CHAITRA SAMANT

231070055

DAA – LAB 02

1. **AIM**

   In this lab we try to design algorithms for linear and binary search using iterative and recursive approach and learn about different coding styles.

2. **PROGRAM**

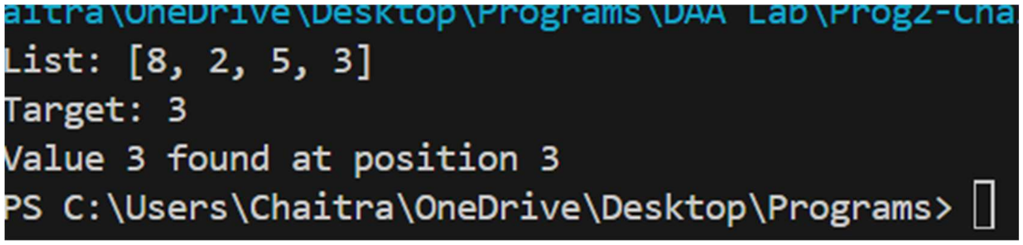
```python
def is_sorted(arr)->bool:
    """
    This function checks if the given list for binary search is
    sorted in ascending order

    Arguments:
        arr (list):The list which needs to be checked if it's sorted

    Returns:
        bool: True if list is sorted. False if it isn't

    """
    for i in range(1,len(arr)):
        if(arr[i]<arr[i-1]):
            return False

    return True


def display_output(x):
    """
    This function displays output after searching

    Arguments:
        x (int): The index returned after calling respective search function

    Returns:
        void: no value returned

    """
    if x>-1:
        print(f"Value {target} found at position {x}")
    else:
        print(f"Value {target} is not found in the given list")
```

```python
def linear_search(arr, target) -> int:
    """
    This function performs linear search on an list of elements

    Arguments:
        arr (list): The list in which target value needs to be searched
        target (int): This is the value that we need to search

    Returns:
        int: The index of the target if it is found and -1 if it isn't

    """
    if arr:
        ind = -1
        for i in range(len(arr)):
            if arr[i] == target:
                ind = i
                break
        return ind
    else:
        print("List is empty")
        exit(1)



def binary_search(arr, target, beg, end) -> int:
    """
    This function performs binary search on a sorted list of elements

    Arguments:
        arr (list): The sorted list in which target value needs to be searched
        target (int): This is the value that we need to search

    Returns:
        int: The index of the target if it is found and -1 if it isn't

    """
    if arr:
        if beg <= end:
            mid = (beg + end) // 2
            if arr[mid] == target:
                return mid
            elif arr[mid] > target:
                return binary_search(arr, target, beg, mid - 1)
            else:
                return binary_search(arr, target, mid + 1, end)
        else:
            return -1
    else:
        print("List is empty")
        exit(1)


# Initialising list and variables
arr = [6,3,1,9,13]
target = 9
beg, end = 0, len(arr) - 1

# Displaying Input
print("List:",arr)
print("Target:",target)


# Linear Search
x = linear_search(arr, target)
display_output(x)

# Binary Search
if(is_sorted(arr)):
    x = binary_search(arr, target, beg, end)
    display_output(x)
else:
    print("List isn't sorted, Binary Search can only be performed on sorted
lists")
```

3. **TESTCASES – Linear Search**

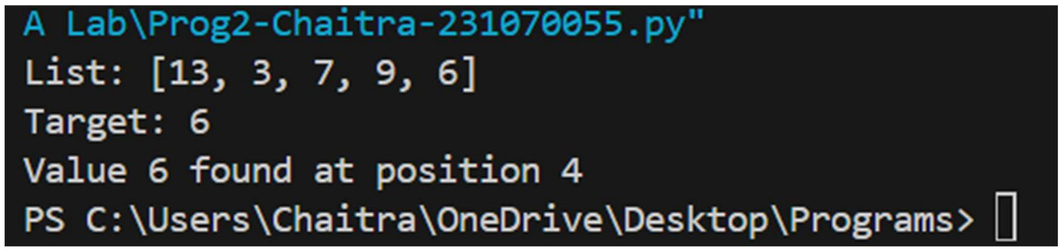
Positive Testcases

a. Target in the list



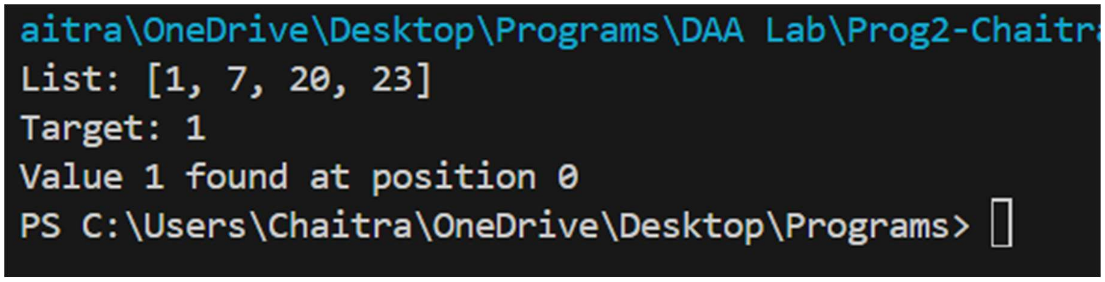
```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Cha
List: [8, 2, 5, 3]
Target: 3
Value 3 found at position 3
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

b. Target at the end of the list



```
A Lab\Prog2-Chaitra-231070055.py"
List: [13, 3, 7, 9, 6]
Target: 6
Value 6 found at position 4
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

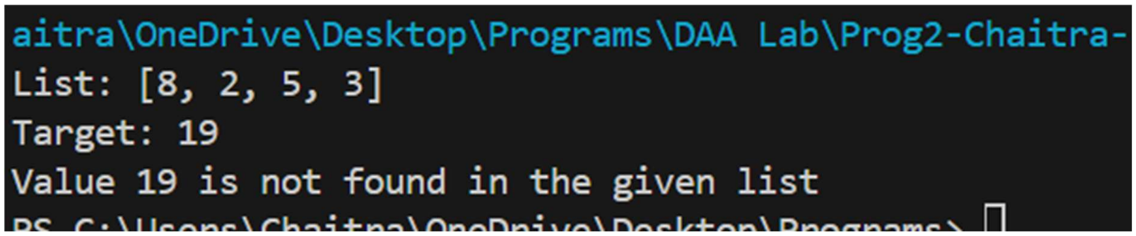
c. Target at the start of the list



```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Chaitr
List: [1, 7, 20, 23]
Target: 1
Value 1 found at position 0
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

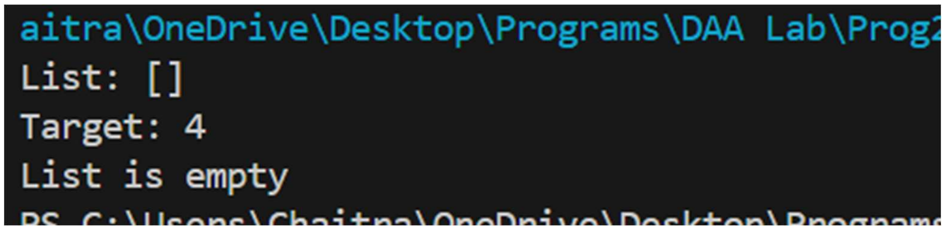
Negative Testcases

a. Target not in list



```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Chaitra-
List: [8, 2, 5, 3]
Target: 19
Value 19 is not found in the given list
```

b. List is Empty

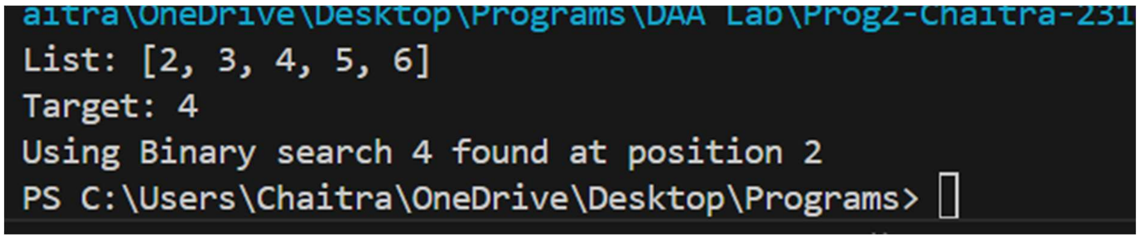


```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2
List: []
Target: 4
List is empty
```

4. **TESTCASES – Binary Search**

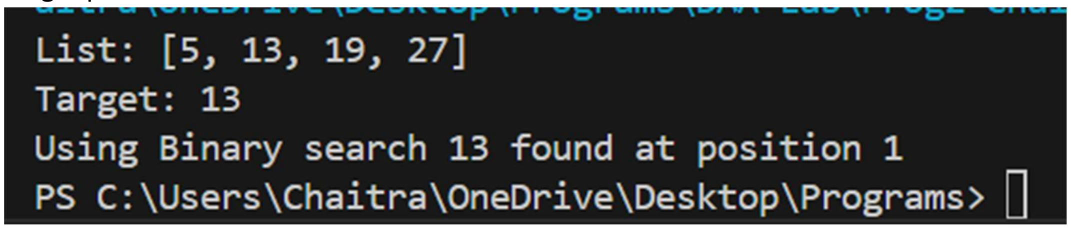
Positive Testcases

a. Target in the middle of the list



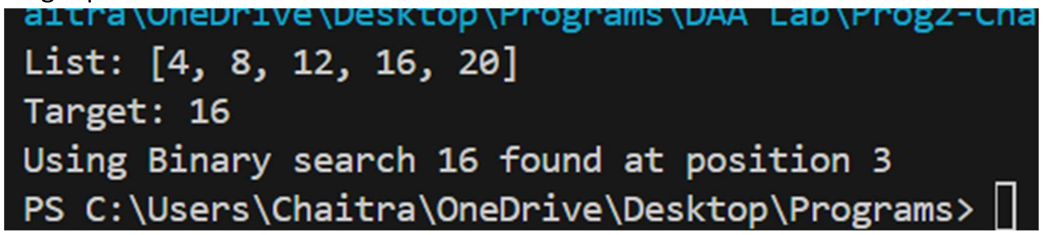
```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Chaitra-231
List: [2, 3, 4, 5, 6]
Target: 4
Using Binary search 4 found at position 2
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

