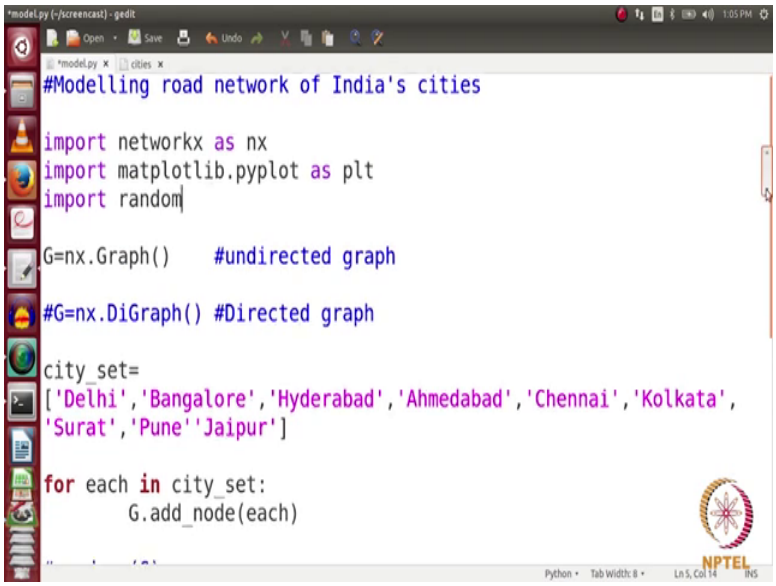
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture – 06
Introduction to Social Networks
Introduction to Networkx-2

(Refer Slide Time: 00:08)



Hello everybody, in the screen cast we are going to do a very interesting programming assignment, what we want to tell you through the screen cast is how do you pick a real world problem and model it with the help of python and networkx. So, we will be using both python as well as networkx in the screen cast. So, that you can learn how to put both of them together and use the real world problem which we will be picking is the network of cities.

So, we have a lot of cities in India. So, we represent every city as a node and then we have edges between these cities we which are the roads connecting these cities. So, we will be assuming that between 2 cities there is one road though there can be many. So, we pick one and then there might be a travelling time associated with each of the road. So, you can visualize this as a network where nodes are the cities and edges are the roads connecting them some roads might be good some roads might be bad some roads can be congested some roads can be free and so on which affect their travelling time if you

might if you would have heard of the travelling salesman problem you would be very easily able to imagine it, but its otherwise also its easy to imagine. So, you have a network where there are cities and there are roads connecting the cities. So, we will be making this network with the help of python and networkx will be visualizing it and we will be looking at various properties of this network.

So, let us get started how do we do it? So, what we are going to do is we are going to model the road network of India let say India's cities, let us say we are going to model the road network of cities in India as I have already told you. So, what is the graph here? So, let me first save this file of this folder screen cast here let say I save it with the name cities dot Mumbai for better event let say model dot py. So, I have a model dot py here. So, let us see how do we model the road network? So, let me first make a graph. So, for making a graph I need the networkx package which needs to be imported.

So, we import networkx as nx. So, now, we can use it and we are going to take the graph s undirected. So, we are going to assume that if there is a road between let say Delhi to Mumbai. So, assume there is a road between Delhi and Mumbai. So, it will take you equal number e it will take you equal time to go from Delhi to Mumbai and from Mumbai to Delhi. So, hence we can have one edge and the graph is undirected. So, here we are going to make an undirected graph G equals to nx dot graph. So, when you write G equals to nx dot graph it automatically creates an undirected graph for you.

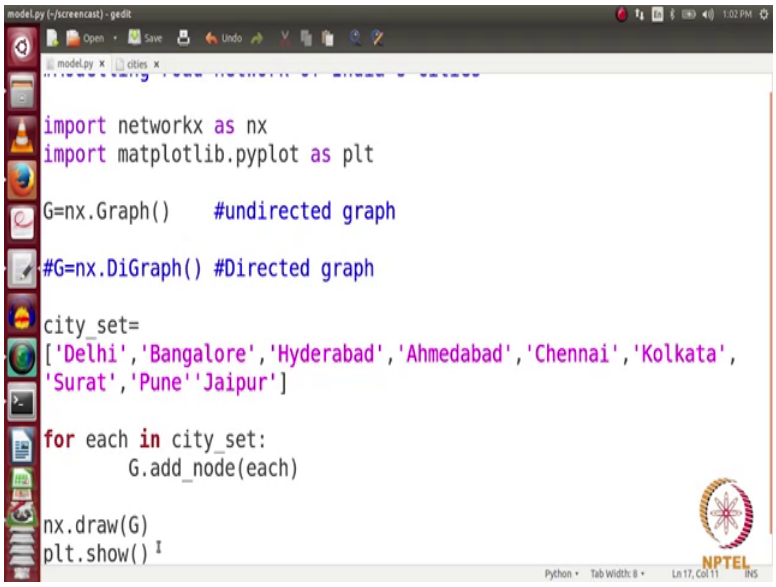
So, let me tell you there if the possibility of making directed graphs as well. So, a simple code by which you can make a directed graph is nx dot digraph. So, if you use this command and next of digraph there will be a directed network created. So, it will be a directed graph here we are working with undirected graph. So, just come into this statement here. So, we have made a graph G equals to nx dot graph here now I have a graph what I want to do is I want to add a nodes to this graph and every node should represent a city.

So, let us have an array of city here let say city set which is the set of cities where from where we will be choosing the cities. So, I have a file here just I looked at Wikipedia and looked at some of the cities let us pick some cities form here let us say, let us pick all these cities control c and let say I will put all these cities here. So, this city may insert strings. So, one thing well we are working with strings you will have to put these inputs.

So, I will put all of the inputs. So, I have a set of cities here. So, this is the set of cities now we want to create nodes in this graph which should be the cities.

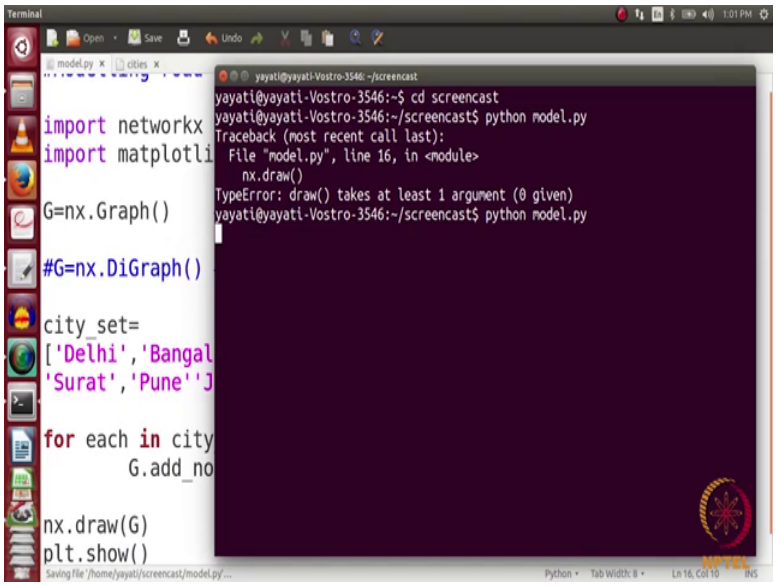
So, what we can do is. So, we have a list here we have to create cities and the values of the cities from should be from this list what we can simply do is for each in city set. So, if you would have gone through the module of using libs in python you would know that each is an iterater which will go through each of the elements of these lists cities set one by one. So, for each in city set what we want to do is we want to add a node this network. So, we do G dot add node which is the command for adding node and each. So, we have these nodes and getting the network now I want to visualize this network we know that the patent which is used for visualization nx dot draw and for visualizing we need to import one more module here draw import matplotlib dot pyplot as plt.

(Refer Slide Time: 06:46)



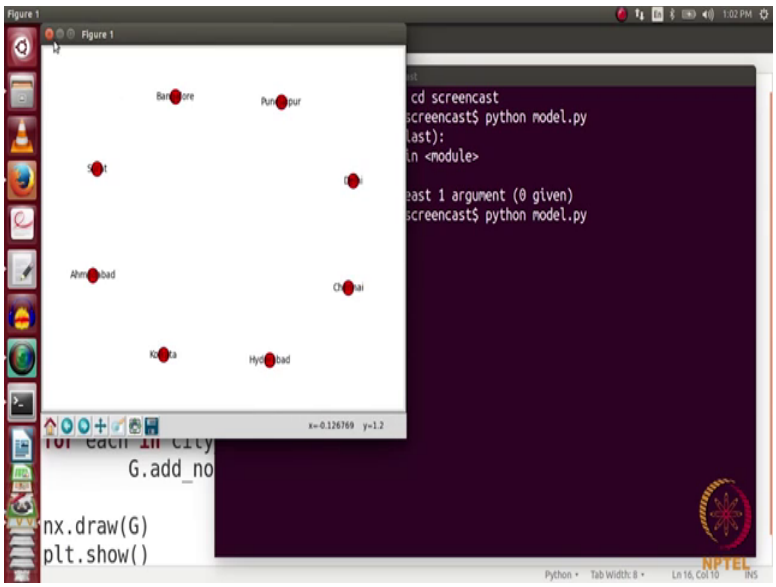
So, what we can do here is nx dot draw and then plt dot show. So, let us run this and let us see how does it work? So, I open up terminal window here.

(Refer Slide Time: 07:00)



So, I will mainly we will be in this terminal window here. So, we are in the home. So, we will first go to this folder, whose name was screen cast and then we run this python; python model dot py some problem. So, it says that the draw function takes at least 1 argument. So, when we are nx dot draw we need to pass the graph here which is G and then run it.

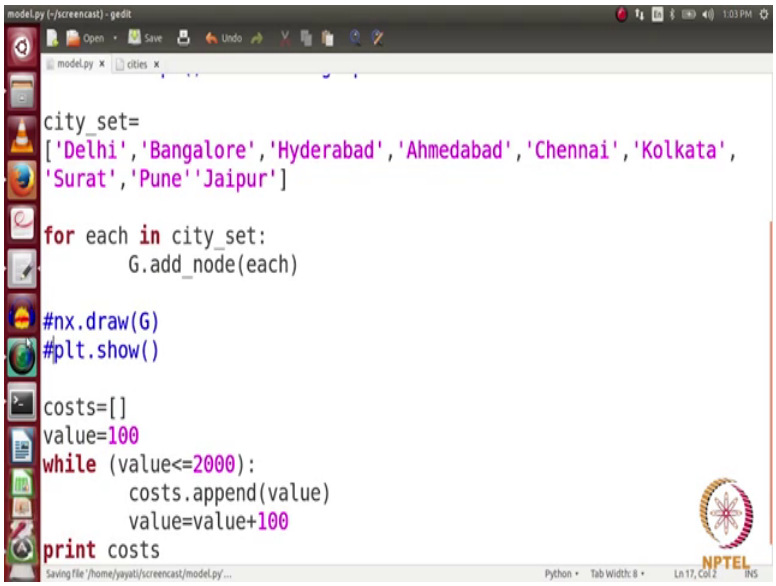
(Refer Slide Time: 07:35)



So you can here; that we have a network created where each node is one city in this network.

So, we added 1, 2, 3, 4, 5, 6, 7, 8 cities, we have added in this network perfect. So, now, we have a graph in which there are cities, there are 8 cities and next n is to add edges between these cities of roads between these cities and every road should have a have travelling cost associated with it. So, this travelling cost we represent it by the weight of an edge. So, we have to add edges here. So, how do we add edges it is the question. So, we are going to add these edges randomly for now how do we do that.

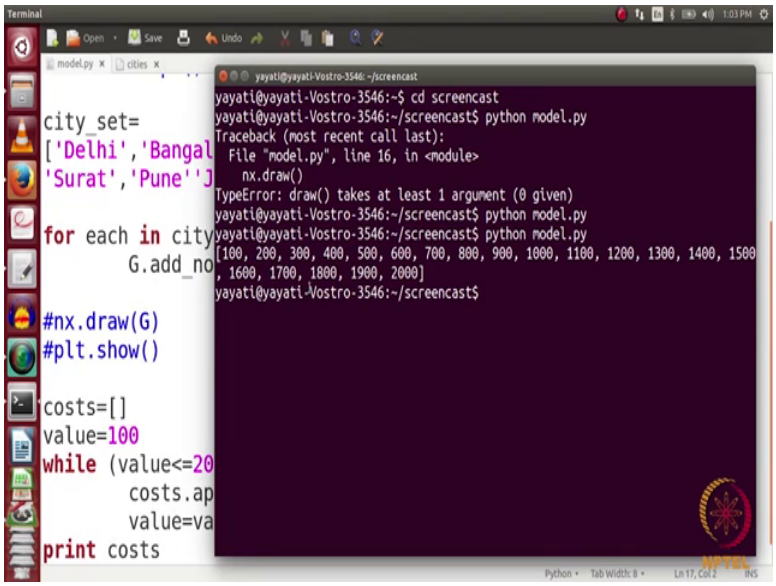
(Refer Slide Time: 08:15)



So, first of all we have a let say array costs what we are going to do is we are going to add some random edges in this network and each edge will have a random value random weight random travelling cost which will be picked from this lift cost. So, we want to first of all populate this list costs so that we can pick values form here. So, what do we do? Let us say I will add some 20 values here let say starting from 100. So, what I will be doing while value is less than or equals to 2000 costs dot append and I will do here value and then value equals to value plus 1. So, it will iterate like 100 sorry it will iterate like 100, 200, 300, 400, it will keep up entering the value and it will go up to 2000.

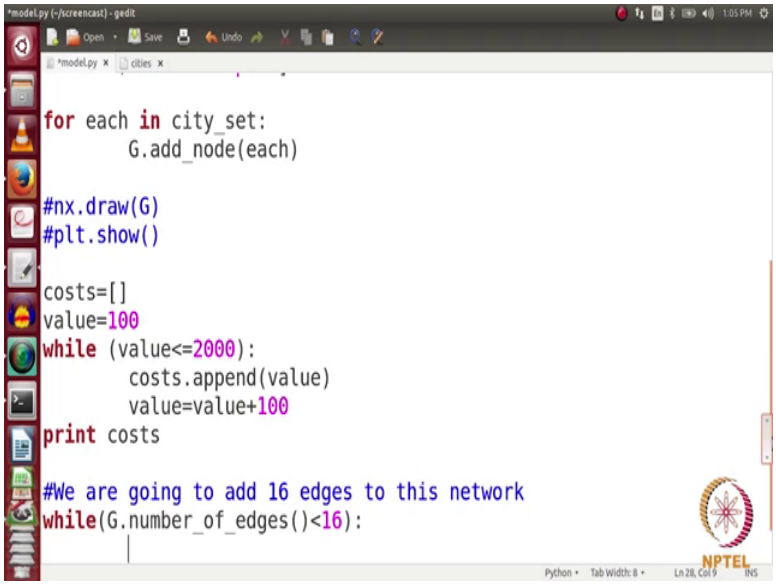
Let us see what this array looks like and we will just comment these 2 things.

(Refer Slide Time: 09:34)



So, this is how our array will look like it has values from 100 to 2000 these are array costs. So, now, we are going to add edges in this network assume that we want to add. So, there are 8 cities. So, for now let us add some 16 edges. So, we are going to add 16 edges through this network.

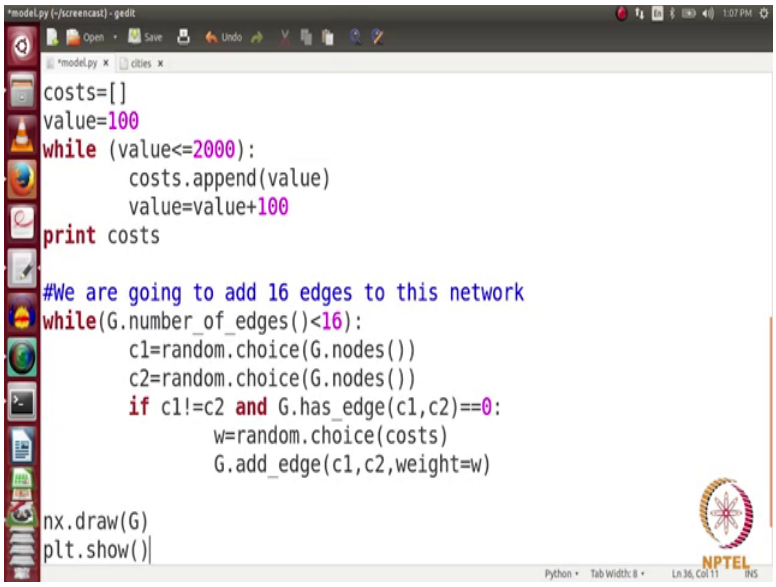
(Refer Slide Time: 09:56)



So, count is in English. So, we are going to add 16 edges to this network. So, till when your port should run while. So, for looking at the number of edges you have this code G dot number of edges is less than 16.

