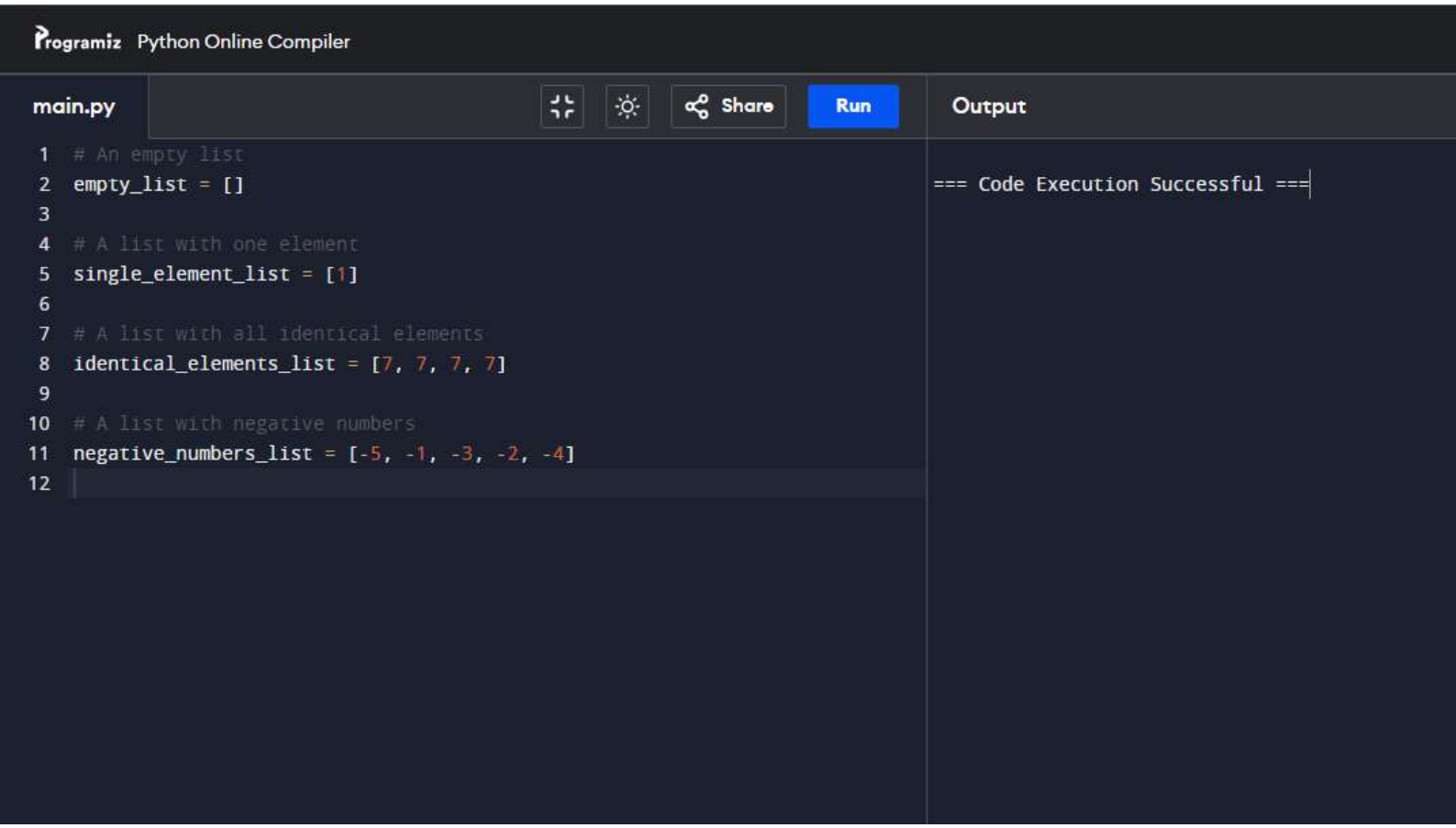

Programiz Python Online Compiler
main.py
Share
Run
Output
# An empty list
empty_list = []
# A list with one element
single_element_list = [1]
# A list with all identical elements
identical_elements_list = [7, 7, 7, 7]
# A list with negative numbers
negative_numbers_list = [-5, -1, -3, -2, -4]
=== Code Execution Successful ===

