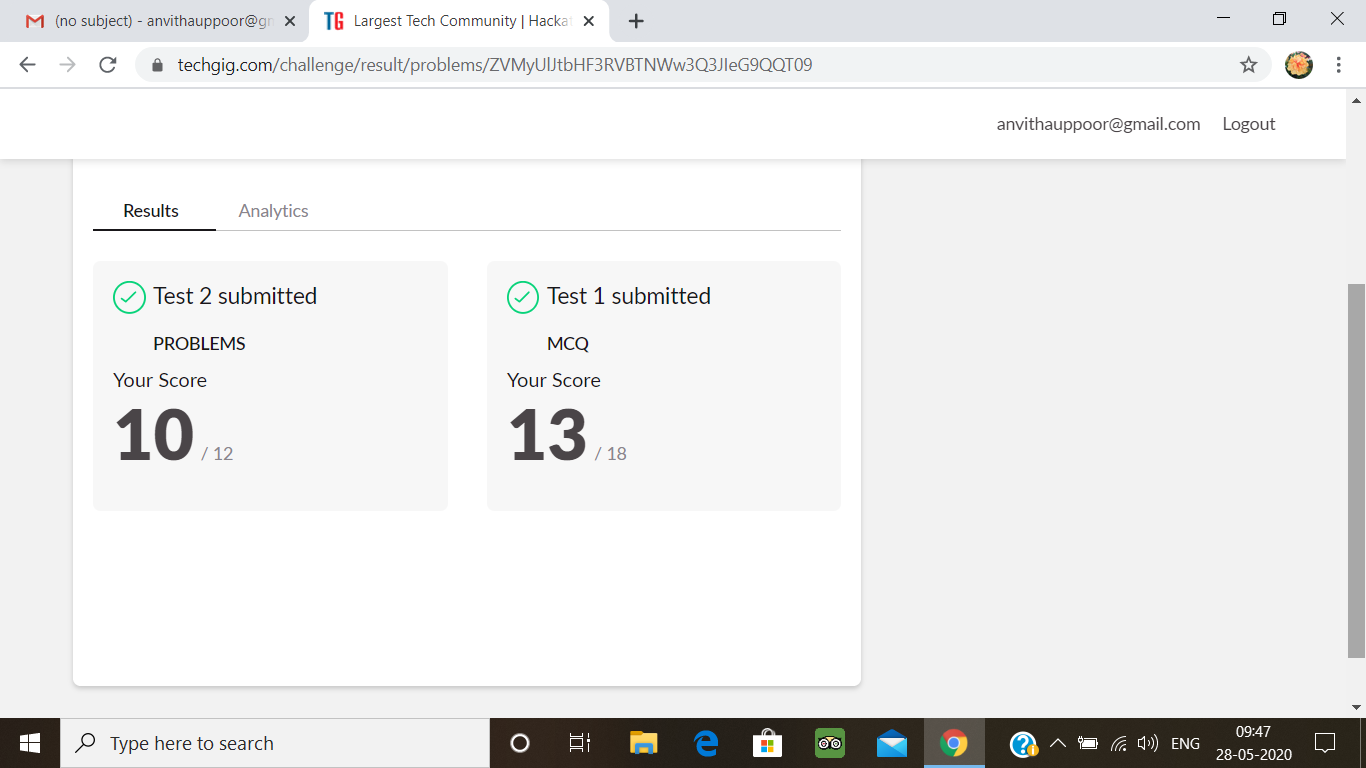
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **28-05-2020** | | | | | **Name:** | **Anvitha U** | |
| **Sem & Sec** | **A** | | | | | **USN:** | **4AL17CS009** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Operating System.** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **23** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **INTRODUCTION TO CLOUD** | | | | | | | |
| **Certificate Provider** | | | COGNITIVE  CLASS | | **Duration** | | | 2hours |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:**1**.** Python program to find digital root of a number.  2. JAVA PROGRM-BALANCED BRAKET. | | | | | | | | |
| **Status: Done** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | <https://github.com/anvithauppoor/online_coding_activity> | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

Online Test Details:

Subject:-Operating System.

