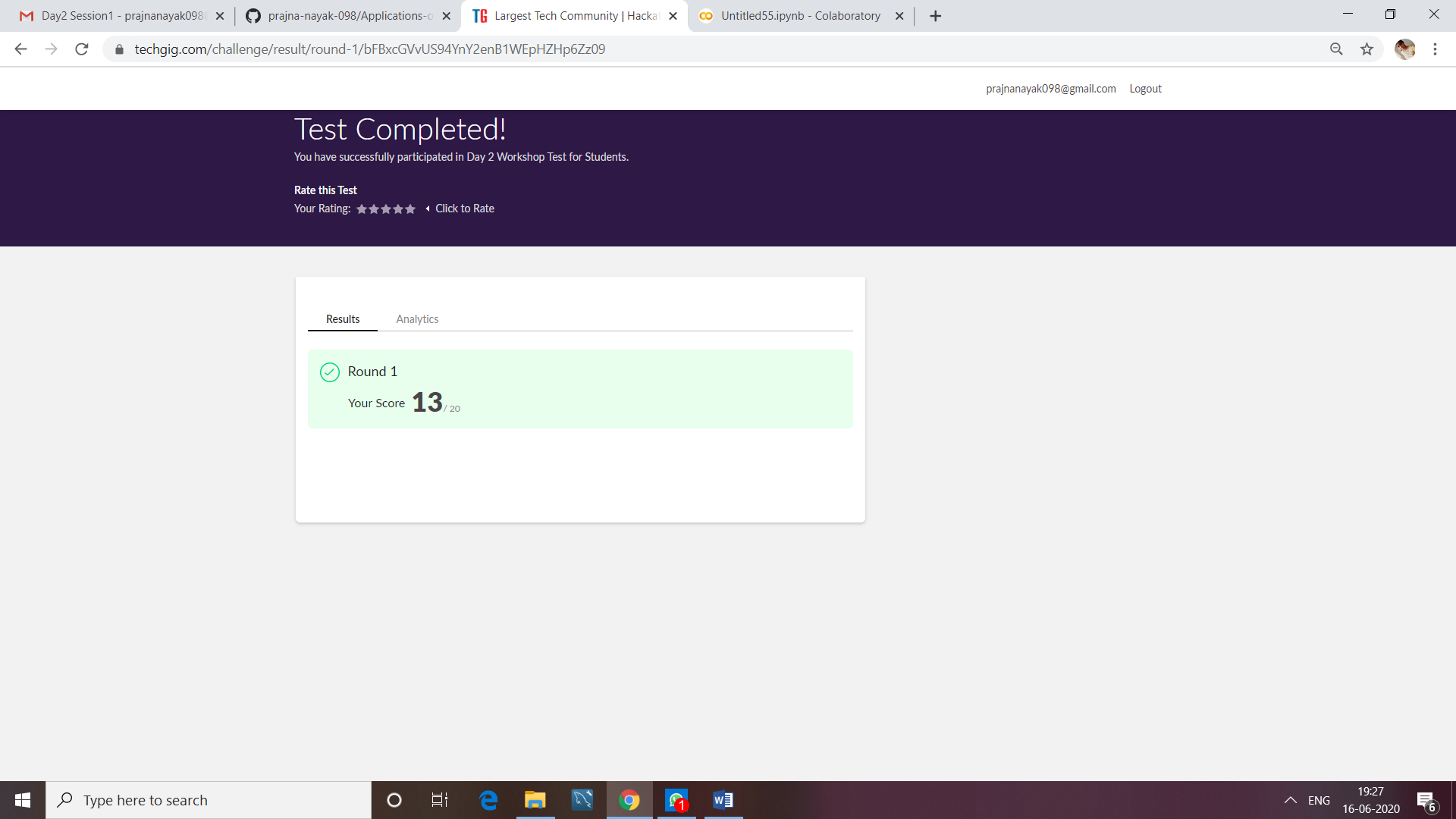
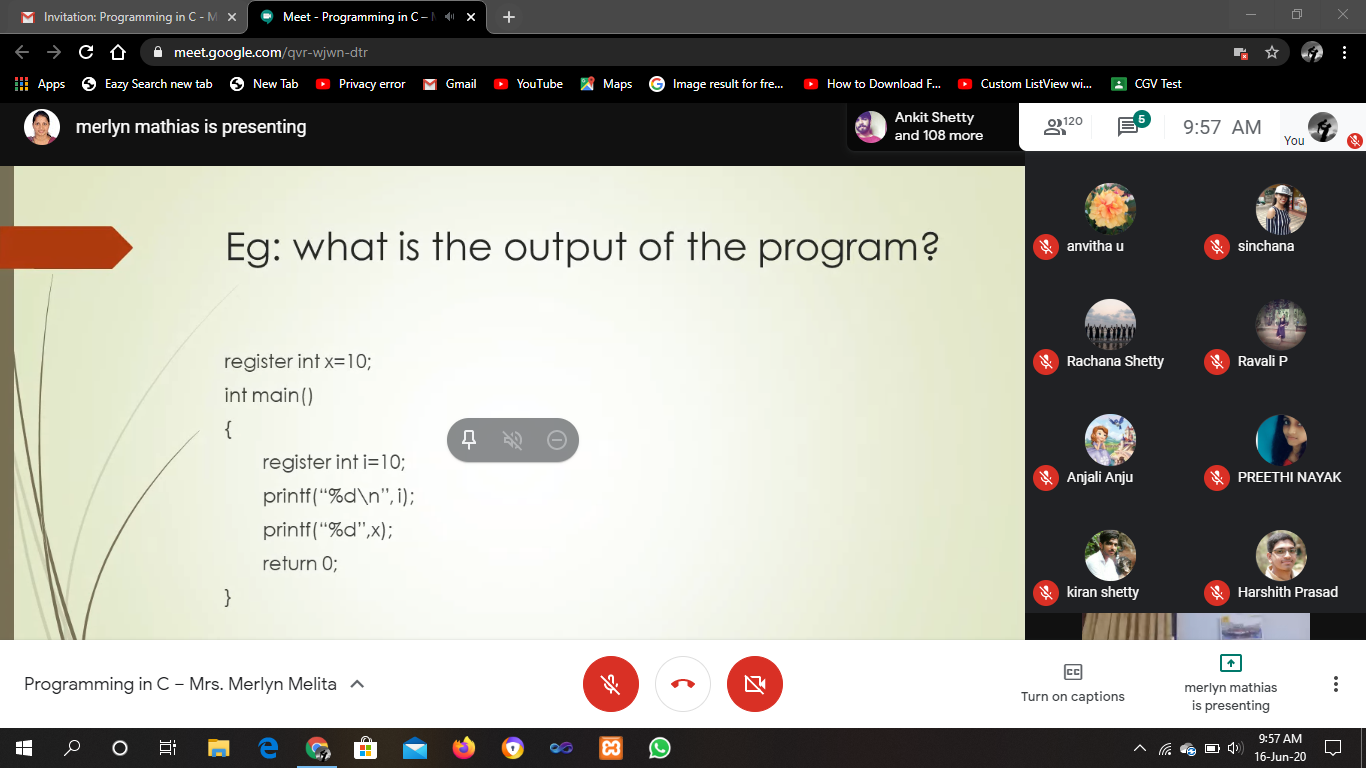
**DAILY ONLINE ACTIVITIES SUMMARY**

