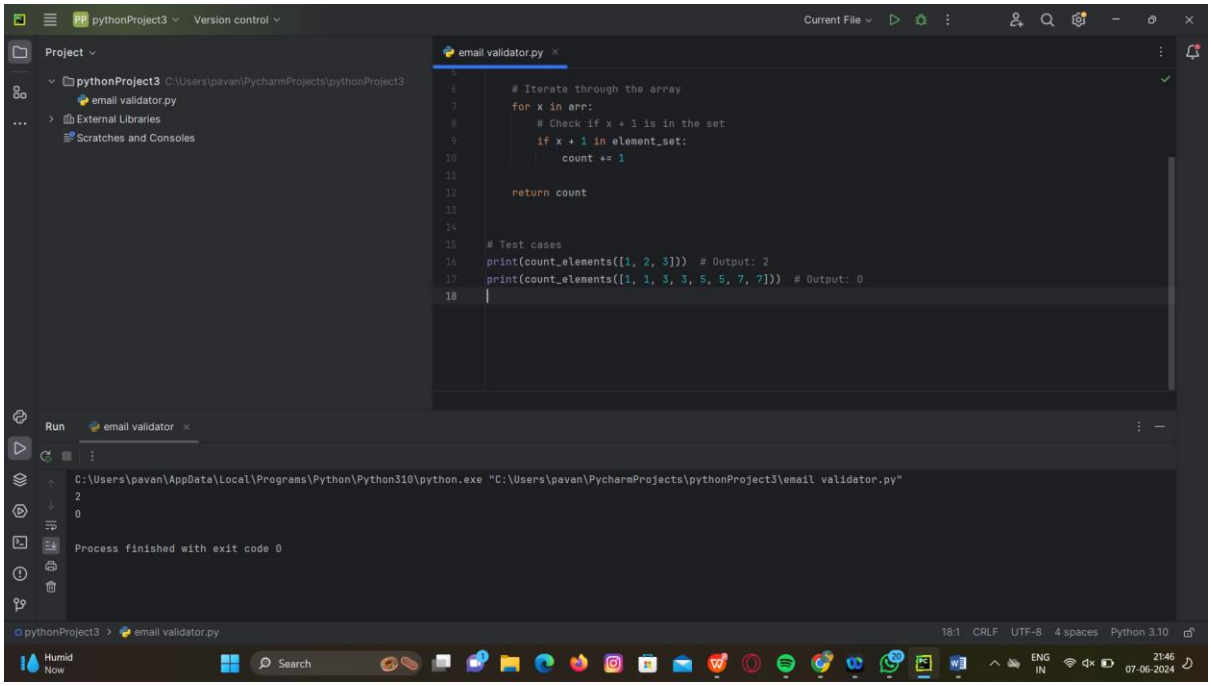
EXPERIMENT 30-40

S

1. Counting Elements

Given an integer array arr, count how many elements x there are, such that x + 1 is also in arr. If there are duplicates in arr, count them separately

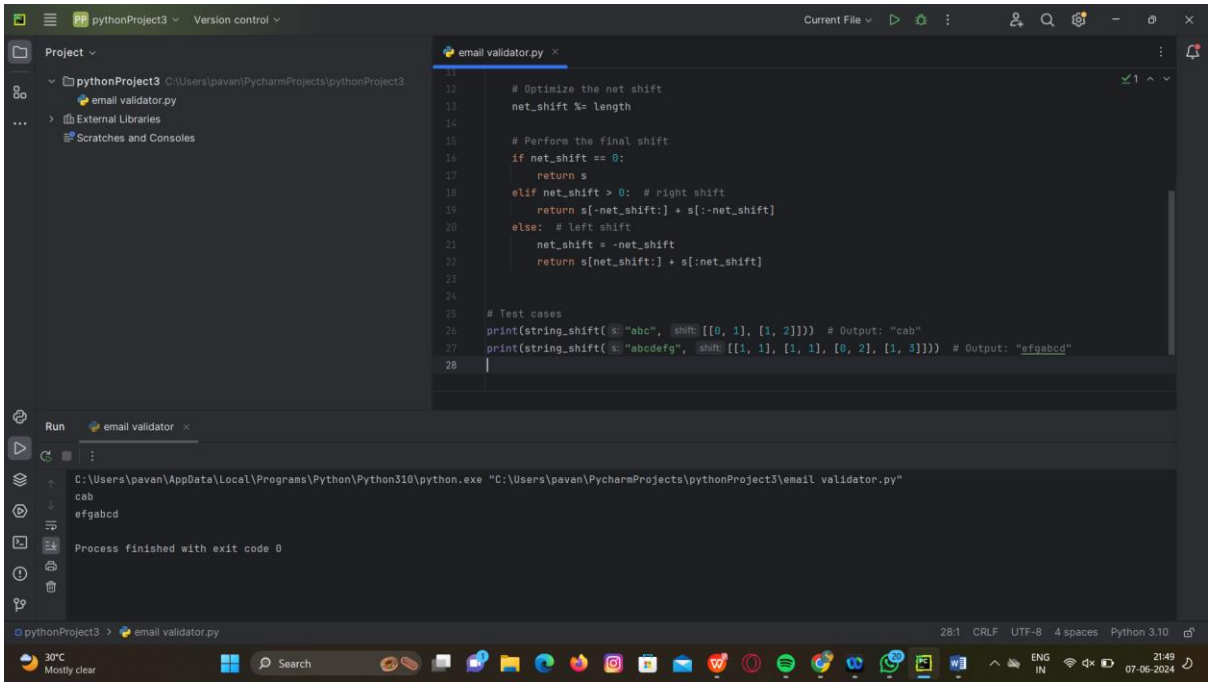
```
def count_elements(arr):
    # Convert the list to a set for O(1) average time complexity for membership checks
    element_set = set(arr)
    count = 0

    # Iterate through the array
    for x in arr:
        # Check if x + 1 is in the set
        if x + 1 in element_set:
            count += 1

    return count

# Test cases
print(count_elements([1, 2, 3]))       # Output: 2
print(count_elements([1, 1, 3, 3, 5, 5, 7, 7]))  # Output: 0
```



