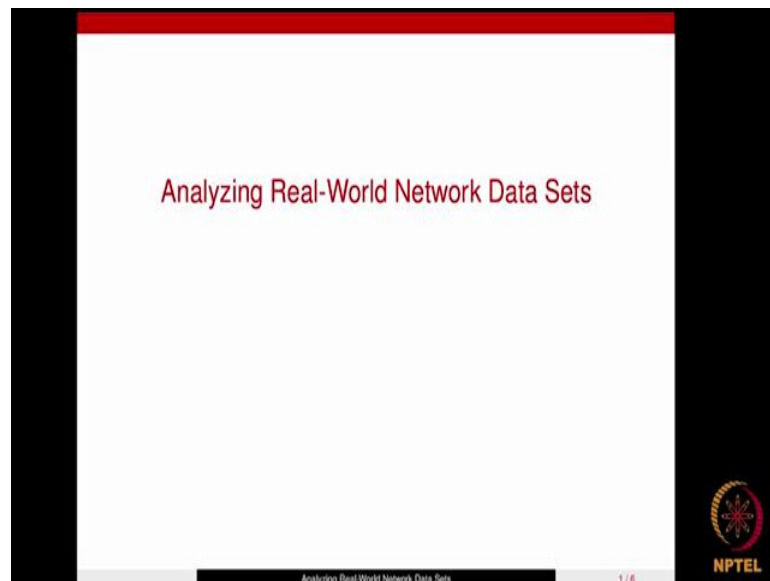


Social Networks
Prof. S.R.S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

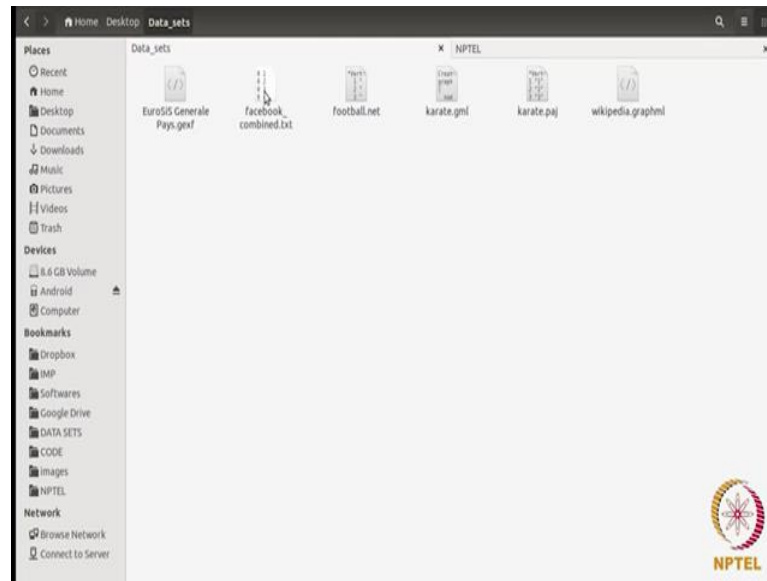
Lecture - 21
Handling Real-world Network Datasets
Datasets: Analyzing Using Networkx

(Refer Slide Time: 00:06)



Hey everyone in the previous video we had downloaded a number of network datasets in different formats, in this video we are going to see how we can analyze them using networkx package of python; let us take a look at the datasets that we have downloaded.

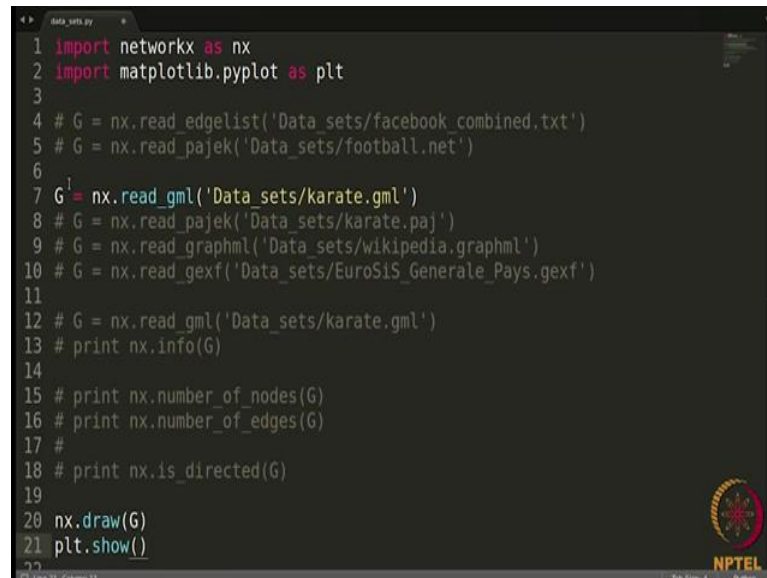
(Refer Slide Time: 00:25)



So, these are the datasets that we downloaded in the previous video. So, we had 6 networks, we have a network in gexf format, we also had a Facebook network in Edgelist format, we had a football network in dot net format which it is equivalent to Pajek format and we had karate club network in gml format, we also had karate network in Pajek format and we also had a Wikipedia network in GraphML format.

So, these were the 6 network datasets that we had, now we going to see how we can analyze them. So, I have all the datasets in this folder, I am going to create a new python file here where I will be writing all the 4. So, datasets.py I am going to open in it in an editor I am using sublime text here can use any editor for that matter.

(Refer Slide Time: 01:26)



```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 # G = nx.read_edgelist('Data_sets/facebook_combined.txt')
5 # G = nx.read_pajek('Data_sets/football.net')
6
7 G = nx.read_gml('Data_sets/karate.gml')
8 # G = nx.read_pajek('Data_sets/karate.paj')
9 # G = nx.read_graphml('Data_sets/wikipedia.graphml')
10 # G = nx.read_gexf('Data_sets/EuroSIS_Generale_Pays.gexf')
11
12 # G = nx.read_gml('Data_sets/karate.gml')
13 # print nx.info(G)
14
15 # print nx.number_of_nodes(G)
16 # print nx.number_of_edges(G)
17 #
18 # print nx.is_directed(G)
19
20 nx.draw(G)
21 plt.show()
22
```

Since we are going to make use of networkx package and I am going to import it, we are also going to visualize the networks. So, I am going to import matplotlib as well. Now let us take the first network in this folder we have let us make use of this Facebook combined dot txt network which is in Edgelist format let me copy this name.

So, now since the Facebook network is in Edgelist format, the function that we are going to make use of is read Edgelist. So, I am going to write `g = nx.read_edgelist` and here I am going to get the name of the network. Now our datasets are kept in a folder. I am sorry I think that is the name (Refer Time: 02:30). So, in that we have this is the name of the network. So, the you see the function that we are using is `read_edgelist` function, which is present in networkx package, and as a parameter we are giving the name of the network. Now this function basically takes the network in a in the Edgelist format and returns a graph object; and then we can apply any function on this graph object.

