NPTEL

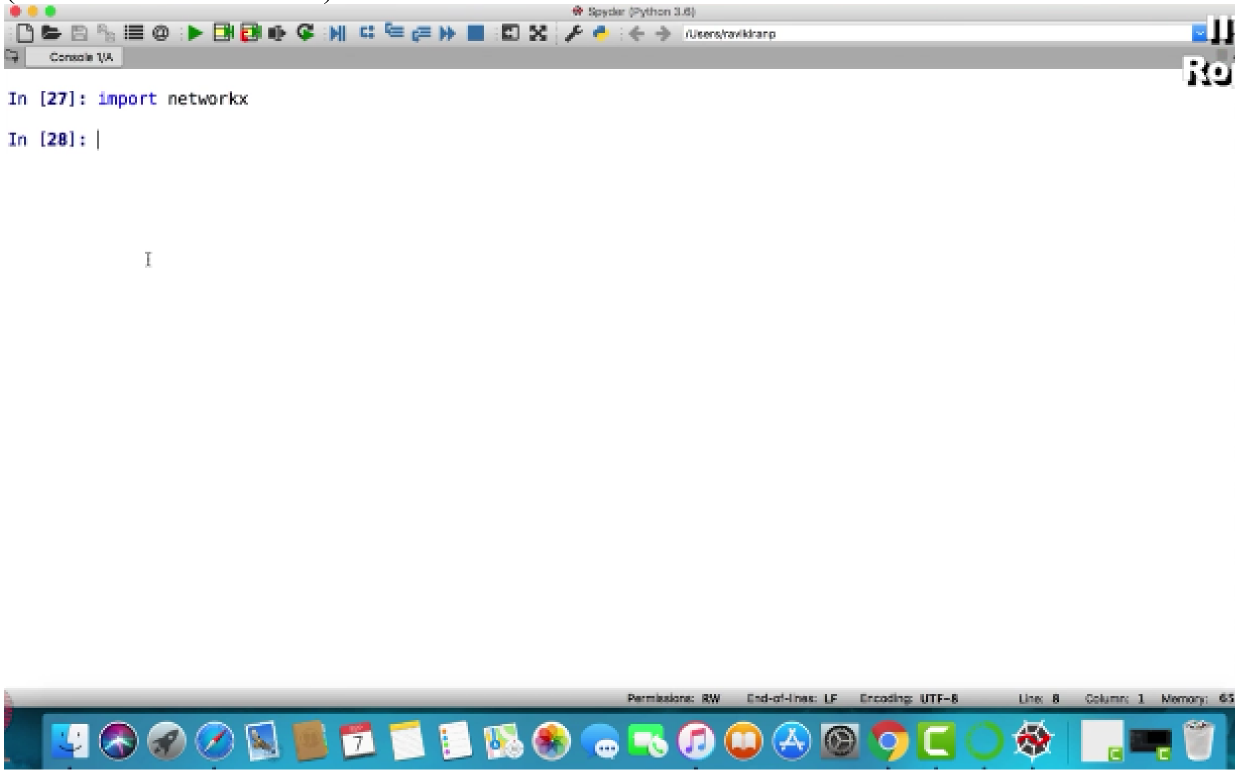
NPTEL ONLINE CERTIFICATION COURSE

Discrete Mathematics
Graph Theory - 1

Introduction to NetworkX

By
Prof. S.R.S Iyengar
Department of Computer Science
IIT Ropar

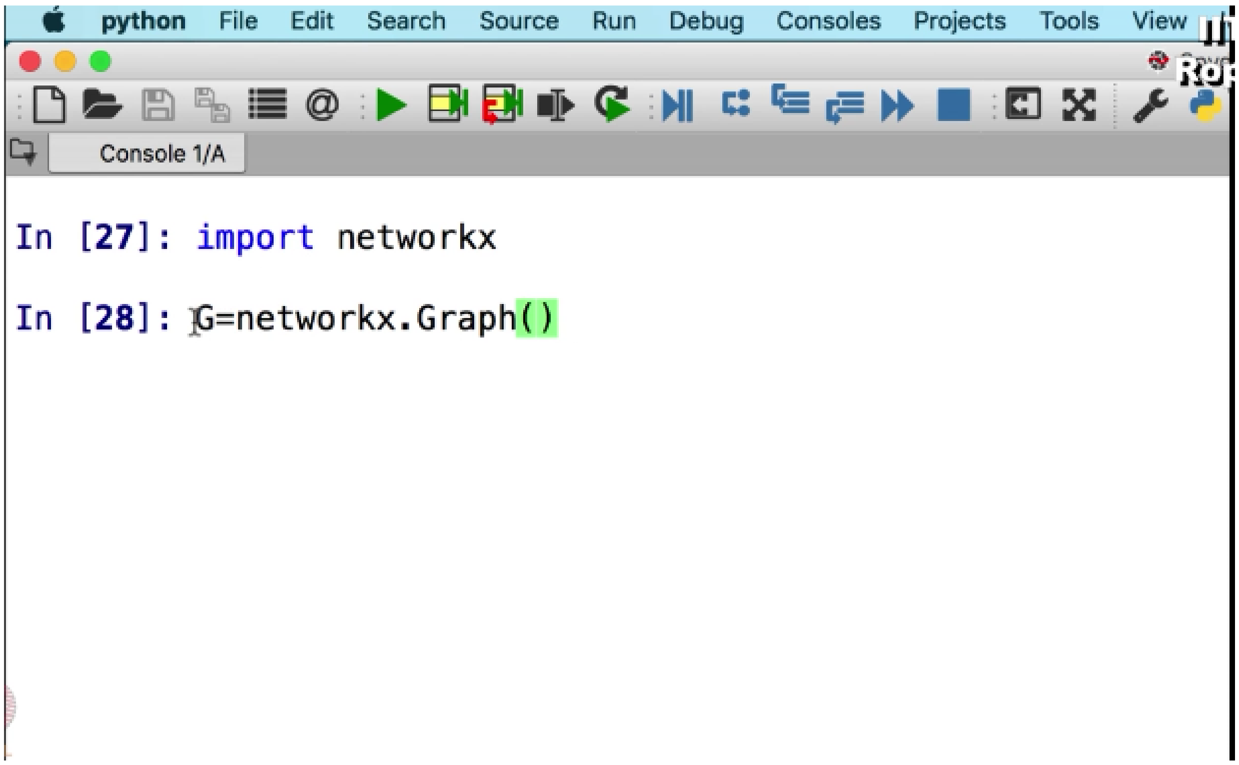
So in the previous video I had told you that we'll be using a new package for graph theory it is called as NetworkX, so what I'm going to do is I'll be importing that, so I will give import NetworkX,
(Refer Slide Time: 00:21)



```
In [27]: import networkx

In [28]:
```

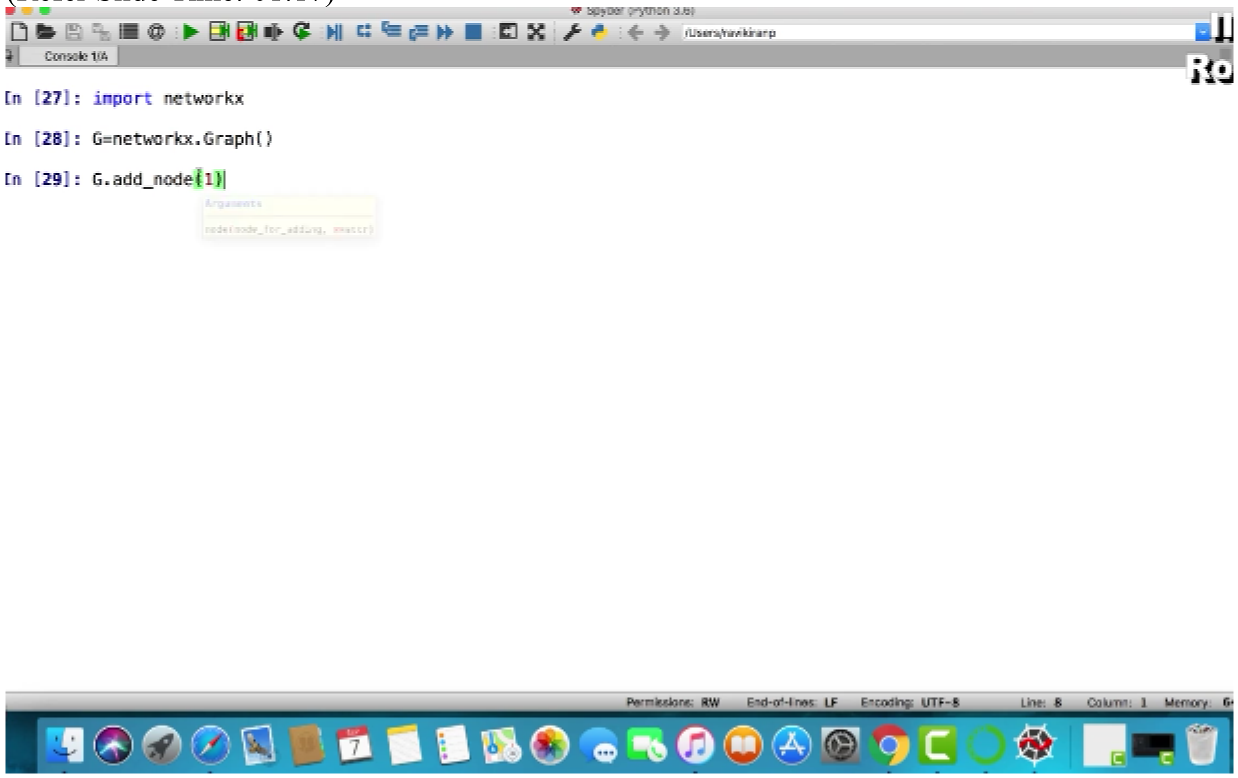
so now this package NetworkX has got loaded.

Now what I'm going to do is we'll be first aiming at creating a graph G, so what I do is G = networkx.graph and two brackets like this,
(Refer Slide Time: 00:44)



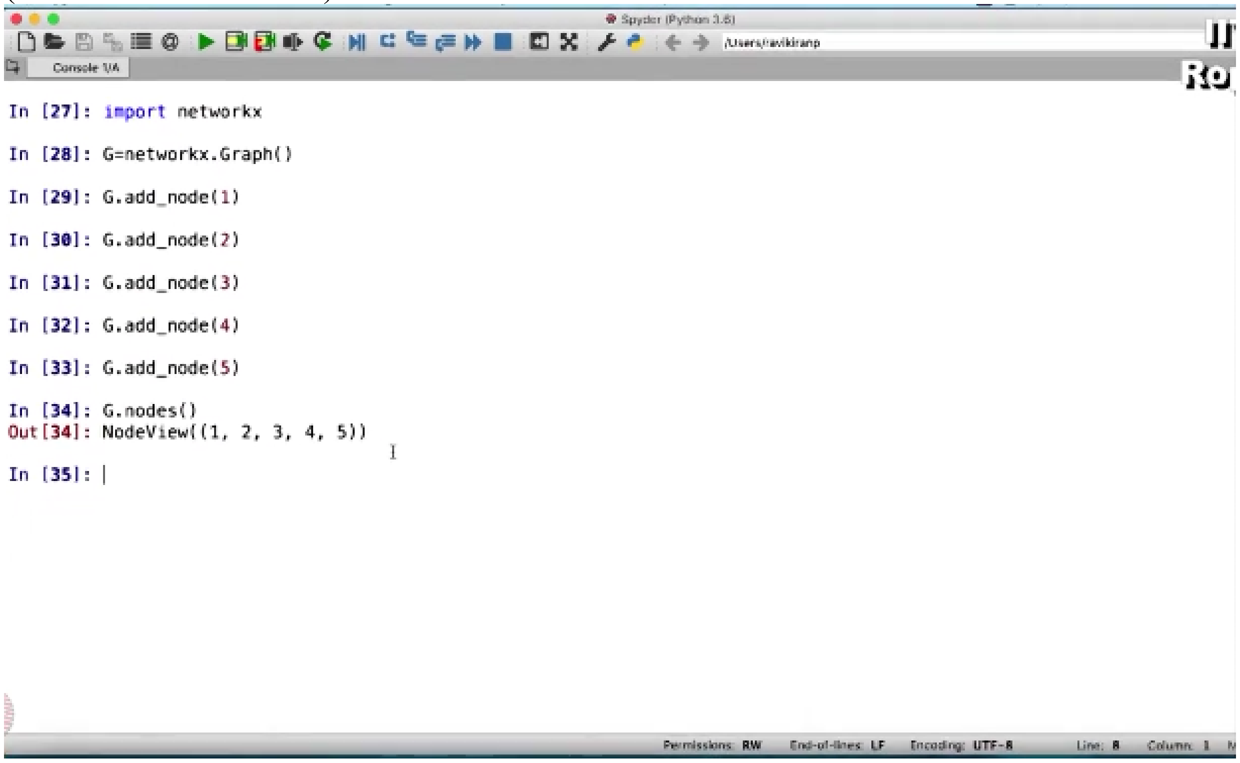
```
In [27]: import networkx

In [28]: G=networkx.Graph()
```

so this says that the graph G has got created. Now once the graph G has got created it does not imply that there are nodes and edges, we have to add them one by one, so what I'm going to do now is I'll add at the nodes, so G.add_node, please note all these commands are, we’ll be using all these commands very frequently from now onwards, G._add node, and then bracket I say1,
(Refer Slide Time: 01:17)



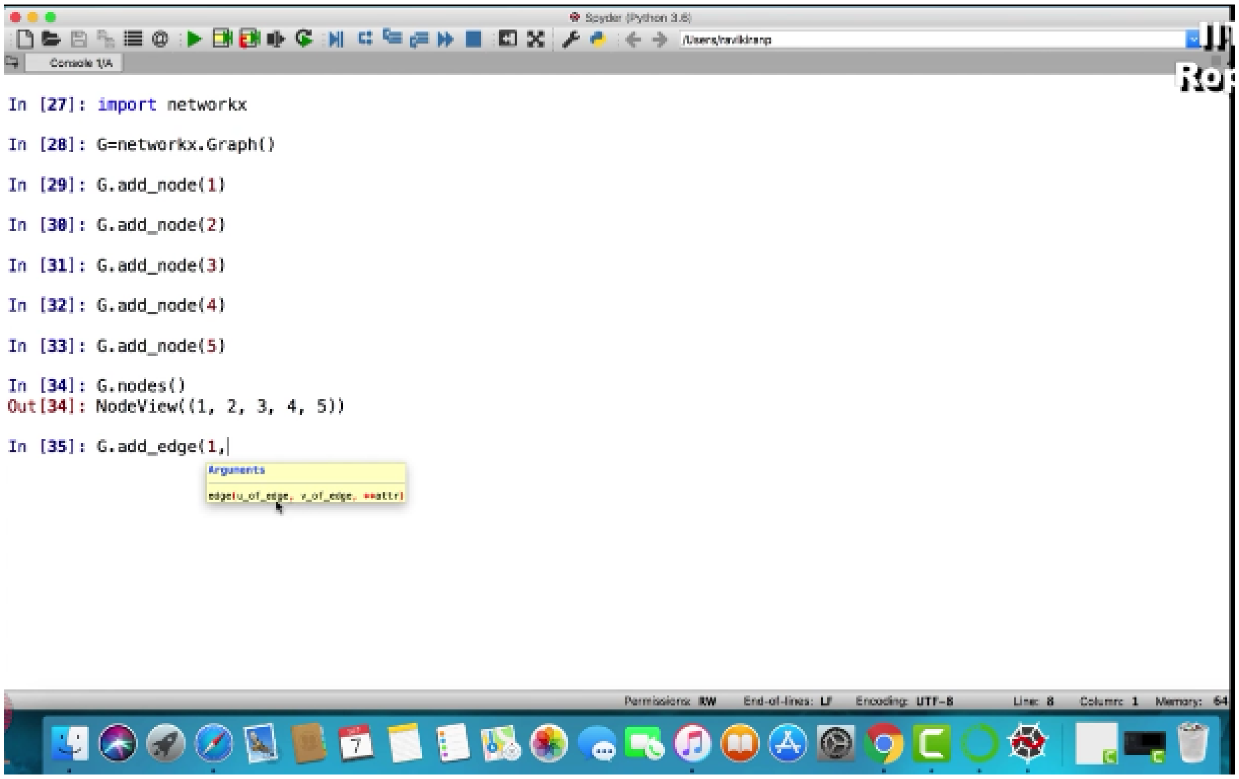
```
In [27]: import networkx

In [28]: G=networkx.Graph()

In [29]: G.add_node(1)
```

this means that node 1 has got created, I go ahead, now 1 has got created I will do the same for the rest G.add_node say 2, I've added node 2, next G.add node say 3, I've added 3 nodes now, let me do a few more G.add node say 4, and then G.add node say 5, yes, so I've added 5 nodes here, now if you want to see the list of all the nodes which are added what I'm going to do is G. nodes and then just these brackets will give you this one node view 1, 2, 3, 4, 5 these are the nodes we have added now.
(Refer Slide Time: 02:18)

