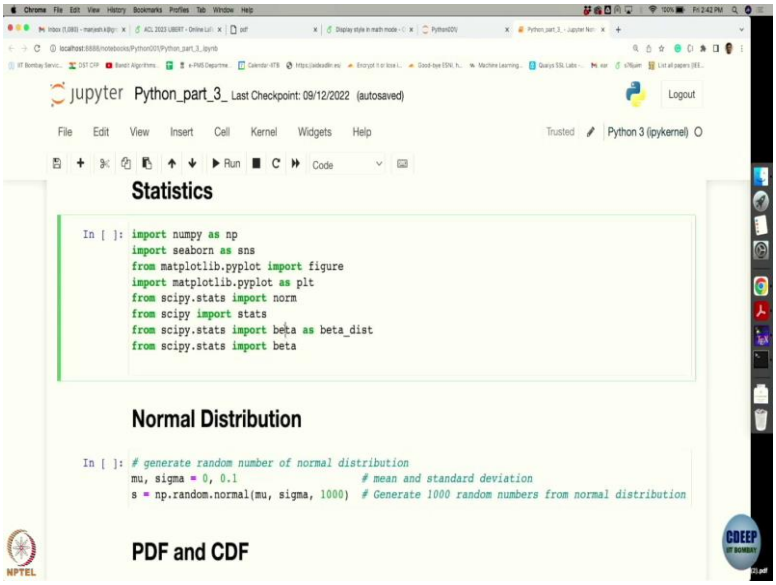
Engineering Statistics
Professor Manjesh Hanawal
Department of Industrial Engineering and Operational Research
Indian Institute of Technology, Bombay
Lecture 59
Generating Random Samples using Python,
Hypothesis Testing using Python

Hello, everyone. So, we were discussing about basic introduction to Python. And in the last class, we talked about how to create variables in Python, how to assign values to them, how to manipulate your strings, how to do some basic mathematical operations on numbers. And then we also looked into how to use a fill statements, looping. And at the end we talked about how to use this NumPy function. And we also saw how to read files, and how to manipulate files. So, today we will continue the discussion. And our focus today will be to understand some of the statistics aspects that are available, or statistics function that are available in Python.

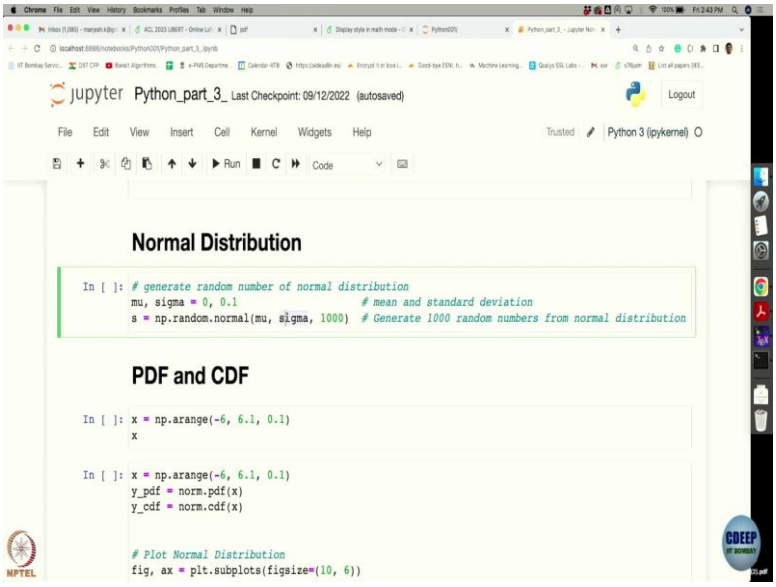
(Refer Slide Time: 1:24)

