ASSIGNMENT-4

Name:G.venkata Praveen

Reg.no:192373023

Dept:cse(D.s)

1. Odd String Difference

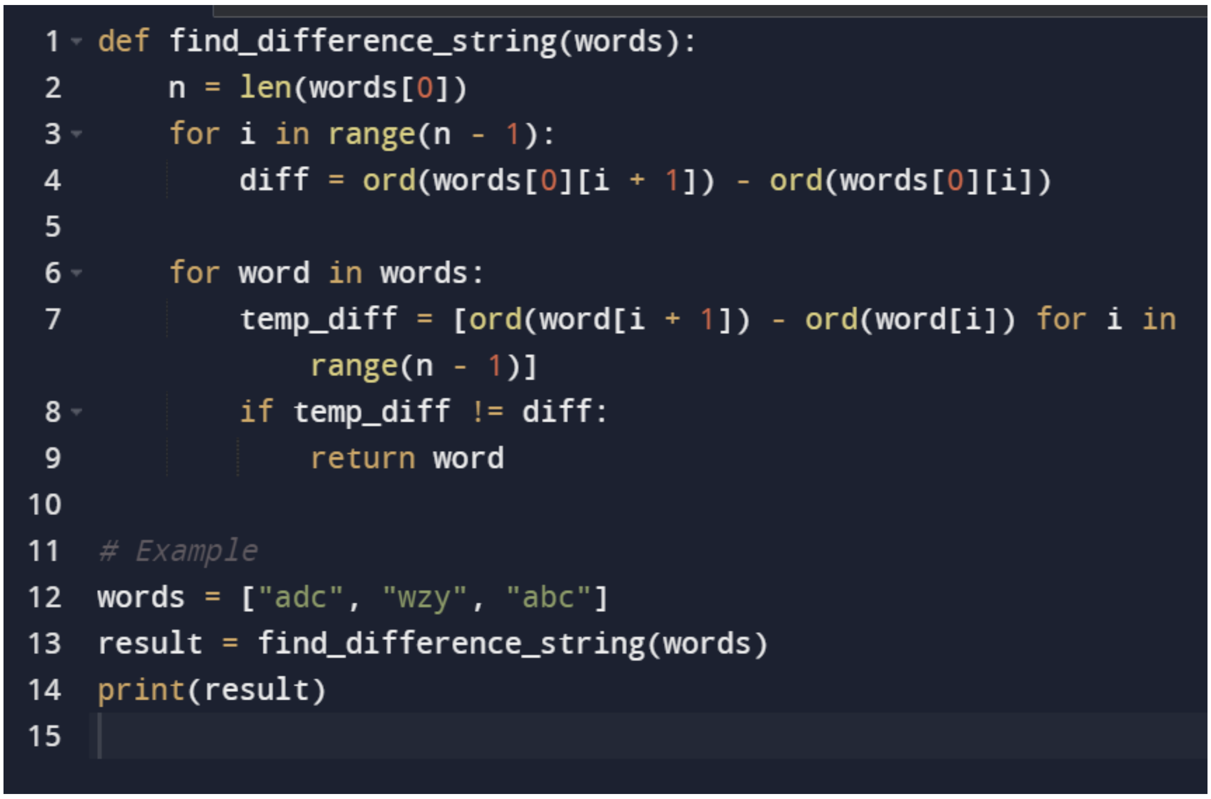
You are given an array of equal-length strings words. Assume that the length of each string is n.

Each string words[i] can be converted into a difference integer array difference[i] of length n - 1 where difference[i][j] = words[i][j+1] - words[i][j] where 0 <= j <= n - 2. Note that the difference between two letters is the difference between their positions in the alphabet i.e. the position of 'a' is 0, 'b' is 1, and 'z' is 25.

For example, for the string "acb", the difference integer array is [2 - 0, 1 - 2] = [2, -1]. All the strings in words have the same difference integer array, except one. You should find that string.

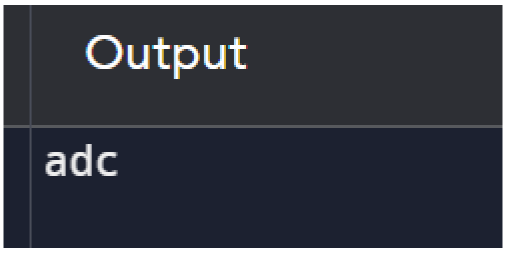
Return the string in words that has different difference integer array.

Program:



```python
def find_difference_string(words):
    n = len(words[0])
    for i in range(n - 1):
        diff = ord(words[0][i + 1]) - ord(words[0][i])

    for word in words:
        temp_diff = [ord(word[i + 1]) - ord(word[i]) for i in
            range(n - 1)]
        if temp_diff != diff:
            return word

# Example
words = ["adc", "wzy", "abc"]
result = find_difference_string(words)
print(result)
```

Output:



```
Output
adc
```

2. Words Within Two Edits of Dictionary

You are given two string arrays, queries and dictionary. All words in each array comprise of lowercase English letters and have the same length.

