

# Department of Computer Science and Engineering

Python Programming(ITO- 804): Assignment 1

# Name: Tanishq Thakur

**Roll number : 191203115 Semester : 8**

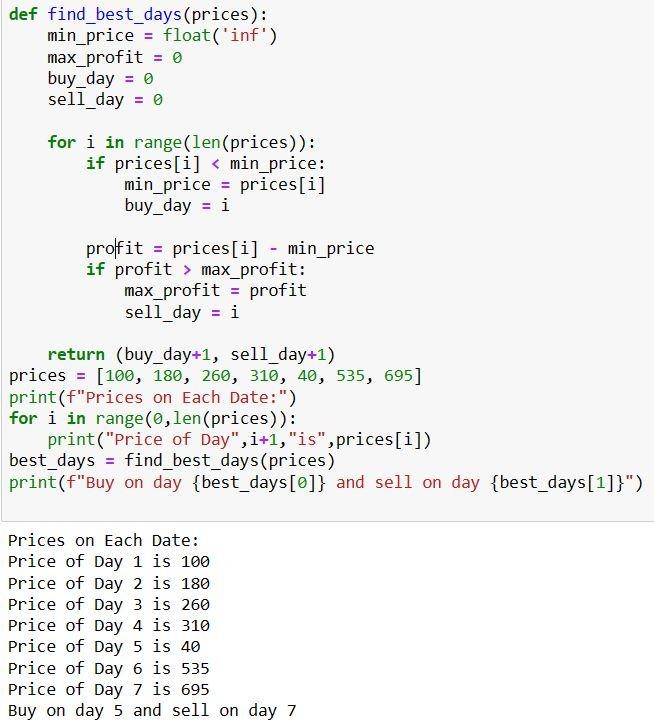
# Submitted to: Mr. Saurabh Sharma

**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:  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]}")

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]

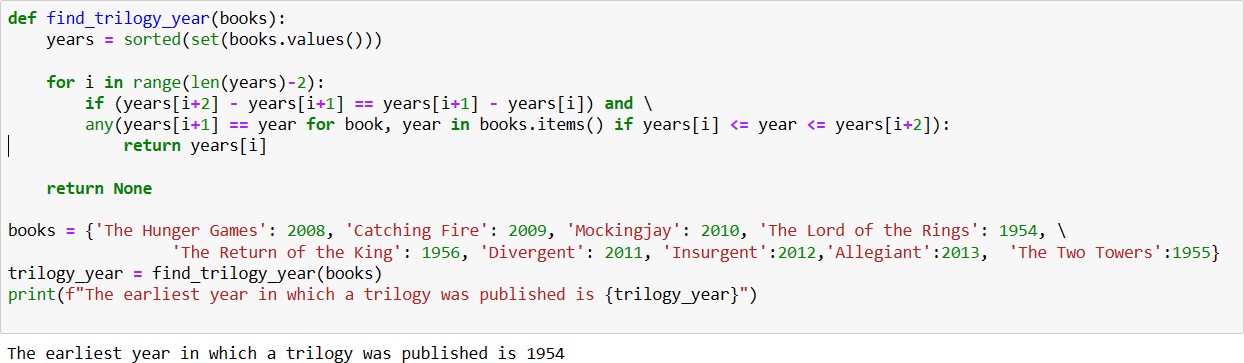
return None

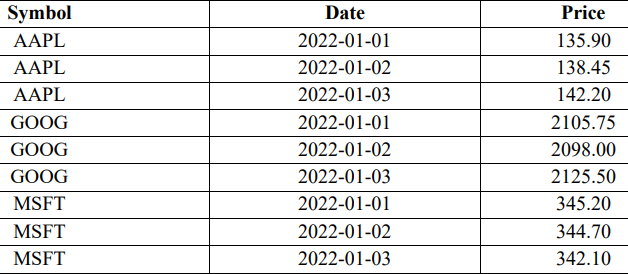
books = {'The Hunger Games': 2008, 'Catching Fire': 2009, 'Mockingjay': 2010, 'The Lord of the Rings': 1954, \

'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}")



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:

Program:

import csv

with open('3\_csv.csv') as file: reader = csv.reader(file)

