

ASSIGNMENT 4

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 $0 \leq j \leq n - 2$. 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.

Example 1: Input: words = ["adc","wzy","abc"]

CODE:

```
def difference_array(word):
    return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]

def odd_string_difference(words):
    difference_arrays = [difference_array(word) for word in words]
    difference_counts = {}

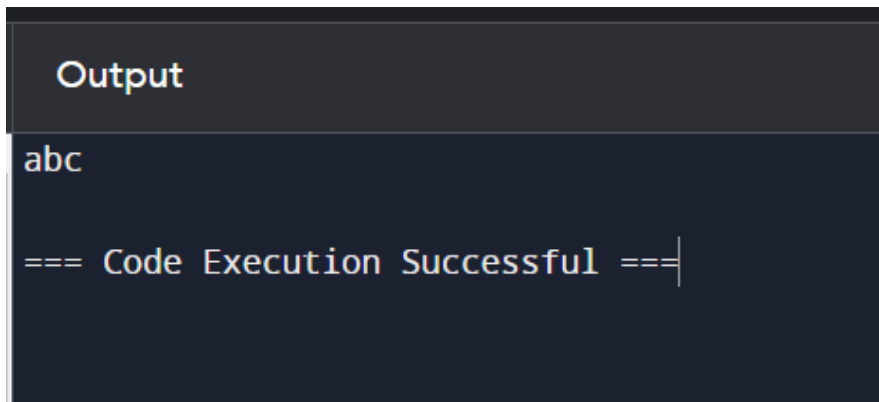
    for diff in difference_arrays:
        diff_tuple = tuple(diff)
        if diff_tuple in difference_counts:
            difference_counts[diff_tuple] += 1
        else:
            difference_counts[diff_tuple] = 1
```

```
    unique_diff = None
    for diff, count in difference_counts.items():
        if count == 1:
            unique_diff = diff
            break

    for word, diff in zip(words, difference_arrays):
        if tuple(diff) == unique_diff:
            return word

words = ["adc", "wzy", "abc"]
print(odd_string_difference(words))
```

OUTPUT:

A screenshot of a code execution environment with a dark background. At the top, the word "Output" is written in a light blue font. Below it, the string "abc" is displayed in a light blue font. At the bottom, the text "=== Code Execution Successful ===" is shown in a light blue font, followed by a vertical cursor line.

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.

**Example 1: Input: queries = ["word", "note", "ants", "wood"],
dictionary = ["wood", "joke", "moat"]**

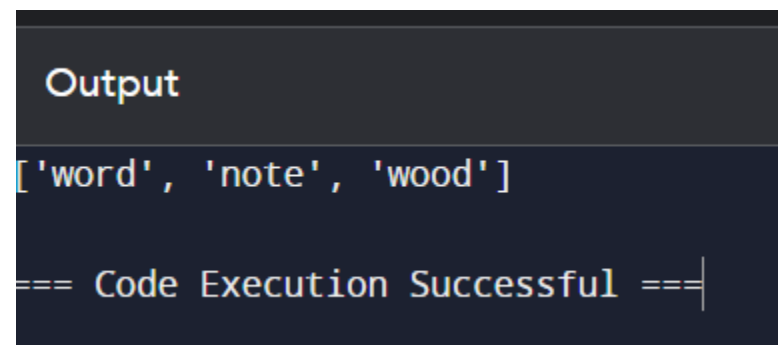
CODE:

```
def within_two_edits(word1, word2):
    differences = sum(1 for a, b in zip(word1, word2) if a != b)
    return differences <= 2

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

queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
print(words_within_two_edits(queries, dictionary))
```

OUTPUT:



The screenshot shows a dark-themed interface with a header labeled "Output". Below the header, the output of the code is displayed as a list: `['word', 'note', 'wood']`. At the bottom of the interface, a status message reads `=== Code Execution Successful ===` followed by a cursor.

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]`.

Example 1: Input: `nums = [2,4,0,9,6]`

CODE:

```
def second_greater(nums):
    n = len(nums)
    answer = [-1] * n
    first_stack = []
    second_stack = []
    for i in range(n):
        while second_stack and nums[second_stack[-1]] < nums[i]:
            idx = second_stack.pop()
            answer[idx] = nums[i]
        temp_stack = []
        while first_stack and nums[first_stack[-1]] < nums[i]:
            temp_stack.append(first_stack.pop())
            while temp_stack:
                second_stack.append(temp_stack.pop())
        first_stack.append(i)
    return answer
nums = [2, 4, 0, 9, 6]
print(second_greater(nums))
nums2 = [3, 3]
print(second_greater(nums2))
```

OUTPUT:

```
Output
[9, 6, 6, -1, -1]
[-1, -1]

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

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.

Example 1: Input: $n = 16$, $target = 6$

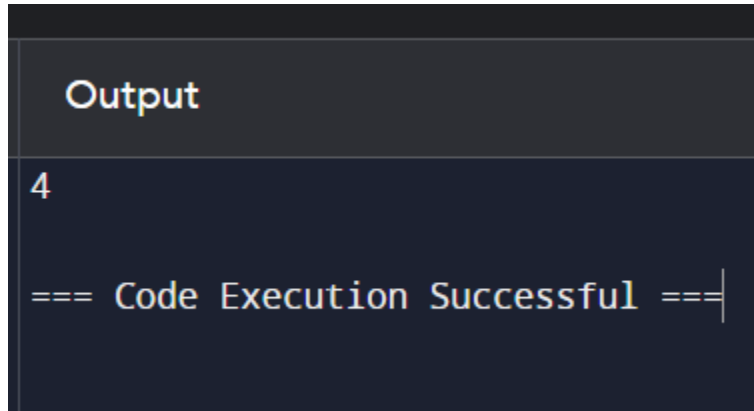
CODE:

```
def sum_of_digits(num):
    return sum(int(digit) for digit in str(num))

def make_integer_beautiful(n, target):
    x = 0
    while sum_of_digits(n + x) > target:
        increment = 10 - (n + x) % 10
        x += increment
    return x
n = 16
target = 6
```

```
print(make_integer_beautiful(n, target))
```

OUTPUT:



```
Output
4
=== Code Execution Successful ===
```

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. In 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`.

Example 1: Input: `nums = [4,2,0,3,1]`.

CODE:

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

```
for i in range(1, n):
    if nums[i] != i:
        target = i
        if nums[i] == 0:
            target = empty_pos
        nums[i], nums[target] = nums[target], nums[i]
        empty_pos = i
        operations += 1
return operations
nums = [4, 2, 0, 3, 1]
print(min_operations_to_sort(nums))
```

OUTPUT:

Output
3
=== Code Execution Successful ===