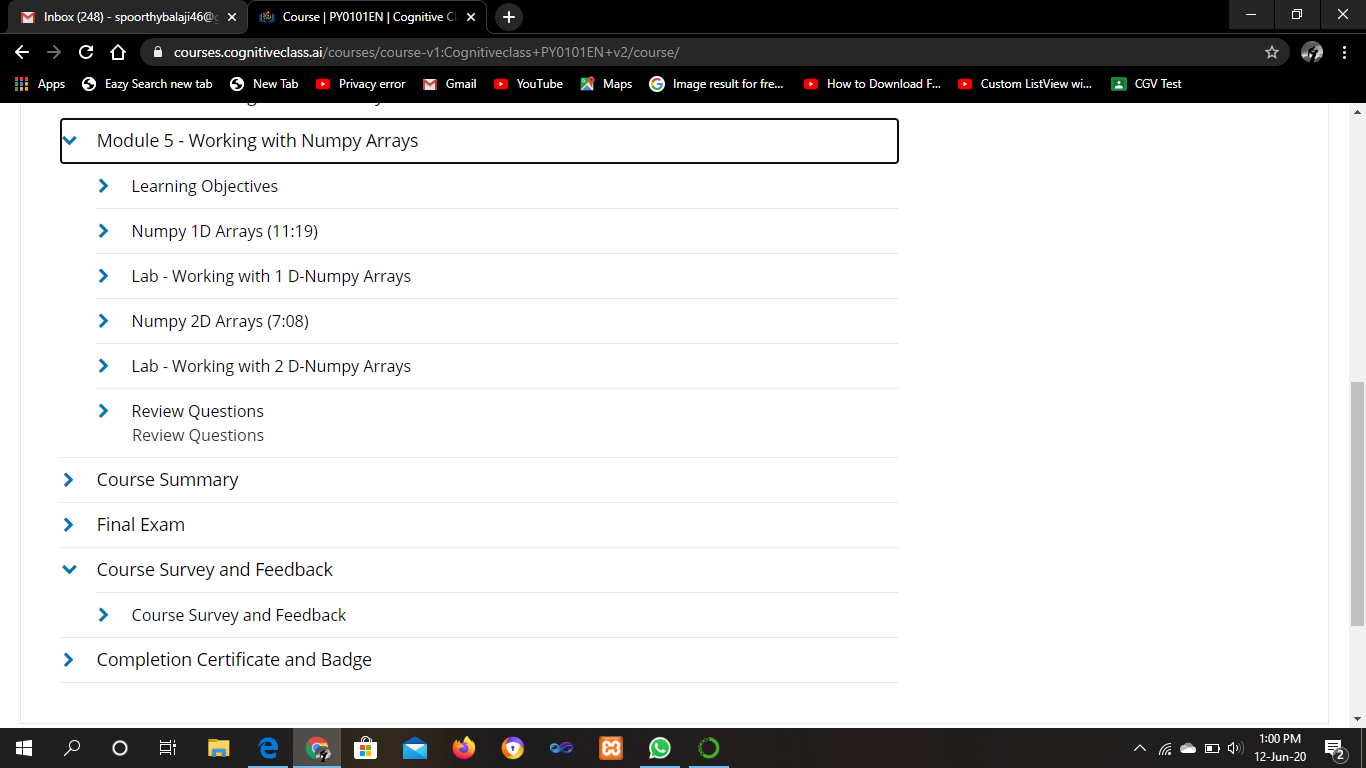
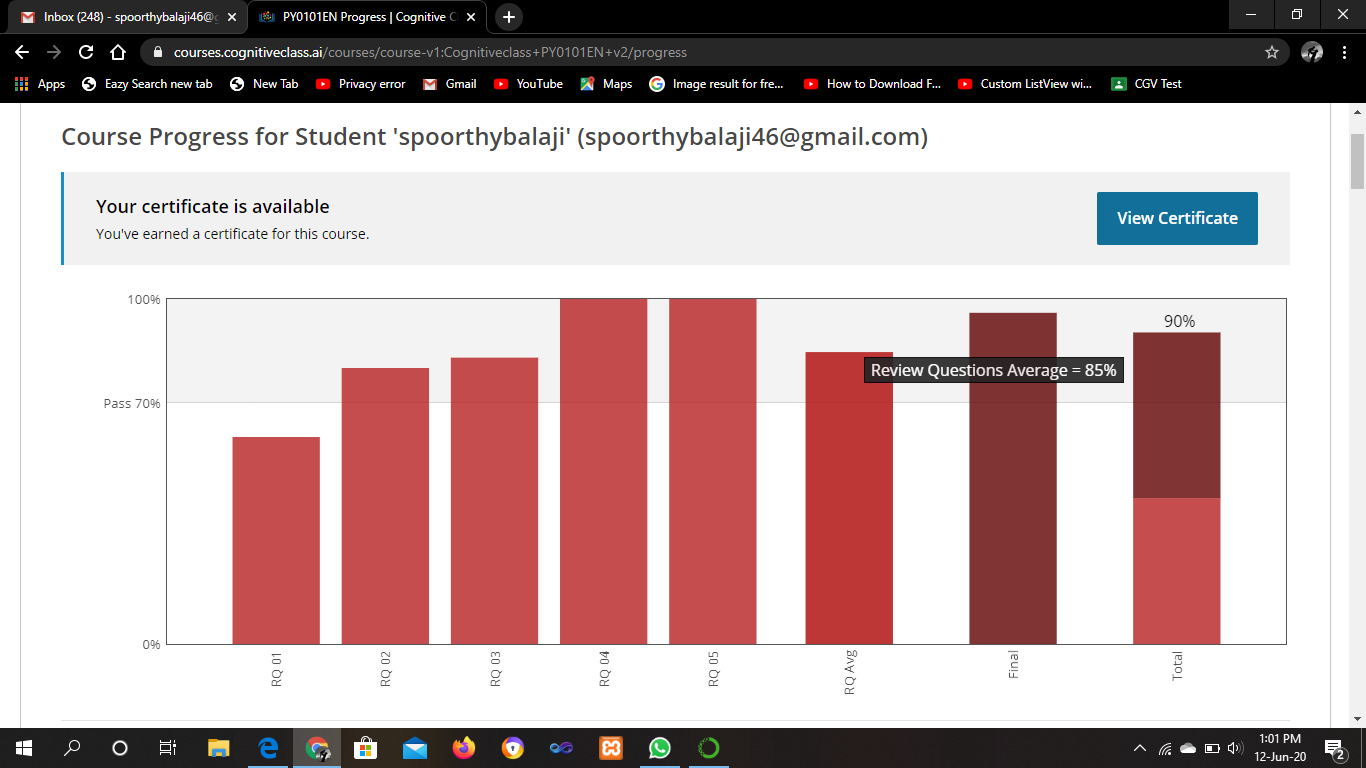
**DAILY ONLINE ACTIVITIES SUMMARY**