So, this will this code will keep running till the number of edges in your network become 16 what you are going to do is we have choose 2 nodes randomly. So, for choosing a node randomly we note we need to import the function random. So, we import the function random here import random so that we can use it.

(Refer Slide Time: 10:58)



We come down. So, we choose city one equals to. So, if you want to choose a value from a list randomly. So, we have random dot choice and we have function G dot nodes. So, G dot nodes give you a set which has already nodes in these networks. So, we are going to choose one of these nodes randomly that is their city 1 and then we choose city 2 which is again random dot choice and again G dot nodes.

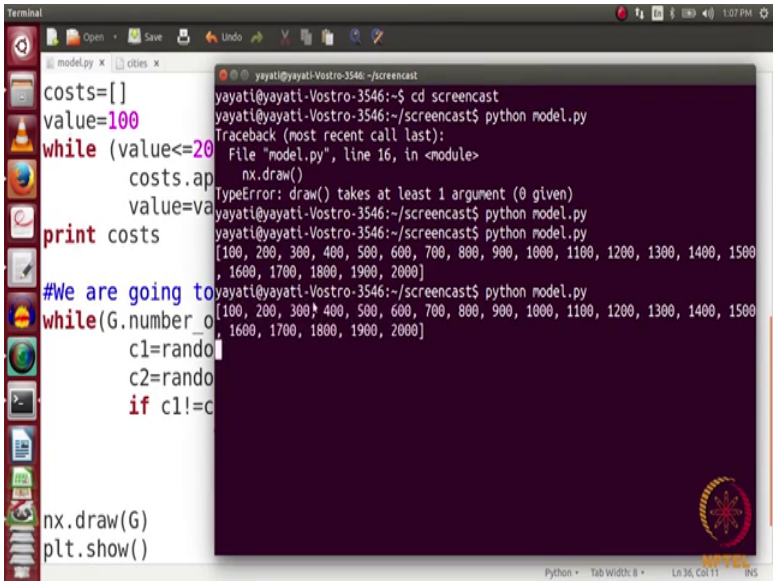
Now, one thing to note here is these 2 cities should not be same. So, only if c 1 is not equals to c 2, we are going to move next what we are going to do is we are going to choose a weight for the edge. So, let say weight which is the travelling cost now this is nothing, but random dot choice again a random function and it will pick a random value from our array costs also. So, the one condition for creating an edges both of these cities should not be equal and the second condition is there should not be an edge which is already present between these 2 nodes.

So, for that we have function and G dot has underscore edge it tells you whether there is an edge between 2 nodes c 1 comma c 2. So, has edge c 1 comma c 2 should be equal to 0 there should be no edge between them. So, in that case we will choose a weight here w

equals to random dot choice costs and then we will add an edge in this network G dot add edge and we add edge from c 1 comma c 2 and the weight of this edge is w which was a random number. So, this code keeps running this loop keeps running till we have 16 edges in the network.

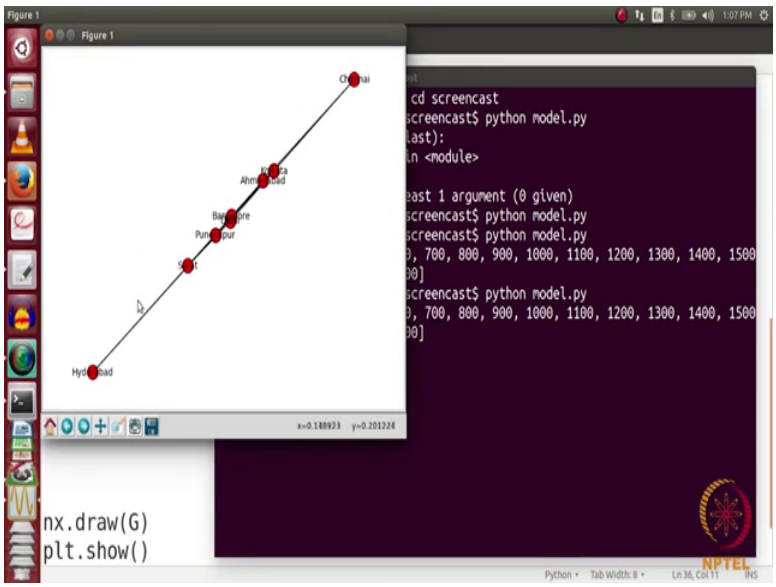
Again I want to see these networks. So, I use nx dot draw G and then plt dot show.

(Refer Slide Time: 13:06)



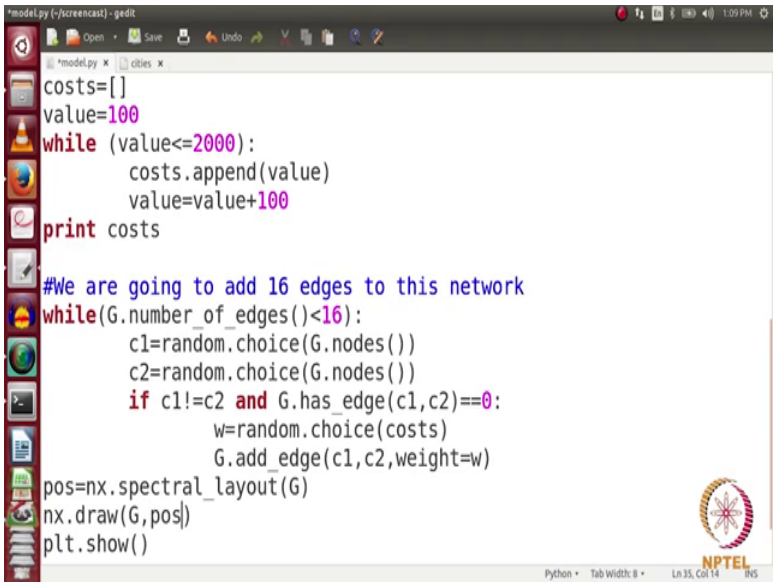
Let us execute this in c.

(Refer Slide Time: 13:09)



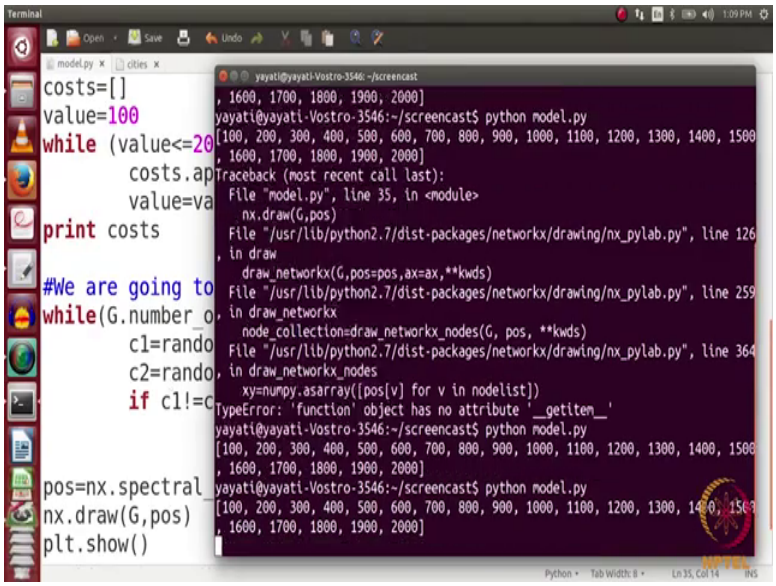
So, you see here. So, the edges are not very clear some of the edges are overlapping here. So, what we can do for this is we can change the layout of this graph. So, one of the layout that we can use is the spectral layout. So, for using this spectral layout we have a command.

(Refer Slide Time: 13:40)



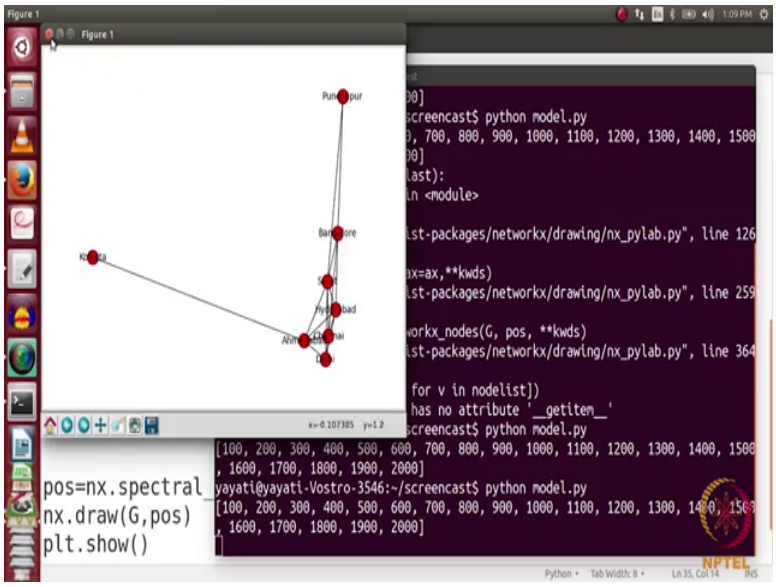
So, let us say we want to choose a layout pos equals to nx dot draw underscore spectral (Refer Time: 13:52) this p o s here (Refer Time: 13:54). So, the command here is nx dot and we want a spectral layout. So, we using spectral underscore layout we will pass a graph G here and while drawing it we use nx dot draw pos.

(Refer Slide Time: 14:20)



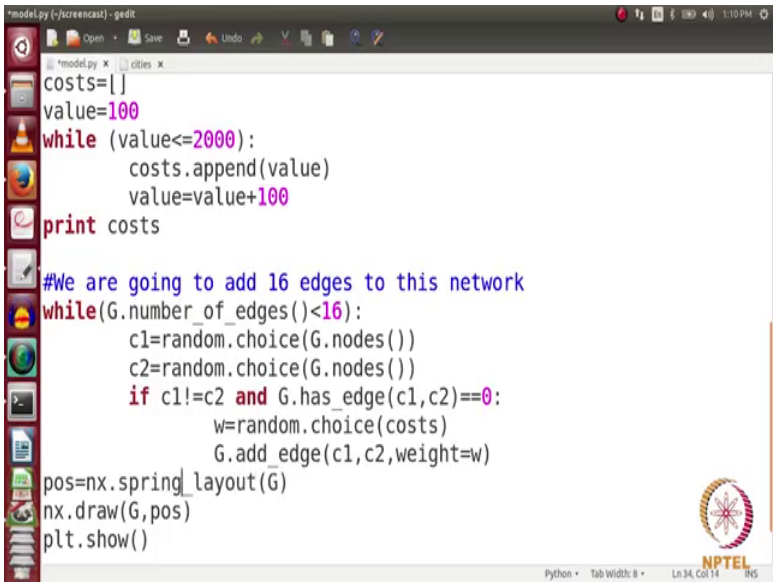
So, let us now visualize these graph python model dot py.

(Refer Slide Time: 14:22)



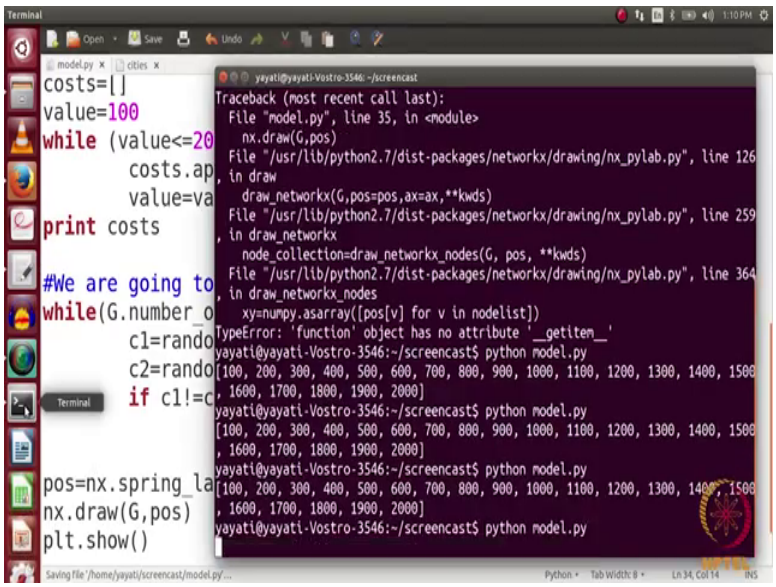
So, when we see it looks better than before ok we can also use some other layouts.

(Refer Slide Time: 14:35)

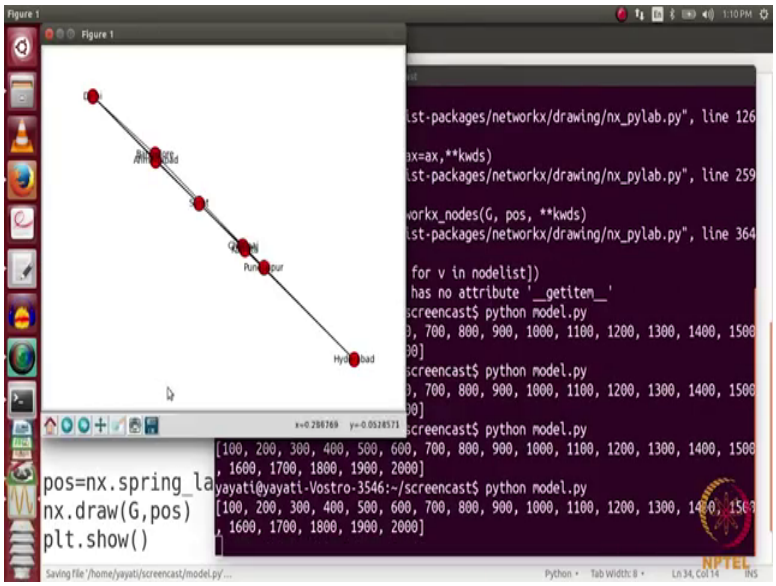


Let us say; let us try to use a spring layout look very nice and circular layout might look better here.