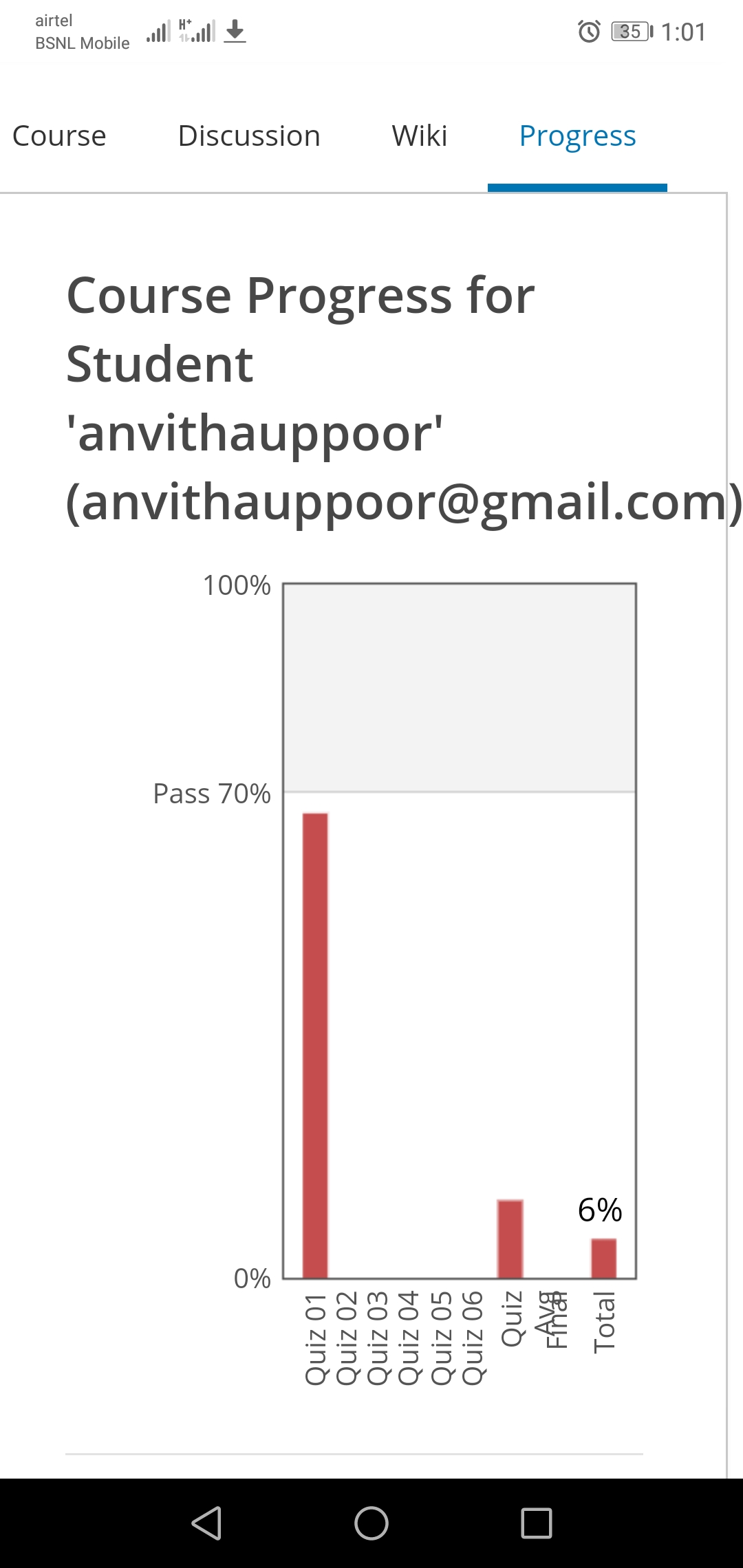


Certification Course Details:

**Introduction to Cloud:**

Today I have studied **:**

* Overview of cloud computing
* Definition and Essential Characteristics of cloud Computing
* History and Evolution of Cloud Computing
* Key Considerations for Cloud Computing
* Key Cloud Service providers and their services
* Hands-on lab: create a cloud account
* Attended module quiz exam



Coding Challenges Details:

1.Python program to find digital root of a number

Description:  
A digital root is the recursive sum of all the digits in a number. Given n, take the sum of the digits of n. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. This is only applicable to the natural numbers.  
digit\_root(0)= 0

digital\_root(16)  
=> 1 + 6  
=> 7

digital\_root(132189)  
=> 1 + 3 + 2 + 1 + 8 + 9  
=> 24 ...  
=> 2 + 4  
=> 6

def DigitalRoot(number):

addper = 0

while number >=10:

number = sum(int(digit)for digit in str(number))