For example, if you want to look at basic information of this network, we can write `nx.info` on as a parameter here give G. So, `info` is function which provides a basic detail as to the number of nodes number of edges etcetera about to graph. So, let us save this file.

(Refer Slide Time: 03:26)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
4039
88234
False
anamika@anamika-Inspiron-5423:~/Desktop$
```



And I am going to open my terminal here, I am going to run this file. So, all right. So, here you see firstly, it tells us the type; the type is graph as in it basically tells us whether it is digraph or directed graph or it is multi graph and it tells us the number of nodes, it also tells us the number of edges and the average degree.

So, these are just a basic detail about the graph, now let us go back to our file and add some more things. So, this is what the basic things if you just want to get the number of nodes you can write. So, now, `nx.number_of_nodes` is a function which returns you the number of nodes if you have to just get the number of nodes, you can use this function and similarly if you want the number of edges if you do not want all the other information you can use this. And in case you want to know whether the graph is directed or not, then you can make use of this function `print nx.is_directed`. So, as a parameter you pass the graph.

So, this should tell us whether the graph is directed or not, let us go back and try to run this. So, here you see after the basic statistics it told us the number of nodes and edges and it is false that is it is non directed. So, this Facebook network that was a friendship network it is basically undirected network. So, that was about how we can we read and Edgelist network into networkx object. Now let us see the other kinds of networks that we have, we also have here dot net format as I told you in the previous video that dot net

format is basically the Pajek format. So, in order to read this network into the networkx object we will use Pajek functions.

So, let me show you how to do it. So, what I am going to do is I am going to use this function read Pajek, which is a function that is used to read a dot net or dot paj file. So, let me check the name of the network football dot net. So, change it. So, I am reading this dot net network through this function read Pajek into a graph object g, and then I applying all these operations let us see and we run this.

(Refer Slide Time: 06:10)

```
Average degree: 43.6910
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 4039
Number of edges: 88234
Average degree: 43.6910
4039
88234
False
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: MultiDiGraph
Number of nodes: 35
Number of edges: 118
Average in degree: 3.3714
Average out degree: 3.3714
35
118
True
anamika@anamika-Inspiron-5423:~/Desktop$
```




So, here you see that the type of graph is multi digraph; that means, there are multiple edges between the nodes, and it is also directed graph and it is telling us a number of nodes number of edges, and since it is a directed graph it is telling us the average in degree and average out degree, and after that again it is giving the result of the functions number of number of nodes number of edges, and it is true which means it is a directed graph.

So, this these are just the basic functions, let us see what other kinds of networks we have. We have gml network we have a Pajek format as well let me show you that for reading the Pajek files as well, you use the same function that is read Pajek. So, the network name is karate dot p a j. So, I will replace this ok.

(Refer Slide Time: 07:15)


```
False
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: MultiDiGraph
Number of nodes: 35
Number of edges: 118
Average in degree: 3.3714
Average out degree: 3.3714
35
118
True
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: MultiGraph
Number of nodes: 34
Number of edges: 78
Average degree: 4.5882
34
78
False
anamika@anamika-Inspiron-5423:~/Desktop$
```



So, when I run this, I am getting that this is a multi graph and the number of nodes is 34 number of edges is 78 and this is a average degree and it is not a directed graph. So, it is. So, I am getting false here ok.

(Refer Slide Time: 07:35)