next(reader) # Skip the header row

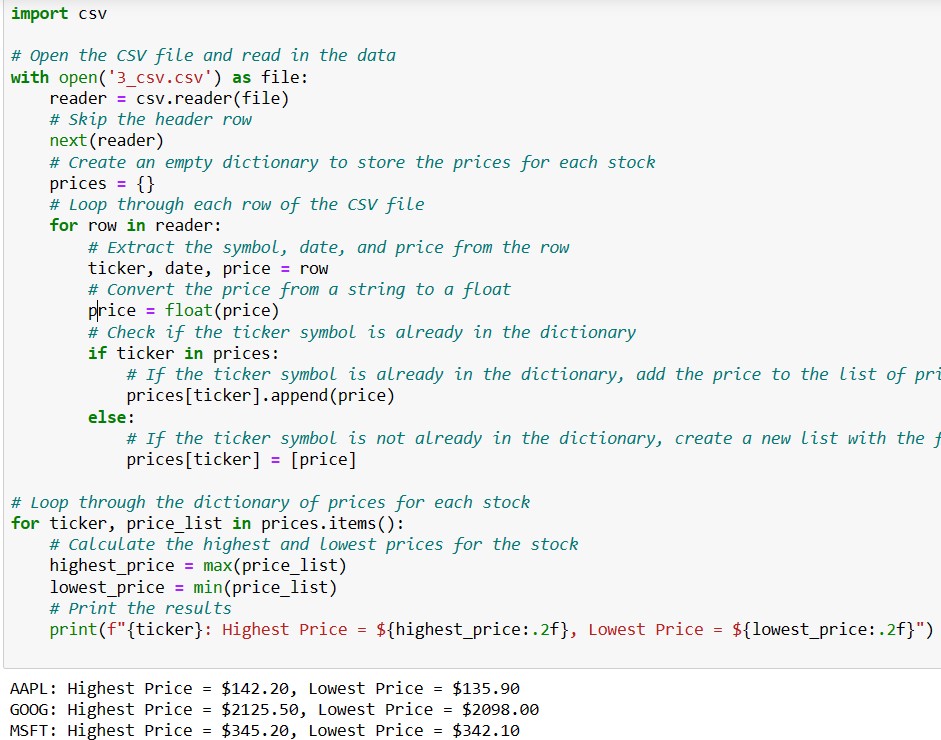
prices = {} # Create an empty dictionary to store the prices for each stock for row in reader:

# 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}")

Q4.

1. 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()

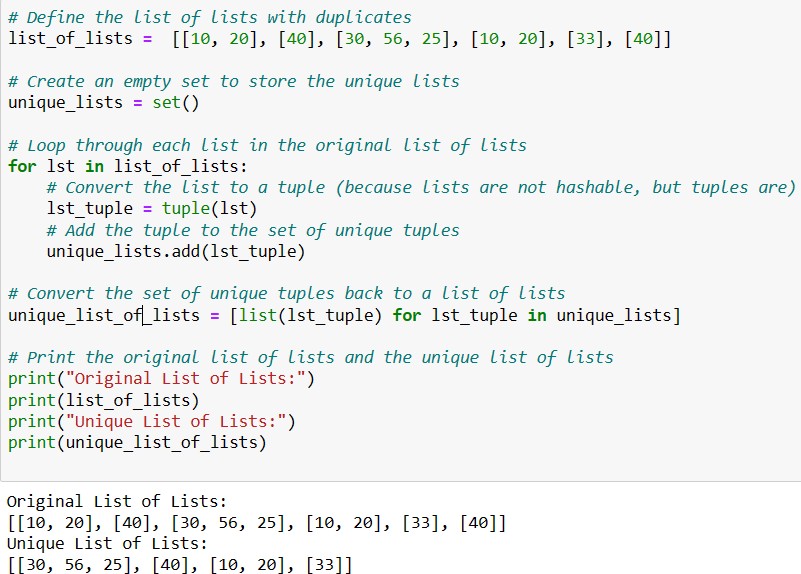
# Loop through each list in the original list of lists for lst in list\_of\_lists:

# 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]

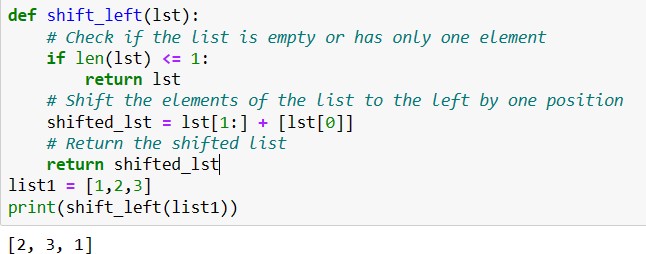
# Print the original list of lists and the unique list of lists print("Original List of Lists:")

print(list\_of\_lists) print("Unique List of Lists:") print(unique\_list\_of\_lists)

1. 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

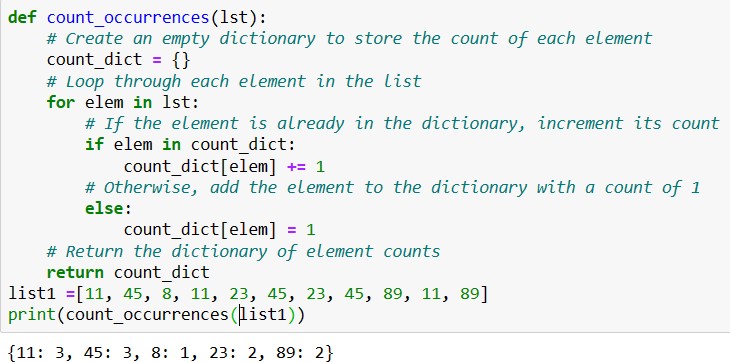
shifted\_lst = lst[1:] + [lst[0]] # Return the shifted list return shifted\_lst

list1 = [1,2,3] print(shift\_left(list1))

1. 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))