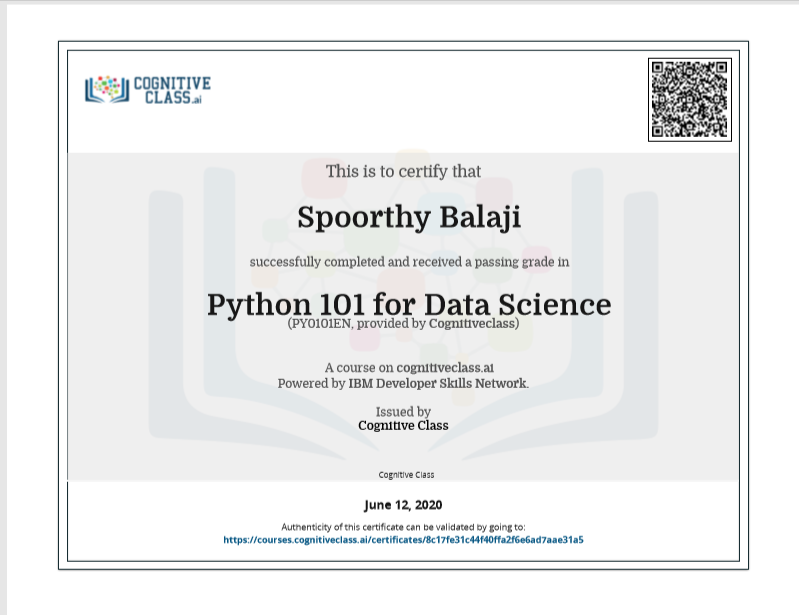
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **12/06/2020** | | | | | **Name:** | **Spoorthy Balaji** | |
| **Sem & Sec** | **6th & B** | | | | | **USN:** | **4al17cs098** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | |  | | | | | | |
| **Max. Marks** | |  | | **Score** | | |  | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Python for Data Science** | | | | | | | |
| **Certificate Provider** | | | **COGNITIVE CLASS.ia** | | **Duration** | | | **9hours** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:** 4 Programs | | | | | | | | |
| **Status: Solved** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **yes** | | | |
| **If yes Repository name** | | | | | <https://github.com/spoorthybalaji/Daily_Status> | | | |
| **Uploaded the report in slack** | | | | | **yes** | | | |