2. Perform String Shifts

```
def string_shift(s, shift):
    net_shift = 0
    length = len(s)

    # Calculate net shift
    for direction, amount in shift:
        if direction == 0:  # left shift
            net_shift -= amount
        else:  # right shift
            net_shift += amount

    # Optimize the net shift
    net_shift %= length

    # Perform the final shift
    if net_shift == 0:
```

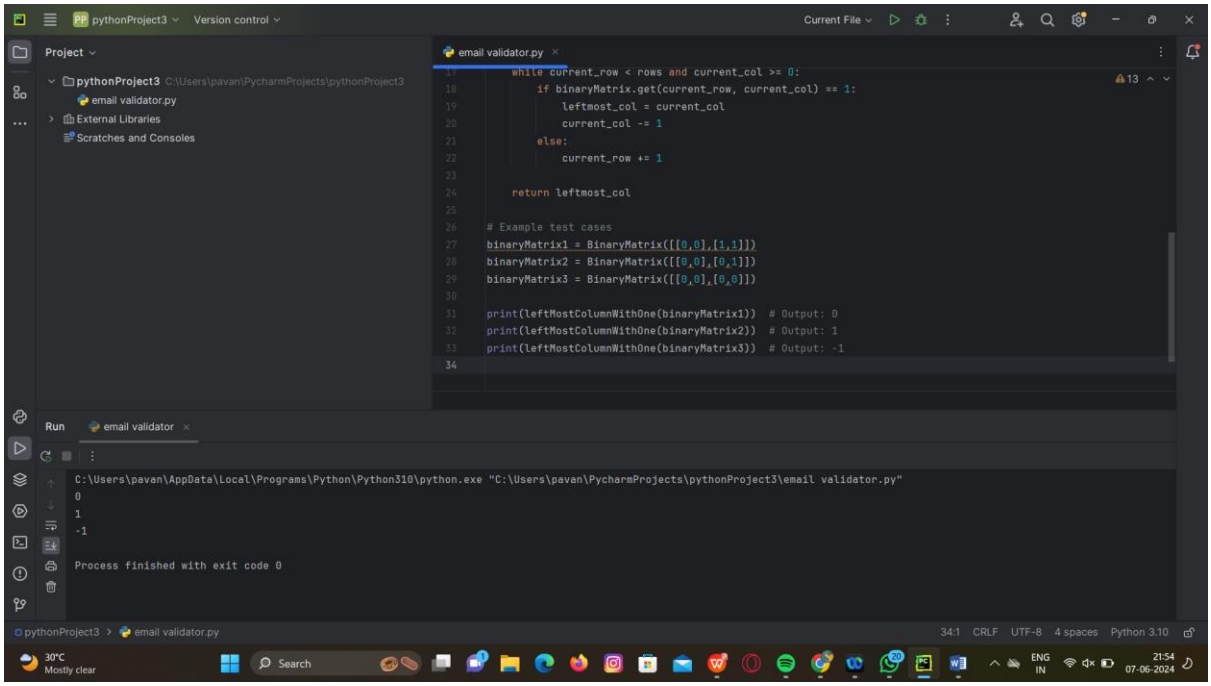
```
        return s
    elif net_shift > 0:  # right shift
        return s[-net_shift:] + s[:-net_shift]
    else:  # left shift
        net_shift = -net_shift
        return s[net_shift:] + s[:net_shift]


# Test cases
print(string_shift("abc", [[0,1],[1,2]]))         # Output: "cab"
print(string_shift("abcdefg", [[1,1],[1,1],[0,2],[1,3]]))  # Output: "efgabcd"
```



