main.py

Share Run

```python
def process_list(input_list):
    if not input_list:
        return input_list
    if len(input_list) == 1:
        return input_list
    if len(set(input_list)) == 1:
        return input_list
    return sorted(input_list)
print("Test Case 1:", process_list([]))
print("Test Case 2:", process_list([1]))
print("Test Case 3:", process_list([7, 7, 7, 7]))
print("Test Case 4:", process_list([-5, -1, -3, -2, -4]))
```

Output Clear

```
Test Case 1: []
Test Case 2: [1]
Test Case 3: [7, 7, 7, 7]
Test Case 4: [-5, -4, -3, -2, -1]

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

main.py

Share Run

```python
def selection_sort(arr):

    for i in range(len(arr)):

        min_index = i
        for j in range(i + 1, len(arr)):
            if arr[j] < arr[min_index]:
                min_index = j

        arr[i], arr[min_index] = arr[min_index], arr[i]

    return arr

input1 = [5, 2, 9, 1, 5, 6]
output1 = selection_sort(input1)
print(f"Sorted Random Array: {output1}")
```

Output Clear

```
Sorted Random Array: [1, 2, 5, 5, 6, 9]

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

main.py

Share Run

```python
def bubble_sort(arr):
    n = len(arr)

    for i in range(n):

        swapped = False

        for j in range(0, n - i - 1):

            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            break

    return arr

input1 = [5, 2, 9, 1, 5, 6]
output1 = bubble_sort(input1)
print(f"Sorted Random Array: {output1}")
```

Output Clear

```
Sorted Random Array: [1, 2, 5, 5, 6, 9]

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

main.py

Share Run

```python
def bubble_sort(arr):
    n = len(arr)

    for i in range(n):

        swapped = False

        for j in range(0, n - i - 1):

            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            break

    return arr

input1 = [64, 25, 12, 22, 11]
output1 = bubble_sort(input1)
print(f"Sorted Array 1: {output1}")
```

Output

```
Sorted Array 1: [11, 12, 22, 25, 64]

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

main.py

Share Run

```python
def find_kth_missing(arr, k):
    missing_count = 0
    current = 1
    i = 0

    while True:

        if i < len(arr) and arr[i] == current:
            i += 1
        else:
            missing_count += 1

            if missing_count == k:
                return current
        current += 1
arr1 = [2, 3, 4, 7, 11]
k1 = 5
print(f"The {k1}th missing positive number is: {find_kth_missing
    (arr1, k1)}")
```

Output

```
The 5th missing positive number is: 9

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

main.py

Share Run

```python
def find_peak_element(nums):
    left, right = 0, len(nums) - 1

    while left < right:
        mid = (left + right) // 2

        if nums[mid] > nums[mid + 1]:

            right = mid
        else:

            left = mid + 1


    return left

nums1 = [1, 2, 3, 1]
print(f"Peak element index in [1, 2, 3, 1]: {find_peak_element
    (nums1)}")
```

Output

```
Peak element index in [1, 2, 3, 1]: 2

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

main.py

Share Run

Output

```python
def strStr(haystack, needle):

    if not needle:
        return 0
    len_haystack, len_needle = len(haystack), len(needle)

    for i in range(len_haystack - len_needle + 1):

        if haystack[i:i + len_needle] == needle:
            return i

    return -1

haystack1 = "sadbutsad"
needle1 = "sad"
print(f"Index of first occurrence of '{needle1}' in '{haystack1}'
    : {strStr(haystack1, needle1)}")
```

```
Index of first occurrence of 'sad' in 'sadbutsad': 0

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

main.py

Share Run

```python
def stringMatching(words):
    result = []
    for i in range(len(words)):
        for j in range(len(words)):
            if i != j and words[i] in words[j]:
                result.append(words[i])
                break
    return result

words1 = ["mass", "as", "hero", "superhero"]
print(f"Substrings in {words1}: {stringMatching(words1)}")  #
```

Output Clear

```
Substrings in ['mass', 'as', 'hero', 'superhero']: ['as', 'hero']

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