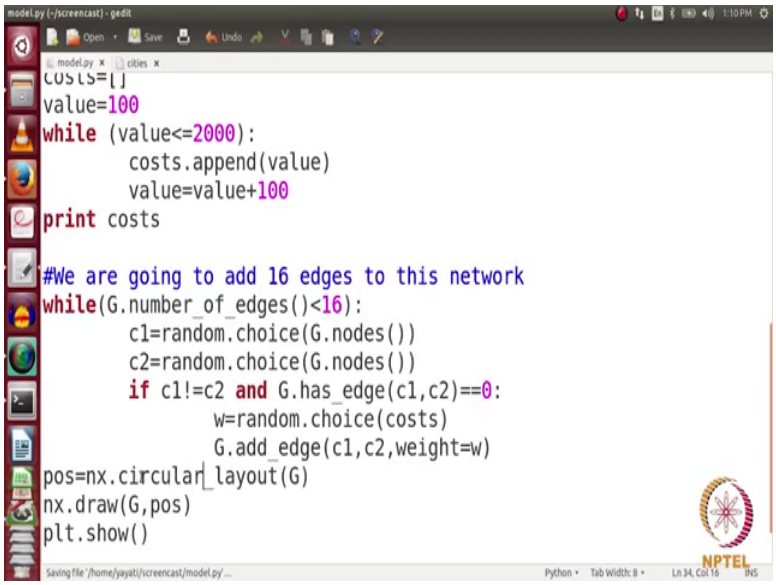
(Refer Slide Time: 14:37)



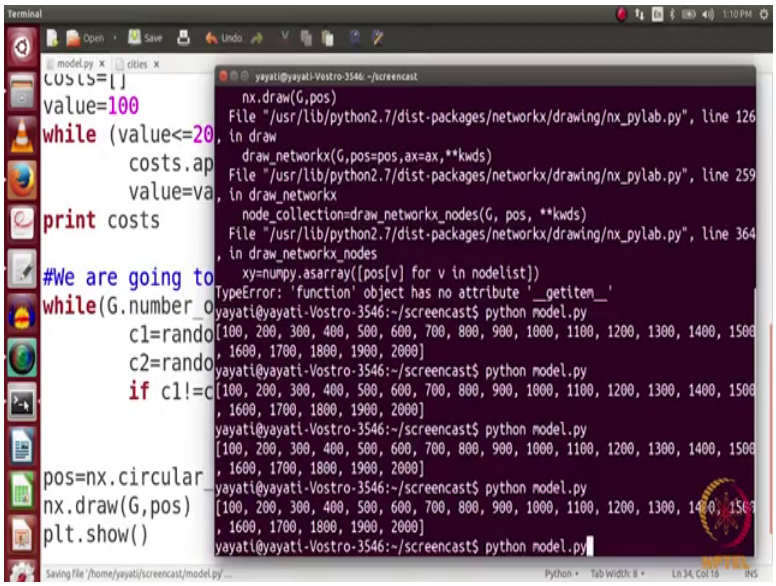
(Refer Slide Time: 14:37)



(Refer Slide Time: 14:43)

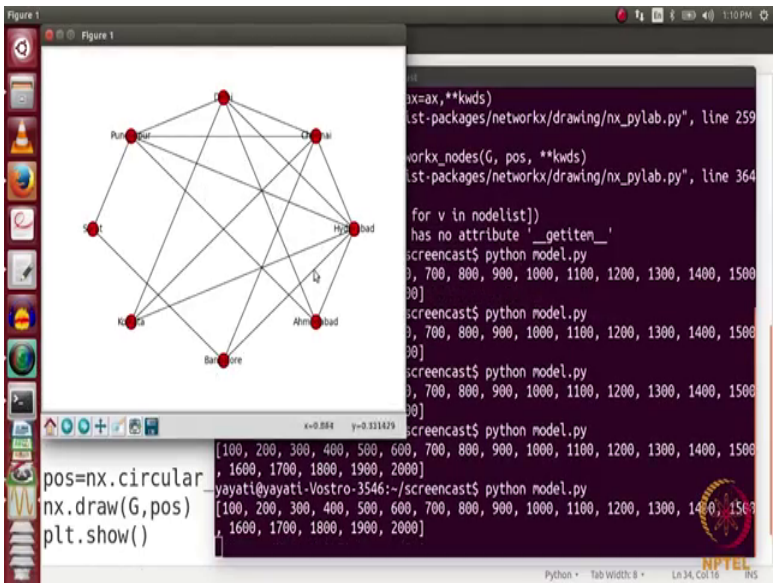


(Refer Slide Time: 14:45)



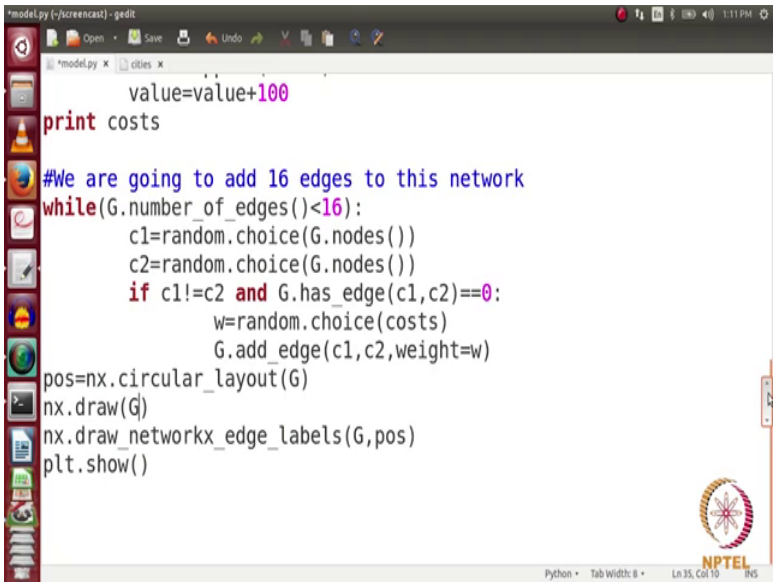
Let us try the circular layout. So, this one looks perfect. So, we have drawn this graph in the circular layout.

(Refer Slide Time: 14:46)



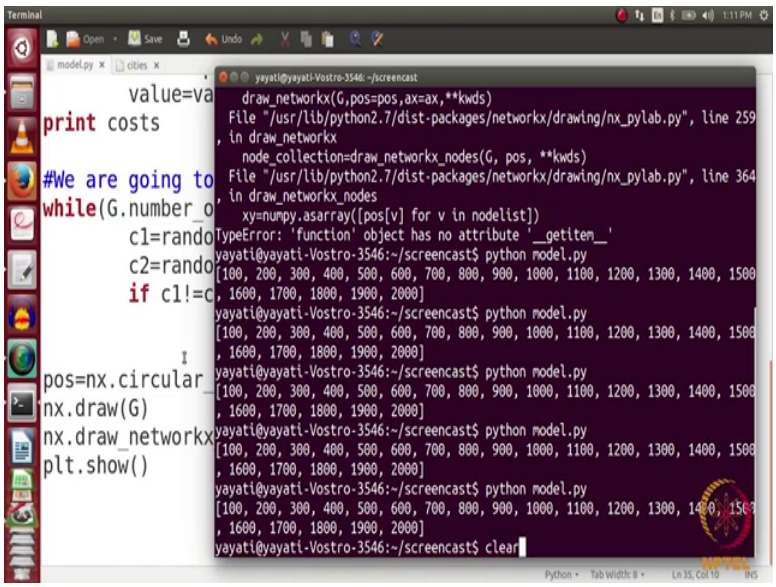
And it looks nice here. So, there are 8 cities and they have 16 edges and each of this edge has some edge weight associated with it. So, you might want to look at this edge weights as well. So, for labeling these edges with their edge weights, so here we have drawn the networkx and let us draw their edge weights are also here.

(Refer Slide Time: 15:11)



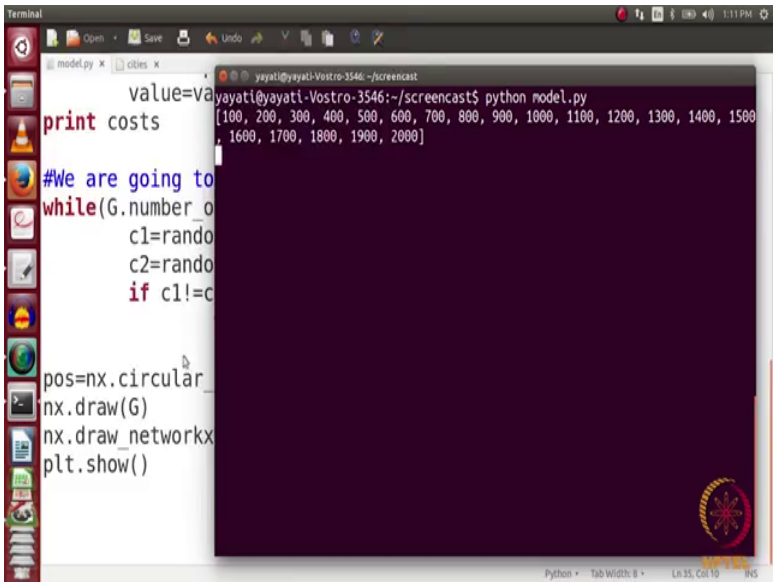
So, we have this command nx dot draw underscore networkx underscore edge underscore labels and then we pass here G comma p o s.

(Refer Slide Time: 15:32)



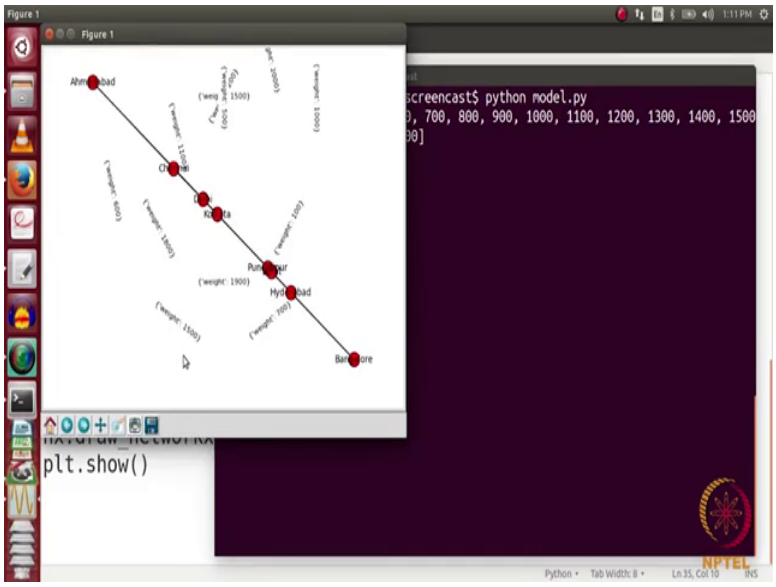
Let us see how does it work and then let me just clear it.

(Refer Slide Time: 15:36)

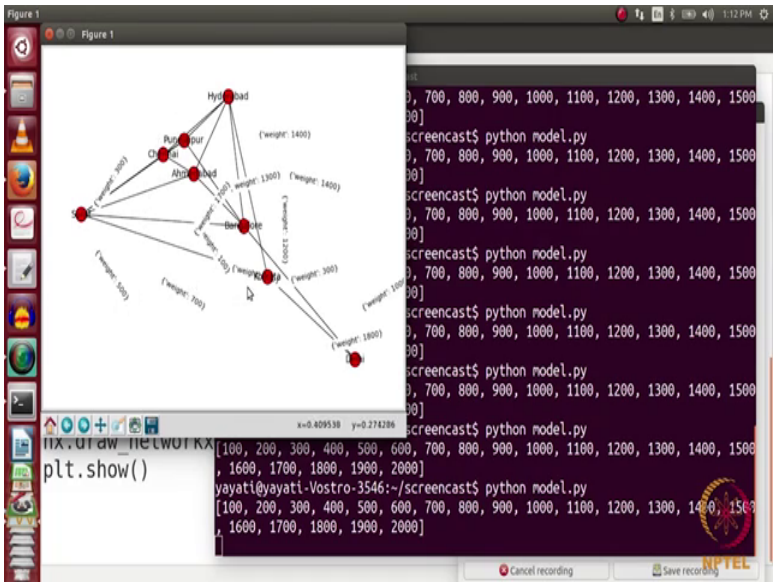


And again we run python model dot py.

(Refer Slide Time: 15:37)



(Refer Slide Time: 15:51)

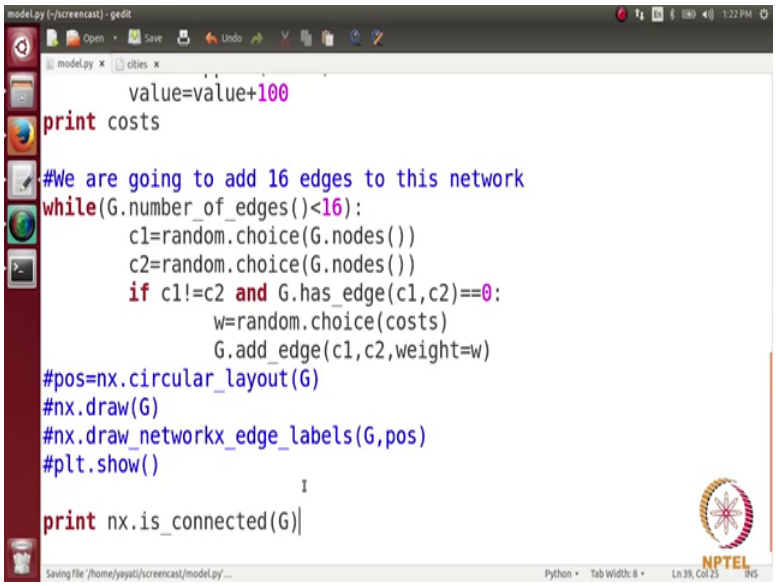


So, you see here, it has to be tried some number of times and when you draw it in a circular layout it turns out to be a pretty nice configuration here no its not very clear, but yeah still you can see the edge weights here. So, now, we have a network which has some 8 nodes and 16 edges and we have visualize this network as well we have drawn these edges randomly. So, we have drawn 16 edges randomly between random pair of nodes.

So, we have now created the network with the cities and the roads what we are next interested in his looking at the paths between these cities and the cost of travelling across that paths as well. So, first let us look at the path between these cities we want to look at the path length between every 2 possible pair of cities and by path length what I mean here is now the some of the weights of the path just to see whether I can reach from city a to city b or not. So, first of all let us that whether even this graph is connected or not because of the graph is connected it means that I can reach from every city to every other city and so on.

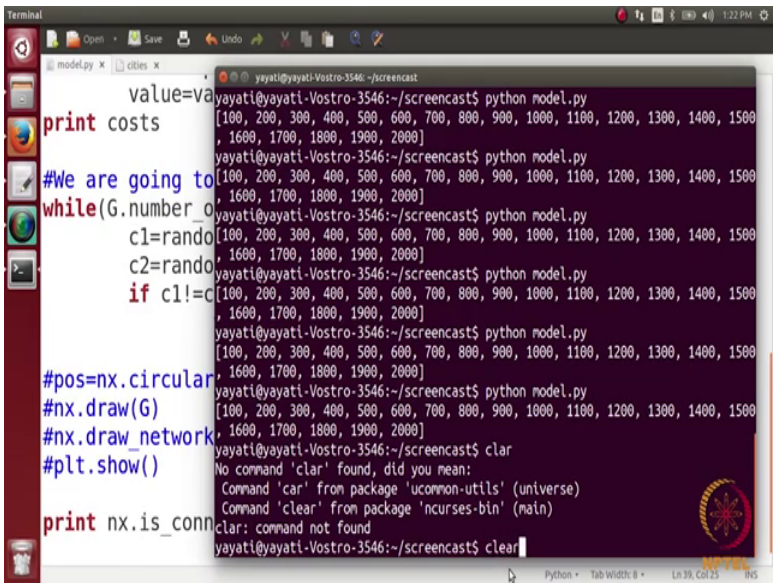
So, let us look at whether this graph is connected or not.

(Refer Slide Time: 17:14)



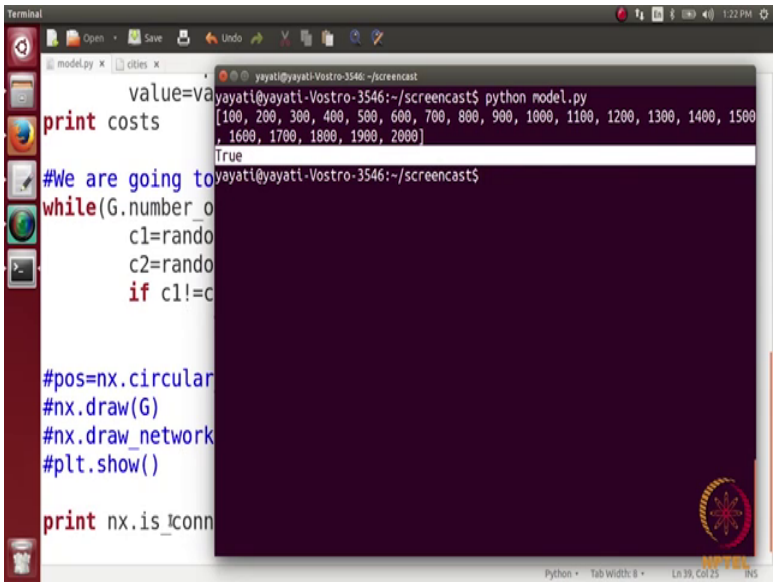
So, we will just print, so we have a command nx dot is connected G which tells us whether this graph is connected or not.

