Assignment-I
Python Programming (ITO- 804)

Submitted By

Bushra Fayaz
191203072
CSE/8th

Submitted To

Saurabh Sharma
Assistant Professor (CSE Department)

Department of Computer Science and Engineering
Model Institute of Engineering and Technology
(Autonomous)
Jammu, India

Index

S.no	Question	Marks
1	Write a Python program that takes a list of daily stock prices as input, and returns the best days to buy and sell stocks in order to maximize profit. The list contains the stock prices for each day, starting from the first day. For example, the list (100, 180, 260, 310, 40, 535, 695) represents the stock prices for 7 days, where the price on the first day is 100, the second day is 180, and so on. The program should find the best days to buy and sell stocks such that the profit obtained is maximum. For instance, in the given list of stock prices, the best days to buy and sell stocks would be: Buy stock on the 1st day (price=100) Sell stock on the 4th day (price=310) Buy stock on the 5th day (price=40) Sell stock on the 7th day (price=695) The program should output these buy and sell days as a tuple or list of integers.	2.5
2	You are given a list of book titles and their corresponding publication years. Your task is to find the earliest year in which a trilogy of books was published. A trilogy is defined as a series of three books published in consecutive years. For example, consider the following list of book titles and publication years: titles = ['The Hunger Games', 'Catching Fire', 'Mockingjay', 'The Lord of the Rings', 'The Two Towers', 'The Return of the King', 'Divergent', 'Insurgent', 'Allegiant'] years = [2008, 2009, 2010, 1954, 1955, 1956, 2011, 2012, 2013] The earliest year in which a trilogy was published is 1954. Write a Python function earliest_trilogy_year(titles: List[str], years: List[int]) -> Optional[int] that takes two lists as input: titles containing the titles of the books, and years containing their corresponding publication years. The function should return the earliest year in which a trilogy of books was published, or None if no such trilogy exists. Examples: titles = ['Book1', 'Book2', 'Book3', 'Book4', 'Book5', 'Book6'] years = [2019, 2021, 2012, 2013, 2016, 2017] print(earliest_trilogy_year(titles, years)) The earliest year in which a trilogy was published is : None A trilogy is defined as a series of three books published in consecutive years. Note: • You can assume that the input lists are non-empty and contain an equal number of elements. • If multiple trilogies exist with the same earliest year, return that year.	2.5
3	Write a Python program that reads in a CSV file of stock prices (e.g. ticker symbol, date, price), and then uses dictionaries and lists to calculate the highest and lowest prices for each stock from following table.	2.5
4	A) Write a Python program to remove duplicates from a list of lists. Sample list: [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]] B) Write a Python program which takes a list and returns a list with the elements "shifted left by one position" so [1, 2, 3] yields [2, 3, 1]. Example: [1, 2, 3] → [2, 3, 1] [11, 12, 13] → [12, 13, 11] C) Iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element. Original list [11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89] And expected output is: Printing count of each item {11: 3, 45: 3, 8: 1, 23: 2, 89: 2}	2.5

Q1. Write a Python program that takes a list of daily stock prices as input, and returns the best days to buy and sell stocks in order to maximize profit.
The list contains the stock prices for each day, starting from the first day. For example, the list (100, 180, 260, 310, 40, 535, 695) represents the stock prices for 7 days, where the price on the first day is 100, the second day is 180, and so on. The program should find the best days to buy and sell stocks such that the profit obtained is maximum. For instance, in the given list of stock prices, the best days to buy and sell stocks would be: Buy stock on the 1st day (price=100) Sell stock on the 4th day (price=310) Buy stock on the 5th day (price=40) Sell stock on the 7th day (price=695) The program should output these buy and sell days as a tuple or list of integers.

Program:

```
def find_best_days(prices):
    min_price = float('inf')
    max_profit = 0
    buy_day = 0
    sell_day = 0

    for i in range(len(prices)):
        if prices[i] < min_price:
            min_price = prices[i]
            buy_day = i

        profit = prices[i] - min_price
        if profit > max_profit:
            max_profit = profit
            sell_day = i

    return (buy_day+1, sell_day+1)
prices = [100, 180, 260, 310, 40, 535,
695]
print(f"Prices on Each Date:")

for i in range(0,len(prices)):
    print("Price of Day",i+1,"is",prices[i])
best_days = find_best_days(prices)
print(f"Buy on day {best_days[0]} and sell on day {best_days[1]}")
```

```
def find_best_days(prices):
    min_price = float('inf')
    max_profit = 0
    buy_day = 0
    sell_day = 0

    for i in range(len(prices)):
        if prices[i] < min_price:
            min_price = prices[i]
            buy_day = i

        profit = prices[i] - min_price
        if profit > max_profit:
            max_profit = profit
            sell_day = i

    return (buy_day+1, sell_day+1)
prices = [100, 180, 260, 310, 40, 535, 695]
print(f"Prices on Each Date:")
for i in range(0,len(prices)):
    print("Price of Day",i+1,"is",prices[i])
best_days = find_best_days(prices)
print(f"Buy on day {best_days[0]} and sell on day {best_days[1]}")
```

```
Prices on Each Date:
Price of Day 1 is 100
Price of Day 2 is 180
Price of Day 3 is 260
Price of Day 4 is 310
Price of Day 5 is 40
Price of Day 6 is 535
Price of Day 7 is 695
Buy on day 5 and sell on day 7
```