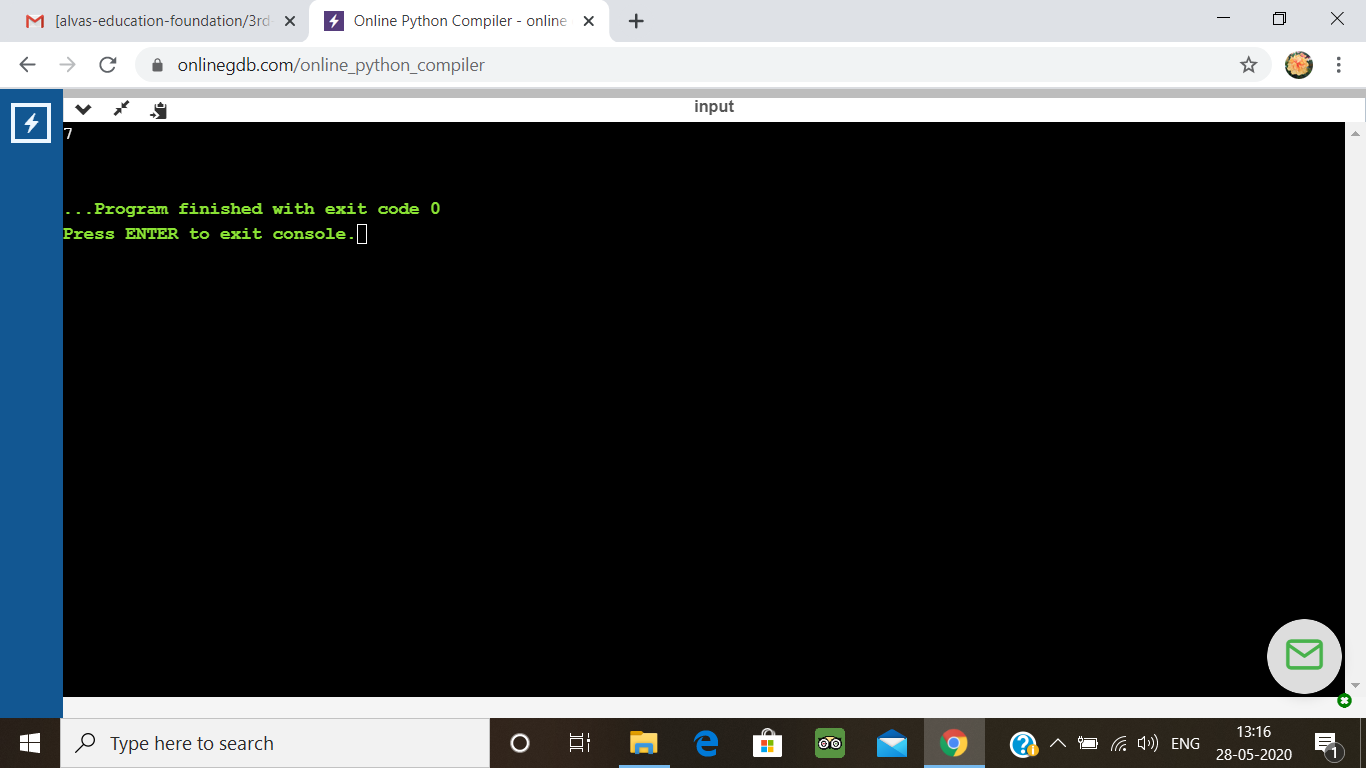
addper +=1

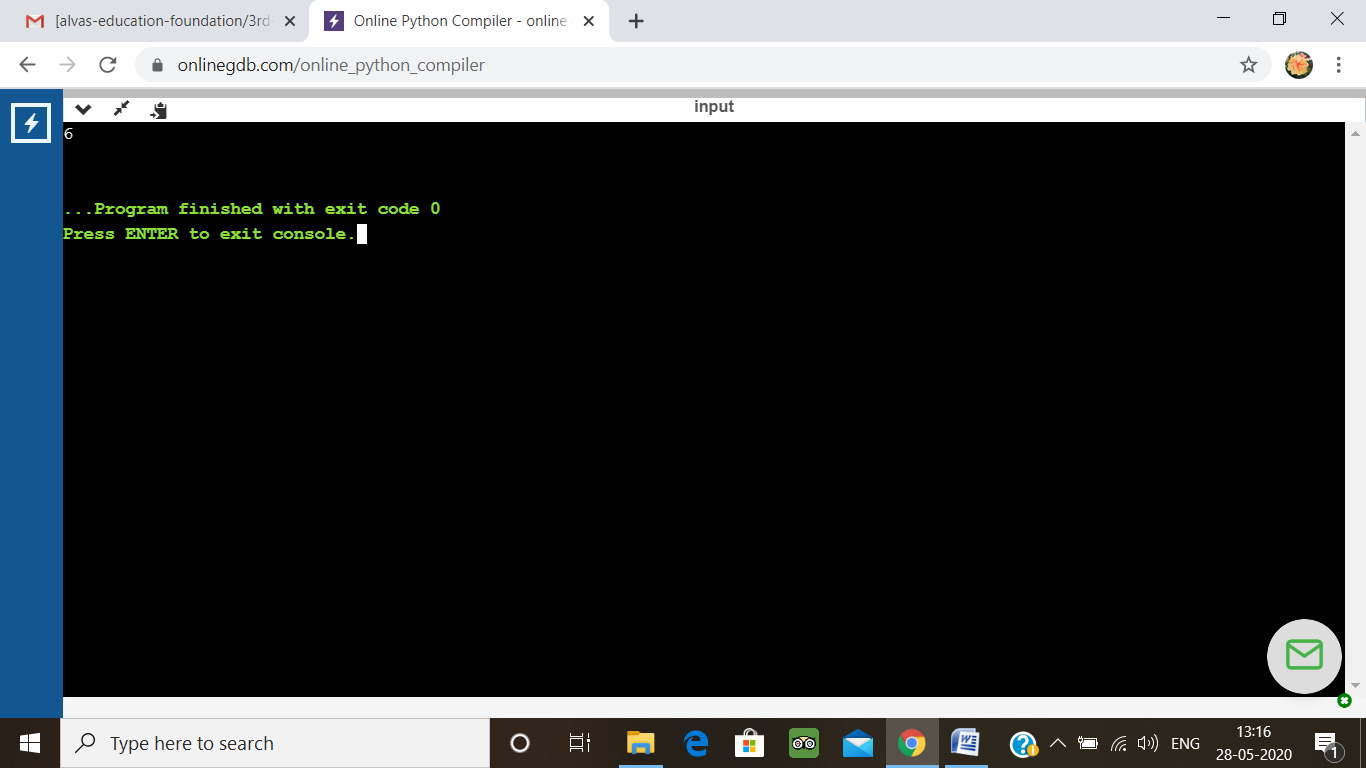
#I highly recommend using return instead of print, but for testing purposes I used print

print(number)

DigitalRoot(132189)

**Output:**





2. JAVA PROGRM-BALANCED BRAKET

Write a function that accepts a string consisting entiring of brackets ({}) and returns whether it is balanced. Every "opening" bracket must be followed by a closing bracket of the same type. There can also be nested brackets, which adhere to the same rule.  
f('()[]{}(([])){[()][]}') // true  
f('())[]{}') // false

import java.util.Stack;

public class Main {

public static void main(String[] args) {

System.out.println(is\_parentheses\_balanced("()[]{}(([])){[()][]}"));

}

public static boolean matchingPeer(char open , char close){

if ( open == '(' && close == ')'){

return true;

}

if ( open == '[' && close == ']'){

return true;

}

else{

return false;

}

}

public static boolean is\_parentheses\_balanced(String equation){

char[] c = equation.toCharArray();

Stack <Character> myStack= new Stack <Character> ();

for (int i = 0; i < c.length; i++){

if(c[i]=='(' || c[i] == '[' ){

myStack.push(c[i]);

}

else if (c[i]== ')' || c[i]==']'){

if(matchingPeer(myStack.peek(),c[i]) == true){

myStack.pop();

} else {

return false;

}

}

}

if(myStack.isEmpty()){

return true;

}

else {

return false;

}

}

}

**Output:**