(Refer Slide Time: 17:38)



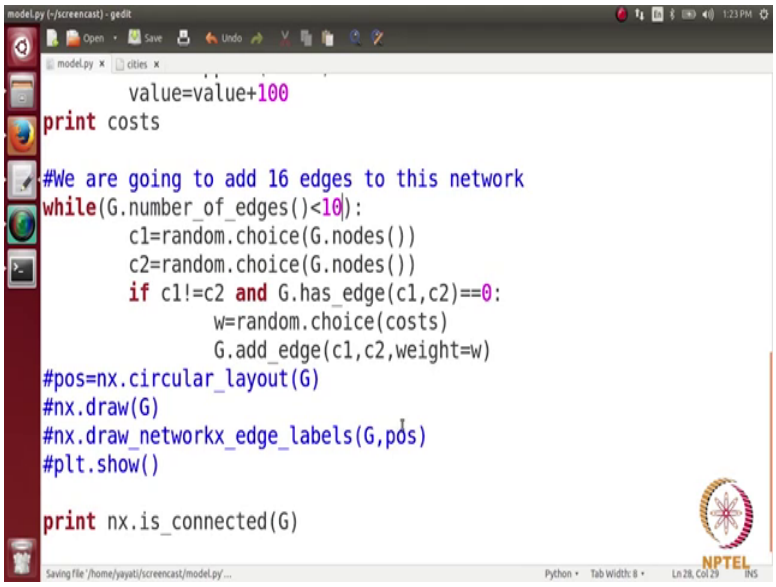
So, let us see, just clear it.

(Refer Slide Time: 17:40)



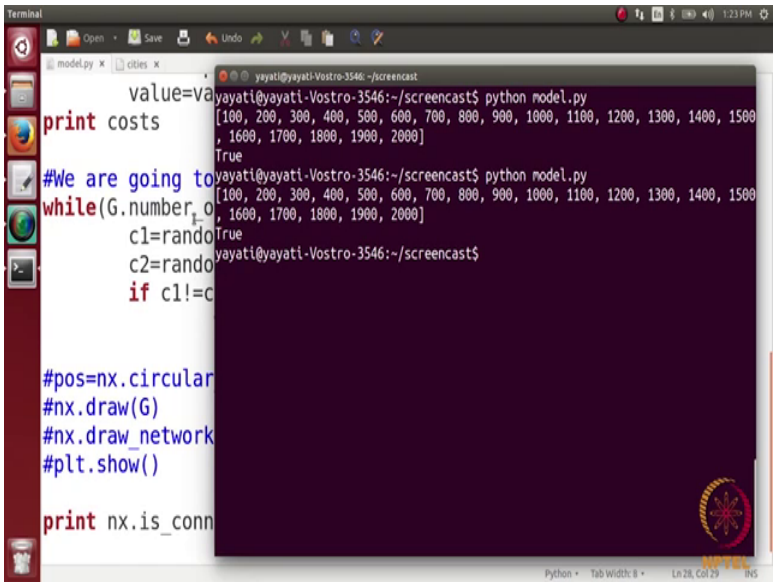
See here at the graph is connected it means that there is path between every 2 pair of nodes. So, this, so, what we want is we want to delete some edges let us say. So, that we do not have some paths between some pair of cities.

(Refer Slide Time: 18:07)



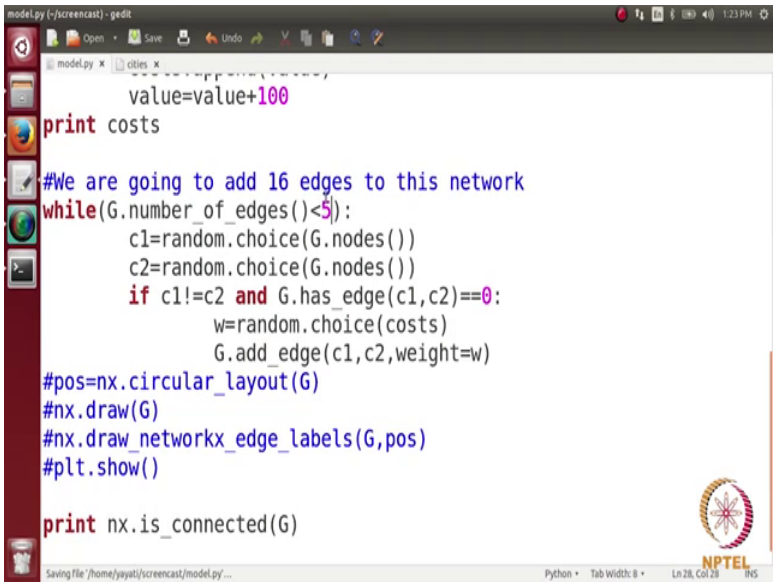
So, what we can do for that is if we can do for doing that is let us reduce the number of edges here from 16 to let say that we will just have 10 edges in this network and let see what happens in that case.

(Refer Slide Time: 18:20)



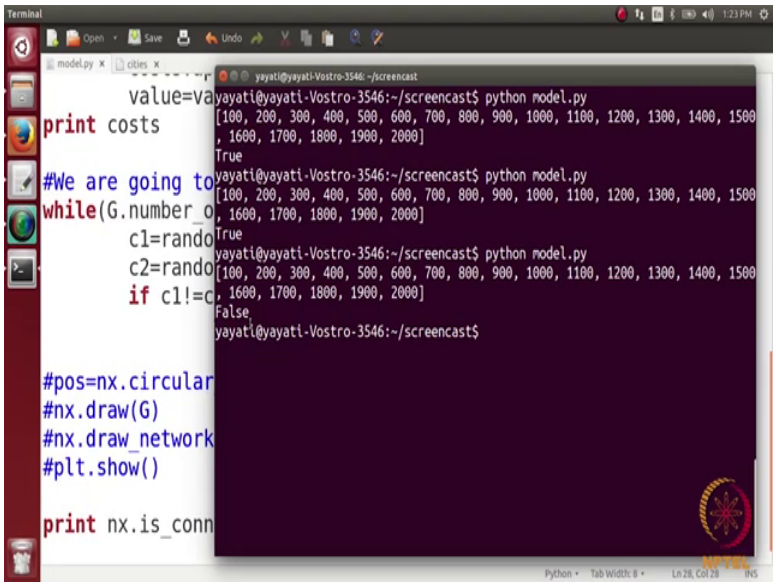
So, we have done edges again the graph is connected. So, it is actually interesting you see it is a network on 8 nodes and we have put just 10 edges even then this turning out be connected.

(Refer Slide Time: 18:34)



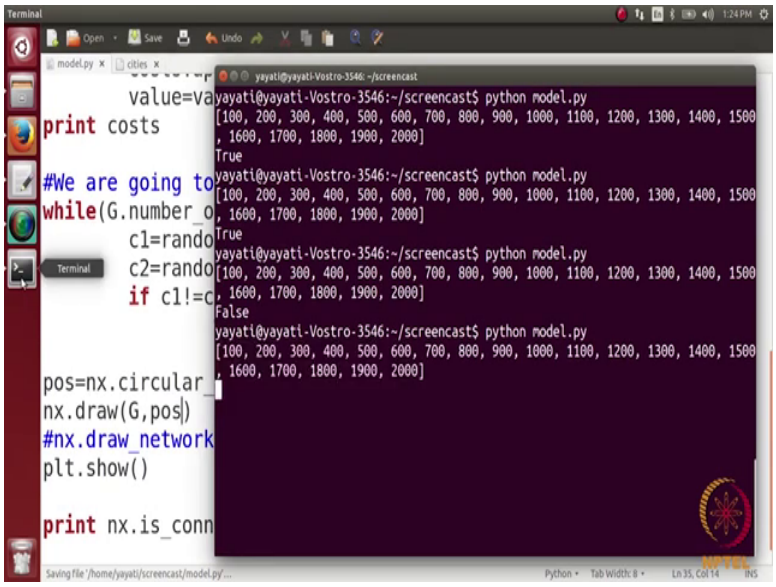
Let us reduce it to 5, let say and then when we put the number of edges be 5, it turns out that the network is not connected. So, it is false network is not connected.

(Refer Slide Time: 18:41)

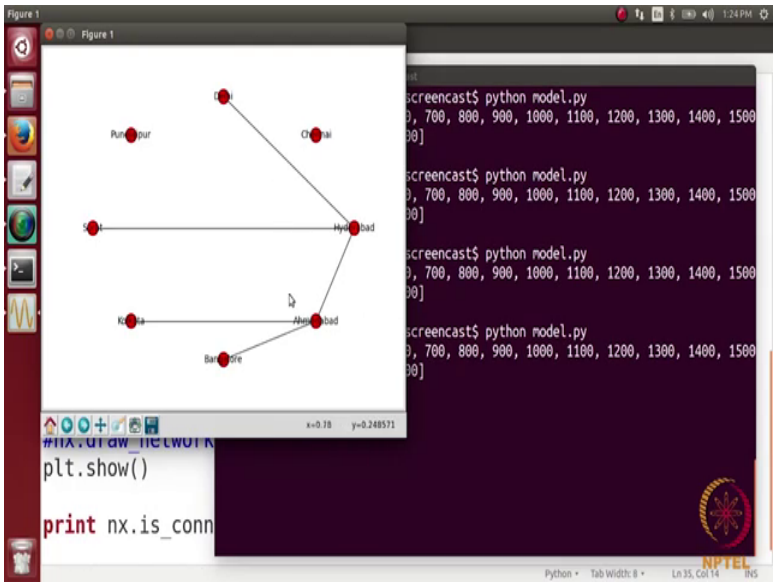


So, let us look at this network as well.

(Refer Slide Time: 18:54)

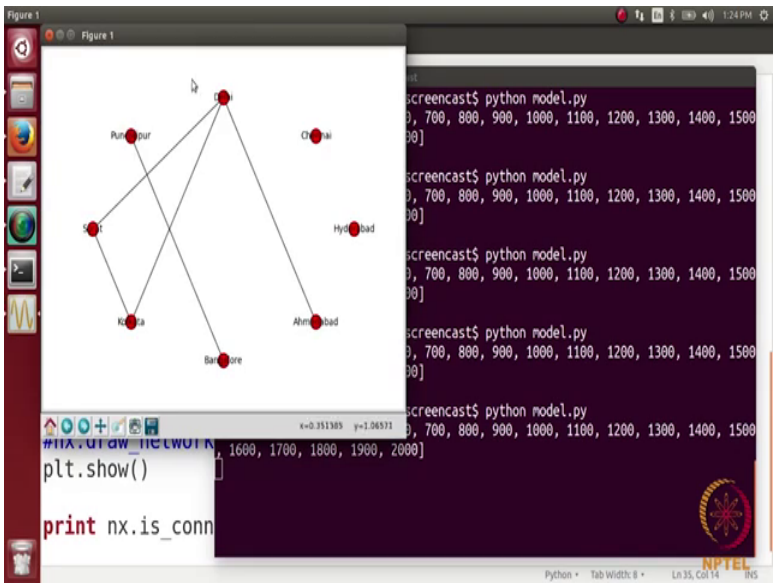


(Refer Slide Time: 18:58)



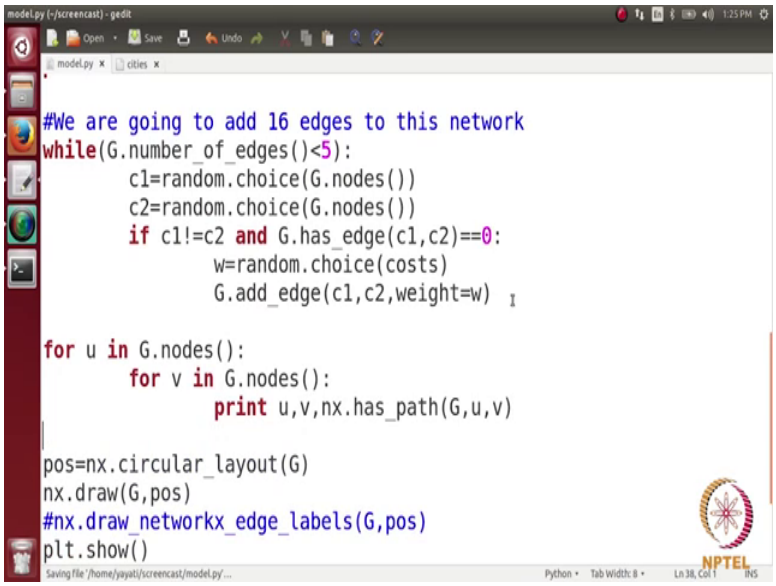
Now, you see the network here right. So, this there is this one city which is not these are which are not connected at all to any of these cities.

(Refer Slide Time: 19:06)



So, let us take one more example. So, it can be like this and so whom, so what is now we want to see assume that we want to see path between which cities path exists and between which cities path does not exist.

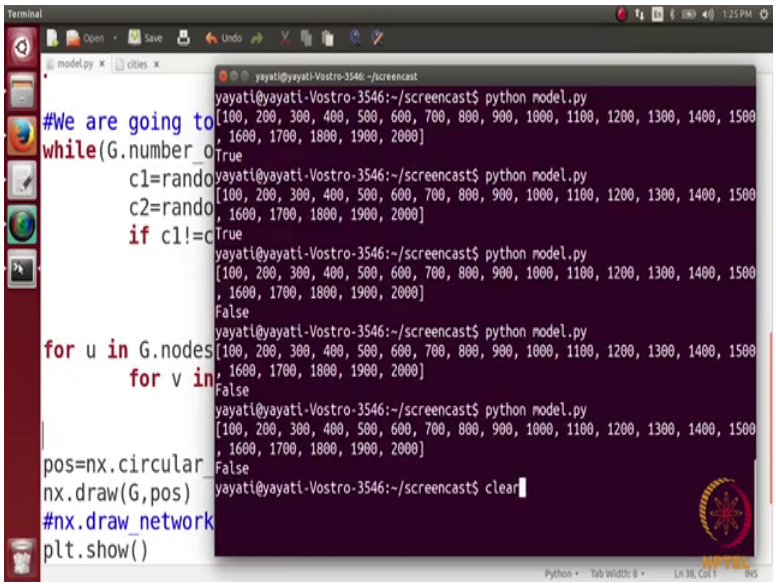
(Refer Slide Time: 19:23)



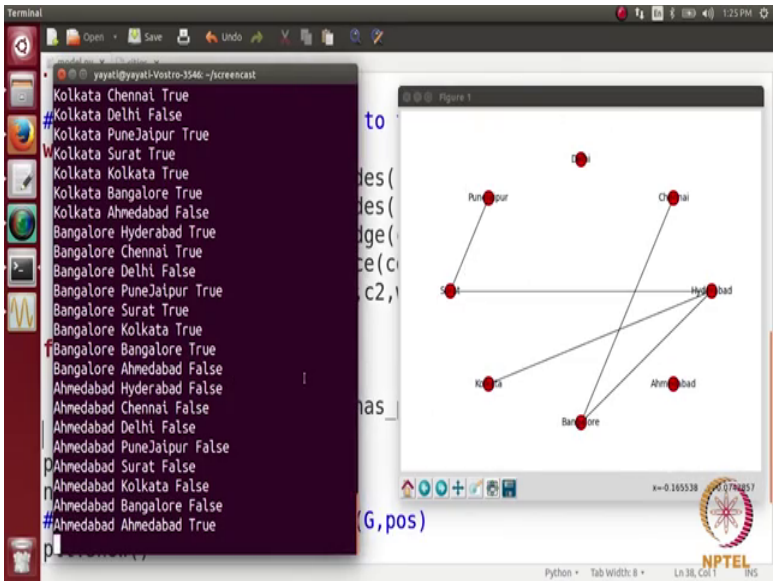
So, what we do for that is for each possible pair of cities we see that whether there is a path between them or not. So, we have a loop here for let us say u in G dot nodes and then for v in G dot nodes what do you do is print u v comma and then we have a function here nx dot has underscore path which tells us whether there is a path between these cities or not.

So, we pass here it G u comma v right. So, what I will do is I will put this code before drawing the networks so that we can see both the things simultaneously and let see now just in.

