



Islamic University of Technology (IUT)

Lab Report - 0

CSE 4618 Artificial Intelligence Lab

Name: Mukit Mahdin
Student ID: 200042170
Program : Software Engineering
Semester: 6th
Academic Year: 2022-2023

Task - 1 :

Introduction :

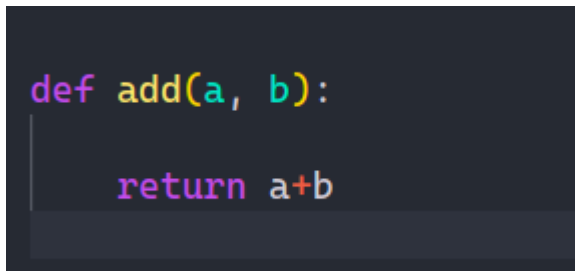
The main goal of this task is to get familiar with basic python functions by performing a basic add operation.

Problem Analysis :

In the tutorial folder, *addition.py* is given along with other necessary files. *addition.py* contains a empty function definition *def add* with two parameters a and b respectively. Add operation need to be performed between this two parameters and returned from the function.

Solution Explanation :

Solution is pretty straightforward. Just returned the sum of the two parameters in the *return* statement.



```
def add(a, b):  
    return a+b
```

And then, run the *autograder.py* to check the completeness of the task.

Challenges :

As the problem was very basic and introductory, I didn't faced any particular problem in this. But I did face some challenges in the conda environment setup and run the programs with the specific python version.

Task - 2 :

Introduction :

The goal of this specific task is to get the total amount of money a buyer needs in a shop to buy the fruits from that shop according to his need.

Problem Analysis :

In the *buyLotsofFruit.py* file, a dictionary is defined called *fruitPrices*. In that dictionary each fruit item is mapped with their respective price per pound. Fruit Items and prices are presented as key-value pair.

Also, a function definition is present in the same file which is *def buyLotsofFruit (orderList)*. The function parameter *orderList* is defined in the main function which is basically a tuple. It has 3 objects as and each object has 2 elements. The first element is the fruit *name* and the second element is the required *pound* of that specific fruit.

So I need to calculate the total according to the *orderList* tuple with the help of the *fruitPrices* dictionary inside the *buyLotsofFruit(orderList)* function and return that.

Solution Explanation :

fruitPrices dictionary has mapping to prices of the fruit from the fruit name. I used that that dictionary when iterating through the *orderList* tuple to get the prices. In the function, I declared a variable called *totalCost* initialized with 0.0. Then, I used a loop to iterate through the elements of *orderList*.

The loop goes through each *fruit* and *pound* in the provided *orderList* tuple. For every fruit, it checks if the fruit exists in the *fruitPrices* dictionary. If it exists, the cost is calculated by multiplying the *pounds* (quantity) with the *price* and adds it to the *totalCost*. If a fruit is not found in the dictionary it indicates that the price is missing, so it returns *None*. After processing all the fruits in the *orderList*, the function returns the final *totalCost*.

Challenges :

I need to go through the documentation in w3Schools to learn about the dictionary and tuple and how to iterate through them. It was a bit time consuming, other than that the task was easy to complete.

Task-3 :

Introduction :

The purpose of this specific task is to find the best (cheapest) shop among some pre-defined shop in the main function. Basically, I need to calculate the total price of a specific order in several given shops and find the cheapest one among them.

Problem Analysis :

Two files are given, *shop.py* and *shopSmart.py*. In the *shop.py* file, necessary functions like *getPriceofOrder*, *getName* are defined which I need to use in the *shopSmart.py* to calculate the total prices and get the cheapest shop. In the *shopSmart.py*, I need to implement the *shopSmart()* function which will return the cheapest shop for a specific order.

The main function has two shops in the *shops* array. And each shop's Price List is declared separately in the main function also. We need to iterate through both *orders* tuple and the *shops* array to get the cheapest shop for the specific order.

Solution Explanation :

The *shopSmart* function takes two parameters. Firstly, *orderList*, which is the required fruits and quantities. The *fruitShops*, which is basically an array of the shops.

I calculated the *total cost* of a specific order using the *getPriceOfOrder* method specific to each shop which is defined in the *shop.py*.

We keep track of the shop offering the lowest cost comparing with the previous *minCost*.

The loop starts by setting values for *minCost* and *cheapShop*. Initially, *minCost* is initialized to *infinity* and *cheapShop* is initialized to *None*. It iterates through each fruit shop in the provided list (*fruitShops*).

For each shop, it calculates the total cost of the given order using the *getPriceOfOrder* method.

If this calculated cost is lower than the current minimum cost, the function updates the minimum cost. This thing repeats for each shop and the loop considers all available options in the *shops* array.

Finally, the function returns the cheapest fruit shop (*cheapShop*) that offers the lowest total cost for the specific order

Challenges :

- Working with the *shop.py* class to get the necessary pre-defined function.
 - Check the minimum cost from each shop comparing the total price every time.
- It took some time to visualize that concept.