

1. Merge Two Sorted List

The screenshot shows a Python IDE with two windows. The left window, titled 'Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)', contains the following code:

```
tail = dummy

while list1 and list2:
    if list1.val < list2.val:
        tail.next = list1
        list1 = list1.next
    else:
        tail.next = list2
        list2 = list2.next
    tail = tail.next

if list1:
    tail.next = list1
else:
    tail.next = list2

return dummy.next

# Helper function to print linked list
def printLinkedList(head):
    while head:
        print(head.val, end=" -> ")
        head = head.next
    print("None")

# Helper function to create linked list from a list
def createLinkedList(lst):
    dummy = ListNode()
    current = dummy
    for value in lst:
        current.next = ListNode(value)
        current = current.next
    return dummy.next

# Test the function
list1 = createLinkedList([1, 2, 4])
list2 = createLinkedList([1, 3, 4])
mergedList = mergeTwoLists(list1, list2)
printLinkedList(mergedList)
```

The right window, titled 'IDLE Shell 3.12.1', shows the output of the program:

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
1 -> 1 -> 2 -> 3 -> 4 -> 4 -> None
>>>
```

2. Merge k Sorted Lists

The screenshot shows a Python IDE with two windows. The left window, titled 'Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)*', contains the following code:

```
import heapq

class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

def mergeKLists(lists):
    min_heap = []
    for idx, node in enumerate(lists):
        if node:
            heapq.heappush(min_heap, (node.val, idx, node))
    dummy = ListNode()
    current = dummy
    while min_heap:
        val, idx, node = heapq.heappop(min_heap)
        current.next = ListNode(val)
        current = current.next
        if node.next:
            heapq.heappush(min_heap, (node.next.val, idx, node.next))
    return dummy.next

def printLinkedList(head):
    while head:
        print(head.val, end=" -> ")
        head = head.next
    print("None")

list1 = ListNode(1, ListNode(4, ListNode(5)))
list2 = ListNode(1, ListNode(3, ListNode(4)))
list3 = ListNode(2, ListNode(6))

lists = [list1, list2, list3]
merged_list = mergeKLists(lists)
printLinkedList(merged_list)
```

The right window, titled 'IDLE Shell 3.12.1', shows the output of the program:

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
1 -> 1 -> 2 -> 3 -> 4 -> 4 -> 5 -> 6 -> None
>>>
```

3. Remove Duplicates from Sorted Array

The screenshot shows a Python IDE with two windows. The left window, titled 'Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)*', contains the following code:

```
def remove_duplicates(nums):
    if not nums:
        return 0

    # Initialize the slow pointer
    slow = 0

    for fast in range(1, len(nums)):
        if nums[fast] != nums[slow]:
            # Move the slow pointer one step forward
            slow += 1
            # Update the value at slow pointer to the current unique element
            nums[slow] = nums[fast]

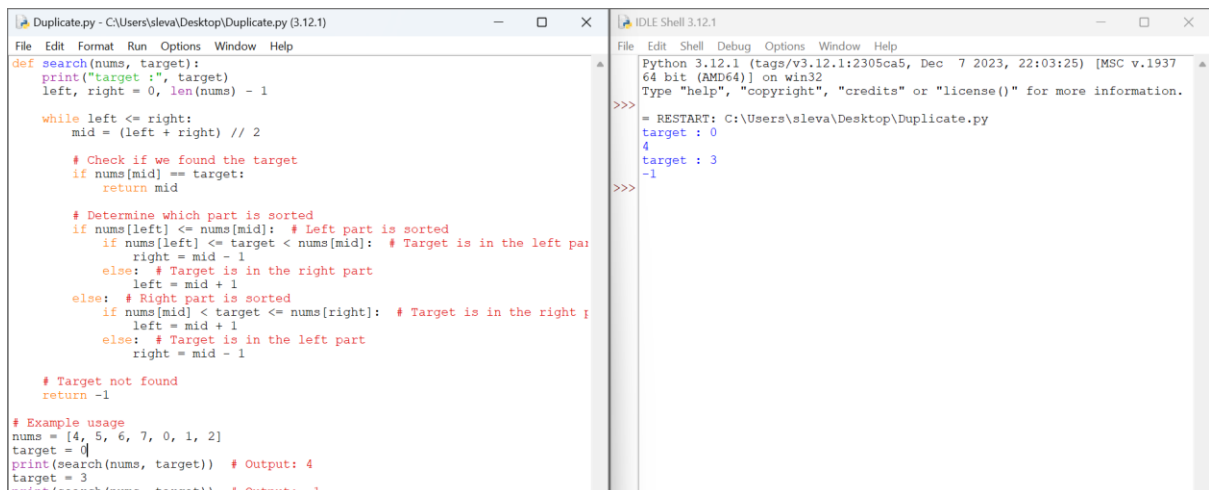
    # The length of the array with unique elements
    return slow + 1

# Example usage
nums = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]
k = remove_duplicates(nums)
print(f"k: {k}") # Output should be the length of unique elements
print(f"nums: {nums[:k]}") # Output should be the unique elements in the array
```

