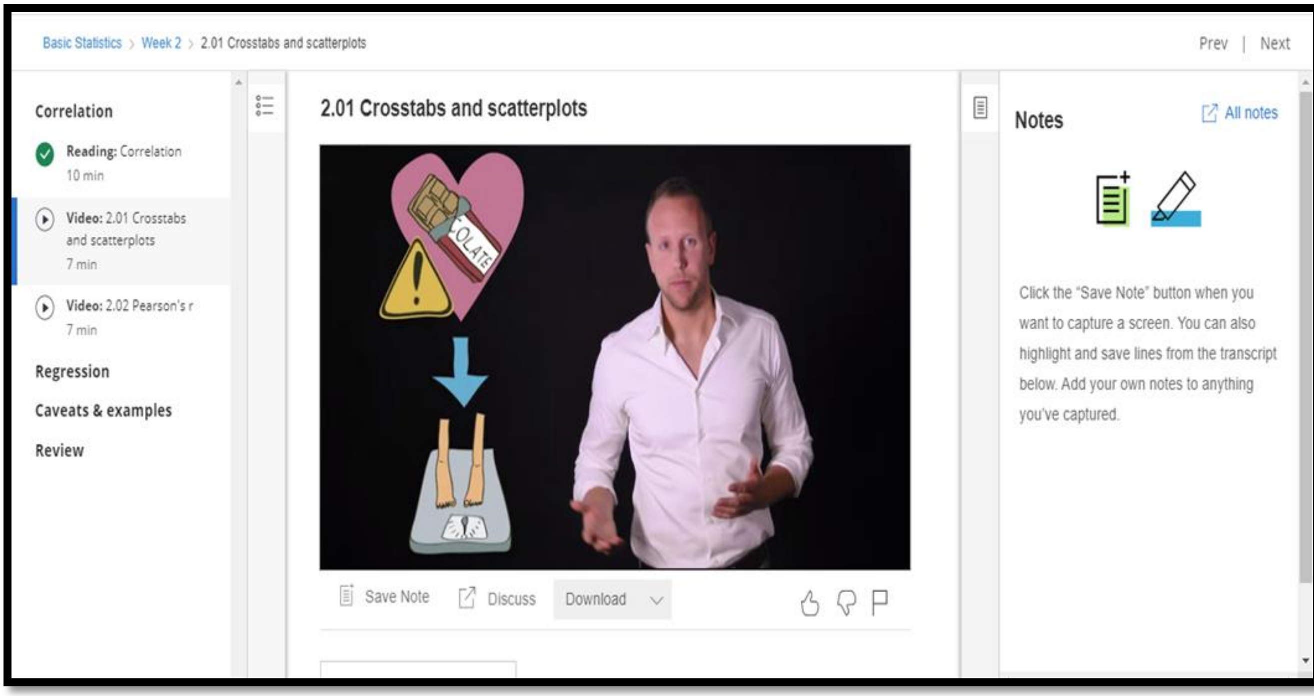
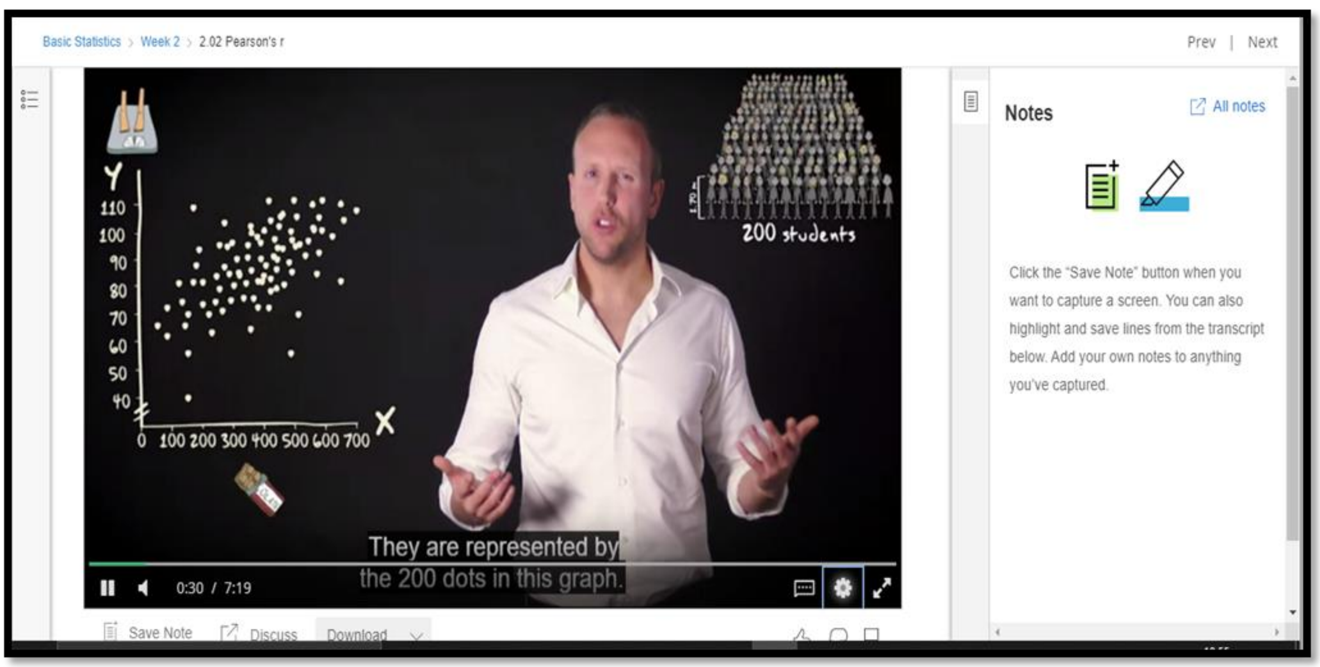
DAILY ASSESSMENT

Date:	**21-July-2020**	**Name:**	**Swastik R Gowda**
Course:	**Coursera- Basic Statistics**	**USN:**	**4AL17EC091**
Topic:	**WEEK 2**	**Semester & Section:**	**6th Sem 'B' Sec**
Github Repository:	**swastik-gowda**		

FORENOON SESSION DETAILS

Image of session





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

Basic Statistics:

Crosstabs and scatterplots:

- 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.
- Today we learnt how we can display a relationship between two variables using tables and using graphs.
- This can be very useful to discover if two variables are correlated or not.

Pearson's r:

- The scatterplot shows at a glance that there is a strong correlation between the two variables. The more chocolate someone eats, the larger the body weight.
- But how strong is this correlation? We will now turn to one of the most often used measures of correlation, the Pearson's r. One of the most important advantages of the Pearson's r, is that it expresses the direction and strength of the linear correlation between two variables with one single number.
- The relation between chocolate consumption and body weight can best be described by this straight line, because all cases closely around a line we can conclude that this is a rather strong correlation.
- Another thing to note is that the line goes up, so more chocolate consumption is associated with higher body weight. We can therefore also say that there is a positive correlation.
- Conclusion, we have a strong, positive and linear relationship here. However, variables could also be correlated in different ways.

Regression - Finding the line:

- You measure the vertical distance between Japan and the line, the distance between Spain and the line and so on. Until you know the distance to the line of every case in your study.
- Every distance is called a residual and you end up with positive residuals, the distances from cases above the line to the line displayed in blue. And negative residuals, distances from cases below the line to the line displayed in red.
- You measure these residuals for every possible line through the scatterplot. So not only for this line, but also for this line, for this line, and for this line. And for every other possible line through the scatter plot.
- Eventually, you choose the line for which the sum of the squared residuals is the smallest, and that's this one.
- Why the squared residuals? Because positive and negative residuals cancel each other out. The sum of the length of the positive residuals, the blue lines, is exactly as big as the sum of the length of the negative residuals, the red lines.
- The best fitting line is called a regression line and the name of the method of analysis is called ordinary least squares regression, which refers to the way we have found the line.

Regression - Describing the line:

- There is one simple formula with which we can describe the regression line. And that's this one. Y hat equals a plus b times x. Y hat is not the actual value of y but it represents the predicted value of y. For example, when x equals 12, y hat equals 28. Notice that the actual value of y in this case is 33. However the predicted value of y is the value of y on the regression line. This means that all the values exactly on the regression line are y hats.
- A is what we call the intercept, or the constant. It is the predicted value of y when x equals 0. It is in other words, the predicted value of y with the regression line crosses the y axis and x does equal 0. In our case, that's -5.63.
- B is what we call the regression coefficient or the slope. It is the change in y hat when x increases with one unit. In our case we see that when x increases with one unit for example, from 4 to 5, the predicted value of y increases with 2.80 units.
- Because we have a straight line the slope of the regression line is the same everywhere. The regression coefficient in our example is 2.80. This leads to the following regression equation. y hat equals 5.63 plus 2.80 times x. Take a look at these two regression lines. They have the same regression coefficients, or b values. When x increases with one unit, the predicted y value of line one and line two increases with the same amount.
- These lines have different intercepts, or a values, however. After all, they cross the y-axis on different positions.
- These two regression lines have different regression coefficients. When x increases with one unit, y hat of line one increase more than y hat of line two. Yet the intercepts of these two lines are the same, because they cross the y-axis at the same spot.

