Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

Laboratory Report

CSE 4618: Artificial Intelligence Lab

Name	**: Mirza Mohammad Azwad**
Student ID	**: 200042121**
Section	**: BSc in SWE**
Semester	**: 6th Semester**
Academic Year	**: 2022-23**
Date	**: 22/01/2024**
Lab No	**: 0**

Contents

Introduction

In this lab we essentially had to do some problem solving in python involving basic python programming and understanding the basics of python syntax. The 3 problems outlined were essentially simple python code snippets that we had to modify to solve these problems and find the desired solution.

The lab was evaluated initially with the autograd.py script running which essentially checks if the modified python code snippets passed the test cases outlined.

We were also introduced to creating virtual environments for python using conda so that we could run the code in the desired environment for this course which is the python 3.6 version that doesn't match my Operating System's python version.

I carried out this lab in a Virtual Machine(VirtualBox) loaded with Ubuntu 22.04LTS as basic linux was a requirements per the taskbook and the was recommended as per the lab instructions

Analysis of the Problems

There were 3 main problems:

1. **addition.py**
   This problem involved this python function that is supposed to return the result of addition by modifying the **add** function

2. **buyLotsOfFruits.py**
   This problem involved a function called *buyLotsOfFruits* and in this function we had to find the total cost calculated for all the fruits and then return the **total computed cost**. The fruits are present as fruitPrices in a dictionary with key value pairs having the fruit name as an **str** and the value or price of each pound as a **float**. *The orderList* is present as a list of tuples denoting the fruit name as **str** with the amount in pounds given as **float**

3. **shopSmart.py**
   In this problem we simply had to find the optimal shop or in other words, the shop with the least total cost. We are provided with 3 shops and we compare their total prices for the mentioned order and then print the title of the shop with the least total cost. We have the *orders* as a list of tuples like the problem above. And we are using another module shop which we imported and use the *FruitShop* instantiation.

Explanation of Solution

1. **addition.py**
   We can simply return the sum of the arguments *a* and *b* created using the '+' arithmetic operator.

```
return a+b
```

2. **buyLotsOfFruit.py**
   In this problem, I simply checked if the fruit mentioned in the *orderList* is present in the *fruitPrices* dictionary. And as the problem stated to return None if the fruit is not present, if it is present in that case we add the price to the totalCost as the product of the per pound price in *fruitPrices*, and the pounds mentioned in the *orderList* gives the cost of the fruit order mentioned in the *orderList*. We treat the dictionary as a key value pair and we can check if a fruit is present by checking using the **in** operator. And by simply passing the fruit name in place of the index we can find the per pound cost from the *fruitPrices* dictionary.

```
totalCost = 0.0
"*** YOUR CODE HERE ***"
for i in orderList:
    if i[0] in fruitPrices:
        totalCost+=i[1]*fruitPrices[i[0]]
    else:
        return None
return totalCost
```

3. **shopSmart.py**
   First we look at the *shop* module where we see the *FruitShop* class which has 2 methods that we can use that is *getPriceOfOrder* which can be used to get the name of the shop in question.

   Using *shop.py***,** in the next step we compute the total cost of the order using *getPriceOfOrder* and we compare with a very large value which we change as per the cost obtained for that shop, the idea is to update the large value *minPrice* to get the current minimum price when compared with each of the existing shops. We iterate over the list using an implicit iterator using the python for loop that iterates over the *fruitShops* list. This helps us get the price for each shop and compare with the *minPrice*. This helps us narrow down the shop and store the cheapest shop for each iteration in the *bestShop*

```
minPrice=float('inf')
bestShop=None
for i in fruitShops:
    price=i.getPriceOfOrder(orderList)
    if minPrice>price:
        minPrice=price
        bestShop=i
return bestShop
```

Problems Faced and Interesting Findings

I faced a problem in the last problem which helped me discover how to set the largest possible float value in python3 using **float('inf')** which essentially sets the largest possible value, initially I used to set the minPrice to a large value denoted by 1e12 which I observed is exceeded as the *auograd.py* clearly shows that the third task does not pass all the test cases.

```
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
*** FAIL: test_cases/q3/select_shop3.test
***     shopSmart(order, shops) must select the cheapest shop
***     student result: "None"
***     correct result: "<FruitShop: shop3>"
*** Tests failed.
```

```
minPrice=1e12
```

When I did update to setting the *minPrice* as a typecast of 'inf', this worked perfectly and passed all the test cases with the interesting finding being how to set a large value in python.