b. Target present in the list



```
List: [5, 13, 19, 27]
Target: 13
Using Binary search 13 found at position 1
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

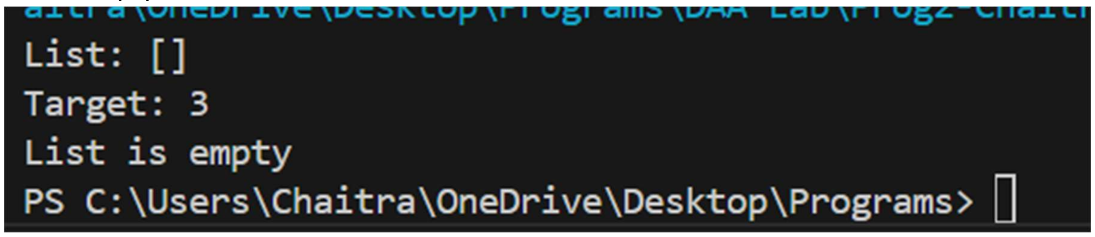
c. Target present in the list



```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Cha
List: [4, 8, 12, 16, 20]
Target: 16
Using Binary search 16 found at position 3
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

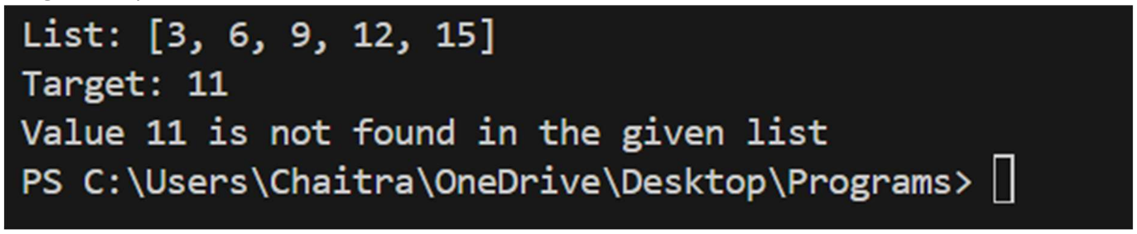
Negative Testcases

a. List is empty



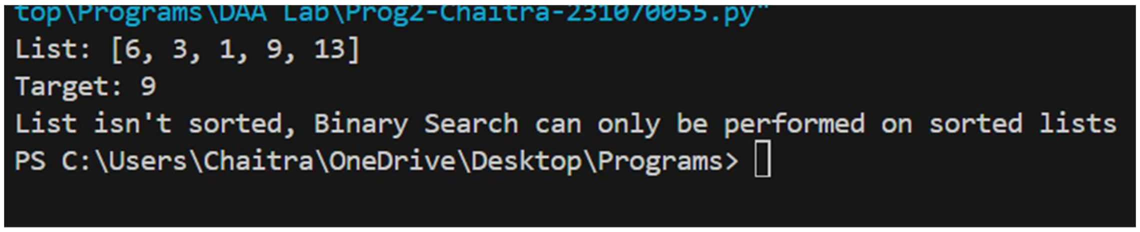
```
aitra\OneDrive\Desktop\Programs\DAA Lab\Prog2-Chaitr
List: []
Target: 3
List is empty
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

b. Target not present in the list



```
List: [3, 6, 9, 12, 15]
Target: 11
Value 11 is not found in the given list
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

c. List is not sorted



```
top\Programs\DAA Lab\Prog2-Chaitra-231070055.py"
List: [6, 3, 1, 9, 13]
Target: 9
List isn't sorted, Binary Search can only be performed on sorted lists
PS C:\Users\Chaitra\OneDrive\Desktop\Programs>
```

5. CONCLUSION

Here we studied how to implement linear and binary search algorithms and the importance of coding styles. Coding styles help to increase readability of the program making it easier to understand for other people. We included docstrings, appropriate indentation, comments and naming conventions. We also mathematically calculated Time complexity of the iterative linear search and recursive binary search.