3. Leftmost Column with at Least a One

```
class BinaryMatrix:
    def __init__(self, matrix):
        self.matrix = matrix


    def get(self, row, col):
        return self.matrix[row][col]
```

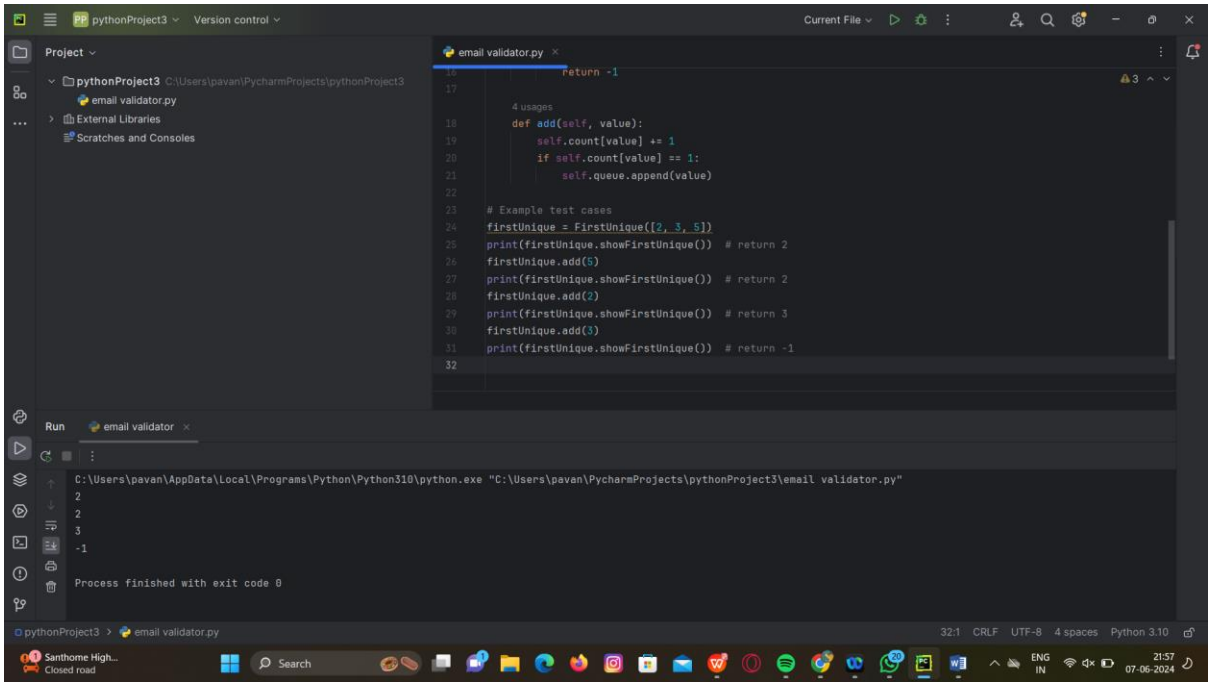
```
    def dimensions(self):
        return [len(self.matrix), len(self.matrix[0])]

def leftMostColumnWithOne(binaryMatrix):
    rows, cols = binaryMatrix.dimensions()
    current_row = 0
    current_col = cols - 1
    leftmost_col = -1

    while current_row < rows and current_col >= 0:
        if binaryMatrix.get(current_row, current_col) == 1:
            leftmost_col = current_col
            current_col -= 1
        else:
            current_row += 1

    return leftmost_col

# Example test cases
binaryMatrix1 = BinaryMatrix([[0,0],[1,1]])
binaryMatrix2 = BinaryMatrix([[0,0],[0,1]])
binaryMatrix3 = BinaryMatrix([[0,0],[0,0]])

print(leftMostColumnWithOne(binaryMatrix1))  # Output: 0
print(leftMostColumnWithOne(binaryMatrix2))  # Output: 1
print(leftMostColumnWithOne(binaryMatrix3))  # Output: -1
```



