Programiz Python Online Compiler

main.py

Share Run

```python
def find_palindromic_string(words):
    for word in words:
        if word == word[::-1]:
            return word
    return ""

# Example 1
words1 = ["abc", "car", "ada", "racecar", "cool"]
output1 = find_palindromic_string(words1)
print(output1)  # Output: "ada"

# Example 2
words2 = ["notapalindrome", "racecar"]
output2 = find_palindromic_string(words2)
print(output2)  # Output: "racecar"
```

Output

```
ada
racecar

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

Programiz Python Online Compiler

main.py

Share Run

```python
def calculate_indices(nums1, nums2):
    answer1 = sum(1 for num in nums1 if num in nums2)
    answer2 = sum(1 for num in nums2 if num in nums1)
    return [answer1, answer2]

# Example 1
nums1 = [2, 3, 2]
nums2 = [1, 2]
print(calculate_indices(nums1, nums2))  # Output: [2, 1]

# Example 2
nums1 = [4, 3, 2, 3, 1]
nums2 = [2, 2, 5, 2, 3, 6]
print(calculate_indices(nums1, nums2))  # Output: [3, 4]

```

Output

```
[2, 1]
[3, 4]

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

Programiz Python Online Compiler

main.py

Share Run

```python
def sum_of_squares_distinct_counts(nums):
    result = 0
    for i in range(len(nums)):
        count = {}
        for j in range(i, len(nums)):
            if nums[j] in count:
                count[nums[j]] += 1
            else:
                count[nums[j]] = 1
            result += sum([val ** 2 for val in count.values()])
    return result

# Example 1
nums1 = [1, 2, 1]
output1 = sum_of_squares_distinct_counts(nums1)
print(output1)  # Output: 15

# Example 2
nums2 = [1, 1]
output2 = sum_of_squares_distinct_counts(nums2)
print(output2)  # Output: 3
```

Output

```
12
6

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

Programiz Python Online Compiler

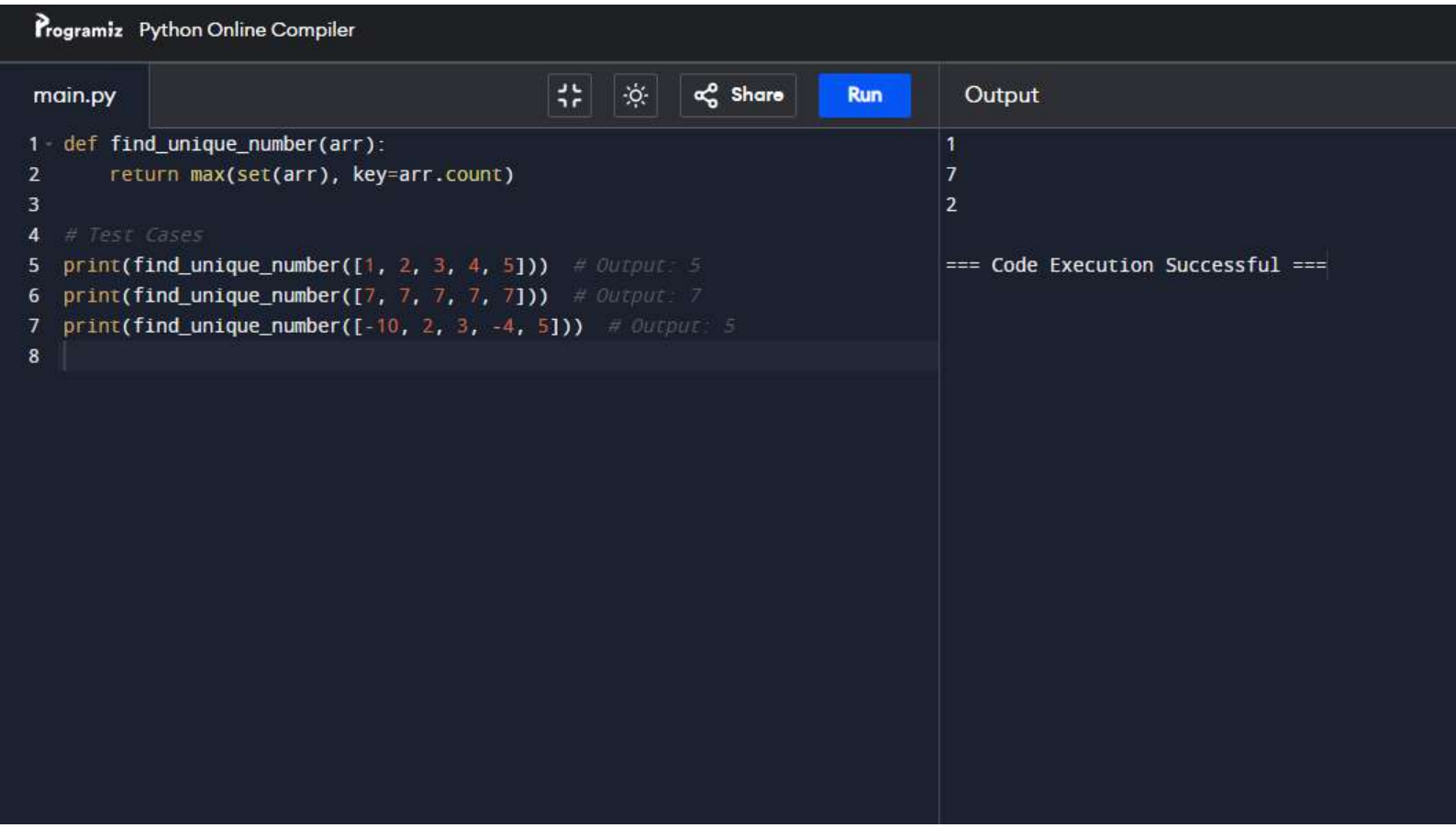
main.py

Share Run

