Remove Element

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in `nums` **in-place**. The order of the elements may be changed. Then return *the number of elements in* `nums` *which are not equal to* `val`.

Consider the number of elements in `nums` which are not equal to `val` be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the elements which are not equal to `val`. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

Custom Judge:

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be **accepted**.

Example 1:

```
Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2, with the first two elements of n
ums being 2.
It does not matter what you leave beyond the returned k (hence they are unde
rscores).
```

Example 2:

```
Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of n
ums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k (hence they are unde
rscores).
```

First things first we will have two variables to solve this problem `index` and `i` where `index` is the last unique element and `i` is the current element we will use `i` and if `i` is not equal to the value we will set the next element to the current element and increment `index` by 1

```
class Solution:
    def removeElement(self, nums: List[int], val: int) → int:
        # index is the last unique element
        # i is the current element
        index = 0
        # iterate through the array
        for i in nums:
            # if the current element is not equal to the value to be removed
            # set the next element to the current element
```

```
            # increment index by one
            if i != val:
                nums[index] = i
                index += 1
        return index
```

Very easy problem this if you want to play around with the input and out puts just edit the nums array and the value you wish to remove

```
if __name__ == "__main__":
    solution = Solution()

    # Example input
    nums = [3, 2, 2, 3] #change
    val = 3 #change

    # Call the removeElement method
    length = solution.removeElement(nums, val)

    # Print the result
    print("Length of array after removing elements:", length)
    print("Array after removing elements:", nums[:length])
```