Q2. You are given a list of book titles and their corresponding publication years. Your task is to find the earliest year in which a trilogy of books was published. A trilogy is defined as a series of three books published in consecutive years. For example, consider the following list of book titles and publication years:
titles = ['The Hunger Games', 'Catching Fire', 'Mockingjay', 'The Lord of the Rings', 'The Two Towers', 'The Return of the King', 'Divergent', 'Insurgent', 'Allegiant']
years = [2008, 2009, 2010, 1954, 1955, 1956, 2011, 2012, 2013] The earliest year in which a trilogy was published is 1954.

Write a Python function earliest_trilogy_year(titles: List[str], years: List[int]) -> Optional[int] that takes two lists as input: titles containing the titles of the books, and years containing their corresponding publication years. The function should return the earliest year in which a trilogy of books was published, or None if no such trilogy exists. Examples:
titles = ['Book1', 'Book2', 'Book3', 'Book4', 'Book5', 'Book6'] years = [2019, 2021, 2012, 2013, 2016, 2017]
print(earliest_trilogy_year(titles, years))
The earliest year in which a trilogy was published is : None
A trilogy is defined as a series of three books published in consecutive years. Note: • You can assume that the input lists are non-empty and contain an equal number of elements. • If multiple trilogies exist with the same earliest year, return that year.

Program:

```
def find_trilogy_year(books):
    years = sorted(set(books.values()))

    for i in range(len(years)-2):
        if (years[i+2] - years[i+1] == years[i+1] - years[i]) and \
        any(years[i+1] == year for book, year in books.items() if years[i] <= year <= years[i+2]):
            return years[i]

    return None

books = {'The Hunger Games': 2008, 'Catching Fire': 2009, 'Mockingjay': 2010, 'The Lord of the Rings': 1954, \
                'The Return of the King': 1956, 'Divergent': 2011, 'Insurgent':2012,'Allegiant':2013, 'The Two Towers':1955}
trilogy_year = find_trilogy_year(books)
print(f"The earliest year in which a trilogy was published is {trilogy_year}")
```

```
def find_trilogy_year(books):
    years = sorted(set(books.values()))

    for i in range(len(years)-2):
        if (years[i+2] - years[i+1] == years[i+1] - years[i]) and \
        any(years[i+1] == year for book, year in books.items() if years[i] <= year <= years[i+2]):
            return years[i]

    return None

books = {'The Hunger Games': 2008, 'Catching Fire': 2009, 'Mockingjay': 2010, 'The Lord of the Rings': 1954, \
                'The Return of the King': 1956, 'Divergent': 2011, 'Insurgent':2012,'Allegiant':2013,  'The Two Towers':1955}
trilogy_year = find_trilogy_year(books)
print(f"The earliest year in which a trilogy was published is {trilogy_year}")
```

```
The earliest year in which a trilogy was published is 1954
```

Q3. Write a Python program that reads in a CSV file of stock prices (e.g. ticker symbol, date, price), and then uses dictionaries and lists to calculate the highest and lowest prices for each stock from following table:

Symbol	Date	Price
AAPL	2022-01-01	135.90
AAPL	2022-01-02	138.45
AAPL	2022-01-03	142.20
GOOG	2022-01-01	2105.75
GOOG	2022-01-02	2098.00
GOOG	2022-01-03	2125.50
MSFT	2022-01-01	345.20
MSFT	2022-01-02	344.70
MSFT	2022-01-03	342.10

Program:

```
import csv
with open('3_csv.csv') as file:
        reader = csv.reader(file)
         next(reader)  # Skip the header row
         prices = {}       # Create an empty dictionary to store the prices for each stock
        for row in reader:
    # Extract the symbol, date, and price from the row
                ticker, date, price = row
    # Convert the price from a string to a float
                price = float(price)
    # Check if the ticker symbol is already in the dictionary
                 if ticker in prices:
                        prices[ticker].append(price)
                else:

prices[ticker] = [price]
for ticker, price_list in
prices.items():
    highest_price =
max(price_list)
   lowest_price =
min(price_list)
    print(f"{ticker}: Highest
Price = ${highest_price:.2f},
Lowest Price =
${lowest_price:.2f}")
```

```
import csv

# Open the CSV file and read in the data
with open('3_csv.csv') as file:
    reader = csv.reader(file)
    # Skip the header row
    next(reader)
    # Create an empty dictionary to store the prices for each stock
    prices = {}
    # Loop through each row of the CSV file
    for row in reader:
        # Extract the symbol, date, and price from the row
        ticker, date, price = row
        # Convert the price from a string to a float
        price = float(price)
        # Check if the ticker symbol is already in the dictionary
        if ticker in prices:
            # If the ticker symbol is already in the dictionary, add the price to the list of pri
            prices[ticker].append(price)
        else:
            # If the ticker symbol is not already in the dictionary, create a new list with the f
            prices[ticker] = [price]

# Loop through the dictionary of prices for each stock
for ticker, price_list in prices.items():
    # Calculate the highest and lowest prices for the stock
    highest_price = max(price_list)
    lowest_price = min(price_list)
    # Print the results
    print(f"{ticker}: Highest Price = ${highest_price:.2f}, Lowest Price = ${lowest_price:.2f}")
```

```
AAPL: Highest Price = $142.20, Lowest Price = $135.90
GOOG: Highest Price = $2125.50, Lowest Price = $2098.00
MSFT: Highest Price = $345.20, Lowest Price = $342.10
```