(Refer Slide Time: 20:13)



(Refer Slide Time: 20:18)

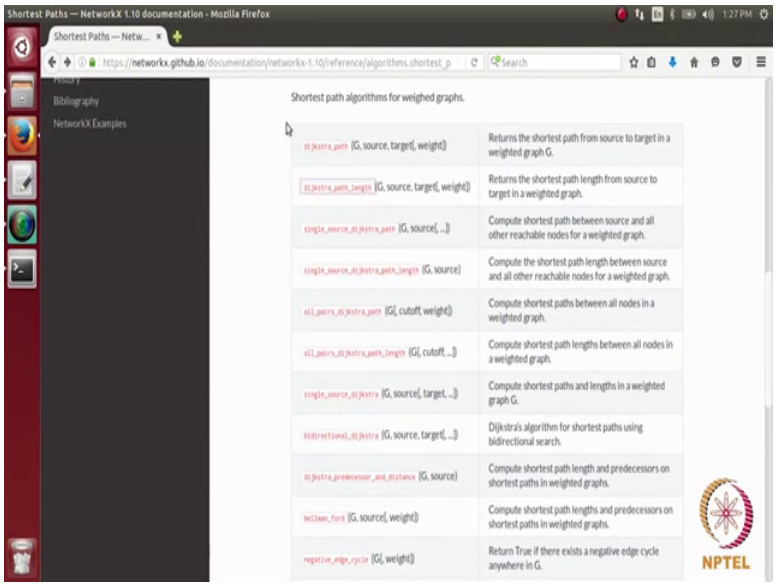


So, you see here. So, this is our network and what is. So, perfect. So, this is our network here and these are the paths. So, it tells you for every possible pair of cities. So, Hyderabad to Hyderabad is; obviously, path is there Hyderabad to Chennai there is a path Hyderabad to Delhi there is no path. So, you see here from let say Hyderabad to Pune and we look from let say Hyderabad to Pune I am very sorry we have mixed within Pune and Jaipur. So, let us say its Pune.

So, from Hyderabad to Pune we looked at there is a path. So, you can go from Hyderabad to Surat and from Surat you can go to Pune. So, this is a fictitious network if the connections might not be really this way it just a fictitious. So, do not look at it in real world then how come Hyderabad is connecting to Surat and then to Pune and so on. So, there is a path from this node Hyderabad to a node here. So, we have listed it for every 2 pair of nodes. So, you can looked at; then from Chennai we can go to Hyderabad to let say Bangalore. So, again you see here if you look the connection between let say Chennai and Hyderabad which is somewhere here yeah. So, it turns out be true and from Delhi you see from Delhi you cannot go anywhere according to these networks. So, every value from Delhi turns out to be false and so on, just close it, stand here.

Now, next what we want to look at is what are the shortest paths in this network. So, we have some really (Refer Time: 22:25) functions in networkx. So, which are the basic standard algorithms implemented in networkx with the help of which you can see the path length of the travelling cost between 2 cities.

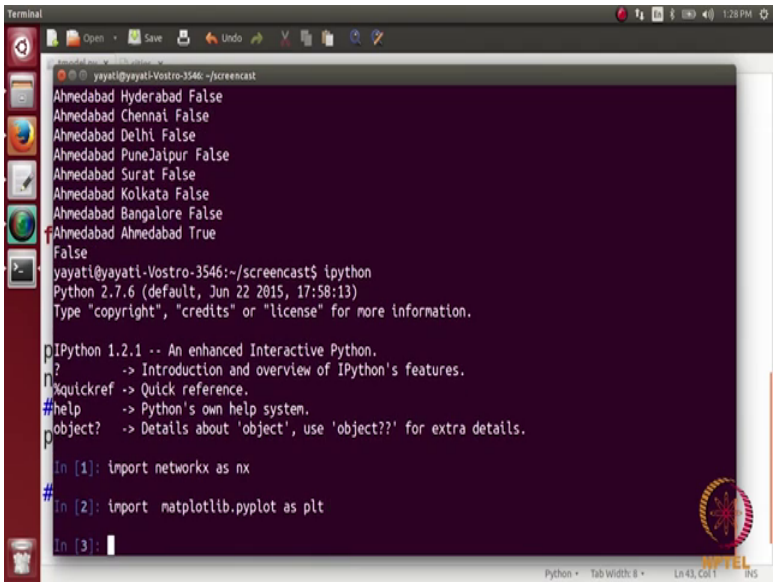
(Refer Slide Time: 22:41)



So, let me quickly show you some of these. So, these are here. So, you can see that these are the functions. So, we have one function discussed standards discussed standards cold path which tells you the shortest paths from source to target and then underscore path length which. So, given 2 nodes it tells you the length of that shortest path then single source discussed a path where you give one source node and it gives you the all the

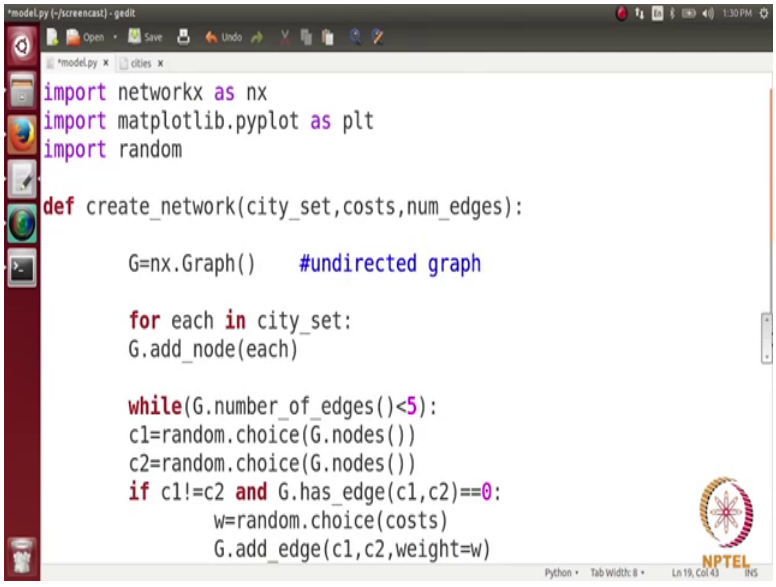
shortest paths and so on. So, we will we are going to use some of these functions and see how do they work in our network? So, for working with this network what will be going to do is let us now switch a little bit to the interactive ipython.

(Refer Slide Time: 23:30)



So, what we are going to do is we are going to turn to ipython and then we are going to import network ipython then we are going to import networkx as nx then import matplotli dot py pyplot as plt perfect. So, we are going to define some functions here. So, let us con let us make this code a little bit of deny stand make some functions from this.

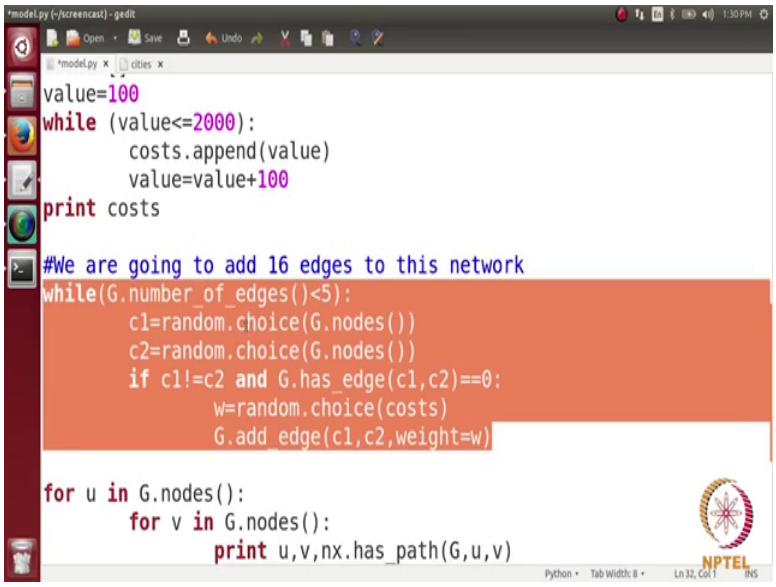
(Refer Slide Time: 23:59)



So, let me define a function fine create network and let say this function takes us input the set of cities the array costs and let say number of edges which you want to create.

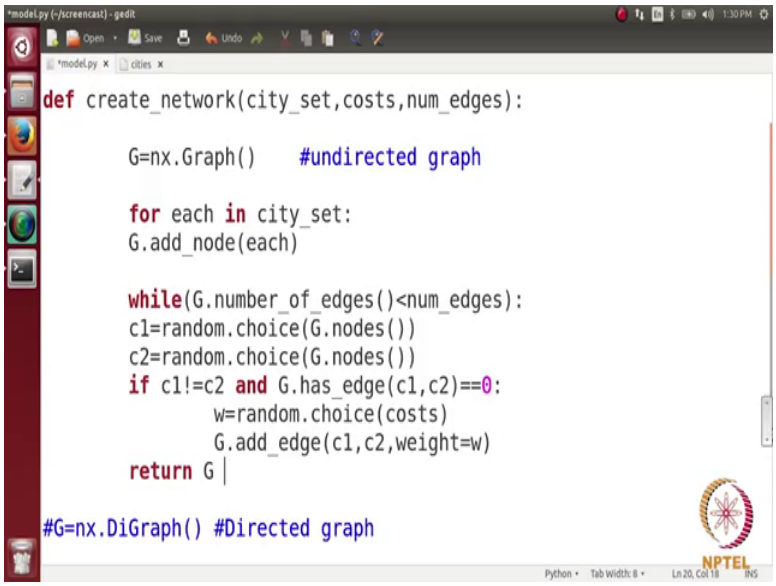
So, what all we will have this inside this function is this term will come here. So, G equals to nx dot graph and then for each in city set we are going to add an edge. So, this particular code comes here for each in city set we are going to add a node and then what else.

(Refer Slide Time: 24:54)

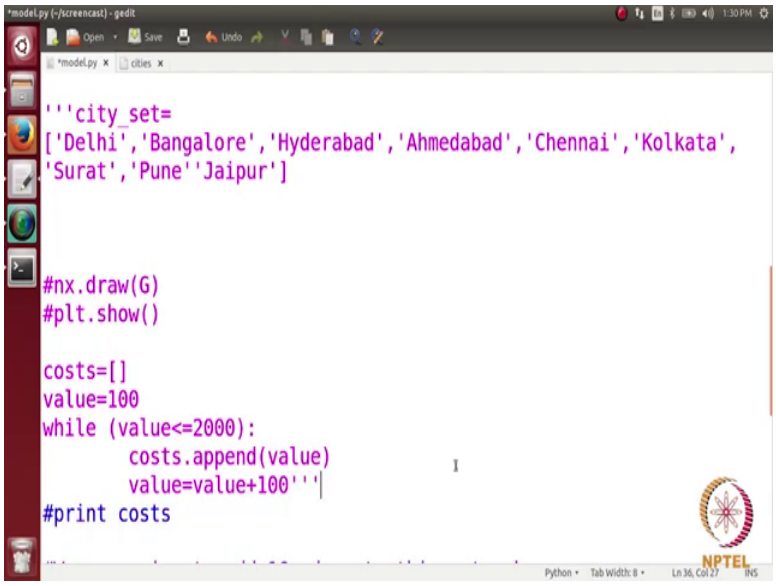


So, this entire code will be shifted there. So, this is when we are we were adding the, we were adding 5 nodes in this networks. So, will basically change this code first of all let us paste it here. So, why 0 number of edges is less than; so we are adding num edges number of edges. So, let me make it num edges here and then of right and this will return you this graph G which you can then visualize.

(Refer Slide Time: 25:31)

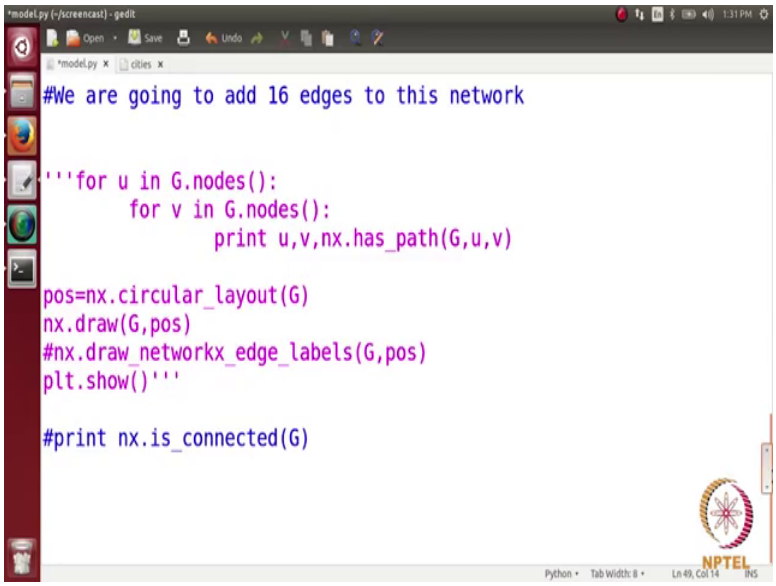


(Refer Slide Time: 25:48)



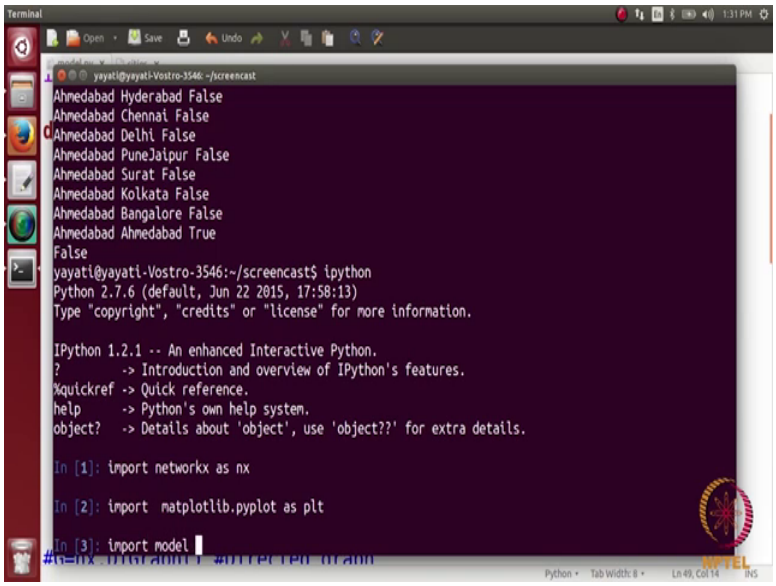
Next comment rest of are go further mean while write commented rest of the codes. So, everything is most and mostly commented here.

(Refer Slide Time: 25:57)



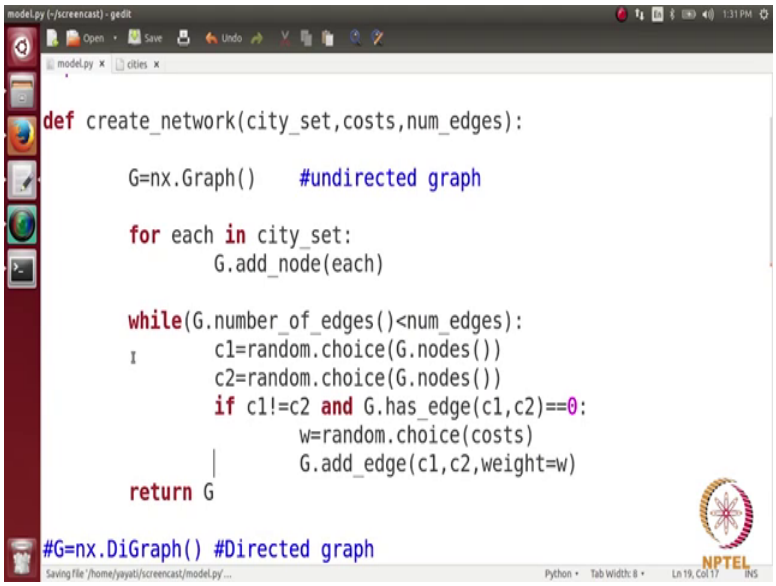
Everything is commented except for this function here. So, this function is simply what it is doing taking a city set taking array costs taking the number of edges and creating the graph.

