DAILY ONLINE ACTIVITIES SUMMARY

Date:	**28/05/2020**	**Name:**	**ASHIKA**
Sem & Sec	**6 A**	**USN:**	**4AL17CS016**

Online Test Summary

Subject	**OPERATING SYSTEM**		
Max. Marks	**30**	**Score**	**20**

Certification Course Summary

Course	**ETHICAL HACKING**		
Certificate Provider	**Great learninng**	**Duration**	**6 hour**

Coding Challenges

Problem Statement:

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

2. JAVA PROGRM-BALANCED BRAKET

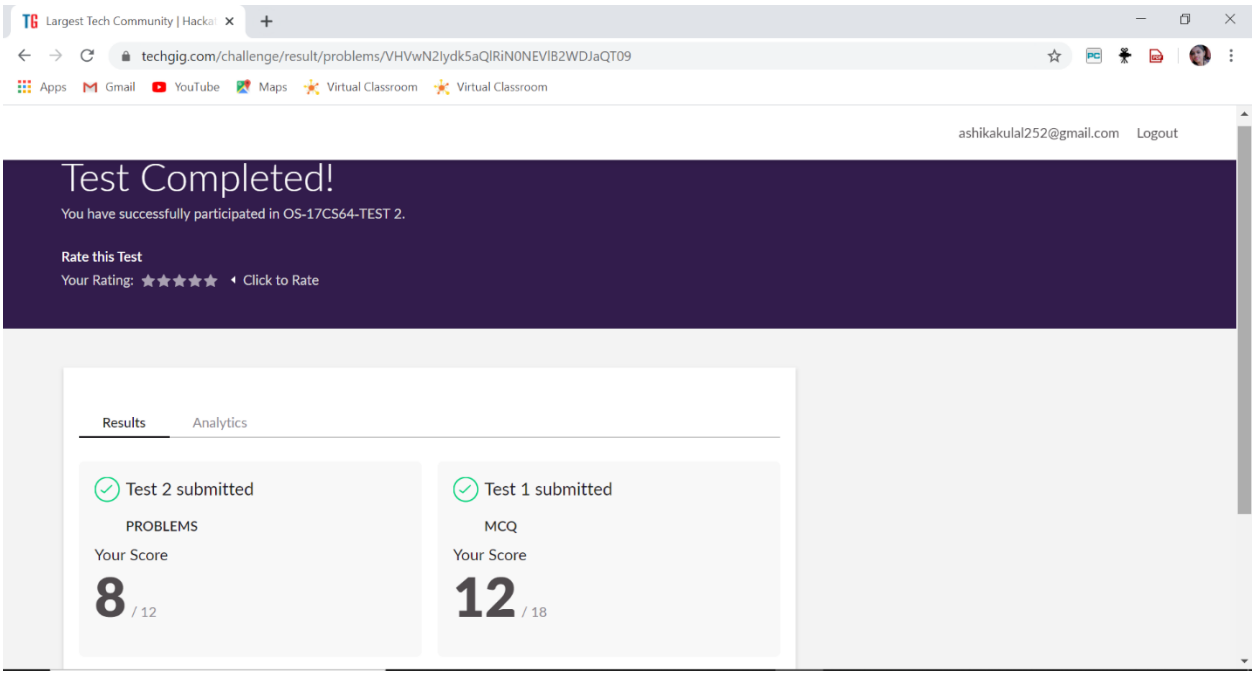
Write a function that accepts a string consisting entiring of brackets ({}) and returns whether it is balanced. Every "opening" bracket must be followed by a closing bracket of the same type. There can also be nested brackets, which adhere to the same rule.

