Project0 实验报告

Unix/Python/Autograder Tutorial

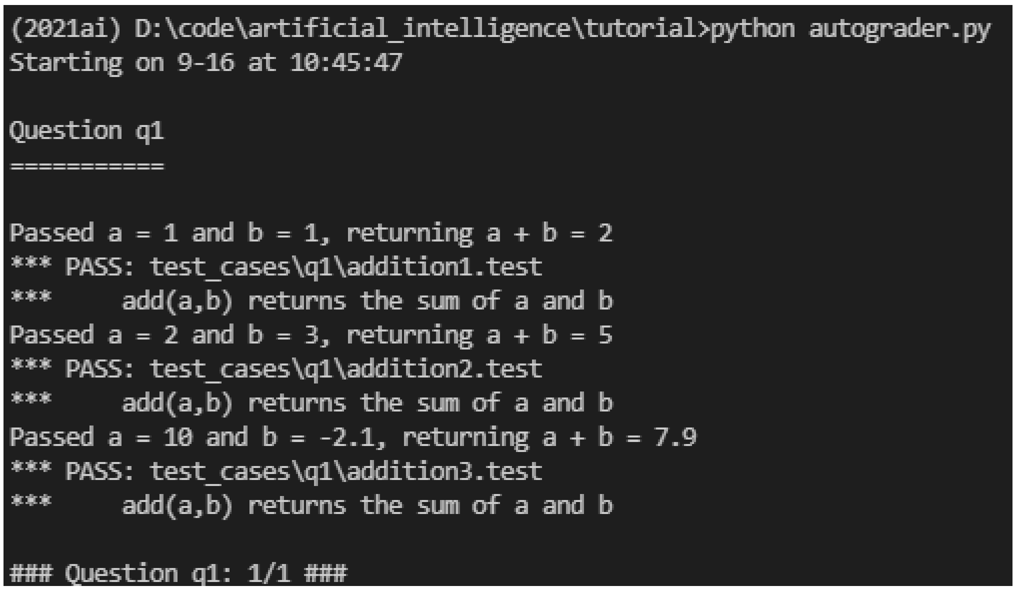
github链接：EssieYiu/artificial_intelligence: 2021 AI Homework (github.com)

Question 1

本题要求实现一个输入为a和b，返回a+b的结果的函数，直接在return a+b即可。另外，根据tutorial的指引，在函数中进行了一行格式化输出，这行输出能在autograder运行时显示，因此可以用于中间结果的debug。具体代码如下：

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

执行结果如下



```
(2021ai) D:\code\artificial_intelligence\tutorial>python autograder.py
Starting on 9-16 at 10:45:47

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 ###
```

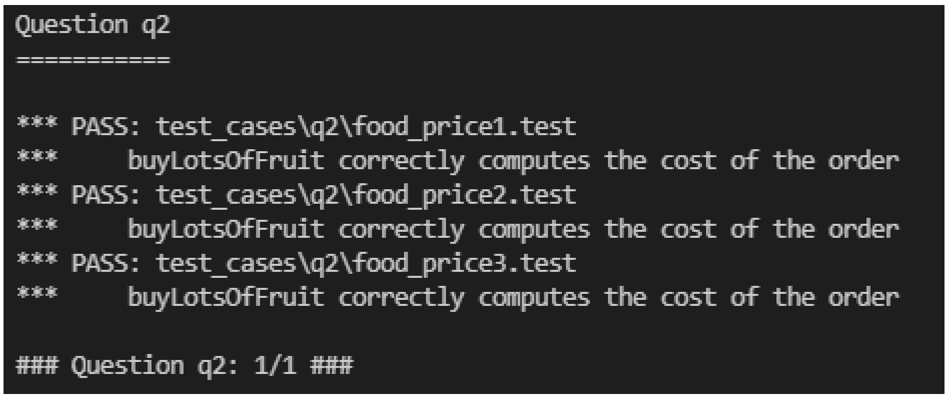
Question 2

本题要求实现一个函数，其输入是一个由水果名字和购买磅数构成的tuple组成的list，输出为总花费。通过for循环遍历这个list里面的每一个（fruit, numPounds)对，再从一个全局定义了的字典中读取每种水果一磅的费用，将购买磅数乘以每一磅费用加到总花费上，当for循环结束的时候，便能够得到购买list中所有水果的总费用。代码如下：

```python
def buyLotsOfFruit(orderList):
    """
        orderList: List of (fruit, numPounds) tuples

    Returns cost of order
    """
    totalCost = 0.0
    "*** YOUR CODE HERE ***"
    for fruit, numPounds in orderList:
        totalCost += numPounds * fruitPrices[fruit]
    return totalCost
```

