

main.py



Share

Run

Output

Clear

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

```
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

Output

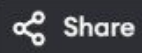
Clear

```
1 def selection_sort(arr):
2
3     for i in range(len(arr)):
4
5         min_index = i
6         for j in range(i + 1, len(arr)):
7             if arr[j] < arr[min_index]:
8                 min_index = j
9
10        arr[i], arr[min_index] = arr[min_index], arr[i]
11
12    return arr
13
14 input1 = [5, 2, 9, 1, 5, 6]
15 output1 = selection_sort(input1)
16 print(f"Sorted Random Array: {output1}")
```

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

=== Code Execution Successful ===

main.py



Share

Run

Output

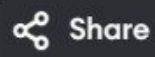
Clear

```
1 def bubble_sort(arr):
2     n = len(arr)
3
4     for i in range(n):
5
6         swapped = False
7
8         for j in range(0, n - i - 1):
9
10            if arr[j] > arr[j + 1]:
11                arr[j], arr[j + 1] = arr[j + 1], arr[j]
12                swapped = True
13
14            if not swapped:
15                break
16
17     return arr
18
19 input1 = [5, 2, 9, 1, 5, 6]
20 output1 = bubble_sort(input1)
21 print(f"Sorted Random Array: {output1}")
```

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

=== Code Execution Successful ===

main.py



Run

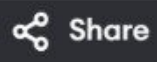
Output

```
1 def bubble_sort(arr):
2     n = len(arr)
3
4     for i in range(n):
5
6         swapped = False
7
8         for j in range(0, n - i - 1):
9
10            if arr[j] > arr[j + 1]:
11                arr[j], arr[j + 1] = arr[j + 1], arr[j]
12                swapped = True
13
14            if not swapped:
15                break
16
17     return arr
18
19 input1 = [64, 25, 12, 22, 11]
20 output1 = bubble_sort(input1)
21 print(f"Sorted Array 1: {output1}")
```

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

=== Code Execution Successful ===

main.py



Run

Output

```
1 def find_kth_missing(arr, k):
2     missing_count = 0
3     current = 1
4     i = 0
5
6     while True:
7
8         if i < len(arr) and arr[i] == current:
9             i += 1
10        else:
11            missing_count += 1
12
13            if missing_count == k:
14                return current
15            current += 1
16 arr1 = [2, 3, 4, 7, 11]
17 k1 = 5
18 print(f"The {k1}th missing positive number is: {find_kth_missing
    (arr1, k1)}")
```

The 5th missing positive number is: 9

=== Code Execution Successful ===

main.py



Share

Run

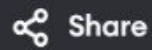
Output

```
1 def find_peak_element(nums):
2     left, right = 0, len(nums) - 1
3
4     while left < right:
5         mid = (left + right) // 2
6
7         if nums[mid] > nums[mid + 1]:
8             right = mid
9         else:
10            left = mid + 1
11
12
13
14
15     return left
16
17 nums1 = [1, 2, 3, 1]
18 print(f"Peak element index in [1, 2, 3, 1]: {find_peak_element(
    nums1)}")
```

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

=== Code Execution Successful ===

main.py



Run

Output

```
1 def strStr(haystack, needle):
2
3     if not needle:
4         return 0
5     len_haystack, len_needle = len(haystack), len(needle)
6
7     for i in range(len_haystack - len_needle + 1):
8
9         if haystack[i:i + len_needle] == needle:
10             return i
11
12     return -1
13
14 haystack1 = "sadbutsad"
15 needle1 = "sad"
16 print(f"Index of first occurrence of '{needle1}' in '{haystack1}'
17       : {strStr(haystack1, needle1)}")
18
```

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

=== Code Execution Successful ===

main.py



Share

Run

Output

Clear

```
1 def stringMatching(words):
2     result = []
3     for i in range(len(words)):
4         for j in range(len(words)):
5             if i != j and words[i] in words[j]:
6                 result.append(words[i])
7                 break
8     return result
9
10 words1 = ["mass", "as", "hero", "superhero"]
11 print(f"Substrings in {words1}: {stringMatching(words1)}") #
```

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

=== Code Execution Successful ===



Search



ENG
IN



12:59
09-10-2024