main.py

Share Run

Output Clear

```python
def process_list(input_list):
    if not input_list:
        return []
    elif len(input_list) == 1:
        return input_list
    elif all(x == input_list[0] for x in input_list):
        return input_list
    else:
        return sorted(input_list)
print(process_list([]))
print(process_list([1]))
print(process_list([7, 7, 7, 7]))
print(process_list([-5, -1, -3, -2, -4]))
```

```
[]
[1]
[7, 7, 7, 7]
[-5, -4, -3, -2, -1]

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

main.py

Share Run

```python
def find_kth_missing(arr, k):
    missing_count = 0
    current_num = 1
    idx = 0
    while missing_count < k:
        if idx < len(arr) and arr[idx] == current_num:
            idx += 1
        else:
            missing_count += 1
            if missing_count == k:
                return current_num
        current_num += 1
    return current_num
print(find_kth_missing([2, 3, 4, 7, 11], 5))
print(find_kth_missing([1, 2, 3, 4], 2))

```

Output Clear

```
9
6

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

main.py

Share Run

```python
def strStr(haystack, needle):
    return haystack.find(needle)
print(strStr("sadbutsad", "sad"))
print(strStr("leetcode", "leeto"))

```

Output Clear

```
0
-1

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

main.py

Share Run

Output Clear

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_index = i
        for j in range(i + 1, n):
            if arr[j] < arr[min_index]:
                min_index = j
        arr[i], arr[min_index] = arr[min_index], arr[i]
    return arr
print(selection_sort([5, 2, 9, 1, 5, 6]))
print(selection_sort([10, 8, 6, 4, 2]))
print(selection_sort([1, 2, 3, 4, 5]))

```

```
[1, 2, 5, 5, 6, 9]
[2, 4, 6, 8, 10]
[1, 2, 3, 4, 5]

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

main.py

Share Run

```python
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr
print(insertion_sort([3, 1, 4, 1, 5, 9, 2, 6, 5, 3]))
print(insertion_sort([5, 5, 5, 5, 5]))
print(insertion_sort([2, 3, 1, 3, 2, 1, 1, 3]))
```

Output

```
[1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
[5, 5, 5, 5, 5]
[1, 1, 1, 2, 2, 3, 3, 3]

=== 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
print(bubble_sort([5, 2, 9, 1, 5, 6]))
print(bubble_sort([10, 8, 6, 4, 2]))
print(bubble_sort([1, 2, 3, 4, 5]))

```

Output Clear

```
[1, 2, 5, 5, 6, 9]
[2, 4, 6, 8, 10]
[1, 2, 3, 4, 5]

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

main.py

Share Run

Output Clear

```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
print(bubble_sort([64, 25, 12, 22, 11]))
print(bubble_sort([29, 10, 14, 37, 13]))
print(bubble_sort([3, 5, 2, 1, 4]))
print(bubble_sort([1, 2, 3, 4, 5]))

```

```
[11, 12, 22, 25, 64]
[10, 13, 14, 29, 37]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

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