Programiz Python Online Compiler

main.py

Share Run

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

# Sorting a Random Array
random_array = [5, 2, 9, 1, 5, 6]
print("Input Random Array:", random_array)
print("Sorted Random Array:", selection_sort(random_array))

# Sorting a Reverse Sorted Array
reverse_sorted_array = [10, 8, 6, 4, 2]
print("\nInput Reverse Sorted Array:", reverse_sorted_array)
print("Sorted Reverse Sorted Array:", selection_sort
    (reverse_sorted_array))

# Sorting an Already Sorted Array
already_sorted_array = [1, 2, 3, 4, 5]

```

Output

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

Input Reverse Sorted Array: [10, 8, 6, 4, 2]
Sorted Reverse Sorted Array: [2, 4, 6, 8, 10]

Input Already Sorted Array: [1, 2, 3, 4, 5]
Sorted Already Sorted Array: [1, 2, 3, 4, 5]

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

Programiz Python Online Compiler

main.py

Share Run

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

```

Output

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

main.py

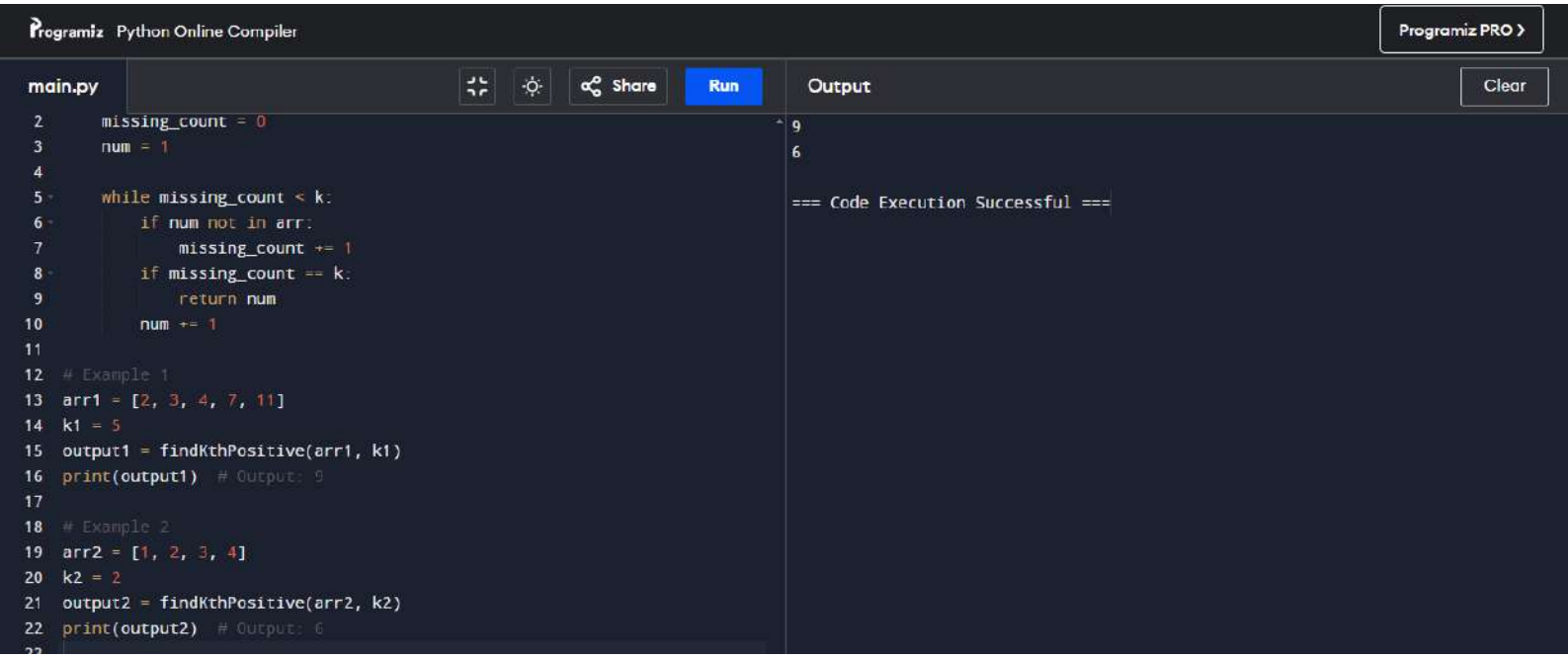
Share Run

```
        key = arr[i]
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

# Test Cases
test_cases = [[64, 25, 12, 22, 11],
              [29, 10, 14, 37, 13],
              [3, 5, 2, 1, 4],
              [1, 2, 3, 4, 5],
              [5, 4, 3, 2, 1],
              [3, 1, 4, 1, 5, 9, 2, 6, 5, 3],
              [5, 5, 5, 5, 5],
              [2, 3, 1, 3, 2, 1, 1, 3]]

for case in test_cases:
    sorted_arr = insertion_sort_with_duplicates(case)
    print(f"Input: {case} -> Output: {sorted_arr}")
```