4.first unique number

```
from collections import deque, defaultdict


class FirstUnique:
    def __init__(self, nums):
        self.queue = deque()
        self.count = defaultdict(int)
        for num in nums:
            self.add(num)


    def showFirstUnique(self):
        while self.queue and self.count[self.queue[0]] > 1:
            self.queue.popleft()
        if self.queue:
            return self.queue[0]
        else:
            return -1
```

```
    def add(self, value):
        self.count[value] += 1
        if self.count[value] == 1:
            self.queue.append(value)


# Example test cases
firstUnique = FirstUnique([2, 3, 5])
print(firstUnique.showFirstUnique())  # return 2
firstUnique.add(5)
print(firstUnique.showFirstUnique())  # return 2
firstUnique.add(2)
print(firstUnique.showFirstUnique())  # return 3
firstUnique.add(3)
print(firstUnique.showFirstUnique())  # return -1
```



5.

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
```

```
        self.left = left
        self.right = right

def isValidSequence(root, arr):
    def dfs(node, index):
        if node is None:
            return False

        if index >= len(arr) or node.val != arr[index]:
            return False

        if node.left is None and node.right is None and index == len(arr) - 1:
            return True

        return dfs(node.left, index + 1) or dfs(node.right, index + 1)

    return dfs(root, 0)