The right window, titled 'IDLE Shell 3.12.1', shows the output of the program:

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
k: 5
nums: [0, 1, 2, 3, 4]
>>>
```

4. Search in Rotated Sorted Array



```
File Edit Format Run Options Window Help
def search(nums, target):
    print("target :", target)
    left, right = 0, len(nums) - 1

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

        # Check if we found the target
        if nums[mid] == target:
            return mid

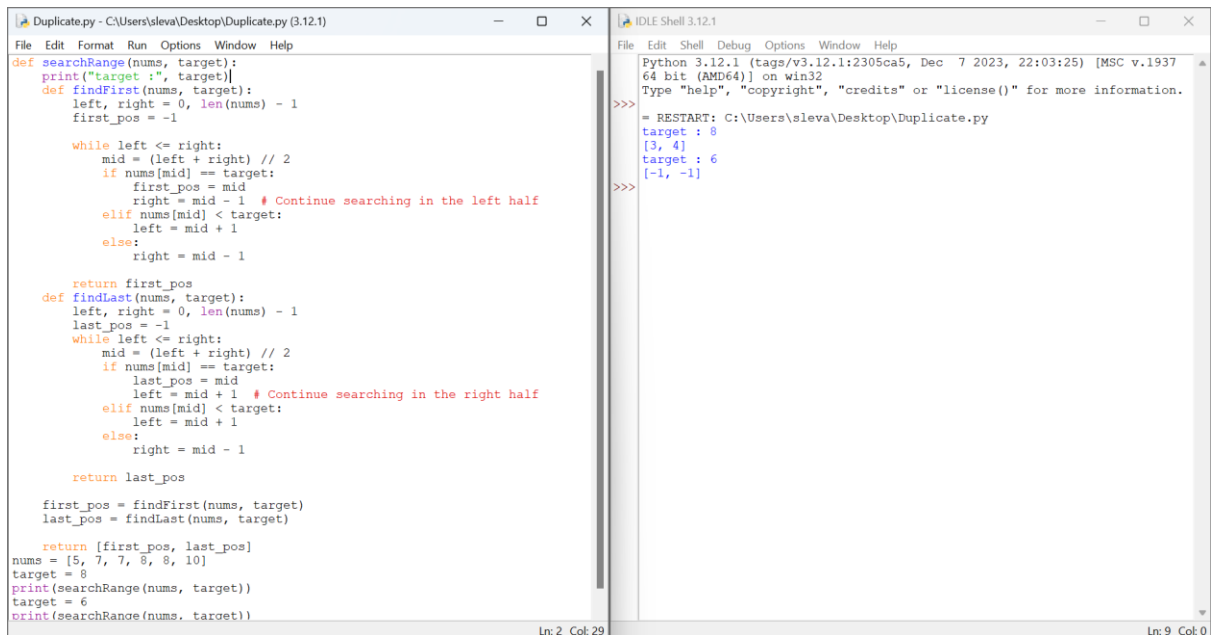
        # Determine which part is sorted
        if nums[left] <= nums[mid]: # Left part is sorted
            if nums[left] <= target < nums[mid]: # Target is in the left part
                right = mid - 1
            else: # Target is in the right part
                left = mid + 1
        else: # Right part is sorted
            if nums[mid] < target <= nums[right]: # Target is in the right part
                left = mid + 1
            else: # Target is in the left part
                right = mid - 1