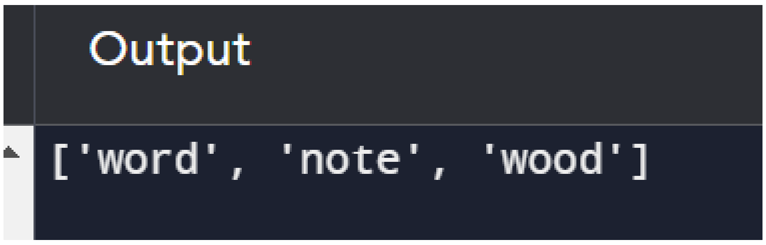
In one edit you can take a word from queries, and change any letter in it to any other letter. Find all words from queries that, after a maximum of two edits, equal some word from dictionary.

Return a list of all words from queries, that match with some word from dictionary after a maximum of two edits. Return the words in the same order they appear in queries

Program:

```python
def is_within_two_edits(word1, word2):
    if word1 == word2:
        return True
    if len(word1) != len(word2):
        return False

    edits = 0
    for c1, c2 in zip(word1, word2):
        if c1 != c2:
            edits += 1
            if edits > 2:
                return False

    return True

def find_matching_words(queries, dictionary):
    result = []
    for query in queries:
        for word in dictionary:
            if is_within_two_edits(query, word):
                result.append(query)
                break

    return result

# Example
queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
output = find_matching_words(queries, dictionary)
print(output)  # Output: ["word", "note", "wood"]
```

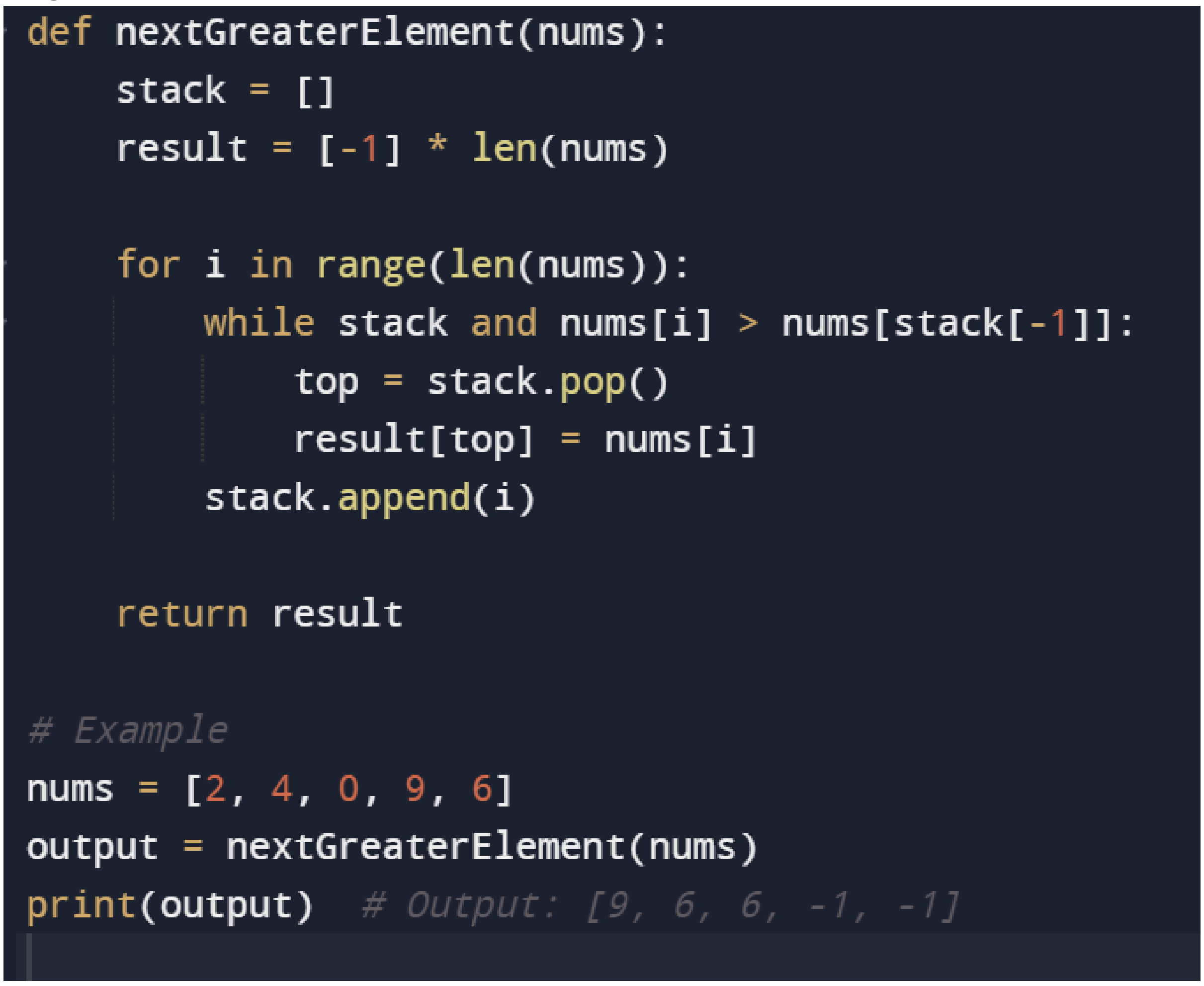
Output:



```
Output

['word', 'note', 'wood']
```

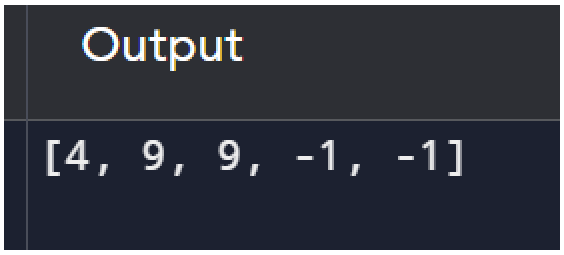
3. Next Greater Element IV
   You are given a 0-indexed array of non-negative integers nums. For each integer in nums, you must find its respective second greater integer.
   The second greater integer of nums[i] is nums[j] such that: j > i
   nums[j] > nums[i]
   There exists exactly one index k such that nums[k] > nums[i] and i < k < j.
   If there is no such nums[j], the second greater integer is considered to be -1.

For example, in the array [1, 2, 4, 3], the second greater integer of 1 is 4, 2 is 3, and that of 3 and 4 is -1.
Return an integer array answer, where answer[i] is the second greater integer of nums[i].
Program:



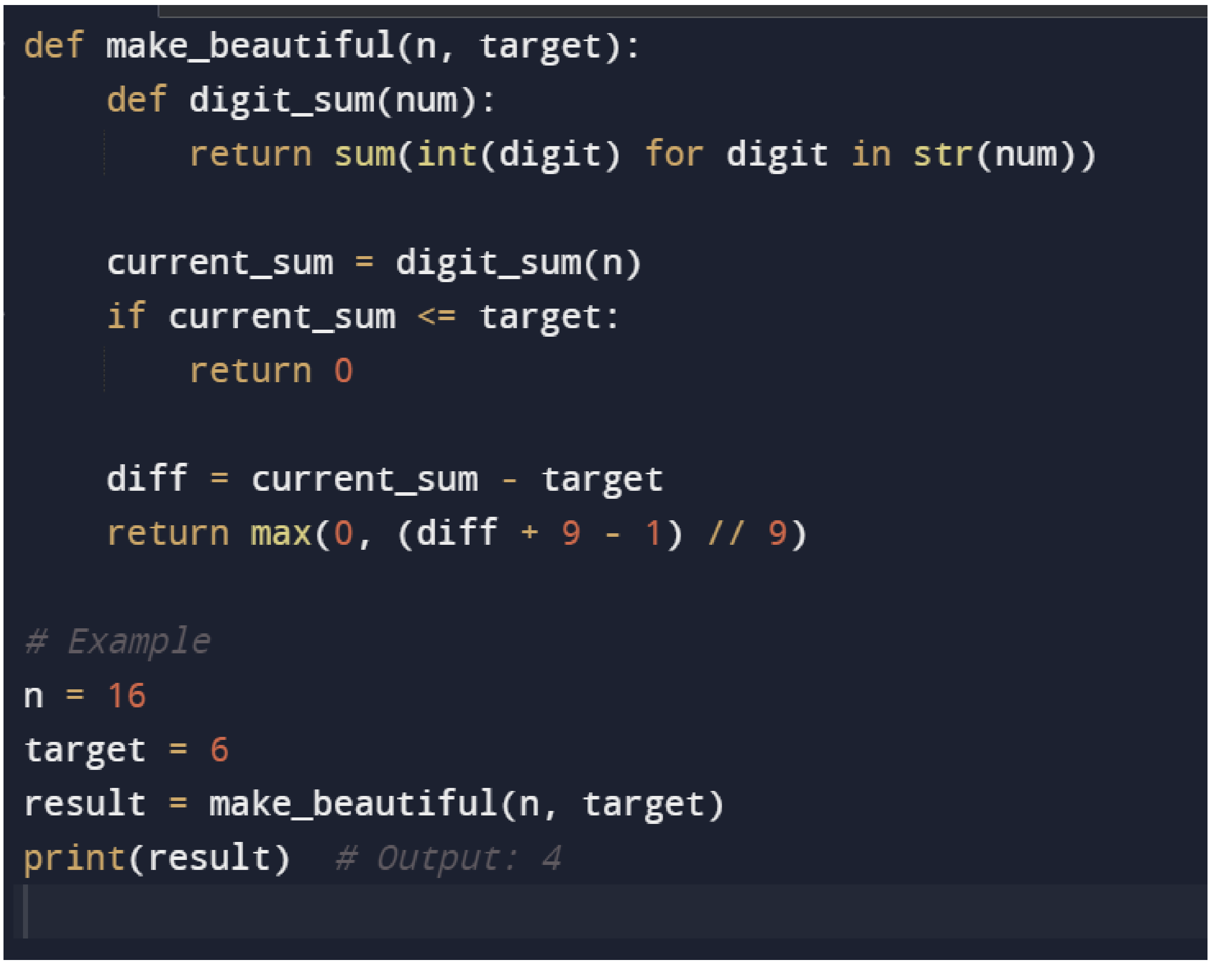
```python
def nextGreaterElement(nums):
    stack = []
    result = [-1] * len(nums)

    for i in range(len(nums)):
        while stack and nums[i] > nums[stack[-1]]:
            top = stack.pop()
            result[top] = nums[i]
        stack.append(i)

    return result

# Example
nums = [2, 4, 0, 9, 6]
output = nextGreaterElement(nums)
print(output)  # Output: [9, 6, 6, -1, -1]
```

