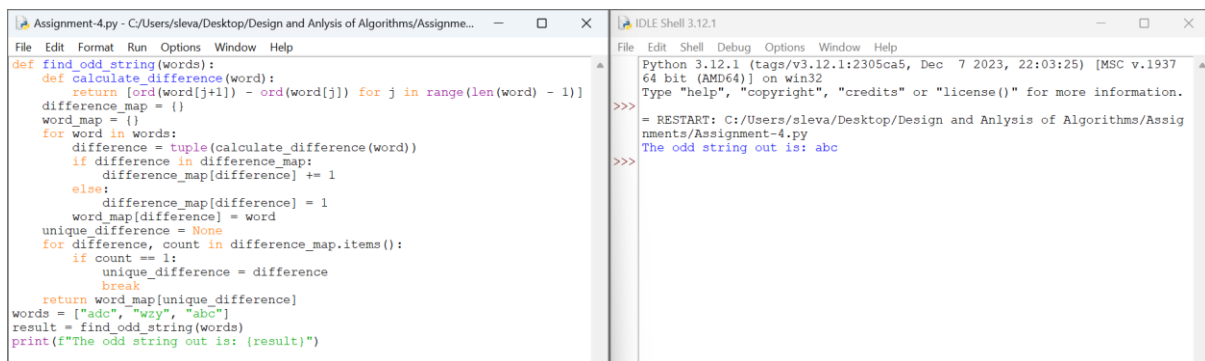


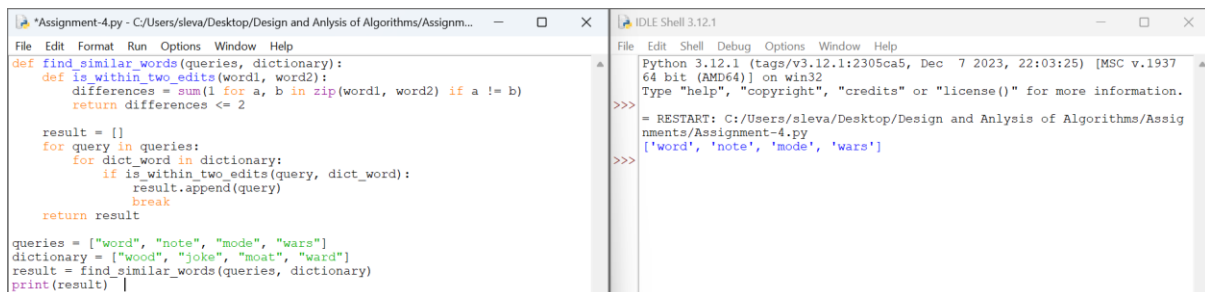
# 1. Odd String Difference



```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def find_odd_string(words):
    def calculate_difference(word):
        return [ord(word[j+1]) - ord(word[j]) for j in range(len(word) - 1)]
    difference_map = {}
    word_map = {}
    for word in words:
        difference = tuple(calculate_difference(word))
        if difference in difference_map:
            difference_map[difference] += 1
        else:
            difference_map[difference] = 1
            word_map[difference] = word
    unique_difference = None
    for difference, count in difference_map.items():
        if count == 1:
            unique_difference = difference
            break
    return word_map[unique_difference]
words = ["adc", "wzy", "abc"]
result = find_odd_string(words)
print(f"The odd string out is: {result}")

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/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
>>> The odd string out is: abc
>>>
```