(Refer Slide Time: 26:18)



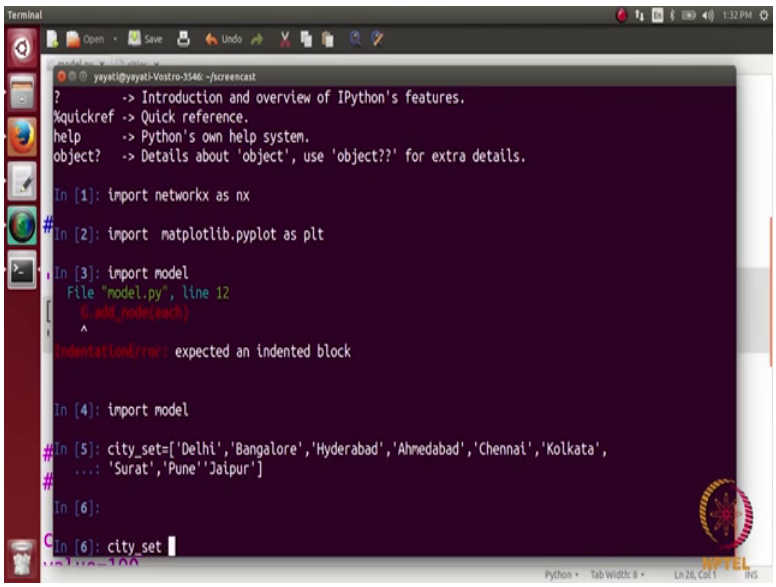
So, how do we do it here? So, we are going to import this model here. So, when added it line 12. So, this is the 4 loop.

(Refer Slide Time: 26:30)



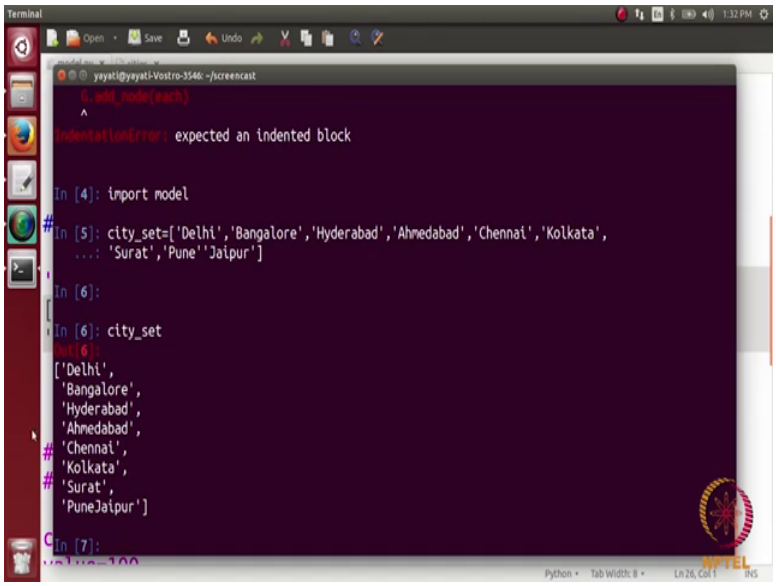
So, here you have to always take care of the indentation and let us import model and we have imported a.

(Refer Slide Time: 26:40)



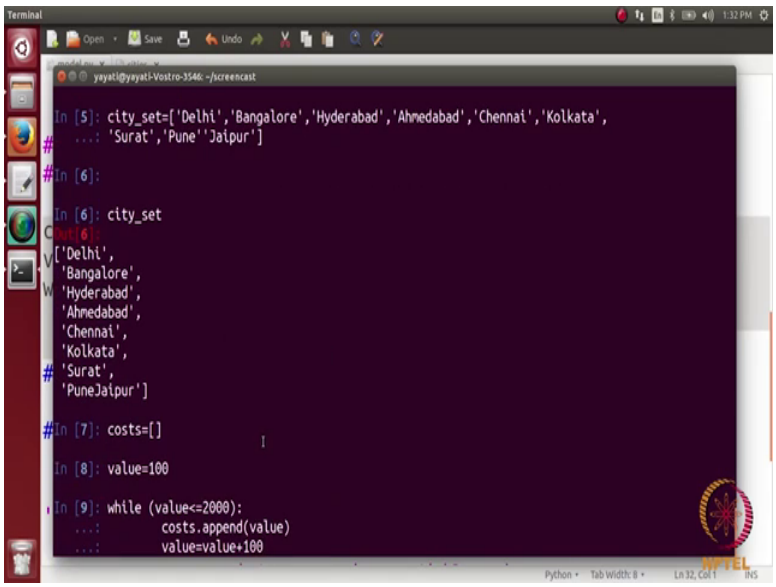
So, to call this function we should have some parameters which are the city set the costs and the number of edges. So, let us define these parameters here. So, I will just copy paste the set of cities here. I have a array here for the city set we can also see this array here.

(Refer Slide Time: 27:03)



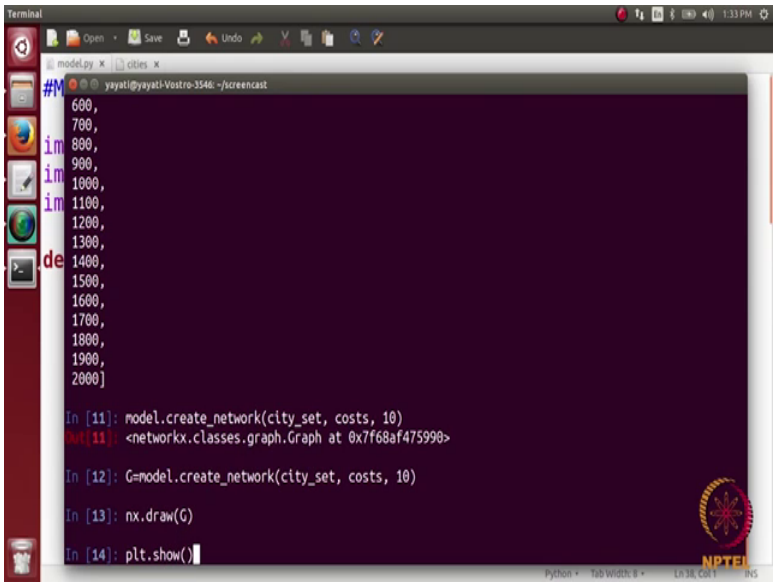
So, we have this city set and then we want these costs to be here. So, in will just take this code yes paste it here.

(Refer Slide Time: 27:16)



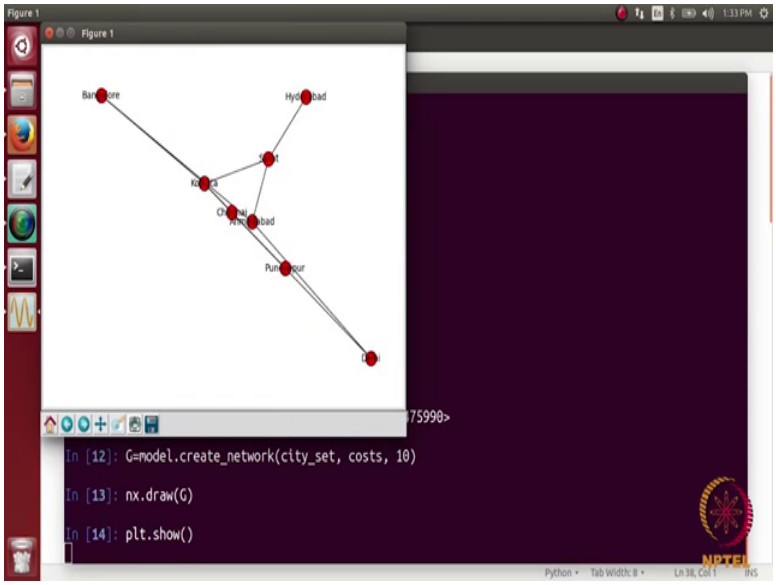
Let us look at the array cost.

(Refer Slide Time: 27:20)



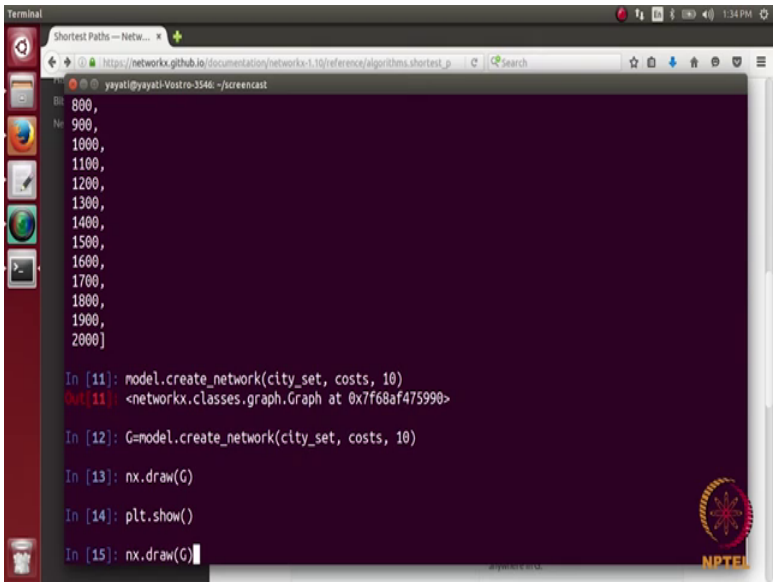
So, we have this array costs here and let say I am going to add some 10 edges. So, let us add 10 edges. So, we are going to call this function and the function name is create underscore network. So, what we are going to do is model dot create underscore network and here we are going to pass the 3 parameters. So, what are a parameters one is the cost city set young. So, we are going to pass here city underscore set and then say costs and then we are going to at 10 edges in this network and. So, we have took collect this network in a variable let say this variable is g. So, G equals to this. So, let us try looking at this graph, so let say nx dot draw G and then plt dot show.

(Refer Slide Time: 28:24)



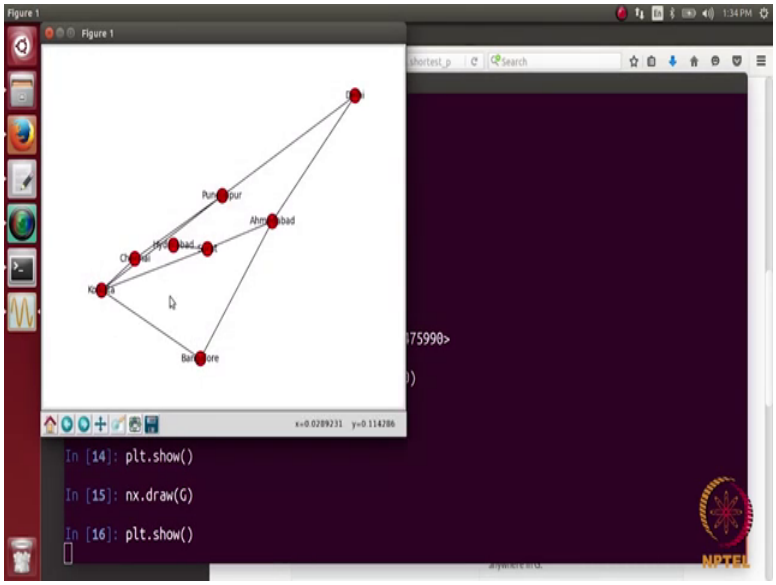
So, we can see this network here and it is connected. So, we have a graph here and every edge has a weight associated with it here. So, now, we what we want to see is we want to see the implementation of the shortest path functions. So, let us one by one take the shortest path functions and see how do they work? So, let us look at the first one here. So, if you give it the graph and the source and the target it will return you the shortest path let us try.

(Refer Slide Time: 29:00)



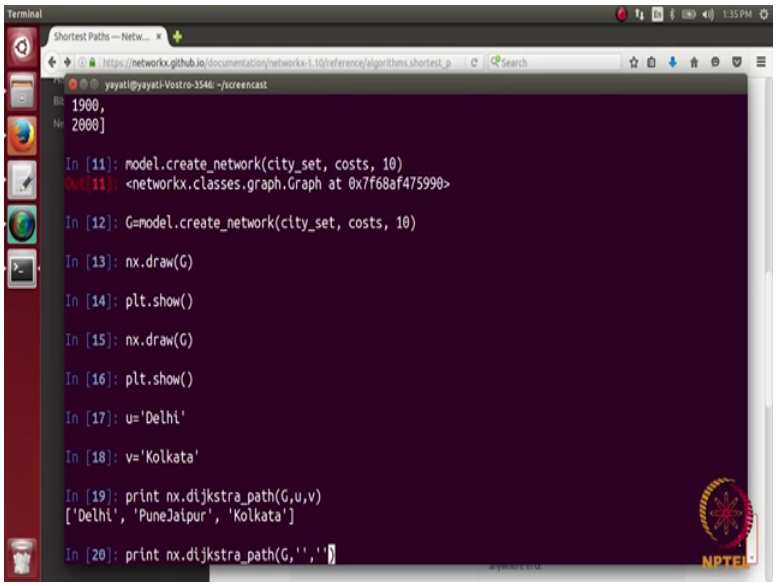
So, let us look a graph once more.

(Refer Slide Time: 29:02)



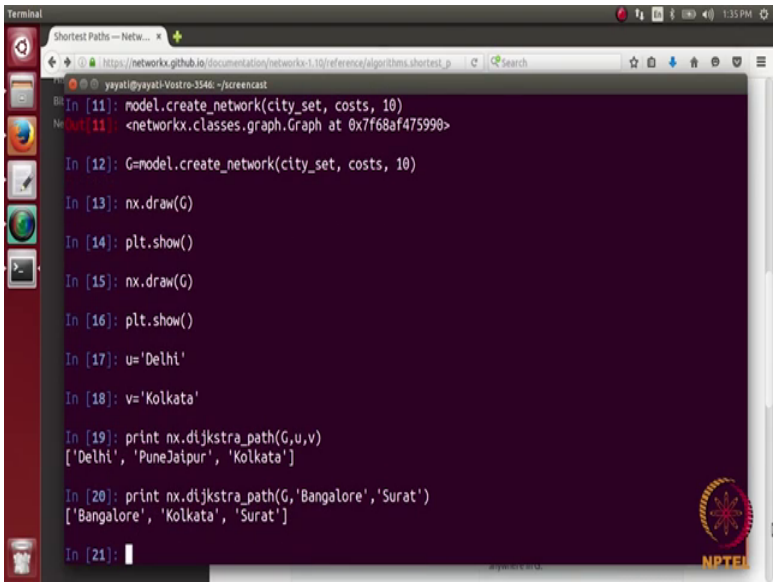
So, let us choose 2 nodes here let say we choose the aliant Kolkata.

(Refer Slide Time: 29:10)



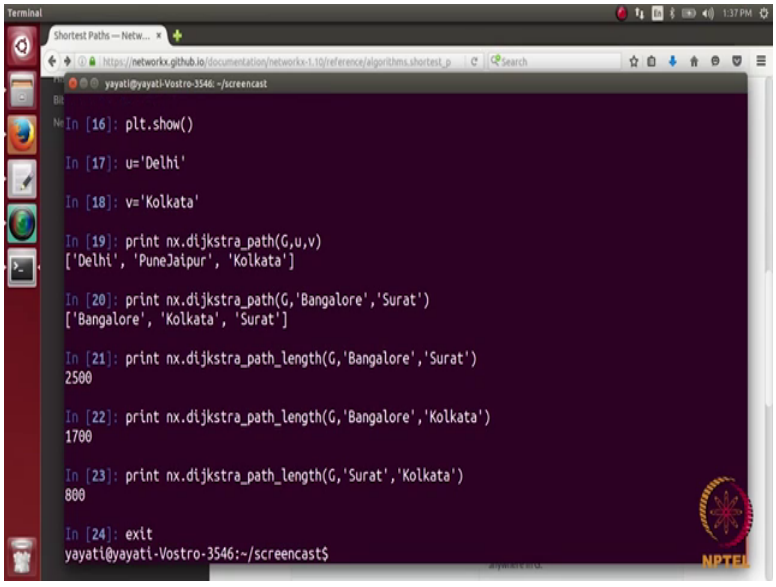
So, let say node number one u is Delhi and second node is now what we are going to do is we are going to look at the shortest path between these 2 nodes. So, what we are going to print is nx dot stra and then goes there underscore path and then you paths here network here and the first node and the second node yeah. So, you get here shortest path here. So, from Delhi you go to Pune and then from Pune you can go to Kolkata, similarly we can actually see between every let us choose another some 2 different cities let say let us choose Bangalore and Surat. So, let us write it down Bangalore and let say yeah. So, you see here that Bangalore and Surat are connected with Kolkata and this is the shortest path of it.

