["Espresso"].description".