```
Number of nodes: 921
Number of edges: 1081
Average in degree: 1.1737
Average out degree: 1.1737
True
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Traceback (most recent call last):
  File "data_sets.py", line 9, in <module>
    G = nx.read_gexf('Data_sets/EuroSiS_Generale_Pays.gexf')
  File "<string>", line 2, in read_gexf
  File "/usr/lib/python2.7/dist-packages/networkx/utils/decorators.py", line
263, in open_file
    result = func(*new_args, **kwargs)
  File "/usr/lib/python2.7/dist-packages/networkx/readwrite/gexf.py", line 1
61, in read_gexf
    G=reader(path)
  File "/usr/lib/python2.7/dist-packages/networkx/readwrite/gexf.py", line 5
74, in __call__
    raise nx.NetworkXError("No <graph> element in GEXF file")
networkx.exception.NetworkXError: No <graph> element in GEXF file
anamika@anamika-Inspiron-5423:~/Desktop$
```



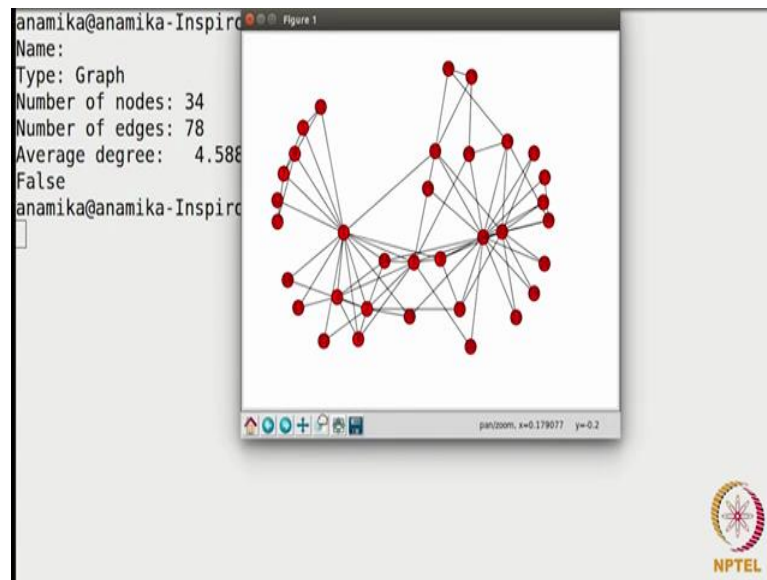
Now, 2 more network formats that we have are GraphML and gexf. So, let me show you quickly show you how we can read them as well. So, graph m l how do we handle it we. So, let me the function that we use it read graph m l and the file name is Wikipedia dot graph m l. So, this is how we will use it; since this is written information, I am going to I

am going to comment this, now let me run this all right. So, what we are getting here is that it is a digraph Wikipedia because this is graph between that the nodes are the articles. So, is it basically tells us whether an article is referred to in the other article or not? So, it is basically a directed graph 921 are the number of nodes number of edges are giving and since it is directed we are getting in degree as well as the out degree, the average in degree and out degree and it is true that means, it a directed graph.

So, let us go back here and the only format that is left is gexf, let me quickly show you how to convert into graph object as well and copying it is name and here let us go back. Since this is a gf; gexf format will make use of function `read_gexf` and (Refer Time: 09:24) name of the function let me rename this, so that it does not create any problem and let me rename this is well. So, it should work let us see. So, this is how we can read various networks in different formats and convert them into a graph object; once they are converted into graph object, we can apply various functions on them, and we can play around with them. Now let also show you how we can visualize a network in networkx package.

So, let me take a small network, let me take karate network itself and we had it in gml format. I think we have in that video we can quickly add it. So, the function is `read_gml`. So, if you have a graph which is in gml format, the function that you will used is `nx.read_gml`. So, all right. So, we read a karate network which was in gml format, we made use in function `nx.read_gml`.

(Refer Slide Time: 11:05)



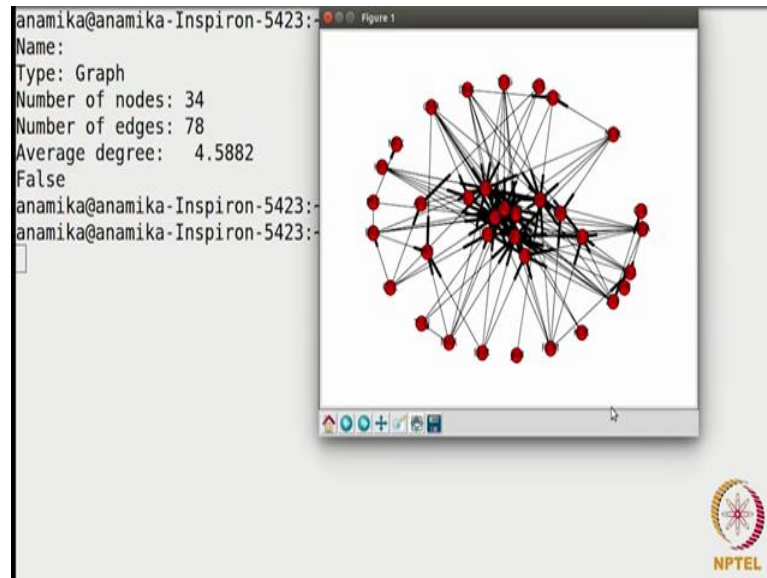
So, let us execute this program. So, this is a simple graph, and this is a number of nodes and edges and it is an undirected graph. Now let me show you how we can visualize this graph? I am going to comment this; I am going to comment this as well.

Now, the function that we used to visualize the graph is `nx.draw`, and the parameter we will use to draw the graph that we have to draw in order to see that graph we have to use this function `plt.show`, basically the show function which is available in matplotlib. So, that is how we will be able to see this graph, now let me run this all right. So, this is all graph the karate network, and the labels are given here. Let me also show you a few features that this interface provides for example, this when you click this you can just move the graph the way you want, and the next option is zoom to rectangle. So, when you click that if you want to carefully observe some part of the graph you can just zoom it and see and if you further want to do that you can do the way you want and then you can go back then you go then go back. So, these are few functions that few features that you can make use of, and this is configured subplots this will be used when we plot something this is just a graph, I will show you the functionality of this later then this is how you can save the figure ok.

Let me close this window now let me go back to the program; let me also show you how directed graph in networkx looks like. Since this karate is an undirected graph let me

comment this and if I am not wrong this football network was a directed network. So, I am doing the same analysis on football dot net.

(Refer Slide Time: 13:13)



Let me execute it again, you see this is how directed graph and networkx looks like. So, you see the arrows are represented like this, if you want to zoom it again you can make use of this feature.

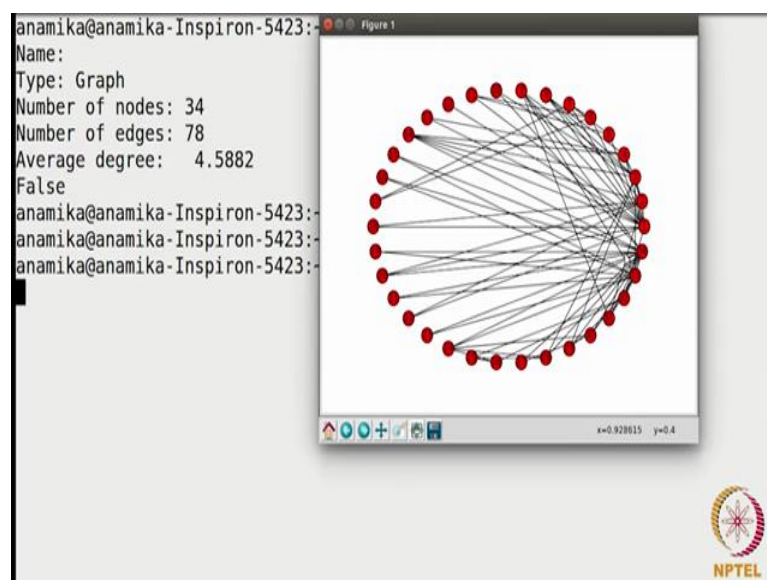
You want to zoom this area you can further zoom it is well. So, this is how you can closely analyze whatever you want in the graph right and you can then go back, or you can just press home here. Then you can save the graph is well let me close this now I am going to continue the rest of the analysis on karate networks.

(Refer Slide Time: 13:51)