Output:



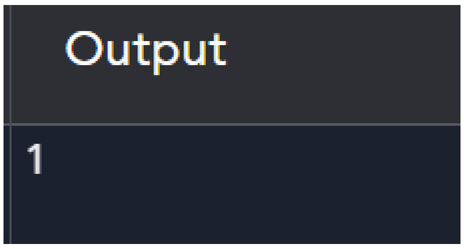
```
Output
[4, 9, 9, -1, -1]
```

4. Minimum Addition to Make Integer Beautiful
 You are given two positive integers n and target.
 An integer is considered beautiful if the sum of its digits is less than or equal to target.
 Return the minimum non-negative integer x such that n + x is beautiful. The input will be generated such that it is always possible to make n beautiful.
Program:



```python
def make_beautiful(n, target):
    def digit_sum(num):
        return sum(int(digit) for digit in str(num))

    current_sum = digit_sum(n)
    if current_sum <= target:
        return 0

    diff = current_sum - target
    return max(0, (diff + 9 - 1) // 9)

# Example
n = 16
target = 6
result = make_beautiful(n, target)
print(result)  # Output: 4
```

Output:



```
Output

1
```

5. Sort Array by Moving Items to Empty Space

   You are given an integer array nums of size n containing each element from 0 to n - 1 (inclusive). Each of the elements from 1 to n - 1 represents an item, and the element 0 represents an empty space.

   one operation, you can move any item to the empty space. nums is considered to be sorted if the numbers of all the items are in ascending order and the empty space is either at the beginning or at the end of the array.

   For example, if n = 4, nums is sorted if:

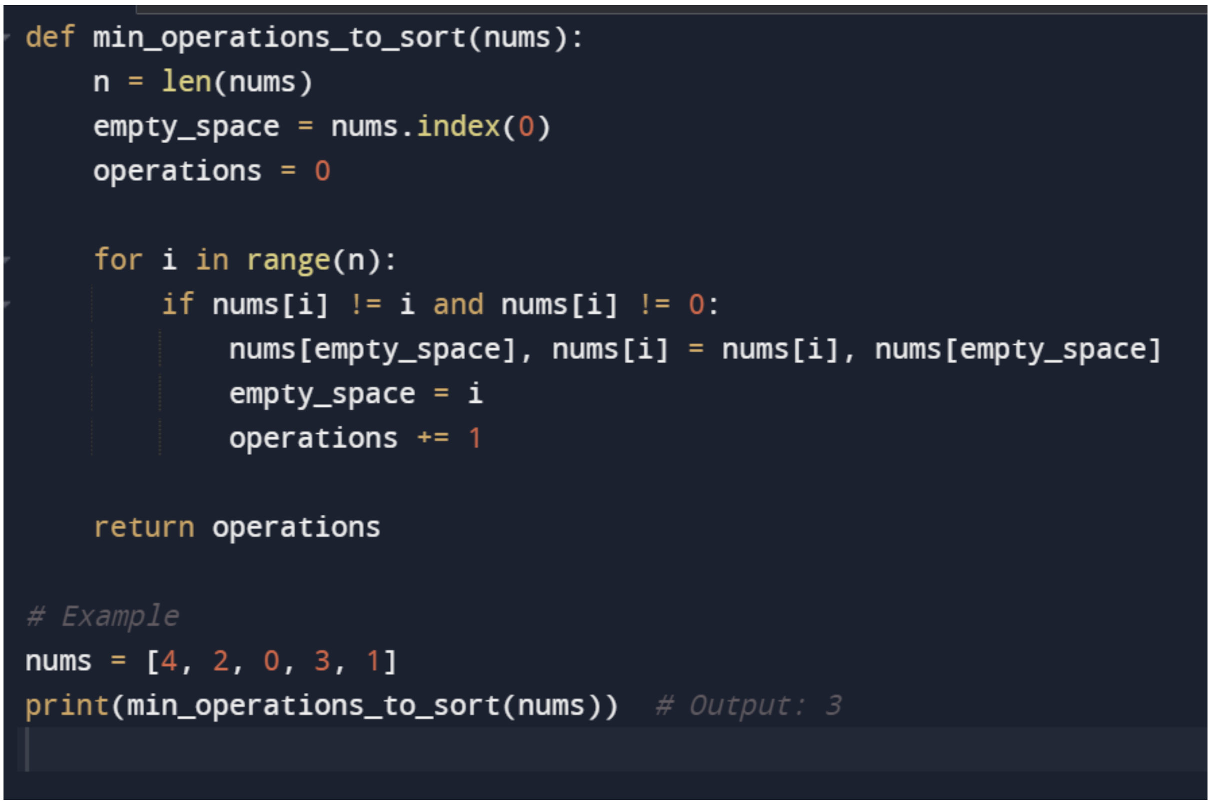
   ● nums = [0,1,2,3] or

   ● nums = [1,2,3,0]

   ...and considered to be unsorted otherwise.

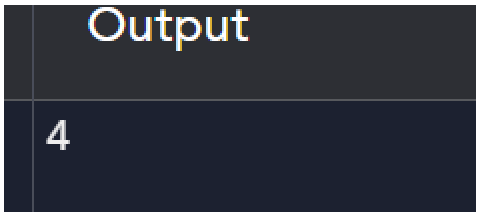
   Return the minimum number of operations needed to sort nums.

   Program:



```python
def min_operations_to_sort(nums):
    n = len(nums)
    empty_space = nums.index(0)
    operations = 0

    for i in range(n):
        if nums[i] != i and nums[i] != 0:
            nums[empty_space], nums[i] = nums[i], nums[empty_space]
            empty_space = i
            operations += 1

    return operations

# Example
nums = [4, 2, 0, 3, 1]
print(min_operations_to_sort(nums))  # Output: 3
```

Output:



```
Output

4
```

6. Apply Operations to an Array

You are given a 0-indexed array nums of size n consisting of non-negative integers. You need to apply n - 1 operations to this array where, in the ith operation (0- indexed), you will apply the following on the ith element of nums:

● If nums[i] == nums[i + 1], then multiply nums[i] by 2 and set nums[i + 1] to 0. Otherwise, you skip this operation.

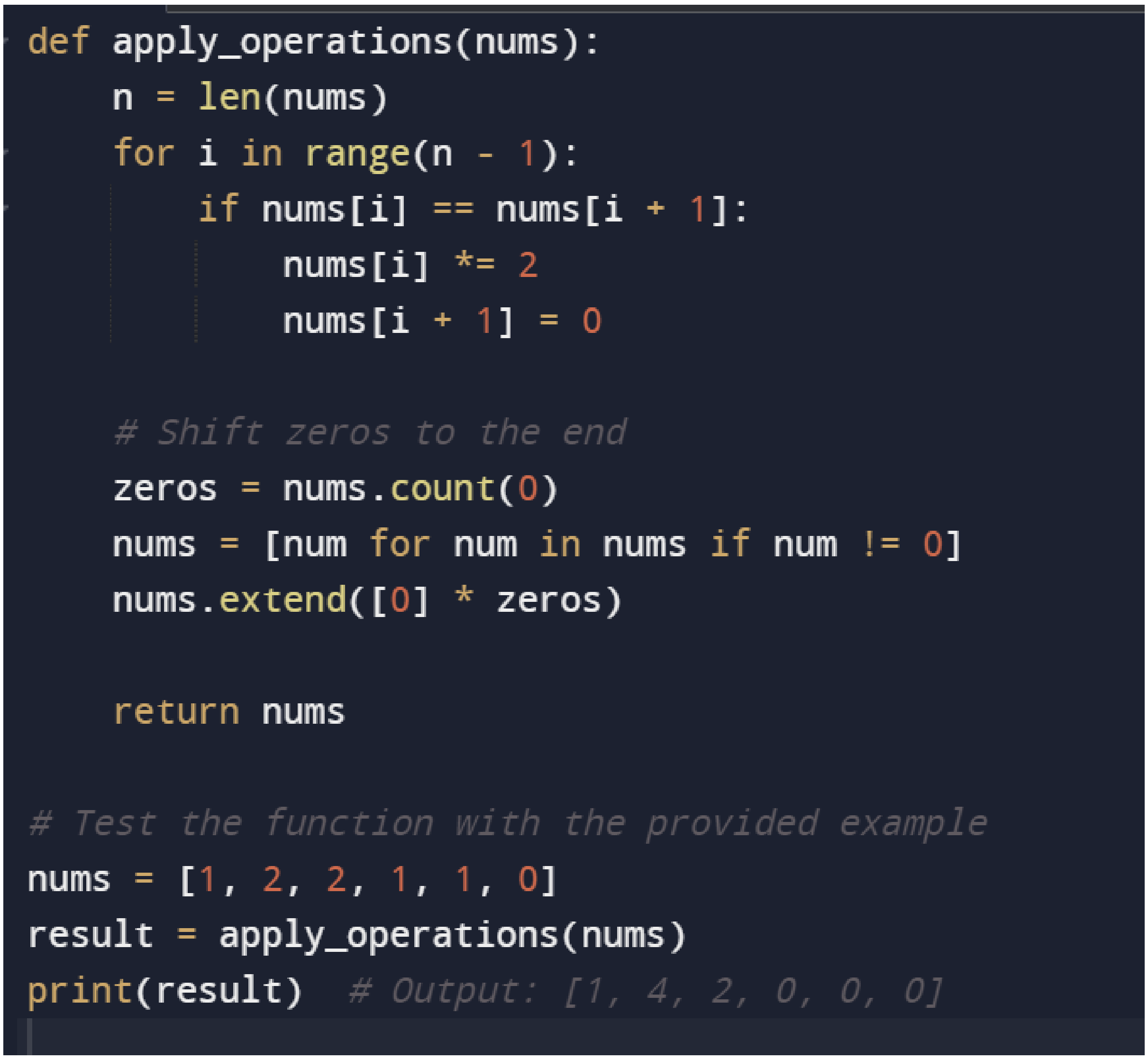
After performing all the operations, shift all the 0's to the end of the array.

