Islamic University of Technology

Department of Computer Science and Engineering

Artificial Intelligence Lab

Course Code- CSE 4618

Prepared By:
Nazia Krim Khan Oishee
ID: 200042137

Date: January 22, 2024

Contents

1 Introduction

The main objective of this lab was to acquire a thorough understanding of Python programming and the Autograder tool.

- Before diving into Python and Autograder, we covered basic commands for navigating in Linux and file/directory manipulation.
- We chose to use Visual Studio Code as our IDE.
- To prevent conflicts between existing Python versions in our computers and the one required for this course (Python 3.6), we opted for Conda. Conda is a convenient tool for managing various environments with distinct Python versions and dependencies. The lab covered the creation, activation, and deactivation of Conda environments.
- The subsequent section focused on the fundamentals of Python, including operators, strings, built-in data structures like lists, tuples, sets, and dictionaries.
- Finally we delved into writing Python scripts, realizing the careful use of tabs and spaces, followed by the creation and usage of functions, classes, objects of classes, and modules.

2 Problem Analysis

In the lab, we were assigned to do three tasks.

2.1 Question1: Addition

First question required us to write a simple function that would accept two numerical values and return the result of addition of these two values. Test cases were run on an initially defined function but the function would always return 0. Our task was to modify the function to return correct result.

2.2 Question2: buyLotsOfFruit function

In the next task, we were asked to create the function 'buyLotsOfFruit(orderList)'. This function would take a list of tuples representing (fruit, pound) pairs and return the total cost of the order. In case of exceptions such as encountering a fruit that is not defined in the 'fruitPrices' variable, the function would print an error message and return 'None'.

2.3 Question3: shopSmart function

Our last task was to fill in the function shopSmart(orders,shops) function which takes an order List and a list of Fruit Shop and returns the Fruit Shop where our order costs the least amount in total.

3 Solution Explanation

3.1 Solution of Question1

This was the easiest task to solve in this lab. The solution was already provided. In the initially given function, the arithmetic operation of addition was missing and the return value was set to 0. The solution only required to to perform the addition, store the value of addition in a variable and return the variable.

```
def add(a, b):
    "Return the sum of a and b"
    resultOfAddition = a + b # Missing logic in iniitially defined
                             # function

    return resultOfAddition
```

3.2 Solution of Question2

I solved the he following task using a simple loop. The function performs in the following way:

- The fruitPrices dictionary stores the prices of different fruits.
- A variable totalCost is initialized to 0 that keeps track of the total cost.
- The function iterates through each tuple (fruit, pounds) in the orderList.
- It checks if the fruit is present in fruitPrices.If so, it calculates the cost and adds it to the total.
- If a fruit is not found in fruitPrices, an error message is printed, and the function returns None.
- The final total cost is returned.

```
def buyLotsOfFruit(orderList):
    totalCost = 0.0

    for fruit, pounds in orderList:
        if fruit in fruitPrices:
            totalCost += fruitPrices[fruit] * pounds
        else:
            print(f"Error: {fruit} is not in fruitPrices. Unable to
               calculate cost.")
            return None

    return totalCost
```

The solution was easy to do and understand.

3.3 Solution of Question3

The function works the following way:

- It initializes variables cheapest_shop and min_cost to None and positive infinity, respectively.

- It then iterates through each shop in the shops list.
- For each shop, it calculates the cost of the given order using shop.getPriceOfOrder(order).
- If the cost is not None and is less than the current minimum cost, it updates the min_cost and sets cheapest_shop to the current shop.
- Finally, it returns the shop with the lowest cost for the given order.

```
def shopSmart(order, shops):
    cheapest_shop = None
    min_cost = float('inf')

    for shop in shops:
        cost = shop.getPriceOfOrder(order)
        if cost is not None and cost < min_cost:
            min_cost = cost
            cheapest_shop = shop

    return cheapest_shop
```

4 Findings and Insights

4.1 Question 1

4.1.1 Findings:

Missing Logic: The initial function lacked the logic for the addition operation, resulting in incorrect answers.

4.1.2 Insights:

Refined code: The modified function is implemented following the required arithmetic operation. It is clear and understandable in terms of signature i.e. function and parameter names, variable name.

4.2 Question2

4.2.1 Findings:

Global Declaration of fruitPrices: The variable fruitPrices is globally declared. The result of the function depends on the fruitPrices, whether the fruit is enlisted in fruitPrices or not.

4.2.2 Insights:

- **Raise an Exception:** In case of encountering a fruit that is not enlisted in fruitPrices, instead of returning None, a more specific exception could be raised. Such as

```

raise ValueError(f"Fruit {fruit} is not in fruitPrices. Unable to
    calculate cost.")
```

  This allows the calling code to catch the exception and handle it in a more refined way.

- **Default Cost for Unknown Fruits:** Instead of returning None, we could have set a default cost for fruits not found in fruitPrices and calculate the total cost with a default value.

```
1
2 default_cost = 0
3 totalCost += default_cost * pounds
```

4.3 Question3

4.3.1 Findings:

Initialization of variables: The variable min_cost and cheapest_shop is initialized with float('inf') before the loop.

4.3.2 Insights:

Initial value of min_cost: I at first initialized the value of min_cost to a larger integer, yet it failed the test cases. So instead of using any hard-coded threshold value, it is more optimal to use float ('inf').

5 Challenges

In this lab, the assigned tasks were generally straightforward. However, the primary challenges were the setup of Anaconda and the utilization of the autograder, especially since it was my first time working with these tools.

Installing Anaconda was time-consuming due to its large size, and the process was also heavily dependent on the speed of the internet connection. An interruption in the network connection led to complications.

Additionally, I encountered difficulties while executing the autograder in a different directory than the one I was actually working. Consequently, this resulted in incorrect responses from the autograder during the initial stages of assessment.