    # Target not found
    return -1

# Example usage
nums = [4, 5, 6, 7, 0, 1, 2]
target = 0
print(search(nums, target)) # Output: 4
target = 3
print(search(nums, target)) # Output: -1
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sleval\Desktop\Duplicate.py
target : 0
4
target : 3
-1
>>>
```

5. Find First and Last Position of Element in Sorted Array



```
File Edit Format Run Options Window Help
def searchRange(nums, target):
    print("target :", target)
    def findFirst(nums, target):
        left, right = 0, len(nums) - 1
        first_pos = -1

        while left <= right:
            mid = (left + right) // 2
            if nums[mid] == target:
                first_pos = mid
                right = mid - 1 # Continue searching in the left half
            elif nums[mid] < target:
                left = mid + 1
            else:
                right = mid - 1

        return first_pos

    def findLast(nums, target):
        left, right = 0, len(nums) - 1
        last_pos = -1

        while left <= right:
            mid = (left + right) // 2
            if nums[mid] == target:
                last_pos = mid
                left = mid + 1 # Continue searching in the right half
            elif nums[mid] < target:
                left = mid + 1
            else:
                right = mid - 1

        return last_pos

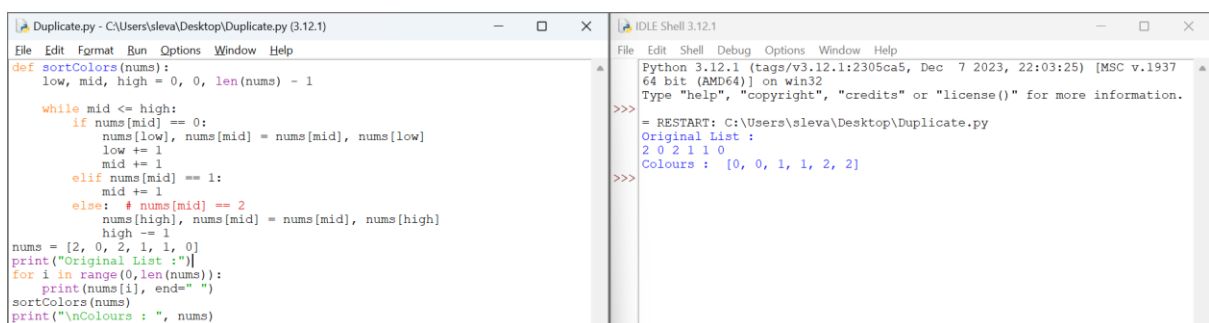
    first_pos = findFirst(nums, target)
    last_pos = findLast(nums, target)

    return [first_pos, last_pos]

nums = [5, 7, 7, 8, 8, 10]
target = 8
print(searchRange(nums, target))
print(searchRange(nums, target))
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sleval\Desktop\Duplicate.py
target : 8
[3, 4]
target : 6
[-1, -1]
>>>
```

6. Sort Colors



```
File Edit Format Run Options Window Help
def sortColors(nums):
    low, mid, high = 0, 0, len(nums) - 1

    while mid <= high:
        if nums[mid] == 0:
            nums[low], nums[mid] = nums[mid], nums[low]
            low += 1
            mid += 1
        elif nums[mid] == 1:
            mid += 1
        else: # nums[mid] == 2
            nums[high], nums[mid] = nums[mid], nums[high]
            high -= 1

    nums = [2, 0, 2, 1, 1, 0]
    print("Original List :")
    for i in range(0, len(nums)):
        print(nums[i], end=" ")
    sortColors(nums)
    print("\nColours : ", nums)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:\Users\sleval\Desktop\Duplicate.py
Original List :
2 0 2 1 1 0
Colours : [0, 0, 1, 1, 2, 2]
>>>
```

7. Remove Duplicates from Sorted List

