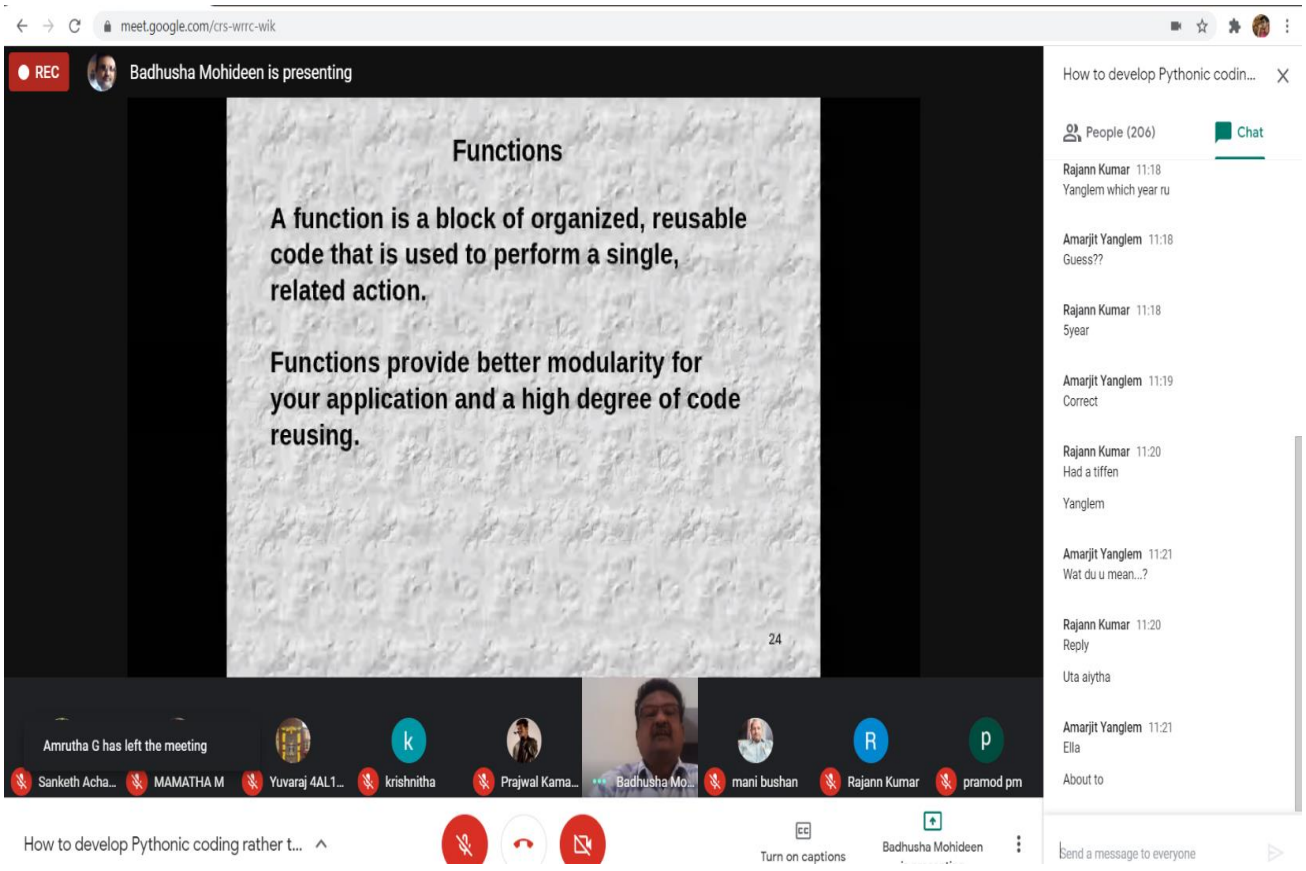
DAILY ASSESSMENT REPORT

Date:	**21 July 2020**	**Name:**	**Gagan M K**
Course:	**How to develop Pythonic coding – Logic Perspective**	**USN:**	**4AL17EC032**
Topic:	• **Day 1**	**Semester & Section:**	**6th sem & 'A' sec**
GitHub Repository:	**Alvas-education-foundation/Gagan-Git**		

FORENOON SESSION DETAILS

Image of session



Report – Report can be typed or hand written for up to two pages.

Python:

Open in Colab

In [ ]:
```
#Python Program to Add Two Numbers getting through key board

# sum of two nos
num1 = int(input("Enter first no"))
num2 = int(input("Enter second no"))

# Adding the two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

```
Enter first no3
Enter second no2
The sum of 3 and 2 is 5
```

Leap Year program

In [ ]:
```
#Python program to check if the input year is a leap year or not

# To get year (integer input) from the user
year = int(input("Enter a year: "))

if ((year % 4) == 0 and (year % 100)  != 0) or ((year % 400) == 0):
            print("{0} is a leap year".format(year))
else:
            print("{0} is not a leap year".format(year))
```

```
Enter a year: 2020
2020 is a leap year
```

Generate a Random Number

In [ ]:
```
#Python Program to Generate a Random Number

# Program to generate a random number between 0 and 9
# import the random module
import random
print(random.randint(0,9))
```

```
2
```

Convert Kilometers to Miles

In [ ]:
```
#Python Program to Convert Kilometers to Miles

# To take kilometers from the user, uncomment the code below
kilometers = int(input("Enter value in kilometers"))

# conversion factor
conv_fac = 0.621371

# calculate miles
miles = kilometers * conv_fac
print('%0.3f kilometers is equal to %0.3f miles' %(kilometers,miles))
```

```
Enter value in kilometers10
10.000 kilometers is equal to 6.214 miles
```

Solve Quadratic Equation

In [ ]:
```
#Python Program to Solve Quadratic Equation

# Solve the quadratic equation ax**2 + bx + c = 0
# importing  complex math module
import cmath

 # To take coefficient input from the users
a = float(input('Enter a: '))
b = float(input('Enter b: '))
c = float(input('Enter c: '))

# calculate the discriminant
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

```
Enter a: 2
Enter b: 4
Enter c: 7
The solution are (-1-1.5811388300841898j) and (-1+1.5811388300841898j)
```

Prime or Not

Open in Colab

1. Program to print the triangle :

i/p 4

1

2 3

3 4 5

4 5 6 7

In [136]:
```
# Function to demonstrate printing pattern of numbers

n=int(input("Enter the Number "));
for i in range(n+1):
    for j in range(i):
        print (j+i ,end=" ")
    print('')
```

```
Enter the Number 4

1
2 3
3 4 5
4 5 6 7
```

2. Star Pattern for n=4

In [131]:
```
# Function to demonstrate printing pattern
def pypart(n):
    for i in range(0, n):
        for j in range(0, i+1):

            print("* ",end="")

        print("\r")

n = int(input("Enter the number: "))
pypart(n)
```

```
Enter the number: 4
*
* *
* * *
* * * *
```

3. Backward Numbers n = 5

5

4 5

3 4 5

2 3 4 5

1 2 3 4 5

In [133]:
```
n=int(input("Enter the Number "));
for i in range(n+1):
    for j in range(i,0,-1 ):
        print (n-j+1 ,end=" ")
    print('')
```

```
Enter the Number 5

5
4 5
3 4 5
2 3 4 5
1 2 3 4 5
```

4. Number Backwords n = 5

5 4 3 2 1

4 3 2 1

3 2 1

2 1

1

- **Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.**
- **Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.**
- **Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.**
- **Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast.**
- **Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.**
- **When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.**
- **The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.**

7. Find the no of words and characters in a string

In [138]:
```
word_count = 0
char_count = 0


usr_input = input("Enter a string : ")

split_string = usr_input.split()

word_count = len(split_string)


for word in split_string:

    char_count += len(word)
print("Total words : {}".format(word_count))
print("Total characters : {}".format(char_count))
```

```
Enter a string : My name is Gagan
Total words : 4
Total characters : 13
```