```
1 import networkx as nx
2 import matplotlib.pyplot as plt
3
4 def plot_deg_dist(G):
5     all_degrees = nx.degree(G).values() #all the degrees
6     unique_degrees = list(set(all_degrees)) #all the unique degrees
7
8     count_of_degrees = []
9
10    for i in unique_degrees:
11        x = all_degrees.count(i)
12        count_of_degrees.append(x)
13
14    plt.loglog(unique_degrees, count_of_degrees, 'yo-')
15    plt.xlabel('Degrees')
16    plt.ylabel('Number of nodes')
17    plt.title('Degree distribution of karate network')
18    plt.show()
19
20
21
22 # G = nx.read_edgelist('Data sets/facebook_combined.txt')
```

So, I am going to on comment object back. Let me also show you that there are different layouts which are available in networkx for example, we have as if now use this function `nx.draw`, we can use another function if you want to different layout. So, we can use `nx.draw_circular`. So, this is one of the layouts there are various layouts available, let me show you the output that we get in this case.

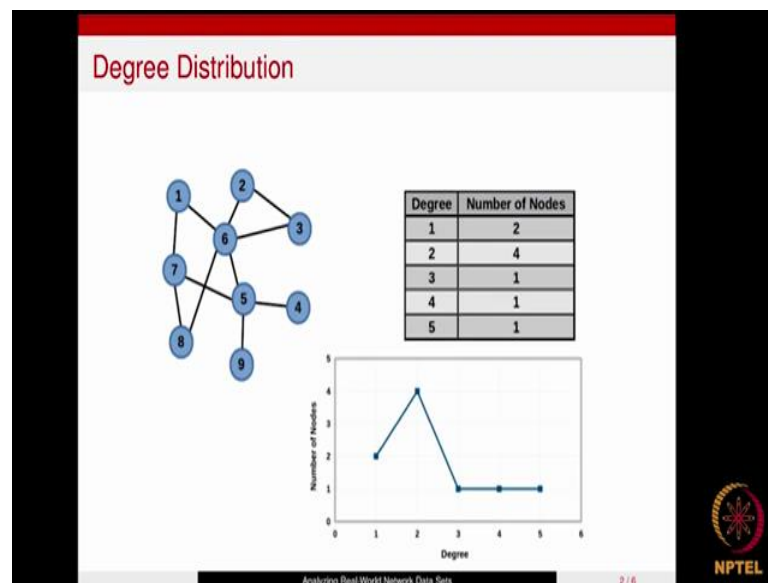
(Refer Slide Time: 14:23)



So, here the all the nodes are arranged along a circle and the edges between them are shown like this. So, this is called circular layout, there are number of other layouts as

well for example, spectral layout and spring layout. So, you can just read the documentation about them. So, we have visualized the network let us close this and let us go back to our program and we can do some basic analysis on this, one of the thing that we can check on this network is we can check the degree distribution.

(Refer Slide Time: 15:00)



Now, what is degree distribution? Degree distribution basically tells us how many nodes there in the network are, that have a degree. So, this is done for all possible degrees that a node can have in the graph; a let us take this example graph. So, here how many nodes are having degree 1? So, node number 4 and node number 9 they are having degree 1. So, corresponding to 1 will have 2 and similarly we can check how many nodes are having degree 2, how many nodes are degree 3 4 and 5. So, these are all the possible degrees that nodes can have in this graph and corresponding to the degree we have the number of nodes that have that particular degree. So, this basically is called the degree distribution of the nodes in a graph. Now it is always nice idea to plot the degree distribution to get a better idea of the graph.

So, when we plot this, we get this kind of distribution. So, on the x axis we maintain the degree and on the y axis we maintain the number of nodes that have that particular degree. So, this is the kind of plot that we get for this example graph, we can check for our datasets what kind of degree distribution to the exhibit. So, let us go back to our program and let us try to check what kind of degree distribution this karate network has.

For that purpose, I am going to create a function to plot the degree distribution of this graph G. So, let me create a function here, before we implement function, I want to show you a few things on the ipython console. So, let me copy this and let me open the ipython console here.

(Refer Slide Time: 17:03)

```
anamika@anamika-Inspiron-5423:~/Desktop$ 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 networkx as nx

In [2]: import matplotlib.pyplot as plt

In [3]: G = nx.read_gml('Data_sets/karate.gml')

In [4]: nx.degree(G)
```



So, basically, I am I just copied those first 2 statements here and let me also copied this statement here ok.

So, we have the karate network in the object G; now if I want to see the degree of each node in this graph, I can make use of this function `nx.degree(G)`.


(Refer Slide Time: 17:33)

```
16: 2,  
17: 2,  
18: 2,  
19: 2,  
20: 3,  
21: 2,  
22: 2,  
23: 2,  
24: 5,  
25: 3,  
26: 3,  
27: 2,  
28: 4,  
29: 3,  
30: 4,  
31: 4,  
32: 6,  
33: 12,  
34: 17}  
  
In [5]: nx.degree(G).values
```



(Refer Slide Time: 17:36)

```
In [4]: nx.degree(G)  
Out[4]:  
{1: 16,  
2: 9,  
3: 10,  
4: 6,  
5: 3,  
6: 4,  
7: 4,  
8: 4,  
9: 5,  
10: 2,  
11: 3,  
12: 1,  
13: 2,  
14: 5,  
15: 2,  
16: 2,  
17: 2,  
18: 2,
```



So, what this function does this degree function it basically returns a dictionary, where the key is the number of the node and the value is the degree of that node. So, here we get the dictionary for all 34 nodes, we are getting the degrees of these nodes. Now let us go one step back and recall what is our (Refer Time: 17:56) is to have the degree distribution of nodes in the graph.

So, basically, we want that for particular degrees how many nodes are there in the network that are have in that degree. So, we basically first of all want to get the possible

degrees that the nodes are having in this network. So, we are getting the dictionary here where we are getting all the possible values of the degrees that is in that the nodes can have. So, what we are interested here in is basically the values. So, what I can write is `nx.degree(G)`. So, this is going to give me a dictionary all I am interested in is the value. So, I am going to write dot values here.


(Refer Slide Time: 18:41)

```
2,  
2,  
2,  
2,  
3,  
2,  
2,  
2,  
5,  
3,  
3,  
2,  
4,  
3,  
4,  
4,  
6,  
12,  
17]  
  
In [6]: set(nx.degree(G).values())
```



(Refer Slide Time: 18:43)

```
32: 6,  
33: 12,  
34: 17}  
  
In [5]: nx.degree(G).values()  
Out[5]:  
[16,  
9,  
10, 1  
6,  
3,  
4,  
4,  
4,  
5,  
2,  
3,  
1,  
2,  
5,  
2,
```




So, it should return me earliest which is having all the values. So, what is get here is basically all the possible degrees that the nodes are having. Now my aim is get the

possible degrees that the nodes can have; here you see there are lot of reputation. So, I want to get the unique degrees. So, in that case what I can do is, I can just write all this inside of function set.

(Refer Slide Time: 19:13)

```
2,  
2,  
5,  
3,  
3,  
2,  
4,  
3,  
4,  
4,  
6,  
12,  
17]  
  
In [6]: set(nx.degree(G).values())  
Out[6]: {1, 2, 3, 4, 5, 6, 9, 10, 12, 16, 17}  
  
In [7]: list(set(nx.degree(G).values()))  
Out[7]: [1, 2, 3, 4, 5, 6, 9, 10, 12, 16, 17]  
  