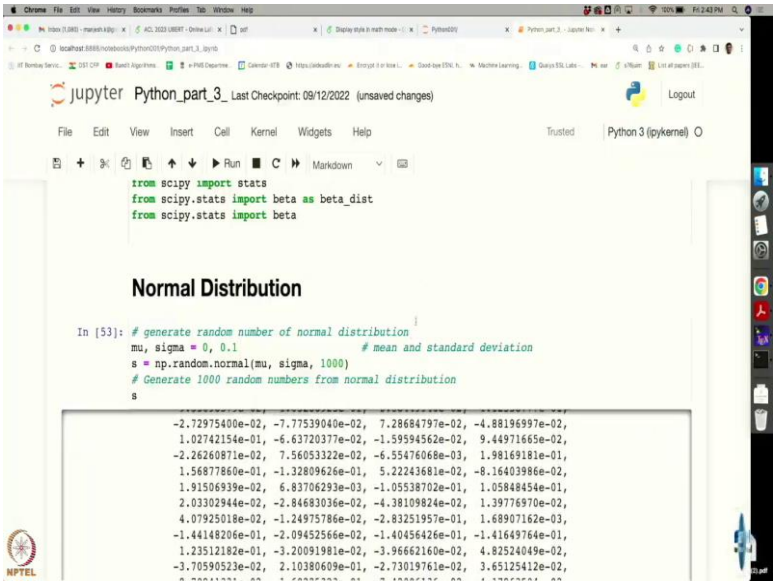


So, when we want to use statistics, we want to how to play with data, maybe you also want to use visualisation aspects so that we can see how the data pattern looks like. So, for this, we would be interested in many libraries. So, here, as usual, I am going to take import the NumPy function. And also, there is another nice library in Python that is mostly to see visualisation of the data. It is called seaborn that also I am going to import as sns. And then there is another nice library called a matplotlib.pyplot, and from that, I am going to import a particular library this is called a figure, I and I will also import this function matplotlib.pyplot as plt, and also there is another function, sorry another library called scipy.

And I am going to also take some functions from that so from this scipy, stats I am going to import, this norm function which is going to help us to deal with quotient distribution, and also from the scipy, I am also going to import stats which we will see that which is going to help us look into various statistical aspects. And I am also going to import beta function as a beta underscore dist. And I will also go into take beta from scipy stats. So, these 2 are actually same, if you see that but here I will just is named beta as beta distribution here it is simply beta.

(Refer Slide Time: 03:45)



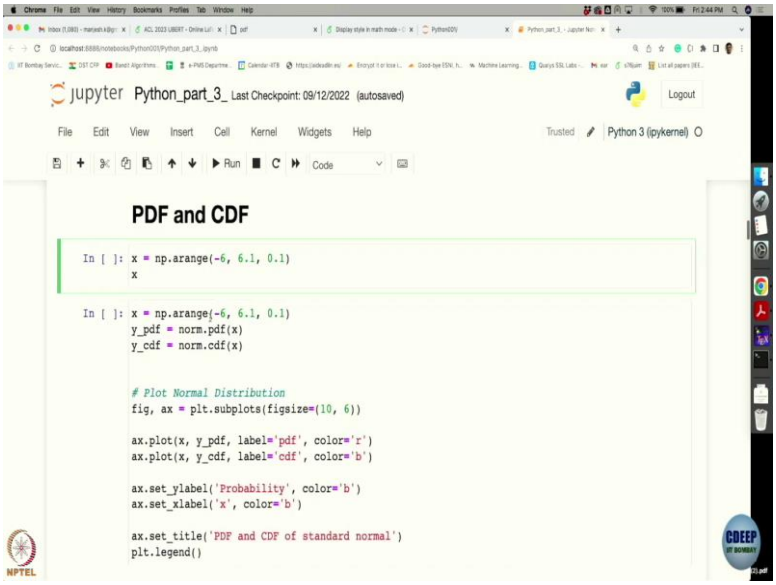


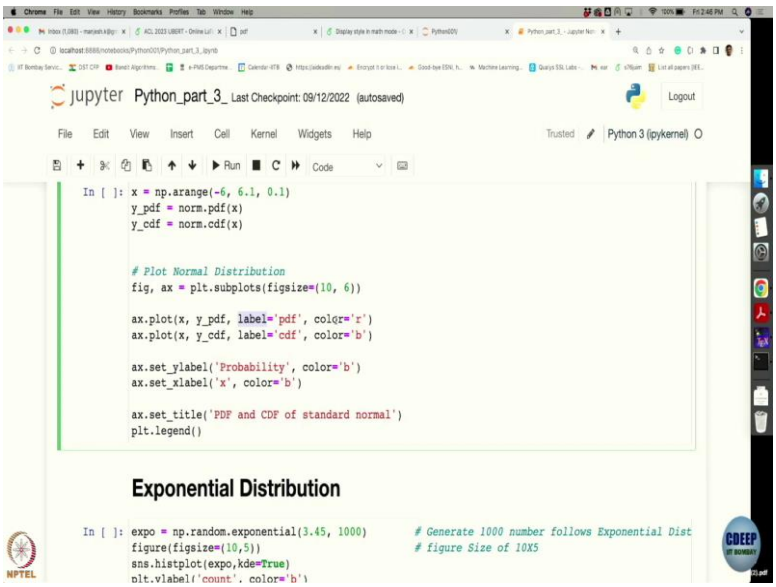
Now, let us get started with normal distributions, how to generate samples using normal distribution. So, for that, suppose you want to generate normal distribution with mu and

sigma given as 0.1. Notice that in Python in the same line you can assign two variables here. So, mu and sigma are assigned values 0 and 0.1 respectively.