# 2. Words Within Two Edits of Dictionary



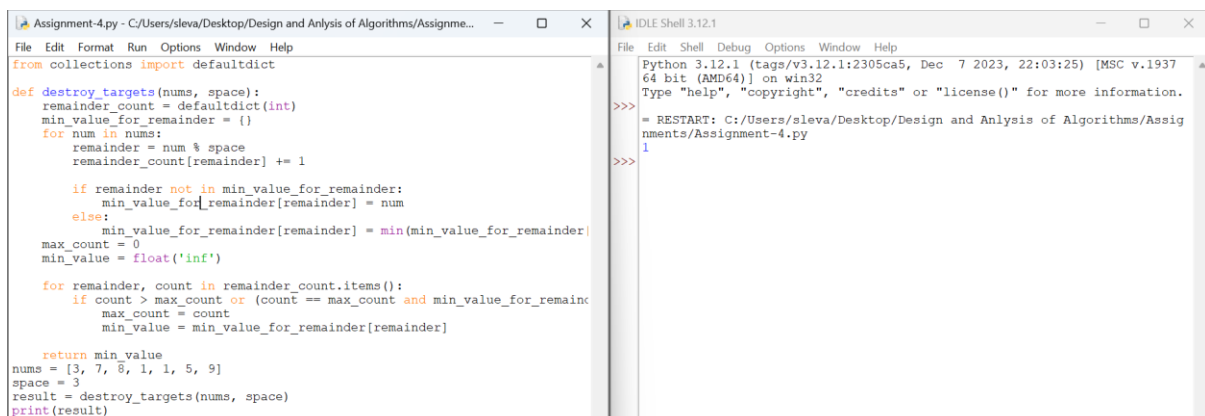
```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def find_similar_words(queries, dictionary):
    def in_within_two_edits(word1, word2):
        differences = sum(1 for a, b in zip(word1, word2) if a != b)
        return differences <= 2

    result = []
    for query in queries:
        for dict_word in dictionary:
            if in_within_two_edits(query, dict_word):
                result.append(query)
                break
    return result

queries = ["word", "note", "mode", "wars"]
dictionary = ["wood", "joke", "meat", "ward"]
result = find_similar_words(queries, dictionary)
print(result)

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/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
>>> ['word', 'note', 'mode', 'wars']
>>>
```

# 3. Destroy Sequential Targets



```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
from collections import defaultdict

def destroy_targets(nums, space):
    remainder_count = defaultdict(int)
    min_value_for_remainder = {}
    for num in nums:
        remainder = num % space
        remainder_count[remainder] += 1

        if remainder not in min_value_for_remainder:
            min_value_for_remainder[remainder] = num
        else:
            min_value_for_remainder[remainder] = min(min_value_for_remainder[remainder], num)
    max_count = 0
    min_value = float('inf')

    for remainder, count in remainder_count.items():
        if count > max_count or (count == max_count and min_value_for_remainder[remainder] < min_value):
            max_count = count
            min_value = min_value_for_remainder[remainder]

    return min_value
nums = [3, 7, 8, 1, 1, 5, 9]
space = 3
result = destroy_targets(nums, space)
print(result)

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/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
>>> 1
>>>
```

## 4. Next Greater Element IV

```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Anlysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
from heapq import heappush, heappop
from collections import deque

def second_greater(nums):
    n = len(nums)
    result = [-1] * n
    stack = []
    min_heap = []

    for i in range(n-1, -1, -1):
        # Remove invalid entries from the heap (those that do not have a valid next greater element)
        while min_heap and min_heap[0][0] <= i:
            heappop(min_heap)

        # If the heap is not empty, the top element is the second greater element
        if min_heap:
            result[i] = nums[min_heap[0][1]]

        # Push the current element to the stack
        while stack and nums[stack[-1]] <= nums[i]:
            index = stack.pop()
            heappush(min_heap, (index, index))

        stack.append(i)

    return result

# Example usage
nums = [2, 4, 0, 9, 6]
print(second_greater(nums)) # Output should be the array of second greater elements
```

```
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/sleeva/Desktop/Design and Anlysis of Algorithms/Assignments/Assignment-4.py
[0, 6, 6, -1, -1]
>>>
```

## 5. Average Value of Even Numbers That Are Divisible by Three

```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Anlysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def average_value(nums):
    # Filter the list to get even numbers divisible by 3
    filtered_numbers = [num for num in nums if num % 2 == 0 and num % 3 == 0]

    # Check if there are no such numbers to avoid division by zero
    if not filtered_numbers:
        return 0

    # Calculate the sum and the count of the filtered numbers
    total_sum = sum(filtered_numbers)
    count = len(filtered_numbers)

    # Calculate the average and round down
    average = total_sum // count

    return average

# Example usage
nums = [1, 3, 6, 10, 12, 15]
print(average_value(nums)) # Output should be 9
```

```
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/sleeva/Desktop/Design and Anlysis of Algorithms/Assignments/Assignment-4.py
9
>>>
```

