PH #0: Python/Autograder Tutorial

Introduction

The projects for this class assume you use Python 2.7. **PH #0** will cover the following:

- A mini-Python tutorial by some toy implementation
- Project grading: Every project's release includes its autograder for you to run yourself.

Files to Edit and Submit: You will fill in portions of *addition.py*, *buyLotsOfFruit.py*, and *shopSmart.py* in **PH_0_PythonAutograder.zip** during the assignment. You should submit these files with your code and comments. Please do not change the other files in this distribution or submit any of our original files other than these files.

Evaluation: Your code will be autograded for technical correctness. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder. However, the correctness of your implementation -- not the autograder's judgements -- will be the final judge of your score. If necessary, we will review and grade assignments individually to ensure that you receive due credit for your work.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. If you copy someone else's code and submit it with minor changes, we will know. These cheat detectors are quite hard to fool, so please don't try. We trust you all to submit your own work only; please don't let us down. If you do, we will pursue the strongest consequences available to us.

Getting Help: You are not alone! If you find yourself stuck on something, contact the course staff for help. Office hours, section, and the discussion forum are there for your support; please use them. If you can't make our office hours, let us know and we will schedule more. We want these projects to be rewarding and instructional, not frustrating and demoralizing. But, we don't know when or how to help unless you ask.

Autograding

To get you familiarized with the autograder, we will ask you to code, test, and submit solutions for three questions. You can download all of the files associated the autograder tutorial as a zip archive: **PH_0_PythonAutograder.zip**. Unzip this file and examine its contents:

addition.py
autograder.py
buyLotsOfFruit.py
grading.py
projectParams.py
shop.py
shopSmart.py
testClasses.py
testParser.py
test_cases
tutorialTestClasses.py

This contains a number of files you'll edit or run:

- 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 (see below)

and others 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. If we run it before editing any files we get a page or two of output:

```
>> python autograder.py
Starting on 1-21 at 23:39:51

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 marks.

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
```

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 which 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 and you get full marks. 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. Note that we will provide the shop.py implementation as a "support" file, so you don't need to submit yours.

Run python autograder.py until question 3 passes all tests and you get full marks. 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
```