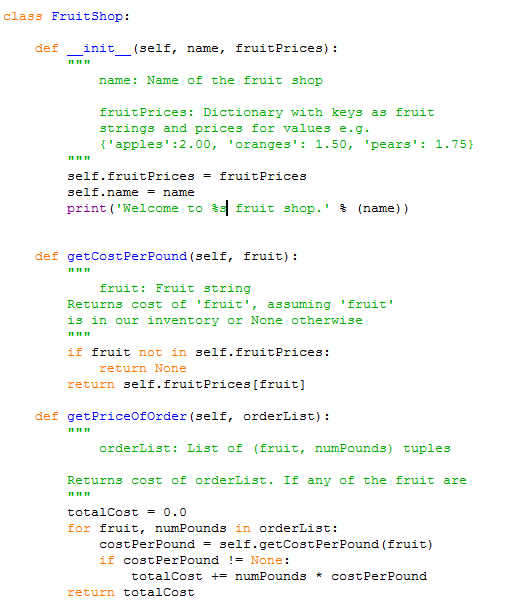
## Lab 02: Object and Classes.

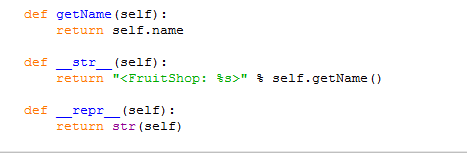
#### Object Basics

Although this isn't a class in object-oriented programming, you'll have to use some objects in the programming projects, and so it's worth covering the basics of objects in Python. An object encapsulates data and provides functions for interacting with that data.

#### Defining Classes

Here's an example of defining a class named FruitShop:



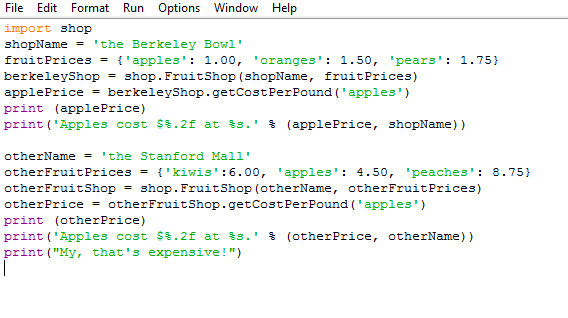


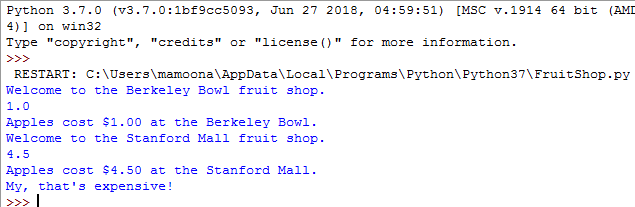
The FruitShop class has some data, the name of the shop and the prices per pound of some fruit, and it provides functions, or methods, on this data. What advantage is there to wrapping this data in a class?

* Please try to answer it

#### Using Objects

So how do we make an object and use it? Make sure you have the FruitShop implementation in shop.py. We then import the code from this file (making it accessible to other scripts) using import shop, since shop.py is the name of the file. Then, we can create FruitShop objects as follows:



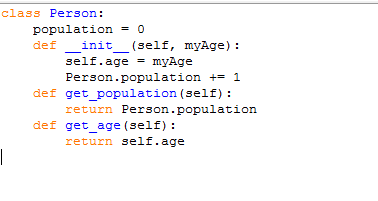


So what just happended? The import shop statement told Python to load all of the functions and classes in shop.py. The line berkeleyShop = shop.FruitShop(shopName, fruitPrices) constructs an instance of the FruitShop class defined in shop.py, by calling the \_\_init\_\_ function in that class. Note that we only passed two arguments in, while \_\_init\_\_ seems to take three arguments: (self, name, fruitPrices). The reason for this is that all methods in a class have self as the first argument. The self variable's value is automatically set to the object itself; when calling a method, you only supply the remaining arguments. The self variable contains all the data (name and fruitPrices) for the current specific instance (similar to this in Java). The print statements use the substitution operator.

#### Static vs Instance Variables

The following example illustrates how to use static and instance variables in Python.

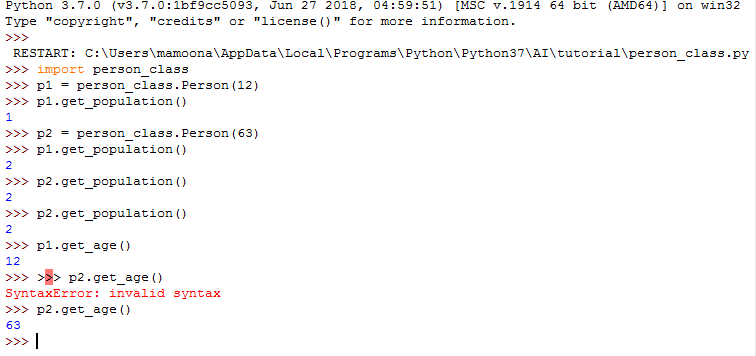
Create the person\_class.py containing the following code:



We first compile the script:

[cs188-ta@nova ~]$ python person\_class.py

Now use the class as follows:

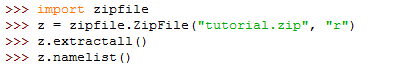


In the code above, age is an instance variable and population is a static variable. population is shared by all instances of the Person class whereas each instance has its own age variable.

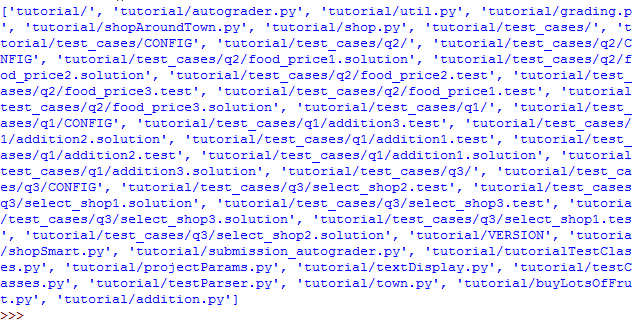
1. Autograding

Before the submission, you can check your code through autograder. Every tasks includes its autograder for you to run yourself. This is the recommended, and fastest, way to test your code, but keep in mind you need to submit the code to GA.

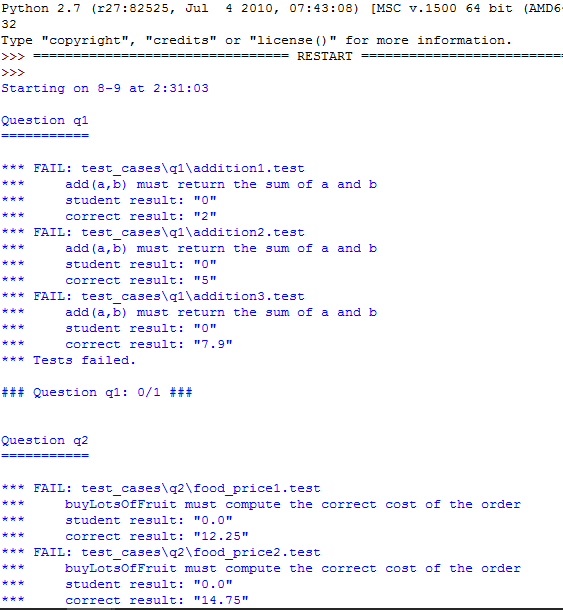
You can download all of the files associated the autograder tutorial as a zip archive: [tutorial.zip](https://s3-us-west-2.amazonaws.com/cs188websitecontent/projects/release/tutorial/v1/001/tutorial.zip) . Unzip this file and examine its contents in python Shell:

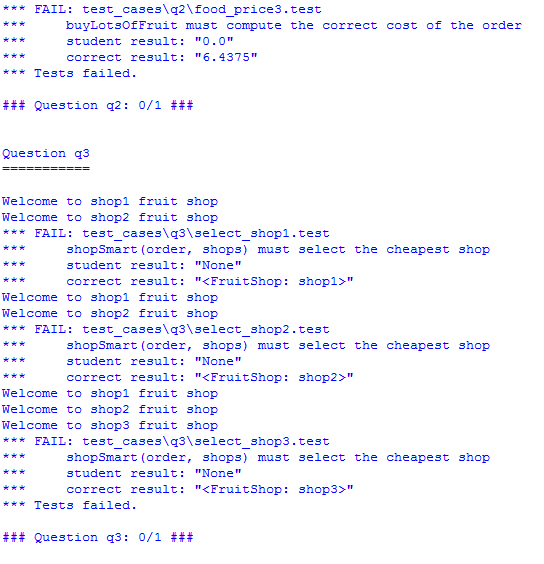


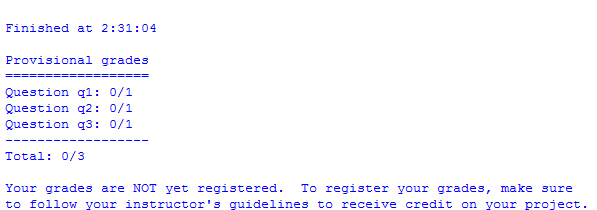
Result:

The command python autograder.py grades your solution to all three problems. If we run it before editing any files we get a page or two of output:

[Lab]$ python autograder.py







For each of the three questions, this shows the results of that question's tests, the questions grade, and a final summary at the end. Because you haven't yet solved the questions, all the tests fail. As you solve each question you may find some tests pass while other fail. When all tests pass for a question, you get full marks.

Looking at the results for question 1, you can see that it has failed three tests with the error message "add(a,b) must return the sum of a and b".

1. Working of AutoGrade