(Refer Slide Time: 30:27)



Next function is for path length where you do not want to see the path, but you want to see the path length. So, we know here. So, we do a path length here and we. So, this path length is returning here 2500. So, the costs associated with going from Bangalore to Surat is 2500.

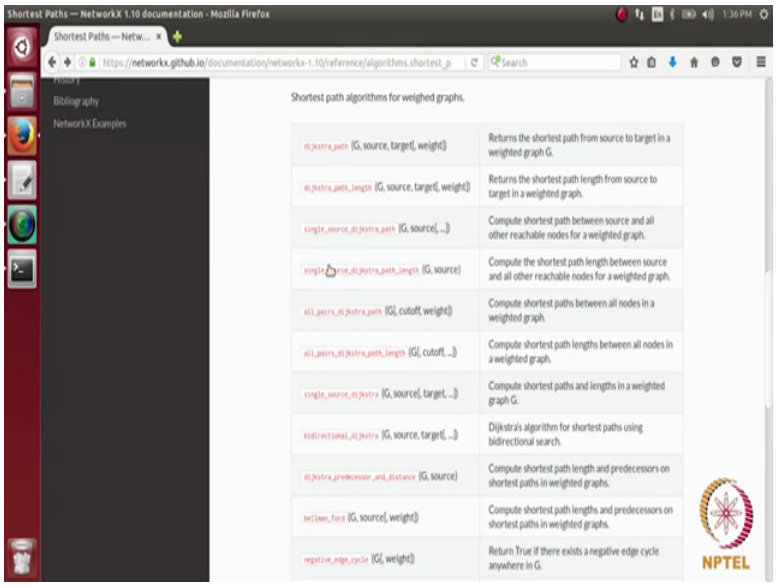
(Refer Slide Time: 30:59)



Why is it turning out to be 2500 let see? So, you let say that if you want to go from Bangalore to Kolkata you can go it in seventeen hundred and let say that you want to go from Kolkata to Surat. So, we can write it this way because it is an undirected graph its 8

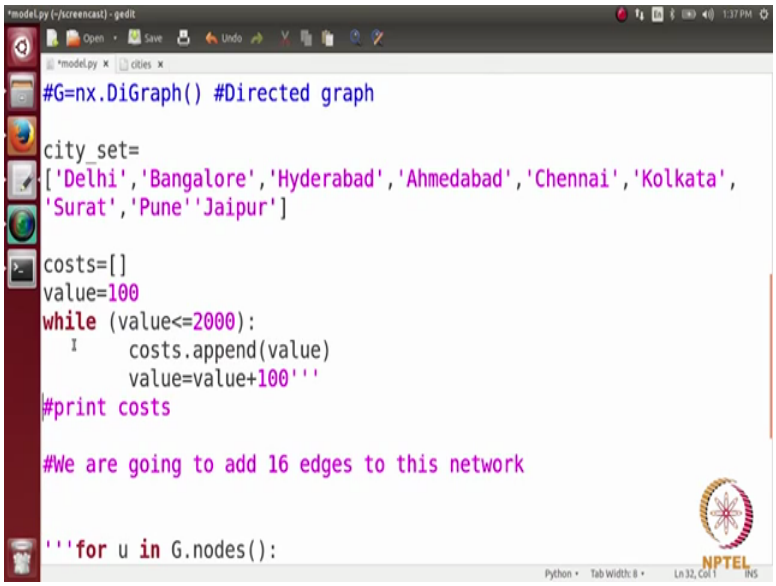
hundred when you some both of these it turns out to be 2500. So, you go from Bangalore to Kolkata which goes you 1700 and then Kolkata to Surat goes you 800 and this some turns out to be 2500 perfect.

(Refer Slide Time: 31:31)



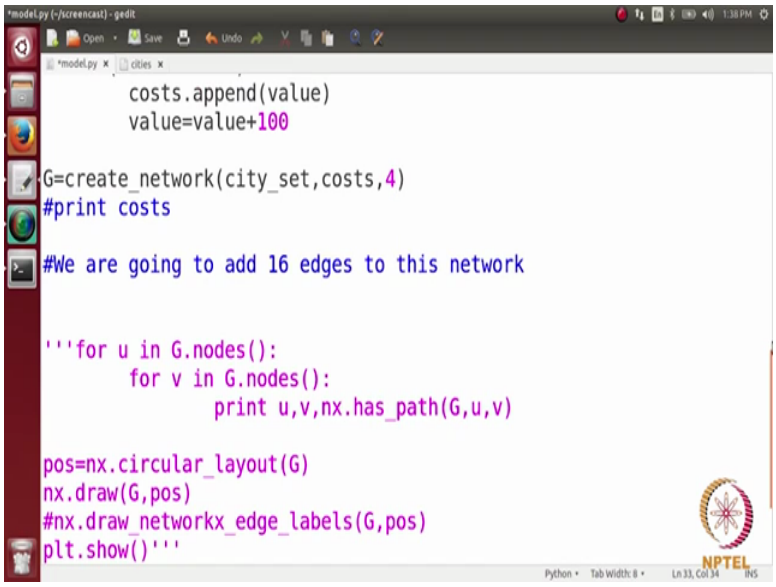
Then we have the single source dijkstra path. So, I will not going to each of this. So, what you can do here is you just give a source node and it will tell you the shortest path from the source node to every then node in the networks. Similarly you can look at its path length and then you have a all pairs extra path which will compute the shortest path between all the nodes in the network and then here also you can look at the path length as well I will just exit.

(Refer Slide Time: 32:05)



Coming back to our core now what we are now what we will do is we are going to plot the curve and let see what the curve is going to do. So, let us set first of all this costs rate to be have this city set here to have this costs here and ok.

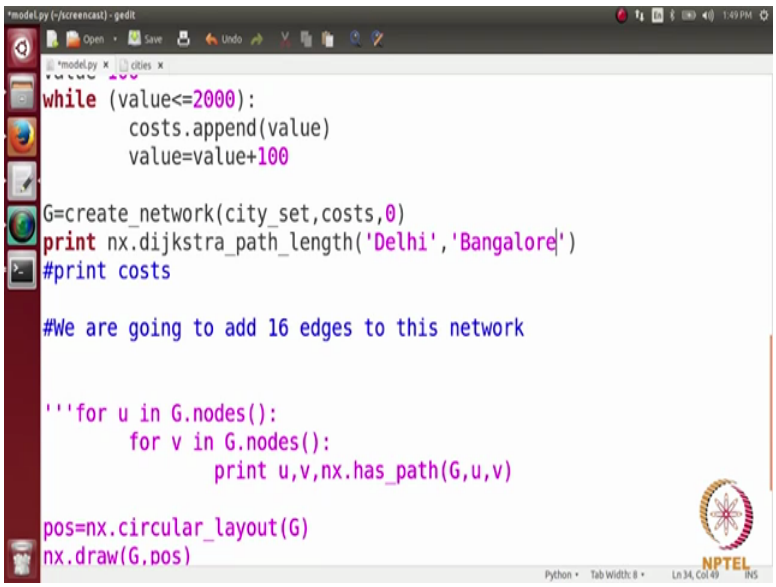
(Refer Slide Time: 32:57)



We do not need the functions and this is for let us (Refer Time: 32:52) it perfect and we are we had to call this function create underscore network and where we are passing the city set and then our costs and then the number of edges which we want to have. So, let us put very less number of edges 4.

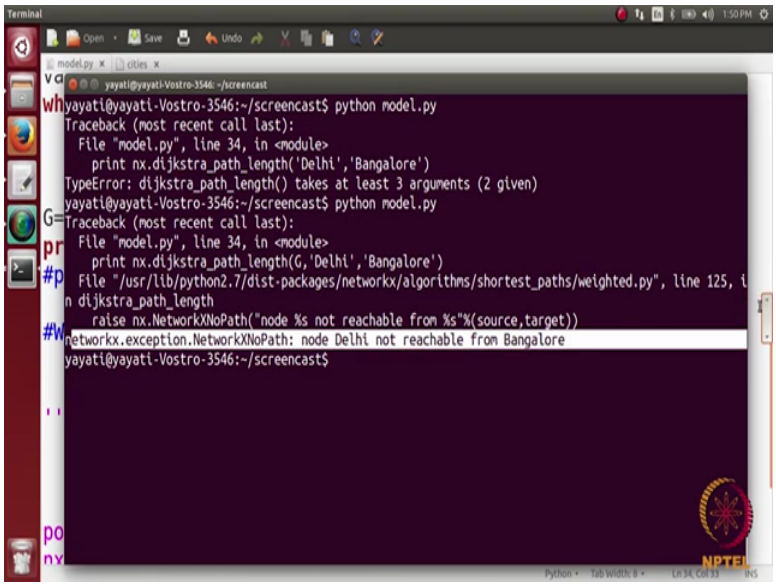
Now, one question what is going to be the path length if there is no connection between 2 nodes. So, we are interested in looking at. So, we have looked at various different extra path finding functions what does this function written if there is no path between a pair of nodes for example, assume that initially we add 0 edges in this network. So, there is no path between any 2 nodes in this networks since there is no edge. So, what happens if I print a; the extra path finding function in this case let us try to do it. So, let us choose to nodes let say Delhi and Bangalore.

(Refer Slide Time: 34:15)



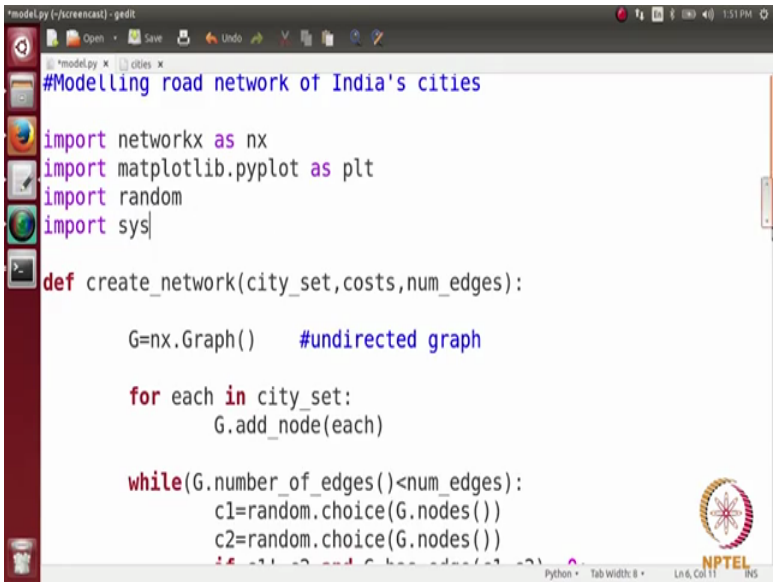
So, what I am going to do is I am going to print n x dot dijkstra and then path and then length path length between see Delhi and Bangalore let us always function in a.

(Refer Slide Time: 34:43)



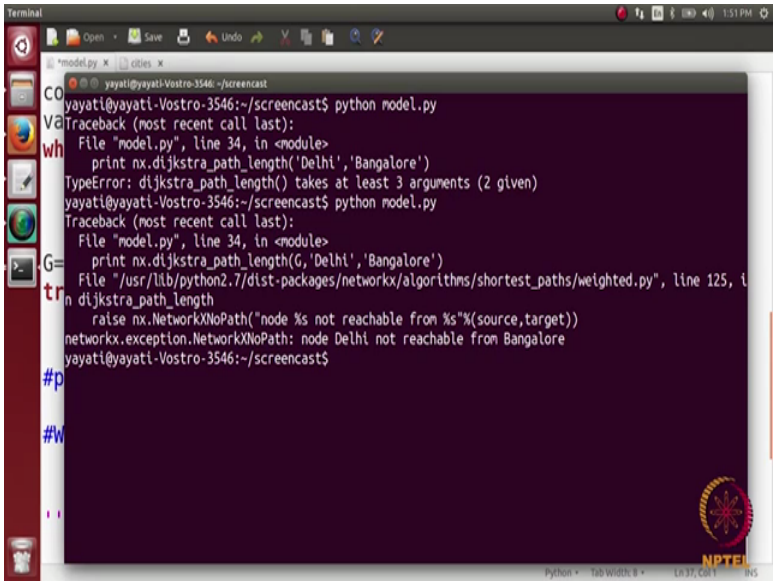
So, it has given a type error that we should have given 3 arguments. So, what is missing here is graph G right nodes run it. So, you see here that there is an exception. So, the code has writtened an exception and the exception is node Delhi not reachable from Bangalore. So, we see that how do we catch these exceptions in the network assume that I do not want to return and exception here rather I want my code to show mw as 0 here at the path length from Delhi to Bangalore not 0, it is infinity, right. So, my answer should be infinity or since a graph is very small let us represent infinity here by a big number let say 10,000. So, we are representing infinity with 10,000 though it is not a very well excepted notion, but we are just using it in this code for the sake of convenience.

(Refer Slide Time: 35:43)



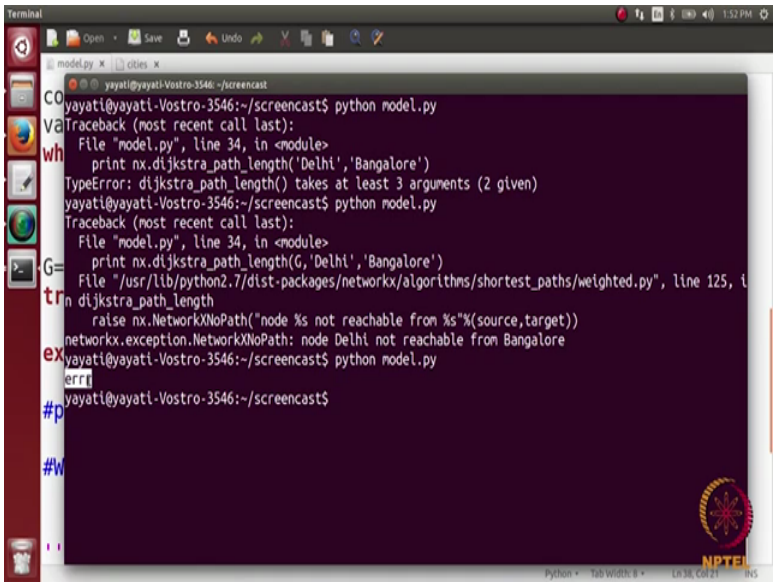
So, what we do this in this case is something called the exception handling which is very simple. So, let us look at we just need to import the module sees here after include importing this package what do I do it I just put the statement inside a try (Refer Time: 35:56).

(Refer Slide Time: 35:53)



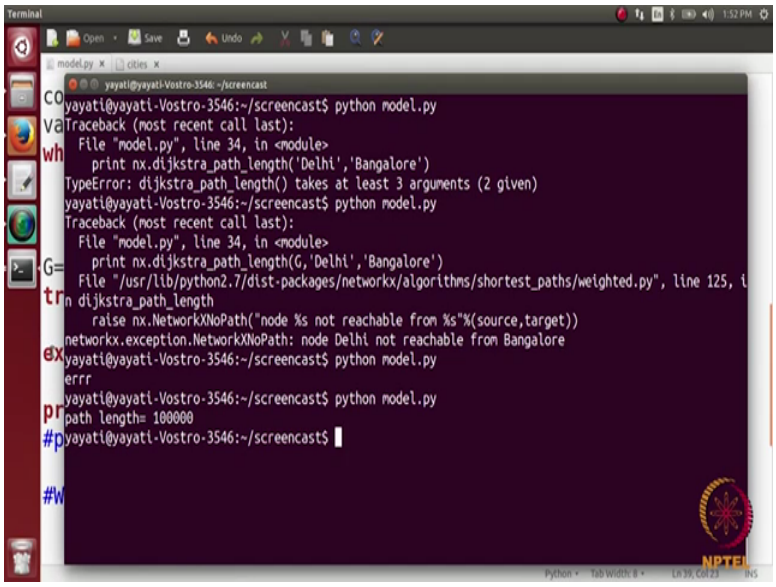
So, what my try (Refer Time: 35:58) does? It gets this path length L as nx extra path length G Delhi Bangalore and in case there is an error. So, we will looked at an error here and these exception handling and this error was a key error means here.

