DAILY ONLINE ACTIVITIES SUMMARY

Date:	**02-06-2020**	**Name:**	**ASHIKA**
Sem & Sec	**6 A**	**USN:**	**4AL17CS016**

Online Test Summary

Subject	**CGV**		
Max. Marks	**30**	**Score**	**24**

Certification Course Summary

Course	**MACHINE LEARNING WITH PYTHON**		
Certificate Provider	**Cognitive class**	**Duration**	**12 hour**

Coding Challenges

Problem Statement:-

1. Python program to return a list containing first and last element using list slicing method

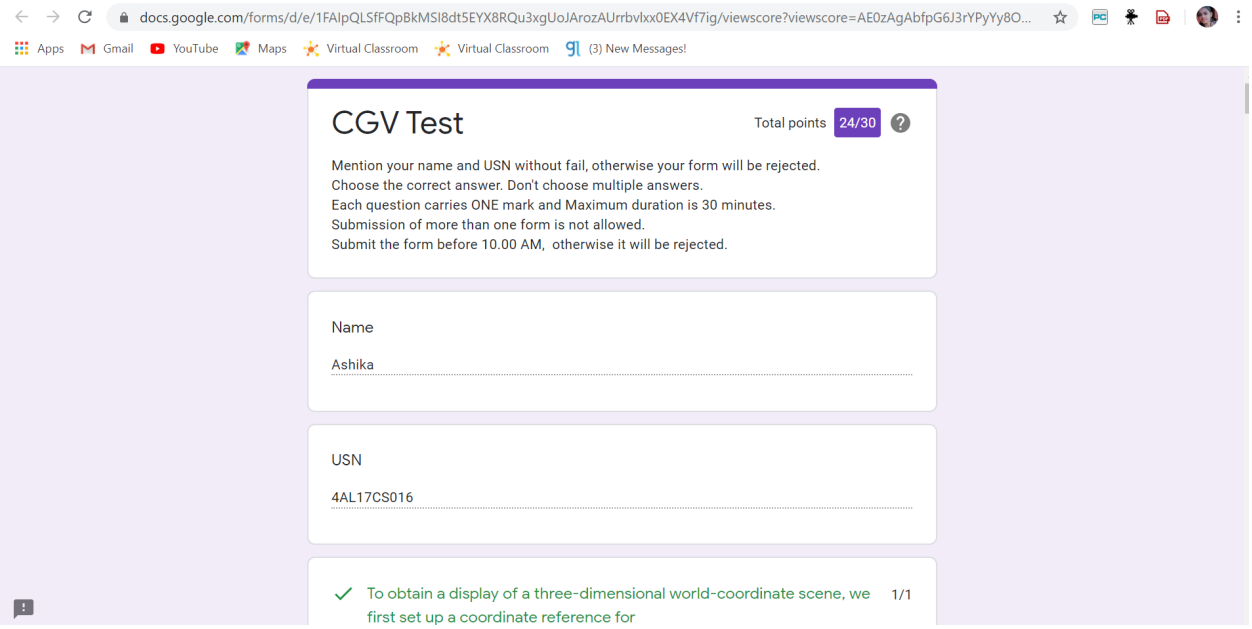
2 Write a program to check if given linked list has a loop or not.
Description:
Sometimes a linked list get corrupt, and two nodes point to the same node, which forms the loop or cycle in the linked list

3 3. Given an array of positive integers. Write a C Program to find inversion count of array.

Status:- done(executed)

Uploaded the report in Github	**yes**
If yes Repository name	https://github.com/ASHIKA-05/DAILY-REPORT
Uploaded the report in slack	**yes**