In [8]: █
```



So, what it will do is, it will convert the output into a set and we know that in set there cannot be in repetitions. So, what we get is the unique values, basically the unique degrees that the nodes can have in this particular network. Now a list is more flexible data structure as compared to set because we can perform lot of operation. So, I can further convert this into list if I want; this is basically up to you how or you want to handle the data.

So, I convert this into a list now what I get finally, is a list of all the unique values of degrees that the nodes can have in this network. So, I should you here so that we can see what is going on in the function. Now let us get back and use these functions in the function that we are creating. So firstly, I want all the degrees. So, I will write `nx.degree(G)`, I want all the values only I do not want the dictionary. So, I will write `dot values`. So, here I get all the degrees let me comment all the degrees.

Now, I am also interesting getting all the unique degrees so that I can see what the possible degree values is. So, what I am going to do the same thing that I did on the console, I am going to pass all degrees here. So, here I will get all the unique degrees. Now to get the degree distribution what I basically want is I want to out of all the values

that are there in this list unique degrees, I want to see how many nodes are having that particular degree. So, basically what I will do is I will fetch one element out of this list that is unique degrees, and I will see how many occurrences of that value there in all degrees are right.

So, probably I can start for loop here. So, I will write for i in unique degrees sorry for i. So, what I want to check is how many occurrences of i are available in all degrees. So, I can start a variable probably x is equal to all degrees dot count. So, we have this function count in a list, which tells us the occurrence number of occurrences of a particular element in that list. So, x will be telling us the number of occurrences of the degree i in the list all degrees. So, probably we can keep a track of all these values. So, we can create another list count of degrees I will I started a empty list and so I am going to append the x values to this list. So, basically the occurrences of the first degree are now stored in the list count of degrees.

So, after this for loop is finished, we will have all the degree distributions in this list that is called count of degrees, and then we can plotted at. So, let us try plotting is it. So, plt.plot as you might be knowing there are 2 parameters that have to be passed here. So, on the x axis we want all the unique degrees, and on the y axis we want have many nodes have that degree which we have stored in count of degrees. So, will pass that here. So, let us try plotting it. So, I will write plt.show we have not call this function x.

(Refer Slide Time: 23:20)

```
data_sets.py
22 # G = nx.read_edgelist('Data_sets/facebook_combined.txt')
23 # G = nx.read_pajek('Data_sets/football.net')
24
25 G = nx.read_gml('Data_sets/karate.gml')
26 # G = nx.read_pajek('Data_sets/karate.paj')
27 # G = nx.read_graphml('Data_sets/wikipedia.graphml')
28 # G = nx.read_gexf('Data_sets/EuroSIS_Generale_Pays.gexf')
29
30 # G = nx.read_gml('Data_sets/karate.gml')
31 # print nx.info(G)
32
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 print 'Density is', nx.density(G)
43
```

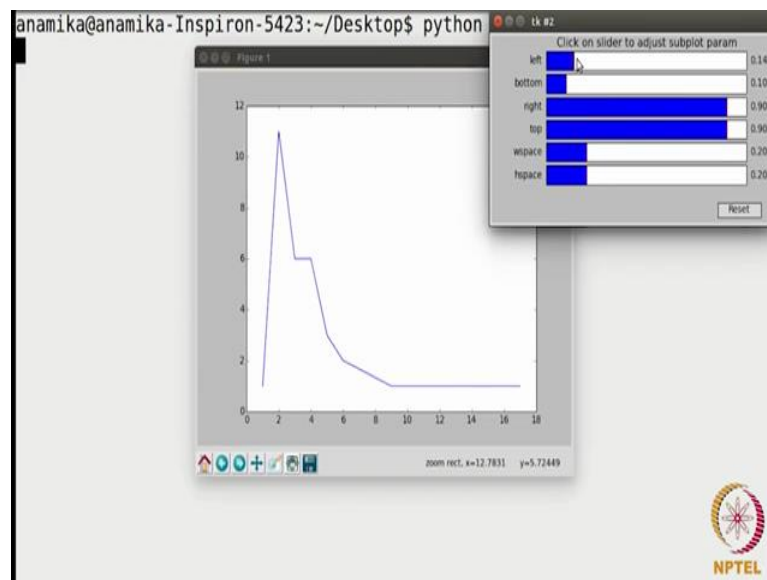

So, let me call this function here and comment this. So, I will call this function plot degree distribution and I will pass this G here ok.

(Refer Slide Time: 23:38)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Name:
Type: Graph
Number of nodes: 34
Number of edges: 78
Average degree: 4.5882
False
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ clear
```



(Refer Slide Time: 23:40)

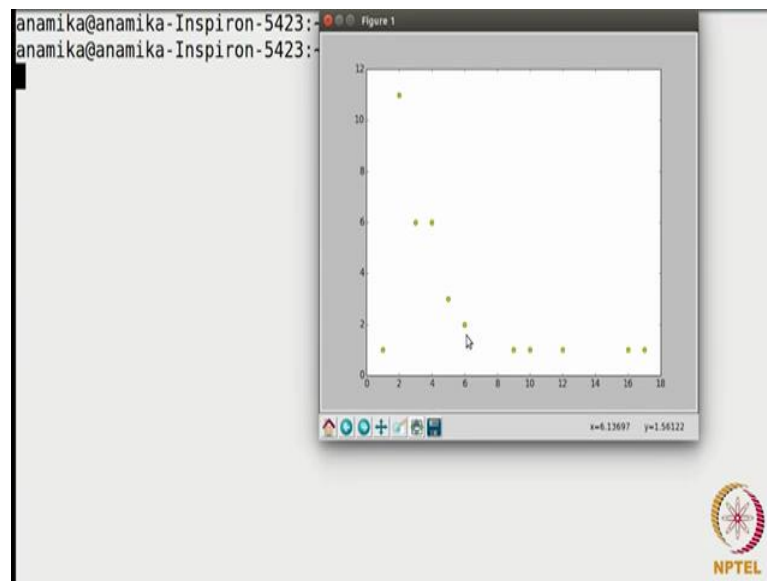


Let us go back here not here will go will try (Refer Time: 23:39) this. So, this is the degree distribution that we are getting for the karate club ah network.

So, this is the x axis where the possible degrees are there and on the y axis the number of nodes having that degree are there, and we you can just again play around with this plot as I told you, you can just use this to move the plot, you can use this to zoom of

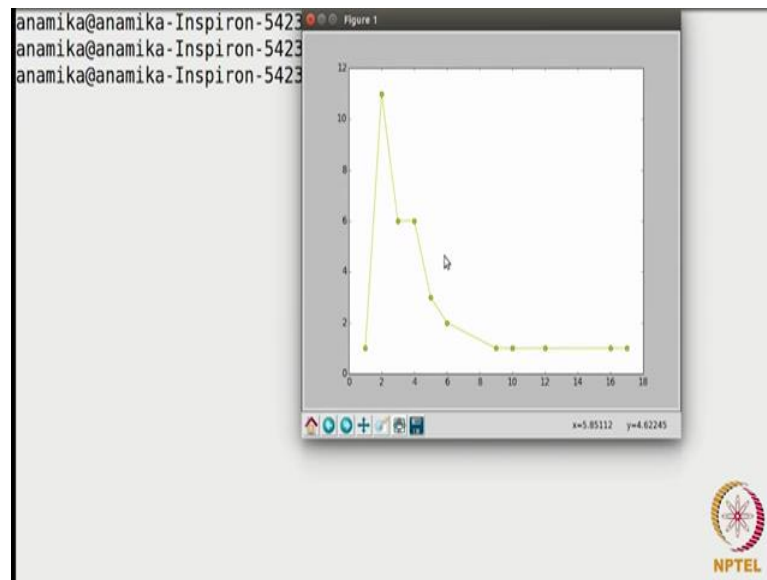
particular part and you can just go back as well and this is a feature that you can use here basically you can just increase or decrease the x and y margins. So, this is up to you whatever way you want it, and you can always reset it is well. So, let me close it. So, this is basically up to you how you can how you want to visualize it. Let be close it there was no x and y axis as you see. So, we can do that. So, let me just quickly do that, before that you can also change the way this plot is appearing so, if I put these dots you will get the plot in the form of these dots ok.

