1. First palindrome in a list

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def first_pal_str(words):
    for i in words:
        if i==i[::-1]:
            return i
    return ""
print(first_pal_str(["fvrb","racecar","dcuciequc"]))
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
racecar
>>>
```

2. Count Indices

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def count_indices(nums1, nums2):
    # Convert nums2 to a set for efficient lookup
    set_nums2 = set(nums2)

    # Calculate answer1
    answer1 = sum(1 for num in nums1 if num in set_nums2)

    # Convert nums1 to a set for efficient lookup
    set_nums1 = set(nums1)

    # Calculate answer2
    answer2 = sum(1 for num in nums2 if num in set_nums1)

    return [answer1, answer2]

# Example usage:
nums1 = [2, 3, 2]
nums2 = [1, 2]
result = count_indices(nums1, nums2)
print(result)  # Output: [2, 1]
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
[2, 1]
>>>
```

3. Sum of Square of distance counts

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def sum_of_squares_of_distinct_counts(nums):
    n = len(nums)
    sum_squares = 0

    for i in range(n):
        distinct_count = {}
        for j in range(i, n):
            if nums[j] in distinct_count:
                distinct_count[nums[j]] += 1
            else:
                distinct_count[nums[j]] = 1

            # Calculate the number of distinct elements
            num_distinct = len(distinct_count)

            # Add the square of the number of distinct elements to the sum
            sum_squares += num_distinct ** 2

    return sum_squares

# Example usage:
nums = [1, 2, 1]
result = sum_of_squares_of_distinct_counts(nums)
print(result)  # Output: 15
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
15
>>>
```

4. Count pairs

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def count_pairs(nums, k):
    from collections import defaultdict

    index_dict = defaultdict(list)
    n = len(nums)
    pair_count = 0

    # Fill the dictionary with indices of each value
    for i in range(n):
        index_dict[nums[i]].append(i)

    # Iterate through the dictionary and check pairs
    for indices in index_dict.values():
        length = len(indices)
        for i in range(length):
            for j in range(i + 1, length):
                if (indices[i] * indices[j]) % k == 0:
                    pair_count += 1

    return pair_count

# Example usage:
nums = [3, 1, 2, 2, 2, 1, 3]
k = 2
result = count_pairs(nums, k)
print(result)  # Output: 4
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
4
>>>
```

5. Max element

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def find_max_element(nums):
    if not nums:
        return None

    max_value = float('-inf')  # Initialize with negative infinity

    for num in nums:
        if num > max_value:
            max_value = num

    return max_value

# Test cases
test_cases = [
    [1, 2, 3, 4, 5],  # Expected Output: 5
    [7, 7, 7, 7, 7],  # Expected Output: 7
    [-10, 2, 3, -4, 5]  # Expected Output: 5
]

for nums in test_cases:
    print(f"Input: {nums}, Output: {find_max_element(nums)}")
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py =============
Input: [1, 2, 3, 4, 5], Output: 5
Input: [7, 7, 7, 7, 7], Output: 7
Input: [-10, 2, 3, -4, 5], Output: 5
>>>
```

6. Max-sort after Sort

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def find_max_after_sort(nums):
    if not nums:
        return None  # Return None for empty list

    sorted_nums = sorted(nums)  # Sort the list

    return sorted_nums[-1]  # Return the last element (maximum)

# Test cases
test_cases = [
    [],                 # Expected Output: None or appropriate message
    [5],                # Expected Output: 5
    [3, 3, 3, 3, 3],    # Expected Output: 3
    [10, 5, 8, 3, 7],   # Expected Output: 10
]

for nums in test_cases:
    print(f"Input: {nums}, Output: {find_max_after_sort(nums)}")
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
Input: [], Output: None
Input: [5], Output: 5
Input: [3, 3, 3, 3, 3], Output: 3
Input: [10, 5, 8, 3, 7], Output: 10
>>>
```