f('()[]{}(([])){[()][]}') // true

f('())[]{}') // false

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: OPERATING SYSTEM

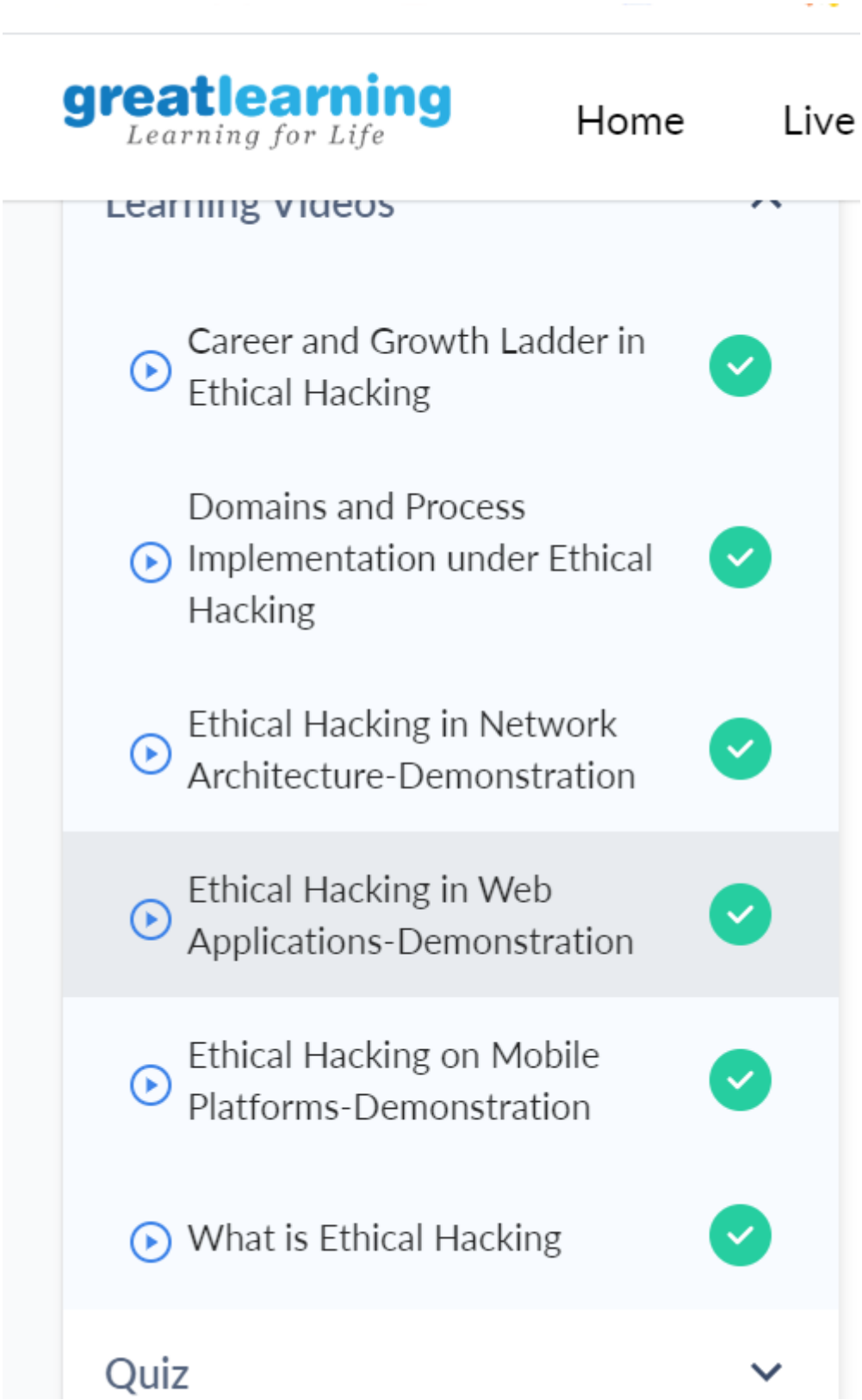


CERTIFICATION COURSE

- Career and Growth Ladder in Ethical Hacking
- Domains and Process Implementation under Ethical Hacking
- Ethical Hacking in Network Architecture-Demonstration
- Ethical Hacking in Web Applications-Demonstration
- Ethical Hacking on Mobile Platforms-Demonstration
- What is Ethical Hacking

Quiz

Claim Your Course Certificate



ONLINE CODEING

1.Python program to find digital root of a number

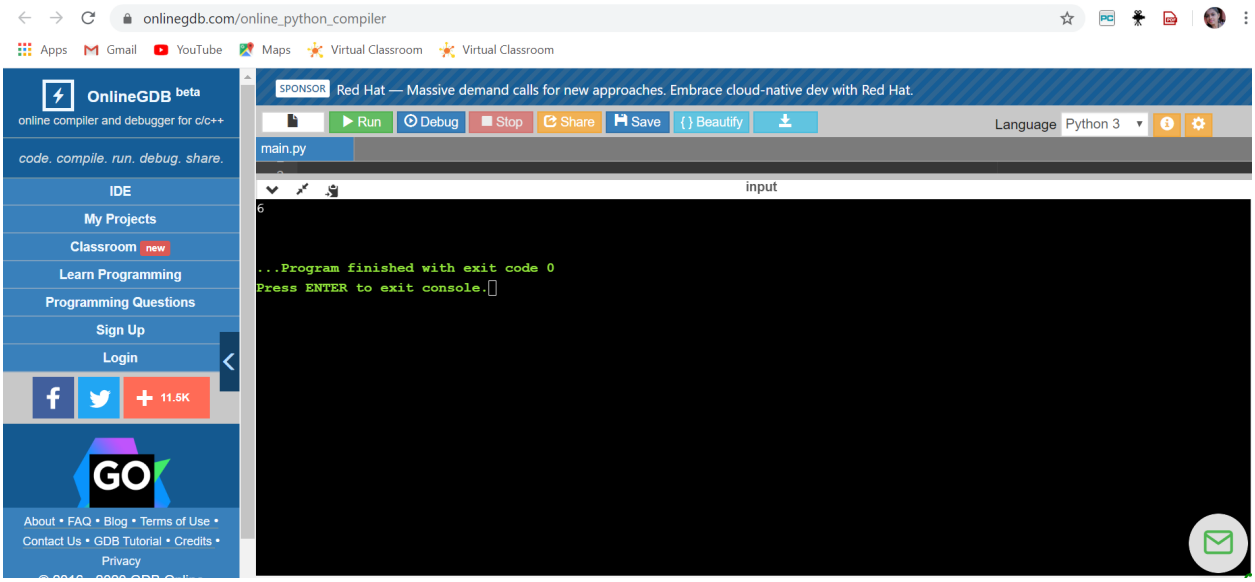
Description:
A digital root is the recursive sum of all the digits in a number. Given n, take the sum of the digits of n. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. This is only applicable to the natural numbers.
digit_root(0)= 0

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: