Submitted By- 200041125

Mehedi Ahamed

Introduction

The aim of this lab is to address three specific problems through the implementation of basic Python programming techniques. Goal is to achieve a perfect score of 3/3 from the autograder, demonstrating that I have correctly and effectively solved all three problems as specified.

Environment Setup

To execute the code for the assigned tasks, we need to set up a new virtual environment using the Conda Package Manager. In this instance, we will install Python version 3.6 within this environment. Virtual environments are essential for managing and isolating various sets of packages and dependencies for distinct projects. This approach helps prevent conflicts between libraries and allows for project-specific configurations. Once the virtual environment is created, ensure to activate it before running any code.

```
PS F:\Lab\2.AI> conda create --name ailab python=3.6
```

```
PS F:\Lab\2.AI> conda activate ailab
```

Problem Analysis 1

The task is to create a function that takes two arguments, adds their values together, and then returns the total.

Solution Explanation

This function takes two values as arguments and returns their sum.

```
    return a+b
```

Autograder Verdict

```
Provisional grades
==================
Question q1: 1/1
------------------
Total: 1/1
```

Findings and Insights

The time complexity is **O(1)** since the time it takes to perform addition does not increase with the size of the input.

Problem Analysis 2

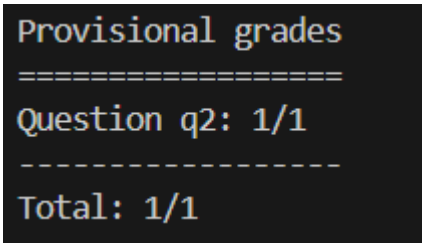
buyLotsOfFruit(orderList) function which takes a list of (fruit, numPounds) tuples and returns the cost of the list.

Solution Explanation

The buyLotsOfFruit(orderList) function computes the total cost for a fruit order based on a given list of fruits and their quantities. It uses a fruitPrices dictionary to retrieve the price of each fruit. If any fruit in the order list is not present in the fruitPrices dictionary, the function outputs a message indicating the fruit's unavailability and returns None. For fruits that are available, the function calculates the total cost by multiplying the fruit's price by its quantity and summing up the costs. The final total cost is then returned.

```
    for fruit, pound in orderList:
        if fruit in fruitPrices:
            totalCost+= pound * fruitPrices[fruit]
```

Autograder Verdict



Findings and Insights

When iterating over each (fruit,numPounds) pair in the orderList, the loop has a time complexity of O(n), where n represents the number of items in the order list. Within this loop, checking if a fruit exists in the fruitPrices dictionary involves a dictionary look-up operation, which operates in O(1) time due to its use of a hash table for quick key retrieval. Consequently, combining the loop iteration with the constant time dictionary access results in an overall time complexity of **O(n)**.

Problem Analysis 3

In the shopSmart.py file, you need to implement the function shopSmart(orders, shops). This function takes two arguments: orders, which is a list of fruit orders, and shops, which is a list of FruitShop objects. The goal of this function is to determine which FruitShop offers the lowest total cost for the given orders. The function should return the FruitShop that provides the most cost-effective price for fulfilling the entire order.

Solution Explanation

The shopSmart function determines which FruitShop offers the lowest total cost for a given list of fruit orders. It starts by assuming the first shop in the list is the cheapest and calculates the cost of fulfilling the order at this shop. It then iterates through the remaining shops, calculating the cost of the same order at each shop. If a shop offers a lower price than the current minimum, it updates the minimum cost and sets this shop as the new cheapest. Finally, the function returns the shop that provides the most cost-effective price for the order.

```
    if (m_cost> cost):
        m_cost = cost
        Shopname = i
```

Autograder Verdict

```
Provisional grades
==================
Question q3: 1/1
------------------
Total: 1/1
```

Findings and Insights

The shopSmart function has a time complexity of **O(n*m)**, where n is the number of shops and m is the number of items in the orderList. This is because the function iterates through each shop (O(n)) and calculates the cost of the order for each shop, which involves iterating through the orderList (O(m)). Therefore, the overall complexity is the product of the two, reflecting the time needed to compute the cost for each shop in the list.

Challenges Faced

Installing Anaconda and setting up the virtual environment for the first time was little bit challenging.

Key Findings

The tasks assessed the learner's fundamental knowledge and abilities in implementing Python code.

All the Codes are available in the below links-

1. **Task 1**
2. **Task 2**
3. **Task 3**