# Example binary tree:
#     0
#    / \
#   1   0
#  /|   |\
# 0 1   0 0
#   |   | \
#   1   0  0

root = TreeNode(0)
root.left = TreeNode(1)
```

root.right = TreeNode(0)

root.left.left = TreeNode(0)

root.left.right = TreeNode(1)

root.left.right.left = TreeNode(1)

root.right.left = TreeNode(0)

root.right.left.right = TreeNode(0)

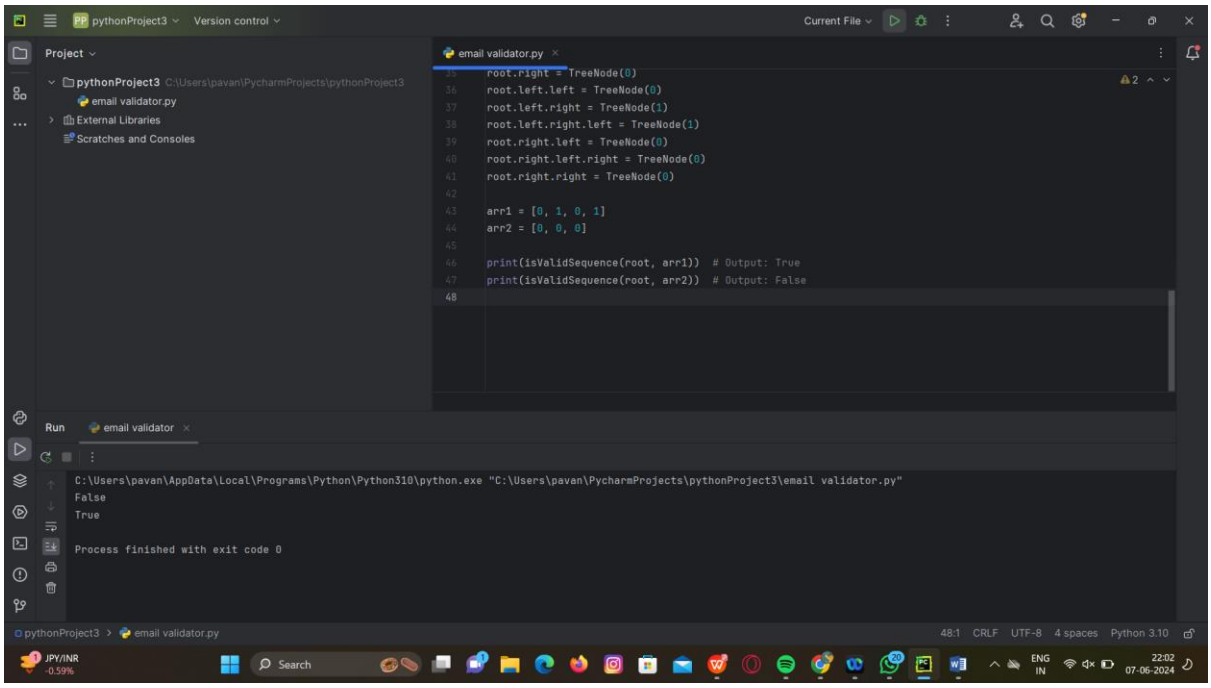
root.right.right = TreeNode(0)

arr1 = [0, 1, 0, 1]

arr2 = [0, 0, 0]

print(isValidSequence(root, arr1)) # Output: True

print(isValidSequence(root, arr2)) # Output: False



6.

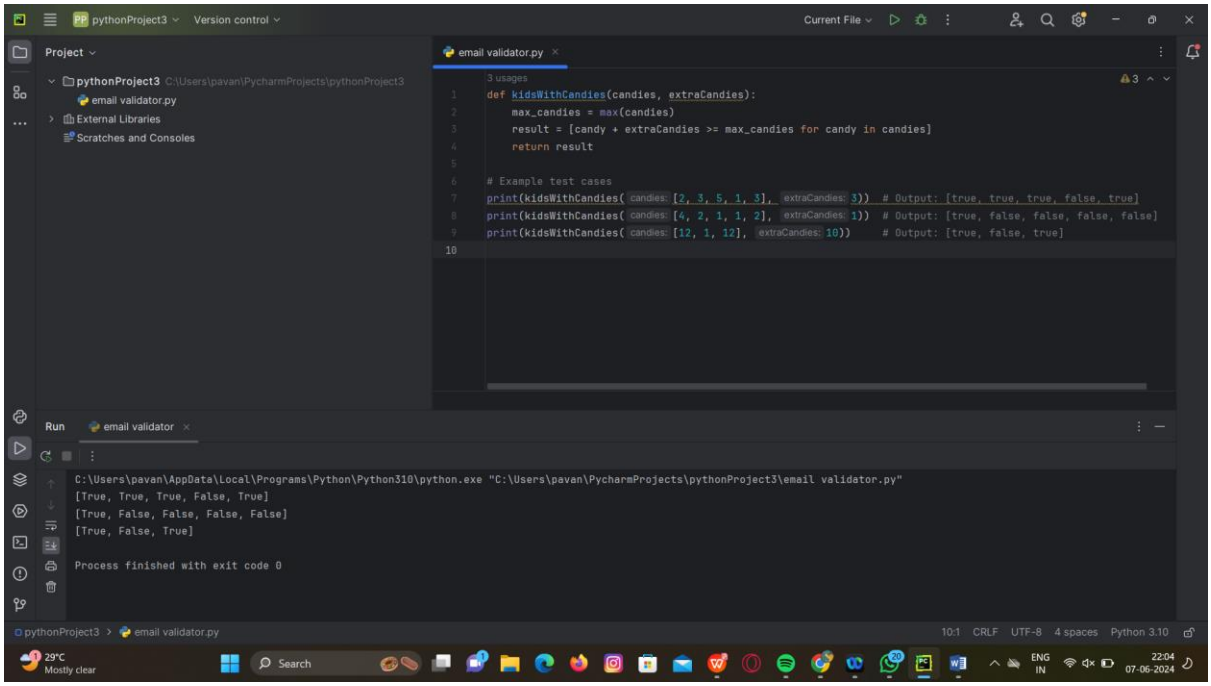
```
def kidsWithCandies(candies, extraCandies):
    max_candies = max(candies)
    result = [candy + extraCandies >= max_candies for candy in candies]
    return result
```

Example test cases

print(kidsWithCandies([2, 3, 5, 1, 3], 3)) # Output: [true, true, true, false, true]

print(kidsWithCandies([4, 2, 1, 1, 2], 1)) # Output: [true, false, false, false, false]

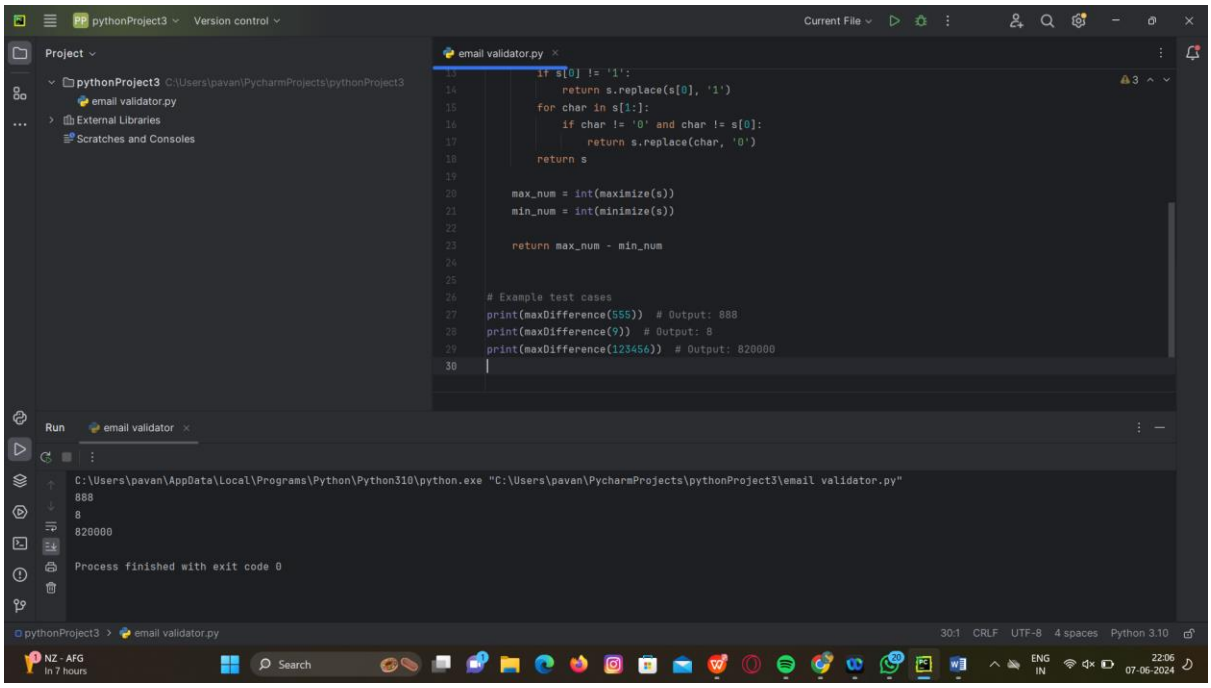
print(kidsWithCandies([12, 1, 12], 10))   # Output: [true, false, true]



7.