(Refer Slide Time: 36:25)



So, we except in the case of an exception what does we do it let say we print error let see how this piece of code works. So, you see here that they words an exception and this exception was because there was no path and we have printed here error.

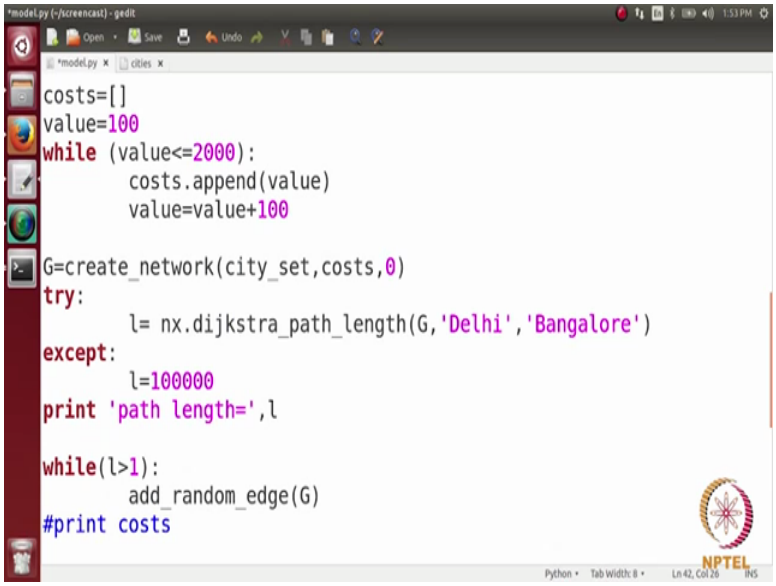
(Refer Slide Time: 36:52)



So, what we can do is instead of printing here error we can set this L to be 10,000; 10,000 we will let say even 100,000 and then we print here path length equals to L and let see here when be run this code again we get here a path length of 100,000. So, what this is doing whenever there is no path between 2 nodes it will written 100,000 which is

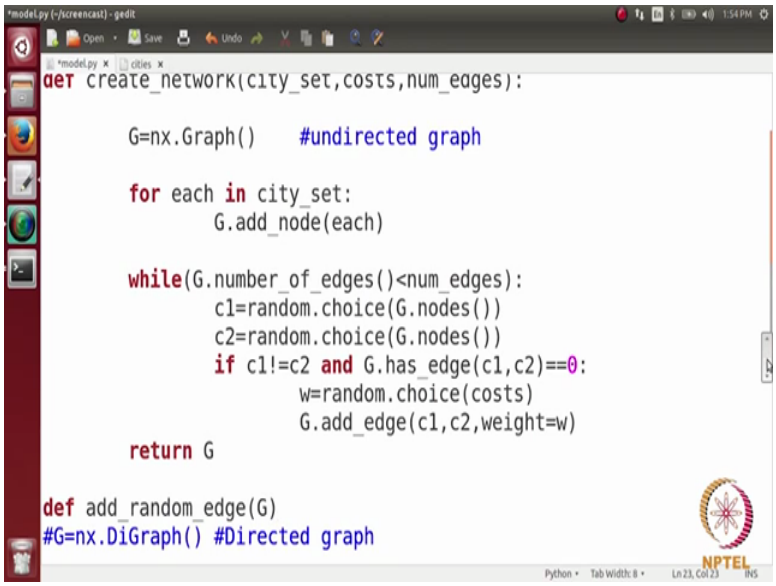
a very big number according to a code according to a network. So, it is a very small network, now what I want to do is what we want to do is we want to see a plot and what will this plot do there we be 2 cities in our network and will be adding random edges and as the at random edges we are interested to see how this path length decreases.

(Refer Slide Time: 37:59)



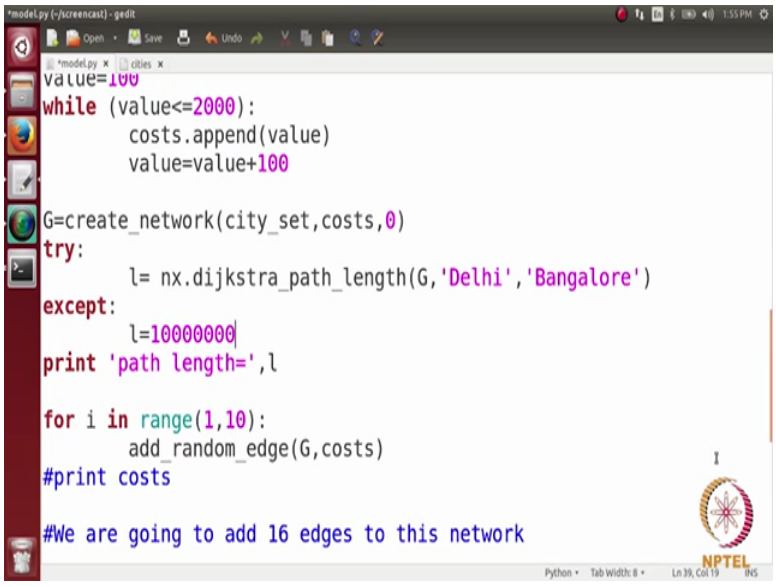
So, let us write a small piece of 4 hold this. So, what we are going to do is initially we have this graph here g. So, what we are going to do is while, we are going to do is we have calculated here I will write and L was here 10,000 what we want L to become is let say one because the minimum path length between 2 nodes is one till they become directly connected. So, till while your L it is greater than one. So, what you do here is you add an edge randomly. So, I will call a function here add random edge right add random edge in G. So, how this function is going to work is.

(Refer Slide Time: 38:50)



Lets create a function here define add random edge G and so we have the city set here.

(Refer Slide Time: 39:08)

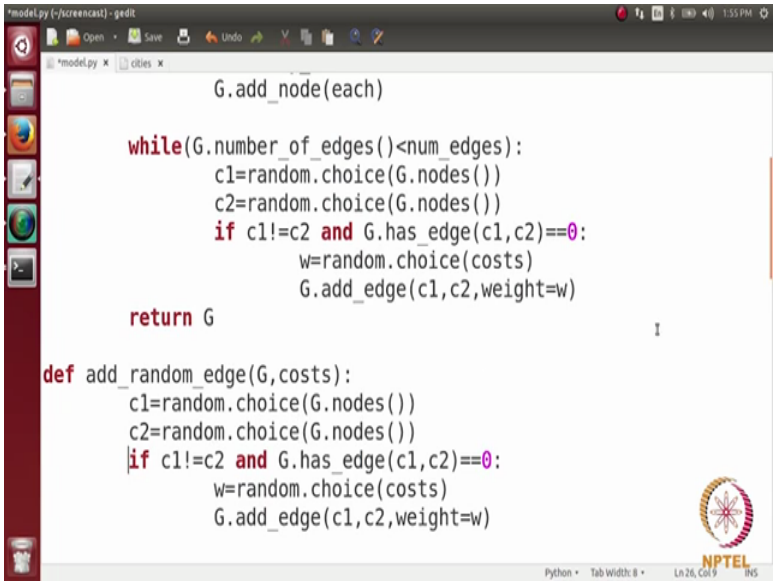


But we want to pass here the costs we goes every node should have a costs one thing here this pass length is actually equal to be cost here right. So, this is equal to a travelling cost.

So, we cannot say it is less than one. So, we will just run this code for some number of steps because path length is never going to be one here because our path length is a travelling cost here. So, what will go is for i in range 1 to 10 you will see for the 10 times

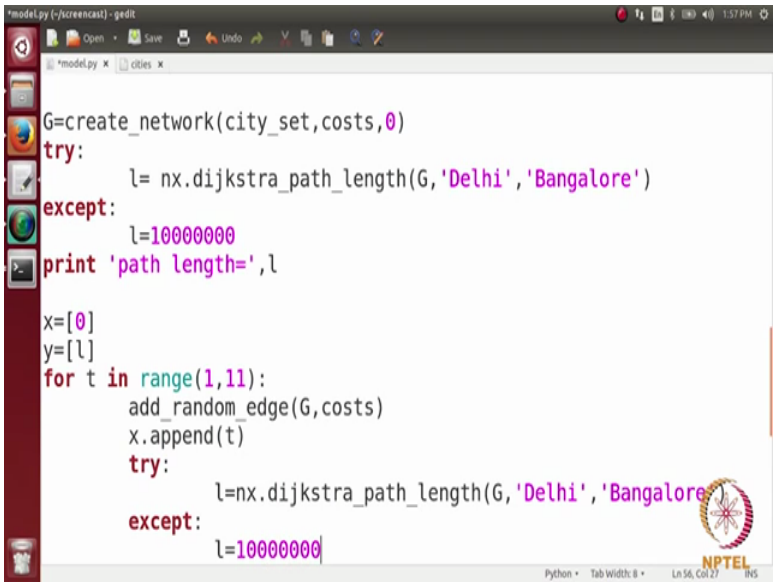
what happens here and let us make it a little bit larger number. So, add random edge G costs.

(Refer Slide Time: 40:06)



So, what it is going to do is what exactly we have done above. So, what we have done is right we have taken 2 random nodes and if there was no end between them we have chosen edge and added between them. So, exactly the same thing we are going to do here. So, essentially we are just copy pasting this code.

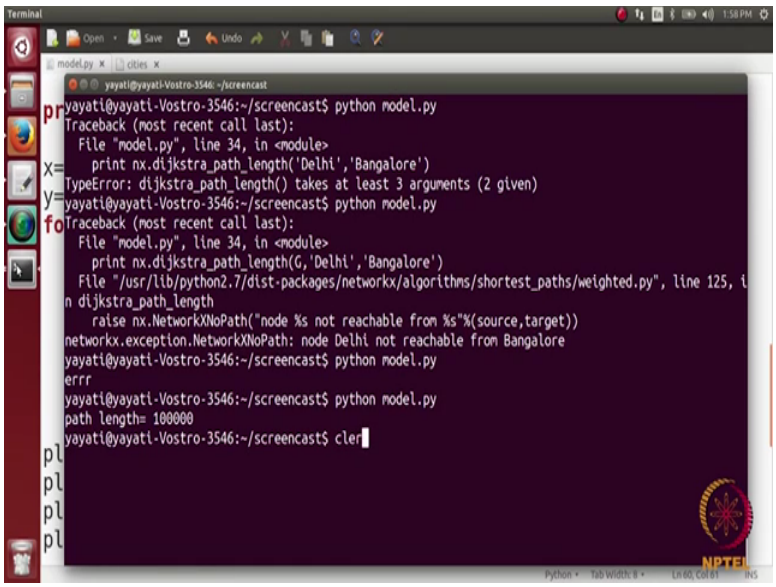
(Refer Slide Time: 40:35)



So, what we are doing is we will choose 2 nodes randomly and if there is no end between these 2 nodes will adder add an edge with their weight being equals to w. So, since done here add random edge G dot costs and after that what we are going to do is. So, since we have to plot this graph over 10 timestamps we need 2 arrays 1 is for x axis, 1 is for y axis in the 11. So, x axis is a time which is essentially I let us represent it as t.

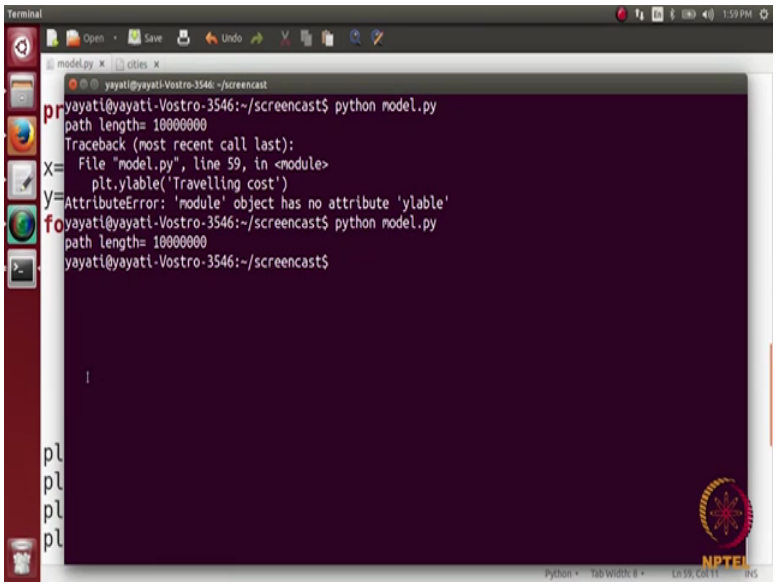
So, we will have an array x and we will have an array y x will be initially having the value 0 right because it is a 0 timestamp when y will initially having the value I which is a very big number code we are going to do here now is x dot append t that is a time and in y what we have to opened again if the dijkstra path length. So, it is again going to be a try function. So, we are going to try what give me are going to try is let us say let us taking L; L equals to n x dot dijkstra and then path length then G comma Delhi comma Bangalore right and then we are going to except and adder. So, this exception is in case where there is no path between the 2.

(Refer Slide Time: 42:32)



So, here we are going to set L equals to it remains the same which is the big number 1, 2, 3, 4, 5, 6. So, there are 7 0s, perfect and then what we are going to do is y dot append L and then we want to plot this. So, for plotting this we have plt dot plot and we have x and then y and we can actually give your axis name let us say that plt dot x label that is a x axis is a time and plt dot y label y axis is a let us say travelling cost and then we have a title let us say change in travelling cost as more roads are added in the network.

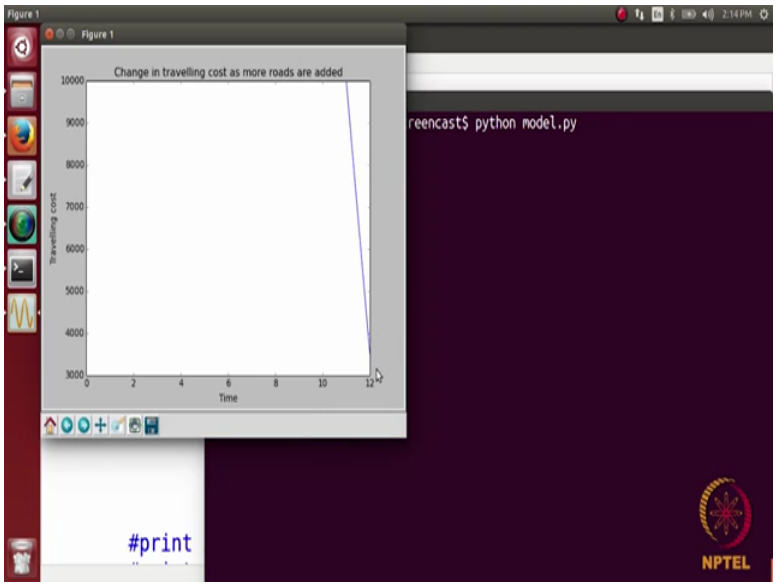
(Refer Slide Time: 43:34)



So, let us try to see. So, clear here the small spelling mistake here. So, I will take their having some problem it is printed are path length. So, here we have mention the small changes to this code. So, the value of L we have just change it to be 10,000 we can just to so that the plot looks nice and should not we where to high, but it is still higher on the path that we have very less as oppose we are just 2500 or something and then this take when y dot append L has been shifted.

So, instead of adding at the end we added inside the loop because we have this break statement inside the try block which is needed. So, if this break statement is there and we could y dot append at the end. So, the value is not append it. So, we append the values to are array y in between the try block and the added block and then we have a plt road show here which we were which was initially missing so which is required to show us the plot.

(Refer Slide Time: 44:56)



So, let us see here. So, when we execute this code. So, we see here something. So, the initial cost was 10,000. So, this was the cost which was which is actually infinity furors right. So, initially Delhi and Bangalore were not connected and after sometime steps. So, after some 11 time steps, so the random edges are keep creating in the network and after some eleven times steps this cost drops and comes to 3,000. So, it becomes connected here were just a small code. So, it might not be very clear to you. So, if you have any doubt regarding this you can leave it in the discussion forums and we can discuss it further. So, this was just a basing basic plotting function which we can do using python and networkx just an example for that.