```
Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
def __init__(self, val=0, next=None):
    self.val = val
    self.next = next

class Solution:
    def deleteDuplicates(self, head: ListNode) -> ListNode:
        current = head

        while current and current.next:
            if current.val == current.next.val:
                # Skip the next node since it's a duplicate
                current.next = current.next.next
            else:
                # Move to the next distinct element
                current = current.next

        return head

def list_to_linkedlist(arr):
    if not arr:
        return None
    head = ListNode(arr[0])
    current = head
    for value in arr[1:]:
        current.next = ListNode(value)
        current = current.next
    return head

def linkedlist_to_list(head):
    arr = []
    current = head
    while current:
        arr.append(current.val)
        current = current.next
    return arr

# Example usage
head = list_to_linkedlist([1, 1, 2])
solution = Solution()
new_head = solution.deleteDuplicates(head)
print(linkedlist_to_list(new_head)) # Output: [1, 2]

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
[1, 2]
>>>
```

8. Merge Sorted Array

```
Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
def merge(nums1, m, nums2, n):
    # Start from the end of nums1 and nums2
    p1 = m - 1 # Pointer for the last element in the first part of nums1
    p2 = n - 1 # Pointer for the last element in nums2
    p = m + n - 1 # Pointer for the last position in nums1

    # While there are elements to be checked in nums2
    while p2 >= 0:
        if p1 >= 0 and nums1[p1] > nums2[p2]:
            nums1[p] = nums1[p1]
            p1 -= 1
        else:
            nums1[p] = nums2[p2]
            p2 -= 1
        p -= 1

# Example usage
nums1 = [1, 2, 3, 0, 0, 0]
m = 3
nums2 = [2, 5, 6]
n = 3
merge(nums1, m, nums2, n)
print(nums1) # Output: [1, 2, 2, 3, 5, 6]

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
[1, 2, 2, 3, 5, 6]
>>>
```

9. Convert Sorted Array to Binary Search Tree

```
Duplicate.py - C:\Users\sleval\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
from typing import List, Optional

class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

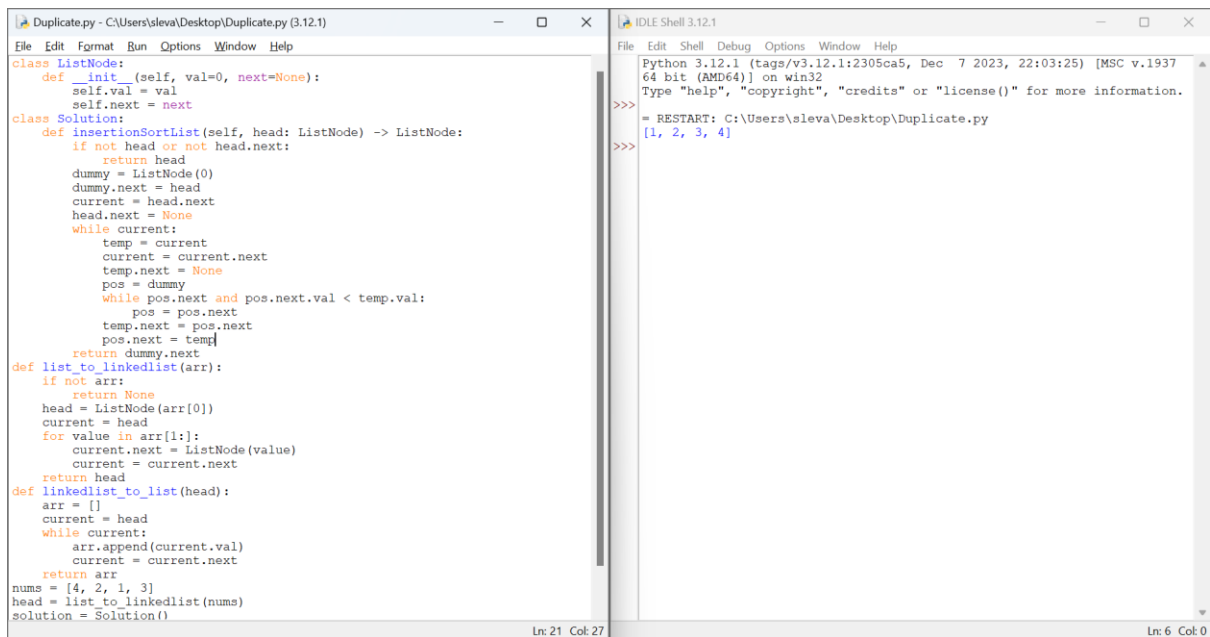
def sortedArrayToBST(nums: List[int]) -> Optional[TreeNode]:
    if not nums:
        return None
    def convertListToBST(left, right):
        if left > right:
            return None
        mid = (left + right) // 2
        node = TreeNode(nums[mid])
        node.left = convertListToBST(left, mid - 1)
        node.right = convertListToBST(mid + 1, right)
        return node
    return convertListToBST(0, len(nums) - 1)