```
def maxDifference(num):
    s = str(num)

    # Function to maximize the number
    def maximize(s):
        for char in s:
            if char != '9':
                return s.replace(char, '9')
        return s

    # Function to minimize the number
    def minimize(s):
```

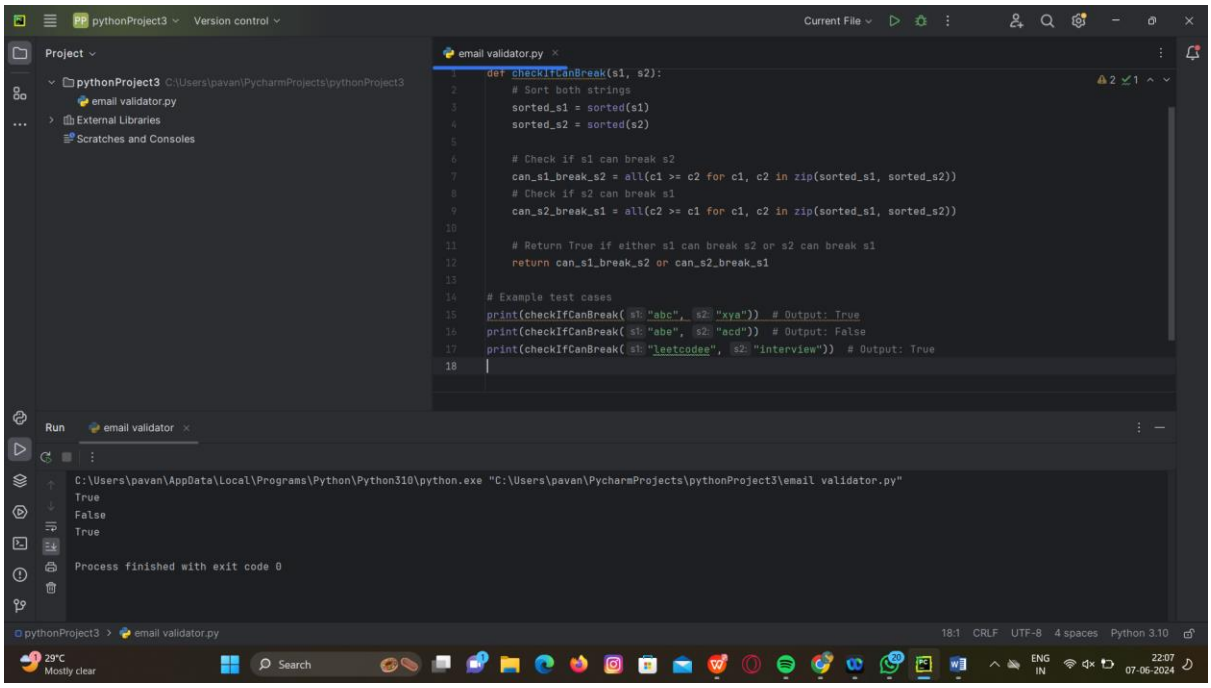
```
        if s[0] != '1':
            return s.replace(s[0], '1')
        for char in s[1:]:
            if char != '0' and char != s[0]:
                return s.replace(char, '0')
        return s

    max_num = int(maximize(s))
    min_num = int(minimize(s))

    return max_num - min_num

# Example test cases
print(maxDifference(555))     # Output: 888
print(maxDifference(9))       # Output: 8
print(maxDifference(123456))  # Output: 820000
```



8.

