ISLAMIC UNIVERSITY OF TECHNOLOGY

Organization of Islamic Cooperation

Board Bazar, Gazipur

Lab Report 0

CSE 4712

Md. Bakhtiar Hasan

21st August, 2022

## Question 01

The problem required us to return the sum of two numbers. This can be achieved using the + (plus) operator in Python.

def add(a, b):  
 print("Passed a=%s and b=%s, returning a+b=%s" % (a,b,a+b))  
 return a+b

PYTHON

This was a simple problem meant to introduce us to Python. As such, there were no difficulties faced or any interesting findings. There are no hyperparameters to tune.

## Question 02

This problem was a bit more advanced than the previous one since it involved interacting with lists. The problem required us to loop over the values of a list of tuples (orderList), each of which represented a fruit and the quantity being bought. For each fruit, we had to retrieve the price from a separate list (fruitPrices) and find the total price for the quantity being bought.

def buyLotsOfFruit(orderList):  
 totalCost = 0.0  
 for fruit, numPounds in orderList:  
 totalCost += fruitPrices[fruit] \* numPounds  
 return totalCost

PYTHON

This section of the code explores the syntax of Python in quite a bit more depth. The for loop in particular highlights how tuples can be traversed, a syntax which is unique to Python.

On top of the above code, there was also an explicit requirement in the problem to print an error message and return None if an invalid fruit item was present in the list. The final code is provided below.

def buyLotsOfFruit(orderList):  
 totalCost = 0.0  
 for fruit, numPounds in orderList:  
 if fruit not in fruitPrices:  
 return None  
 totalCost += fruitPrices[fruit] \* numPounds  
 return totalCost

PYTHON

## Question 03

The final problem introduced us to how objects work in Python. The problem itself was somewhat similar to the previous one, except that we now had multiple shops instead of a single one. We were again given a list of tuples, each representing a fruit and the quantity of the food being bought.

Additionally, we were given a second list of FruitShop objects. FruitShop is a helpful class which was created beforehand and provides us with the method getPriceOfOrder, which takes orderList as a parameter. Internally, this method works like the buyLotsOfFruit method we created in the previous problem, except that it just skips over invalid fruits instead of throwing an error.

To solve the problem, we had to loop over the list of FruitShop objects, call the getPriceOfOrder method on each object and keep track of the shop which had the least overall cost.

def shopSmart(orderList, fruitShops):  
 if len(fruitShops) == 0:  
 return None  
   
 minCost = fruitShops[0].getPriceOfOrder(orderList)  
 bestShop = None  
   
 for shop in fruitShops:  
 cost = shop.getPriceOfOrder(orderList)  
 if cost <= minCost and cost != 0.0:  
 minCost = cost  
 bestShop = shop  
  
 return bestShop

Two additional requirements were self-imposed:

1. The length of the list of FruitShop objects was checked, and if no FruitShop objects were given, None was returned. This condition was checked at the start of the code.
2. If the cost returned by a FruitShop object was 0, then none of the required fruits were present in that shop. In this case, the shop was not considered. This condition was checked before updating the bestShop variable.

One issue with the code lies in how the getPriceOfOrder method works. Since it just skips over invalid fruit items without throwing any errors, it is possible that the best shop will not actually have all the fruits required by the order.