Correlation is not causation:

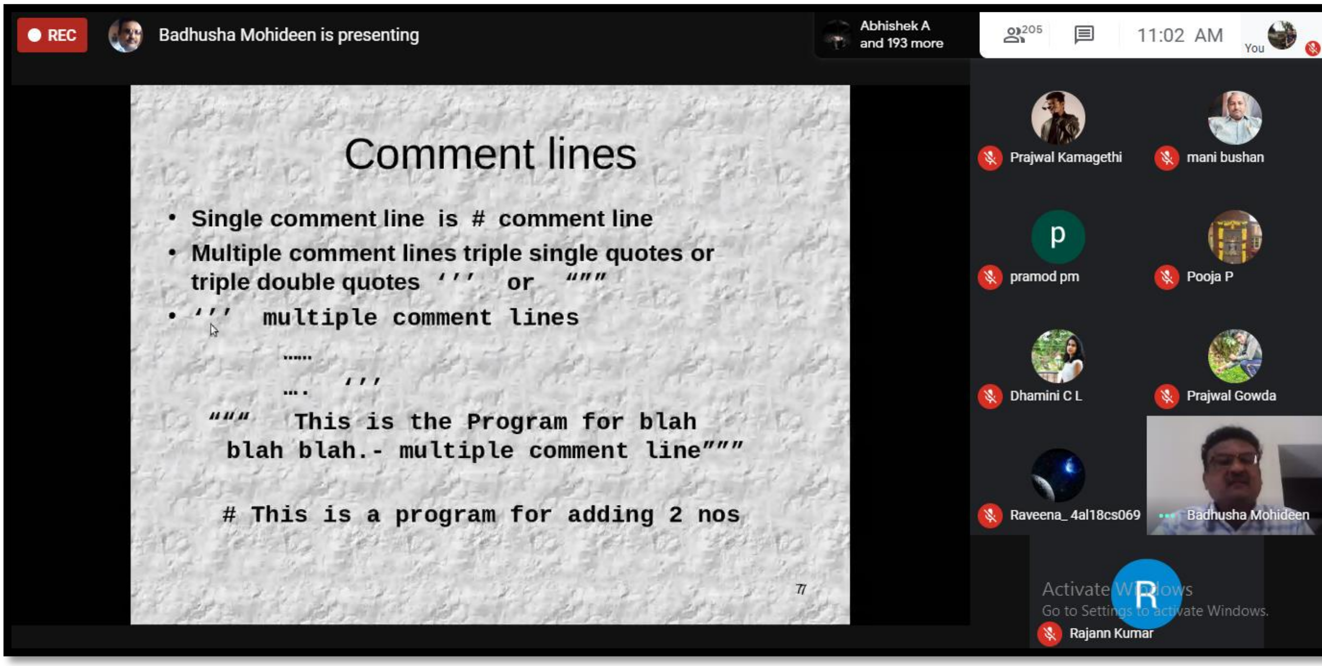
- When we do a regression analysis, we assume that the independent variable, x, explains the dependent variable, y.
- Building on that assumption we can make this scatterplot, and let the computer draw the line that best describes the linear relationship between the two variables.
- With this line, and the corresponding regression equation, we can predict the values of the dependent variables based on the values of the independent variable. Moreover, with r squared, we can also assess how well the line fits our data.
- However for at least two reasons we need to be very very careful when we interpret the results.
- The first reason is that on the basis of a regression analysis we can never prove that there was a causal relationship between the two variables. We can, in other words, never be certain that one of the variables is the cause of the other variable.
- This translates into one single and not very complicated but extremely important message. Correlation is not the same as causation.
- For instance, research suggests that eating a lot of chocolate increases your body weight. And this scatterplot shows that the more chocolate people eat, the larger the body weight tends to be.

