1. Counting Elements

main.py Save Run Output Clear

```python
def count_elements(arr):
    return sum(1 for x in arr if x + 1 in arr)

# Example 1
arr1 = [1, 2, 3]
print(count_elements(arr1))  # Output: 2

# Example 2
arr2 = [1, 1, 3, 3, 5, 5, 7, 7]
print(count_elements(arr2))  # Output: 0
```

```
2
0

=== Code Execution Successful ===
```

2. Perform String Shifts

main.py Save Run Output

```python
def stringShift(s, shift):
    total_shift = 0
    for sh in shift:
        if sh[0] == 0:
            total_shift -= sh[1]
        else:
            total_shift += sh[1]

    total_shift %= len(s)
    return s[-total_shift:] + s[:-total_shift]
s1 = "abc"
shift1 = [[0,1],[1,2]]
print(stringShift(s1, shift1))
s2 = "abcdefg"
shift2 = [[1,1],[1,1],[0,2],[1,3]]
print(stringShift(s2, shift2))
```

```
cab
efgabcd

=== Code Execution Successful ===
```

3. Leftmost Column with at Least a One

main.py Save Run Output

```python
def generate_row_sorted_binary_matrix(rows, cols):
    import random
    matrix = [[random.randint(0, 1) for _ in range(cols)] for _ in range(rows)]
    for row in matrix:
        row.sort()
    return matrix

# Example Usage
rows = 2
cols = 2
binary_matrix = generate_row_sorted_binary_matrix(rows, cols)
print(binary_matrix)
```

```
[[1, 1], [1, 1]]

=== Code Execution Successful ===
```

4. First Unique Number

main.py

```python
from collections import deque

class FirstUnique:
    def __init__(self, nums):
        self.queue = deque()
        self.count = {}
        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]
        return -1
    def add(self, value):
        if value in self.count:
            self.count[value] += 1
        else:
            self.count[value] = 1
            self.queue.append(value)
firstUnique = FirstUnique([2, 3, 5])
print(firstUnique.showFirstUnique())
firstUnique.add(5)
print(firstUnique.showFirstUnique())
firstUnique.add(2)
print(firstUnique.showFirstUnique())
```

Output

```
2
2
3
-1

=== Code Execution Successful ===
```

5. Check If a String Is a Valid Sequence from Root to Leaves Path in a Binary Tree

main.py

```python
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None
def isValidSequence(root, sequence):
    def dfs(node, index):
        if not node or index >= len(sequence) or node.val != sequence[index]:
            return False
        if not node.left and not node.right and index == len(sequence) - 1:
            return True
        return dfs(node.left, index + 1) or dfs(node.right, index + 1)
    return dfs(root, 0)
root = TreeNode(0)
root.left = TreeNode(1)
root.right = TreeNode(0)
root.left.left = TreeNode(0)
root.left.right = TreeNode(1)
root.right.left = TreeNode(0)
root.left.left.right = TreeNode(1)
root.left.right.left = TreeNode(0)
root.left.right.right = TreeNode(0)
sequence = [0, 1, 1]
print(isValidSequence(root, sequence))
```

Output

```
False

=== Code Execution Successful ===
```

6. Kids With the Greatest Number of Candies

main.py

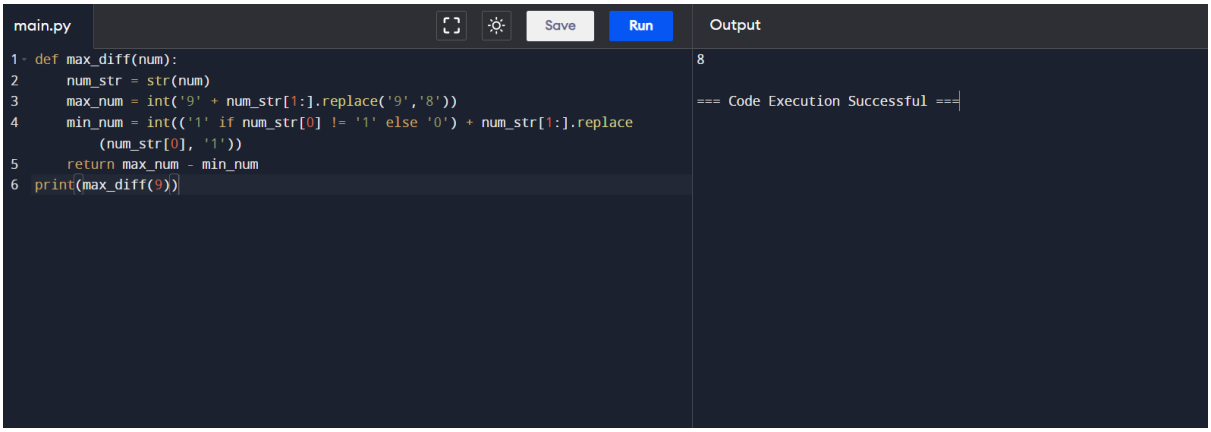
```python
def kidsWithCandies(candies, extraCandies):
    max_candies = max(candies)
    result = [candy + extraCandies >= max_candies for candy in candies]
    return result
candies = [2, 3, 5, 1, 3]
extraCandies = 3
print(kidsWithCandies(candies, extraCandies))
```

