class - 1 (Array)

Finding time & space Complexity :-

Given an array we want to find it's time and space complexity:-

$a = [1, 2, 3, 4, 5] \longrightarrow$ target = 4 Space Comp O(1)

```
for i in range (len(a)):
        if a[i] == target :
              print ("found")
    Else:
              print ("not found")
```

} $\longrightarrow$ time comp. O(n)

Time Complexity:- As we traverse all the element in case of worst case then the worst case time Complexity will be O(n)

Space Complexity:- As the given array does not gradually increase in size (we append element without loop) the worst case space complexity will be O(1)

(*) Copying element from one list to another:-

Given, ~~a[illegible]~~ a = [4, 5, 7, 8]

<u>Space</u>

```
b=[ ] -> O(1)
for i in range (len(a)):
    b[i] = a[i]   -> O(n)
```

Here the initialize empty list b is gradually increasing so it's space complexity will be O(~~n~~)

<u>time</u>

```
b ~~=~~[ ]   -> O(1)
for i in range (len(a)):
    b[i] = a[i]   -> O(n)
```

Here we are traversing all the way to the list a and copying it's element to list b. That's why time complexity is O(n).

(*) Reversing ~~a~~ list with time comp. O(n) and space complexity O(n):-

Given, a=[2,3,4,5]

```
b=[ ]
~~for i in range~~ (len(a)):
n = len(a)-1
i = 0      j = 0
while  i <= n :
b[j] = a[n]
```

<u>Space</u>   <u>time</u>

```
O(1) <- b = [ ]                  -> O(1)
O(1) <- n = len(a) - 1           -> O(1)
        i = 0
        while (n>,0):
O(n) <-   { b[i] = a[n]
             n -= 1          } -> O(n)
             i += 1
```

Here empty list b gradually grows and that's why space complexity is O(n) and as we have to traverse all the elements of list a that's why time complexity is also O(n)

⊛ Reversing a list with space complexity

time O(1) :-

```
time                                      space
O(1) ← a = [4,6,7,8]               →  O(1)
O(1) ← { i = 0
         j = len(a) -1 }           →  O(1)
         while i<j:
             t1 = a[i]
             t2 = a[j]
O(1) ←       a[i] = t2             →  O(n/2) ≅ O(n)
             a[j] = t1
             i+=1
             j-=1
```

Here we didn't use any extra list and we reverse the list with the existing list so there is no increase in the size of list - That's why space complexity is O(1)

And for time complexity we have to traverse the list half of the length and that's why the worst case complexity is O(n)