```
def checkIfCanBreak(s1, s2):
```

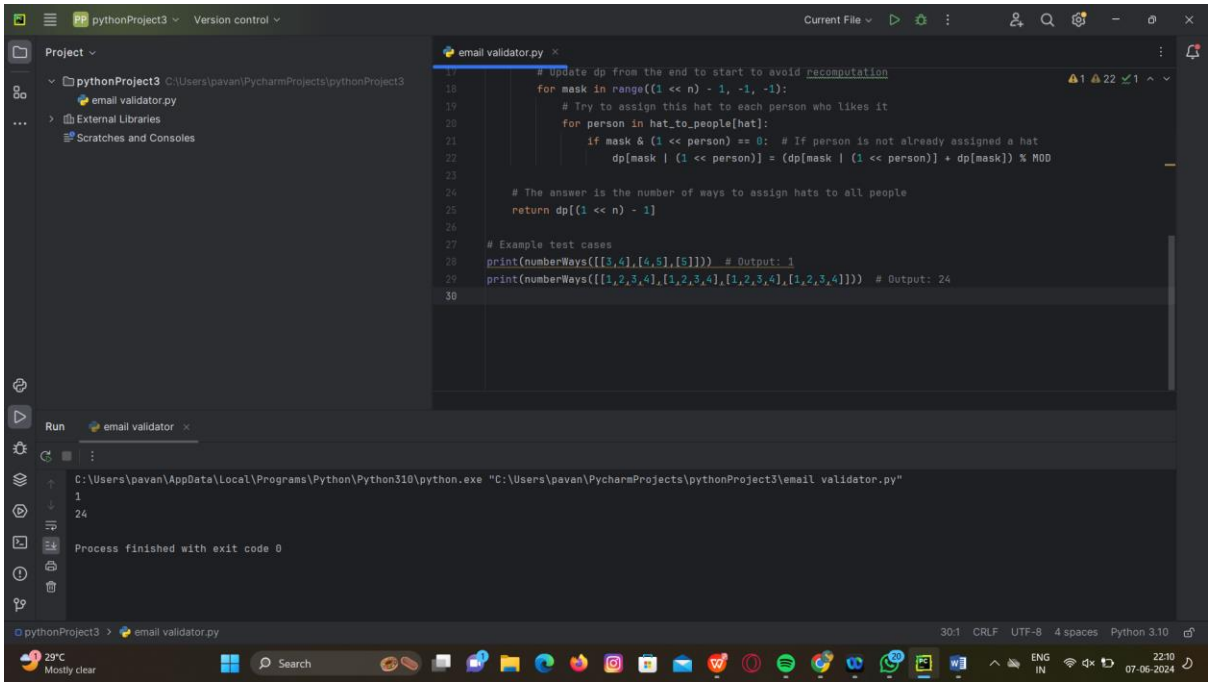
```
    # Sort both strings
    sorted_s1 = sorted(s1)
    sorted_s2 = sorted(s2)

    # Check if s1 can break s2
    can_s1_break_s2 = all(c1 >= c2 for c1, c2 in zip(sorted_s1, sorted_s2))
    # Check if s2 can break s1
    can_s2_break_s1 = all(c2 >= c1 for c1, c2 in zip(sorted_s1, sorted_s2))