And now in nine pi, there is our function random dot normal which will help us to generate the samples according to normal distribution, we have to just pass on this mu and sigma values and how many samples we want, then it will generate us. So, let us say that now I am just executing a sales now. Now, I have this data generated s which is like about 1000 data points, which are gaussian distributed.

(Refer Slide Time: 04:55)





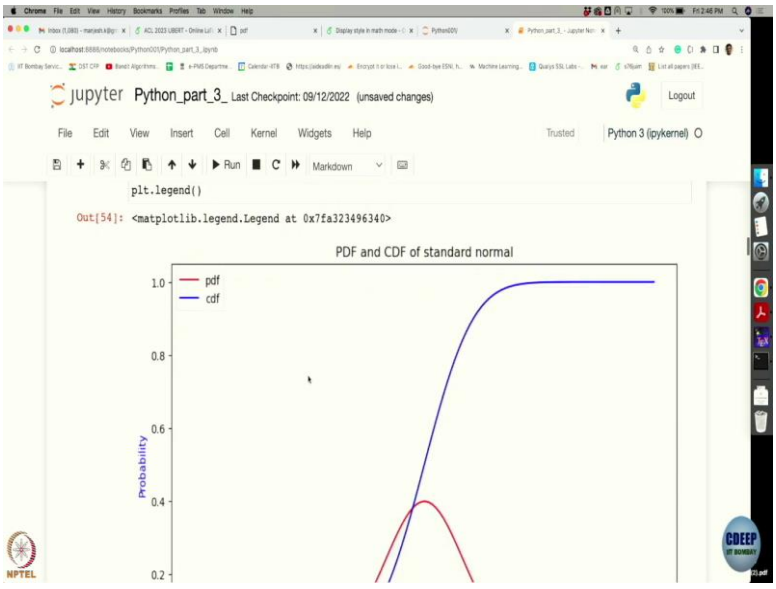
Now, I want to see let us see that, now you want to, let us say you want to see the PDF or CDF of quotient distribution. So, for that you first need to specify, and at which point you want to see the PDF, CDF. So, let us say I want to see the plots between minus 2 to 6.1. So, I

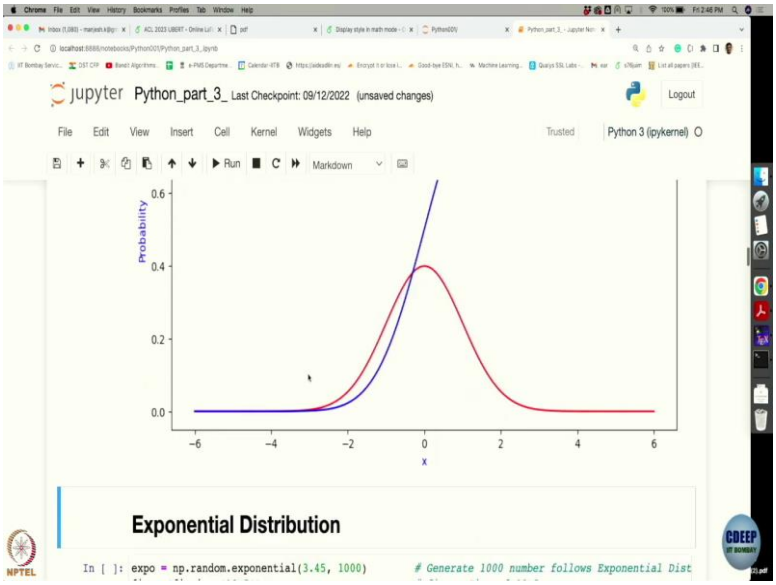
am going to use this function np a range to get those values. And then, this is the same thing written here. Now, what I will do is, I will use this function norm.pdf, notice that norm that we got it from scipy stats, which we imported as norm, and .pdf. So, this is going to generate me, give me this PDF of a normal distribution at the points x.