(Refer Slide Time: 24:56)



So, you get yellow dots here and you can also put line here and dots is well. So, we will get both the things.

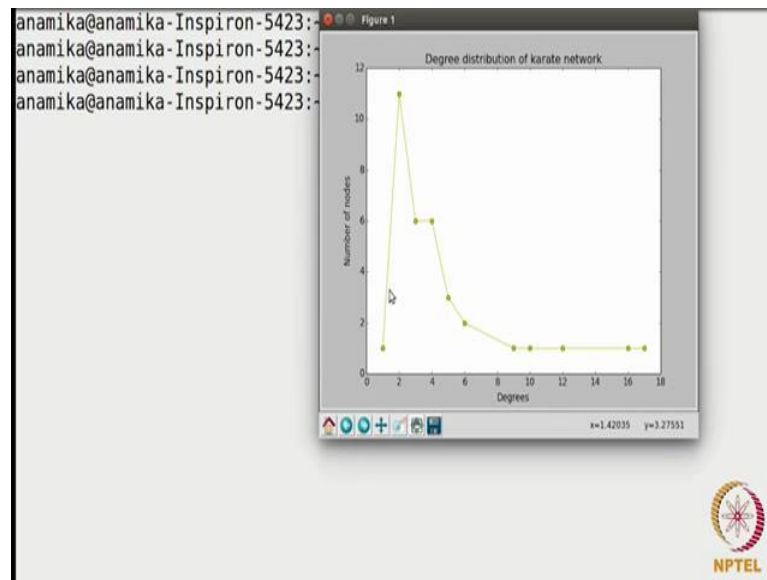
(Refer Slide Time: 25:06)



So, this is plot that you getting; one thing to observe here is that most of the real world networks exhibit power law degree distribution, which means that there are very few nodes which have very high degree and there are lot of nodes which have very less degree. So, the same is being followed in this small network is well, apart from one exception of this node. So, that might happen in some cases, but in general real-world networks have this power law degree distribution ok.

So, let us go back and add a few more things may be let us add the x label. So, let us add degrees here and then we can add the y label as well may be number of nodes and maybe we can add the title as well and degree distribution of karate network.

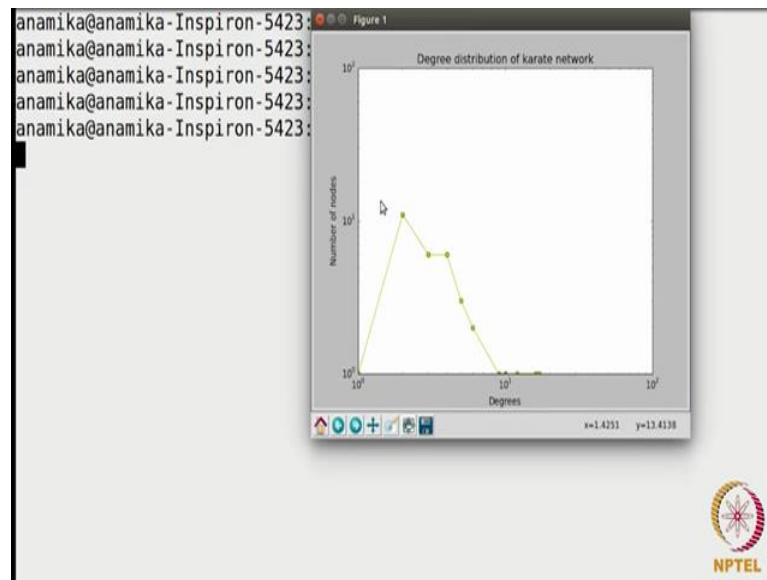
(Refer Slide Time: 26:17)



So, let us sum this you get the title and the x and y labels as well. So, you can just use more features more functions and decorate this plot the way you want. There is one more thing that usually is done in case of power law degree distribution, we can also check the log log plot of that.

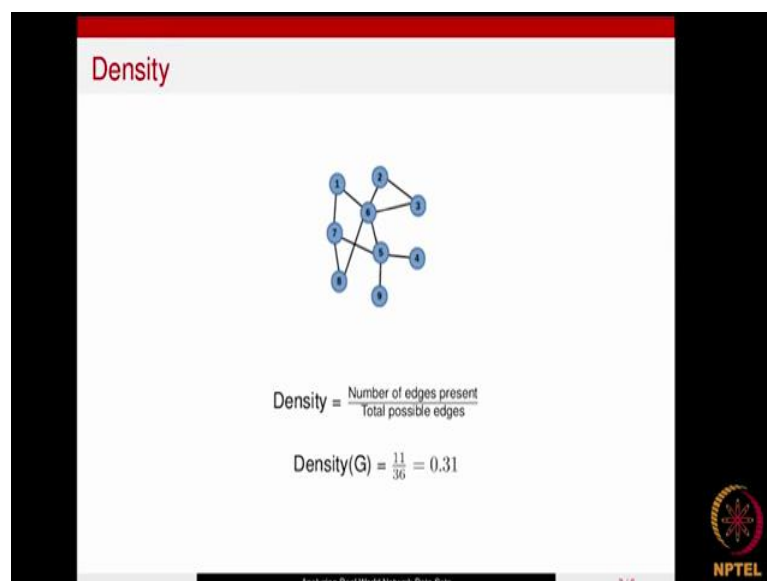
So, how can we do that we can just replace this plot by log log. So, in that case it will give as a log log plot which basically means in take the log of x axis it take the log of y axis, and if network is following complete power log the log log plot should be in a straight line. So, let us see what kind of plot we get in this case.

(Refer Slide Time: 27:01)



So, here firstly, we had section here. Secondly, it was not perfect power law. So, we have this kind of line which is not exactly straight. So, this is the kind of log log plot that we are getting in this case. So, this one about the degree distribution, now I am closing it let us go back here and see some more properties that we can analyze on this network we have done with degree distribution let us go ahead. So, next thing that we can check is density.

(Refer Slide Time: 27:27)



Density value of graph basically tells us whether it is a sparse graph or it is a dense graph with respect to the number of edges present. So, if there are n nodes in a network, the total possible edges that that network can have will be nC_2 . Out of these nC_2 edges how many what is the fraction of the edges that are present in the graph is basically what is stored by the density value.

So, if it is a simple graph the density value will be between 0 and 1, and if it is empty graph the density will be 0 if it is complete graph density will be 1; however, if it is a multi graph where more than one edges are allowed between 2 nodes, in that case density value will be more than one can be more 1 for example, in this diagram you see simple graph with 9 nodes. So, the total possible edges that can be there in this network will be 9 choose 2 which is 9 into 8 divided by 2 that is 36. Now the number of edges that are present in this graph are 11. So, the density is going to be 11 divided by 36 and that is equal to 0.31. So, the graph is not very dense. So, this is the kind of indication that the density value gives us.

(Refer Slide Time: 29:06)

```
In [9]: G = nx.complete_graph(100)
In [10]: nx.density(G)
Out[10]: 1.0
In [11]: H = nx.Graph()
In [12]: H.add_nodes_from([1,2,3,4])
In [13]: nx.density(H)
Out[13]: 0.0
In [14]:
```



So, we can go back to the console and see the density value for few networks for example, let me go here and let us create (Refer Time: 29:04). Let us create a complete graph. So, I am going to write `G = nx.complete_graph` of say 100 nodes. So, since it is a complete graph what should be the density value let us check `nx.density`. So, this is the function density, which is available in networkx, which will give us the density value;

obviously, in case of complete graph it is going to be 1. Let me now create in other graph let me repeat nx.Graph. So, I have not added an a edges into it let me add few nodes here. So, I am going to pass list. So, I am passing only 4 nodes.

Let us check the density value of this network. So, since we have not added any edge; obviously, the density value was going to be 0; and let us go back here and see what is going to be the density value for our network karate network. So, what I will write is print (Refer Time: 30:26) density is. So, n x dot density sorry and I am to pass G here let us go back (Refer Time: 30:40) density is 0.139.

(Refer Slide Time: 30:40)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Density is 0.139037433155
anamika@anamika-Inspiron-5423:~/Desktop$
```

1



So, basically it is sort of parts graph, so that we can check. So, that is about the density, let us go back and see few more things that we can check on these networks.

(Refer Slide Time: 30:55)

Clustering Coefficient

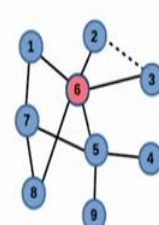

$$\text{Clustering Coefficient} = \frac{\text{Actual Number of friendships}}{\text{Total possible friendships}}$$


Analyzing Real-World Network Data Sets 4 / 6

Next you can see clustering coefficient. So, for a given node clustering coefficient basically tells us the number of lengths that are present amongst the neighbors of this node with respect to the total number of lengths that can be possible.

(Refer Slide Time: 31:15)

Clustering Coefficient contd.


$$\text{Clustering Coefficient}(6) = \frac{1}{10}$$


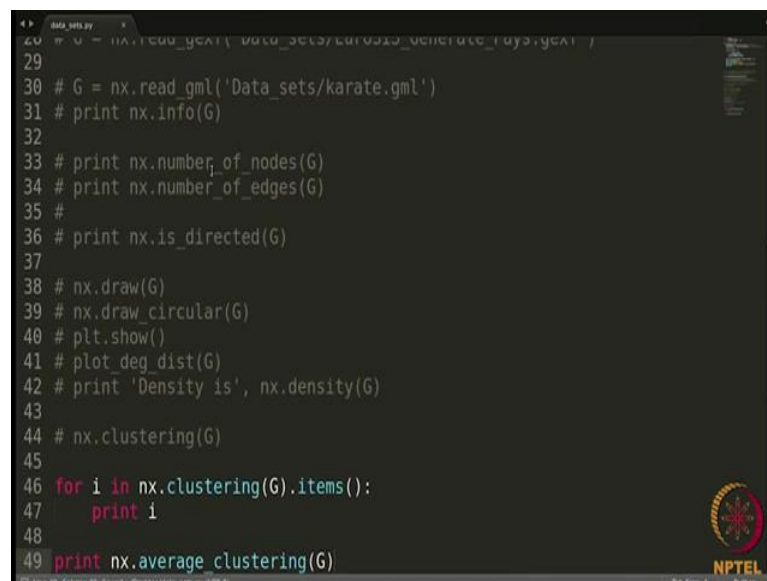
Analyzing Real-World Network Data Sets 5 / 6

I will show you using example let us take this network let us try to find out the clustering coefficient for these nodes 6. So, as you can see this node as 5 neighbors write these are 1 2 3 5 and 8. So, there are 5 neighbors. So, what we have to check here is the number of

links that are present amongst these neighbors, that is the number of links amongst 1, 2, 3, 5 and 8.

As you can see, we see only one link that is there between 2 and 3; there is no other link present amongst these neighbors. So, on the numerator we will put 1 and, on the denominator, we will put the total possible links that that can be there amongst these 5 nodes. So, amongst 5 nodes total possible links can be 5C_2 which is $5 * 4$ divided by 2 that that is 10. So, in this case the clustering coefficient of this nodes 6 will be $1/10$. So, in case of friendship network, this clustering coefficient basically tells us how closely net the friends of node are. So, we can calculate the clustering coefficient value for every node and then we can find the average as well. So, average clustering coefficient for they tell us the amount of clustering present amongst the nodes in the graph.

(Refer Slide Time: 32:44)



```
29
30 # G = nx.read_gml('Data_sets/karate.gml')
31 # print nx.info(G)
32
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 # print 'Density is', nx.density(G)
43
44 # nx.clustering(G)
45
46 for i in nx.clustering(G).items():
47     print i
48
49 print nx.average_clustering(G)
```

So, let us go back and try to check the same for our network. So, I am going to comment this, the function that we can use for finding the clustering is `nx.clustering`; however, this function basically returns a dictionary which gives the clustering coefficient value for every node. So, you can always I treat over this dictionary. So, what I will do is, for `i` in `nx.clustering G`. So, I am interested in all the items. So, I will write dot items I want to print `i`. So, I am going to comment this.

So, what we are doing here is this nx.clustering is returning a dictionary, which contains the clustering coefficient values for all the nodes, we are just going to print them. So, let us run this.

(Refer Slide Time: 33:38)

