CSCI 3202 Introduction to Artificial Intelligence
Instructor: Hoenigman
Assignment 1
Due Wednesday, September 3, before 3pm

Python Refresher

The purpose of this assignment is to re-familiarize your self with Python. If you are new to Python, before starting this assignment, you may want to also walk through the Python tutorial found here:

http://inst.cs.berkeley.edu/~cs188/fa11/projects/tutorial/tutorial.html

All files needed for this assignment are included in the `tutorial.zip` file on Moodle.

This assignment is reproduced from the Python Refresher assignment in the CS188x Berkeley edX Artificial Intelligence course.

The `tutorial.zip` file should contain the following files:

- `addition.py:` source file for question 1
- `buyLotsOfFruit.py`: source file for question 2
- `shop.py`: source file for question 3
- `shopSmart.py`: source file for question 3
- `autograder.py`: autograding script

Other files, which you can ignore:

- `test_cases`: directory contains the test cases for each question
- `grading.py`: autograder code
- `testClasses.py`: autograder code
- `tutorialTestClasses.py`: test classes for this particular project
- `projectParams.py`: project parameters

The command:

```
python autograder.py
```

grades your solution to all three problems in this assignment. If you run it before editing any files, you should get a page or two of output:

```
Question q1
===========
*** FAIL: test_cases/q1/addition1.test
***   add(a,b) must return the sum of a and b
```

```
***  student result: "0"
***  correct result: "2"
*** FAIL: test_cases/q1/addition2.test
***  add(a,b) must return the sum of a and b
***  student result: "0"
***  correct result: "5"
*** FAIL: test_cases/q1/addition3.test
***  add(a,b) must return the sum of a and b
***  student result: "0"
***  correct result: "7.9"
*** Tests failed.

### Question q1: 0/1 ###


Question q2
===========
*** FAIL: test_cases/q2/food_price1.test
***  buyLotsOfFruit must compute the correct cost of the order
***  student result: "0.0"
***  correct result: "12.25"
*** FAIL: test_cases/q2/food_price2.test
***  buyLotsOfFruit must compute the correct cost of the order
***  student result: "0.0"
***  correct result: "14.75"
*** FAIL: test_cases/q2/food_price3.test
***  buyLotsOfFruit must compute the correct cost of the order
***  student result: "0.0"
***  correct result: "6.4375"
*** Tests failed.

### Question q2: 0/1 ###


Question q3
===========
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** FAIL: test_cases/q3/select_shop1.test
***  shopSmart(order, shops) must select the cheapest shop
***  student result: "None"
***  correct result: "<FruitShop: shop1>"
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** FAIL: test_cases/q3/select_shop2.test
***  shopSmart(order, shops) must select the cheapest shop
***  student result: "None"
***  correct result: "<FruitShop: shop2>"
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
*** FAIL: test_cases/q3/select_shop3.test
```

```
***   shopSmart(order, shops) must select the cheapest shop
***   student result: "None"
***   correct result: "<FruitShop: shop3>"
*** Tests failed.

### Question q3: 0/1 ###


Finished at 23:39:51

Provisional grades
==================
Question q1: 0/1
Question q2: 0/1
Question q3: 0/1
------------------
Total: 0/3
```

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 points.

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". The answer your code gives is always 0, but the correct answer is different.

Question 1: Addition
Open `addition.py` and look at the definition of add:

```
def add(a, b):
    "Return the sum of a and b"
    "*** YOUR CODE HERE ***"
    return 0
```

Modify add to return the sum of a and b, and re-run the autograder. You should now see a passing result for Question 1.

Question 2: buyLotsOfFruit function

Add a `buyLotsOfFruit(orderList)` function to `buyLotsOfFruit.py` which takes a list of (fruit,pound) tuples and returns the cost of your list. If there is some fruit in the list that doesn't appear in `fruitPrices` it should print an error message and return `None`. Please do not change the `fruitPrices` variable.

Run python autograder.py until question 2 passes all tests. Each test will confirm that `buyLotsOfFruit(orderList)` returns the correct answer given various possible inputs. For example, test_cases/q2/food_price1.test tests whether:

```
Cost of [('apples', 2.0), ('pears', 3.0), ('limes', 4.0)] is
12.25
```

Question 3: shopSmart function

Fill in the function `shopSmart(orders,shops)` in `shopSmart.py`, which takes an `orderList` (like the kind passed in to `FruitShop.getPriceOfOrder`) and a list of `FruitShop` and returns the `FruitShop` where your order costs the least amount in total. Don't change the file name or variable names, please.

Run `python autograder.py` until Question 3 passes all tests. Each test will confirm that `shopSmart(orders,shops)` returns the correct answer given various possible inputs. For example, with the following variable definitions:

```
orders1 = [('apples',1.0), ('oranges',3.0)]
orders2 = [('apples',3.0)]
dir1 = {'apples': 2.0, 'oranges':1.0}
shop1 =  shop.FruitShop('shop1',dir1)
dir2 = {'apples': 1.0, 'oranges': 5.0}
shop2 = shop.FruitShop('shop2',dir2)
shops = [shop1, shop2]
```

test_cases/q3/select_shop1.test tests whether:

```
shopSmart.shopSmart(orders1, shops) == shop1
```

and test_cases/q3/select_shop2.test tests whether:

```
shopSmart.shopSmart(orders2, shops) == shop2
```

To Submit Your Assignment

Submit all code files as one zip file on Moodle. Include all files in the original tutorial directory, including both the files you modified and the files you did not modify. Store the output of the autograder in a file using the command

```
python autograder.py > myOutput
```

and include the output in what you submit to Moodle. Print out only the files you modified, and the output of the autograder. Staple these pages together, and turn them in at the beginning of class on Wednesday, September 3. Your grade on this assignment will be the grade that the autograder returns, so it's important to get your code working.