CERTIFICATION COURSE









ONLINE CODING

**1. Python program to Count the Number of Vowels Present in a String using Sets**

**Step1: Get a string**

**Step2: find the no of occurrence of all vowels in the string using data structure set**

def vowel\_count(str):

count = 0

vowel = set("aeiouAEIOU")

for alphabet in str:

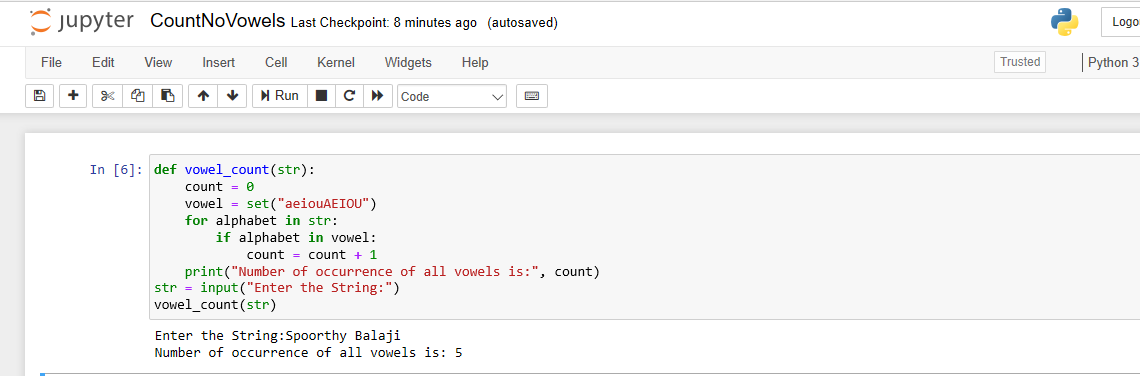
if alphabet in vowel:

count = count + 1

print("Number of occurrence of all vowels is:", count)

str = input("Enter the String:")

vowel\_count(str)



**2. Write a Java program to find maximum width of a binary tree**

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** BinaryTree {

**public** **static** **class** Node{

**int** data;

        Node left;

        Node right;

**public** Node(**int** data){

**this**.data = data;

**this**.left = **null**;

**this**.right = **null**;

        }

      }

**public** Node root;

**public** BinaryTree(){

        root = **null**;

      }

**public** **int** findMaximumWidth() {

**int** maxWidth = 0;

**int** nodesInLevel = 0;

          Queue<Node> queue = **new** LinkedList<Node>();

**if**(root == **null**) {

              System.out.println("Tree is empty");

**return** 0;

          }

**else** {

              //Add root node to queue as it represents the first level

              queue.add(root);

**while**(queue.size() != 0) {

                  nodesInLevel = queue.size();

                  maxWidth = Math.max(maxWidth, nodesInLevel);

**while**(nodesInLevel > 0) {

                     Node current = queue.remove();

**if**(current.left != **null**)

                         queue.add(current.left);

**if**(current.right != **null**)

                         queue.add(current.right);

                     nodesInLevel--;

                  }

              }

          }

**return** maxWidth;

      }

**public** **static** **void** main(String[] args) {

          BinaryTree bt = **new** BinaryTree();

          //Add nodes to the binary tree

          bt.root = **new** Node(1);

          bt.root.left = **new** Node(2);

          bt.root.right = **new** Node(3);

          bt.root.left.left = **new** Node(4);

          bt.root.left.right = **new** Node(5);

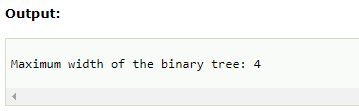
          bt.root.right.left = **new** Node(6);

          bt.root.right.right = **new** Node(7);

          bt.root.left.left.left = **new** Node(8);

          System.out.println("Maximum width of the binary tree: " + bt.findMaximumWidth());

      }  }



3. Write a Python program to implement Magic Square

def generateSquare(n):

    magicSquare = [[0 for x in range(n)]

                      for y in range(n)]

    i = n / 2

    j = n - 1

    num = 1

    while num <= (n \* n):

        if i == -1 and j == n:

            j = n - 2

            i = 0

        else:

            if j == n:

                j = 0

            if i < 0:

                i = n - 1

        if magicSquare[int(i)][int(j)]:

            j = j - 2

            i = i + 1

            continue

        else:

            magicSquare[int(i)][int(j)] = num

            num = num + 1

        j = j + 1

        i = i - 1

    print ("Magic Square for n =", n)

    print ("Sum of each row or column",n \* (n \* n + 1) / 2, "\n")

    for i in range(0, n):

        for j in range(0, n):