Subject: CGV



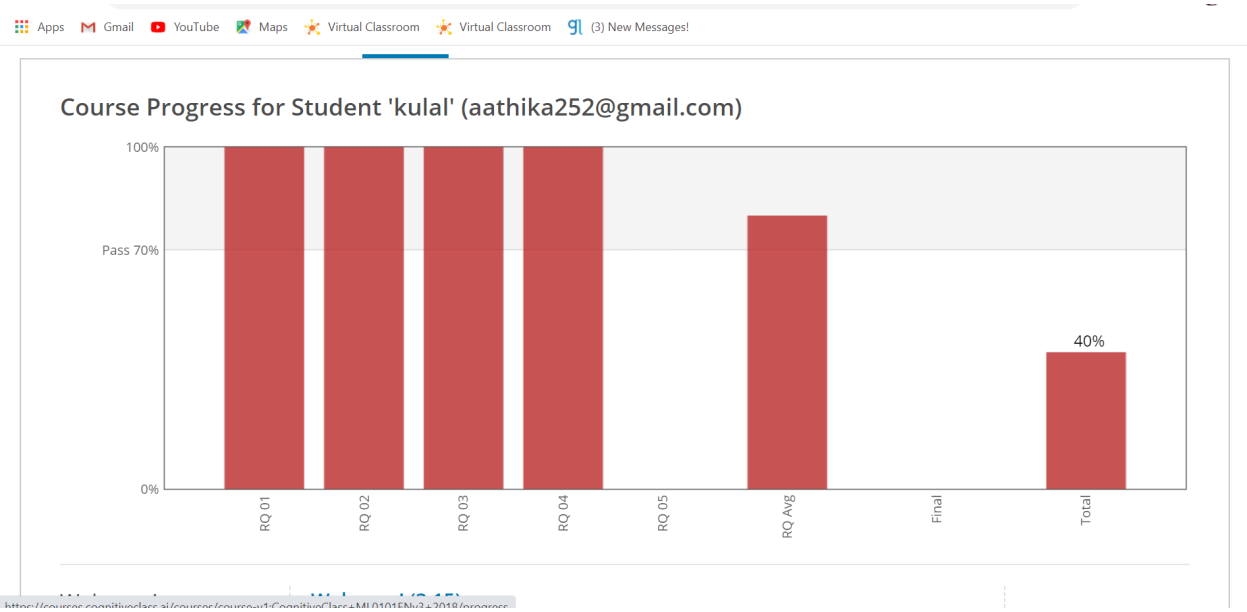
CERTIFICATION COURSE

Today I have studied

Collaborating filtering in machine learning

Learning the similarity weights

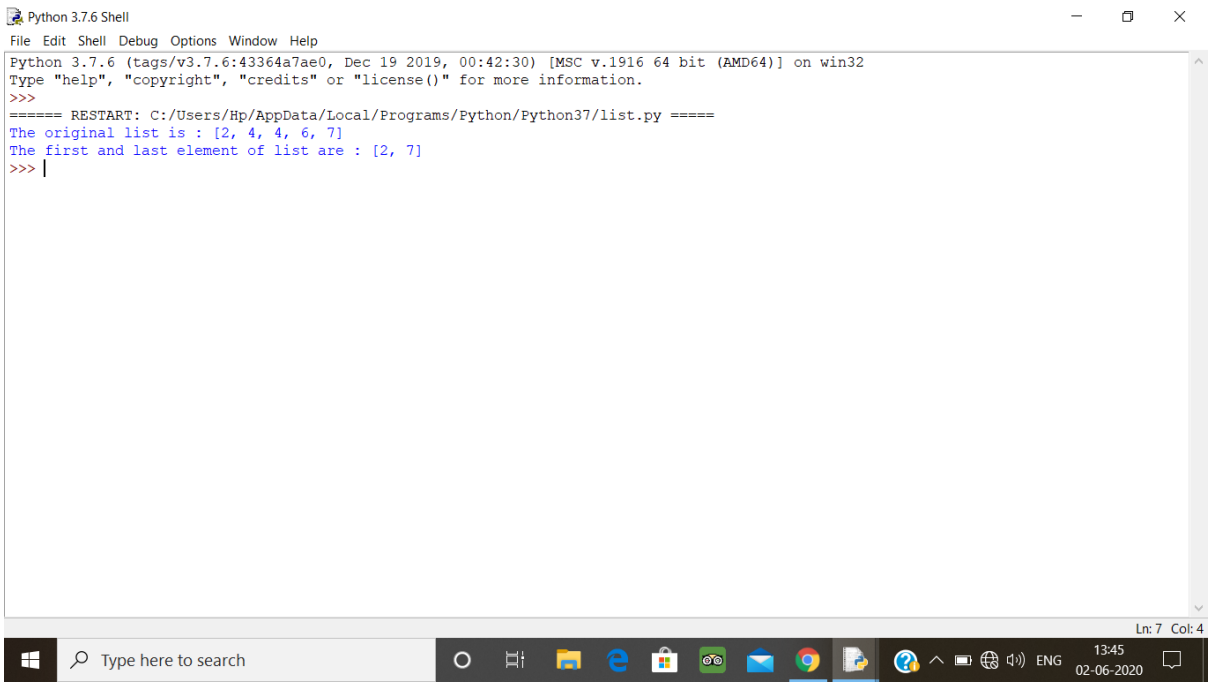
Accurate recommendations of product



Online codeing

2. Python program to return a list containing first and last element using list slicing method

```
test_list = [2,4,4,6,7]
print ("The original list is : " +  str(test_list))
res = [ test_list[0], test_list[-1] ]
print ("The first and last element of list are : " +  str(res))
```



2. Write a program to check if given linked list has a loop or not.

Description:
Sometimes a linked list get corrupt, and two nodes point to the same node, which forms the loop or cycle in the linked list