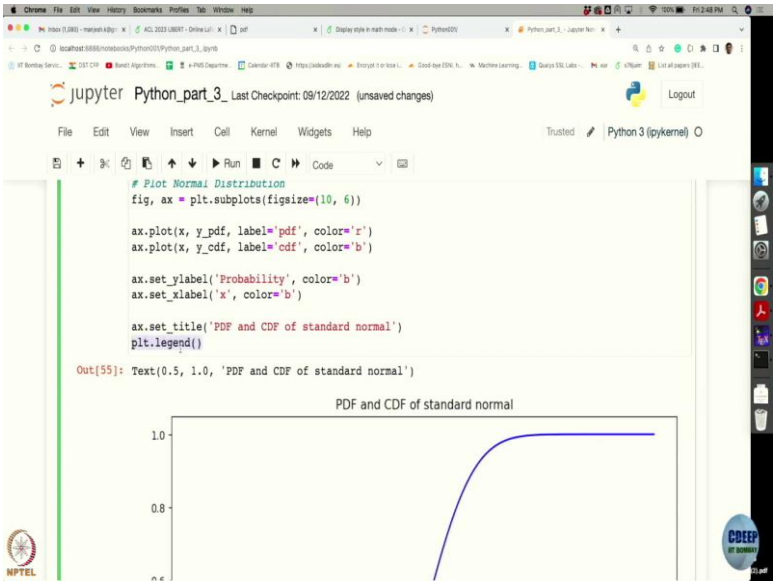
Similarly, I have this CDF now, if I am going to call norm.cdf function, this is going to give me CDF of this normal distribution at point x, at 0.6. So, here in this norm.pdf, can take other variables like the mu and sigma squares. But here, we have not passed anything other than the points where we want to see a PDF. So, it is going to give us a PDF which is normally distributed that is, with mu 0 and variance 0.1.

Now, we want to see that, for that, we need to use this plot function which we imported as plt, this matplotlib.pyplot, we imported as plot. Now, I am going to first say I have to, first I am going to, I think you are, so first, I am going to create this variable ax, which is plot.subplots. And this is going to create figure which has of size 10, cross 6, that is 10 comma 6 is giving you the aspect ratio. Now, I have this x, I want to see the y values of this PDF function. And now I can, this plot function also help me in assigning the labels, I can write label equals to PDF, and also use the colour for those lines that are going to show.

(Refer Slide Time: 07:50)







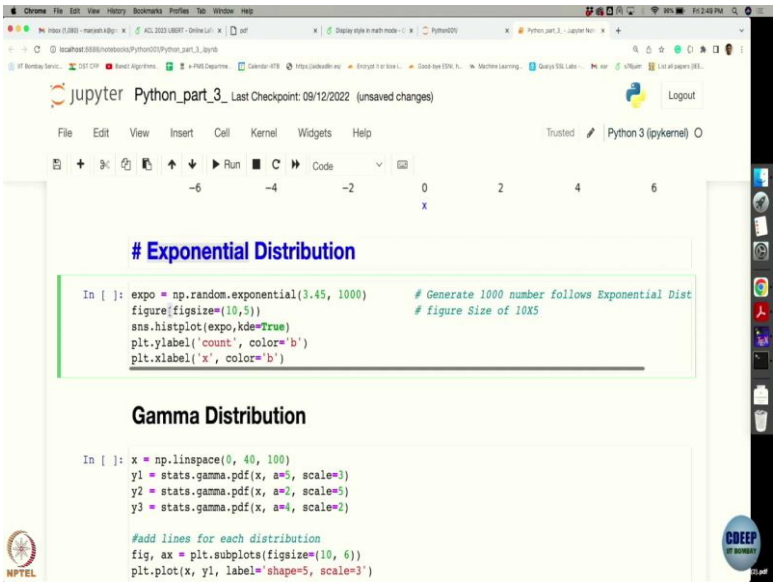
First let us see this how it looks. So, now if I do this, you will see that range is between minus 6 to 6. And this is like a plot of a quotient distribution. And this is like a plot of a CDF of your quotient distributions, notice that it is at 0, you are going to see the value of half as expected.

