Scientific Computing Using Python

Professor. Vivek Aggarwal and Professor. Mani Mehra

Department of Mathematics

Indian Institute of Technology, Delhi

Lecture No. 06

Welcome to Scientific Computing in Python. In the last few lectures, we have learnt the basics of Python. So today, we will do some more commands related to plotting. Let's get started. This is sixth lecture.

Today, I will first explain how we can give input. So, I will first explain the input function, which we will use a lot. Now, if I have to give the value of x and someone asks me to input the value, what do I want to input? Suppose we are asked, "your name please." Okay. After that, I write print. Here, I define print(Your name is,',x) and here I print x. Okay, so I defined it like this. What did I do here? I defined a function. The name of that function is input. Whatever is written inside it will appear on the screen. Then I will input the value there. Let's run it. So, it appears as "Your name, please." Suppose I write my name "Rakesh" and press enter, then it appears that your name is Rakesh. So, in this way, anyone can ask us for input. How can we give input? With the help of this input function.

So, I write:

y = input('Your age please:')

If we are asked, "What is your age?" and we write it like this, then it will ask us. After that, I will print it using:

print('your age is', y)

I'll give a little gap, and whatever value we input here will be printed. So, I run it. Suppose I write 28 and press enter. Then we get the output. What happened first? The function input asked for the value with the message "Your age, please." I entered 28, and this value went into the variable y. Then the next command printed it: "your age is 28" .So, like this, we have printed 28. We've entered input values. We will use this input function many times. We can define it like this.

Now we will learn how to plot in Python. The main question is: how to plot? For plotting, we have to use what is called the Matplotlib library.

What is Matplotlib library? This library will be used for plotting. We use plotting for visualization because if we are given data , if we've done something like finding roots or solutions , and we cannot visualize it, it becomes difficult to understand. So, it is always good if we can visualize it. Visualizing means that we generate an image that allows us to see the structure easily. To do this, we will plot. If someone doesn't have Matplotlib installed in Python, they can write:

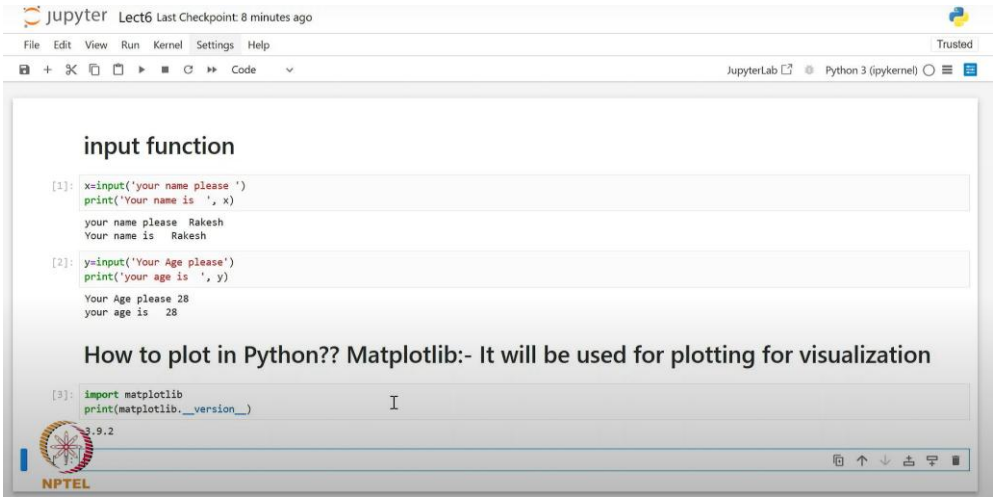
import matplotlib

We can also check its version by writing:

print(matplotlib.__version__)

After we run this, the version will be printed, like 3.9.2, which is installed on our computer. So, the version of matplotlib installed in our computer is 3.9.2.

(Refer slide time: 6:37)



How will we use this now, to use Matplotlib, we use a submodule called pyplot (Python Plot). And we will use a command in this called plt. Now, let's see how to use this. We import it like this:

import matplotlib.pyplot as plt

This way, we can refer to it as plt, which is like a nickname. After that, we import NumPy to work with arrays:

import numpy as np

Now I will try to plot something. I define an array as xpoints as:

xpoints = np.array([0, 5])

This is a simple array with two elements. I also define ypoints as:

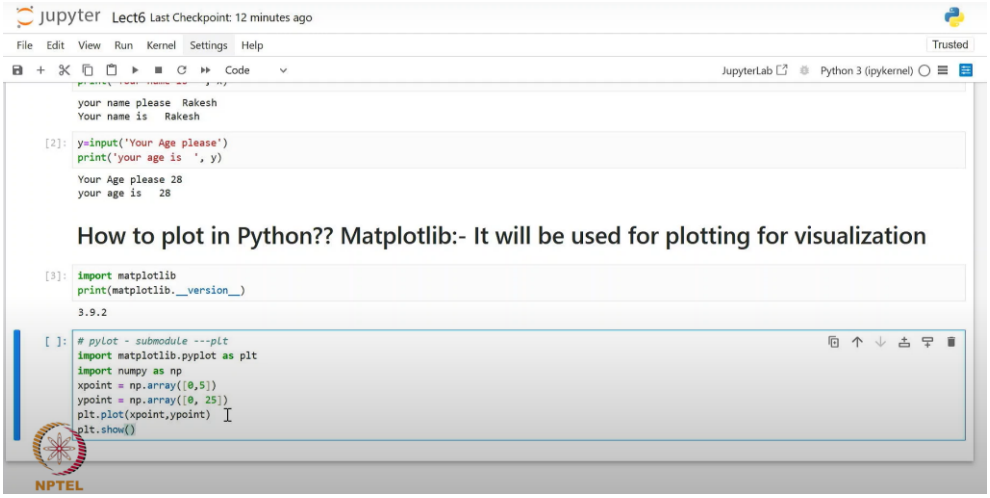
ypoints = np.array([0, 25])

Now we use the submodule in which there are lots of command, we are using one of them that is plot. So, we will write:

plt.plot(xpoints, ypoints)

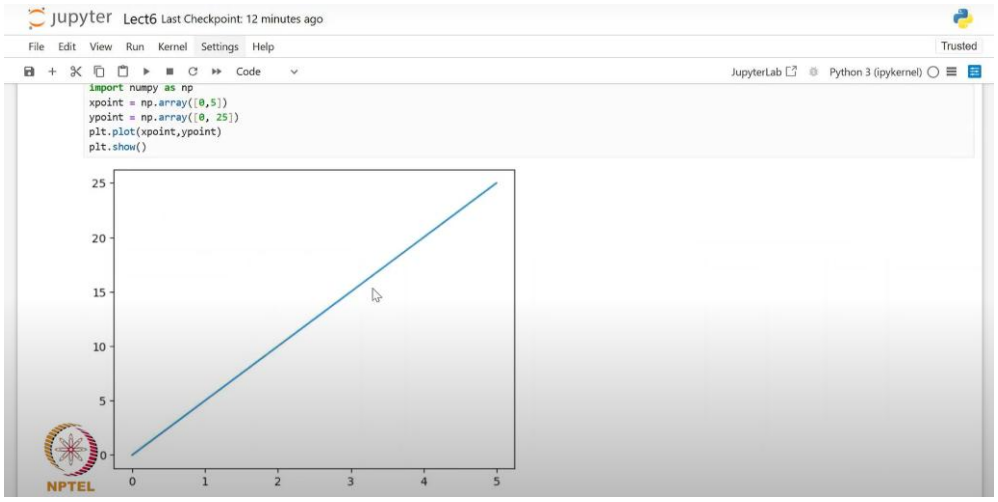
plt.show()

(Refer slide time: 9:57)



This is to show our plot. When we run this, a plot appears. On the x-axis, we've taken values from 0 to 5, and on the y-axis, from 0 to 25. These are just two points: (0, 0) and (5, 25). We plotted them. So, for y axis it went up to 25 and for x axis it went to 5. Like that we have plotted it. What do we have? Just these two points. Therefore, the x-axis goes from 0 to 5, and the y-axis from 0 to 25. We can plot for any values at any time.

(Refer slide time: 10:11)

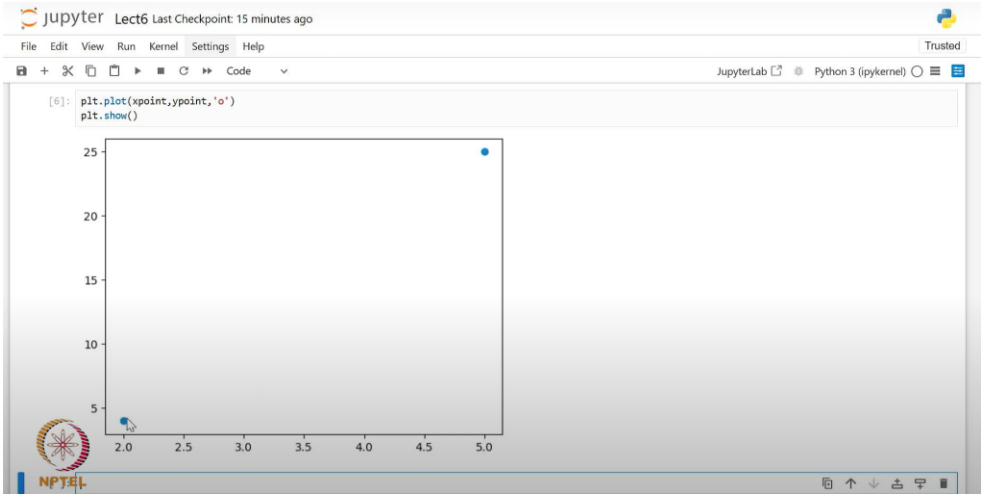


Suppose I want to change it from 2 to 5 on the x-axis and 4 to 25 on the y-axis. Then we modify the arrays accordingly and plot again. Now, you'll notice the graph shows a line for 2 to 5 on x axis and 4 to 25 on y axis. So, this is our line and we have visualize it. It is showing linear line. Although we provided only two points (2, 4) and (5, 25), a full line appears. If I want to emphasize that, I will show it with the help of point. So, write:

plt.plot(xpoint, ypoint, 'o')

plt.show()

(Refer slide time: 12:50)



This ‘o’ is called marker. We get new graph and it shows only two dots — one at (2, 4) and one at (5, 25) — instead of a connecting line. This makes it clear that we had only two data points and no data between that.

Earlier, without the marker, a line automatically appeared connecting the two points. By default, plt.plot() connects all data points with a line. If we use a marker, it shows only the data points. So, if we do not define it like this, then what will the plot do? Whatever values we take, whatever numbers we take, it will draw a line between them automatically.

Okay, so now we have come to know that we have only two points in this. Now we can take more values — it is not that we will always have only two points. The first thing we have to do is import the Matplotlib library.

import matplotlib.pyplot as plt

After that, we use NumPy, written as np.

import numpy as np

Now, we have to define this. So what do I do now? I write xpoints, define it using NumPy’s array function, and define the array like this:

xpoints = np.array([0,2,3,5,7,10])

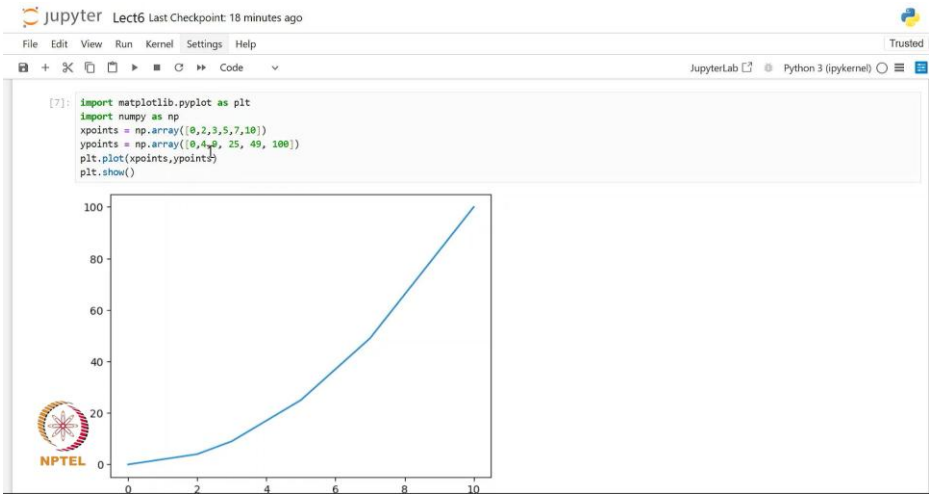
I take values for the y-axis as well. Suppose I take values like:

ypoints = np.array([0,4,9,25,49,100])

plt.plot(xpoints,ypoints)

plt.show()

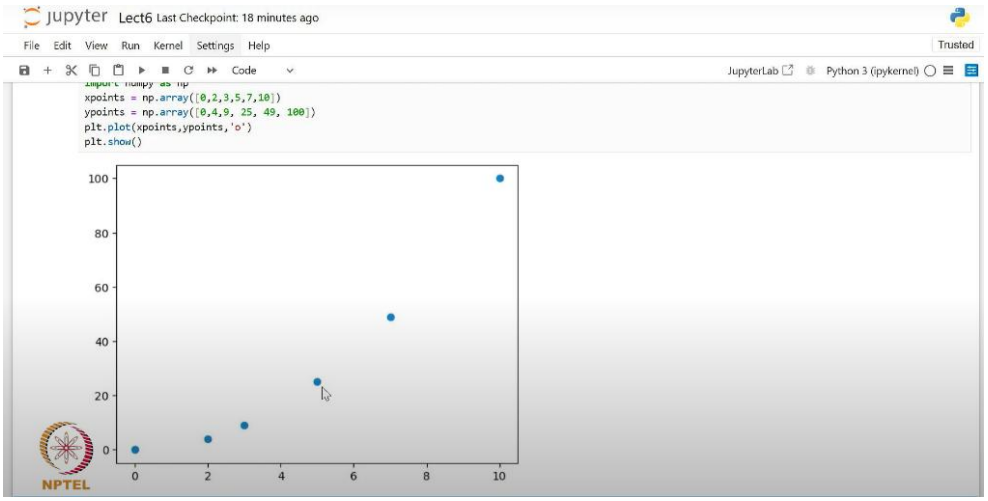
(Refer slide time: 16:17)



Now, when we run this, we will see that the graph is not a straight line. Why not a straight line? Because the elements I took for the x-axis (the domain of the function) were linear, but I squared them for the y-values. So, what is it doing now? It's plotting points like (0, 0), (4, 16), (5, 25), etc.

Now, if I want this plot to only show the values where our points exist (without connecting them by lines), I can disable the line and just use markers. But if I run the plot again as it is, it will automatically connect each pair of consecutive points with a line. It draws a line from (0, 0) to (4, 16), then from (4, 16) to (5, 25), and so on.

(Refer slide time: 16:40)

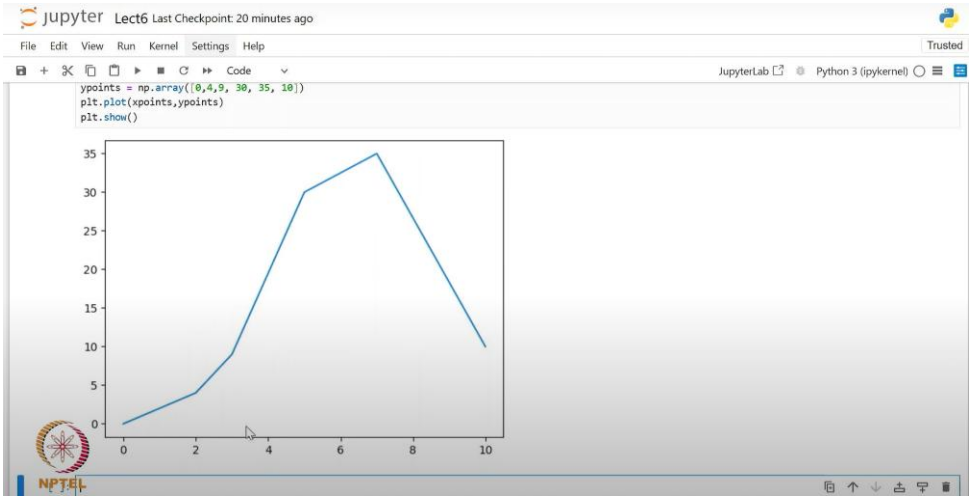


So the plot assumes these are continuous data points. As to draw a line, you need two points. As soon as it gets two values, it will automatically draw a line between them. So, we have this in x and y axis. So, we have draw these points. What plotting do is it will draw lined between them , like between 0 and 2 then 2 to 4 and so on This is why we say it produces piecewise lines, forming what is known as a piecewise linear function.

If I remove this and show the connecting ,we'll automatically be able to interpret the nature of the data. From here, we can conclude that the nature of our data is not linear. Instead, it is square in form — this is parabolic-type data. So, we can plot this very easily.

Let me change this. Now, I'll insert some more data. Suppose I copy the array and change the y-values. Instead of squaring the x values, I now input some random numbers: 0, 4, 9, 30, 35, 10. I enter the new values, and a new type of graph appears.

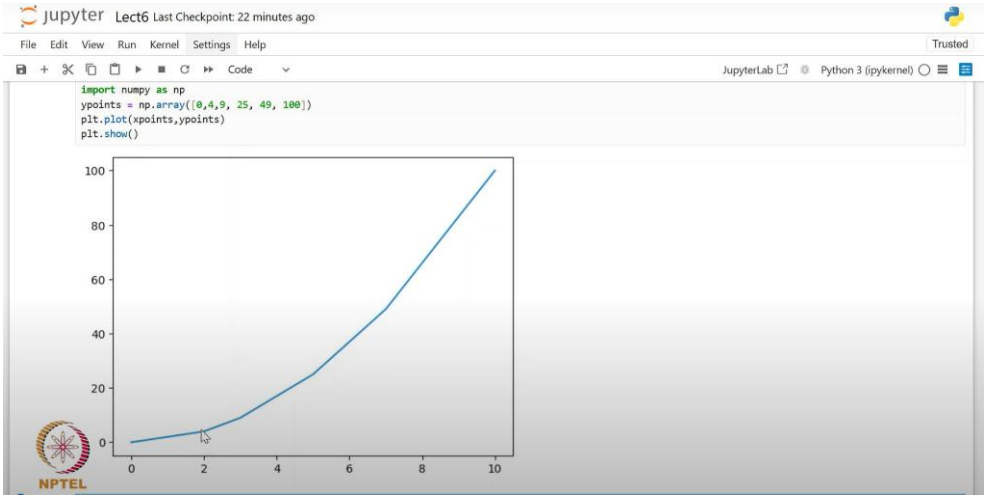
(Refer slide time: 18:41)



In this new graph, the data increases at first, then the values stop increasing as much, and then they decrease. So, in this case, we don't exactly know what type of graph this is. But we can say one thing for sure: the data is not linear. Through this kind of visualization, we can get a sense of the type of data we're dealing with. The most important reason we visualize data is to understand how the data looks. If we plot it, we immediately gain insight into the behavior of the data.

Now suppose I remove the x coordinate and plot only the y-values. Let's see what happens.

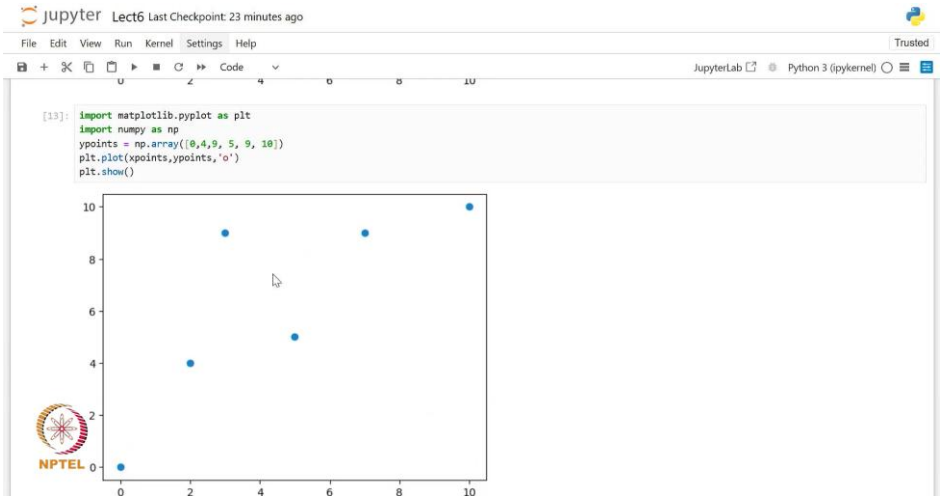
(Refer slide time: 19:57)



In this form, you can see the first value is taken at index 0. Then four values are taken, followed by two values, and so on. If we plot it, Python automatically understands that we have a sequence of six values. It will assume x-values as 0, 1, 2, ... automatically and plot accordingly.

If we want to check what values were assigned, we can print y. You'll see values like 0, 4, 9, 16, 25, 36 (if they were squares). Suppose now I take a different value, like [0, 4, 9, 5, 9, 10] and plot it. Then this value will appear on the graph.

(Refer slide time: 21:06)



However, if we had defined x earlier, it is still in memory. Python remembers the x-values we defined before. So, if I want a linear x-axis, I can redefine x as a linear sequence and then plot it accordingly. So, we can plot like this too. We used plt.plot() and plotted the values.

Now, we also used markers in our plots. These markers help us identify data points clearly. For example, if we use the marker 'o', which is round, it represents each data point with a circle.

We can also use other markers. For example, using '*' will represent points with a star. If I plot with stars, we will get a plot where each data point is marked with a star.

Now suppose I want to represent the data with both a line and a star marker. I can use something like '-*'. This will connect the points with dashed lines and also show stars at the data points. Alternatively, I could use 'o' instead of '*', and get lines with circular markers. This allows us to represent plots in many different ways to better understand how the data behaves.

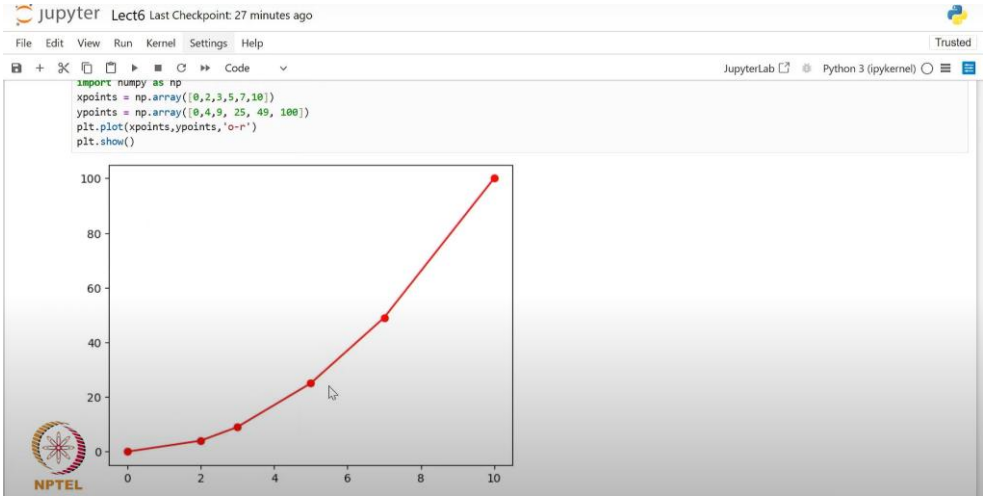
Now, by default, all the plots are shown in blue color. If we want to change the color — say, to red or green can we do that too. The format string in plt.plot() allows us to specify marker, line style, and color all in one go.

So, we should also know that if we want to do our plotting in red colour, we want to do it in green colour. So, I have taken this data. So now what do I do? In this case, what will I do in plotting? I will use a marker. So, I write it here. There is a parameter, we call it format string. So, we write format string in short form fmt. So, what is it basically doing? In which format should our plot come? Okay, we have to write it in its form. So, what is its syntax? So write marker there. Okay, after the marker I will show the values that will come. After that will come how to draw the line and after that will come colour. So like this we can write in a format.

Now like I have written this, I plotted this. Okay, so what did I do? I put a marker here. So I put "o" in the marker and then in the line, write my color red. So I wrote it like this:

plt.plot(xpoints,ypoints, ‘o-r’) and plotted it, so it came out.

(Refer slide time: 25:08)

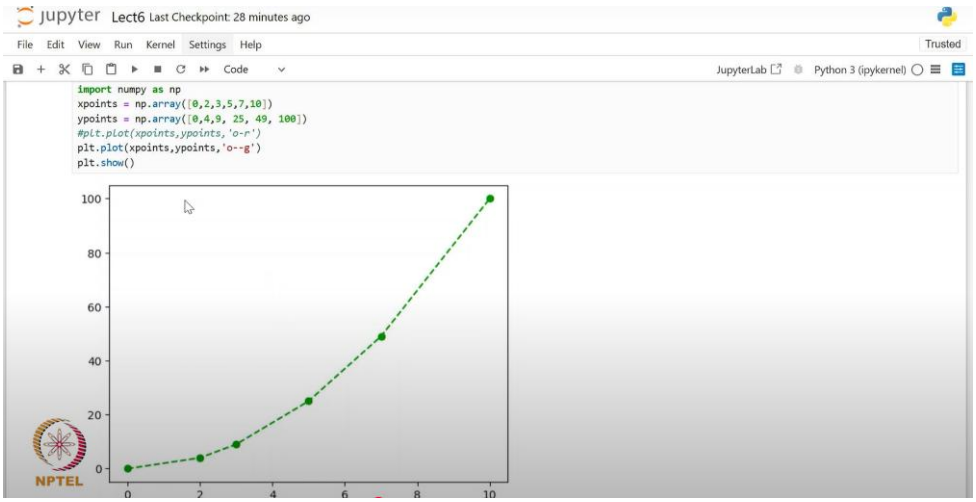


So, what happened in this? See, earlier it was round "o", so from here I took those values. The line that is coming in the middle, It took that line and r means I made the color red. So what did it do to this? It made the color red. So, from here our graph has become red. Okay, if I want to change this, then suppose I just change this plot, this command. Control C, Control I, I comment it out so that it is not used and now I wrote in it. I wrote dash dash line and wrote green here. So see, now I plotted it.

plt.plot(x, y, '--g') # Dashed green line

So what happened? The values which were points are being represented by this, but the line, dash dash line which is there, is being represented by this line and "g" became green.

(Refer slide time: 26:17)



Okay, so like this we can take from this value. Now what did I do? Let me Control V this also, I will comment and what did I do now? I will take a star “*” instead of taking "o" and I

also wrote a colon type and made it black instead of green and pressed enter and blue became blue.

plt.plot(x, y, '*:b') # Star marker, dotted line, blue color

So, what happened now is that our dotted line has appeared and a star is displayed here and "b" for blue, we made it blue. So, in this way we can plot any graph. So now if we have any, like we plotted this, if we have to make this plot a little smaller, then it is possible that we may not be able to see this plot properly. So, what can you do? You can also define the marker size. So, what do I do? Marker size. So, marker size means what should be the size of the markers. So, we also write it as ms. The short form of marker size is ms. So, ms will set the width and size of the marker.

So, I take the same and plot it here. So, I write plot and take the same command. So, suppose I did it here, but I wrote this. If I take any value, then I will define it here. Let's define marker is equal to ‘o’ , I took that. Okay and here I defined MS, marker size. I did that. Suppose I defined it as 10. Okay and here I wrote show.

plt.plot(xpoints, ypoints, marker = 'o', ms = 10)

plt.show()

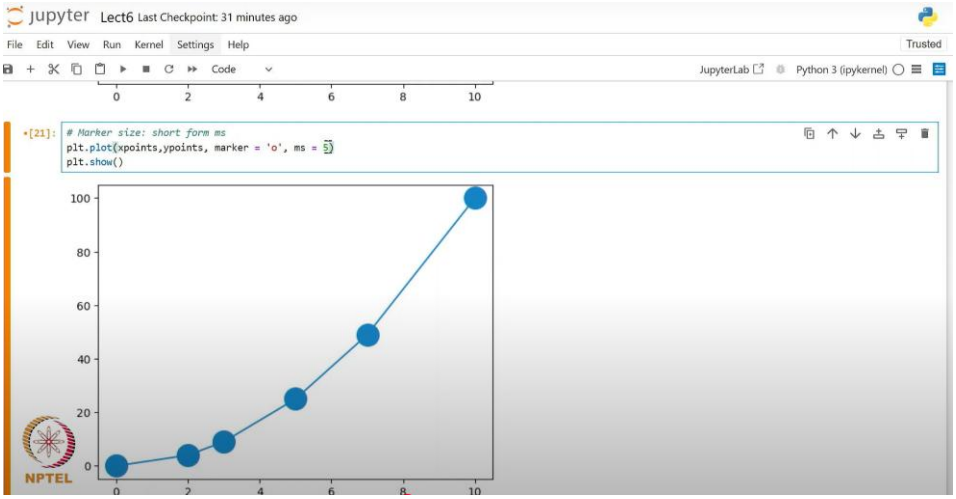
See, now you can see that the size of the marker has become bigger. Okay, so how big the marker size is visible to us. I will make it 20.

plt.plot(xpoints, ypoints, marker='o', ms=20)

plt.show()

So, and see how big the size has become. We are increasing the size of the marker. We have not increased the size of the line yet. We have only increased the size of the marker.

(Refer slide time: 29:13)



So, it depends on how big the markers are to be defined. If we want to make them smaller, then we will define it like this. Okay, so this value is this. The size of the markers, we can define it. Now what can we do? We can also define the marker color. Marker color. So what has to happen in the marker color? That we have to define it. So, we write it as its short form.

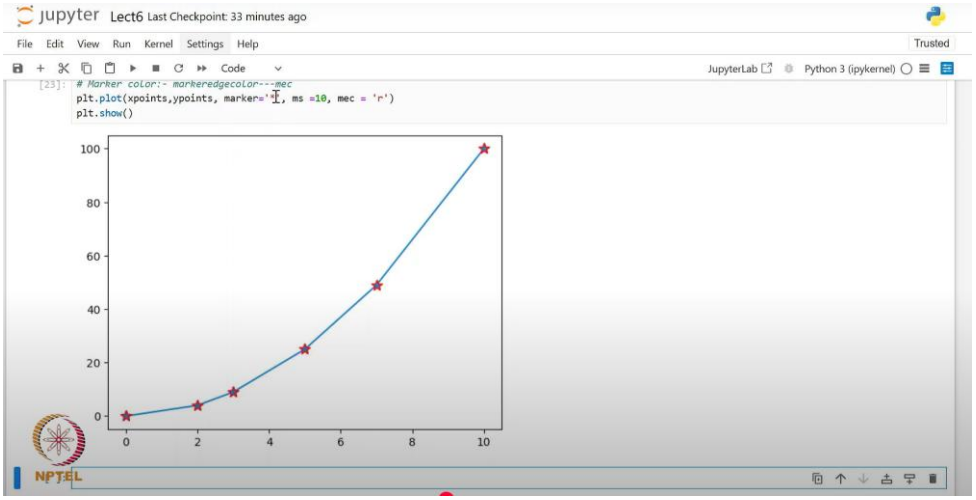
Marker edge color. Like that. This is called markeredgecolor and it can be written as mec which will be the short form. So, this will tell us what color we want for the marker.

So, like I just did the plot. Here I have defined marker. Okay, marker, I have given it star. Let's take star. There are a lot of markers, we have a whole list of them. You can download it from Python as well. You can see what you want. I have made it ms. Suppose I have made it 10. Now I will define it here mec. So, what should be the marker edge color defined? So, I told brother make it red. Okay, so I wrote it here. So, the plot will appear from here and then I said show me the plot as well.

plt.plot(xpoints, ypoints, marker='*', ms=10, mec='r')

plt.show()

(Refer slide time: 31:17)



So, see, now we have the same plot in which the marker is a star whose size is 10 and its color will be red. Okay, so from here its color should be red.

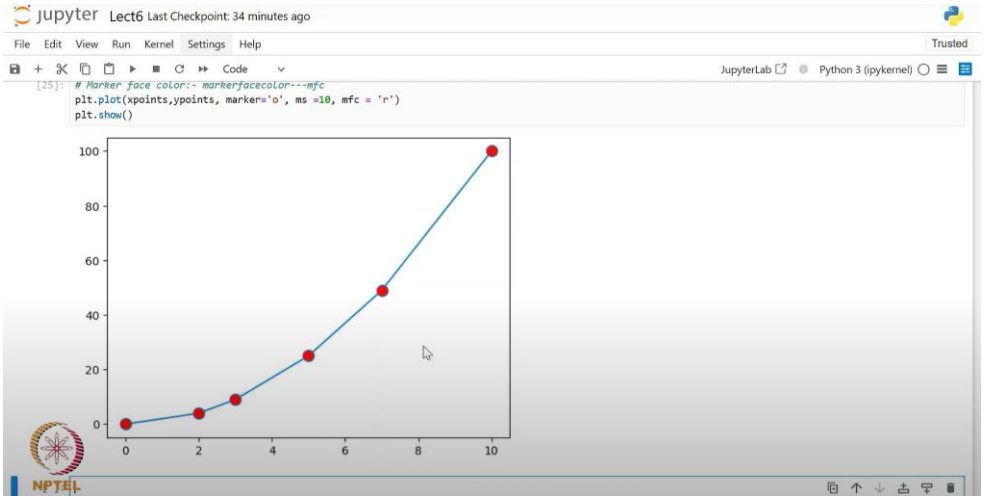
So, the color of this marker should be red. We changed it and applied it. It has appeared. So in this way we can do it. You can also change the color of the markers, its size as well as the color inside it. Now look, the marker that is coming here is round, so the circle of this round circle is red, but the color of the values inside it is now the same as it was before. So, this is the problem. we want that this should not happen. The color inside it should also be the same. So for that now we have one more command. So I will copy it from here. We will write it again if it is unnecessary.

So now this. Now we tell it that the face of the marker should also change. So we call it marker face color. Marker face color. So what will be the short form of this? mfc.

plt.plot(xpoints, ypoints, marker='o', ms=12, mfc='r')

plt.show()

(Refer slide time: 32:40)



So, I wrote MFC and defined it here and plotted it. So, look, now the face of this has become red. So now in our data, it is very easily showing where the points are. Here it is red. It can be seen from a distance that how our array points, the data points, are coming here. So with all these commands, we can show markers.

Now what do we have to do? Marker. So let's go. Now the line is. Suppose we want to increase the width of the line or we want to make the color of the line also. So what would we like to do for that? Now what do we have to do? We have to define the line in Matplotlib. How can we increase the size of the line? Right, so we call it line style. Line style and the short form is ls. LS will change the style of the line.

plt.plot(xpoints, ypoints, linestyle= ‘dashed’)

plt.show()

The previous command. I plotted that. Now what do I do? I am plotting x and y. Okay, now what am I doing? I wrote line style. Line style is equal to. I have defined it. Suppose we define it. Do it. Okay. So, see this line is in dot form. So the whole line is in dash form. Okay, so we have written it in the form of a dot like this.

Now, if I do not want to say line style again and again, then I can write it like this. I can also take its short form. What is the short form? LS.

plt.plot(xpoints, ypoints, ls= ‘dashed')

plt.show()

So what did I do? I wrote it as ls. So what happened? ls became line style. This will be its short form. Now I wrote dash. So I can also write it as colon time. See, dot came. Okay. So it means it is in dot form. Right. What is happening here? It is coming in dash form. So like this we have a list in which we have defined the style, the color of the lines and how can we write it. Because I have told that wherever there are two points, a line will come in between them. Okay, so we can change the color of that line.

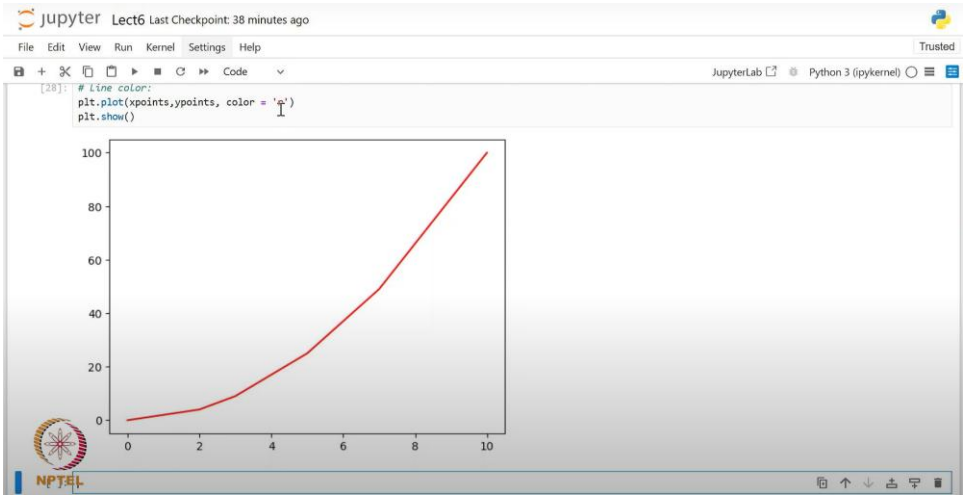
Now no, now I mean I changed the style in it. Now what am I doing? Now I am changing the color of the line. Line color. Or if you want to change the line color, then how to do it?

plt.plot(xpoints, ypoints ,color='r')

plt.show()

So, I will do this. I also did control v and now I wrote this. Suppose what do I do? I write red and here I write color. Okay, so now let us see what happens. See, the red color came. So, the default values that it takes in plotting are line and it is not taking dash, but I have defined that the value of red will be red.

(Refer slide time: 36:17)



You will say that no, I wanted a dot. Okay, so I will do this as well.

plt.plot(xpoints, ypoints , ls = ‘:’ , color='r') # Dotted red line

plt.show()

So this dot came and it came in the form of red. Okay, so like this we can define all the things. Now we can define it. Even we can write marker inside it. So, the marker will also come. Okay, its size will also come. So, we can define all these values like this.

Now what I have to do is to increase the size of the line. Okay, so I write line size. Now let us see how the size of the line will increase. So, the definition that is used for that, the command is linewidth. Okay, so this will have to be defined.

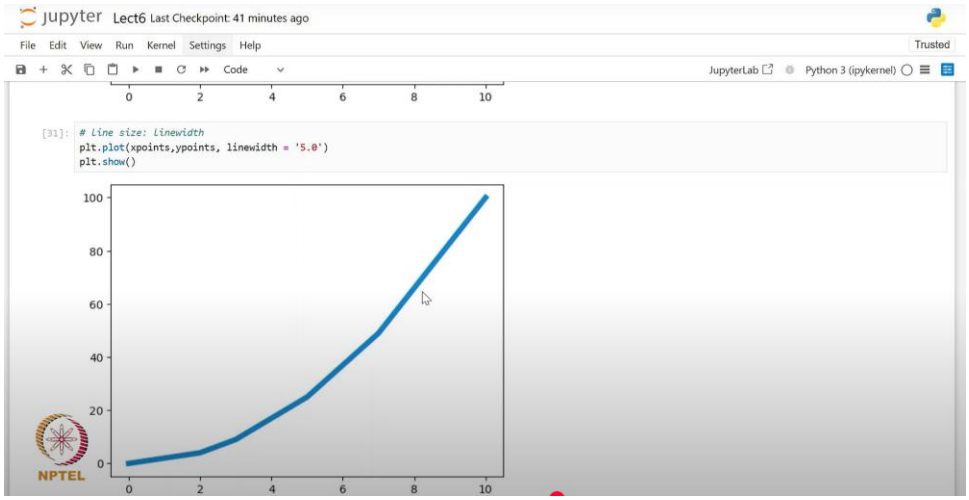
plt.plot(plt.plot(xpoints, ypoints , linewidth= ’10.0’)

plt.show()

See how much the width of the line has increased. Right?

So, if we have a plot of our line, because when we do some work, many plots can be made. Subplots can be made. Their size will have to be reduced. So, if we want to make it smaller, then it is possible that the default values of the line may not be shown to you. That means it may not be visible at all. So what will you do? You will decrease or increase this value. Now, like there is 10 in it, then I suppose it will be 5. So, this is 5. Right? So, this value is showing quite well that its line width will be shown. That is, even if we make this graph smaller, then in this way we can see it.

(Refer slide time: 38:51)



What can we do? We can define the width. We can also define the markers. We can define everything. So where is it used? All this is used when we put multiple data in a single graph. So, we can see what happens. Multiple lines in the same plot.

import matplotlib.pyplot as plt

import numpy as np

Now I defined one data x1, so I defined x1 as a np.array and I took its value. I took the value 0, 3, 5, 7, 9, and 10.

x1 = np.array([0, 3, 5, 7, 9, 10])

Okay, so after that I defined y1, now I defined the value, so I defined a np.array, I defined the values in it. Let me square them, so 0, 9, 25, 49, 81, 100.

y1 = np.array([0, 9, 25, 49, 81, 100])

Now what did I do, I defined one and I have defined the array. Okay, so let's say I take the same x1. Now I have defined another y2, I have defined y2 and I have written y2 like this. Let's do it like this, let's define:

y2 = np.array([0, 6, 10, 14, 18, 20])

I have defined this, okay I have done the second one. Now what do I have to do? Plot both of them and see.

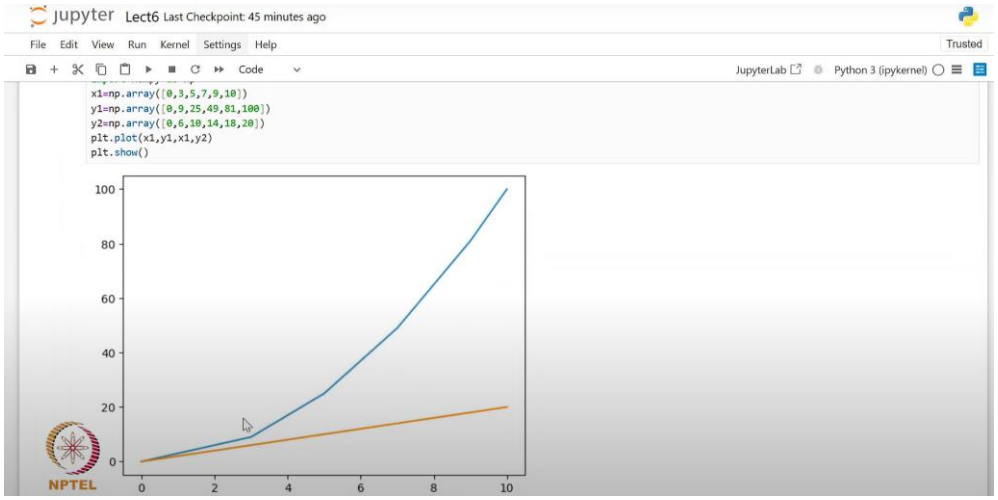
plt.plot(x1, y1)

plt.plot(x1, y2)

plt.show()

So, this comes now. What is it doing? It will plot both of them together: x1 and y1 which we have as one pair of data and another pair of data, I have defined x1 and y1 and plotted it.

(Refer slide time: 42:58)



Now look, in this we have two graphs, one with this color and the other with a different color. So, what happens is that it automatically takes different colors by default. Okay.

So now the colors that we have come to us come in this form. Now what we want is that their size should also be different, the colors should also be different and the ones that are to be displayed should also be different. Okay.

So, let's plot. Right now, we have colored them like this. Now what we want is that we give a title to this graph, name it something. Okay, so what I do is define the title here. It will be in the form of string.

plt.title("Plot showing two functions")

Now what I want is to define the axes as well, which are the axes. So, I write 'Axes' or 'xlabel'. So:

plt.xlabel("X -axis")

plt.ylabel("Y-axis")

So now let's see what will happen if I change it. So, if we see this, the title will come up. Plot showing two functions , 'X-axis' and 'Y-axis'. Okay, so we can show it like this.

So, I have plotted it . Now what do I do? After this plotting, I want that these values or their colors should change and their fonts should also change. So now I can do this.

Now what do I do? I define a font for myself, like we have many fonts. So, what did I do, I defined a font for myself, font1. So, font1 is a family it will define what type of we should have it.

font1 = {'family': 'serif',

'color': 'black',

'size': 10}

After that whatever color it is, I gave it values and defined it. I took black. Okay. I defined and size is 10. Now what did I do, I defined font2.

font2 = {'family': 'serif',

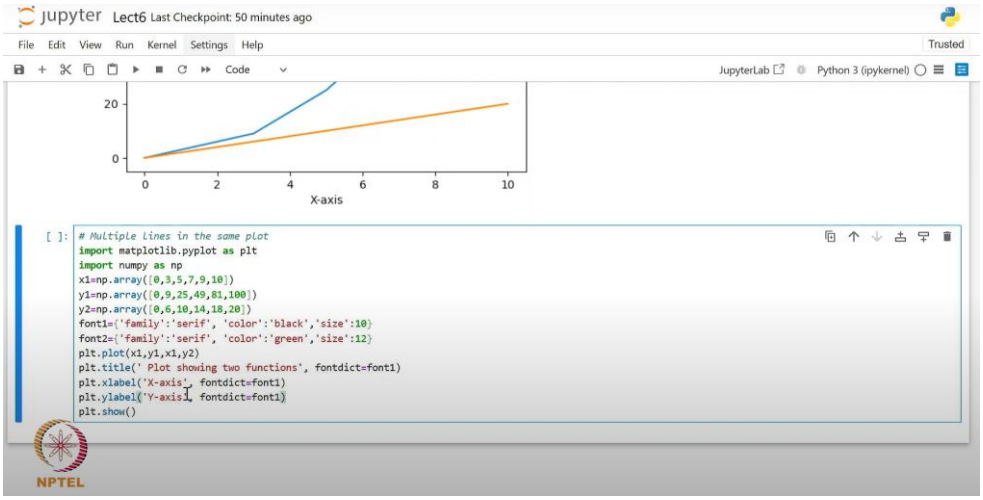
'color': 'green',

'size': 12}

Okay, in that, I took some other color, I took green, I took its size 12. So, I took different sizes.

Now what did I do, I plotted this. Now what do I do?

(Refer slide time: 48:23)



So the label, the title:

plt.title("Plot showing two functions", fontdict=font1)

plt.xlabel("X Axis", fontdict=font1)

plt.ylabel("Y Axis", fontdict=font2)

plt.show()

Okay, so let's see what happens. Here the graph has come, now the font has changed. See the "Plot showing two functions", the font has come written in the form, its size is 15. Okay, because we had defined font1 and this x-axis and y-axis have been defined like this.

So, the x-axis is in black and I had defined green color, it came there and its size is 12. Similarly, I want to make the size of font1 15, okay. I defined it like this, so it became 15. So ,it came written in big size, x came big, y one came small.

So, what will happen with this is that we have to make a lot of plots, so every plot is defining some different function, defining its behavior. So, for that what we do is that there should be classification of their function.

And after that what should be the axis, and so these will be small plots. So, after that we will not be able to see them properly. So for that we can define the function.

Suppose we have some such values. This has come to us. So, now what do I want to do. Now see the graph that is coming in this form. But you must have seen that our graph paper shows grid. Okay.

So, what do we have to do to show the grid?

Now I want that my graph should not be like this, it should show grid. Okay. So now we will define the grid line. How to define the grid line?

Now see, I have taken this plot or I have taken array. Okay, so as we have defined it. So, I defined it. After that I defined the title of the plot.

plt.title("New plot")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.plot(x1, y1)

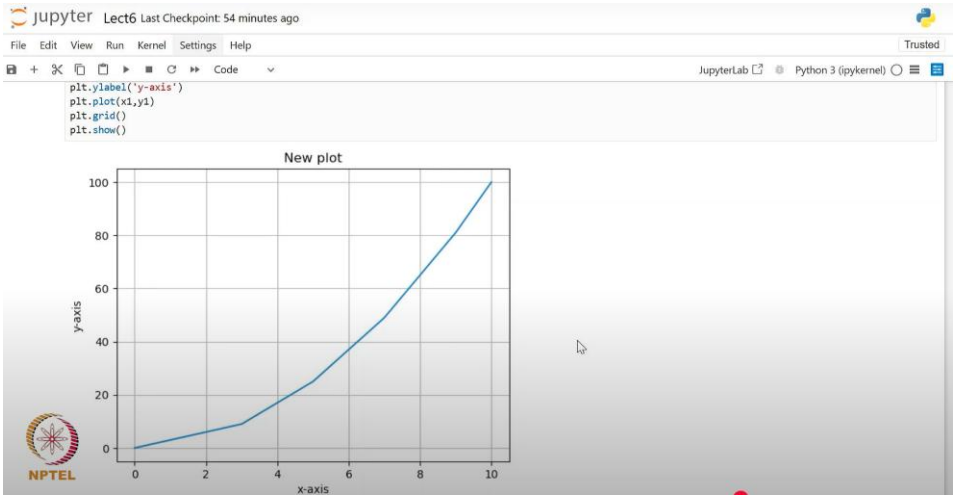
plt.grid()

plt.show()

So what did I do now? In this, our graph – now we called the grid() function. So let's see what happens. See it came.

So what we wrote was "New Plot", I wrote new plot x axis, y axis and the same plot, but now the grid is done.

(Refer slide time: 52:16)



So from the grid we came to know that our graph paper is ready, we showed it easily in it. So, what happens from here is that we get the benefit, with the grid we can easily measure things – that which point has what values, how many values are coming, right? We can easily see all these things.

So, this is the advantage of defining a grid for plotting so that we get to know what the grid is showing, what values it is showing, our points and what are coming where. So, in this way we have shown this graph.

Now, as I said, we have defined all these things to make the graph more interactive so that it is helpful in visualization. We can know by looking at the graph what is being shown by this data because till date the graphs that you have shown are defined for functions like:

f = sin(x), cos(x), x**2 # Polynomial, quadratic, parabolic

So, we have taken all those functions and we know that their graphs will be of this type, but now we do not have functions, now we only have data, so we have to analyze the data, we have to see what it is showing. So, what do we have to do for that first of all? We will have to do visualization and we will do it from here.

Now see what we have to do now, what we do now is that we have to make small plots, we will call them subplots. So now we define subplots. What does subplots mean?

In the graph that we made in the figures, we showed one data in one graph, the second data in the second graph, then the third data in the third graph and their size became smaller.

So now what did we define for subplots? What do we define? We define a matrix dimension. So what did we have to do for making subplots? Subplots are shown with matrix. So how do we do it?

import matplotlib.pyplot as plt

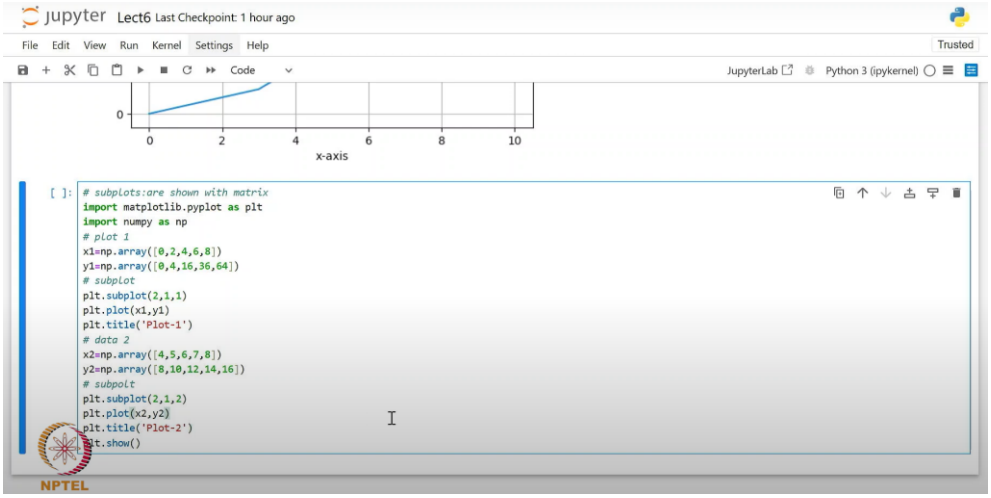
After that, I defined numpy as np, I wrote this. Now what do I do? I define plot 1. I defined x = x1, I defined x1, I defined it equal to—I wrote numpy.array and after that, I defined its value: 0 2 4 6—I defined it—8 five value. After that, I defined y1 = np.array, okay, and I wrote it. Suppose let's define y: 0, 4, 16, 36, 64 y. Okay.

Now what do I do? I want to plot this. So, what did I do? I defined subplot. Okay. So ,subplot—now what will we define? See, now I will write subplot. So how do we write this? So I wrote plt.subplot. Okay, plt.subplot and I defined its value (2,1,1). What does it mean? I defined a matrix which has two rows, one column, and its first element is one here. So, plot it at that place.

And after that, I wrote plot: plt.plot, and suppose I defined x1, y1, I wrote it like this. Okay. And I also gave it a title and wrote in the title: I wrote "plot 1", like this. Okay. Now what did I do? I do the same thing further and I defined it and defined another array. Okay. Now I am doing all the work for the second plotting, so I define data 2.

So, what did I define? I defined x2 = np.array and I defined it. Okay. I defined it: 4, 5, 6, 7, 8. Okay. And I defined y2 = np.array and I defined the values inside it. So I took any value of it. Okay. I would multiply it by t. Let's say: 8, 10, 12, 14, 16. Now what did I do with this? I want to plot this also. Okay. So, I defined subplot in it.

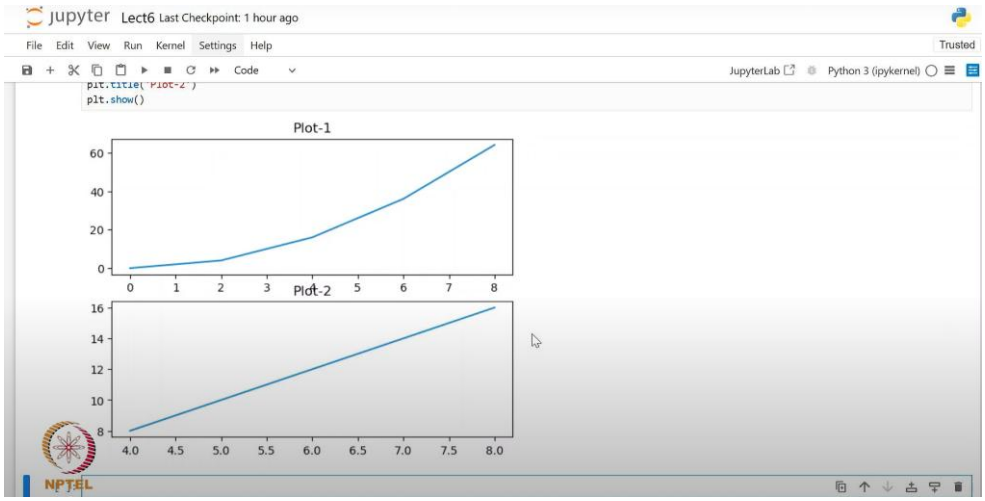
(Refer slide time: 59:55)



plt.subplot—I defined it. Now this was our matrix(2,1,1). Now I will take its value, so I define it here. Right? So what is this? There were two row and one column. So, it means the data will come vertically. This time I wrote it. After that, let me copy it, I did control C, I did control V. And after that, I showed all the plots.

It came. See what it is doing now. Now let's see that we have ‘Plot-1’. It had to be written in a string; the title has to be written in a string. Okay. So now the values that came to us are this. Okay. So let’s run it and write it correctly. We did not write it. We plotted. See, this came.

(Refer slide time: 1:00:03)



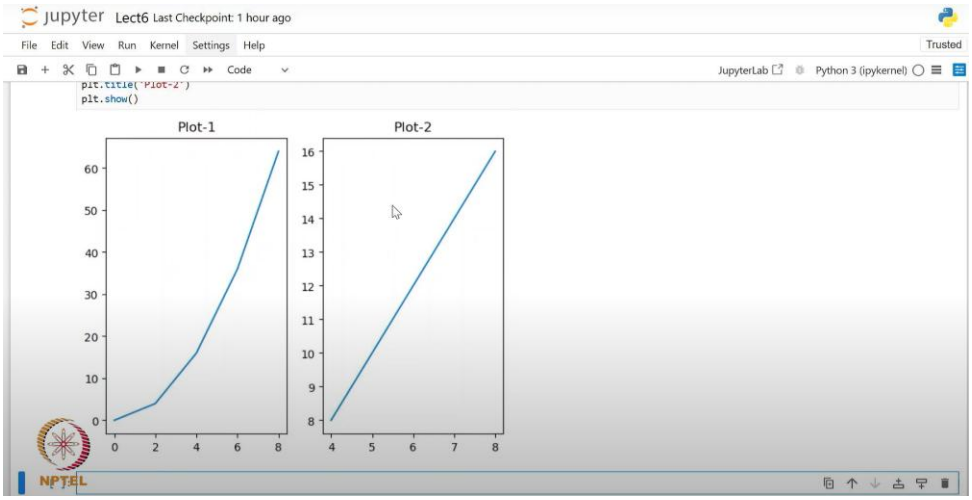
"Plot One" came, whose values are this. So I had squared it, so it came in the form of a square. And "Plot Two" is a straight line because it was multiplied two times.

And if we see in this, what did I say? Two rows, one column. So what did we do in two rows, one column? We made it in the form of a column matrix. So a matrix will be formed which has two rows. So the first element in the first row was "Plot One", and the element in the second row was "Plot Two".

Now what I want to do with the same thing is that I want to write side by side. We should write the plots in horizontal form instead of vertical. So, I will copy this. I copied it and I pressed Control-C and I pressed Control-V here. Now what I did was I defined a matrix

which has one row and two columns. So, there is one row and two columns. I did not do anything else. I just defined it like this. Now let's see what happens. So, this has come. So, see, side by side, horizontal has come. So "Plot One" has come, and the plot has come here. Why? Because we have changed the dimension of the matrix that we had defined. So, like this, we can change the dimension.

(Refer slide time: 1:01:24)



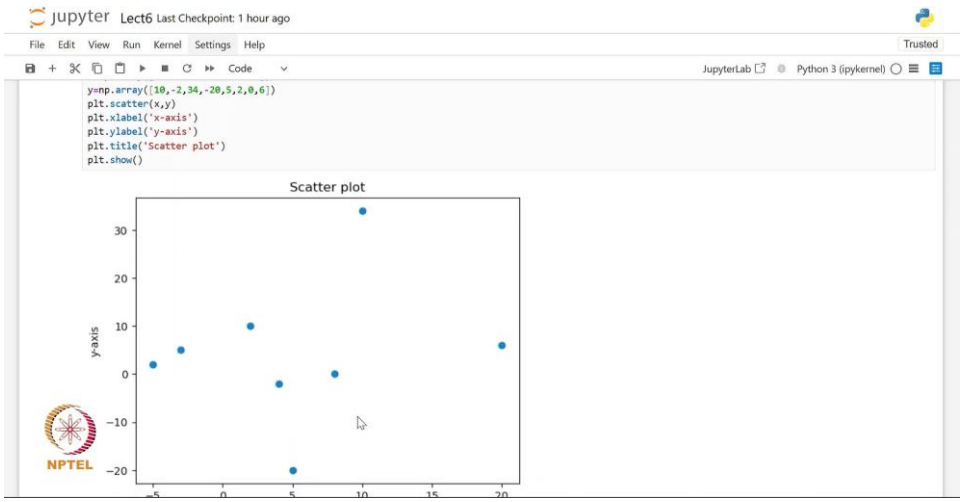
So, plotting of all is done. Title is done. Subplots are done. Now we have shown how we can create subplots. So now we want to do one more command which is a very important command. So, like this, we can create many subplots. I will change the matrix by it. I can define a 3 by 3 matrix. If we have nine small subplots, then we will create a 3 by 3 matrix and locate each position in it. With the help of this subplot, which is ours, we will locate it with this and we will have nine plots.

So, now we have one more command which will be useful for us and that is scatter. Scatter plot. So, what is a scatter plot? It tells us how the data is scattered. So, what do we do? In the beginning, we have to write this: numpy. So I will copy this. Now I have some data x, so np.array like this. I have defined a data. We have a lot of data. So I wrote that data: 2, 4, 10, 5, -3, -5, 8, 20. I have defined it like this.

I have defined another data: y = np.array(). Now I have defined that. So, I have defined it like this. Suppose: 10, -2, 34, -20, 5, and after that I took 2, 0, 6 like this. Okay. So, the second one I defined. Now, what we have to do is to plot the scatter. So, we will write plt.scatter and define this. Let's plot both the x-axis with strings. Okay.

And what I do to them, I will also label it as plt.xlabel so that we can know what is there inside. So I wrote "x-xis", okay, like this. I wrote plt.ylabel and I defined it. Let's write "y-axis". In the title, I wrote plt.title. Okay. So, we just wrote that it is a 'scatter plot'. After that, we defined plt.show.

(Refer slide time: 1:05:44)



Okay. So what did we do? We defined an array that we want to show that how both are scattered. We have a two graphs and how are they scattered with each other. So, we defined the x-label, defined y-label, defined scatter plot. So, lets see, what happened in this is that the data points that we have are in scattered form. So, this is what we call a scatter plot.

So, these values or scatter plot let us know that our data is random data in this case. So, we can plot it like this. So, we have plotted one set of data, we can plot two sets of data, we can plot three sets of data. Same is true in that. If we plot it like this, we will get a scatter plot. So, we use this thing a lot when we have a lot of data. By scatter plotting it, we can know that the data is random, what type of random data it is.

So, all these things that are happening are for visualization. So today we stop here. So today we saw very important Python commands. We saw how we can do visualization with the help of Python. So today we will finish our basic commands related to Python, and from the next lecture we will start the main scientific computing, which is our numerical method. So, thank you for watching this lecture.