from collections import deque
def printTree(root: TreeNode):
    if not root:
        return []
    result = []
    queue = deque([root])
    while queue:
        node = queue.popleft()
        if node:
            result.append(node.val)
            queue.append(node.left)
            queue.append(node.right)
        else:
            result.append(None)
    while result and result[-1] is None:
        result.pop()
    return result

nums = [-10, -3, 0, 5, 9]
root = sortedArrayToBST(nums)
print(printTree(root)) # Output: [0, -10, 5, None, -3, None, 9]

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleval\Desktop\Duplicate.py
[0, -10, 5, None, -3, None, 9]
>>>
```

10. Insertion Sort List



```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

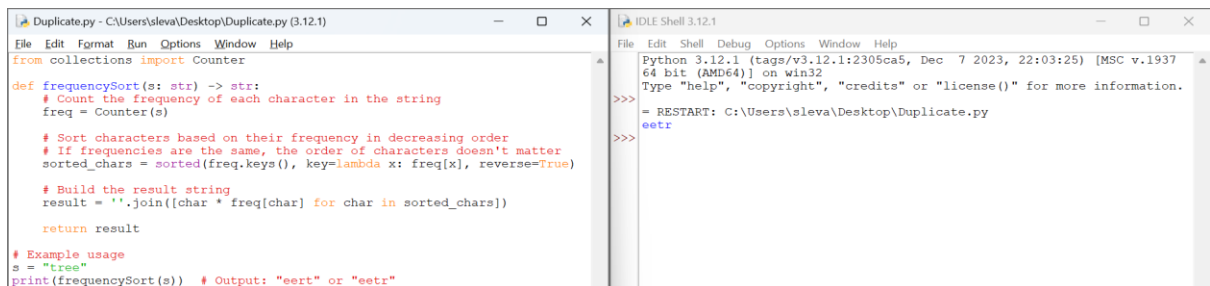
class Solution:
    def insertionSortList(self, head: ListNode) -> ListNode:
        if not head or not head.next:
            return head
        dummy = ListNode(0)
        dummy.next = head
        current = head.next
        head.next = None
        while current:
            temp = current
            current = current.next
            temp.next = None
            pos = dummy
            while pos.next and pos.next.val < temp.val:
                pos = pos.next
            temp.next = pos.next
            pos.next = temp
        return dummy.next

    def list_to_linkedlist(self, arr):
        if not arr:
            return None
        head = ListNode(arr[0])
        current = head
        for value in arr[1:]:
            current.next = ListNode(value)
            current = current.next
        return head

    def linkedlist_to_list(self, head):
        arr = []
        current = head
        while current:
            arr.append(current.val)
            current = current.next
        return arr

nums = [4, 2, 1, 3]
head = list_to_linkedlist(nums)
solution = Solution()
solution.insertionSortList(head)
```

11. Sort Characters By Frequency



```
from collections import Counter

def frequencySort(s: str) -> str:
    # Count the frequency of each character in the string
    freq = Counter(s)