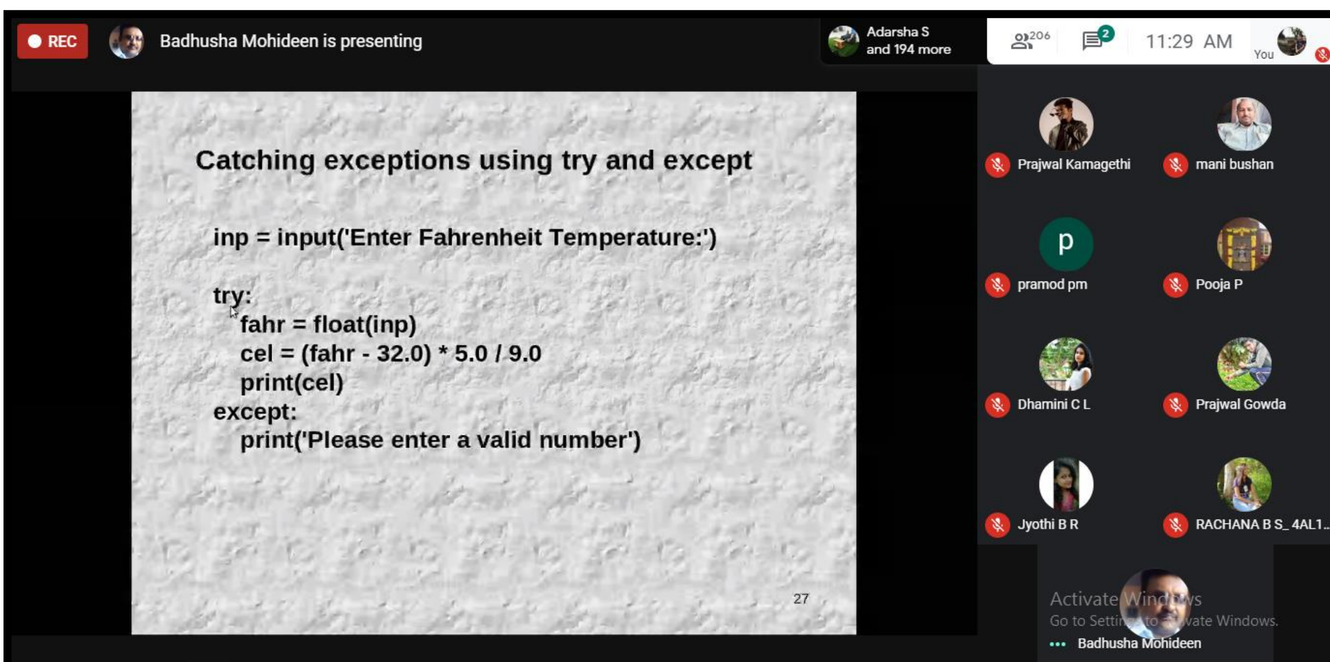
However, we need to be careful here. It might also be the case that causality runs in the opposite direction.

Date:	**21-July-2020**	**Name:**	**Swastik R Gowda**
Course:	**How to develop Pythonic coding – Logic Perspective**	**USN:**	**4AL17EC091**
Topic:	**Day 1**	**Semester & Section:**	**6th Sem 'B' Sec**

AFTERNOON SESSION DETAILS

Image of session





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

Open In Colab

In [ ]:
```python
#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
#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
#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
#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
#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

Find the length of a string without using len functions

In [24]:
```
#6
str = input("Enter a string: ")
count = 0
for s in str:
      count = count+1
print("Length of the input string is:", count)
```

```
Enter a string: Swastik
Length of the input string is: 7
```

Find the no of words and characters in a string

In [25]:
```
#7
string=input("Enter string:")
word=1
for i in string:
      if(i==' '):
            word=word+1
print("Number of words in the string:")
print(word)
```

```
Enter string:It is what it is though we don't know what it is
Number of words in the string:
12
```

Find the no of occurrences of a word in a string

In [26]:
```
#8
def countOccurences(str, word):
    a = str.split(" ")
    count = 0
    for i in range(0, len(a)):
        if (word == a[i]):
           count = count + 1
    return count
str =input("Enter the sentence:")
word =input("Enter the word:")
print("The word is repeated:",countOccurences(str, word),"times")
```

```
Enter the sentence:It is what it is even though we don't know what it actually is
Enter the word:is
The word is repeated: 3 times
```

- Python is an interpreter, 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.