    # Return True if either s1 can break s2 or s2 can break s1
    return can_s1_break_s2 or can_s2_break_s1

# Example test cases
print(checkIfCanBreak("abc", "xya"))  # Output: True
print(checkIfCanBreak("abe", "acd"))  # Output: False
print(checkIfCanBreak("leetcodee", "interview"))  # Output: True
```



9.

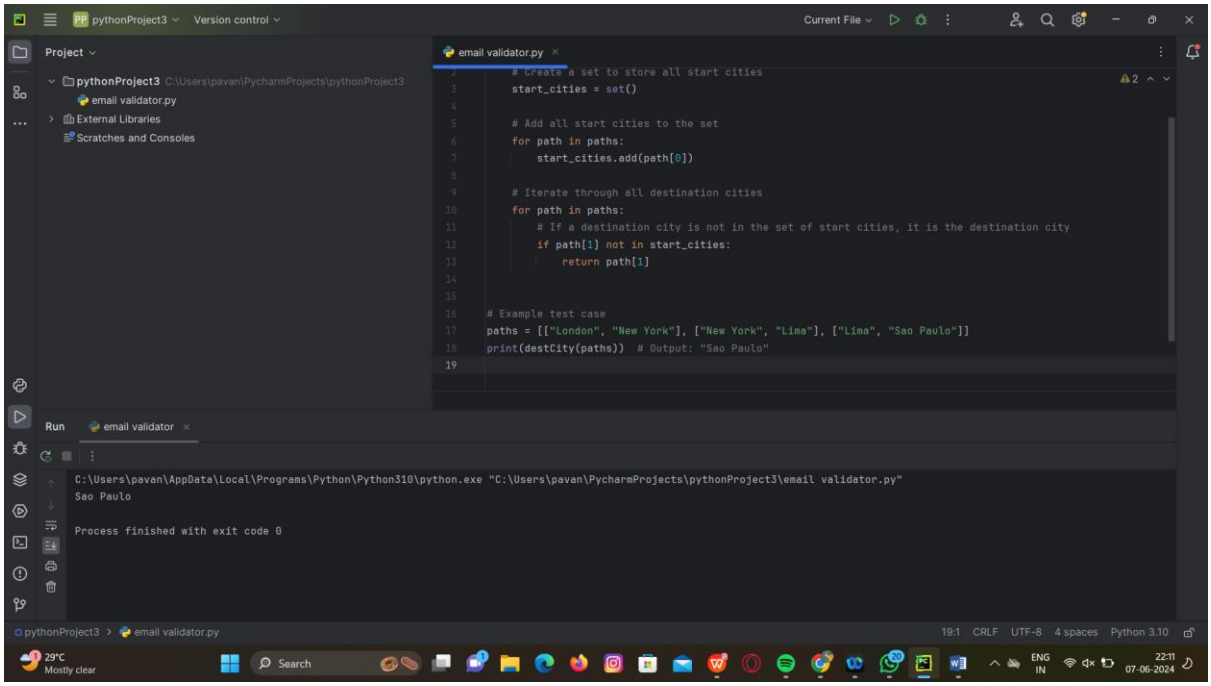
```
def numberWays(hats):
```

```
    MOD = 10**9 + 7
    n = len(hats)

    # Map each hat to the list of people who like it
    hat_to_people = [[] for _ in range(41)]
    for person, person_hats in enumerate(hats):
        for hat in person_hats:
            hat_to_people[hat].append(person)

    # dp[mask] will be the number of ways to assign hats to the people represented by mask
    dp = [0] * (1 << n)
    dp[0] = 1

    # Iterate over all hats
    for hat in range(1, 41):
        # Update dp from the end to start to avoid recomputation
        for mask in range((1 << n) - 1, -1, -1):
            # Try to assign this hat to each person who likes it
            for person in hat_to_people[hat]:
                if mask & (1 << person) == 0:  # If person is not already assigned a hat
                    dp[mask | (1 << person)] = (dp[mask | (1 << person)] + dp[mask]) % MOD

    # The answer is the number of ways to assign hats to all people
    return dp[(1 << n) - 1]

# Example test cases
print(numberWays([[3,4],[4,5],[5]]))  # Output: 1
print(numberWays([[1,2,3,4],[1,2,3,4],[1,2,3,4],[1,2,3,4]]))  # Output: 24
```



10.

```
def destCity(paths):
    # Create a set to store all start cities
    start_cities = set()

    # Add all start cities to the set
    for path in paths:
        start_cities.add(path[0])

    # Iterate through all destination cities
    for path in paths:
        # If a destination city is not in the set of start cities, it is the destination city
        if path[1] not in start_cities:
            return path[1]

# Example test case
paths = [["London", "New York"], ["New York", "Lima"], ["Lima", "Sao Paulo"]]
print(destCity(paths))  # Output: "Sao Paulo"
```


pythonProject3
Version control
Current File
Project
pythonProject3 C:\Users\pavan\PycharmProjects\pythonProject3
email validator.py
External Libraries
Scratches and Consoles
email validator.py
# Create a set to store all start cities
start_cities = set()
# Add all start cities to the set
for path in paths:
start_cities.add(path[0])
# Iterate through all destination cities
for path in paths:
# If a destination city is not in the set of start cities, it is the destination city
if path[1] not in start_cities:
return path[1]
# Example test case
paths = [["London", "New York"], ["New York", "Lima"], ["Lima", "Sao Paulo"]]
print(destCity(paths)) # Output: "Sao Paulo"
Run
email validator
C:\Users\pavan\AppData\Local\Programs\Python\Python310\python.exe "C:\Users\pavan\PycharmProjects\pythonProject3\email validator.py"
Sao Paulo
Process finished with exit code 0
pythonProject3 > email validator.py
19:1 CRLF UTF-8 4 spaces Python 3.10
29°C
Mostly clear
Search
ENG
IN
22:11
07-06-2024