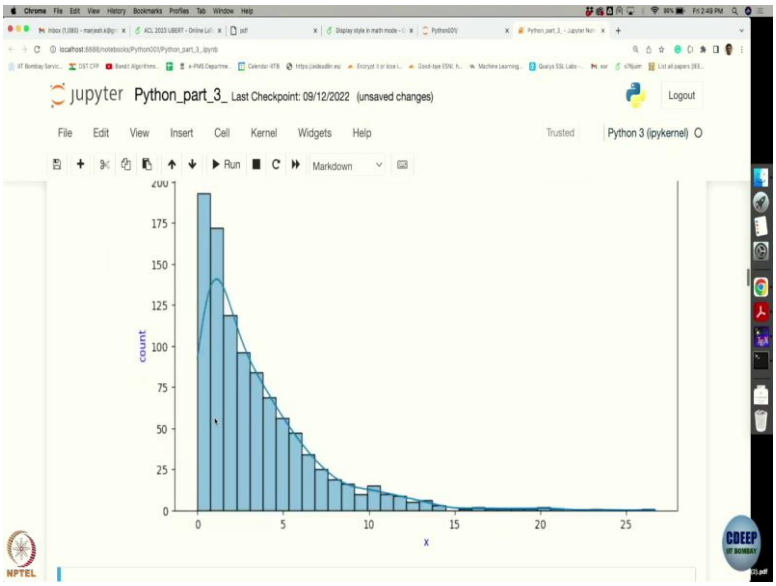
Now, these things, the labels, which you put PDF, they have come here, and the colour red is assigned to the PDF function and blue is assigned to the CDF function. And if you want to label x and y labels, that provision is there, you can simply say, set y level as simply probability and with whatever colour you want to appear. And similarly, you can label x.

And so that is why this probability index has appeared with colour blue. And you can also give the title to this plot. And here I have given it as PDF and CDF standard normal. So, now let us see, this plot legend actually plots this legend here. If you do not put this, then this

legend will not appear. So, if I do this, there is no legend here. But if I do this the legend will appear.

(Refer Slide Time: 9:20)





Like this, we can generate RC, PDFs and CDF for various distribution. Now, I am going to use another one exponential distribution. But here instead of looking into PDF, CDF, let us try to generate the sample according to the distributions with the specified parameters. So, suppose let us say I want to generate exponentially distributed samples with parameter 3.45.

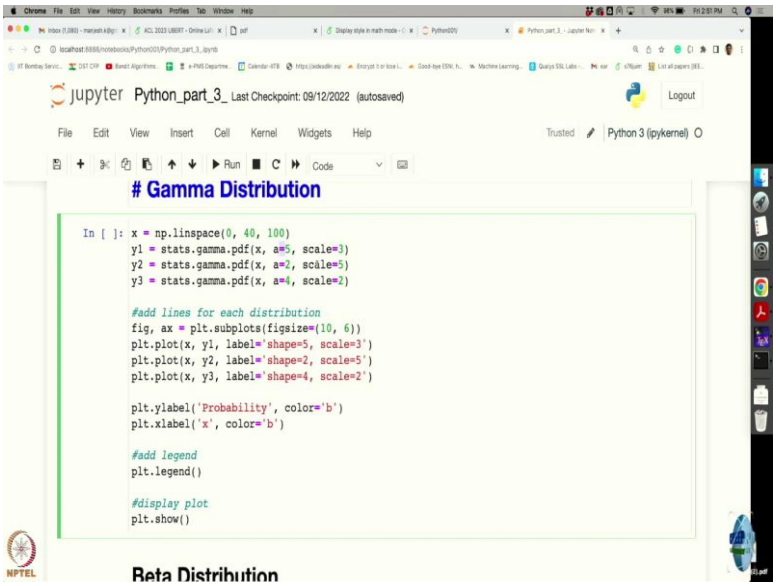
Now, there is a function for it called random.exponential. I can pass on that whatever the lambda value I want, and how many samples I want, and it will generate those value. And now, I want to see now I have a data, now I want to see that histogram wise how they look. So, I hope all of you understand histogram basically splits the groups the data into various

bins, and see that in each bin how many points are falling and the number of samples in a particular bin will give you the height of that bin.

To show that, I am going to use this function figure, which has aspect ratio 10 into 5. And then I am going to use this sns function which we have defined, and in that I am going to call this function his plot passing on the data I have. Now, let us and let us see what happens. Now, if you see that I have this.

So, here it is saying it has basically looks like it has grouped 3, 4, 5, 6 between it has grouped that data into this bins, the bins are have shown by this vertical bars, and the height of this vertical told us out of 1000 samples how many samples fill in each of these bins. And if you see that roughly it has this exponential structure to it. And again, we can label this x and y, labels so y labels are named as count here and x label simply x.

