Search in Rotated Sorted Array

Question:

https://leetcode.com/problems/search-in-rotated-sorted-array/

There is an integer array `nums` sorted in ascending order (with distinct values).

Prior to being passed to your function, `nums` is possibly rotated at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (0-indexed). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.

Given the array `nums` after the possible rotation and an integer `target`, return the index of `target` if it is in `nums`, or `-1` if it is not in `nums`.

Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

```
Input: nums = [1], target = 0
Output: -1
```

Approach 1:

Using Linear Search directly to look for the target.

Solution 1:

https://gist.github.com/vermaayush680/474209fbc059c22877c4093fe07a6b13

```
for i in range(len(nums)):
        if target==nums[i]:
                return i
return -1
```

Approach 2:

Using a variation of binary search to find the target. We know that we are rotating around a pivot so everything before the pivot will be sorted and everything after the pivot will be sorted.

Solution 2:

https://gist.github.com/vermaayush680/d3d4b1677b78ec18e8f3575345214c59

```
l, r = 0, len(nums)-1
while l <= r:
        mid = l + (r-l)//2
        if nums[mid] == target:
                return mid
        if nums[l] <= nums[mid]:
                if nums[l] <= target < nums[mid]:
                        r = mid - 1
                else:
                        l = mid + 1
        else:
                if nums[mid] < target <= nums[r]:
                        l = mid + 1
                else:
                        r = mid - 1
return -1
```