Output

Clear

```
Input: [11, 12, 22, 25, 64] -> Output: [11, 12, 22, 25, 64]
Input: [10, 13, 14, 29, 37] -> Output: [10, 13, 14, 29, 37]
Input: [1, 2, 3, 4, 5] -> Output: [1, 2, 3, 4, 5]
Input: [1, 2, 3, 4, 5] -> Output: [1, 2, 3, 4, 5]
Input: [1, 2, 3, 4, 5] -> Output: [1, 2, 3, 4, 5]
Input: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9] -> Output: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
Input: [5, 5, 5, 5, 5] -> Output: [5, 5, 5, 5, 5]
Input: [1, 1, 1, 2, 2, 3, 3, 3] -> Output: [1, 1, 1, 2, 2, 3, 3, 3]

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


Programiz Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
missing_count = 0
num = 1
while missing_count < k:
if num not in arr:
missing_count += 1
if missing_count == k:
return num
num += 1
# Example 1
arr1 = [2, 3, 4, 7, 11]
k1 = 5
output1 = findKthPositive(arr1, k1)
print(output1) # Output: 9
# Example 2
arr2 = [1, 2, 3, 4]
k2 = 2
output2 = findKthPositive(arr2, k2)
print(output2) # Output: 6
9
6
=== Code Execution Successful ===

Programiz Python Online Compiler

main.py

Share Run

```python
def find_peak_element(nums):
    left, right = 0, len(nums) - 1
    while left < right:
        mid = left + (right - left) // 2
        if nums[mid] < nums[mid + 1]:
            left = mid + 1
        else:
            right = mid
    return left

```

Output

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

Programiz Python Online Compiler

main.py | Share | Run | Output

```python
def strStr(haystack, needle):
    if needle in haystack:
        return haystack.index(needle)
    else:
        return -1

# Example 1
haystack1 = "sadbutsad"
needle1 = "sad"
print(strStr(haystack1, needle1))  # Output: 0

# Example 2
haystack2 = "leetcode"
needle2 = "leeto"
print(strStr(haystack2, needle2))  # Output: -1

```

```
0
-1

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

Programiz Python Online Compiler

main.py

Share Run

```python
def string_substrings(words):
    substrings = []
    for word in words:
        for other_word in words:
            if word != other_word and word in other_word:
                substrings.append(word)
                break
    return list(set(substrings))

# Test the function with examples
words1 = ["mass", "as", "hero", "superhero"]
print(string_substrings(words1))  # Output: ['as', 'hero']

words2 = ["leetcode", "et", "code"]
print(string_substrings(words2))  # Output: ['et', 'code']

words3 = ["blue", "green", "bu"]
print(string_substrings(words3))  # Output: []

```

Output

```
['as', 'hero']
['et', 'code']
[]

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