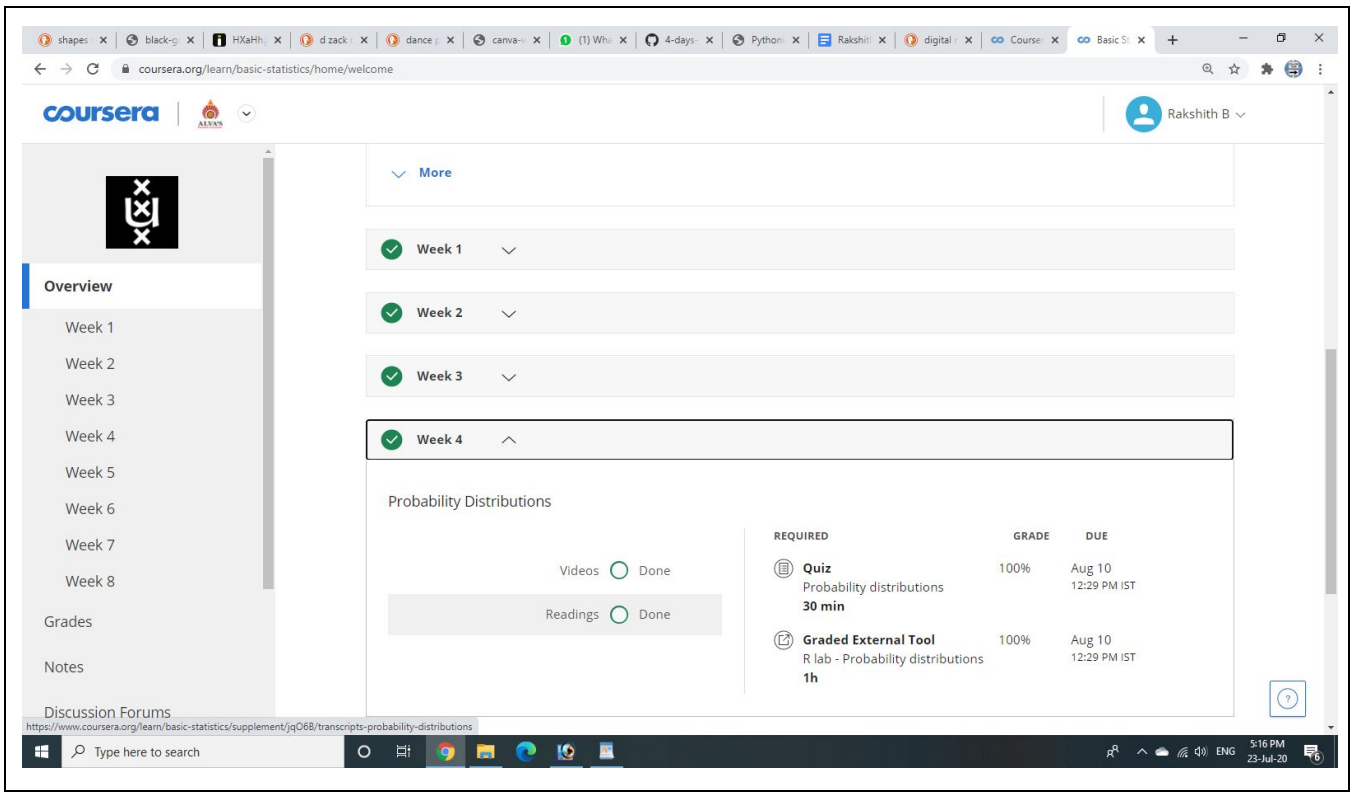
REPORT JULY 22

Date:	23 JULY 2020	Name:	Rakshith B
Course:	Coursera	USN:	4AL16EC409
Topic:	Basic statistics	Semester & Section:	6th SEM B
Github Repository:	Rakshith-B		

Image of the Session



Before we can understand probability we first have to understand another concept: **randomness**. The first video explains this concept. It also shows that even though randomness is everywhere around us, humans are nonetheless bad in assessing it.

Once we understand randomness we can define **probability** as a way to **quantify randomness**. The second video explains how this quantification can be accomplished by experiments which record the **relative frequency** that certain **events** of interest occur. It follows that probabilities are always **larger or equal to zero** and **smaller or equal to one**; and also that the **sum of the probabilities for all possible events equals one**. Due to the very nature of random events, the experiments may have to continue for a while before the relative frequencies represent the probabilities accurately, but the law of large numbers dictates that it will do so eventually.

Recognizing and understanding randomness as well as the ability to reason about it are important skills, not only to apply statistical analysis but also to make sense of things happening around us every day. Here, I will explain that humans are pathologically bad at dealing with randomness. I will use an example along the way.

Imagine you're on a beach watching the waves roll in.

And then your attention is caught by a beautiful shell which is distinctive in shape and larger than it's neighboring shells. So you start to search for another one. This will be an unpredictable enterprise. The shells may be distributed at random at this huge beach. Hence, the time it will take you to find another will be uncertain. You may not be able to find a similar at all.

The more you think about it, the more you'll realize that randomness is pervasive in everyday life. It is therefore not surprising that we have a rich vocabulary to communicate it,

with terms like uncertainty, chance, risk and likelihood.

Also, degrees of variability and uncertainty can expressed quite subtly. Consider for example this sequence. Rarely. Seldom. Sometimes. Common. Frequent. Often.

Importantly, whether something is random is not just a property of that phenomenon, but very much also a consequence of our knowledge about it.

If you'd have been at this beach before, you might have spotted this special shell previously, so that this may change your search strategy and increase your chance of finding more of them.

Also the scale of your search matters. While you may not be very certain about finding another shell within a few minutes at a short stretch of beach, the chance to find one when you take a bit more time and cover a larger stretch of beach will increase.