执行结果如下



```
Question q2
===========

*** PASS: test_cases\q2\food_price1.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases\q2\food_price2.test
***     buyLotsOfFruit correctly computes the cost of the order
*** PASS: test_cases\q2\food_price3.test
***     buyLotsOfFruit correctly computes the cost of the order

### Question q2: 1/1 ###
```

Question 3

本题要求实现一个选择购买水果花费最少的商铺，函数输入为一个由（水果，购买磅数）的tuple组成的list以及一个由Fruitshop类的实例构成的list，要求输出购买花费为最少的店铺。

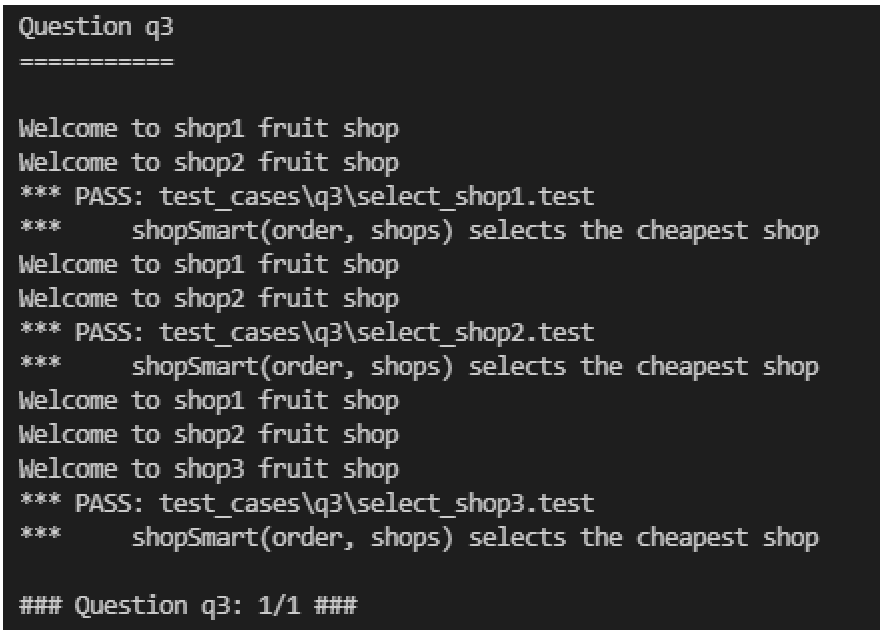
因此，定义一个变量min_cost记录目前购买水果最少的花费，初始化为一个足够大的数，每计算完在一个店铺购买指定水果所需费用之后，和min_cost进行比较，若小于min_cost，则将目前最便宜的商铺记录下来，并更新min_cost。

对于每一个店铺，需要再用一层for循环，遍历（水果名称，购买磅数）的list，用一个变量current_shop_total_cost记录在这家店铺购买水果花费的总额。遍历list的时候调用Fruitshop类的getCostPerPound方法获取到这家店铺某种水果的每磅价格，再乘以购买磅数，加到在这家店的花费总额上。

最后，当两层for循环结束的时候，cheapest_shop变量中记录的就是最便宜的商铺，注意到，Fruitshop类中定义了`__str__(self)`方法，对Fruitshop实例使用str方法，得到要求的结果。

```python
def shopSmart(orderList, fruitShops):
    """
        orderList: List of (fruit, numPound) tuples
        fruitShops: List of FruitShops
    """
    "*** YOUR CODE HERE ***"
    min_cost = 9999999
    for shop in fruitShops:
        current_shop_total_cost = 0
        for fruit, numPound in orderList:
            current_shop_total_cost += shop.getCostPerPound(fruit) * numPound
        if current_shop_total_cost < min_cost:
            cheapest_shop = str(shop)
            min_cost = current_shop_total_cost
    return cheapest_shop
```