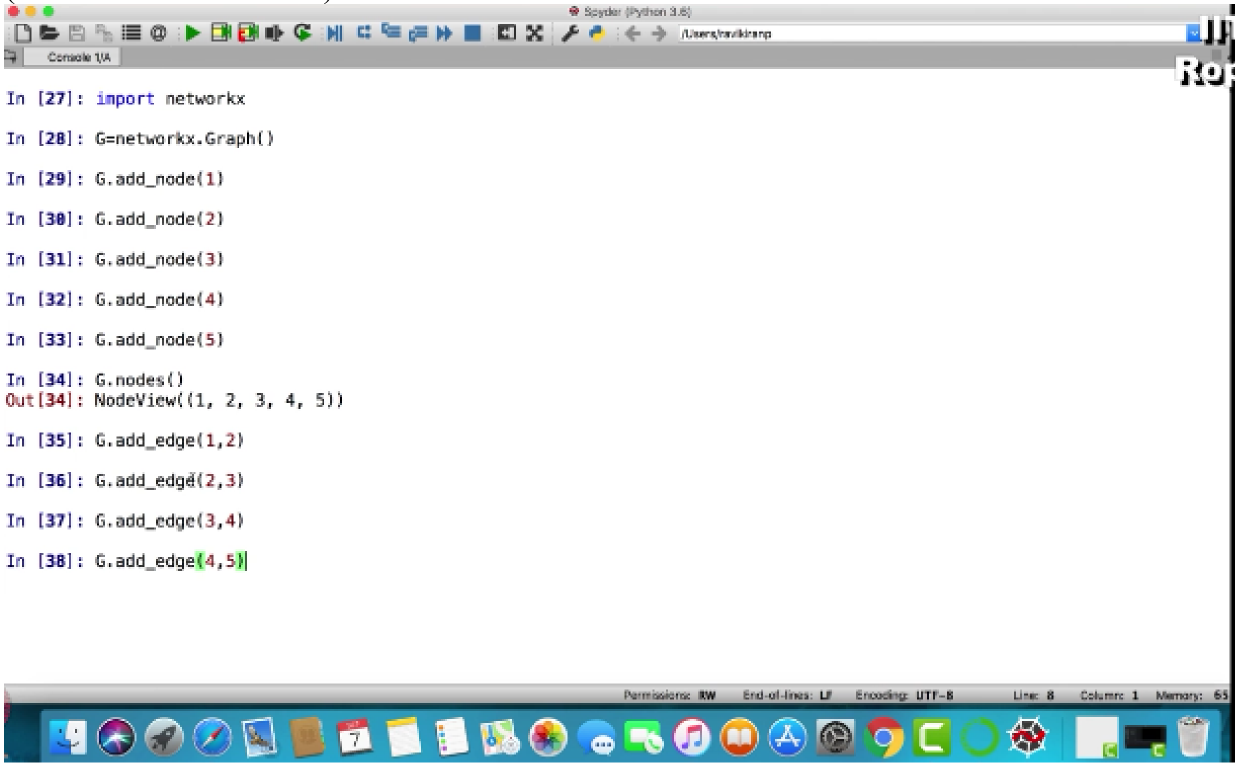


```
In [27]: import networkx

In [28]: G=networkx.Graph()

In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: |
```

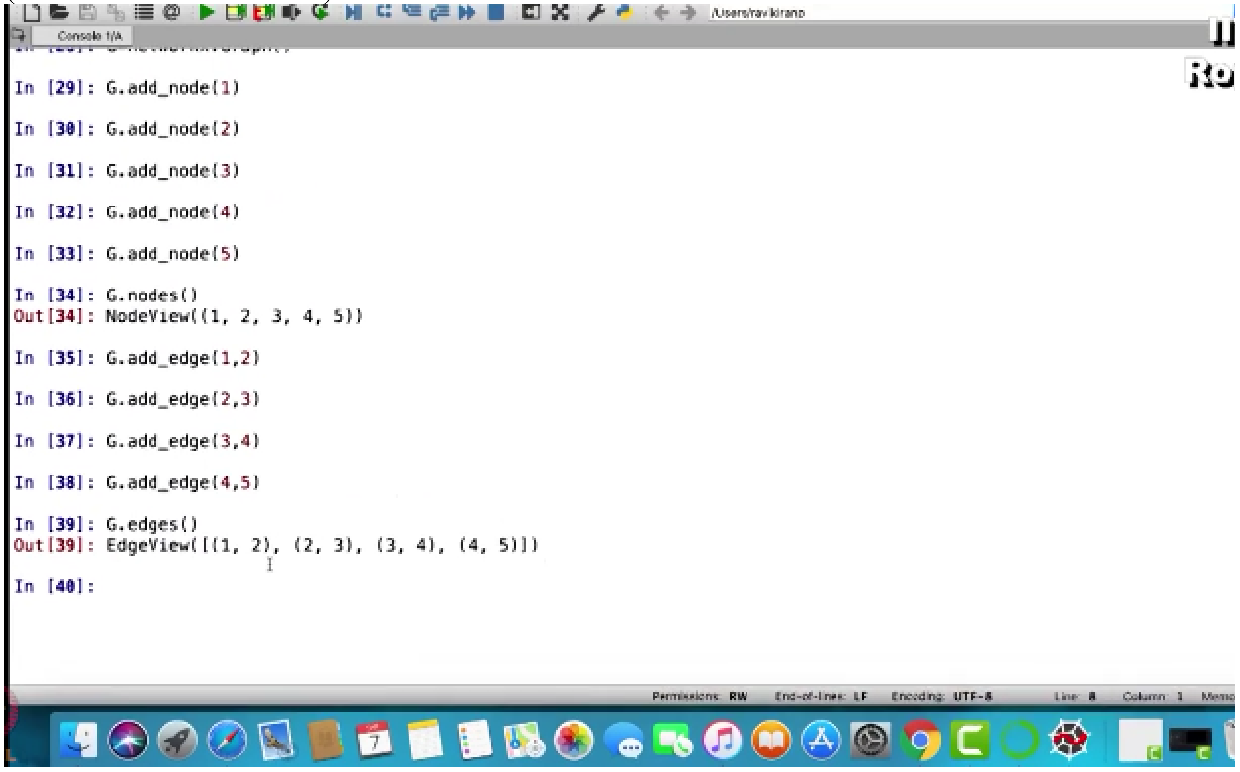
Now we have added nodes now what remains is to add edges, so what we are going to do the command for that, yes you must be guessing it rightly G.add_edge, this adds edge between the vertices which you have given already, and which you are going to select now within this bracket, you see here U of edge, V of edge, do you see this?
(Refer Slide Time: 02:44)


```
In [27]: import networkx

In [28]: G=networkx.Graph()

In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,
```

So it says that you have to give vertices, you have to mention vertices in this bracket, now like this so what I'm going to do now? Enter yes, so this has added the edge 1, 2 so between 1 and 2 these vertices and edge has got added just like how we added nodes we will be adding vertices now, so G.add_edge let me add one more 2, 3, G.add_ edge 3, 4, G.add_edge 4, 5.
(Refer Slide Time: 03:27)

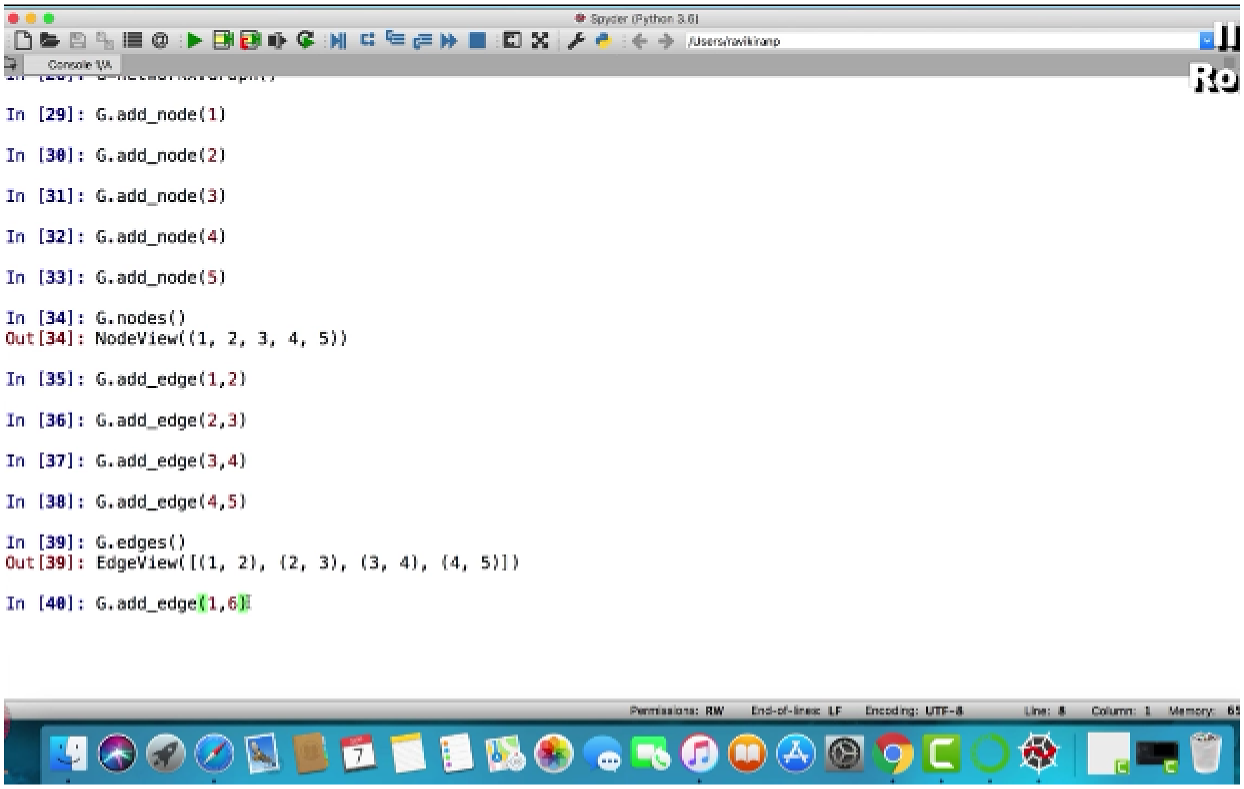

```
In [27]: import networkx

In [28]: G=networkx.Graph()

In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,2)

In [36]: G.add_edge(2,3)

In [37]: G.add_edge(3,4)

In [38]: G.add_edge(4,5)
```

Now I have added say 4 edges here, just like how we displayed the nodes we can display the edges too, so it will be G.edges and same like that, yeah, do you see something? Edge view (1, 2) (2, 3) (3, 4) and (4, 5) it displays all the edges that I have added till now.
(Refer Slide Time: 03:50)



```
In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,2)

In [36]: G.add_edge(2,3)

In [37]: G.add_edge(3,4)

In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]:
```

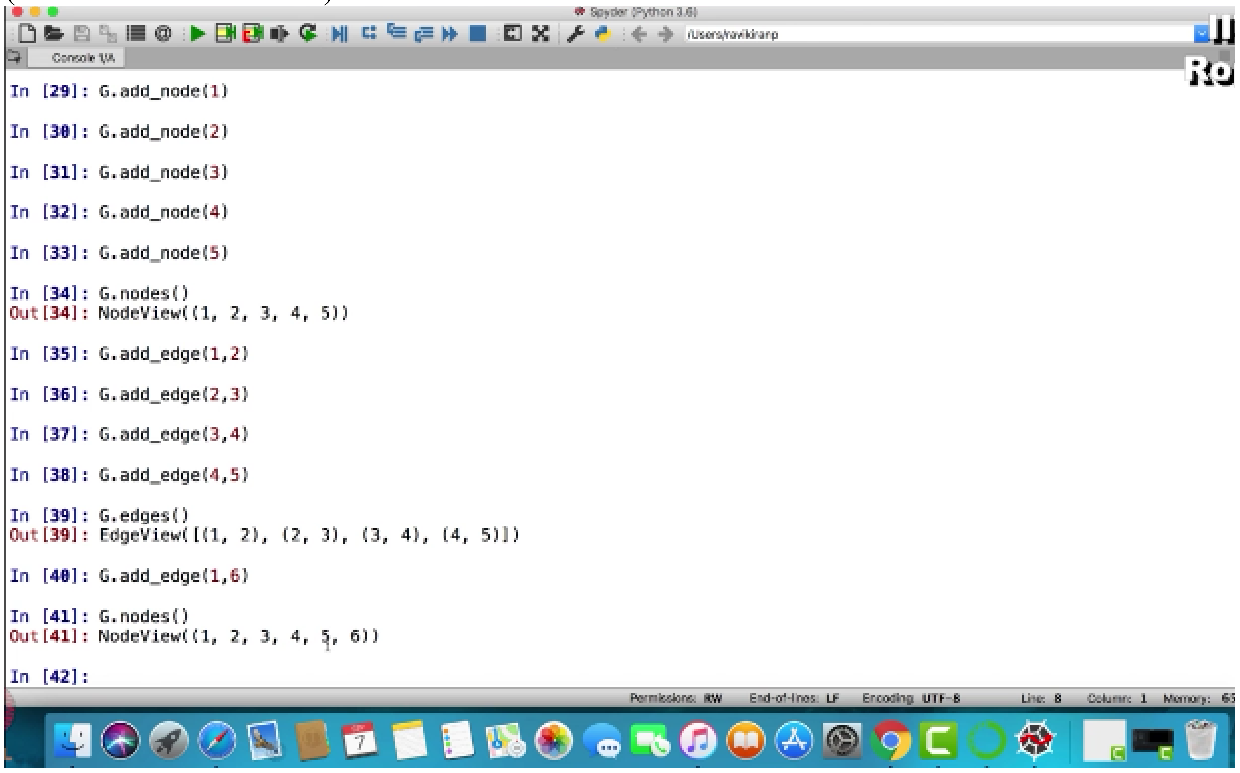
Now let me show you something like this G.add_edge, see once you have completed adding your edge it does not mean you can add, you cannot add any more edges you can always do that, but what is the best part here is if I give something like G.add_ edge say 1, 6, do you see that vertex 6 or node 6, I have not added here, G.add_node 6 is not there, but still when I give this command G.add_edge 1, 6
(Refer Slide Time: 04:29)



```
In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,2)

In [36]: G.add_edge(2,3)

In [37]: G.add_edge(3,4)

In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)
```

the note gets automatically created which one? 6, the node 6 gets automatically created and then creates this edge 1, 6.

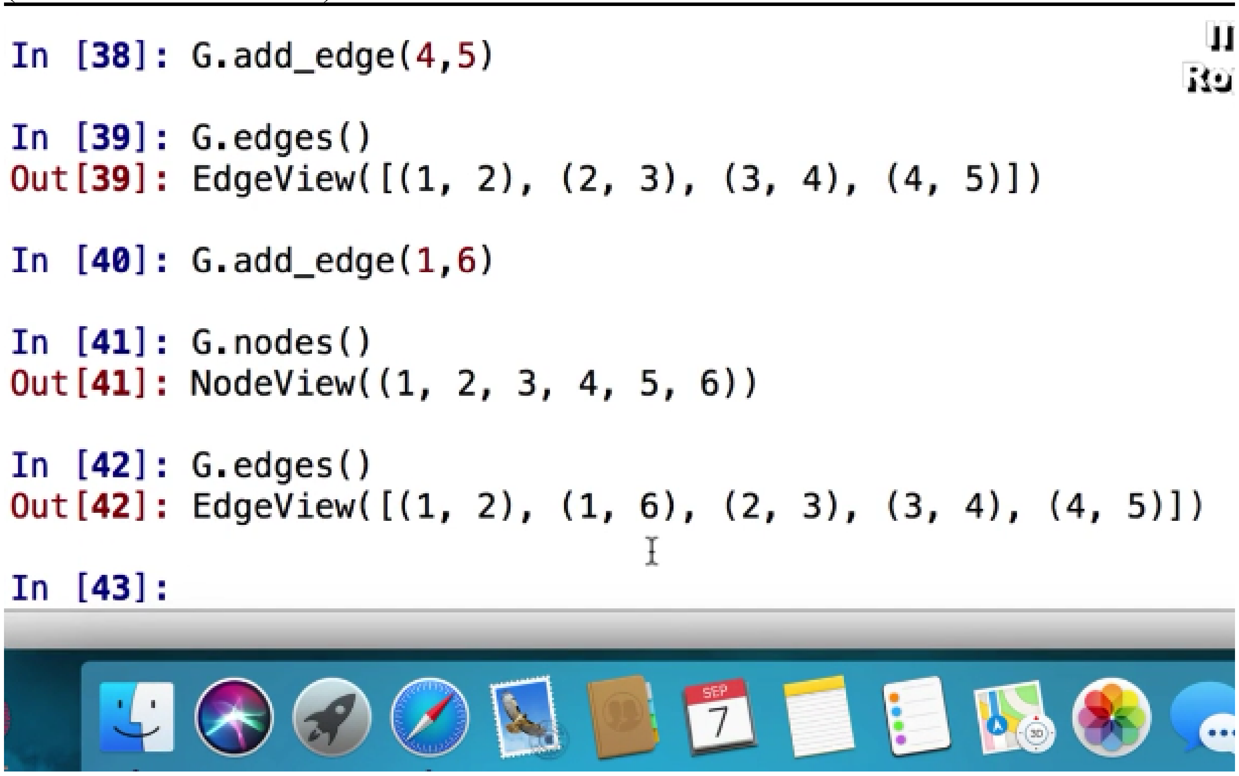
So let us do that, now let us just check G.nodes, see do you see 6 here
(Refer Slide Time: 04:51)



```
In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,2)

In [36]: G.add_edge(2,3)

In [37]: G.add_edge(3,4)

In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)

In [41]: G.nodes()
Out[41]: NodeView((1, 2, 3, 4, 5, 6))

In [42]:
```