(Refer Slide Time: 11:41)



jupyter Python_part_3_ Last Checkpoint: 09/12/2022 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3 (ipykernel)

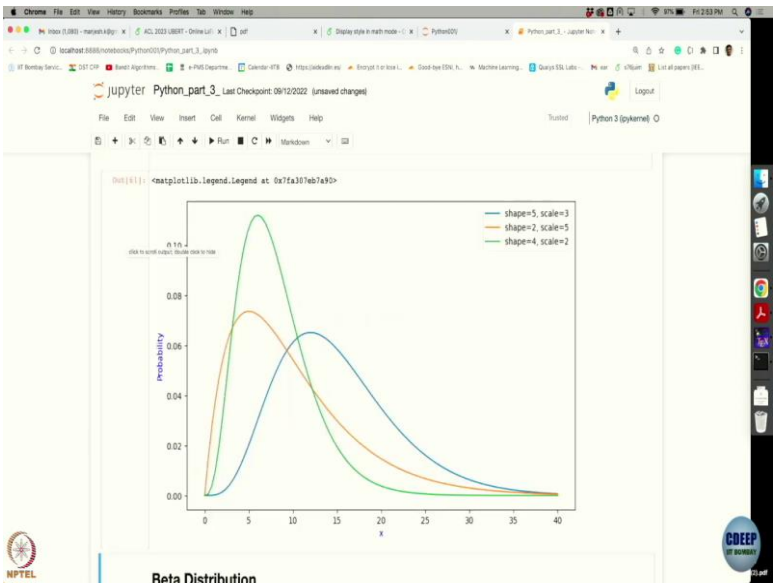
Run Code

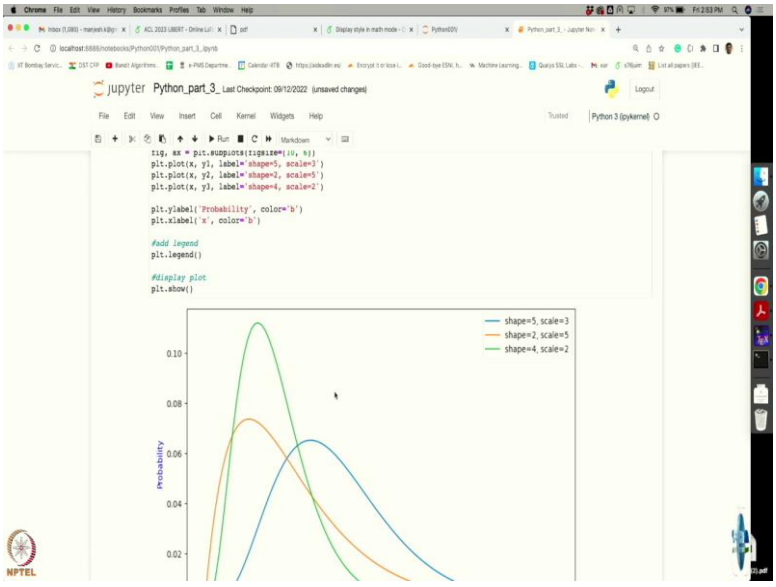
Statistics

In [51]:
```
import numpy as np
import seaborn as sns
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
from scipy.stats import norm
from scipy import stats
from scipy.stats import beta as beta_dist
from scipy.stats import beta
```

Normal Distribution

In [53]:
```
# generate random number of normal distribution
mu, sigma = 0, 0.1                  # mean and standard deviation
s = np.random.normal(mu, sigma, 1000)
# Generate 1000 random numbers from normal distribution
```





Similarly, we can do it for other distributions like here, let us say in this exercise, I am going to see the PDF of how gamma distribution, now first as I said, when I want to see a PDF I need to first specify at what point I want to see the specify, here I have specified this to be range between 0 to 40 and 0 to 40 is to be split into 100 points with linear spacing.