● For example, the array [1,0,2,0,0,1] after shifting all its 0's to the end, is [1,2,1,0,0,0].

Return the resulting array.

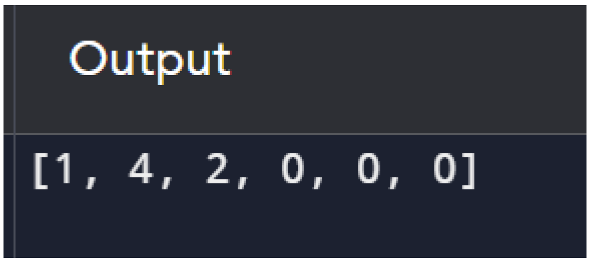
Note that the operations are applied sequentially, not all at once.

Program:



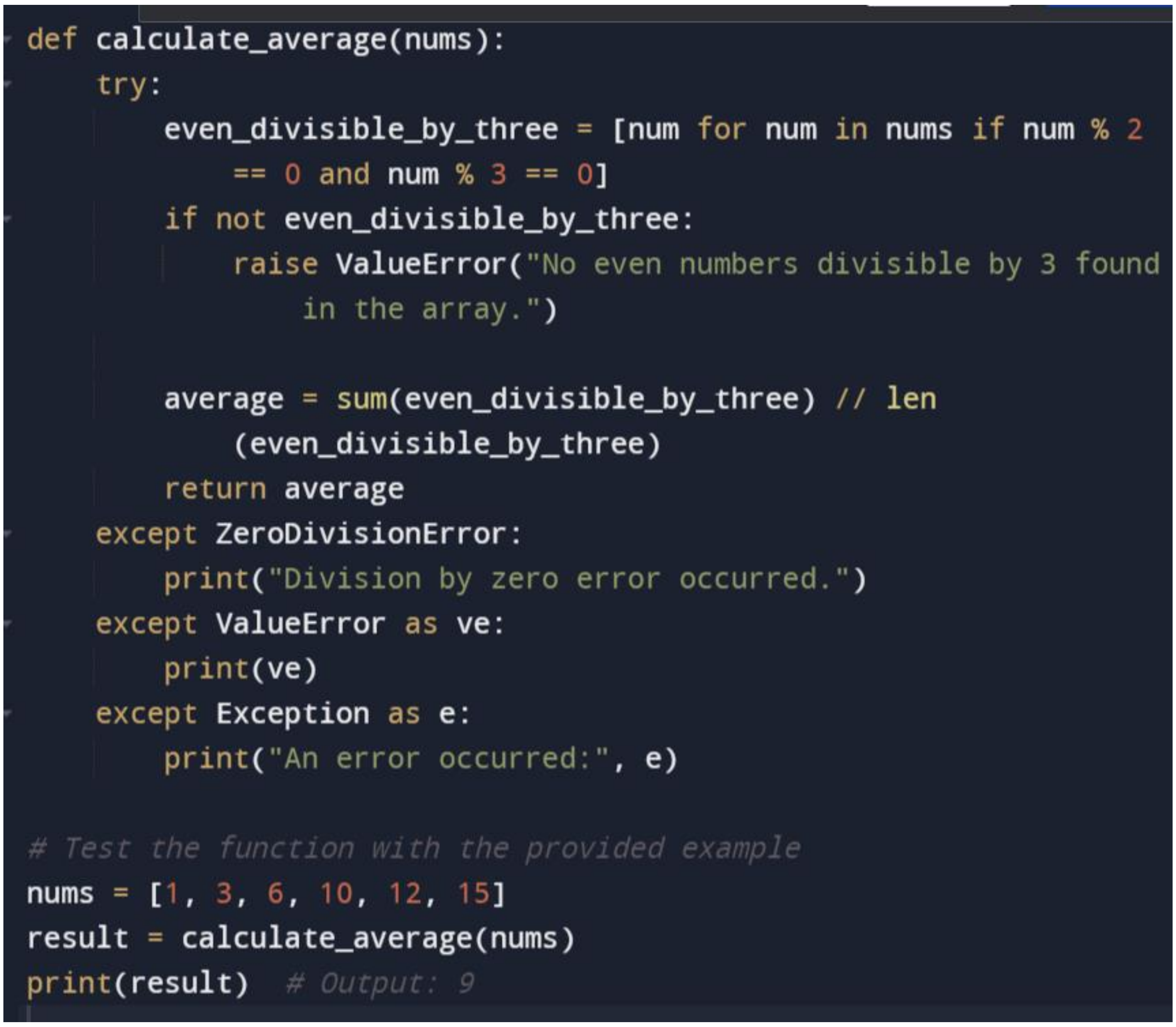
```python
def apply_operations(nums):
    n = len(nums)
    for i in range(n - 1):
        if nums[i] == nums[i + 1]:
            nums[i] *= 2
            nums[i + 1] = 0

    # Shift zeros to the end
    zeros = nums.count(0)
    nums = [num for num in nums if num != 0]
    nums.extend([0] * zeros)

    return nums

# Test the function with the provided example
nums = [1, 2, 2, 1, 1, 0]
result = apply_operations(nums)
print(result)  # Output: [1, 4, 2, 0, 0, 0]
```