Q4.
a) Write a Python program to remove duplicates from a list of lists. Sample list: [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]

Program:

```
# Define the list of lists with duplicates
list_of_lists =  [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]

# Create an empty set to store the unique lists
unique_lists = set()

# Loop through each list in the original list of lists
for lst in list_of_lists:
    # Convert the list to a tuple (because lists are not hashable, but tuples are)
    lst_tuple = tuple(lst)
    # Add the tuple to the set of unique tuples
    unique_lists.add(lst_tuple)

# Convert the set of unique tuples back to a list of lists
unique_list_of_lists = [list(lst_tuple) for lst_tuple in unique_lists]

# Print the original list of lists and the unique list of lists
print("Original List of Lists:")
print(list_of_lists)
print("Unique List of Lists:")
print(unique_list_of_lists)
```

```
# Define the list of lists with duplicates
list_of_lists =  [[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]

# Create an empty set to store the unique lists
unique_lists = set()

# Loop through each list in the original list of lists
for lst in list_of_lists:
    # Convert the list to a tuple (because lists are not hashable, but tuples are)
    lst_tuple = tuple(lst)
    # Add the tuple to the set of unique tuples
    unique_lists.add(lst_tuple)

# Convert the set of unique tuples back to a list of lists
unique_list_of_lists = [list(lst_tuple) for lst_tuple in unique_lists]

# Print the original list of lists and the unique list of lists
print("Original List of Lists:")
print(list_of_lists)
print("Unique List of Lists:")
print(unique_list_of_lists)
```

```
Original List of Lists:
[[10, 20], [40], [30, 56, 25], [10, 20], [33], [40]]
Unique List of Lists:
[[30, 56, 25], [40], [10, 20], [33]]
```

b)Write a Python program which takes a list and returns a list with the elements "shifted left by one position" so [1, 2, 3] yields [2, 3, 1]. Example: [1, 2, 3] → [2, 3, 1] [11, 12, 13] → [12, 13, 11]

Program:

```
def shift_left(lst):
    # Check if the list is empty or has only one element
    if len(lst) <= 1:
        return lst
    # Shift the elements of the list to the left by one position
    shifted_lst = lst[1:] + [lst[0]]
    # Return the shifted list
    return shifted_lst
list1 = [1,2,3]
print(shift_left(list1))
```

```
def shift_left(lst):
    # Check if the list is empty or has only one element
    if len(lst) <= 1:
        return lst
    # Shift the elements of the list to the left by one position
    shifted_lst = lst[1:] + [lst[0]]
    # Return the shifted list
    return shifted_lst
list1 = [1,2,3]
print(shift_left(list1))
```

```
[2, 3, 1]
```

c)Iterate a given list and count the occurrence of each element and create a dictionary to show the count of each element. Original list [11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89] And expected output is: Printing count of each item {11: 3, 45: 3, 8: 1, 23: 2, 89: 2}

Program:

```
def count_occurrences(lst):
    # Create an empty dictionary to store the count of each element
    count_dict = {}
    # Loop through each element in the list
    for elem in lst:
        # If the element is already in the dictionary, increment its count
        if elem in count_dict:
            count_dict[elem] += 1
        # Otherwise, add the element to the dictionary with a count of 1
        else:
            count_dict[elem] = 1
    # Return the dictionary of element counts
    return count_dict
list1 =[11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89]
print(count_occurrences(list1))
```

```
def count_occurrences(lst):
    # Create an empty dictionary to store the count of each element
    count_dict = {}
    # Loop through each element in the list
    for elem in lst:
        # If the element is already in the dictionary, increment its count
        if elem in count_dict:
            count_dict[elem] += 1
        # Otherwise, add the element to the dictionary with a count of 1
        else:
            count_dict[elem] = 1
    # Return the dictionary of element counts
    return count_dict
list1 =[11, 45, 8, 11, 23, 45, 23, 45, 89, 11, 89]
print(count_occurrences(list1))
```

```
{11: 3, 45: 3, 8: 1, 23: 2, 89: 2}
```