7. Unique element

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def unique_elements(input_list):
    seen = set()
    unique_list = []

    for num in input_list:
        if num not in seen:
            seen.add(num)
            unique_list.append(num)

    return unique_list

# Test cases
test_cases = [
    [3, 7, 3, 5, 2, 5, 9, 2],      # Expected Output: [3, 7, 5, 2, 9] (Order p
    [-1, 2, -1, 3, 2, -2],         # Expected Output: [-1, 2, 3, -2] (Order ma
    [1000000, 999999, 1000000]     # Expected Output: [1000000, 999999]
]

for nums in test_cases:
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
Input: [3, 7, 3, 5, 2, 5, 9, 2], Output: [3, 7, 5, 2, 9]
Input: [-1, 2, -1, 3, 2, -2], Output: [-1, 2, 3, -2]
Input: [1000000, 999999, 1000000], Output: [1000000, 999999]
>>>
```

8. Bubble Sort

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

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

    # Traverse through all array elements
    for i in range(n):
        swapped = False

        # Last i elements are already in place
        for j in range(0, n-i-1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True

        # If no elements were swapped in the inner loop, then break
        if not swapped:
            break

    return arr

# Example usage:
arr = [64, 34, 25, 12, 22, 11, 90]
print("Original array:", arr)
sorted_arr = bubble_sort(arr)
print("Sorted array:", sorted_arr)
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
Original array: [64, 34, 25, 12, 22, 11, 90]
Sorted array: [11, 12, 22, 25, 34, 64, 90]
>>>
```

9. Binary Search

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def binary_search(arr, x):
    left, right = 0, len(arr) - 1

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

        # Check if x is present at mid
        if arr[mid] == x:
            return f"Element {x} is found at position {mid}"
        elif arr[mid] < x:
            left = mid + 1  # x is in the right half
        else:
            right = mid - 1  # x is in the left half

    return f"Element {x} is not found in the array"

# Test case
arr = [3, 4, 6, -9, 10, 8, 9, 30]
x = -9
result = binary_search(arr, x)
print(result)  # Output: Element 10 is found at position 4
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
Element -9 is found at position 3
>>>
```

10. Sort in ascending order

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def merge_sort(nums):
    if len(nums) <= 1:
        return nums

    # Divide the array into two halves
    mid = len(nums) // 2
    left_half = nums[:mid]
    right_half = nums[mid:]

    # Recursively sort each half
    left_sorted = merge_sort(left_half)
    right_sorted = merge_sort(right_half)

    # Merge the sorted halves
    sorted_nums = merge(left_sorted, right_sorted)

    return sorted_nums

def merge(left, right):
    result = []
    i = j = 0

    # Merge the two sorted arrays
    while i < len(left) and j < len(right):
        if left[i] <= right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1

    # Append remaining elements
    result.extend(left[i:])
    result.extend(right[j:])

    return result

# Test case
nums = [3, 7, 1, 4, 2, 8, 5]
sorted_nums = merge_sort(nums)
print("Sorted array:", sorted_nums)
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
Sorted array: [1, 2, 3, 4, 5, 7, 8]
>>>
```

11. Find paths

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def findPaths(m, n, N, i, j):
    # Directions array for moving in 4 possible directions (up, down, left, r
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]

    # 3D DP array to store the number of ways to move out of the grid
    dp = [[[0 for _ in range(n)] for _ in range(m)] for _ in range(N + 1)]

    # Initialize the base cases
    for step in range(1, N + 1):
        for row in range(m):
            for col in range(n):
                for direction in directions:
                    newRow, newCol = row + direction[0], col + direction[1]
                    if newRow < 0 or newRow >= m or newCol < 0 or newCol >= r
                        dp[step][row][col] += 1
                    else:
                        dp[step][row][col] += dp[step - 1][newRow][newCol]

    # Return the result stored in dp[N][i][j]
    return dp[N][i][j]

# Test cases
print(findPaths(2, 2, 2, 0, 0))  # Output: 6
print(findPaths(1, 3, 3, 0, 1))  # Output: 12
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py ========
========
6
12
>>>
```

12. Rob-linear

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def rob_linear(nums):
    if not nums:
        return 0
    if len(nums) == 1:
        return nums[0]

    dp = [0] * len(nums)
    dp[0] = nums[0]
    dp[1] = max(nums[0], nums[1])

    for i in range(2, len(nums)):
        dp[i] = max(dp[i-1], dp[i-2] + nums[i])

    return dp[-1]

def rob(nums):
    if not nums:
        return 0
    if len(nums) == 1:
        return nums[0]

    # Scenario 1: Exclude the last house
    max1 = rob_linear(nums[:-1])

    # Scenario 2: Exclude the first house
    max2 = rob_linear(nums[1:])

    # Return the maximum of both scenarios
    return max(max1, max2)

# Test cases
print(rob([2, 3, 2]))  # Output: 3
print(rob([1, 2, 3, 1]))  # Output: 4
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
3
4
>>>
```