```
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
(1, 0.15)
(2, 0.3333333333333333)
(3, 0.24444444444444444)
(4, 0.6666666666666666)
(5, 0.6666666666666666)
(6, 0.5)
(7, 0.5)
(8, 1.0)
(9, 0.5)
(10, 0.0)
(11, 0.6666666666666666)
(12, 0.0)
(13, 1.0)
(14, 0.6)
(15, 1.0)
(16, 1.0)
(17, 1.0)
(18, 1.0)
(19, 1.0)
(20, 0.3333333333333333)
```



(Refer Slide Time: 33:45)

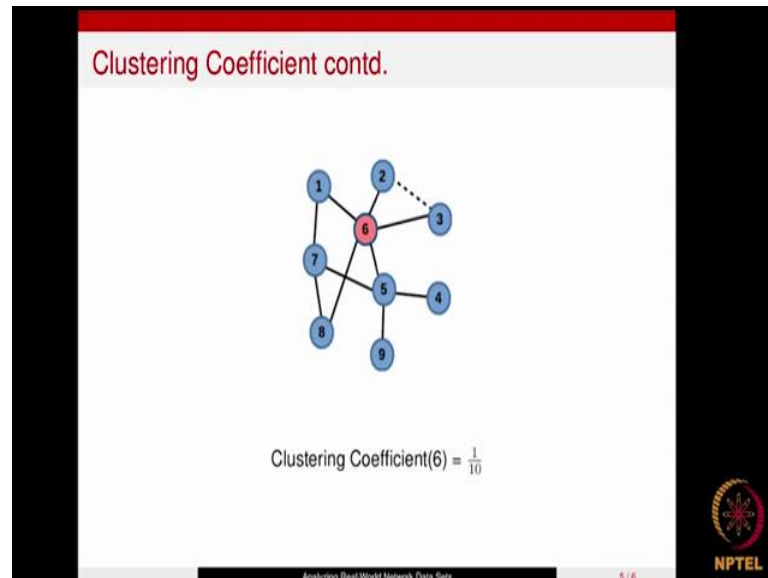
```
(18, 1.0)
(19, 1.0)
(20, 0.3333333333333333)
(21, 1.0)
(22, 1.0)
(23, 1.0)
(24, 0.4)
(25, 0.3333333333333333)
(26, 0.3333333333333333)
(27, 1.0)
(28, 0.16666666666666666)
(29, 0.3333333333333333)
(30, 0.6666666666666666)
(31, 0.5)
(32, 0.2)
(33, 0.19696969696969696)
(34, 0.11029411764705882)
0.570638478208
anamika@anamika-Inspiron-5423:~/Desktop$ python data_sets.py
Diameter is 5
anamika@anamika-Inspiron-5423:~/Desktop$
```



So, we are getting this dictionary, where for every node we are getting the clustering coefficient value. So, if you want the average clustering coefficient, we can either average these values or we can directly make use of another function which is nx.dot average I am sorry average clustering. So, this should tell us the average clustering

present in the network. So, you see 0.57 is an average clustering. So, more this value more the clustering and more title in it the people in the friendship network r.

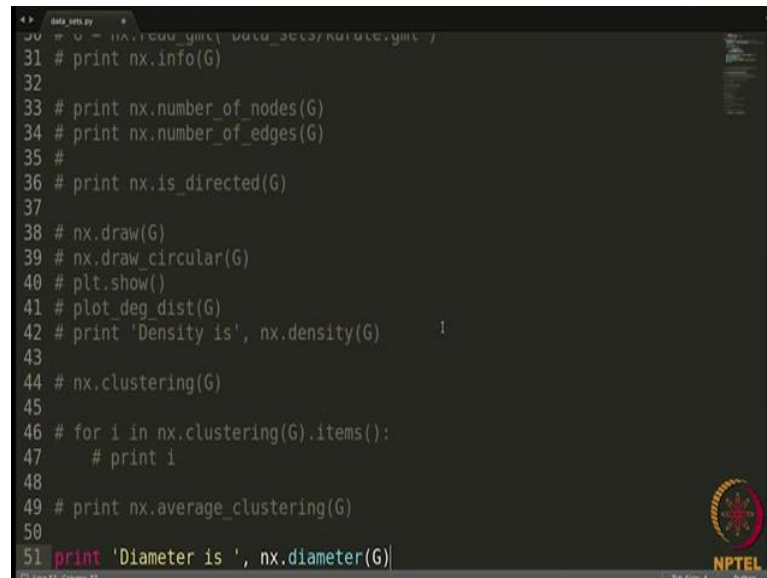
(Refer Slide Time: 34:26)



That was about the clustering coefficient let us go back and check few more properties. So, what is the diameter of a network? Diameter is basically the maximum shortest path that we have to travel to go from one node to the other for example, if you know about all pair shortest path algorithm, it basically the returns the metrics where the values are the length of the shortest path being the 2 nodes. So, is it as that for every pair of the nodes? So, whatever is a maximum value in that metrics will be the diameter of the network.

In other words, it is the shortest path between 2 most distant nodes in the network. So, for example, if you see node 1 and node 9. So, if you have to go from 1 to 9, you will have to traverse this path 1 to 6 to 5 to 9, there is no other shortest path between them. So, the length of this this path is 3, and we do not see any other shortest path which is longer than this 3, if you want to go from 1 to 4 again the length to the path is 3, if you want to go from 3 to 9 the length is 3. So, we do not see any other shortest path which is more than 3. So, that is why the diameter of this network will be 3, we can check the diameter of r network here is well.

(Refer Slide Time: 35:54)



```
31 # print nx.info(G)
32
33 # print nx.number_of_nodes(G)
34 # print nx.number_of_edges(G)
35 #
36 # print nx.is_directed(G)
37
38 # nx.draw(G)
39 # nx.draw_circular(G)
40 # plt.show()
41 # plot_deg_dist(G)
42 # print 'Density is', nx.density(G)
43
44 # nx.clustering(G)
45
46 # for i in nx.clustering(G).items():
47 #     print i
48
49 # print nx.average_clustering(G)
50
51 print 'Diameter is ', nx.diameter(G)
```

So, I am going to comment this and let us check the diameter. So, I will write diameter is. So, I will the function that we can use is `nx.diameter(G)`.

So, it should give us the diameter. So, diameter is 5. So, so there are 34 nodes here, and the diameter is 5; it is basically observe that in real world networks the diameters is basically very less because the nodes are connect to each other and that makes the distance between them very small and let us how the diameter reduces. So, these were just the few point of an analysis that we performed on the networks that we downloaded. That the main thing to notice here is that once you get the network in the networkx graph object you can just play around with it you can apply all the functions that are available, you can just read the documentation and apply the functions which are relevant for that network and you can go ahead with your analysis.