执行结果如下



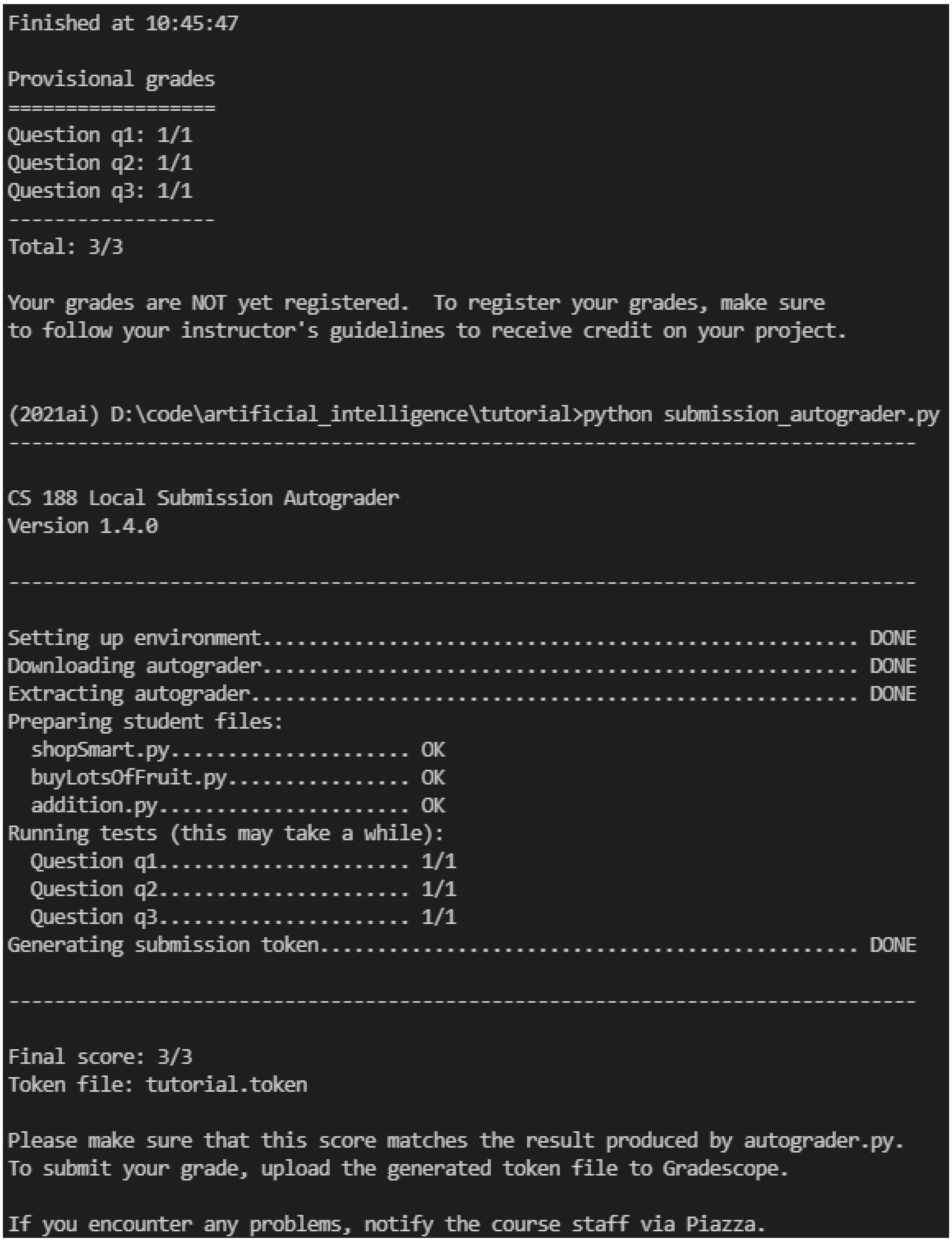
```
Question q3
===========

Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases\q3\select_shop1.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
*** PASS: test_cases\q3\select_shop2.test
***     shopSmart(order, shops) selects the cheapest shop
Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
Welcome to shop3 fruit shop
*** PASS: test_cases\q3\select_shop3.test
***     shopSmart(order, shops) selects the cheapest shop

### Question q3: 1/1 ###
```

附录

所有完成情况



```
Finished at 10:45:47

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

Your grades are NOT yet registered.  To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.


(2021ai) D:\code\artificial_intelligence\tutorial>python submission_autograder.py
--------------------------------------------------------------------------------

CS 188 Local Submission Autograder
Version 1.4.0


--------------------------------------------------------------------------------

Setting up environment................................................... DONE
Downloading autograder................................................... DONE
Extracting autograder.................................................... DONE
Preparing student files:
  shopSmart.py..................... OK
  buyLotsOfFruit.py................ OK
  addition.py...................... OK
Running tests (this may take a while):
  Question q1...................... 1/1
  Question q2...................... 1/1
  Question q3...................... 1/1
Generating submission token.............................................. DONE


--------------------------------------------------------------------------------

Final score: 3/3
Token file: tutorial.token

Please make sure that this score matches the result produced by autograder.py.
To submit your grade, upload the generated token file to Gradescope.

If you encounter any problems, notify the course staff via Piazza.
```

完整代码文件

addition.py

```
# addition.py
# -----------
# Licensing Information:  You are free to use or extend these projects for
# educational purposes provided that (1) you do not distribute or publish
# solutions, (2) you retain this notice, and (3) you provide clear
# attribution to UC Berkeley, including a link to http://ai.berkeley.edu.
#
# Attribution Information: The Pacman AI projects were developed at UC Berkeley.
# The core projects and autograders were primarily created by John DeNero
# (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# Student side autograding was added by Brad Miller, Nick Hay, and
# Pieter Abbeel (pabbeel@cs.berkeley.edu).


"""
Run python autograder.py
"""


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

buyLotsOfFruit.py

```
# buyLotsOfFruit.py
# -----------------
# Licensing Information:  You are free to use or extend these projects for
# educational purposes provided that (1) you do not distribute or publish
# solutions, (2) you retain this notice, and (3) you provide clear
# attribution to UC Berkeley, including a link to http://ai.berkeley.edu.
#
# Attribution Information: The Pacman AI projects were developed at UC Berkeley.
# The core projects and autograders were primarily created by John DeNero
# (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# Student side autograding was added by Brad Miller, Nick Hay, and
# Pieter Abbeel (pabbeel@cs.berkeley.edu).


"""
To run this script, type

  python buyLotsOfFruit.py

Once you have correctly implemented the buyLotsOfFruit function,
the script should produce the output:

Cost of [('apples', 2.0), ('pears', 3.0), ('limes', 4.0)] is 12.25
"""
from __future__ import print_function

fruitPrices = {'apples': 2.00, 'oranges': 1.50, 'pears': 1.75,
               'limes': 0.75, 'strawberries': 1.00}


def buyLotsOfFruit(orderList):
    """
        orderList: List of (fruit, numPounds) tuples

    Returns cost of order
    """
    totalCost = 0.0
    "*** YOUR CODE HERE ***"
    for fruit, numPounds in orderList:
        totalCost += numPounds * fruitPrices[fruit]
```

```python
    return totalCost


# Main Method
if __name__ == '__main__':
    "This code runs when you invoke the script from the command line"
    orderList = [('apples', 2.0), ('pears', 3.0), ('limes', 4.0)]
    print('Cost of', orderList, 'is', buyLotsOfFruit(orderList))
```

```python
# shopSmart.py
# ------------
# Licensing Information:  You are free to use or extend these projects for
# educational purposes provided that (1) you do not distribute or publish
# solutions, (2) you retain this notice, and (3) you provide clear
# attribution to UC Berkeley, including a link to http://ai.berkeley.edu.
#
# Attribution Information: The Pacman AI projects were developed at UC Berkeley.
# The core projects and autograders were primarily created by John DeNero
# (denero@cs.berkeley.edu) and Dan Klein (klein@cs.berkeley.edu).
# Student side autograding was added by Brad Miller, Nick Hay, and
# Pieter Abbeel (pabbeel@cs.berkeley.edu).


"""
Here's the intended output of this script, once you fill it in:

Welcome to shop1 fruit shop
Welcome to shop2 fruit shop
For orders:  [('apples', 1.0), ('oranges', 3.0)] best shop is shop1
For orders:  [('apples', 3.0)] best shop is shop2
"""
from __future__ import print_function
import shop


def shopSmart(orderList, fruitShops):
    """
        orderList: List of (fruit, numPound) tuples
        fruitShops: List of FruitShops
    """
    "*** YOUR CODE HERE ***"
    min_cost = 9999999
    for shop in fruitShops:
        current_shop_total_cost = 0
        for fruit, numPound in orderList:
            current_shop_total_cost += shop.getCostPerPound(fruit) * numPound
        if current_shop_total_cost < min_cost:
            cheapest_shop = str(shop)
            min_cost = current_shop_total_cost
    return cheapest_shop


if __name__ == '__main__':
    "This code runs when you invoke the script from the command line"
    orders = [('apples', 1.0), ('oranges', 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]
    print("For orders ", orders, ", the best shop is", shopSmart(orders,
shops).getName())
    orders = [('apples', 3.0)]
    print("For orders: ", orders, ", the best shop is", shopSmart(orders,
shops).getName())
```