13. Climb Staircase

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def climb_stairs(n):
    if n == 1:
        return 1
    if n == 2:
        return 2

    prev2 = 1
    prev1 = 2

    for i in range(3, n + 1):
        current = prev1 + prev2
        prev2 = prev1
        prev1 = current

    return prev1

# Test cases
print(climb_stairs(4))  # Output: 5
print(climb_stairs(3))  # Output: 3
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
5
3
>>>
```

14. Unique paths

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def unique_paths(m, n):
    # Create a 2D dp array with dimensions m x n
    dp = [[1] * n for _ in range(m)]

    # Fill the dp array
    for i in range(1, m):
        for j in range(1, n):
            dp[i][j] = dp[i-1][j] + dp[i][j-1]

    return dp[m-1][n-1]

# Test case
print(unique_paths(7, 3))  # Output: 28
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
28
>>>
```

15. Largest characters

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def largeGroupPositions(s):
    if not s:
        return []

    n = len(s)
    result = []
    start = 0
    count = 1

    for i in range(1, n):
        if s[i] == s[i - 1]:
            count += 1
        else:
            if count >= 3:
                result.append([start, i - 1])
            start = i
            count = 1

    # Check the last group
    if count >= 3:
        result.append([start, n - 1])

    return result

# Test case
s = "abbxxxxzzy"
print(largeGroupPositions(s))  # Output: [[3, 6]]
```

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
[[3, 6]]
>>>
```

16. Game of life

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

```python
def gameOfLife(board):
    if not board:
        return

    m = len(board)
    n = len(board[0])

    # Create a copy of the original board
    next_board = [[board[row][col] for col in range(n)] for row in range(m)]

    # Directions for 8 neighbors (horizontal, vertical, diagonal)
    directions = [(-1, -1), (-1, 0), (-1, 1),
                  (0, -1),           (0, 1),
                  (1, -1), (1, 0), (1, 1)]

    # Iterate through each cell in the board
    for i in range(m):
        for j in range(n):
            live_neighbors = 0

            # Count live neighbors
            for di, dj in directions:
                ni, nj = i + di, j + dj
                if 0 <= ni < m and 0 <= nj < n and board[ni][nj] == 1:
                    live_neighbors += 1

            # Apply rules to update next state
            if board[i][j] == 1:  # Live cell
                if live_neighbors < 2 or live_neighbors > 3:
                    next_board[i][j] = 0
            else:  # Dead cell
                if live_neighbors == 3:
                    next_board[i][j] = 1

    # Update the original board with the next state
    for i in range(m):
        for j in range(n):
            board[i][j] = next_board[i][j]

# Test case
board = [[0,1,0],
```

Ln: 48 Col: 0

IDLE Shell 3.12.1

```
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\sleva\Desktop\Duplicate.py
[[3, 6]]
>>>
============= RESTART: C:\Users\sleva\Desktop\Duplicate.py =============
[[0, 0, 0], [1, 0, 1], [0, 1, 1], [0, 1, 0]]
>>>
```

Ln: 9 Col: 0

17. Champagne Tower

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)

File Edit Format Run Options Window Help

```python
def champagneTower(poured, query_row, query_glass):
    # Initialize the pyramid with 0.0 champagne
    dp = [[0.0] * (r + 1) for r in range(query_row + 1)]
    dp[0][0] = poured

    # Iterate through each row
    for i in range(query_row):
        for j in range(len(dp[i])):
            if dp[i][j] >= 1:
                excess = dp[i][j] - 1
                dp[i][j] = 1.0
                dp[i+1][j] += excess / 2
                dp[i+1][j+1] += excess / 2

    return dp[query_row][query_glass]

# Test case
poured = 1
query_row = 1
query_glass = 1
print(champagneTower(poured, query_row, query_glass))  # Output: 0.0
```

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\sleva\Desktop\Duplicate.py
0.0
>>>
```