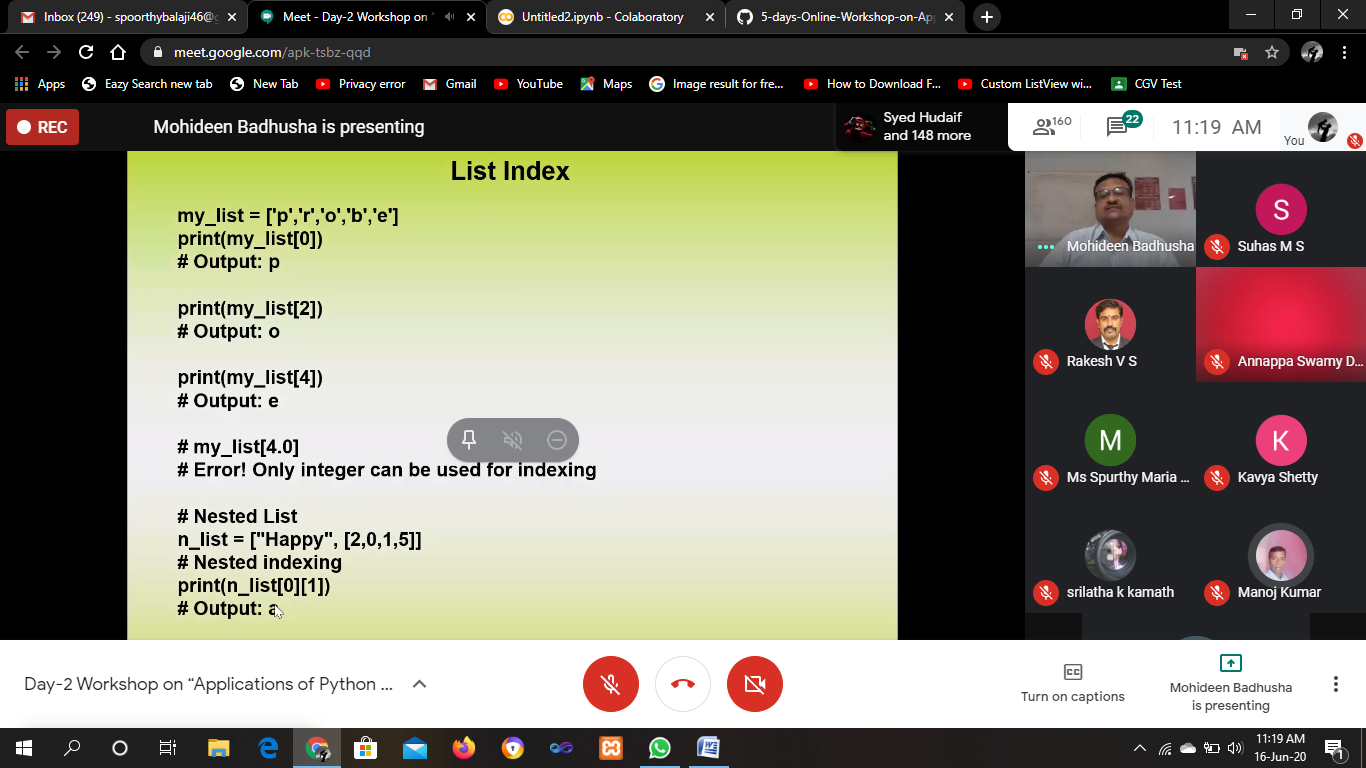
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **16/06/2020** | | | | | **Name:** | **Prajna** | |
| **Sem & Sec** | **6th & A** | | | | | **USN:** | **4al17cs059** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Python (workshop quiz)** | | | | | | |
| **Max. Marks** | | **20** | | **Score** | | | **13** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | 9:00 am to 11:00 am - Programming in C  11:00 am to 1:15pm - Applications of python in DA and ML | | | | | | | |
| **Certificate Provider** | | | **Merlyn Mathias**  **Dr. Mohideen**  **Badusha** | | **Duration** | | | **4hours** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:** 2 Programs | | | | | | | | |
| **Status: Solved** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | **Online Coding Repository**  <https://github.com/prajna-nayak-098/Daily-Report>  **Programming in C Repository**  <https://github.com/prajna-nayak-098/prgm-to-count-distinct-element>  **Workshop on Application of python**  <https://github.com/prajna-nayak-098/Applications-of-python-in-DA-and-ML1> | | | |
| **Uploaded the report in slack** | | | | | **yes** | | | |

