

Assignment 0 (Ungraded): Python/Autograder Tutorial

Acknowledgement

This assignment reuses the material from UC Berkeley CS188 Intro to AI (<http://ai.berkeley.edu/>).

Introduction

The purpose of this assignment is to help you to get familiar with Python. Therefore, this assignment is unmarked.

The assignment for this class assume you use Python 2.7.

Assignment 0 will cover the following:

- A mini-UNIX tutorial (particularly important if you work on instructional machines),
- A mini-Python tutorial,
- 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 [tutorial.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.

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
```

The tests called this with `a` and `b` set to different values, but the code always returned zero. Modify this definition to read:

```
def add(a, b):  
  
    "Return the sum of a and b"  
  
    print "Passed a=%s and b=%s, returning a+b=%s" % (a,b,a+b)  
  
    return a+b
```

Now rerun the autograder (omitting the results for questions 2 and 3):

```
[cs188-ta@nova ~/tutorial]$ python autograder.py -q q1
```

Starting on 1-21 at 23:52:05

Question q1

=====

Passed a=1 and b=1, returning a+b=2

*** PASS: test_cases/q1/addition1.test

*** add(a,b) returns the sum of a and b

Passed a=2 and b=3, returning a+b=5

*** PASS: test_cases/q1/addition2.test

*** add(a,b) returns the sum of a and b

Passed a=10 and b=-2.1, returning a+b=7.9

*** PASS: test_cases/q1/addition3.test

*** add(a,b) returns the sum of a and b

Question q1: 1/1

Finished at 23:41:01

Provisional grades

=====

Question q1: 1/1

Question q2: 0/1

Question q3: 0/1

Total: 1/3

You now pass all tests, getting full marks for question 1. Notice the new lines "Passed a=..." which appear before "*** PASS: ...". These are produced by the print statement in `add`. You can use print statements like that to output information useful for debugging. You can also run the autograder with the option `--mute` to temporarily hide such lines, as follows:

```
[cs188-ta@nova ~/tutorial]$ python autograder.py -q q1 --mute
```

```
Starting on 1-22 at 14:15:33
```

```
Question q1
```

```
=====
```

```
*** PASS: test_cases/q1/addition1.test
```

```
***      add(a,b) returns the sum of a and b
```

```
*** PASS: test_cases/q1/addition2.test
```

```
***      add(a,b) returns the sum of a and b
```

```
*** PASS: test_cases/q1/addition3.test
```

```
***      add(a,b) returns the sum of a and b
```

```
### Question q1: 1/1 ###
```

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 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
```