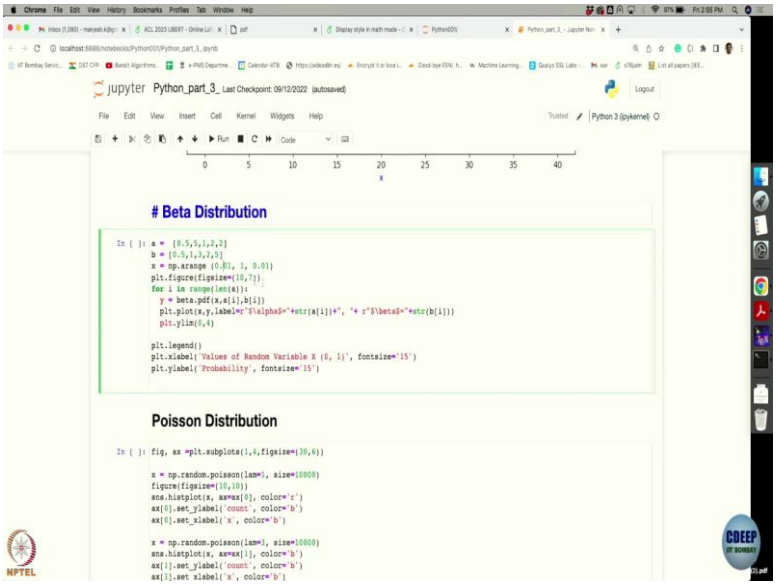
Now, these stats function so, the stat library we have which we imported as from the scipy package, I can call the gamma function and do the generate the PDF, I need to pass the points we are going to see now here I also need to specify what is the here I specified shift and scale values. Now, I have plotted actually 3 here for different possible different values of a and scale, and I want to also visualise them now.

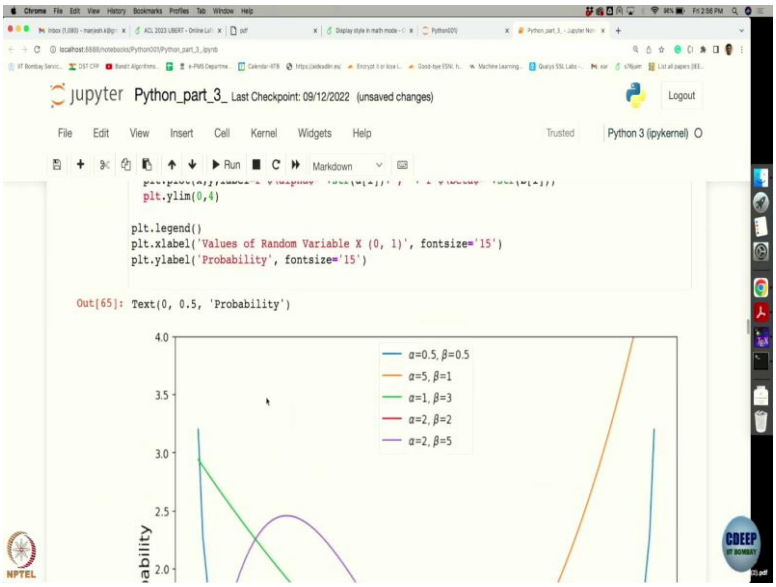
To visualise, so now the graph is generated here let us say to visualise what I am going to do is again I am going to use this function subplots in plot and specify the aspect ratio and I will call each one of them now. I want to see first the y 1 and y 2 and y 3 so I passed on them to your plot function and also each one of them I have labelled with sorry not I said shift, it is actually shape parameter and the scale parameters.

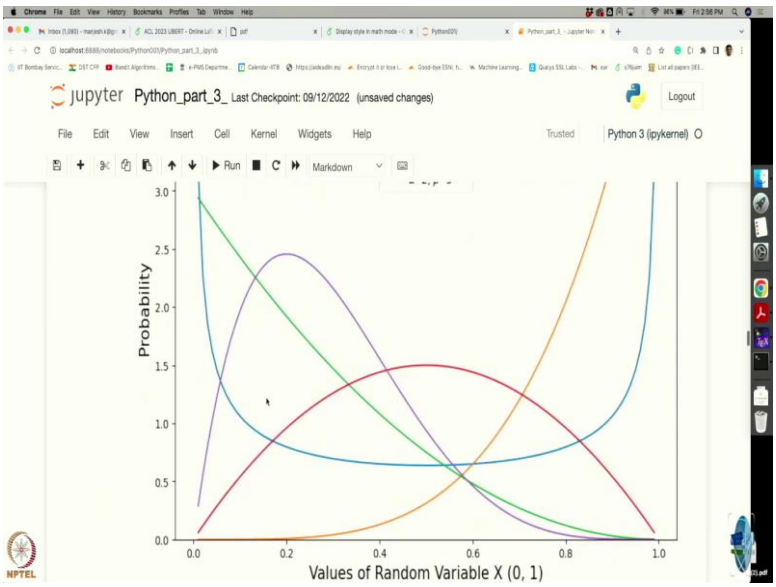
And this is how it looks and here we have also put a legend and, let us see what happens if I do not have this plot here. And now when I say plot, I do not see much difference but legend is there let us see again, now if I put this, only see the difference I see is that whatever that some naming was coming on the top that disappeared. So, this map plot legend, something came here and if I say plot, I think it had just to show me a clean image without any additional things which I did not ask for it.

