main.py

Share Run

```python
def count_intersections(nums1, nums2):
    answer1 = sum(1 for i in nums1 if i in nums2)
    answer2 = sum(1 for i in nums2 if i in nums1)
    return [answer1, answer2]
nums1 = [2, 3, 2]
nums2 = [1, 2]
output = count_intersections(nums1, nums2)
print(output)
```

Output

```
[2, 1]

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

main.py

Share Run

```python
def first_palindromic_string(words):
    for word in words:
        if word == word[::-1]:
            return word
    return ""
words = ["aba", "car", "ada", "racecar", "coolooc"]
print(first_palindromic_string(words))

```

Output Clear

```
aba

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

main.py

Share Run

Output

```
def get_unique_elements(input_list):
    return list(set(input_list))
input1 = [3, 7, 3, 5, 2, 5, 9, 2]
print(get_unique_elements(input1))
```

```
[2, 3, 5, 7, 9]

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

```
a=[1,2,3,48,6,9]
b=max(a)
print(b)
```

```
48

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

main.py

Share Run

```python
def countPairs(nums, k):
    count = 0
    n = len(nums)
    num_indices = {}

    for i in range(n):
        if nums[i] in num_indices:
            for j in num_indices[nums[i]]:
                if (i * j) % k == 0:
                    count += 1
            num_indices[nums[i]].append(i)
        else:
            num_indices[nums[i]] = [i]
    return count
print(countPairs([3,1,2,2,2,1,3], 2))
print(countPairs([1,2,3,4], 1))

```

Output

```
4
0

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

Programiz

Python Online Compiler

main.py

Share

Run

Output

```python
def sum_of_squares_distinct_counts(nums):
    result = 0
    for i in range(len(nums)):
        for j in range(i, len(nums)):
            distinct_count = len(set(nums[i:j+1]))
            result += distinct_count ** 2
    return result
nums = [1, 2, 1]
output = sum_of_squares_distinct_counts(nums)
print(output)

```

```
15

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

```python
def process_numbers(numbers):
    if not numbers:
        return "The list is empty."
    sorted_numbers = sorted(numbers)
    return sorted_numbers[-1]
print(process_numbers([]))
print(process_numbers([5]))
```

```
The list is empty.
5

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

main.py

Share Run

```python
def binary_search(arr, key):
    left, right = 0, len(arr) - 1
    while left <= right:
        mid = (left + right) // 2
        if arr[mid] == key:
            return f"Element {key} is found at position {mid}"
        elif arr[mid] < key:
            left = mid + 1
        else:
            right = mid - 1
    return f"Element {key} is not found"
arr = sorted([3, 4, 6, -9, 10, 8, 9, 30])
print(binary_search(arr, 10))
print(binary_search(arr, 100))
```

Output

```
Element 10 is found at position 6
Element 100 is not found

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

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
arr = [64, 34, 25, 12, 22, 11, 90]
sorted_arr = bubble_sort(arr)
print("Sorted array:", sorted_arr)
```

```
Sorted array: [11, 12, 22, 25, 34, 64, 90]

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

main.py

Share Run

```python
def quick_sort(nums):
    if len(nums) <= 1:
        return nums
    pivot = nums[len(nums) // 2]
    left = [x for x in nums if x < pivot]
    middle = [x for x in nums if x == pivot]
    right = [x for x in nums if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
nums = [5, 2, 9, 1, 5, 6]
sorted_nums = quick_sort(nums)
print(sorted_nums)

```

Output

```
[1, 2, 5, 5, 6, 9]

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