Solution Review: First Non-Repeating Integer in a list

This review provides a detailed analysis of the different ways to find the first non-repeating integer in a list.

We'll cover the following

- Solution: Brute Force
  - Time Complexity

Solution: Brute Force

```
def find_first_unique(lst):
    for index1 in range(len(lst)):
        index2 = 0
        # iterate the second list using index2
        while(index2 < len(lst)):
            if (index1 != index2 and lst[index1] == lst[index2]):
                break
            index2 += 1
        if (index2 == len(lst)):
            return lst[index1]
    return None


print(find_first_unique([9, 2, 3, 2, 6, 6]))

```

We start with the first element of the list and compare it with [illegible] elements while traversing the list. If no other same element with the same value is found, then this is the first non-repeating element in the list (9 in our example). If, however, a similar element is found, we skip to the second element of the list to check if it is unique. The process gets repeated for the entire list.

Time Complexity

The time complexity of this solution is $O(n^2)$ since the entire list is iterated n times for each element→ $n \times n$

> **Note:** The solution provided above is not the optimal solution for this problem. We can write a more efficient solution using hashing. We will cover that approach in Hashing Chapter: Challenge 9