Question1: Say you have an array for which the ith element is the price of a given stock on day i. If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Answer

def max\_profit(prices):

if not prices:

return 0

min\_price = prices[0]

max\_profit = 0

for price in prices:

# Update min\_price with the minimum price seen so far

min\_price = min(min\_price, price)

# Calculate the profit if we sold at current price

profit = price - min\_price

# Update max\_profit with the maximum profit seen so far

max\_profit = max(max\_profit, profit)

return max\_profit

# Test

prices = [7,1,5,3,6,4]

print(max\_profit(prices)) # Output: 5 (buy on day 2 and sell on day 5)

Question:2. Write a function that finds the most repeated character in a string.

ANswer:

def most\_repeated\_char(s):

# If the string is empty, return None

if not s:

return None

# Create a dictionary to count occurrences of each character

char\_count = {}

for char in s:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

# Find the character(s) with the maximum count

max\_count = max(char\_count.values())

most\_repeated\_chars = [char for char, count in char\_count.items() if count == max\_count]

# If there's only one character that is most repeated, return it

if len(most\_repeated\_chars) == 1:

return most\_repeated\_chars[0]

else:

# If there are multiple characters that are most repeated, you can return all of them

# as a list or any other logic as needed.

# For simplicity, this code will return the first one encountered.

return most\_repeated\_chars[0]

# Test

s = "aabbccc"

print(most\_repeated\_char(s)) # Expected output: 'c'

Question3: Write a function that returns the longest consecutive subsequence in a list of numbers.

ANSWER:

def longest\_consecutive\_subsequence(nums):

if not nums:

return []

nums = sorted(set(nums))

longest\_streak = []

current\_streak = [nums[0]]

for i in range(1, len(nums)):

if nums[i] - nums[i - 1] == 1:

current\_streak.append(nums[i])

else:

if len(current\_streak) > len(longest\_streak):

longest\_streak = current\_streak

current\_streak = [nums[i]]

return longest\_streak if len(longest\_streak) > len(current\_streak) else current\_streak

numbers = [1, 2, 3, 5, 6, 7, 8, 10]

print(longest\_consecutive\_subsequence(numbers)) # Output: [5, 6, 7, 8]

Question4: Write a function to find the longest common prefix of a list of strings.

def longest\_common\_prefix(strings):

if not strings:

return ""

prefix = strings[0]

for s in strings[1:]:

while not s.startswith(prefix):

prefix = prefix[:-1]

return prefix

strings = ["flower", "flow", "flight"]

print(longest\_common\_prefix(strings)) # Output: "fl"

Question5: Write a function to find the two numbers in a list that sum up to a specific target.

def two\_sum(nums, target):

num\_dict = {}

for i, num in enumerate(nums):

complement = target - num

if complement in num\_dict:

return [num\_dict[complement], i]

num\_dict[num] = i

return None

numbers = [2, 7, 11, 15]

target\_value = 9

print(two\_sum(numbers, target\_value)) # Output: [0, 1]

Question6: Write a function that reverses a string, but maintains the position of all non-alphabetic characters.

def reverse\_alphabet\_only(s):

s = list(s)

i, j = 0, len(s) - 1

while i < j:

if not s[i].isalpha():

i += 1

elif not s[j].isalpha():

j -= 1

else:

s[i], s[j] = s[j], s[i]

i += 1

j -= 1

return ''.join(s)

string = "ab@cd#ef$gh"

print(reverse\_alphabet\_only(string)) # Output: "hg@fe#dc$ba"

Question 7: Write a function to multiply two matrixes and verify using numpy.

Answer:

def matrix\_multiply(A, B):

"""

Multiply matrix A by matrix B.

Parameters:

- A: List[List[int/float]]

- B: List[List[int/float]]

Returns:

- result: List[List[int/float]]

"""

# Get dimensions of matrices

rows\_A = len(A)

cols\_A = len(A[0])

rows\_B = len(B)

cols\_B = len(B[0])

# Check if matrices can be multiplied

if cols\_A != rows\_B:

raise ValueError("Number of columns in A must be equal to number of rows in B")

# Initialize result matrix with zeros

result = [[0 for \_ in range(cols\_B)] for \_ in range(rows\_A)]

# Multiply matrices

for i in range(rows\_A):

for j in range(cols\_B):

for k in range(cols\_A):

result[i][j] += A[i][k] \* B[k][j]

return result

# Test

A = [

[1, 2, 3],

[4, 5, 6]

]

B = [

[7, 8],

[9, 10],

[11, 12]

]

print(matrix\_multiply(A, B))