
**Department of Computer Science and Engineering**

# Python Programming(ITO- 804): Assignment 1

**Name:** Manas Baigra

**Roll No:** 191203093

**Semester:** 8th A1

**Submitted to:** Mr. Saurabh Sharma

## 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:<br>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.<br>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))<br><br>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. give me
# solution of this program

def best_days_to_buy_sell_stocks(prices):
    # initialize variables to keep track of best buying and selling days
    best_buy_day = 0
    best_sell_day = 0
    max_profit = 0

    # iterate over each day's price except the last day
    for i in range(len(prices)-1):
        # iterate over the prices of the following days
        for j in range(i+1, len(prices)):
            # check if the profit of buying on day i and selling on day j is greater than the current max_profit
            if prices[j] - prices[i] > max_profit:
                max_profit = prices[j] - prices[i]
                best_buy_day = i
                best_sell_day = j

    return (best_buy_day+1, best_sell_day+1) # return as a tuple of days (adding 1 to indices to get the actual day number)

# Driver code
prices = (100, 180, 260, 310, 40, 535, 695)
best_days = best_days_to_buy_sell_stocks(prices)
print("Buy stock on day", best_days[0], "and sell on day", best_days[1])


     Buy stock 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',
# 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.

def earliest_trilogy_year(books):
    years = [year for title, year in books]
    years.sort()

    for i in range(len(years)-2):
        if years[i]+1 == years[i+1] and years[i+1]+1 == years[i+2]:
            return years[i]
    return None

# Example usage:
books = [("Book 1", 2010), ("Book 2", 2011), ("Book 3", 2012), ("Book 4", 2015), ("Book 5", 2016)]
```

```
print(earliest_trilogy_year(books))
```

```
2010
```

```
# 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: (2.5 Marks)
# 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

import csv
highest_prices = {}
lowest_prices = {}
with open('/content/Stock_CSV_FILE.csv', 'r') as file:
    reader = csv.reader(file)
    next(reader)
    for row in reader:

        ticker, date, price = row


        if ticker not in highest_prices:
            highest_prices[ticker] = float(price)
            lowest_prices[ticker] = float(price)
        else:
            highest_prices[ticker] = max(highest_prices[ticker], float(price))
            lowest_prices[ticker] = min(lowest_prices[ticker], float(price))

# print the results
for ticker in highest_prices:
    print(f"{ticker}: Highest price = {highest_prices[ticker]}, Lowest price = {lowest_prices[ticker]}")
```

```
AAPL: Highest price = 142.2, Lowest price = 135.9
GOOG: Highest price = 2125.5, Lowest price = 2098.0
MSFT: Highest price = 345.2, Lowest price = 342.1
```

```
# 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]]
# 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}

# A)

def remove_duplicates(lst):
    seen = set()
    result = []
    for sublst in lst:
        tup = tuple(sublst)
        if tup not in seen:
            seen.add(tup)
            result.append(list(tup))
    return result

# Example usage:
lst = [[1, 2, 3], [4, 5, 6], [1, 2, 3], [7, 8, 9], [4, 5, 6]]
print("Q4 (A)",remove_duplicates(lst))

# B)

def shift_left(lst):
```

```
def shift_left(lst):
    if len(lst) < 2:
        return lst
    else:
        return lst[1:] + [lst[0]]

# Example usage:
lst = [1, 2, 3, 4, 5]
print("Q4 (B)",shift_left(lst))

# C)

def count_elements(lst):
    count_dict = {}
    for element in lst:
        if element in count_dict:
            count_dict[element] += 1
        else:
            count_dict[element] = 1
    return count_dict

# Example usage:
lst = [1, 2, 3, 1, 2, 1, 4, 5, 4]
print("Q4 (C)",count_elements(lst))
```

```
Q4 (A) [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
Q4 (B) [2, 3, 4, 5, 1]
Q4 (C) {1: 3, 2: 2, 3: 1, 4: 2, 5: 1}
```