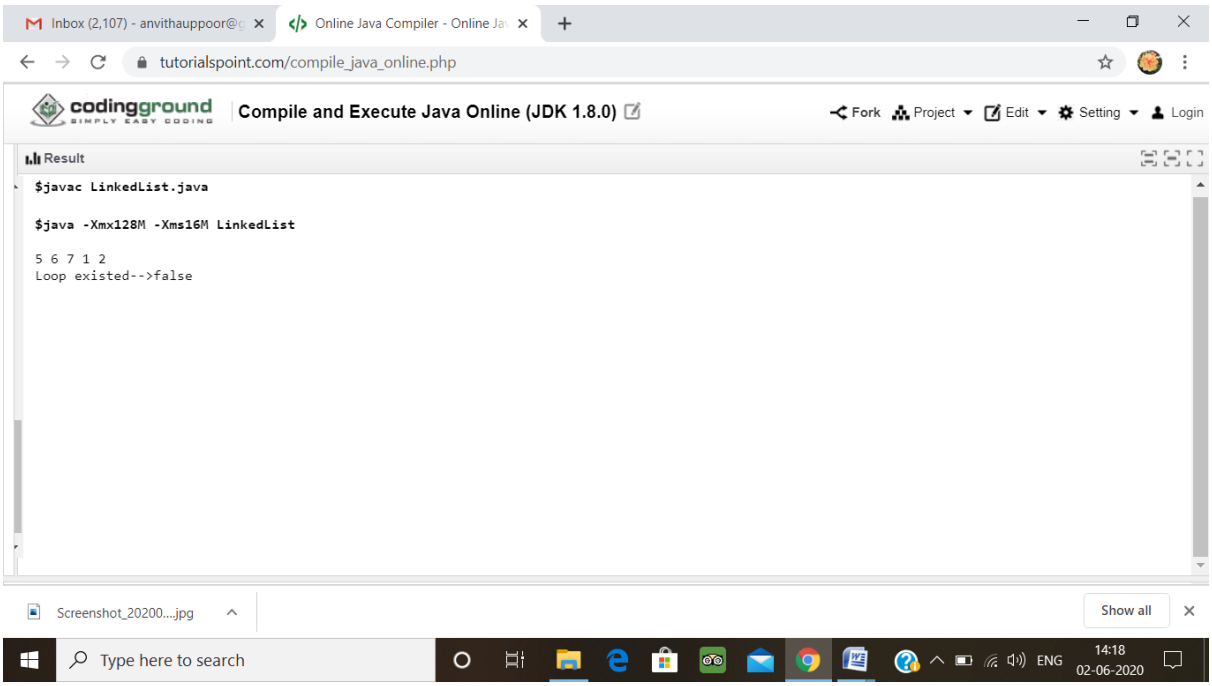
```
public class LinkedList{


        private Node head;


        private static class Node {

                private int value;

                private Node next;
```

```
        Node(int value) {
                this.value = value;

        }
}

public void addToTheLast(Node node) {

        if (head == null) {
                head = node;
        } else {
                Node temp = head;
                while (temp.next != null)
                        temp = temp.next;

                temp.next = node;
        }
}


public void printList() {
        Node temp = head;
        while (temp != null) {
```

```
                System.out.format("%d ", temp.value);
                temp = temp.next;
        }
        System.out.println();
}

public boolean ifLoopExists() {
        Node fastPtr = head;
        Node slowPtr = head;
        while (fastPtr != null && fastPtr.next != null) {
                fastPtr = fastPtr.next.next;
                slowPtr = slowPtr.next;
                if (slowPtr == fastPtr)
                        return true;

        }
        return false;
}

public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.addToTheLast(new Node(5));
        list.addToTheLast(new Node(6));
```

```
        list.addToTheLast(new Node(7));
        list.addToTheLast(new Node(1));
        list.addToTheLast(new Node(2));


        list.printList();
        System.out.println("Loop existed-->" + list.ifLoopExists());

    }
}
```

Output:



3. Given an array of positive integers. Write a C Program to find inversion count of array.

Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.
Input:
The first line of input contains an integer T denoting the number of test cases. The first line of each test case is N, the size of array. The second line of each test case contains N elements.
Output:
Print the inversion count of array.
Constraints:
1 ≤ T ≤ 100
1 ≤ N ≤ 107
1 ≤ C ≤ 1018
Example:
Input:
1
5
2 4 1 3 5
Output:
3
Explanation:
Testcase 1: The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3)

```
#include <stdio.h>

int getInvCount(int arr[], int n)

{

int i,j;

   int inv_count = 0;

   for ( i = 0; i < n - 1;i++)

      for ( j = i + 1; j < n; j++)

         if (arr[i] > arr[j])

            inv_count++;


   return inv_count;
```

```
}

int main(int argv, char** args)

{

    int arr[] = { 2,4,1,3,5 };

    int n = sizeof(arr) / sizeof(arr[0]);

    printf(" Number of inversions are %d \n", getInvCount(arr, n));

    return 0;

}
```

Output: