Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture – 04
Introduction to Social Networks
Introduction to Phyton-2

(Refer Slide Time: 00:08)


```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]:
```

Hi everyone, in one of the previous videos you would have seen how we can make use of list data structure and we saw various functions that we can use with list. In this video we are going to look at yet another data structure that python provides and that is dictionary. So, we are going to see how we can create a dictionary and we are going to look at various functions that we can use with it. Before we get started; let me tell you in which situations, it is ideal to use a dictionary data structure.

So, dictionary is used when we have to store elements that have some sort of attribute attached with them; that is we have some elements and we also have to store some sort of property along with every element. In other words, in dictionary every element has two things; that is key and value, so there will a key value pair for every element, where these values stores the attribute of the key. Another thing to note is that dictionary is stored in memory as hash tables, which makes it faster than as compared to other data structures.

So, if you have some million entries and on one side you store them in the form of list and on the other side you store them in the form of dictionary, accessing these elements through the dictionary will always be faster. This is because accessing elements in dictionary does not go in sequential manner. So, let us get started and I believe you people might be comfortable with ipython, so I will continue using that.

Now, let me create an example dictionary here; assume my aim is to store name the few people and along with that I also want to store the age of these people. So, here I have this age which is an attribute that is attached to every person. So, here I have two things the key is the name of the people and value is their age, so this is the sort of information that I want to store; let me create an example dictionary here. So, I am going to add a few people's names and their age, let me add few elements here; so I add another element and I add another element alright. So, this is how we create a dictionary, so you note here that every element has two things; so this is the name which is the key here and this is the age which is the value here. So, this is the key value pair and we separate them with comma and we add another element and so on and the elements are kept inside these basis; so you can print this dictionary.

An interesting thing to note here is that these values might not necessarily be integers. So, rather they can be strings, they can be floats, they can be another list as well, they can be another dictionary as well. So, this is sort of flexibility that dictionary will provides us while its operation. Now, let me see and let me tell you how we can access the elements of a dictionary, so it is important that whenever we access we should have the key. Now, let me show you usage which does not work with dictionary; let me try accessing the first element by tracking this d; 0.

(Refer Slide Time: 03:32)


```
IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]: d[0]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-3-17371c6688f6> in <module>()
----> 1 d[0]

KeyError: 0

In [4]: ▋ I
```

Now, this is not going to work; now this is because there is the elements in dictionary are not associated with index unlike the case of a list. For example, in list you can access the elements using their index; however, it does not happen in dictionary; this is because there is not association of any index with the elements. The only association that takes place in a dictionary is between the key and the value. So, just in case you want to access this element annie, you would have to access it through the key.

(Refer Slide Time: 04:04)


```
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: d = {'annie':25, 'yoyo':35, 'sid':65}

In [2]: d
Out[2]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [3]: d[0]
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-3-17371c6688f6> in <module>()
----> 1 d[0]

KeyError: 0

In [4]: d['annie']
Out[4]: 25

In [5]: ▋ I
```

So, I will write the key here and I will be getting the value that is 25, so this is how we access the elements of a dictionary. Now, why is this sort of thing happening, so d; 0 as I showed you is not working, this is because the order of elements in dictionary is not important. Basically, we cannot control the order in which the elements are displayed; let me print dictionary d.

(Refer Slide Time: 04:30)

This is what I wanted to show you, we had added first annie and then we had added yoyo and then we had added sid. However, when we are printing this dictionary; we are getting a different order. Now this is because it is up to the dictionary how it is storing the elements in the memory, so they are not like; they are not stored in the same sequence in which we add the elements.

Now, let me show you how we can update the elements in a dictionary; for example, I want to change the value of one of these elements.

(Refer Slide Time: 05:00)

```
KeyError: 0

In [6]: d
Out[6]: {'annie': 25, 'sid': 65, 'yoyo': 35}

In [7]: d['sid'] = 75

In [8]: d
Out[8]: {'annie': 25, 'sid': 75, 'yoyo': 35}

In [9]: d['john'] = 15

In [10]: d
Out[10]: {'annie': 25, 'john': 15, 'sid': 75, 'yoyo': 35}

In [11]: del d['john']

In [12]: d
Out[12]: {'annie': 25, 'sid': 75, 'yoyo': 35}

In [13]: clear
```

So, assume I want to change the value of this element to be this and then I print the dictionary, so you see the value has been updated and now let us see how we can add more elements into this dictionary. So, let me add one more person here say john and I want its age to be some this number and now let me print this, so this element has been added into the dictionary.

Now point to be noted here is that; the syntax for adding an element like this and the syntax for updating some value in the dictionary are the same. So, just in case the element is not presented to be added and in case the element is present, its value will be updated; so it is that simple. Now, let me show you how we can delete entries from a dictionary, so the keyword for that is del. Now, assume I want to remove this element john out of this dictionary, so I will access it using the key again as I told you. So, I will write this and this element should have been removed; let me check I will print this, the element has been removed.

Similarly, if you want to remove all the elements from the dictionary; you can also do that by using a simple function that is d dot clear. So, what this function do, it will remove all the elements from the dictionary, but the dictionary will stay, so you can add more elements later. I will not use it because I do want this dictionary as if now and in case you want to remove the entire dictionary from the memory, you have another

keyword; you can just write del d, d is the name of the dictionary and the dictionary will be removed, so again I will not execute it.

(Refer Slide Time: 06:44)

```
In [14]: d.has_key('annie')
Out[14]: True

In [15]: d.keys()
Out[15]: ['yoyo', 'annie', 'sid']

In [16]: d.values()
Out[16]: [35, 25, 75]

In [17]: d.items()
Out[17]: [('yoyo', 35), ('annie', 25), ('sid', 75)]

In [18]: for i in d.items():
    ....:     print i
    ....:
('yoyo', 35)
('annie', 25)
('sid', 75)

In [19]: for i in d.items():
    print i[0], 'has age: ', i[1]
```

Now, let me tell you a few functions that might be useful while we are doing the analysis of social networks. One of the functions that is quite used is; to check whether an element is present in the dictionary or not. So, assume I want to check whether the dictionary has an element annie or not, so I will write d dot has key; so this is an important function its quite frequently used. So, since I do not; I cannot see the entire dictionary, I can just execute this function and see whether annie is present in it or not, so its gave me true which means the element is present.

If I want to access all the keys from the dictionary, I can just write d dot keys and it will give me a list. So, this is basically a list of all the key values from all the elements, similarly if we want to access all the values; we have this function; d dot values, again it gave me a list, which is having all the ages of all the people.

Now, there is one more function which is again quite frequently used and that is items. So, in case we want to access key as well as value for every element; we can make use of this function d dot items. So, you see here again we got a list, but the element of the list are basically tuples, where every tuple has two things; first is key and second is value. Also, if we want to access these elements one by one; we can use a for loop, let me show

you that. So, what I want is; I want to access the individual elements out of these tuples that we are getting from these items function.

So, I will write for i in; d dot items, so this d dot items is going to return me a list of tuples and I am just going to print them; let us see. So what we did? We accessed these elements one by one that is all. Now, if you want to access key separately and value separately; you can use the index of these elements. For example, here I can write i; 0, so i; 0 will point to the key here and i; 1 will point to the value here, so I can write i; 0 has age for example, i; 1.

(Refer Slide Time: 09:05)


```
In [16]: d.values()
Out[16]: [35, 25, 75]

In [17]: d.items()
Out[17]: [('yoyo', 35), ('annie', 25), ('sid', 75)]

In [18]: for i in d.items():
    ....:     print i
    ....:
('yoyo', 35)
('annie', 25)
('sid', 75)

In [19]: for i in d.items():
    print i[0], 'has age: ', i[1]
    ....:
yoyo has age:  35
annie has age:  25
sid has age:  75

In [20]:
```

So, what we are doing here is; we are accessing the individual elements of the tuple by the index, so that is always possible.

(Refer Slide Time: 09:13)


```
In [22]: d1 = {x:x**2 for x in range(11)}

In [23]: d1
Out[23]: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81,
10: 100}

In [24]: d1 = {x:x**2 for x in range(11) if x%2 == 0}

In [25]: d1
Out[25]: {0: 0, 2: 4, 4: 16, 6: 36, 8: 64, 10: 100}

In [26]: d1.has_key(3)
Out[26]: False

In [27]:
```

Now, let me show you how we can quickly create a dictionary, where the elements follow some sort of pattern. So, I am going to make use of for loop for creation of a dictionary, so let me create this dictionary; where the elements should be of this sort and I will use for loop for x in range say 11; I am sorry in 11. So, this dictionary is created, so the elements in the dictionary are of this sort, where key is x and the value is x square. So, this operator you might be knowing in python, it is exponentiation operation and I am using for loop; where the x is going from, so this range returns a list going from 0 to 10, so 11 is excluded.

Let us print this dictionary and see what it contains, so we have 0, 1, 2, 3 up to 10 as keys and the squares as the values. So, this is how we can quickly create a dictionary; we can also play around with it. For example, I want only even numbers; so I can add another condition here, if x mod 2 is equal to 0 that is all. So, let me see what the dictionary contains, its now contains only even numbers are is the values and sorry even numbers as the keys and squares as the values.

We can also check the function that we just learnt; has key, so if I write even dot has key; if I write 3 here, it should return false because there is not odd number as the key here. So, I think that was our pretty much basic introduction to dictionaries.

Plotting is an integral part of any sort of analysis, since in this course we are going to analyze social networks; we might need to plotters us to better visualize them. Therefore,

in this video we are going to see how we can make use of the package called matplotlib, which is provided by python for plotting purpose.
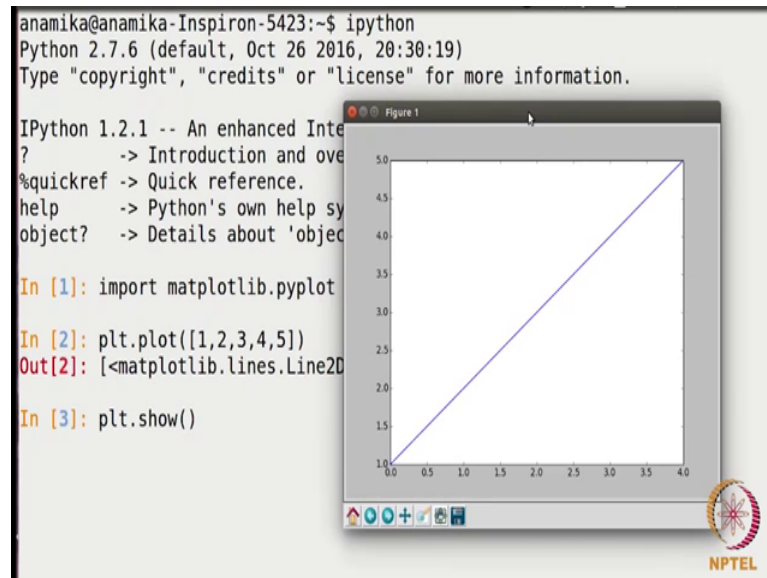
(Refer Slide Time: 11:15)


```
anamika@anamika-Inspiron-5423:~$ ipython
Python 2.7.6 (default, Oct 26 2016, 20:30:19)
Type "copyright", "credits" or "license" for more information.

IPython 1.2.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import matplotlib.pyplot as plt

In [2]: plt.plot([1,2,3,4,5])
Out[2]: [<matplotlib.lines.Line2D at 0x7f5f46224650>]

In [3]: plt.show()
```

So, let us get started, so you might be to import this package first of all. So, the package name is matplotlib, inside matplotlib we have pyplot which contains all the function for plotting. I am going to give it a short name so that it is easy to use, now this means we are going to use plt instead of matplotlib dot plyplot now on; so that will make things easier.

This package has a lot of functions for plotting and to customize our plots as well. The very basic function that this function has is called plot, so we are going to first use that function plot. So, I am going to write plt dot plot, after that I will be giving various parameters to this plot function. Now again this plot function has various versions, to start with we are going to look at the basic version of this plot function which accepts a single list as a parameter. So, what I am going to do is; I am going to pass a list having some 5 elements here alright, so this is a very basic usage of this plot function. In order to see the plot we need to use a function that is called show, so I am going to write plt dot show.

(Refer Slide Time: 12:39)

When I do that, I get this plot in the other window and as you can see this is very basic plot with no customization, there is no title of the plot, there is not title on the x and x and y axis and you see the x axis is starting from 0 and the y axis is starting from 1. So, let me tell you the reason for it; we called plot function with single parameter with which was only one list. Now when we do that, the values at the past in the single list they are taken as y values, which means this 1, 2, 3, 4, 5 are taken as the y values of the plot and since we have not given any values for the x axis, they are automatically taken up by matplotlib.

Now, since the index in python starts from 0; the x values also start from 0 automatically, which means x is equal to 0 is assigned to y value of 1, x is equal to 1 is assigned to y value of 2 and so on. So, you can see in this plot; x is equal to 0 has y (Refer Time: 13:44) 1, x is equal to 1 has y (Refer Time: 13:46) 2, x is equal to 2 has y value of 3 and so on. So, of course we can always change that, but when we provide only one parameter through this plot function this is how it happens. But we can always customize things; we can make this plot specific to our requirements, so we are going to see how we can do that using other functions. Now, I am going to close this window and I am back on the command window.

(Refer Slide Time: 14:17)

Now, let us look at some versions of this plot function; let us use your plot function where we are passing both x and y values. So, what we are going to do is; we will be passing two lists as parameter, the first list will be the values for the x axis and the second list will be the values for the y axis. So, I am going to pass say 1, 2, 3, 4 and 5 is x values and for y values, let me just pass the squares for example, so this is what I am passing here; so I am explicitly passing both the x and y values alright. Now to see the plot, I will write plt dot show and I see this plot.

(Refer Slide Time: 14:56)
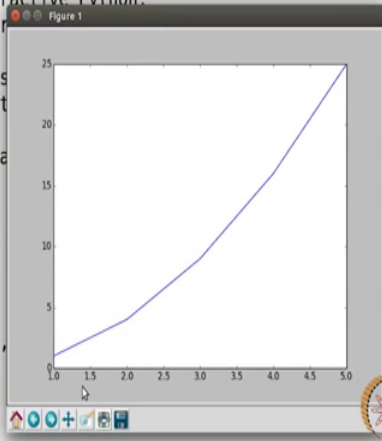
So, here the values; the corresponding to 1, we have 1 corresponding to 2, we have 4 and so on. Now, I am closing this window; I am going to show you some more versions of plot function.
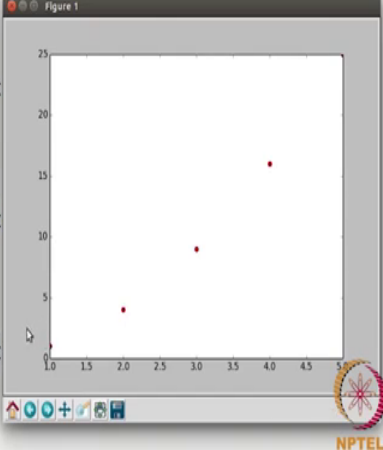
(Refer Slide Time: 15:15)

Let us get back here, so by default we were getting a blue line here as the plot; we can always change it, we can change the colour, we can change appearance of the plot. For example, by default we were getting a line; we can always change it to some points. Let me give you an example here, so I want the plot to be red in colour and say I want the dots; the points and not the line. So, I will write r; o here; let us see what we get, so every time you call this plot function, you have to write plt dot show as well; in order to see the plot.

(Refer Slide Time: 15:50)
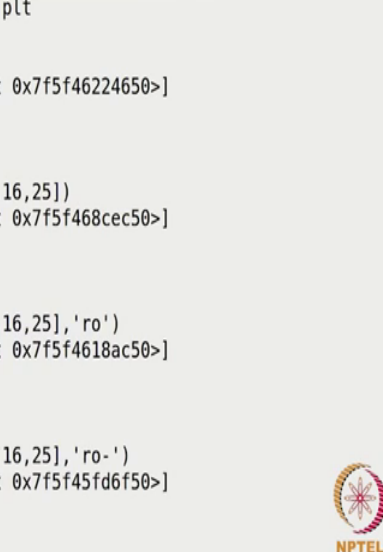
Now, here you see instead of the line I am now getting the points and that too red in colour.
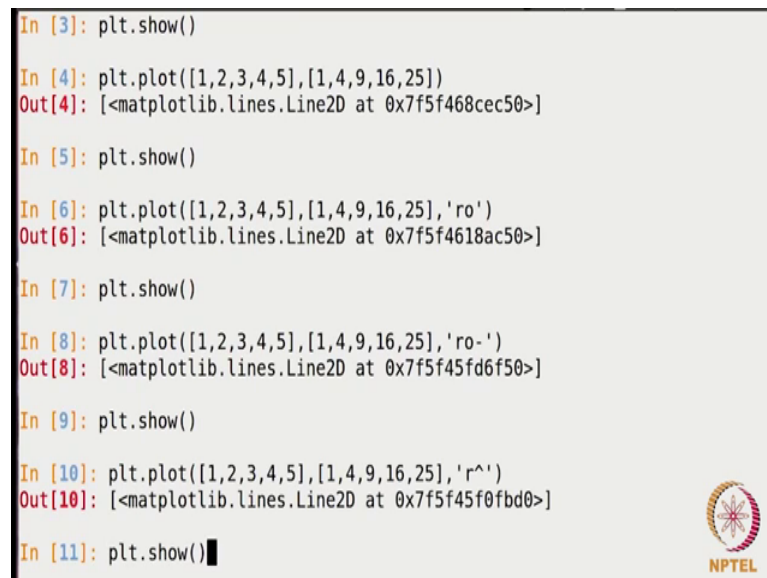
(Refer Slide Time: 15:57)

And in case you want the line as well, so you can put a dash here enter and then plt dot show, now you get the points as well as line.
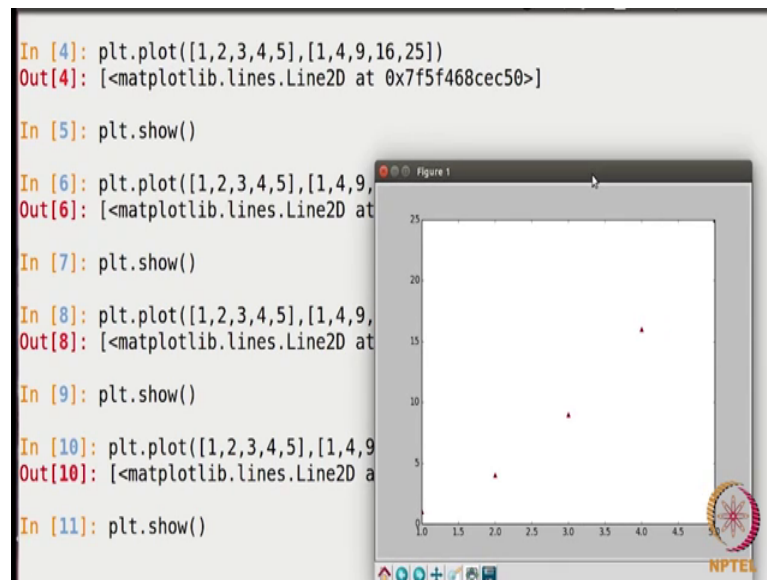
(Refer Slide Time: 16:05)
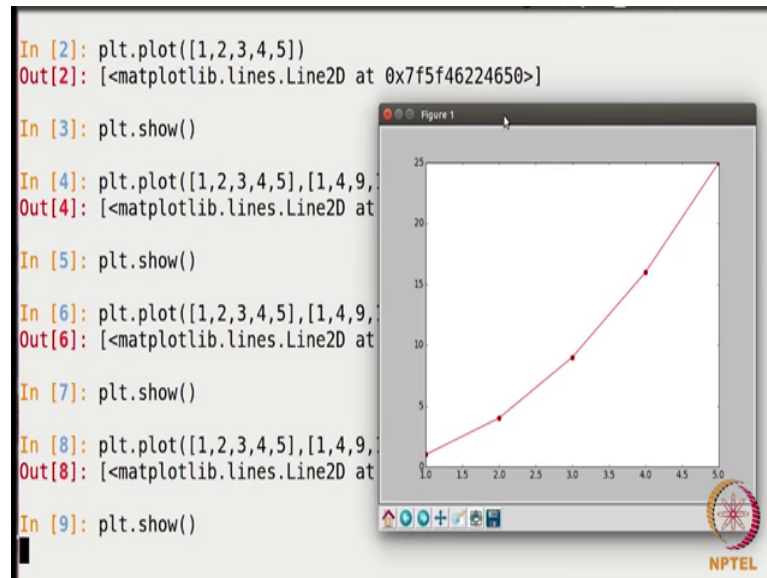
So, these are some of the versions; you can even change the other things, let me show you some more markers here so that you can play around with them.
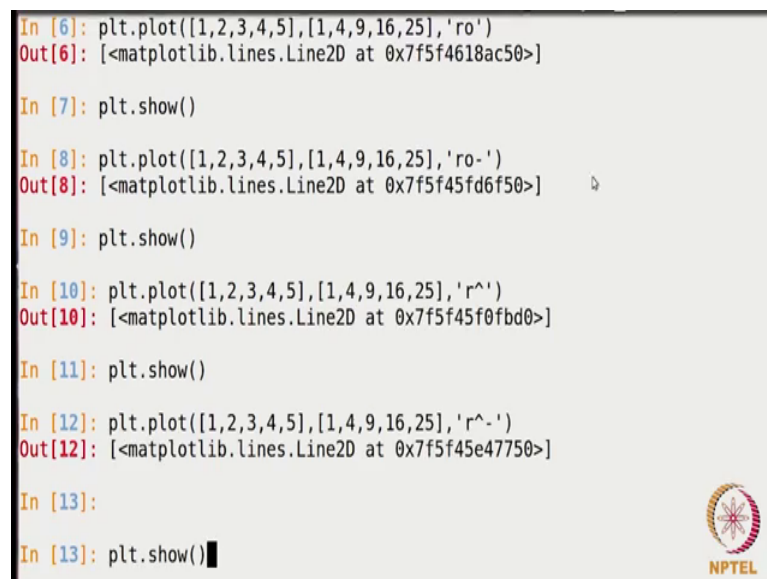
(Refer Slide Time: 16:23)

For example, if you want blue triangles; I will just write this symbol here and then I will write plt dot show.
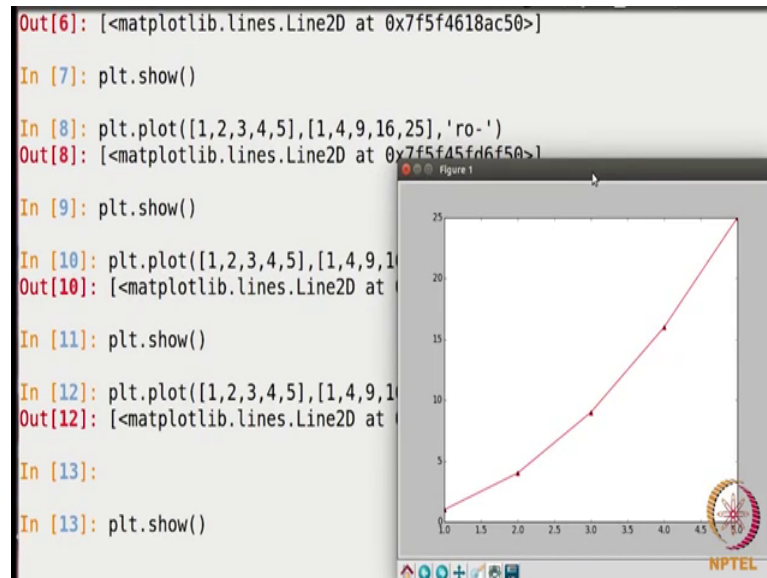
(Refer Slide Time: 16:29)

Now, you see you are getting the triangles here.
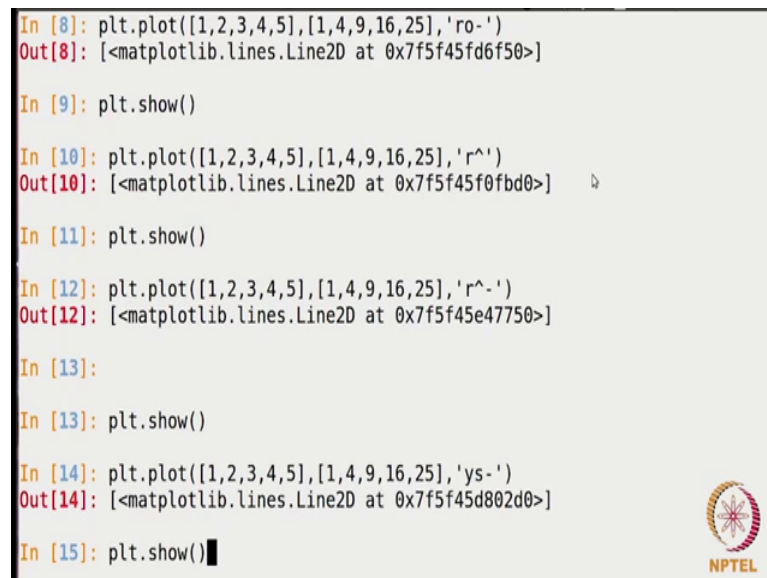
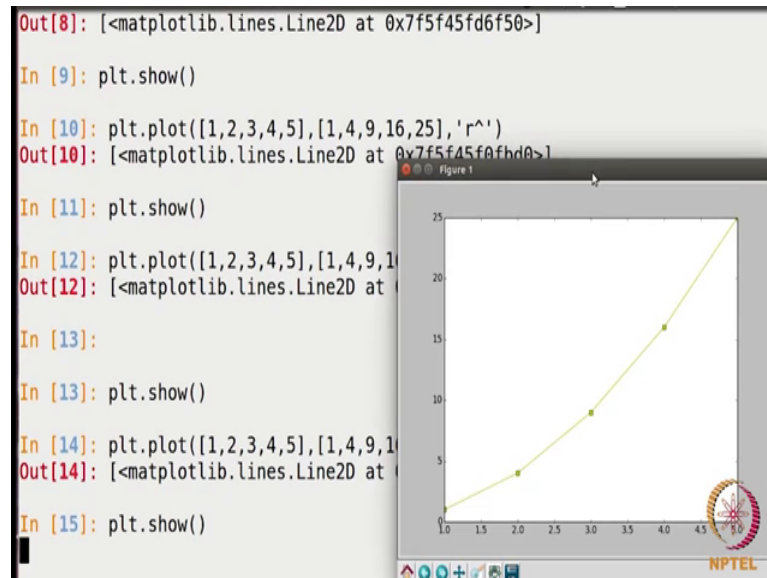(Refer Slide Time: 16:33)

(Refer Slide Time: 16:39)

And if you want triangles as well as lines you can again; you can do that, so you are getting both the things.

(Refer Slide Time: 16:42)

And if you want the squares, so here you can write s and say I want yellow squares and then I will write plt dot show.

(Refer Slide Time: 16:51)

So, we getting squares here and the line is also there.

(Refer Slide Time: 16:58)
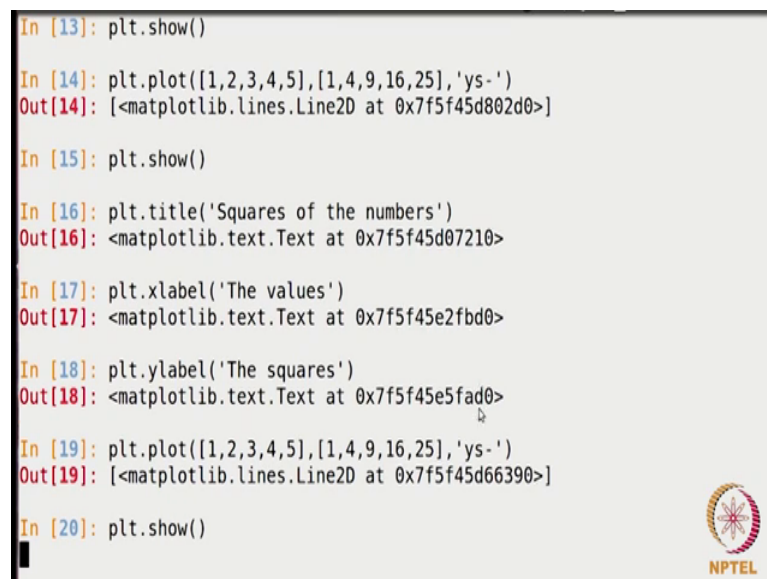
So, there you can basically look at the documentation and you will see there are various and you can just play around with them and use them as per your requirement. As you saw that, there was no title for the plot, there was no title on the x and y axis. So, let us try doing that; the function that we will be using for doing that, there is title function. So, we will write plt dot title and we can keep the title of the plot here. Let me give something like squares of the numbers and I also want to give title for the x axis. The

function that we will use for that is x label, so I will write plt dot x label; say the values. I also want to give a label for the y axis, so I will write plt dot l label and may be the squares.

Now, again I will call this plt dot plot and then I will call plt dot show.

(Refer Slide Time: 18:06)
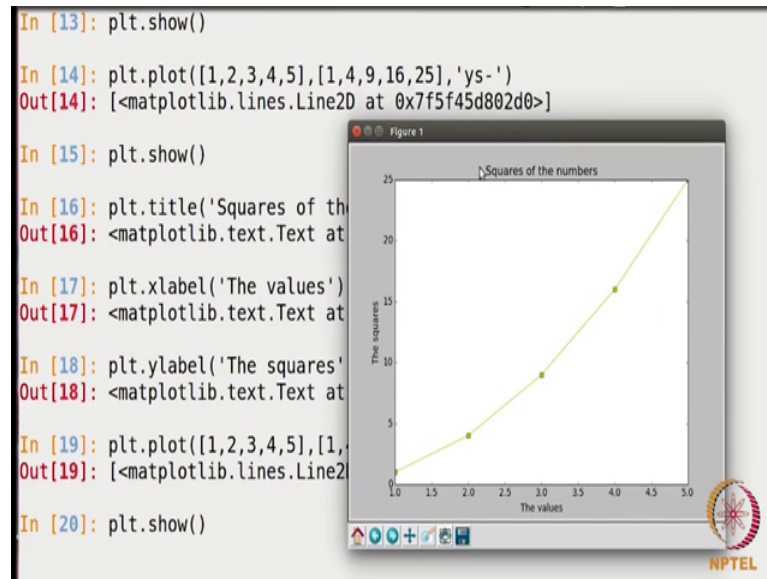
Now, you see we have got this; the title of the plot, we have the title of the x and y axis as well.

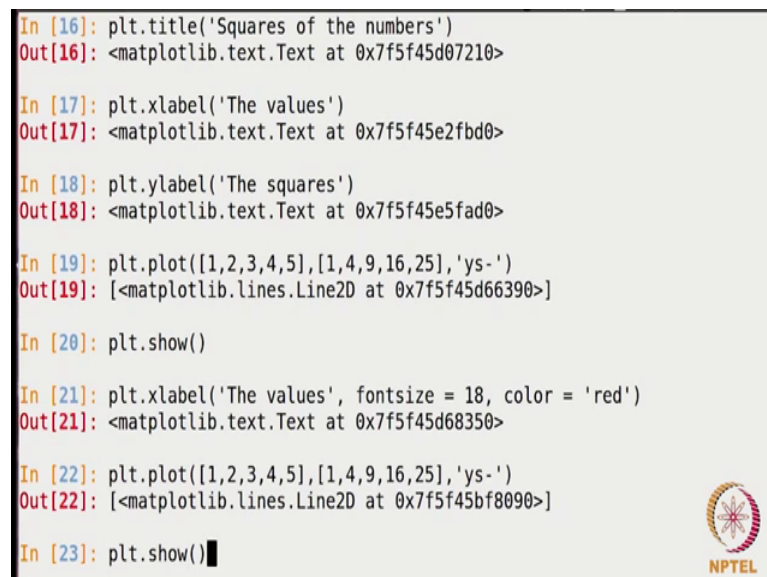(Refer Slide Time: 18:18)

Let me also show you that you can also change the colour and the font size of these labels as well, as per your requirement. So, let me give you one example here; I want to change the x label to be say red in colour. So, we already made use this function rate x label plt dot x label sorry, so we had only passed the label here. So, I am going to pass few more parameters here to customize it, so I am going to change the font size say 18 and I also want to change the colour; so I will write colour is equal to say red. Now label is changed, so I will call the plot command again and then I will show it.

(Refer Slide Time: 19:11)
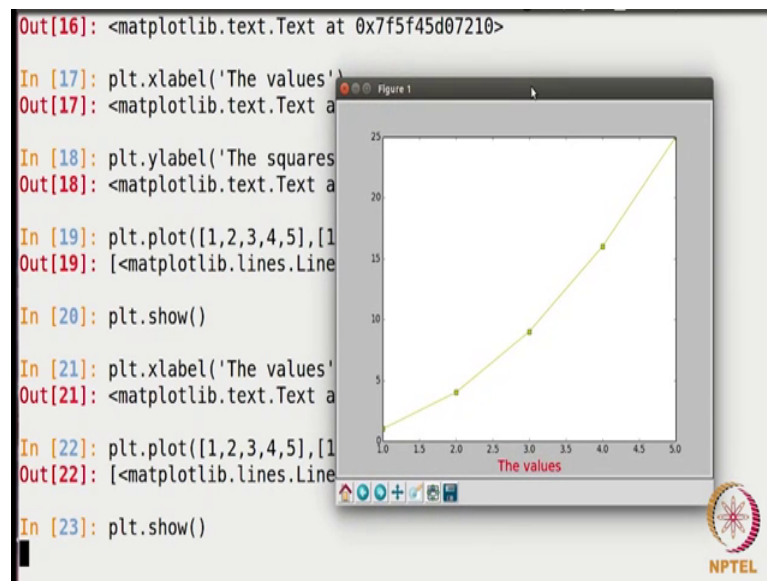
So, you see the x label has changed, the colour has changed, the font has changed; so you can always; you know keep it the way you want it. Another thing that I want you to notice here is that; the values on x axis is starting from 1 to 5 and the values on the y axis is starting from 0 to 25 and if you see the values that we had passed are; the x axis values are from 1 to 5 and y values are from 1 to 25. So, it has automatically chosen the values where to start from and where to end; where we can always change it, we can always customize it. So, let me show you how we can do that.

(Refer Slide Time: 19:52)

```
In [19]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[19]: [<matplotlib.lines.Line2D at 0x7f5f45d66390>]

In [20]: plt.show()

In [21]: plt.xlabel('The values', fontsize = 18, color = 'red')
Out[21]: <matplotlib.text.Text at 0x7f5f45d68350>

In [22]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[22]: [<matplotlib.lines.Line2D at 0x7f5f45bf8090>]

In [23]: plt.show()

In [24]: plt.axis([0,6,0,30])
Out[24]: [0, 6, 0, 30]

In [25]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[25]: [<matplotlib.lines.Line2D at 0x7f5f45ba2790>]

In [26]: plt.show()
```
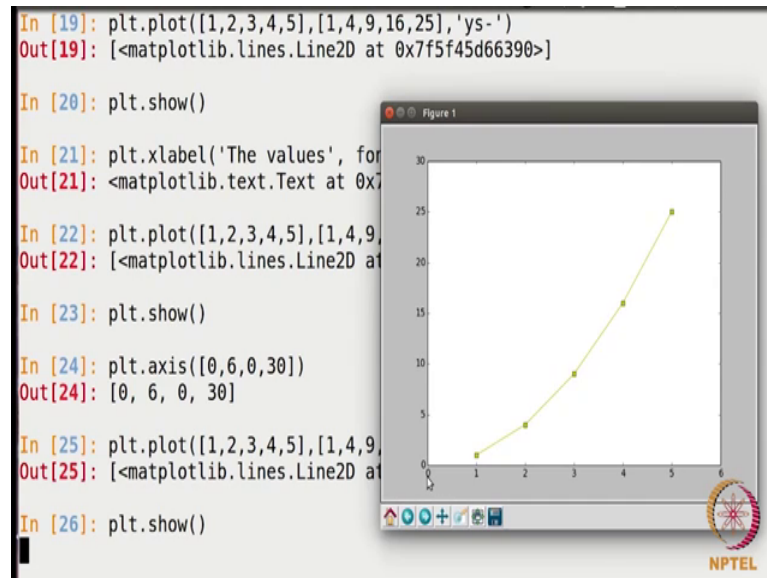
The function that we use to change the starting and ending of the axis is called plt dot axis itself. So, I will write plt dot axis; now this function accepts one parameter which is a list having four values, so the first value is the starting of x axis, the second value is the ending of x axis, the third value is the starting of y axis and the fourth value is the ending of y axis.

So, here you see our values; x values are starting from 1 to 5, so let me in order to show you its usage; I am going to give it 0, the starting of x axis and the ending of x axis; I am just like that going to give 6; just to show you how it works. And now I have to give the starting y axis, I am going to give 0 here and the ending of y axis; you see it is up to 25. So, let me just give it 30 so that we see how it works alright, so again I will call plt dot plot and then I will call plt dot show.

(Refer Slide Time: 20:56)


```
In [19]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[19]: [<matplotlib.lines.Line2D at 0x7f5f45d66390>]

In [20]: plt.show()

In [21]: plt.xlabel('The values', fo
Out[21]: <matplotlib.text.Text at 0x7

In [22]: plt.plot([1,2,3,4,5],[1,4,9,
Out[22]: [<matplotlib.lines.Line2D a

In [23]: plt.show()

In [24]: plt.axis([0,6,0,30])
Out[24]: [0, 6, 0, 30]

In [25]: plt.plot([1,2,3,4,5],[1,4,9,
Out[25]: [<matplotlib.lines.Line2D a

In [26]: plt.show()
```

Now, you see here; x values are now starting 0 to 6 and y values are now starting from 0 to; are now going from 0 to 30 alright. So, you can this is always in control you can change it the way you want it.
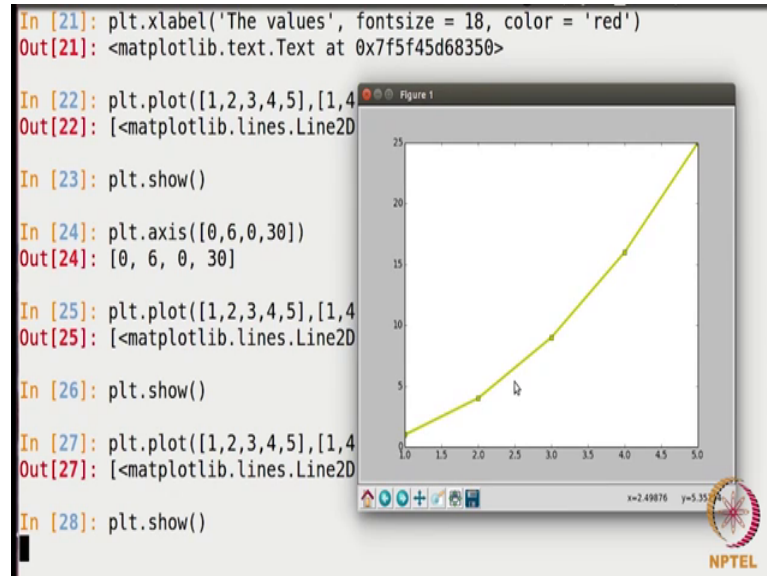
(Refer Slide Time: 21:15)


```
In [21]: plt.xlabel('The values', fontsize = 18, color = 'red')
Out[21]: <matplotlib.text.Text at 0x7f5f45d68350>

In [22]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[22]: [<matplotlib.lines.Line2D at 0x7f5f45bf8090>]

In [23]: plt.show()

In [24]: plt.axis([0,6,0,30])
Out[24]: [0, 6, 0, 30]

In [25]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-')
Out[25]: [<matplotlib.lines.Line2D at 0x7f5f45ba2790>]

In [26]: plt.show()

In [27]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-', linewidth = 3.0)
Out[27]: [<matplotlib.lines.Line2D at 0x7f5f45ad5910>]

In [28]: plt.show()
```
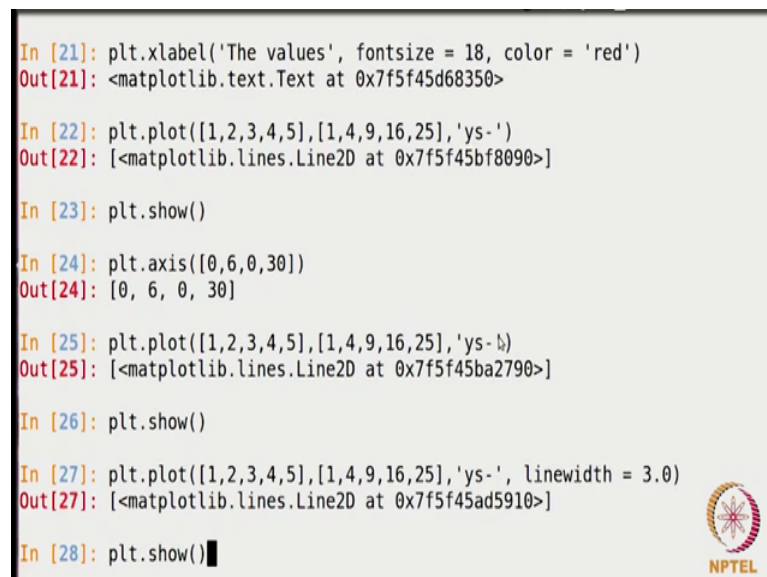
Let me now show you couple of more features of this plot function; for example, you got a very thin line as a plot. If you want to increase the thickness of this line you can do that, let me show you how to do that. So, plt dot plot; let me reuse the same function, so here we were passing some three parameters, I am going to add one more parameter here
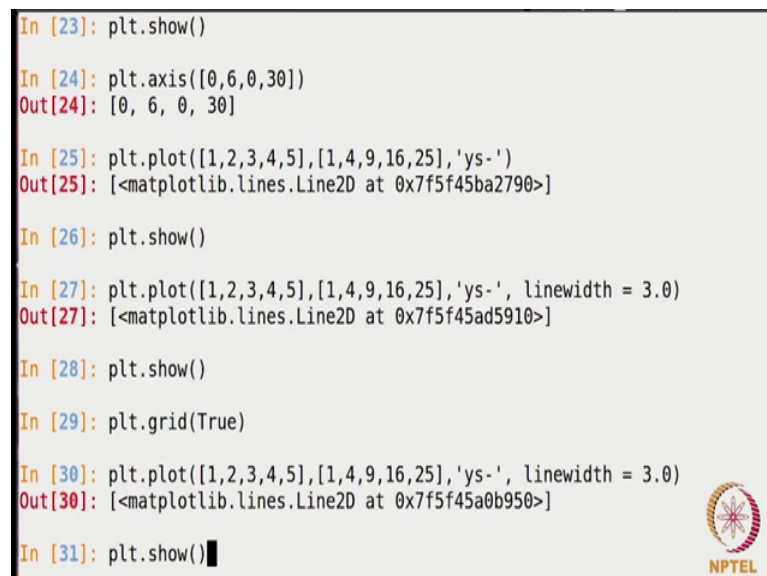
for the thickness of the line. So, the keyword is line width here, so line width is equal to say 3.0; by default it is 1.0, so I give this line width and then I will call plt dot show.

(Refer Slide Time: 21:48)

Now, you see the thickness of this line has increased, so you can always change that it is all about decorating your plot, so that it is better visualizing.
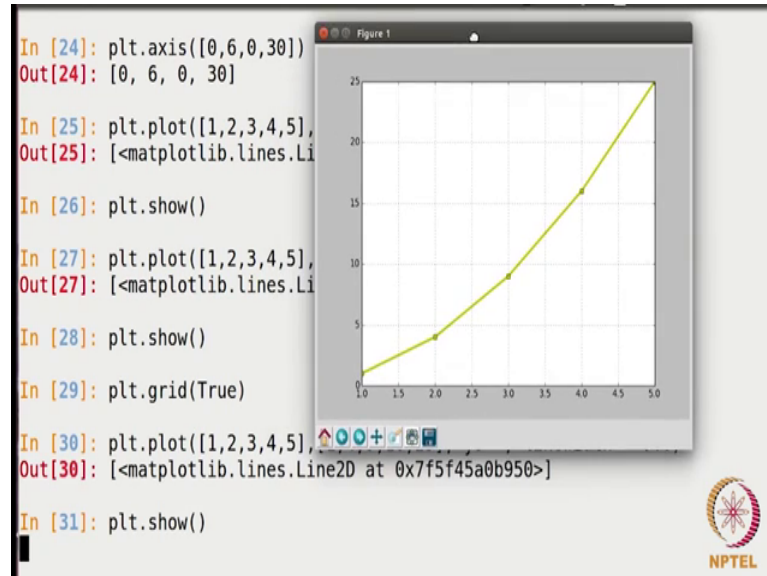
(Refer Slide Time: 22:02)

Another thing that is sometimes needed is a grid that you want to display in your plot sometimes. So, by default it is turned off, but you can always turn it on by passing a true parameter there, so the function that we use there is grid. So, I will write plt dot grid and

I will pass true here, which is by default false. So, once I do that and I call this plot function and then I will call the show function.

(Refer Slide Time: 22:30)


```
In [24]: plt.axis([0,6,0,30])
Out[24]: [0, 6, 0, 30]

In [25]: plt.plot([1,2,3,4,5],
Out[25]: [<matplotlib.lines.Li

In [26]: plt.show()

In [27]: plt.plot([1,2,3,4,5],
Out[27]: [<matplotlib.lines.Li

In [28]: plt.show()

In [29]: plt.grid(True)

In [30]: plt.plot([1,2,3,4,5],
Out[30]: [<matplotlib.lines.Line2D at 0x7f5f45a0b950>]

In [31]: plt.show()
```

You see there is a grid here, so that is there in the background you can always turn on and off as per your requirement. You can also save this plot if you want; there are various functions, there are various features here that you can make use of ok.

(Refer Slide Time: 22:51)


```
In [28]: plt.show()

In [29]: plt.grid(True)

In [30]: plt.plot([1,2,3,4,5],[1,4,9,16,25],'ys-', linewidth = 3.0)
Out[30]: [<matplotlib.lines.Line2D at 0x7f5f45a0b950>]

In [31]: plt.show()

In [32]: x = [i for i in range(20)]

In [33]: plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-33-724b24d2d8a2> in <module>()
----> 1 plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'

In [34]:
```

Let us get back here, as of now we were showing only one plot inside the plotting region. But that is not necessary; we can always display multiple plots inside the same plotting

region. Let me show you an example for that, let me create the x values by using a loop for example, that will also show you how you can use a loop to create a list, if you do not know already.

So, I right i for i in range; 21, if I write this; I will get the values from 0 to 20; excluding 21. So, basically I will be getting 21 values and let me keep 20 here, so I got the x values and then I want to plot this x along with its squares. So, I will write x and this is exponential operator and I want it red in colours, maybe in line. Now in case I want to also display the x cube as well in the same plotting reason, I can do that. So, let me show you how to do that, so I will write x cube here and say I want that blue in colour, blue lines and assume I want to display x to the 4 as well in the same plotting region. So, I will write x to the 4 here and say I want that yellow in colour; I will do this.

But there is an important point to be noted here, which I want you to know. When I run this command, it should show me an error; let me see here, it is showing an error. Now what is the reason for that? The reason is that x is a list, this is a list and then we are applying this exponentiation operator on a list, which is not allowed in python. So, what we can do here is that; we can make use of a package which is called numpy, which allows such sort of operation. So, if we if we convert this list x into a numpy array; then in that case such operations are allowed. So, the crux is that this kind of operation is not allowed on lists, but it is allowed on numpy arrays. So, let us do one thing we will make use of this numpy package.

(Refer Slide Time: 25:10)

Let me put that; now I will basically convert this x into a numpy array. So, I will write numpy dot array, this is a function, so I am passing x as a list and I am getting it as a numpy array. Now after that, if I called this plot; it is now giving me an error and it is working fine, now we can do p l t dot; sorry dot show.