8. Find the no of occurrences of a word in a string

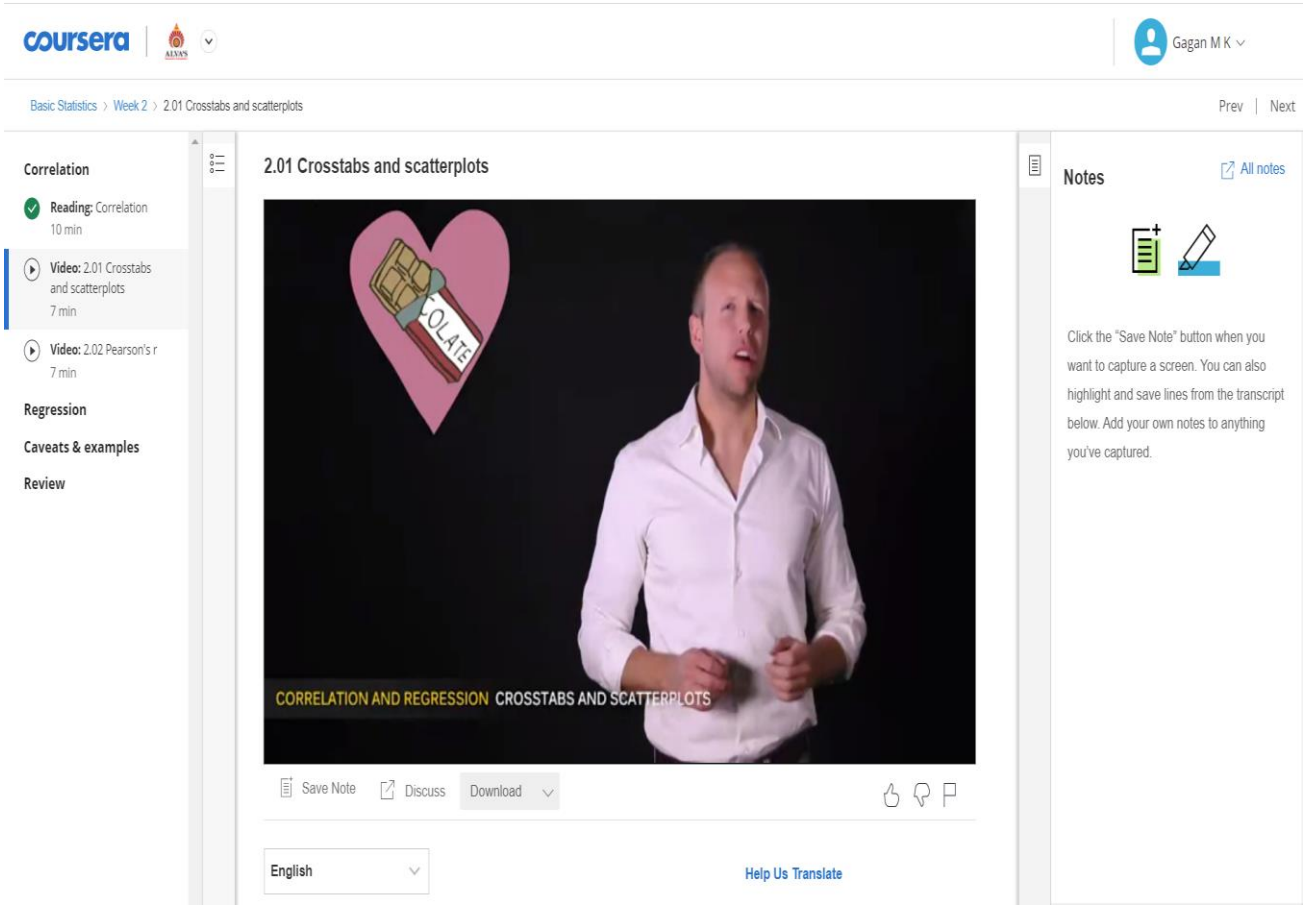
In [139]:
```
# define string
string = "Python is awesome, isn't it?"
substring = "is"

count = string.count(substring)

# print count
print("The count is:", count)
```

```
The count is: 2
```

Date:	21 July 2020	Name:	Gagan M K
Course:	Basic Statistics	USN:	4AL17EC032
Topic:	• Week 2	Semester & Section:	6th sem & 'A' sec

AFTERNOON SESSION DETAILS

Image of session:



Report – Report can be typed or hand written for up to two pages.

- **Many people like eating chocolate. But most people are somewhat cautious with their chocolate consumption. Because it might well be the case that eating a lot of chocolate, increases your body weight.**
- **Play video starting at 16 seconds and follow transcript**
- **In this video today, I'll talk about how we can display a relationship between two variables using tables and using graphs. This can be very useful to help you discover if two variables are correlated or not.**
- **Play video starting at 29 seconds and follow transcript**
- **Let us investigate the relationship between eating chocolate and body weight a little bit further.**
- **Play video starting at 35 seconds and follow transcript**
- **Suppose I have selected 200 female students at my university, who are all one meter seventy tall. This way, height is a constant and cannot account for differences in body weight or chocolate consumption.**
- **Play video starting at 48 seconds and follow transcript**
- **I asked the students to report their body weight and their weekly chocolate consumption. They could choose between the categories less than 50 kilograms, 50 to 69 kilograms, 70 to 89 kilograms, and 90 kilograms or more.**
- **Play video starting at 1 minute 3 seconds and follow transcript**
- **They could indicate their chocolate consumption by choosing less than 50 grams per week, between 50 and 150 grams per week, and more than 150 grams per week.**
- **Play video starting at 1 minute 15 seconds and follow transcript**
- **Here are the results. What you see here is a contingency table. A contingency table, enables you to display the relationship between two ordinal or nominal variables. It is similar to a frequency table. But the major difference is that a frequency table always concerns only one variable, whereas a contingency table concerns two variables.**
- **Play video starting at 1 minute 36 seconds and follow transcript**
- **In our study we have two variables. Body weight and chocolate consumption.**
- **Play video starting at 1 minute 42 seconds and follow transcript**
- **The table shows that we have 33 individuals with a body weight of less than 50 kilograms. 27 of them eat less than 50 grams of chocolate per week.**
- **Play video starting at 1 minute 53 seconds and follow transcript**
- **You can also see, that 90 individuals eat between 50 and 150 grams of chocolate per week. 7 of them weigh 90 kilograms or more.**
- **Play video starting at 2 minutes 4 seconds and follow transcript**
- **In this form, the table does not tell you much yet about a correlation between the two variables, because the columns and rows contain different numbers of cases. It provides more insight, when you compute percentages. And in this case, we commute column percentages. This means that for every cell, we compute a percentage of cases in that cell, compared to the total number of cases in the corresponding column. So for instance, in this cell, we have 24 cases. The total number of cases in the column is 60**