But in spite of many words, our ability to memorize in our daily experience with randomness, we are not very good at assessing it quantitatively at all.

On one hand, we see all sorts of patterns in what is really random data. There's a word for it, apophenia.

And on the other hand, we are unable to make up random data ourselves.

An example of a failed attempt to create random data is this fabricated map of random shell locations which turn out to be spaced too regular.

This is how a realistic random point pattern looks like with much more clusters.

Another example of over interpreting randomness is the so called gambler's fallacy, the false idea that a random phenomenon can be predicted from a series of preceding random phenomena.

And who doesn't recognize this. If you have thrown a six four times in a row with a die, it feels as if it's going to be very unlikely to throw a six again the fifth time.

Yet, the probability of this outcome was and continues to be one sixth.

The reason that we have a bad head for randomness is that our brain tends to measure randomness in the effort it takes to memorize a pattern.

And it turns out that memorizing frequently changing short sequences is harder than longer sequences.

Given this state of affairs it's really important to learn about formal ways for quantifying randomness, reasoning about it and generating realistic random patterns. It will help to avoid mistakes, predict more accurate and be more efficient when you try to make sense of the world around you. Let me summarize what I hope you understood from this video.

Randomness is not an intrinsic property of a phenomenon. It also depends amongst others of prior knowledge, observation method, and a scale at which the phenomenon is considered.

Play video starting at 4 minutes 20 seconds and follow transcript

4:20

While there are many words to express aspects of randomness, humans are not very good in assessing it quantitatively. We suffer from apophenia, the over interpretation of what are purely random patterns and are also bad in constructing randomness.

Date:	23 JULY 2020	Name:	Rakshith B
Course:	Pythonic coding	USN:	4AL16EC409
Topic:	Python	Semester & Section:	6th SEM B
Github Repository:	Rakshith_B_colab		

```
#Program for prime number -bad code

i=25

for x in range(2, i//2+1):

     if i%x==0:

          print("The    number    {}    is    not    prime".format(i))
          break

if x ==i//2:

     print ("{} is a prime number".format(i))

#Program to find the largest number in a list.

a=[]

n=int(input("Enter number of elements:"))

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

    b=int(input("Enter element:"))

    a.append(b)

a.sort()

print("Largest element is:",a[n-1])
```

Try this More pythonic 2 lines program equivalent to above codings as shown below !

```
x=0
print('The greatest no is',max([int(input(x)) for _ in range(int(input("Enter no")))]))


#Python Program to put the even and odd elements in a list into two different lists.
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
even=[]
odd=[]
for j in a:
    if(j%2==0):
        even.append(j)
    else:
        odd.append(j)
print("The even list",even)
print("The odd list",odd)


#Instead of  forementioned 14 lines program, the equivalent 4 lines !
#Pythonic program is here ....
x=0
l=[int(input(x)) for _ in range(int(input("Enter n")))]
print('even list is',[ i for i in l if i%2])
print('odd list is',[i for i in l if not i%2])
```

```
#Python Program to merge two lists and sort it.
a=[]
c=[]
n1=int(input("Enter number of elements:"))
for i in range(1,n1+1):
    b=int(input("Enter element:"))
    a.append(b)
n2=int(input("Enter number of elements:"))
for i in range(1,n2+1):
    d=int(input("Enter element:"))
    c.append(d)
new=a+c
new.sort()
print("Sorted list is:",new)
#Instead of  forementioned 13 lines program, the equivalent 5 lines !
# Pythonic program is here ....
x=0
l=[int(input(x))   for   _   in   range(int(input("Enter   how   many elements")))]
m=[int(input(x))  for  _  in  range(int(input("Enter how many elements ")))]
new=l+m
new.sort()
print("Sorted list is:",new)


#Python Program to sort the list according to the second element in the sublist.
a=[['A',34],['B',21],['C',26],['E',29]]
for i in range(0,len(a)):
```

```
    for j in range(i+1,len(a)):
        if(a[i][1]>a[j][1]):
            temp=a[j]
            a[j]=a[i]
            a[i]=temp
print(a)

#Instead of  forementioned 8 lines program, the equivalent 3 lines !
#Pythonic program is here ....
a=[['A',34],['B',21],['C',26],['E',29]]
a.sort(key = lambda x: x[1])
print(a)

#Python Program to find the second largest number in a list
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:"))
    a.append(b)
for i in range(0,len(a)):
    for j in range(0,len(a)-i-1):
        if(a[j]>a[j+1]):
            temp=a[j]
            a[j]=a[j+1]
            a[j+1]=temp
print('Second largest number is:',a[n-2])
```

```
#Instead of  forementioned 12 lines program, the equivalent 4 lines !
#Pythonic program is here ....
x=0
l=[int(input(x))  for  _  in  range(int(input("Enter  how  many elements")))]
l.sort()
print("Second largest element is :",l[-2])

#Program to create a list of tuples with the first element as the number and the second element as the square of the number.
l_range=int(input("Enter the lower range:"))
u_range=int(input("Enter the upper range:"))
a=[(x,x**2) for x in range(l_range,u_range+1)]
print(a)

#The aforementioned program is already pythonic.
#We can still make it more pythonic as follows
a=[(x,x**2) for x in range(int(input("Enter the lower range:")),\
                           int(input("Enter the upper range:"))+1)]
print(a)

# Of course, We can write in the most pythonic way with one line ! as follows

print([(x,x**2) for x in range(int(input("Enter the lower range:")),\
                           int(input("Enter the upper range:"))+1)])
```

```
#Python Program to generate random numbers from 1 to 20 and append them to the list.

import random

a=[]

n=int(input("Enter number of elements:"))

for j in range(n):

    a.append(random.randint(1,20))

print('Randomised list is: ',a)
```