Output:



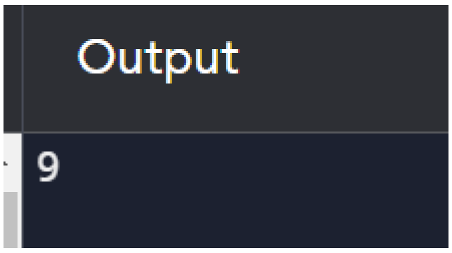
```
Output

[1, 4, 2, 0, 0, 0]
```

7. Average Value of Even Numbers That Are Divisible by Three
   Given an integer array nums of positive integers, return the average value of all even integers that are divisible by 3.
   Note that the average of n elements is the sum of the n elements divided by n and rounded down to the nearest integer.
   Program:



```python
def calculate_average(nums):
    try:
        even_divisible_by_three = [num for num in nums if num % 2
            == 0 and num % 3 == 0]
        if not even_divisible_by_three:
            raise ValueError("No even numbers divisible by 3 found
                in the array.")

        average = sum(even_divisible_by_three) // len
            (even_divisible_by_three)
        return average
    except ZeroDivisionError:
        print("Division by zero error occurred.")
    except ValueError as ve:
        print(ve)
    except Exception as e:
        print("An error occurred:", e)

# Test the function with the provided example
nums = [1, 3, 6, 10, 12, 15]
result = calculate_average(nums)
print(result)  # Output: 9
```

