Google Question

- Given an array, return the first recurring character

Example1 : array = [2,1,4,2,6,5,1,4]

- It should return 2

Example 2 : array = [2,6,4,6,1,3,8,1,2]

- It should return 6

In [19]:

```python
def frc(array):
    saver = dict()
    for item in array:
        if item in saver:
            return item
        else:
            saver[item] = 0
    return None

# Testing frc function
array = [2,1,4,2,6,5,1,4]
array2 = [2,6,4,6,1,3,8,1,2]

print(frc(array))
print(frc(array2))
```

```
2
6
```

Steps taken:

1- Create a dictionary to store the items.

2- To find the first recurring character, we loop over the array; each item will be stored in the dictionary created above as we go along the loop.

3 - Before adding an item to the dictionary, we verify if the item is already there.

4 - If so, we return said item and break out of the loop.

5 - If not, we add the element to the dictionary and keep looping.

6 - When there is nothing else to loop and we could not find a (first) recurring character, the program will return 'None'.

The time complexity is O(n), because we only loop through the array once, and the search we do in the dictionary is of O(1) time, since we are essentially implementing a hash table.

We can also do the same using nested loops:

In [20]:

```python
def naive_frc(array):
    l = len(array)
    i = 0
    frc = None
    while(i < l):
        j = i + 1
        while(j < l):
            if array[i] == array[j]:
                l = j
                frc = array[j]
                continue
            else:
                j += 1
        i += 1
    return frc

print(naive_frc(array))
print(frc(array2))
```

```
2
6
```

This function is going to grab the first item of the array and check it against the rest until if finds a recurring character. If it does not find anything, it will proceed to the second item of the array, do the same and so on until it finds a recurring character.

The time complexity is O(n^2) but space complexity is O(1), which is better than the earlier solution.