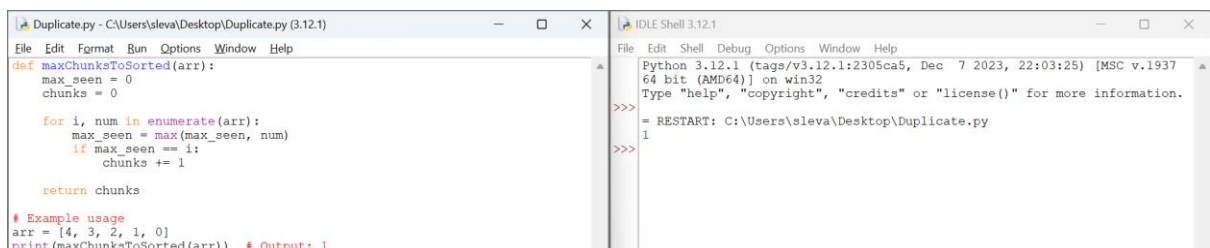
    # Sort characters based on their frequency in decreasing order
    # If frequencies are the same, the order of characters doesn't matter
    sorted_chars = sorted(freq.keys(), key=lambda x: freq[x], reverse=True)

    # Build the result string
    result = ''.join([char * freq[char] for char in sorted_chars])

    return result

# Example usage
s = "tree"
print(frequencySort(s)) # Output: "eert" or "eetr"
```

12. Max Chunks To Make Sorted



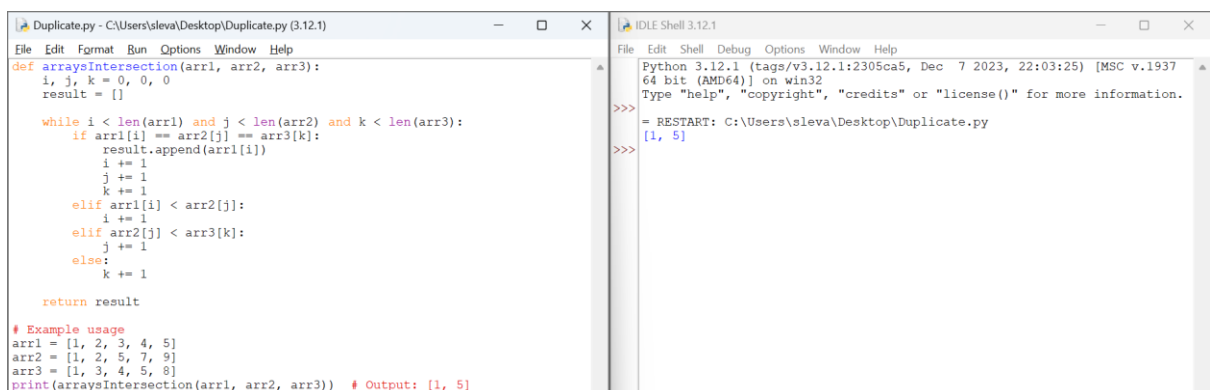
```
def maxChunksToSorted(arr):
    max_seen = 0
    chunks = 0

    for i, num in enumerate(arr):
        max_seen = max(max_seen, num)
        if max_seen == i:
            chunks += 1

    return chunks

# Example usage
arr = [4, 3, 2, 1, 0]
print(maxChunksToSorted(arr)) # Output: 1
```

13. Intersection of Three Sorted Arrays



```
def arraysIntersection(arr1, arr2, arr3):
    i, j, k = 0, 0, 0
    result = []

    while i < len(arr1) and j < len(arr2) and k < len(arr3):
        if arr1[i] == arr2[j] == arr3[k]:
            result.append(arr1[i])
            i += 1
            j += 1
            k += 1
        elif arr1[i] < arr2[j]:
            i += 1
        elif arr2[j] < arr3[k]:
            j += 1
        else:
            k += 1

    return result

# Example usage
arr1 = [1, 2, 3, 4, 5]
arr2 = [1, 2, 5, 7, 9]
arr3 = [1, 3, 4, 5, 8]
print(arraysIntersection(arr1, arr2, arr3)) # Output: [1, 5]
```