Output:



```
Output
9
```

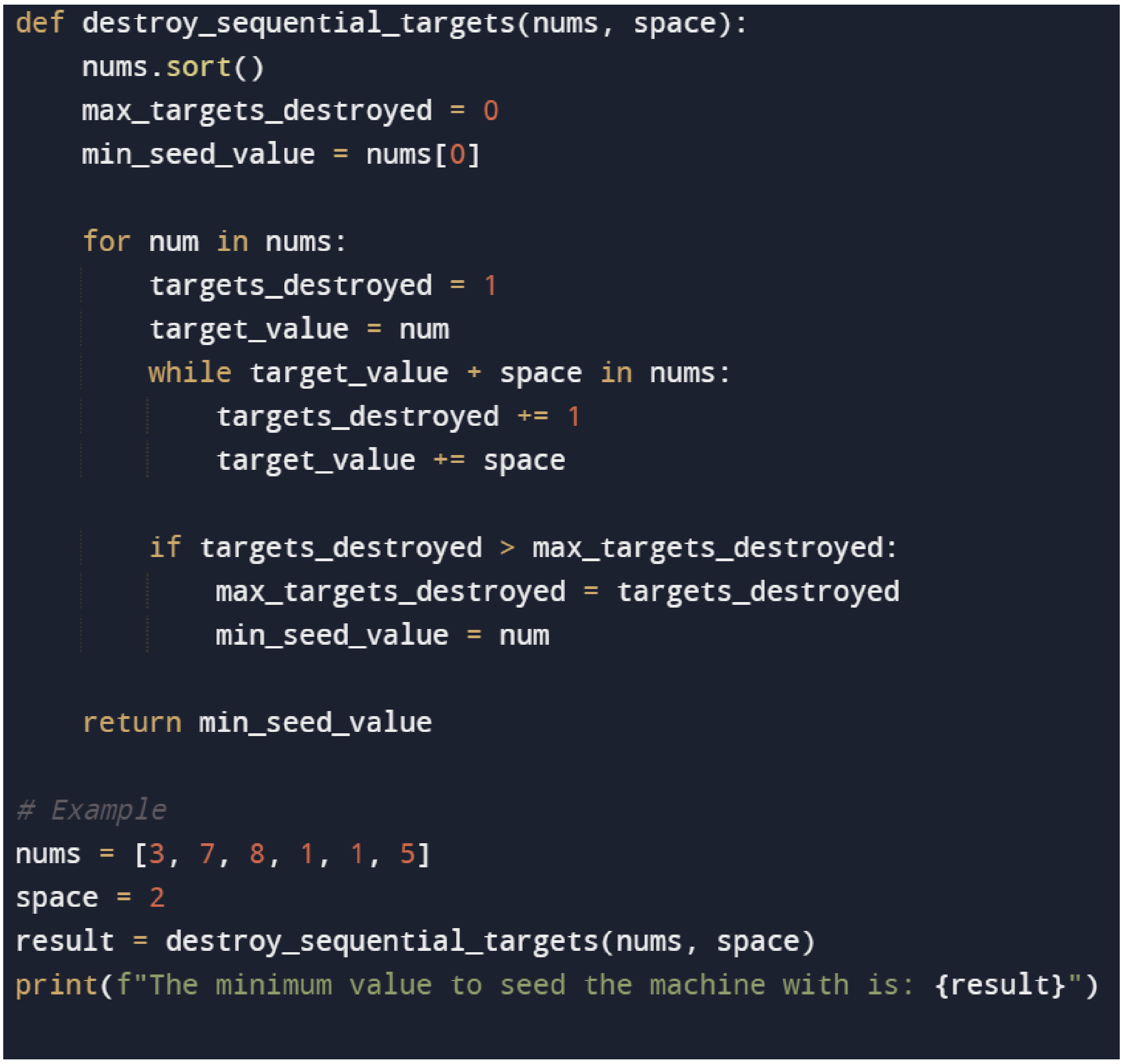
8. Destroy Sequential Targets

   You are given a 0-indexed array nums consisting of positive integers, representing targets on a number line. You are also given an integer space.

   You have a machine which can destroy targets. Seeding the machine with some nums[i] allows it to destroy all targets with values that can be represented as nums[i] + c * space, where c is any non-negative integer. You want to destroy the maximum number of targets in nums.

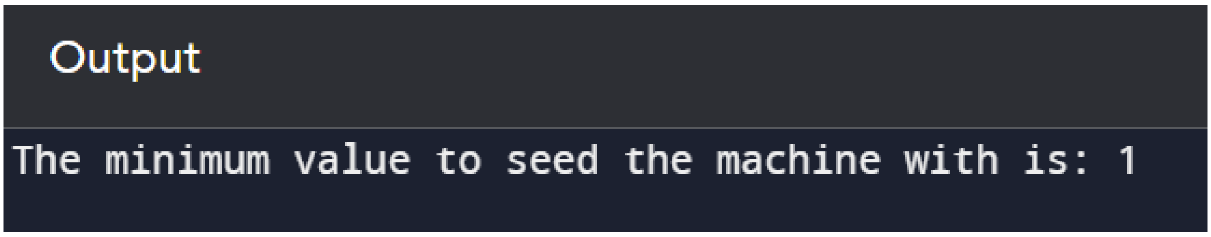
   Return the minimum value of nums[i] you can seed the machine with to destroy the maximum number of targets.

   Program:



```python
def destroy_sequential_targets(nums, space):
    nums.sort()
    max_targets_destroyed = 0
    min_seed_value = nums[0]

    for num in nums:
        targets_destroyed = 1
        target_value = num
        while target_value + space in nums:
            targets_destroyed += 1
            target_value += space

        if targets_destroyed > max_targets_destroyed:
            max_targets_destroyed = targets_destroyed
            min_seed_value = num

    return min_seed_value

# Example
nums = [3, 7, 8, 1, 1, 5]
space = 2
result = destroy_sequential_targets(nums, space)
print(f"The minimum value to seed the machine with is: {result}")
```

Output:



Output

```
The minimum value to seed the machine with is: 1
```