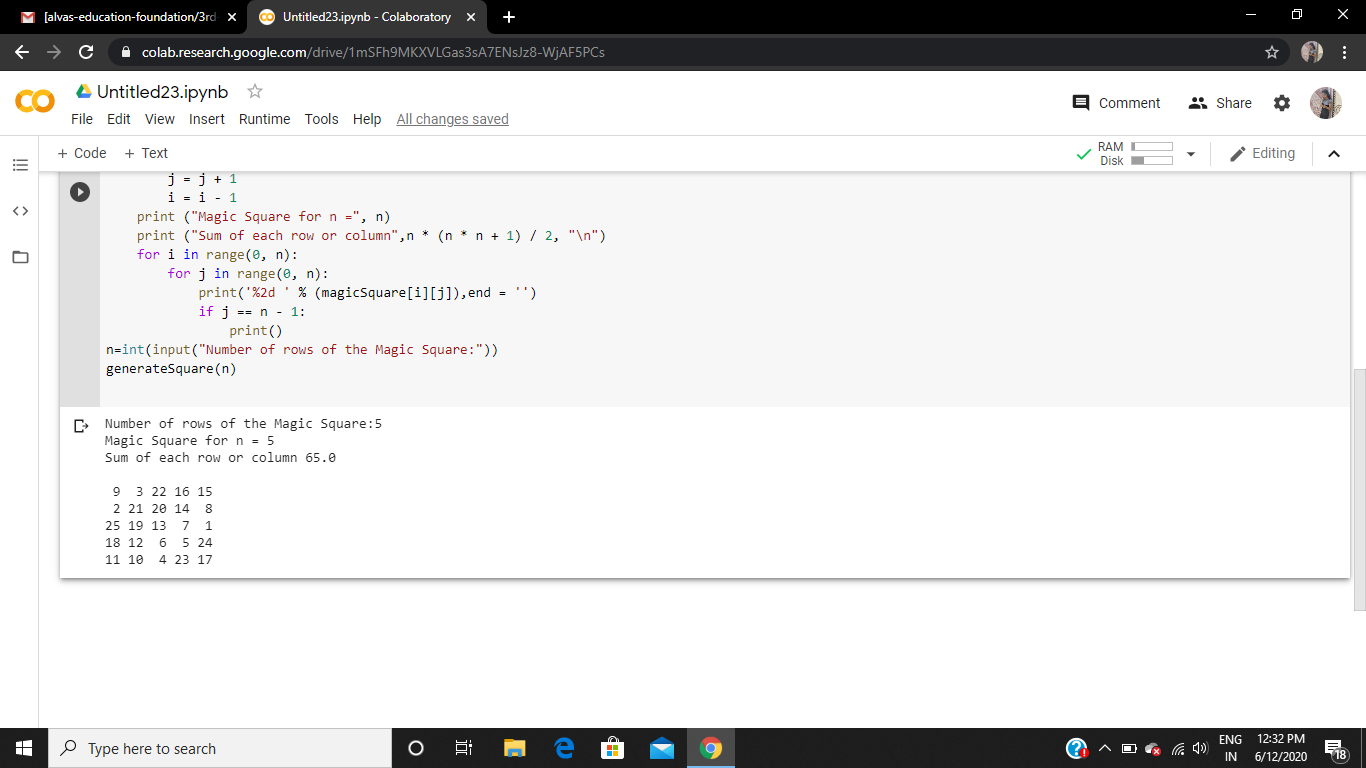
            print('%2d ' % (magicSquare[i][j]),end = '')

            if j == n - 1:

                print()

n=int(input("Number of rows of the Magic Square:"))

generateSquare(n)



**4. Python program to print the pattern:**

def pattern(n):

k = 2 \* n - 2

for i in range(0, n-1):

for j in range(0, k):

print(end=" ")

k = k - 2

for j in range(0, i + 1):

print("\* ", end="")

print("\r")

k = -1

for i in range(n-1,-1,-1):

for j in range(k,-1,-1):

print(end=" ")

k = k + 2

for j in range(0, i + 1):

print("\* ", end="")

print("\r")

n=int(input("enter n value"))

pattern(n)