Output

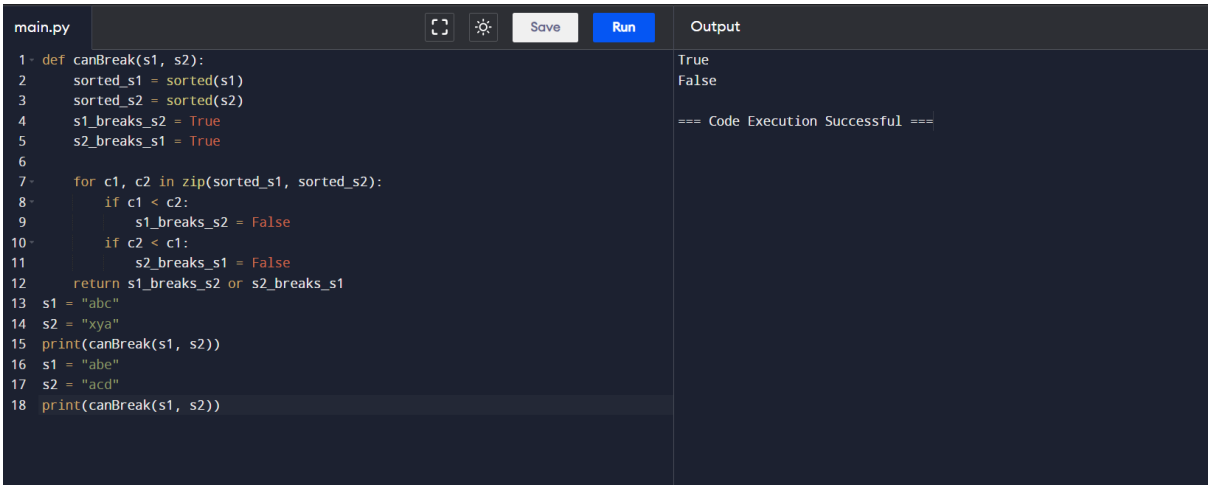
```
[True, True, True, False, True]

=== Code Execution Successful ===
```

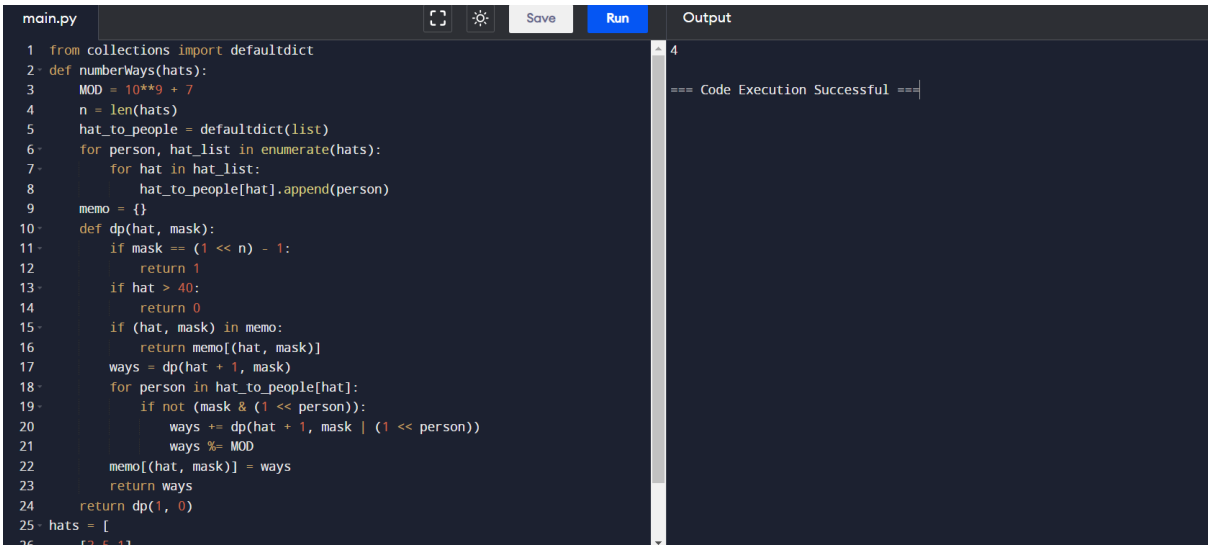
7. Max Difference You Can Get From Changing an Integer



8. Check If a String Can Break Another String



9. Number of Ways to Wear Different Hats to Each Other



10. Destination City

main.py

Save Run

```python
def destCity(paths):
    start_cities = set(cityA for cityA, cityB in paths)
    for cityA, cityB in paths:
        if cityB not in start_cities:
            return cityB
paths = [["London", "New York"], ["New York", "Lima"], ["Lima", "Sao Paulo"]]
print(destCity(paths))
```

Output

```
Sao Paulo

=== Code Execution Successful ===
```