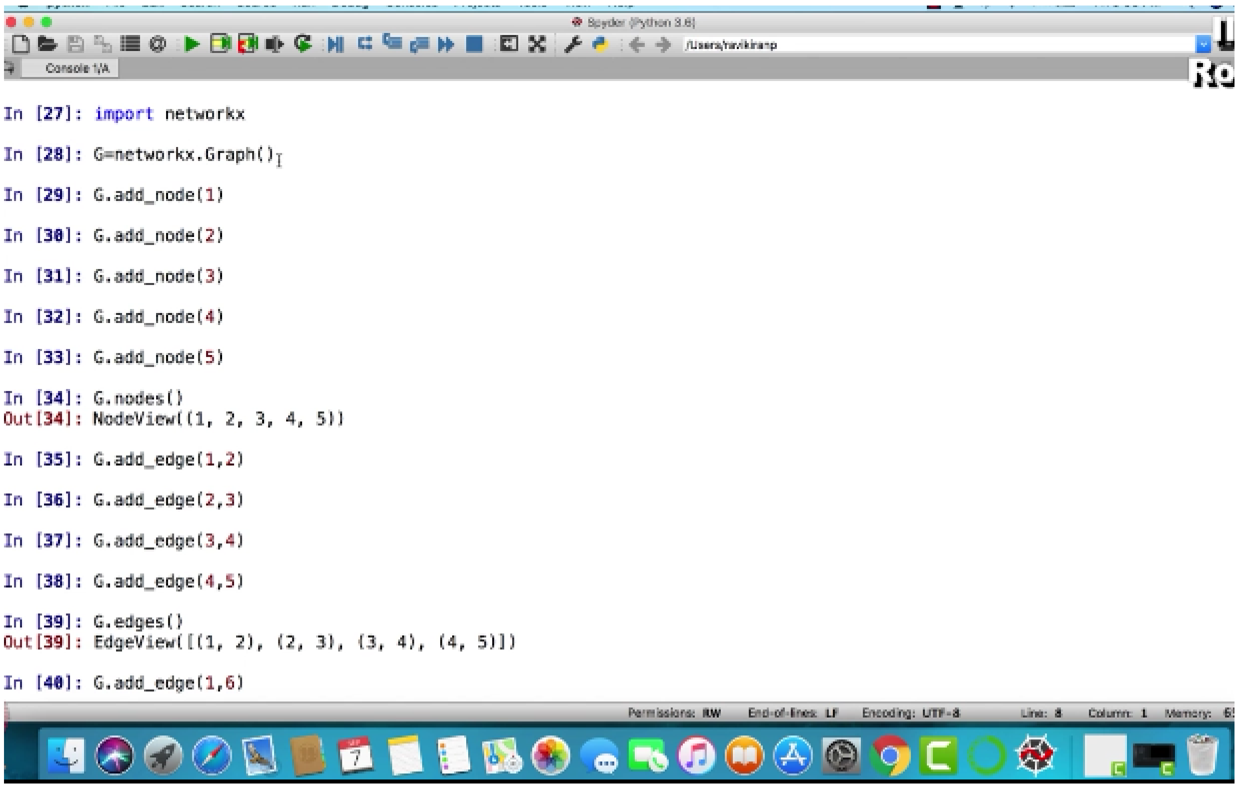
so the nodes are 1, 2, 3, 4, 5, 6, the node 6 has just now got created.

Now let us check the edges do you see 6 has got added here,
(Refer Slide Time: 05:03)



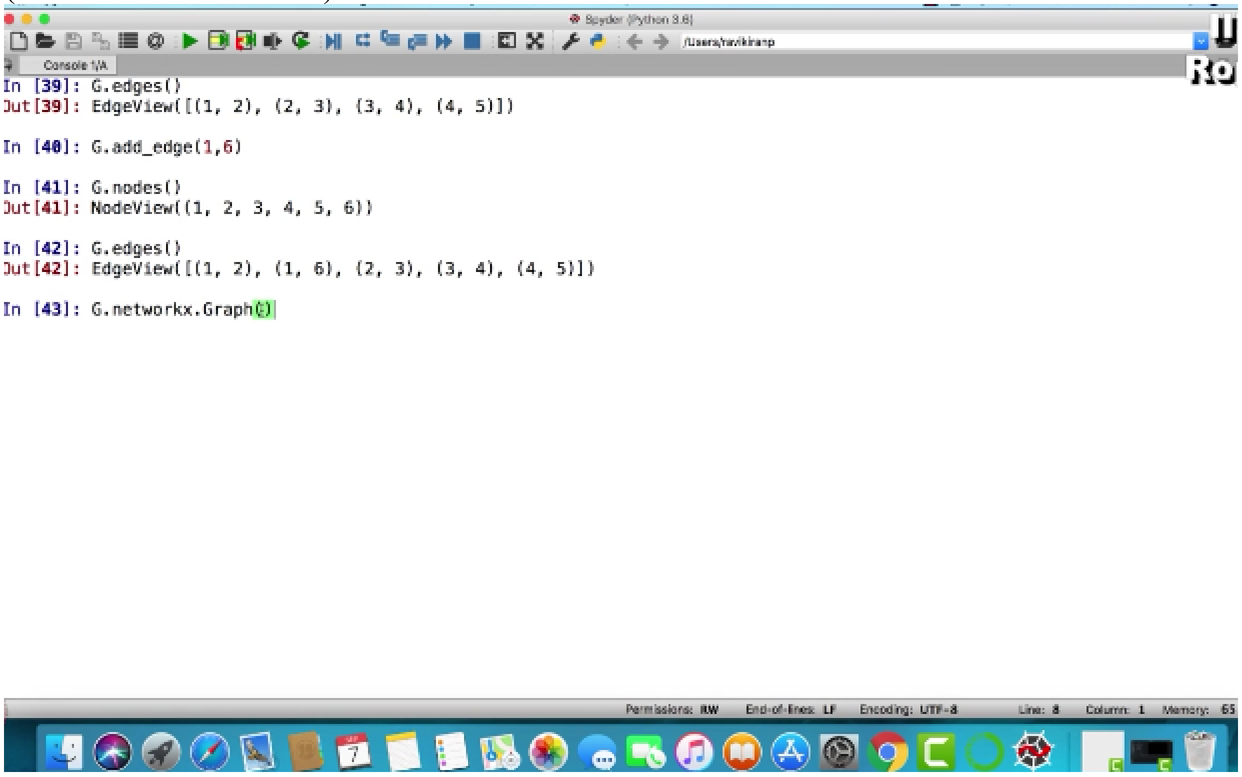
```
In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)

In [41]: G.nodes()
Out[41]: NodeView((1, 2, 3, 4, 5, 6))

In [42]: G.edges()
Out[42]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [43]:
```

so (1, 2) (1, 6) (2, 3) (3, 4) and (4, 5) so these are the edges in our graph, so now every time I have to use networkx.graph if you remember this was the initial command which how we started, do you see this input line 28
(Refer Slide Time: 05:28)



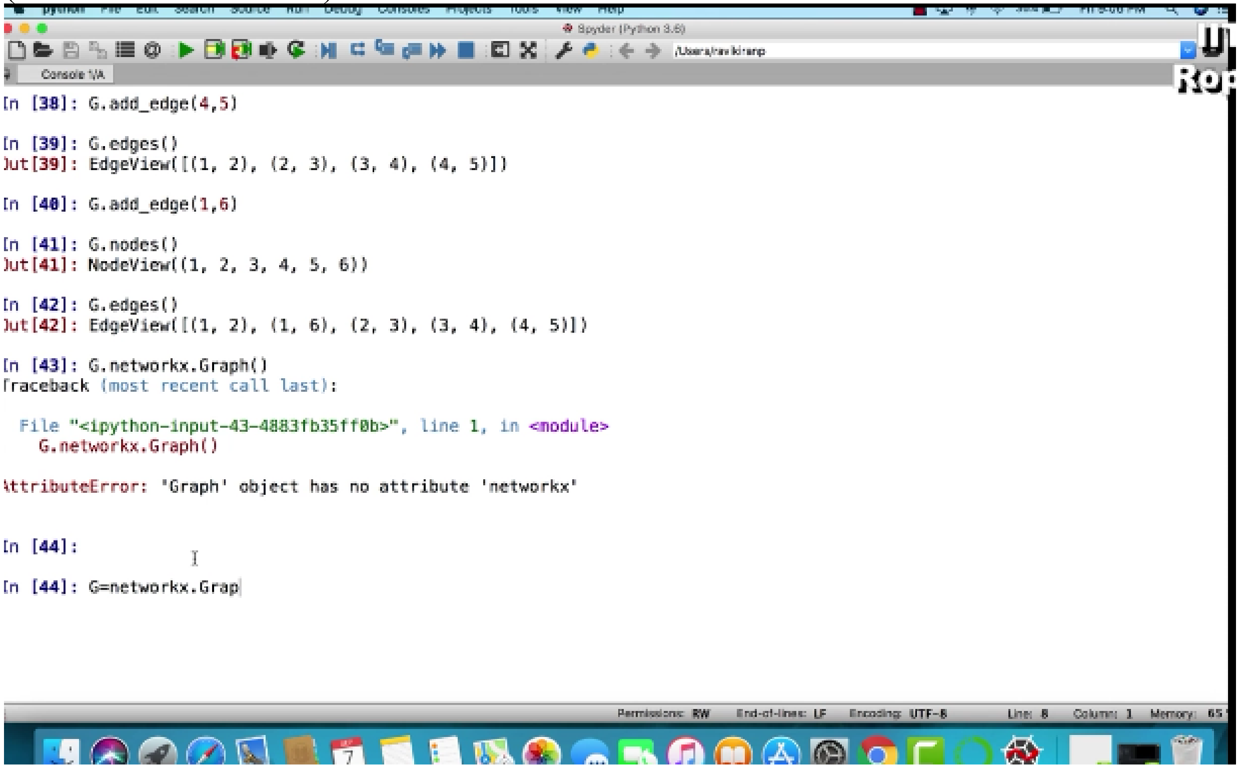
```
In [27]: import networkx

In [28]: G=networkx.Graph()

In [29]: G.add_node(1)

In [30]: G.add_node(2)

In [31]: G.add_node(3)

In [32]: G.add_node(4)

In [33]: G.add_node(5)

In [34]: G.nodes()
Out[34]: NodeView((1, 2, 3, 4, 5))

In [35]: G.add_edge(1,2)

In [36]: G.add_edge(2,3)

In [37]: G.add_edge(3,4)

In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)
```

so each time I have to do this it becomes difficult for me to type this entire word NetworkX, so what I'm going to do now NetworkX graph like this,
(Refer Slide Time: 05:39)



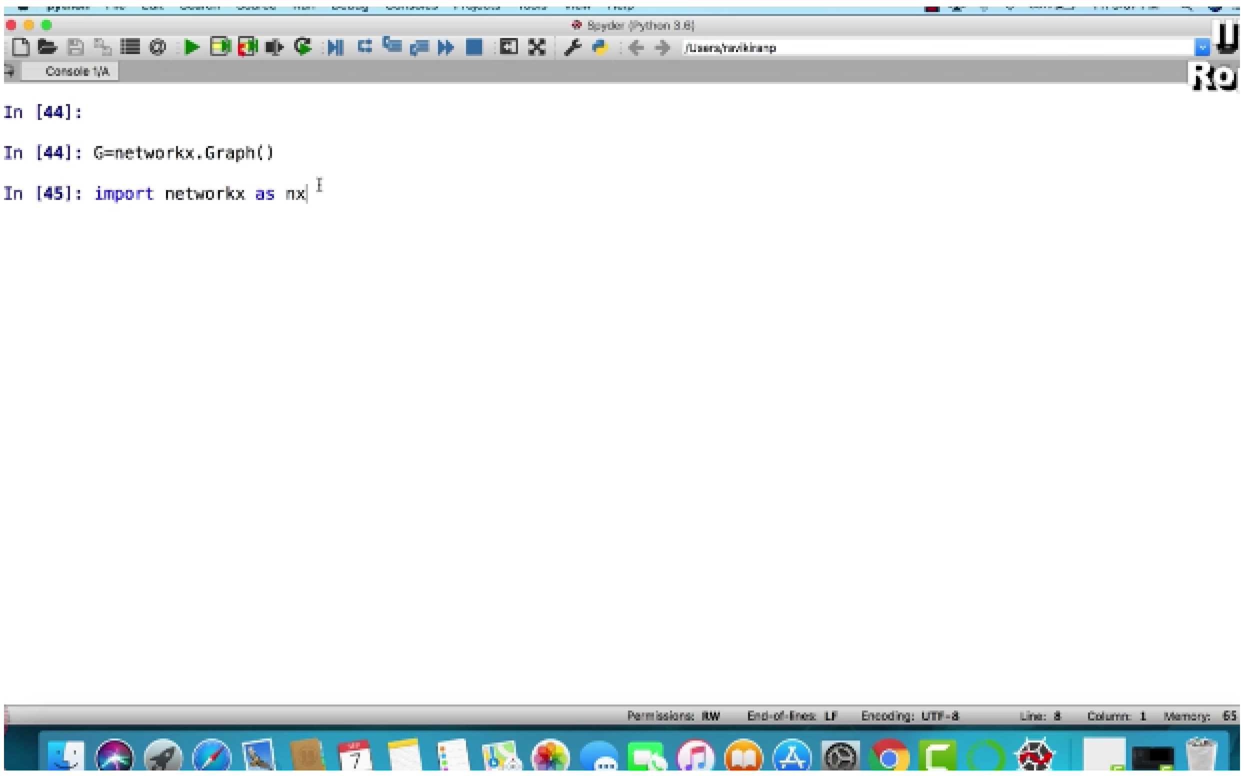
```
In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)

In [41]: G.nodes()
Out[41]: NodeView((1, 2, 3, 4, 5, 6))

In [42]: G.edges()
Out[42]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [43]: G.networkx.Graph()
```

what I'm going to do is so each time we have to use this NetworkX like this G = networkx.graph
(Refer Slide Time: 05:50)



```
In [38]: G.add_edge(4,5)

In [39]: G.edges()
Out[39]: EdgeView([(1, 2), (2, 3), (3, 4), (4, 5)])

In [40]: G.add_edge(1,6)

In [41]: G.nodes()
Out[41]: NodeView((1, 2, 3, 4, 5, 6))

In [42]: G.edges()
Out[42]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [43]: G.networkx.Graph()
Traceback (most recent call last):

  File "<ipython-input-43-4883fb35ff0b>", line 1, in <module>
    G.networkx.Graph()

AttributeError: 'Graph' object has no attribute 'networkx'


In [44]:

In [44]: G=networkx.Grap
```

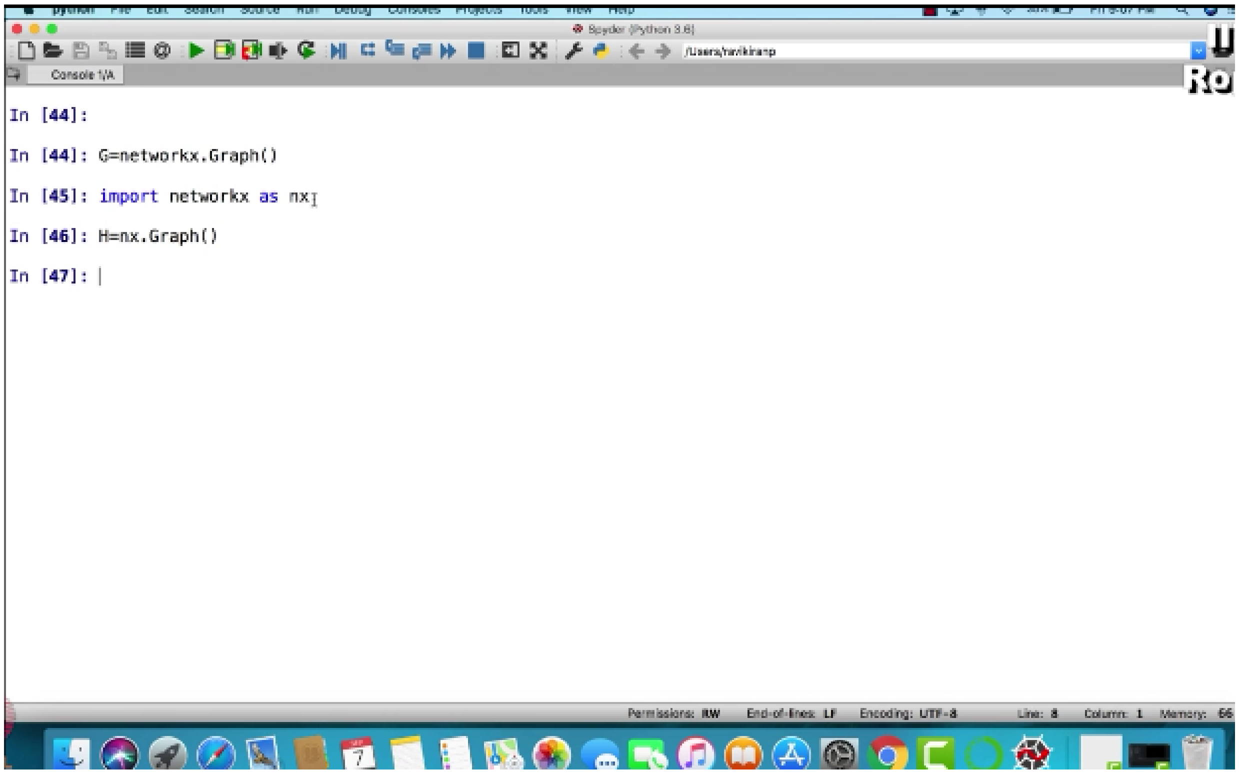
you see I had already used it before in the first line here do you see that I have to type NetworkX again and again, right, so what I'm going to do instead of that is I directly import NetworkX as NX
(Refer Slide Time: 06:14)



```
In [44]:

In [44]: G=networkx.Graph()

In [45]: import networkx as nx
```

so I'll be using NetworkX as NX from now, it's a short form and it becomes easier for us that way.