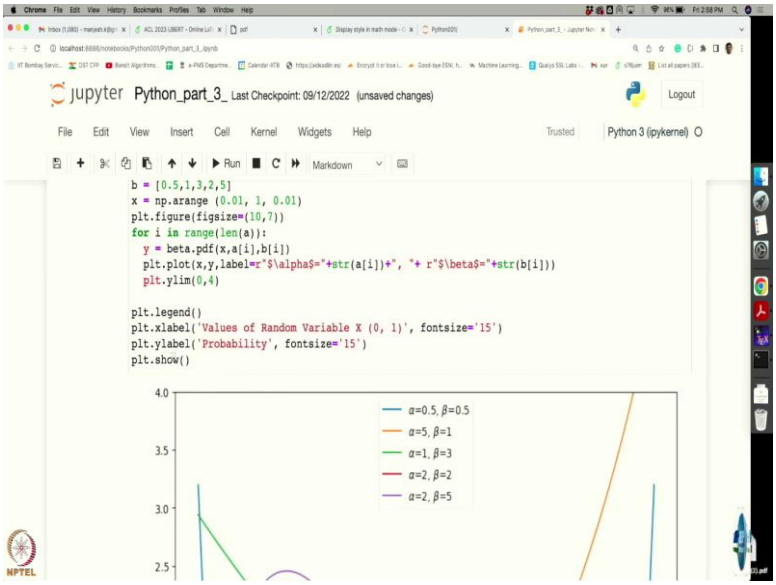
Next, we can similarly look for beta distribution here again I have this beta distribution, but here see like I hat passed on, if I want to generate multiple gamma distribution, I specified the scale and shape parameters one for 3 set of scale and shape parameters, but and I have to then call this plot function 3 times, maybe this can be avoided if he intelligently use our loops. And that is what we will demonstrate now, for the beta distribution here.

(Refer Slide Time: 15:35)









Suppose beta distributions we want to generate the CDF for 5 different pairs of parameters, alpha and beta parameters. So, these alpha parameters I am taking 0.5, 5, 1, 2, 2 here and similarly, beta values have taken 0.5, 1, 3, 2, 5 here and the x values where I want to see that I have taken it to be the interval .01 to 1 with increment of 0.01. So, there are about 100 points.

Now, this further plot figure out taken the finger size to be 10 into 7. And now, instead of calling this plot function now for each pair, I can try to write a call them iteratively through a loop function. Here I am defining variable i, which is going to range length of a, so I want to have to plot 5 figures here, so I am just defining that to be length of a. And then I can just use the beta.pdf function and pass x, a of i, and b of i.

And then I can simply draw that by using this plot function. So, here, notice that first let us plot this. So, now nice, we got a nice plot here. Now, you see that in to write the label, I have used this r, this is useful whenever I want to use these symbols like here, I want alpha to be 0.5, alpha, I cannot directly write here like this.

But I can use the LaTeX code for alpha, which is backslash alpha here and Python to know about this, I will write an r at the beginning. So, now if you see that it is going to be alpha here, and then it is equal, alpha equal. And then I have added a string here, this is a string I am adding and this corresponding to the value of a at location i.

And after that, if you see, I have added another string here, which is a comma and a space, which has appeared here. And then again, I need to get this LaTeX code for beta. To let it know that I am writing a LaTeX command here I will try, I am going to write r, and then

again, I will append the string I want to show b i, but bi I cannot directly put. So, I will convert into string and appended here, and I can also specify the range of y here which are specified by giving the y limit. So, in this way, the plotting and visualisation of the CDF is pretty easy. Just by a few lines of code we will get and you can play around with them.

And of course, we can label them we can do the x and y label as per we are wish, and also specify the font sizes. So, there will be multiple options here like size. And if you want to add some colours and all, if you do not specify any colours by default it is going to take it as black. But if you want to specify some colours, you have to pass on that argument here. So, let us add this plt.show here and see what happens which we saw last time. So, you see that when I did plt.show it cleaned up like that no other text, other than these figures.