**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **12-06-2020** | | | | | **Name:** | **Sneha K Bakale** | |
| **Sem & Sec** | **6th B** | | | | | **USN:** | **4al17cs095** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **-** | | | | | | |
| **Max. Marks** | | **-** | | **Score** | | | **-** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **Ethical hacking** | | | | | | | |
| **Certificate Provider** | | | **Great learning Academy** | | **Duration** | | | **6.0 hours** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement: Programs given** | | | | | | | | |
| **Status: Completed** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **Yes** | | | |
| **If yes Repository name** | | | | | **https://github.com/Sneha35/OnlineCourse-And-Coding.git** | | | |
| **Uploaded the report in slack** | | | | | **Yes** | | | |

Online Test Details: (Attach the snapshot and briefly write the report for the same)

Certification Course Details: (Attach the snapshot and briefly write the report for the same)

Coding Challenges Details: (Attach the snapshot and briefly write the report for the same)

**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.**

s = input("Enter The String: ")

v = set("aeiouAEIOU")

c = 0

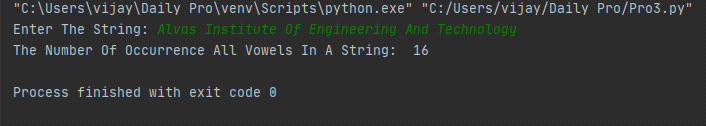
for i in s:

if i in v:

c += 1

print("The Number Of Occurrence All Vowels In A String: ", c)

**Output:**



**2. A magic square of order n is an arrangement of n^2 numbers, usually distinct integers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A magic square contains the integers from 1 to n^2.**

def gSqua(n):

magic\_square = [[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 magic\_square[int(i)][int(j)]:

j = j - 2

i = i + 1

continue

else:

magic\_square[int(i)][int(j)] = num

num = num + 1

j = j + 1

i = i - 1

print("Magic Squre For n =", n)

print('Sum Of Each Row Or Column Or Diagonal: ', n \* (n \* n + 1) / 2, "\n")

for i in range(0, n):

for j in range(0, n):

print('%2d ' % (magic\_square[i][j]), end='')

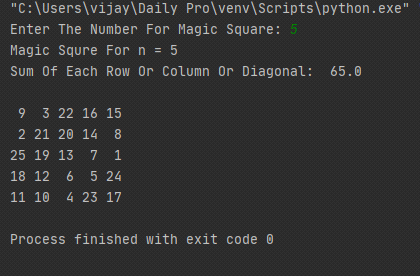
if j == n - 1:

print()

n = int(input("Enter The Number For Magic Square: "))

gSqua(n)

**Output:**



**3. Python program to print the pattern.**

n = int(input("Enter The N Value: "))

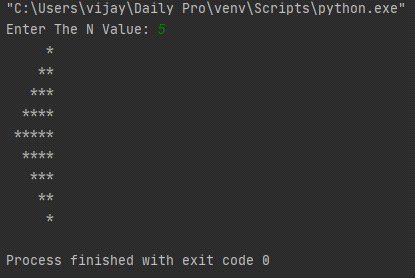
for i in range(1, n + 1):

print(" " \* (n - i), "\*" \* i, end="\n")

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

print(" " \* (n - i), "\*" \* i, end="\n")

**Output:**



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

**class** Node

{

**int** data;

Node left, right;

Node(**int** item)

{

data = item;

left = right = **null**;

}

}

**class** BinaryTree

{

Node root;

**int** getMaxWidth(Node node)

{

**int** maxWidth = 0;

**int** width;

**int** h = height(node);

**int** i;

**for** (i = 1; i <= h; i++)

{

width = getWidth(node, i);

**if** (width > maxWidth)

maxWidth = width;

}

**return** maxWidth;

}

**int** getWidth(Node node, **int** level)

{

**if** (node == **null**)

**return** 0;

**if** (level == 1)

**return** 1;

**else** **if** (level > 1)

**return** getWidth(node.left, level - 1)

+ getWidth(node.right, level - 1);

**return** 0;

}

**int** height(Node node)

{

**if** (node == **null**)

**return** 0;

**else**

{

**int** lHeight = height(node.left);

**int** rHeight = height(node.right);

**return** (lHeight > rHeight) ? (lHeight + 1) : (rHeight + 1);

}

}

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

{

BinaryTree tree = **new** BinaryTree();

tree.root = **new** Node(1);

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

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

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

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

tree.root.right.right = **new** Node(18);

tree.root.right.right.left = **new** Node(9);

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

System.***out***.println("Maximum width is: " + tree.getMaxWidth(tree.root));

}

}

**Output:**