(Refer Slide Time: 25:41)

So, here you see I am getting all the 3 plots here; yellow, blue and red; I am getting all the plots here. Now since this due to these dimensions of this y axis, you are not really

able see this red line, but you can always do that as I told you, you can customize, you can change the starting and ending of y axis; so you can play around with that alright.

(Refer Slide Time: 26:05)


```
In [35]: x = numpy.array(x)

In [36]: plt.plot(x, x**2, 'r-', x, x**3, 'b-', x, x**4, 'y-')
Out[36]:
[<matplotlib.lines.Line2D at 0x7f5f45937090>,
 <matplotlib.lines.Line2D at 0x7f5f45937310>,
 <matplotlib.lines.Line2D at 0x7f5f459379d0>]

In [37]: plt.show()

In [38]: import random

In [39]: x = [random.randint(1,100) for i in range(100)]

In [40]: y = [random.randint(1,100) for i in range(100)]

In [41]: plt.scatter(x,y)
Out[41]: <matplotlib.collections.PathCollection at 0x7f5f4584fc50>

In [42]: plt.show()
```

As of now we have seen only a simple plot function, we have seen a simple plot in form of lines or points, but let me tell you that we can also have a different kinds of plots using this matplotlib package; for example, we can have scatter plot, we can have bared plot, we can have heat max as well. So, there are various kinds of plots that we can draw; now I will be showing you some two, three examples just to give an idea.

I will show you how to draw scatter plot, so in order to draw the plot let me create a list randomly also that will show you how you can make use of a random function. Before that I need to import the package, so I will import random package and then I am going to create a list which will be having a 100 random values. So, how do we do that? I will make use of the function randint from this package random and I want the values to be between 1 and 100 and I want 100 such value, so I will write for i in range 100.

So, I will getting 0 to 99, I will be getting 100 values here alright, so let me also have y values randomly. So, basically I am having a 100 random values as x values and 100 random values as y values. Now, let me draw the scattered plot; the function is scatter itself. So, I will plt dot scattered x, y and in order to see the plot, we have to write p l t dot show.

(Refer Slide Time: 27:49)
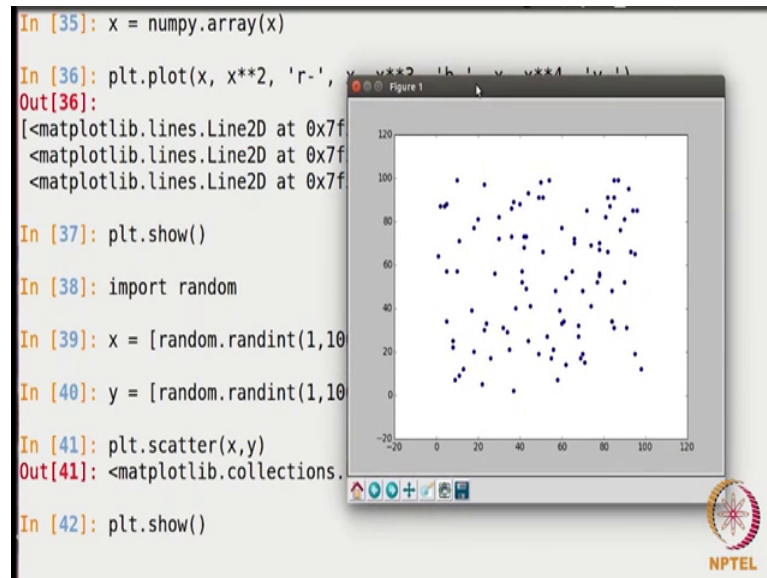
So, here we get the plot; now again the kinds of operation that we applied in the simple plot function, like we changed the axis, we changed the colour, we changed the titles and everything that also we can do here. For example, here you see by default; the x axis starting from minus 20 and same goes with the y axis which is not looking so good.
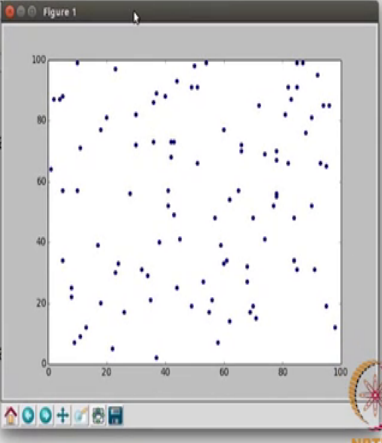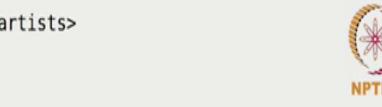
(Refer Slide Time: 28:16)

Let me change that, so what I will do is plt dot axis, I am going to pass a list with four values; starting of x axis to be 0 and ending to be 100, starting of y to be 0 and ending to

be 100. Now, I will call this plot function again, sorry scattered plt dot scatter and then plt dot show.

(Refer Slide Time: 28:41)

Now, you see x and y are starting from 0 now and it is looking quite better, you can clear on, we can change the colours of these dots alright, let us get back here.
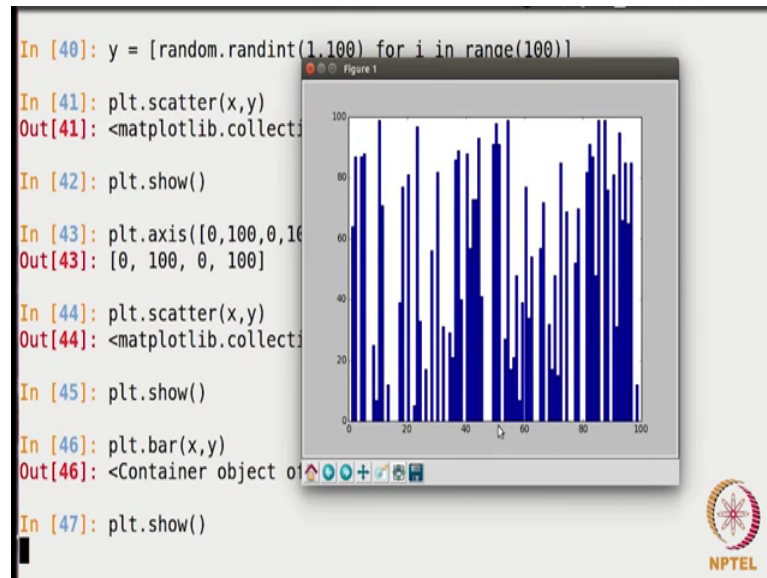
(Refer Slide Time: 28:52)

Now, let me show you another kind of plot that you can have using this package. If you want to bar chart or a few data, so you can use a function bar; plt dot bar, I am going use the same x and y values here again, so that is all; plt dot show.

(Refer Slide Time: 29:12)

We get a bar chart here, so by default we always get a blue colour, so axis is fine here 0 to 100 because we had set it already and so you can again clear on with customizing this plot as per your requirements. So, these were just few examples of the kind of flexibility that this package provides you. You can look into the documentation and look for more functions in as per your data as per your requirement.

Thank you.