## 6. Minimum Addition to Make Integer Beautiful

```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Anlysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def min_beautiful_number(n, target):
    # Calculate the sum of digits of n
    digit_sum = sum(int(digit) for digit in str(n))

    # Calculate the difference between target and digit_sum
    diff = target - digit_sum

    # If difference is non-negative, return it as the answer
    if diff >= 0:
        return diff

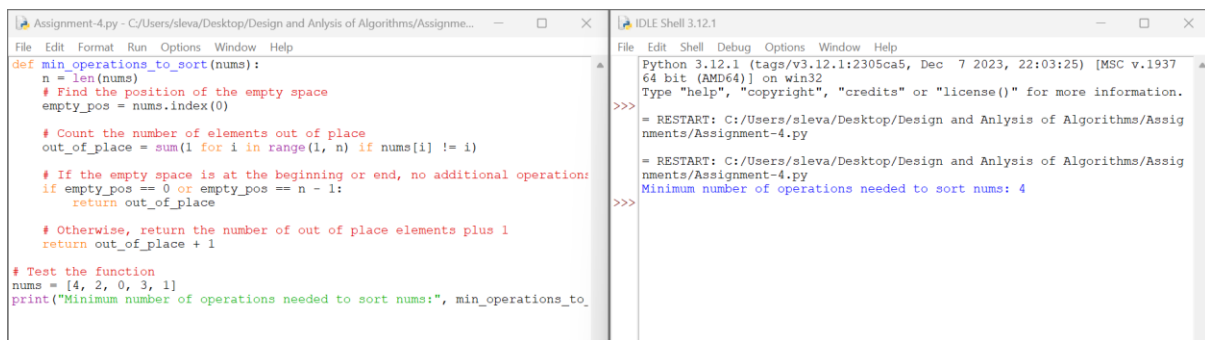
    # If difference is negative, find the minimum non-negative integer x
    # such that adding x to n makes the sum of digits equal to the target
    x = 0
    while diff < 0:
        # Increment x by multiplying 10^(position) where position is the current position
        diff += 9 * 10**x
        x += 1

    return x

# Example usage
n = 16
target = 6
print(min_beautiful_number(n, target)) # Output should be 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/sleeva/Desktop/Design and Anlysis of Algorithms/Assignments/Assignment-4.py
4
>>>
```

## 7. Sort Array by Moving Items to Empty Space



```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def min_operations_to_sort(nums):
    n = len(nums)
    # Find the position of the empty space
    empty_pos = nums.index(0)

    # Count the number of elements out of place
    out_of_place = sum(1 for i in range(1, n) if nums[i] != i)

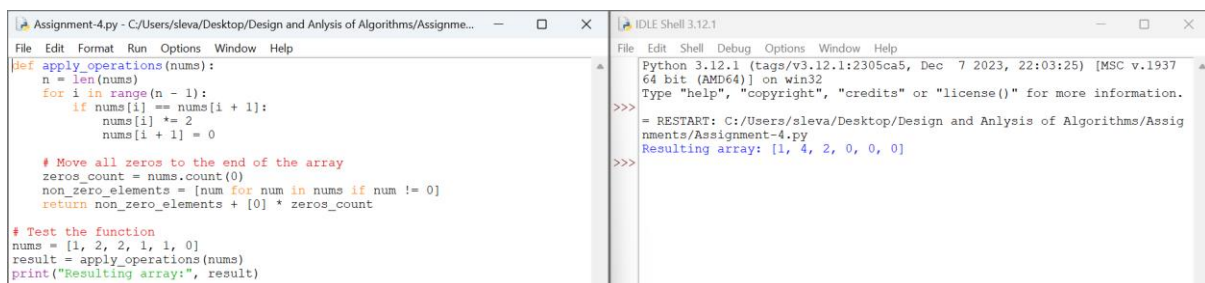
    # If the empty space is at the beginning or end, no additional operations:
    if empty_pos == 0 or empty_pos == n - 1:
        return out_of_place

    # Otherwise, return the number of out of place elements plus 1
    return out_of_place + 1

# Test the function
nums = [4, 2, 0, 3, 1]
print("Minimum number of operations needed to sort nums:", min_operations_to_

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/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
>>>
= RESTART: C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
Minimum number of operations needed to sort nums: 4
>>>
```

## 8. Apply Operations to an Array



```
Assignment-4.py - C:/Users/sleeva/Desktop/Design and Analysis of Algorithms/Assignme...
File Edit Format Run Options Window Help
def apply_operations(nums):
    n = len(nums)
    for i in range(n - 1):
        if nums[i] == nums[i + 1]:
            nums[i] *= 2
            nums[i + 1] = 0

    # Move all zeros to the end of the array
    zeros_count = nums.count(0)
    non_zero_elements = [num for num in nums if num != 0]
    return non_zero_elements + [0] * zeros_count

# Test the function
nums = [1, 2, 2, 1, 1, 0]
result = apply_operations(nums)
print("Resulting array:", result)

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/sleeva/Desktop/Design and Analysis of Algorithms/Assignments/Assignment-4.py
Resulting array: [1, 4, 2, 0, 0, 0]
>>>
```