Python (Workshop Quiz):



**SNAPSHOTS**

****

****

ONLINE CODING

**1. Python Program to Calculate the Number of Words and the Number of Characters Present in a String**

string=input("Enter string:")

char=0

word=1

for i in string:

char=char+1

if(i==' '):

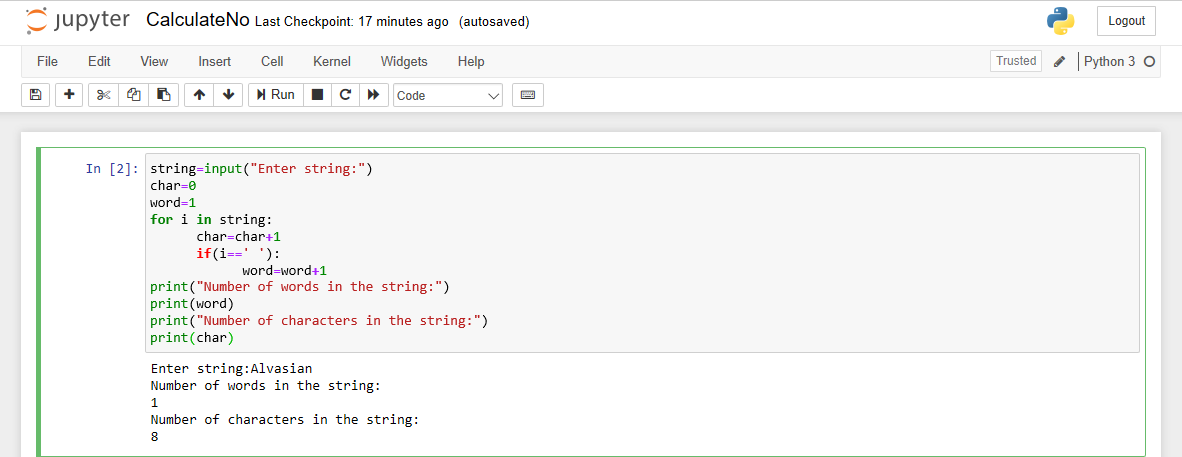
word=word+1

print("Number of words in the string:")

print(word)

print("Number of characters in the string:")

print(char)



**2. Write a Python program to check whether a given a binary tree is a valid binary search tree (BST) or not?**

class TreeNode(object):

    def \_\_init\_\_(self, x):

        self.val = x

        self.left = None

        self.right = None

def is\_BST(root):

    stack = []

    prev = None

    while root or stack:

        while root:

            stack.append(root)

            root = root.left

        root = stack.pop()

        if prev and root.val <= prev.val:

            return False

        prev = root

        root = root.right

    return True

root = TreeNode(2)

root.left = TreeNode(1)

root.right = TreeNode(3)

result = is\_BST(root)

print(result)

root = TreeNode(1)

root.left = TreeNode(2)

root.right = TreeNode(3)

result = is\_BST(root)

print(result)