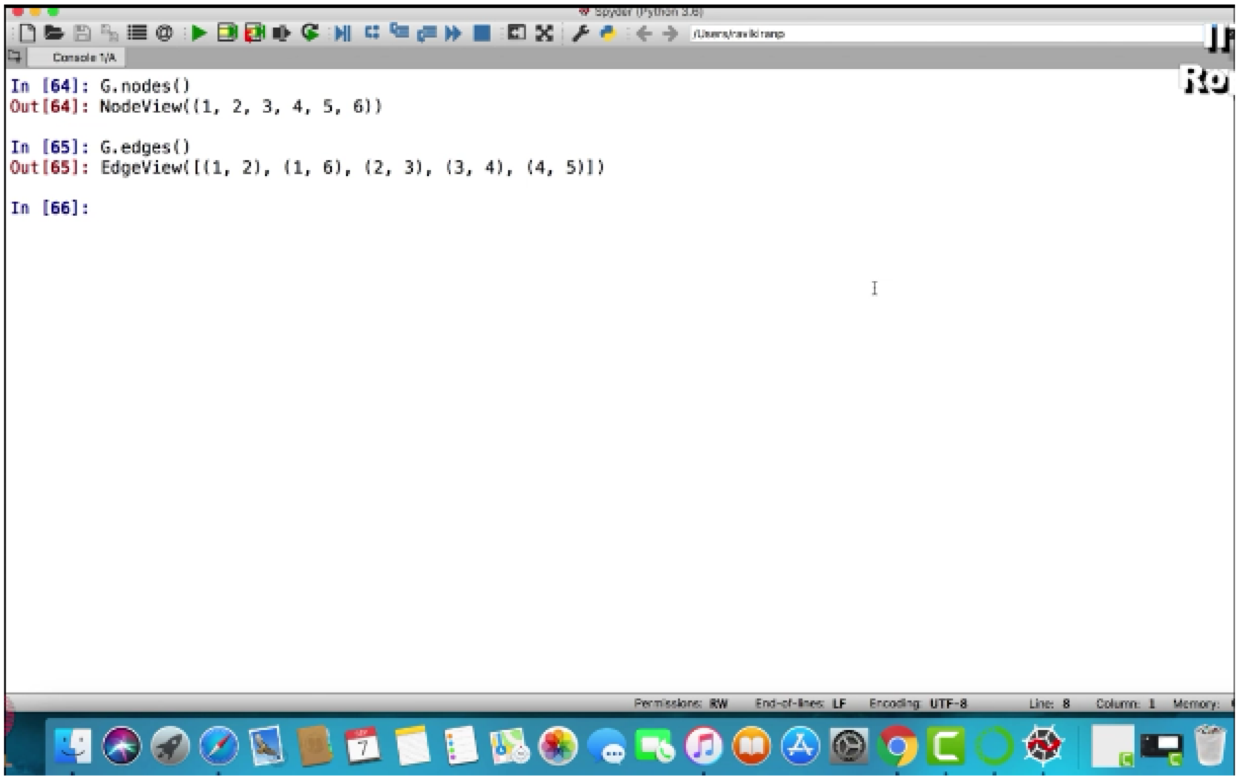
So now if I have to create a new graph let's say H = nx.graph I need not type NetworkX again so H = nx.graph, this graph H has got created,
(Refer Slide Time: 06:40)



```
In [44]:

In [44]: G=networkx.Graph()

In [45]: import networkx as nx

In [46]: H=nx.Graph()

In [47]:
```

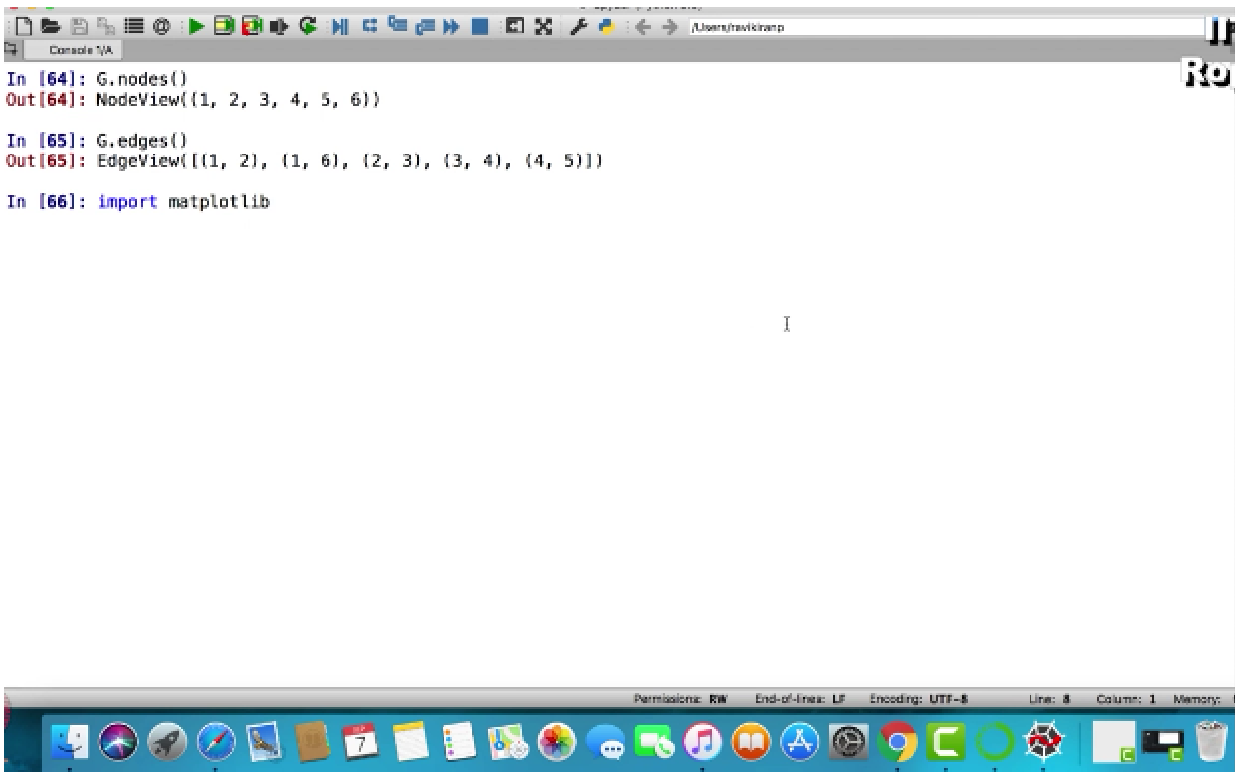
so do you see the importance and significance of NX, so from now on instead of NetworkX, typing NetworkX we will just say NX, so we have created the graph G and H, let me just go back to G.

Let me show you what are the nodes G.nodes, nodes in G are 1, 2, 3, 4, 5, 6 and G.edges, so this will show you all the edges till now (1, 2) (1, 6) (2, 3) (3, 4) (4, 5), so these are the edges please cross check
(Refer Slide Time: 07:13)



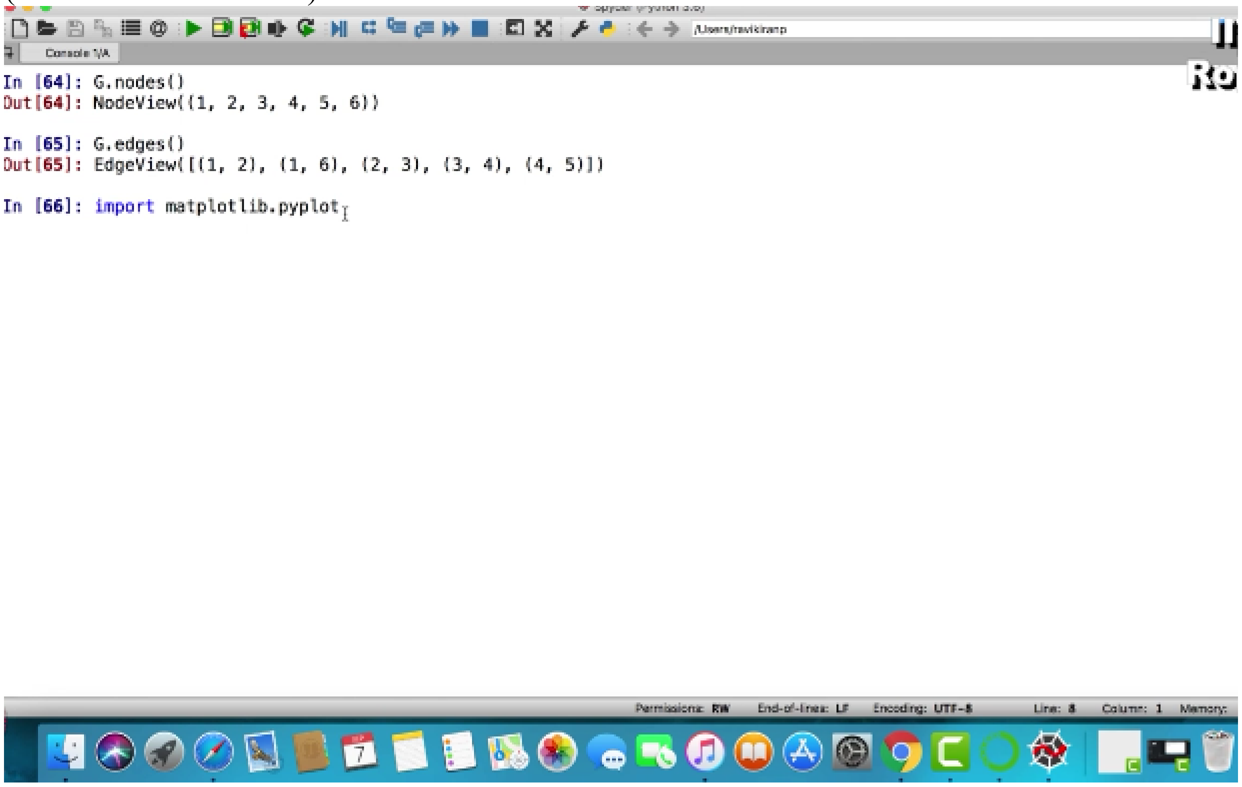
```
In [64]: G.nodes()
Out[64]: NodeView((1, 2, 3, 4, 5, 6))

In [65]: G.edges()
Out[65]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [66]:
```

Now you have the graph there but you want to visualize it, right, you want to see the actual graph, so at this point what are we going to do? We are going to use another new library called the mat plot lib, it will be clear with you as we use it, so what am I going to do, we are going to use that package as by importing it, so import mat plot lib, so did you see this?
(Refer Slide Time: 07:44)



```
In [64]: G.nodes()
Out[64]: NodeView((1, 2, 3, 4, 5, 6))

In [65]: G.edges()
Out[65]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [66]: import matplotlib
```

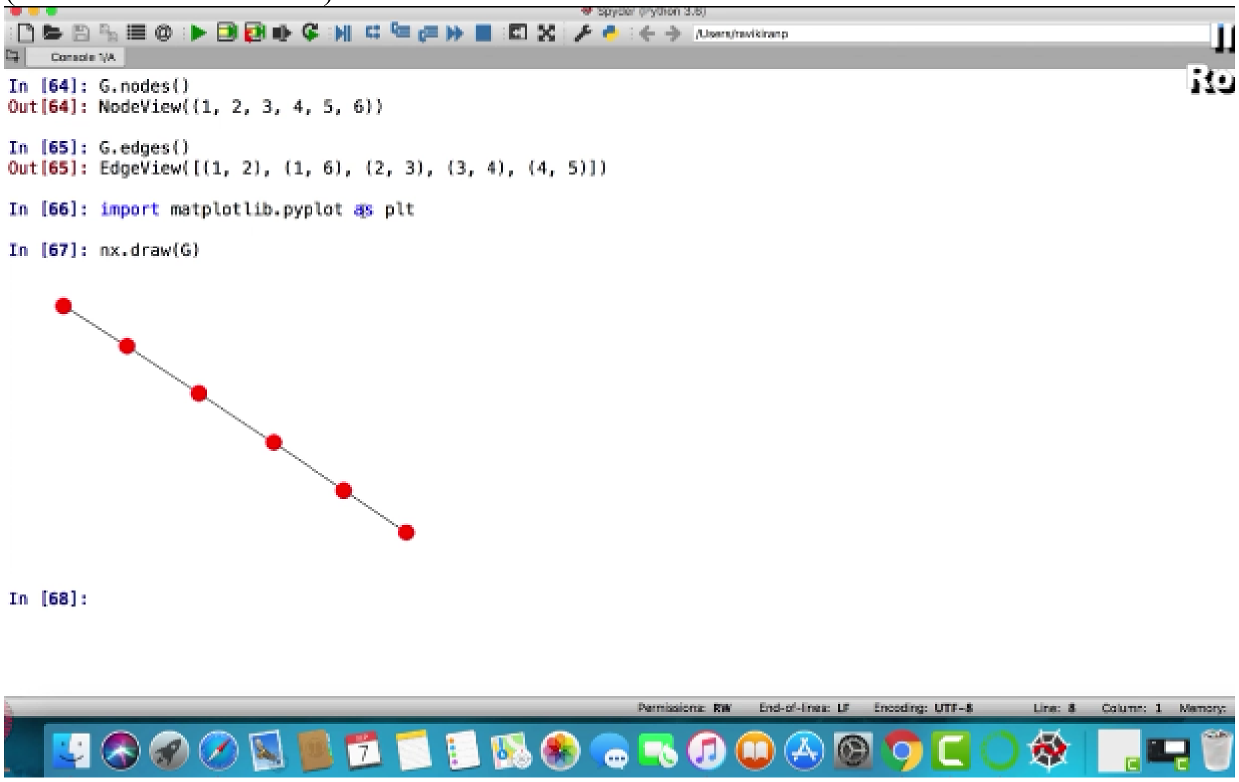
Import mat plot lib, but I am going to use only pyplot this is the sub package what I am going to use, so matplotlib.pyplot this is very relevant to us,
(Refer Slide Time: 7:55)



```
In [64]: G.nodes()
Out[64]: NodeView((1, 2, 3, 4, 5, 6))

In [65]: G.edges()
Out[65]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [66]: import matplotlib.pyplot
```

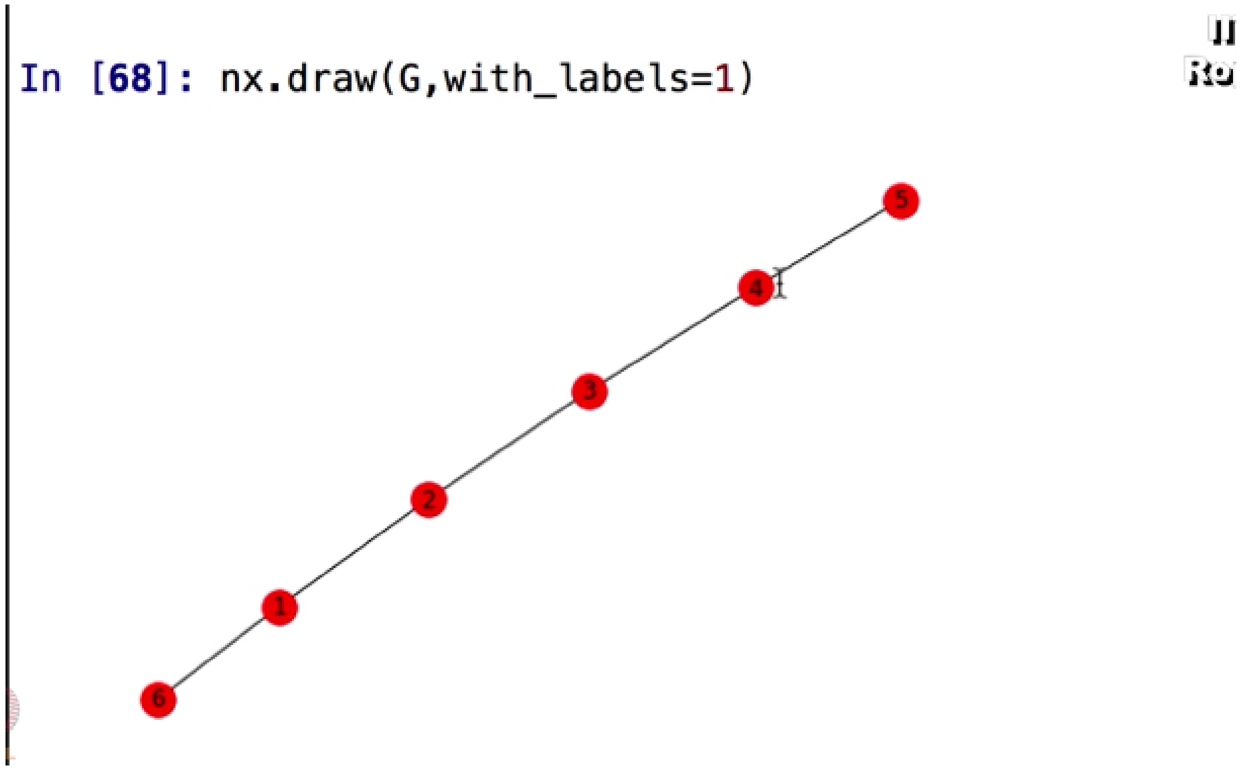
and then it becomes difficult to type this entire phrase matplotlib.pyplot every time, and hence I import this as PLT, you see PLT is much easier than typing this entire thing.

So now this package has got imported, so what are we going to do, nx. networkx., now we are going to draw the graph so you're going to give the command draw, what are you going to draw? G, the graph, do you see the graph here,
(Refer Slide Time: 08:32)



```
In [64]: G.nodes()
Out[64]: NodeView((1, 2, 3, 4, 5, 6))

In [65]: G.edges()
Out[65]: EdgeView([(1, 2), (1, 6), (2, 3), (3, 4), (4, 5)])

In [66]: import matplotlib.pyplot as plt

In [67]: nx.draw(G)

In [68]:
```

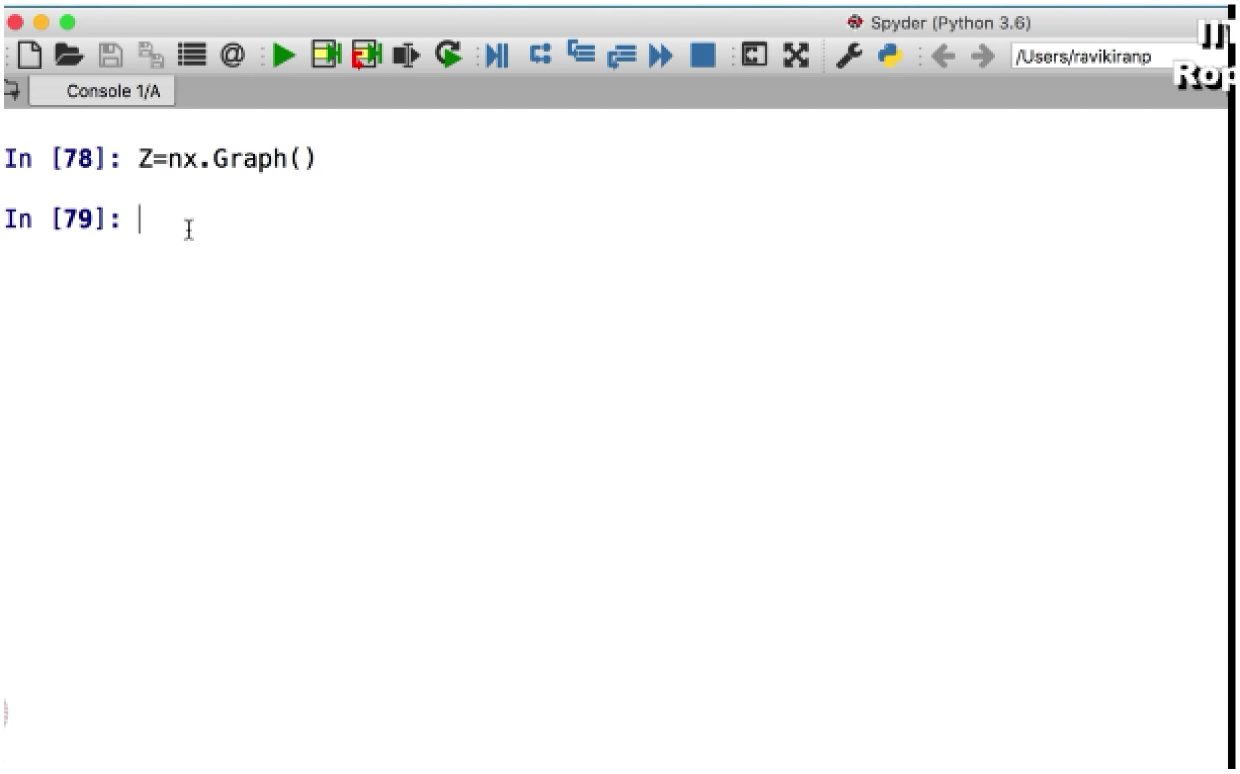
but we don't know which vertex is 1, which vertex is 2, and so on, so what are we going to do? The immediate need for us is to label the vertices, right, so we are going to label the vertices using the command nx.draw in brackets G, you want labels, right, so G with labels, with _labels = 1, so what does this mean? It's going to draw G with labels this command is true, why? I have given one here, so this becomes true and the graph gets labels, do you see this? (1, 6) (2, 3) (4, 5)
(Refer Slide Time: 09:24)



```
In [68]: nx.draw(G,with_labels=1)
```

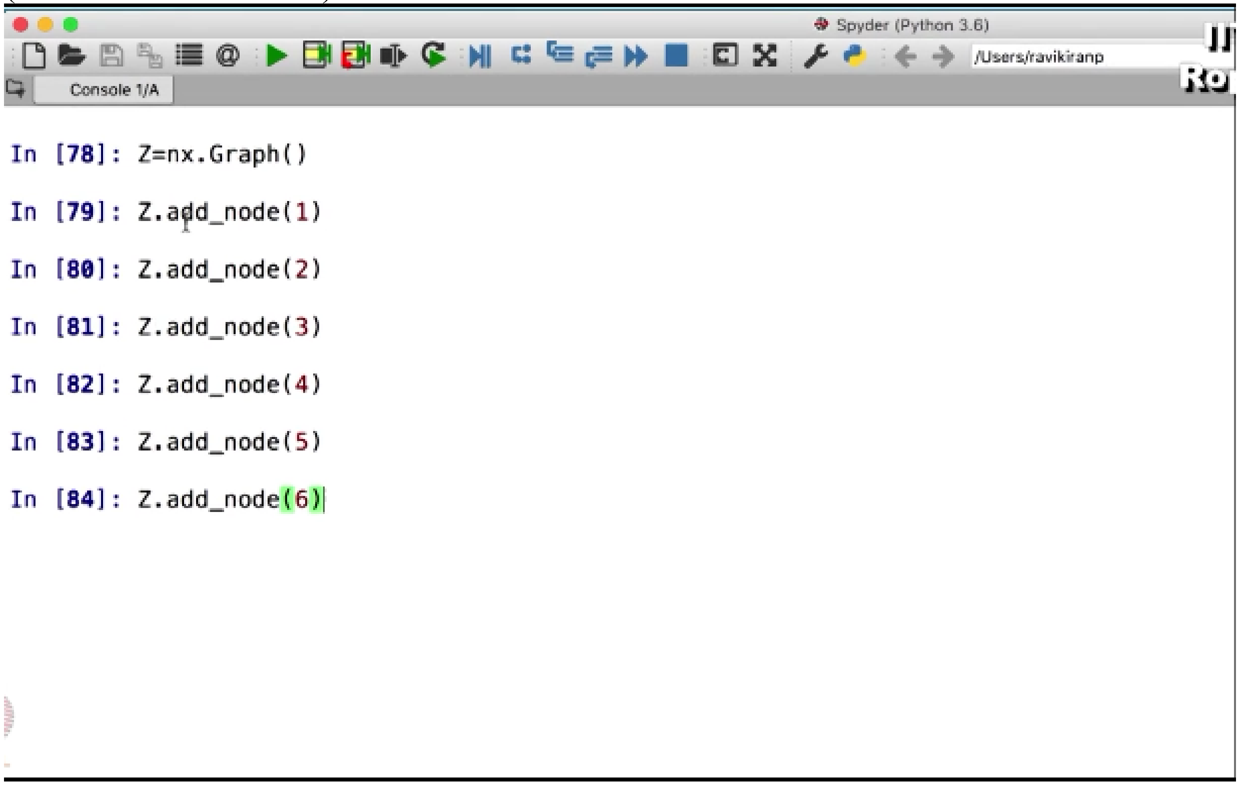
so we have now obtained our graph with labels, so now as a practice we're just going to create another beautiful graph, so what we have to do is NetworkX has already got imported, so we need not import it again, so we can directly start adding our edges and creating the graph, so let me just create a new graph before that you see we have all these commands here, so I just want to remove all of them and have a new screen, what I'm going to do is I'll give this command clear, with this command everything gets cleared and you get your new input length start typing your commands here, so you see we have already imported NetworkX we need not do it again, so what I'm going to do is let me create a new graph say Z, Z = NX, we had imported it as NetworkX, nx.graph like this,

(Refer Slide Time: 10:26)



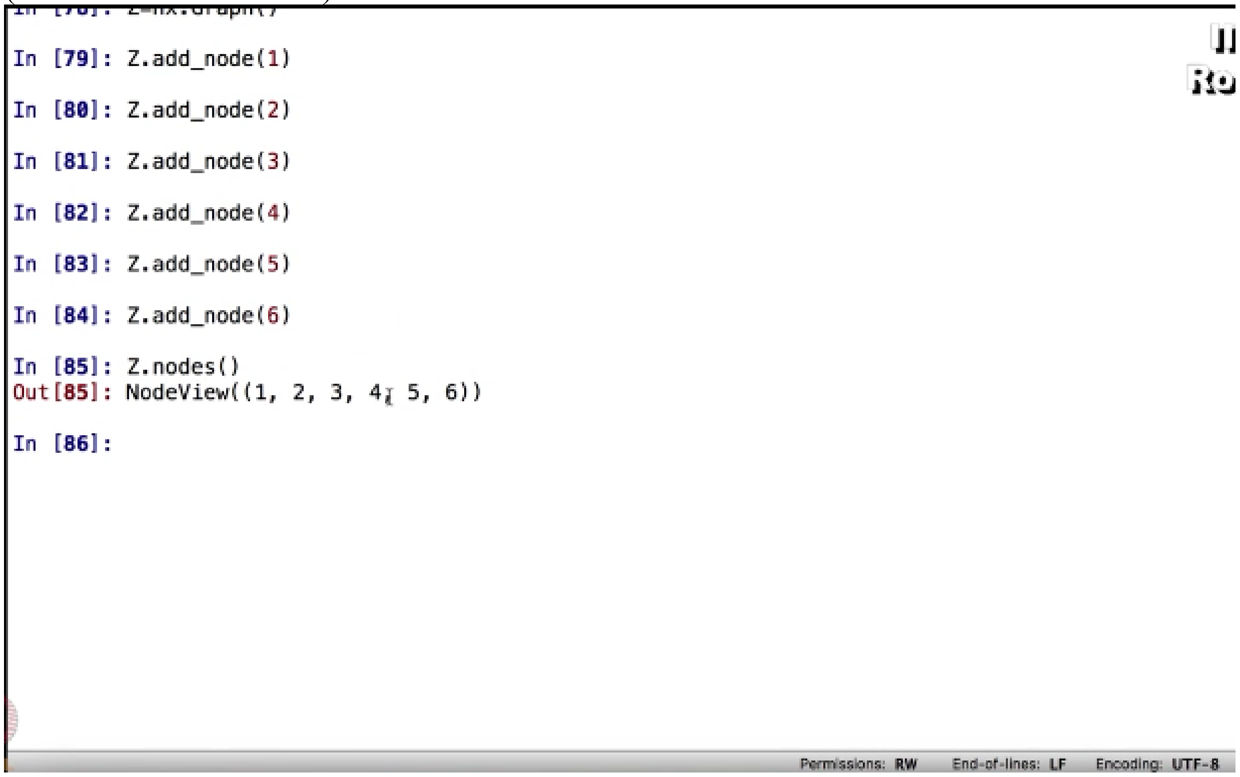
```
In [78]: Z=nx.Graph()

In [79]: |
```

so now the graph has got created, what do we have to do next? We have to add edges and nodes, so what I'm going to do is Z.add_node as 1, and then Z.add_node 2, and then Z.add_node 3, let us take a few more Z.add_node 4, and then Z.add_node say 5, and then Z.add_node the last one 6,
(Refer Slide Time: 11:12)



```
In [78]: Z=nx.Graph()

In [79]: Z.add_node(1)

In [80]: Z.add_node(2)

In [81]: Z.add_node(3)

In [82]: Z.add_node(4)

In [83]: Z.add_node(5)

In [84]: Z.add_node(6)
```

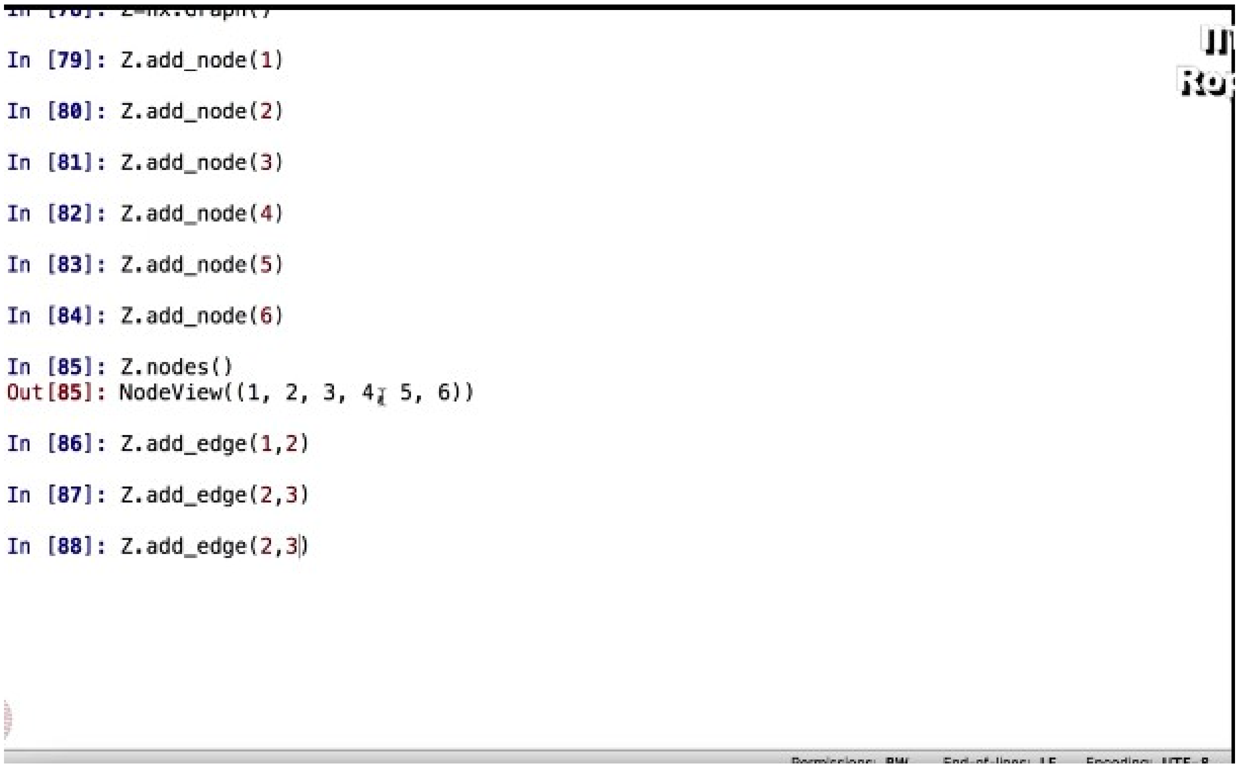
so we have added 6 vertices, let us just list them all, Z.nodes in brackets, so this is giving all our nodes here.
(Refer Slide Time: 11:26)



```
In [79]: Z.add_node(1)

In [80]: Z.add_node(2)

In [81]: Z.add_node(3)

In [82]: Z.add_node(4)

In [83]: Z.add_node(5)

In [84]: Z.add_node(6)

In [85]: Z.nodes()
Out[85]: NodeView((1, 2, 3, 4, 5, 6))

In [86]:
```

Permissions: RW End-of-lines: LF Encoding: UTF-8

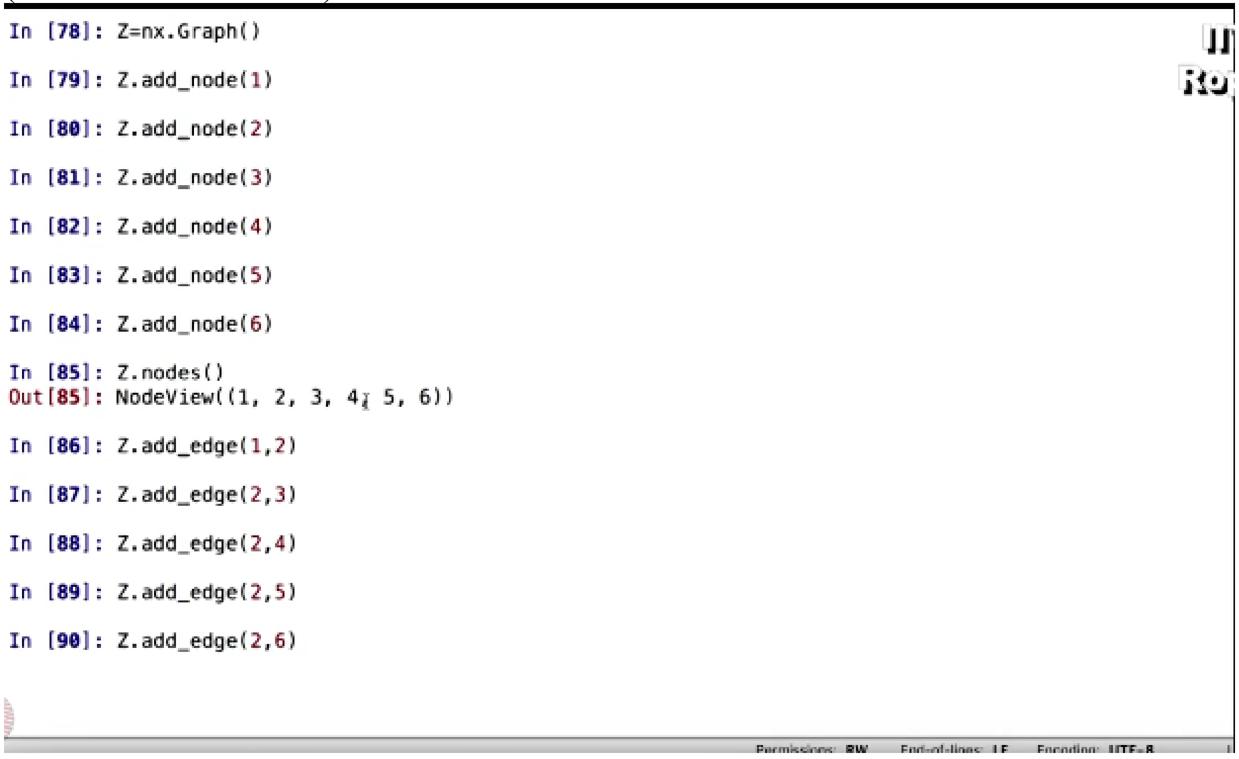
Now let us add some edges Z.add_edge say (1, 2), again Z.add_edge say (2, 3), for simplicity you can always use your up arrow and get the command like this
(Refer Slide Time: 11:44)



```
In [78]: Z=nx.Graph()

In [79]: Z.add_node(1)

In [80]: Z.add_node(2)

In [81]: Z.add_node(3)

In [82]: Z.add_node(4)

In [83]: Z.add_node(5)

In [84]: Z.add_node(6)

In [85]: Z.nodes()
Out[85]: NodeView((1, 2, 3, 4, 5, 6))

In [86]: Z.add_edge(1,2)

In [87]: Z.add_edge(2,3)

In [88]: Z.add_edge(2,3)
```

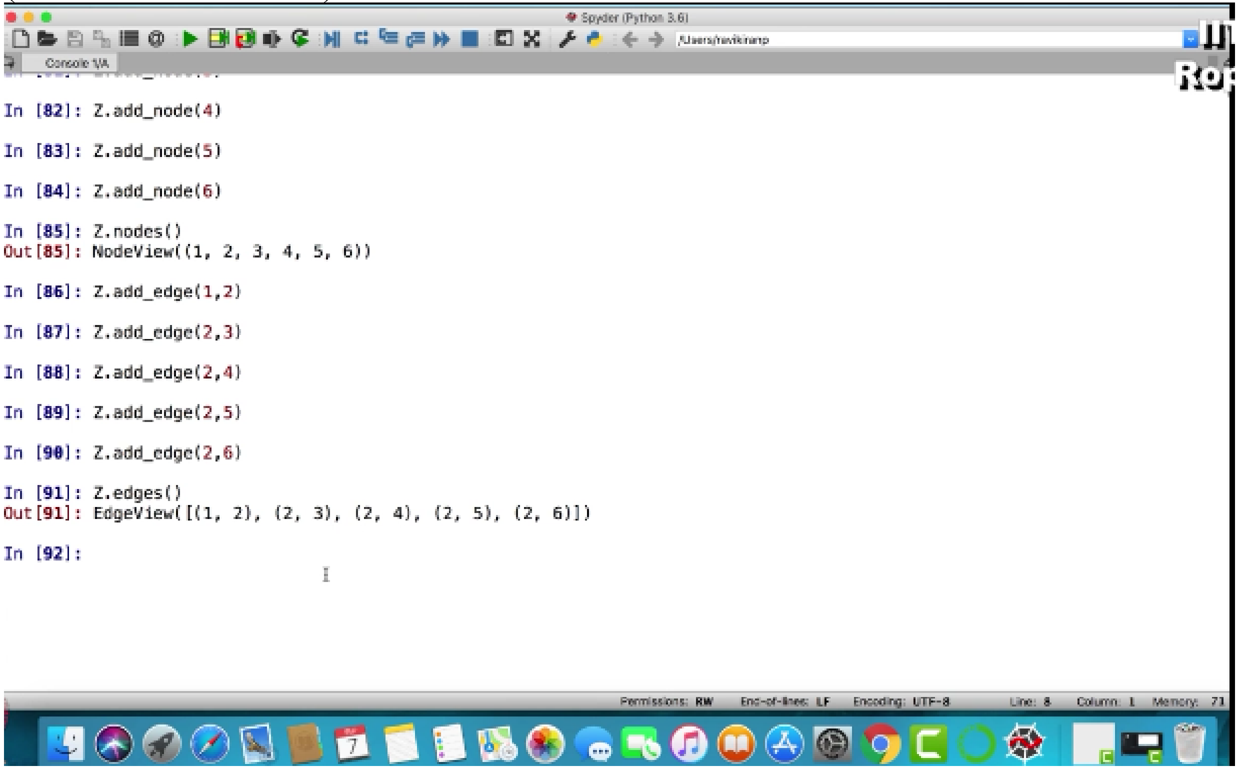
and then go back to your bracket and change what you had to write there, so (1, 2) (2, 3) I have given, then let me give (2, 4) and then again (2, 5), so do you observe what I'm doing, I'm just giving all edges as connected to do, so (2, 6), yes

(Refer Slide Time: 12:09)



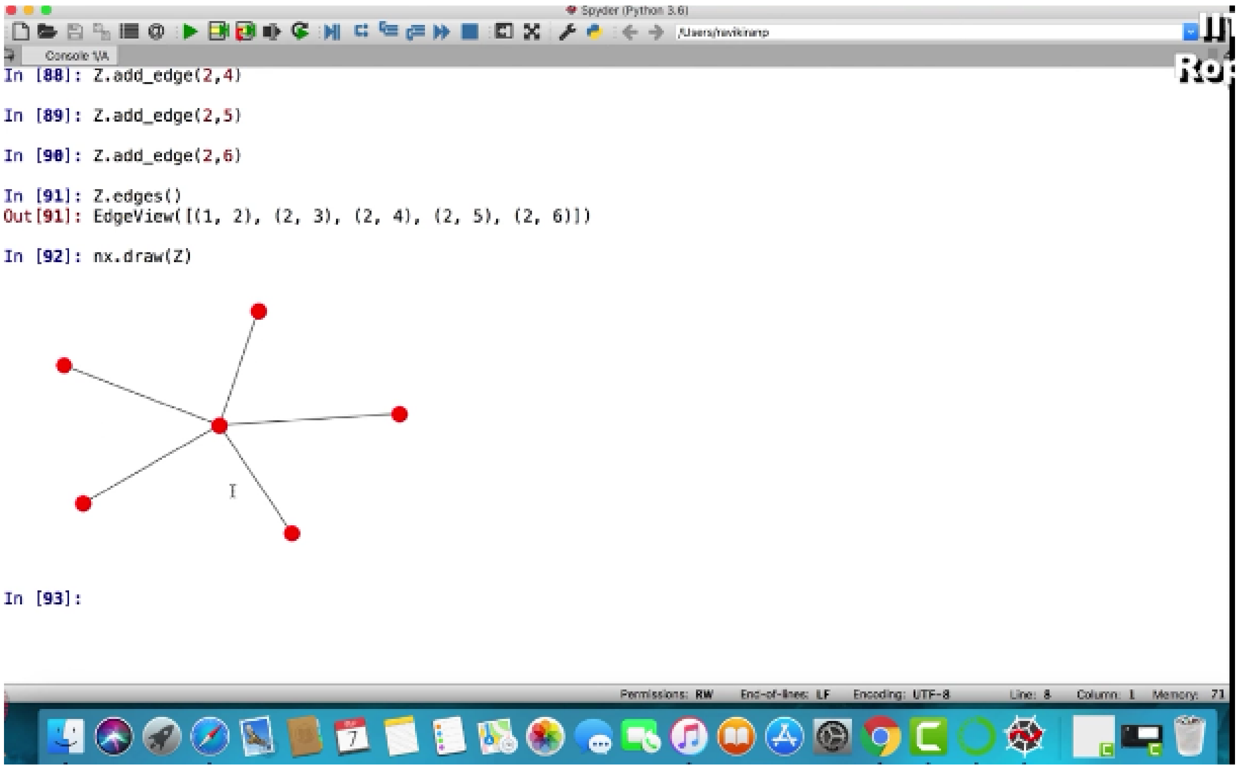
```
In [78]: Z=nx.Graph()

In [79]: Z.add_node(1)

In [80]: Z.add_node(2)

In [81]: Z.add_node(3)

In [82]: Z.add_node(4)

In [83]: Z.add_node(5)

In [84]: Z.add_node(6)

In [85]: Z.nodes()
Out[85]: NodeView((1, 2, 3, 4, 5, 6))

In [86]: Z.add_edge(1,2)

In [87]: Z.add_edge(2,3)

In [88]: Z.add_edge(2,4)

In [89]: Z.add_edge(2,5)

In [90]: Z.add_edge(2,6)
```

so let me check all the edges now, Z.edges brackets, yeah, (1, 2) (2, 3) (2, 4) (2, 5) (2, 6) do you see all the edges and the nodes here.
(Refer Slide Time: 12:26)



```
In [82]: Z.add_node(4)

In [83]: Z.add_node(5)

In [84]: Z.add_node(6)

In [85]: Z.nodes()
Out[85]: NodeView((1, 2, 3, 4, 5, 6))

In [86]: Z.add_edge(1,2)

In [87]: Z.add_edge(2,3)

In [88]: Z.add_edge(2,4)

In [89]: Z.add_edge(2,5)

In [90]: Z.add_edge(2,6)

In [91]: Z.edges()
Out[91]: EdgeView([(1, 2), (2, 3), (2, 4), (2, 5), (2, 6)])

In [92]:
```

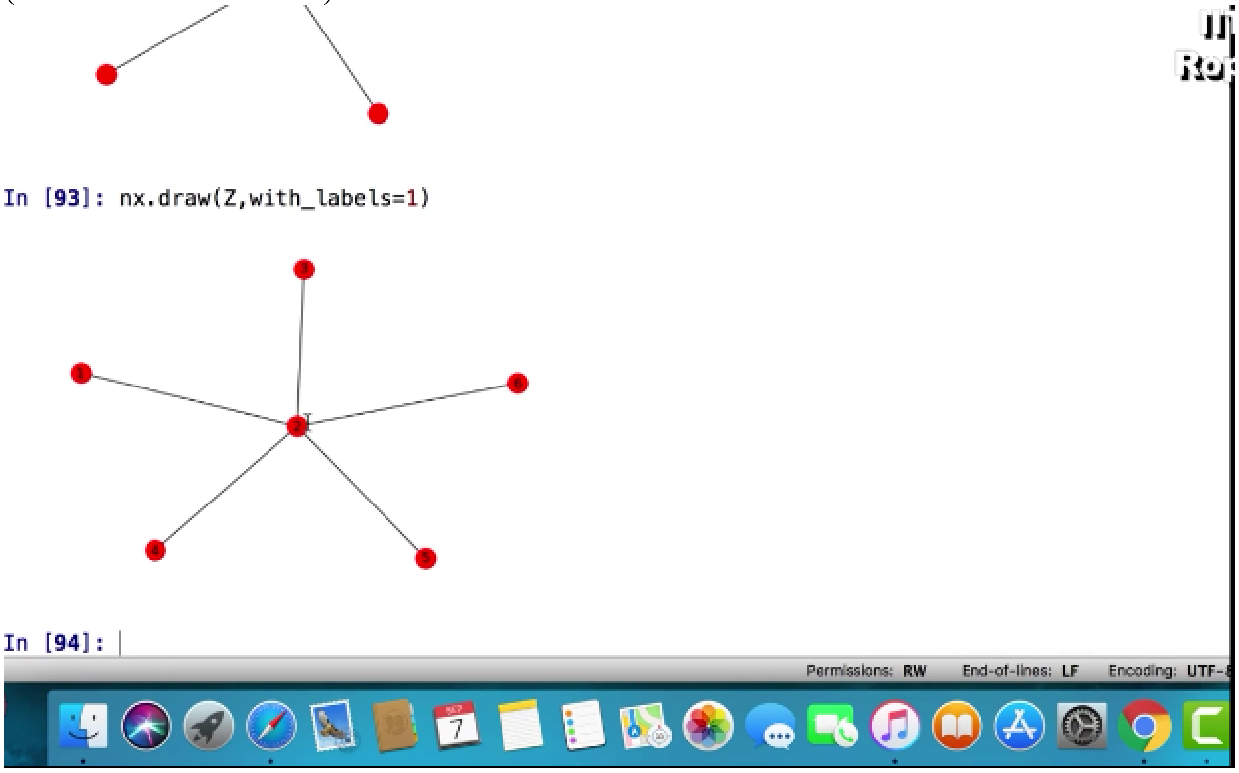
So now let me draw the graph we have already imported back plot libs, so we need not do it again, so what I'm going to do is nx.draw this was the command to draw a graph Z, whatever you have labeled your graph as you need to give that inside here, nx.draw Z, do you see something this beautiful graph here,
(Refer Slide Time: 12:49)



```
In [88]: Z.add_edge(2,4)

In [89]: Z.add_edge(2,5)

In [90]: Z.add_edge(2,6)

In [91]: Z.edges()
Out[91]: EdgeView([(1, 2), (2, 3), (2, 4), (2, 5), (2, 6)])

In [92]: nx.draw(Z)

In [93]:
```

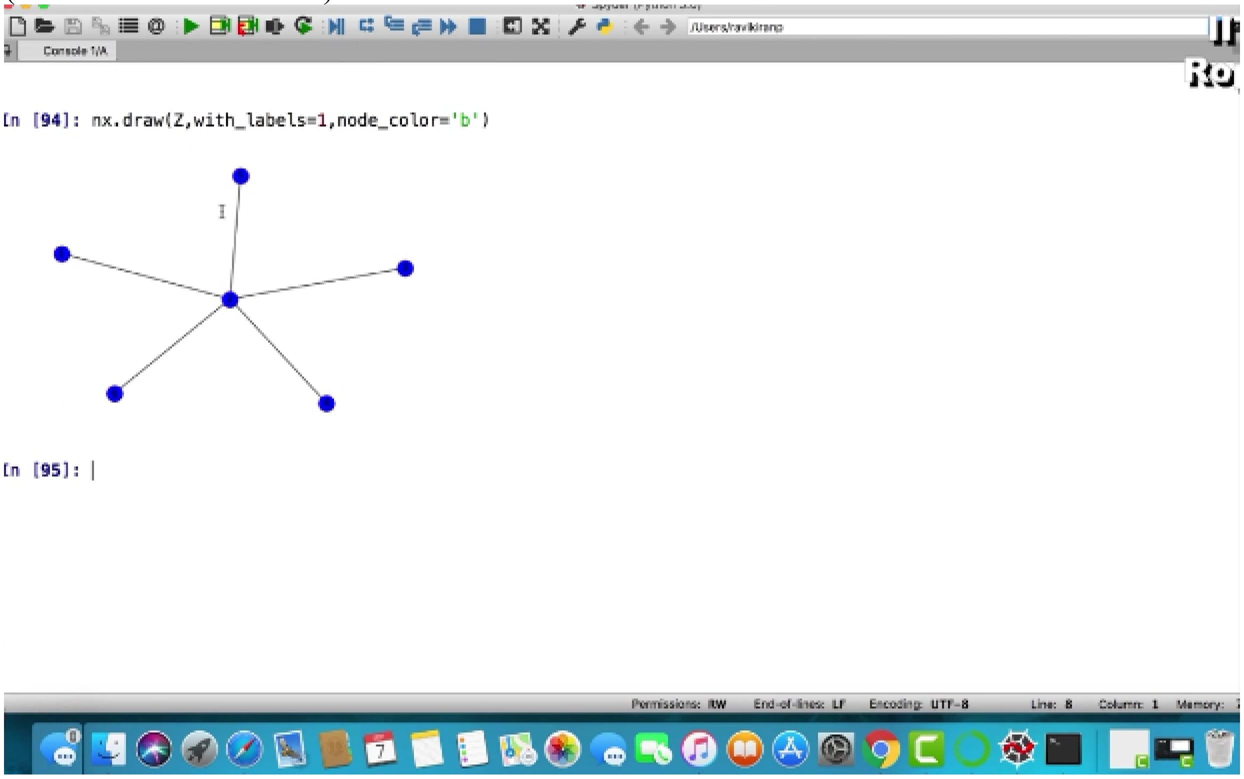
but again as you remember we need to add labels in case we need the labeled vertices so what I'm going to do is nx.draw it's all small its case sensitive, please note that nx.draw Z with labels = 1, so this is true labels it means that I am going to give my labels now, let us see the graph, yes,

(Refer Slide Time: 13:18)



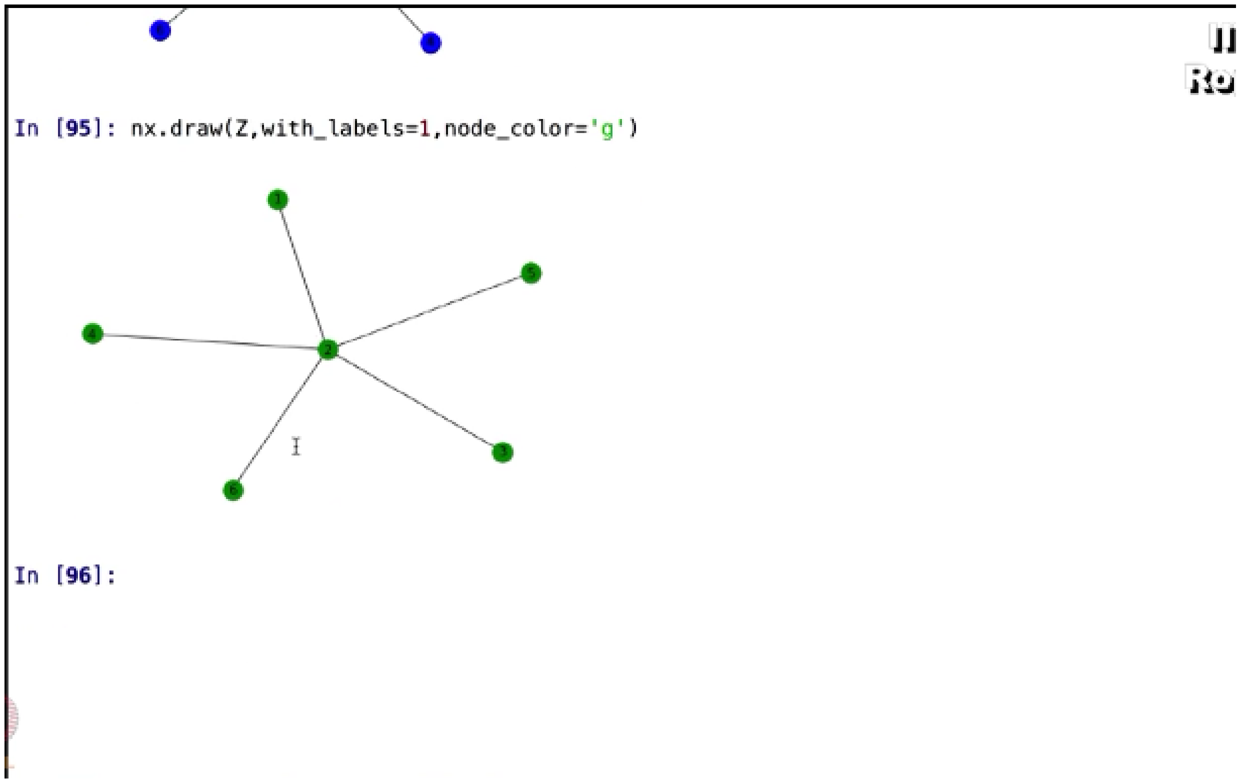
```
In [93]: nx.draw(Z,with_labels=1)

In [94]:
```

so 2, this vertex 2 is connected to 1, 3, 4, 5 and 6, did you see this beautiful graph? So you can just play around like this and create more graphs by randomly adding nodes and vertices, so you see we have got this graph here but don't you feel that this red color is quite boring to see always, so let us change the node color to something else, how do we do that? Yes, so you give nx.draw as earlier the graph was Z and then with labels as true, and then here what do you have to do is node_color, node_color = n codes you give B, B stands for blue, let us see, did you see that we have got blue color nodes here, isn't this very interesting
(Refer Slide Time: 14:15)



```
In [94]: nx.draw(Z,with_labels=1,node_color='b')

In [95]: |
```

now let me change the node color say to G, green,
(Refer Slide Time: 14:23)



```
In [95]: nx.draw(Z,with_labels=1,node_color='g')
```

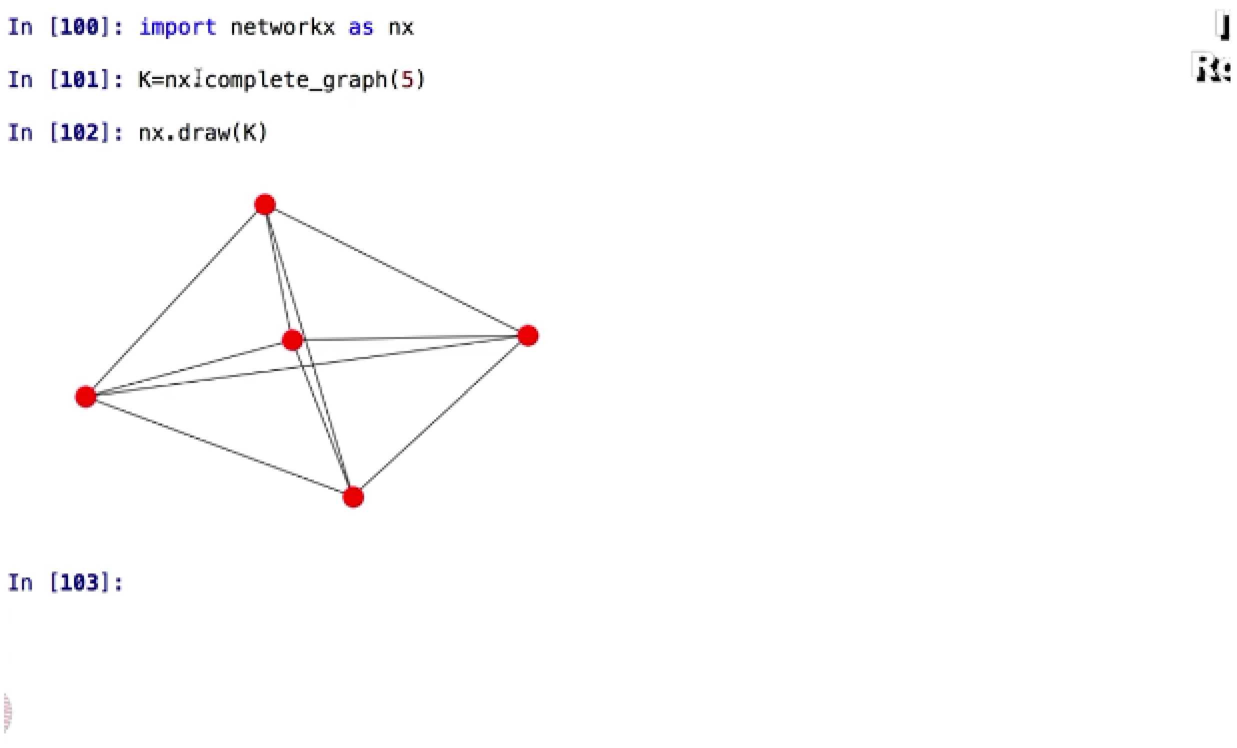
```
In [96]:
```

where we've got green color nodes here.

Now please don't give any random color and try and you'll end up getting an error you see you have to give some standard colors, and you see these beautiful nodes.

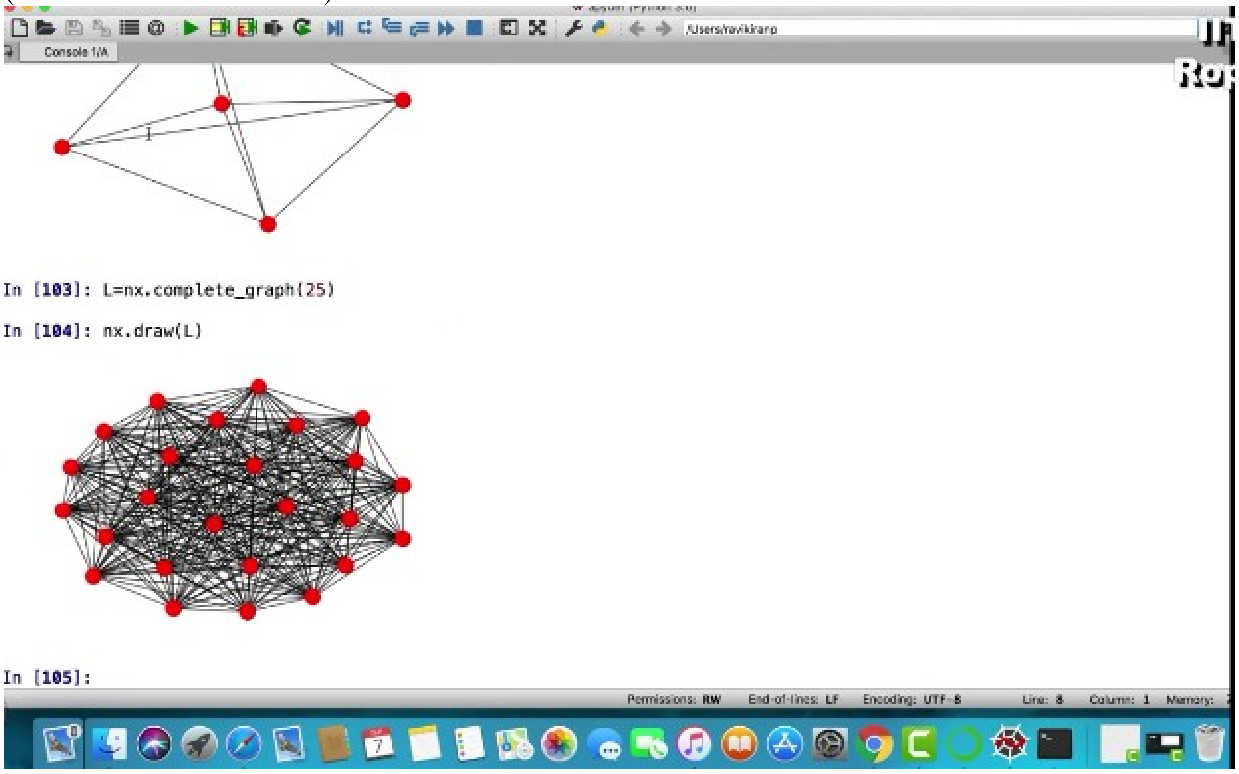
Now let me clear my screen, yeah, so now we are going to start with a new topic, what am I going to do now? We will see you every time import your NetworkX as NX and you add edge, you create a graph, you add edge, you add nodes, don't you think this is a very tedious process suppose you want to draw a graph on say 25 or say 100 nodes, can you manually sit and add all your edges every time? No, it's just impossible, so there are various methods to create the graph one among them is using the built-in functions which comment with NetworkX, so what is it? You see we have seen some standard graphs like a complete graph, a path graph, and then a cycle, so we can do that just with a one-line code.

So let me import NetworkX as NX, now I'll start doing it, let me see K = nx. you see complete_graph and in brackets you have to now give how many nodes you wanted on, let me just give 5 to begin with, now your graph K has got created, nx.draw if you give in brackets, K is your graph, do you see this graph on 5 nodes here this complete graph on 5 nodes.
(Refer Slide Time: 16:18)


```
In [100]: import networkx as nx

In [101]: K=nx.complete_graph(5)

In [102]: nx.draw(K)
```

```
In [103]:
```

Now let me just do another one, let me say L = nx.complete_graph on let me say some 25, 25 nodes, and then let us see, visualize it using nx.draw and L,
(Refer Slide Time: 16:42)

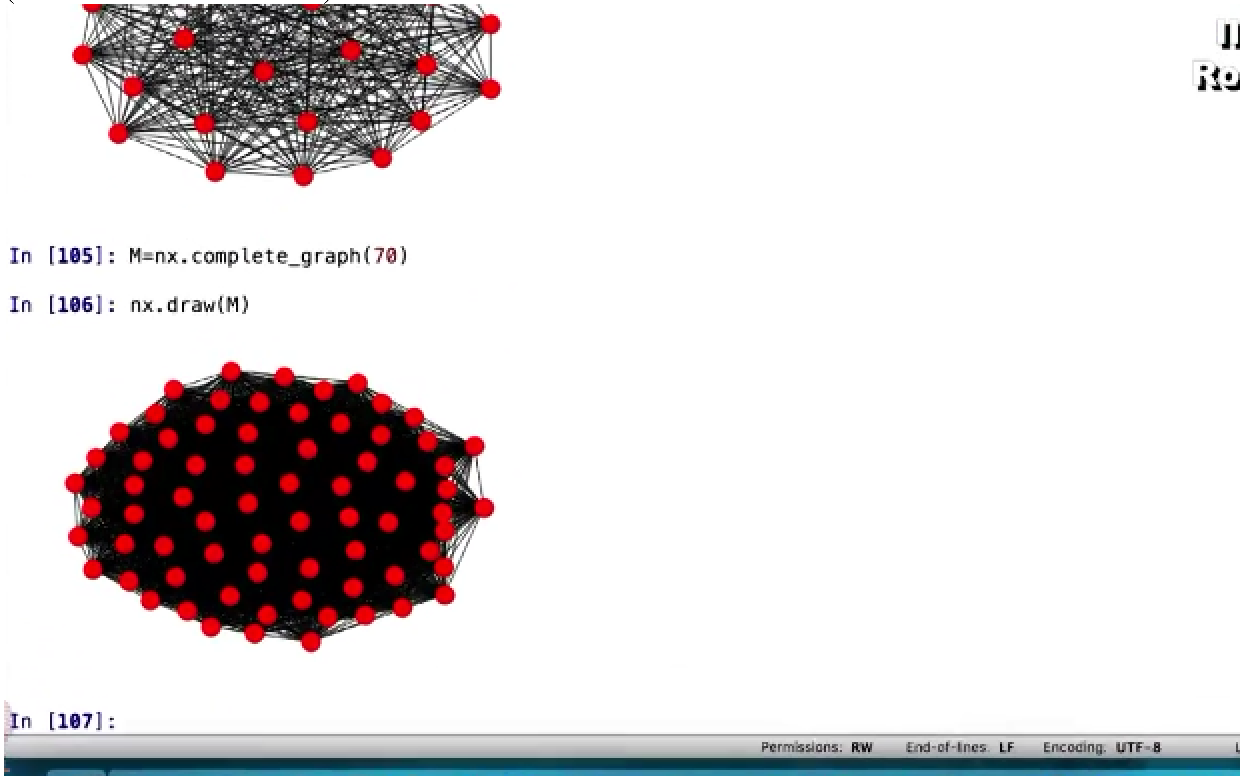


```
In [103]: L=nx.complete_graph(25)

In [104]: nx.draw(L)
```

```
In [105]:
```

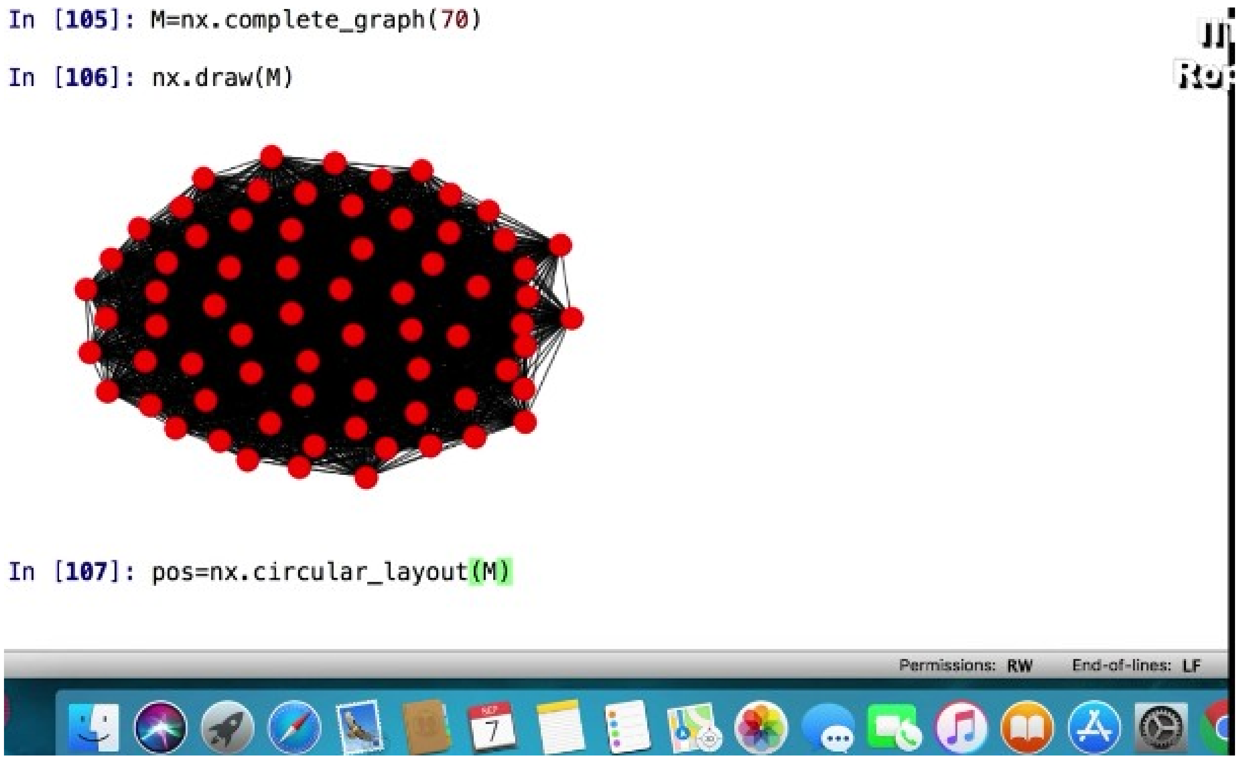
do you see this complicated network here, isn't this very interesting to just play around using an nx.complete_graph give various number of nodes and then visualize the graphs.

Now let me make your life a little more complicated let me say M = nx.complete_graph and let me give some 70 nodes, and then nx.draw M,
(Refer Slide Time: 17:15)



```
In [105]: M=nx.complete_graph(70)

In [106]: nx.draw(M)

In [107]:
```

do you see the structure here? You are not even able to identify the edges hidden in-between, now what comes to our rescue now is another beautiful feature called layout, what does it do? It helps us segregate the edges properly and visualize them clearly, so now I will give something like this position POS, position = nx.circular_layout, and what this circular layout of this graph, okay, my graph is M not G,
(Refer Slide Time: 18:00)

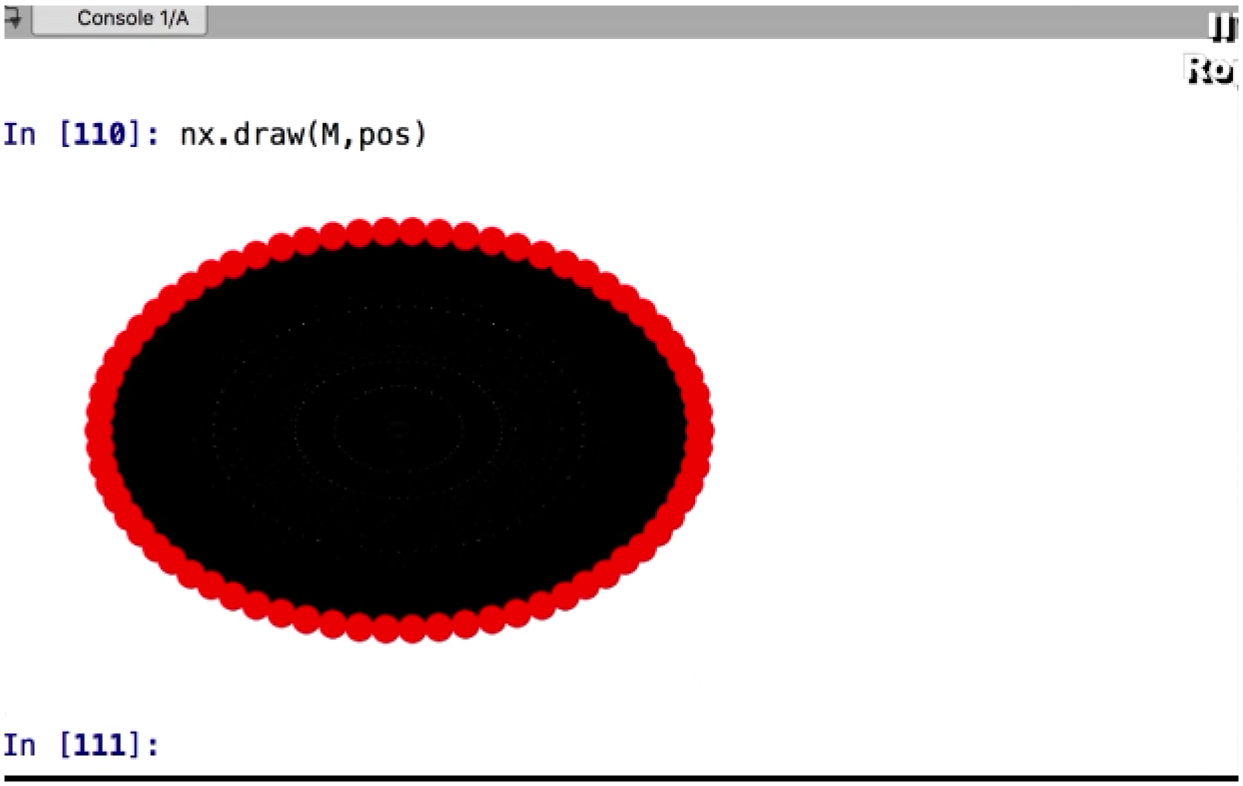


```
In [105]: M=nx.complete_graph(70)

In [106]: nx.draw(M)
```

```
In [107]: pos=nx.circular_layout(M)
```

yes, so now I have given this layout called circular, there are several other layouts, we will be seeing one by one, so position nx.circular_layout M.

Now let me see how the graph looks like nx.draw and then the graph M, POS, so the graph looks like this,
(Refer Slide Time: 18:25)

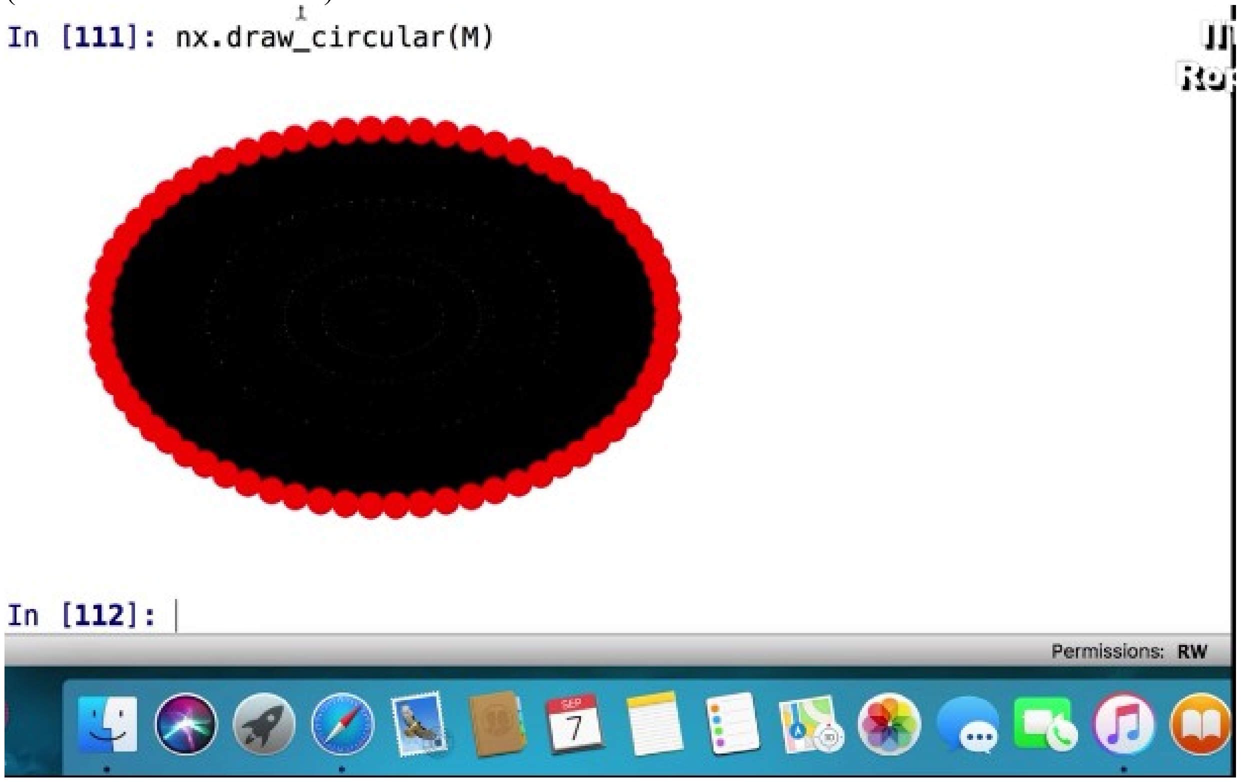


```
In [110]: nx.draw(M,pos)
```

```
In [111]:
```

yes, did you see all the nodes are in the circular format towards the end of it like this, see it becomes difficult if we have to each time use nx.draw as well as nx.circular_layout, well we have a simplified command nx.draw_circular and in the bracket let me give M

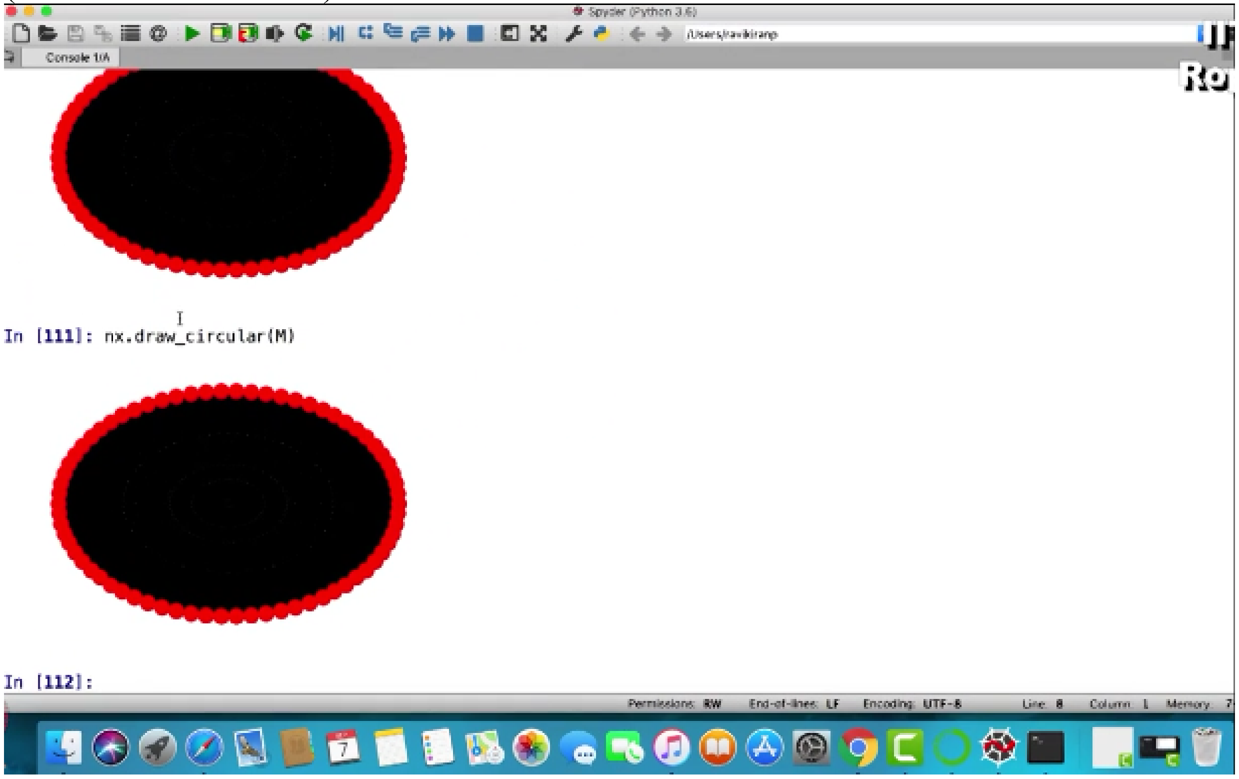
(Refer Slide Time: 18:55)



```
In [111]: nx.draw_circular(M)
```

```
In [112]:
```

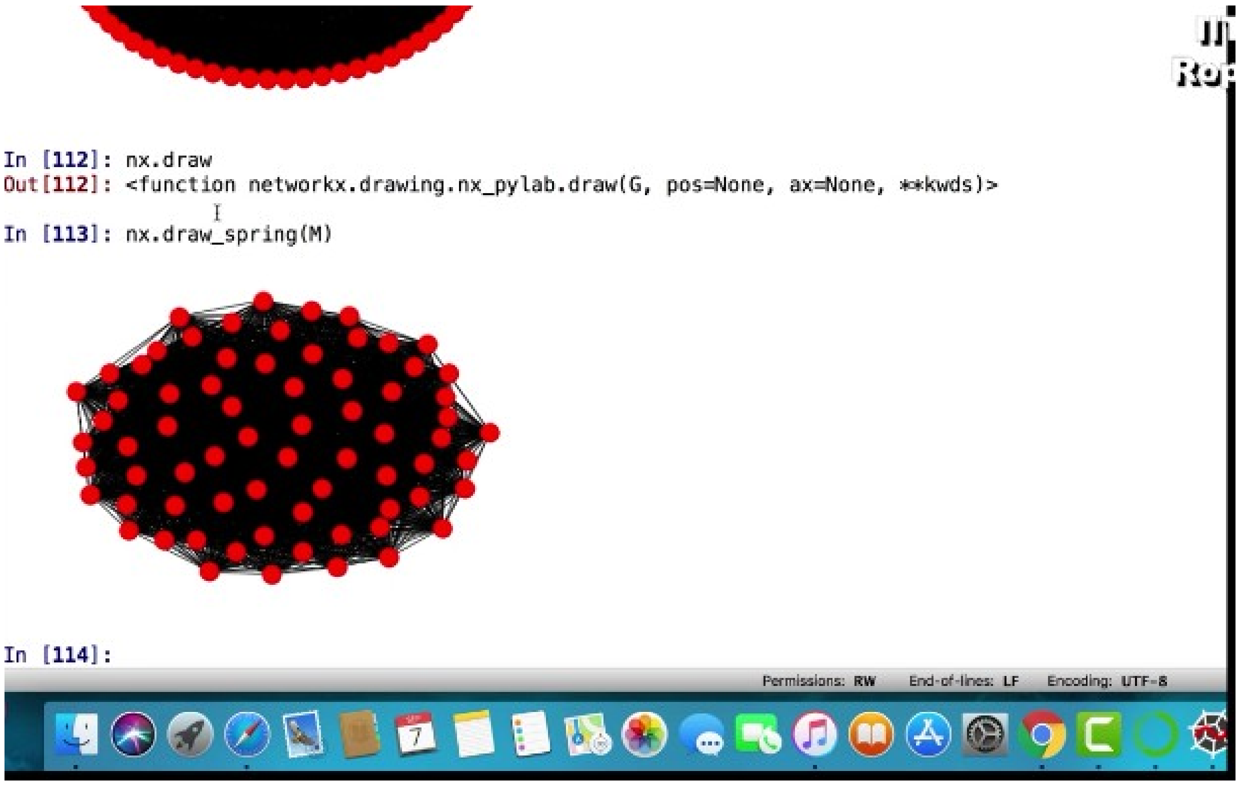
you see it just came with one command, you need not use 2 commands,
(Refer Slide Time: 18:58)



```
In [111]: nx.draw_circular(M)

In [112]:
```

well what I want to mention here is the significance of these built-in functions in NetworkX, we'll be seeing more of them.

Let us now see another layout nx., oh I'm sorry, it is nx.draw_spring, there are some layout called like spring, let us see how it looks
(Refer Slide Time: 19:23)

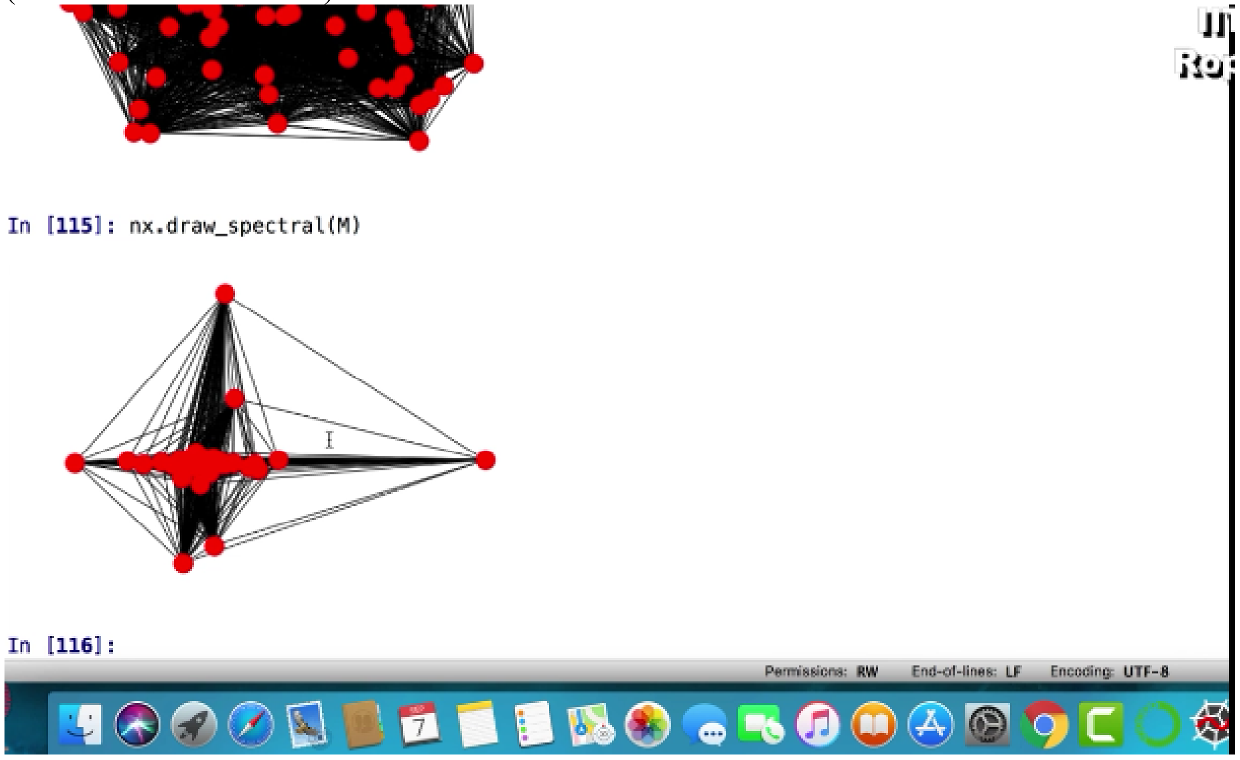


```
In [112]: nx.draw
Out[112]: <function networkx.drawing.nx_pylab.draw(G, pos=None, ax=None, **kwds)>

In [113]: nx.draw_spring(M)
```

```
In [114]:
```

do you see this is yet another layout or a visual feature of this graph, nx.draw, let me say random, it's just going to give a random look to the graph, and M we get a random one like this. Now nx.draw let me say spectral M,

(Refer Slide Time: 19:58)



```
In [115]: nx.draw_spectral(M)
```

```
In [116]:
```

do you see how the graph is looking, so these are various layouts, we will be seeing more functions in the next video.