```python
def count_pairs(nums, k):
    count = 0
    for i in range(len(nums)):
        for j in range(i+1, len(nums)):
            if nums[i] == nums[j] and (i * j) % k == 0:
                count += 1
    return count

# Example 1
nums1 = [3, 1, 2, 2, 2, 1, 3]
k1 = 2
output1 = count_pairs(nums1, k1)
print(output1)  # Output: 4

# Example 2
nums2 = [1, 2, 3, 4]
k2 = 1
output2 = count_pairs(nums2, k2)
print(output2)  # Output: 0

```

Output

```
4
0

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


Programiz Python Online Compiler
main.py
Share
Run
Output
def find_unique_number(arr):
return max(set(arr), key=arr.count)
Test Cases
print(find_unique_number([1, 2, 3, 4, 5])) # Output: 5
print(find_unique_number([7, 7, 7, 7, 7])) # Output: 7
print(find_unique_number([-10, 2, 3, -4, 5])) # Output: 5
1
7
2
=== Code Execution Successful ===

Programiz Python Online Compiler

main.py

Share Run

```python
def find_max_in_sorted_list(input_list):
    if not input_list:
        return None  # or appropriate message for empty list
    sorted_list = sorted(input_list)
    return sorted_list[-1]

# Test Cases
# Empty List
assert find_max_in_sorted_list([]) == None

# Single Element List
assert find_max_in_sorted_list([5]) == 5

# All Elements are the Same
assert find_max_in_sorted_list([3, 3, 3, 3, 3]) == 3

```

Output

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

Programiz Python Online Compiler

main.py

Share Run

```python
def get_unique_elements(input_list):
    return list(set(input_list))

# Test Cases
# Some Duplicate Elements
input_list_1 = [3, 7, 3, 5, 2, 5, 9, 2]
print(get_unique_elements(input_list_1))  # Output: [2, 3, 5, 7, 9]

# Negative and Positive Numbers
input_list_2 = [-1, 2, -1, 3, 2, -2]
print(get_unique_elements(input_list_2))  # Output: [-2, -1, 2, 3]

# List with Large Numbers
input_list_3 = [1000000, 999999, 1000000]
print(get_unique_elements(input_list_3))  # Output: [1000000, 999999]

```

Output

```
[2, 3, 5, 7, 9]
[2, 3, -1, -2]
[1000000, 999999]

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

Programiz Python Online Compiler

main.py

Share Run

Output

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Time Complexity Analysis:
# Best Case: O(n)
# Average Case: O(n^2)
# Worst Case: O(n^2)

```

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

Programiz Python Online Compiler

main.py

Share Run

```python
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] < x:
            low = mid + 1
        elif arr[mid] > x:
            high = mid - 1
        else:
            return f"Element {x} is found at position {mid}"
    return f"Element {x} is not found"

# Test Case
arr = [3, 4, 6, -9, 10, 8, 9, 30]
x = 10
print(binary_search(arr, x))

x = 100
print(binary_search(arr, x))

```

Output

```
Element 10 is not found
Element 100 is not found

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

Programiz Python Online Compiler

main.py

Share Run

```
    left = merge_sort(arr[:mid])
    right = merge_sort(arr[mid:])
    return merge(left, right)

def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result.extend(left[i:])
    result.extend(right[j:])
    return result

nums = [12, 11, 13, 5, 6, 7]
sorted_nums = merge_sort(nums)
print(sorted_nums)

```

Output

```
[5, 6, 7, 11, 12, 13]

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