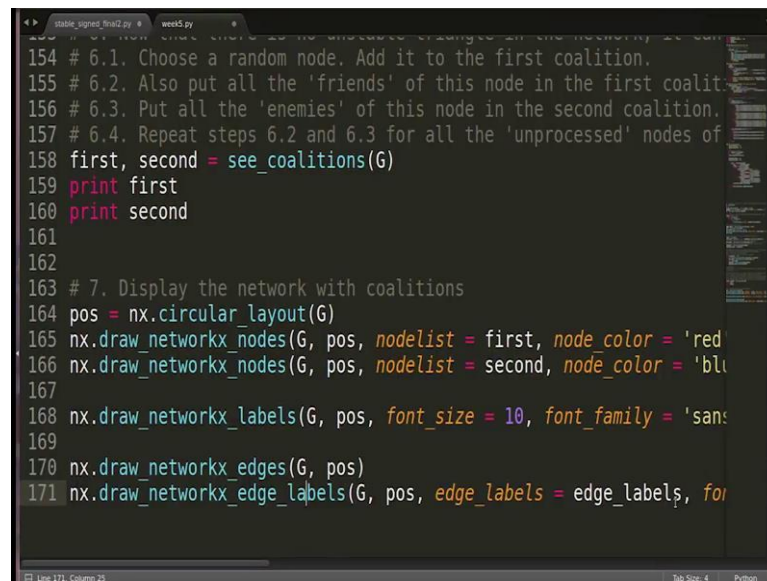


**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**Homophily (Continued) & Positive and Negative Relationships**  
**Lecture – 74**  
**Visualizing coalitions and the evolution**

(Refer Slide Time: 00:00)

A screenshot of a Python code editor window with a dark background. The code is written in a light-colored font and includes comments and function calls for visualizing a network with two coalitions. The code is as follows:

```
154 # 6.1. Choose a random node. Add it to the first coalition.
155 # 6.2. Also put all the 'friends' of this node in the first coalition.
156 # 6.3. Put all the 'enemies' of this node in the second coalition.
157 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
158 first, second = see_coalitions(G)
159 print first
160 print second
161
162
163 # 7. Display the network with coalitions
164 pos = nx.circular_layout(G)
165 nx.draw_networkx_nodes(G, pos, nodelist = first, node_color = 'red')
166 nx.draw_networkx_nodes(G, pos, nodelist = second, node_color = 'blue')
167
168 nx.draw_networkx_labels(G, pos, font_size = 10, font_family = 'sans-serif')
169
170 nx.draw_networkx_edges(G, pos)
171 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 8)
```

Of the implementation where we are going to visualize the communities that is the coalitions that we informed in the previous video. So, one thing that we want to do here is that we want to display the nodes in one coalition by a different color and the nodes in the second coalition by different color; so that we can clearly see the kinds of relationships amongst the nodes inside; the inside the coalition and across the coalition we want to visualize that.

So, what we can do is we can use the same layout circular layout; so we will write `nx.circular layout G`. Now if we write `nx.draw`, it will just draw the network simply without any added features there. Since we want to display the nodes in a different in two different colors, we will have to use separate commands.

So, what we can use here is `nx.draw` in `networkx` nodes; in that function we will pass the lists list of nodes which have to be colored with the single color. So what we can write is

`nx.draw_networkx` nodes the parameters will be first the graph and then the layout and the third parameter which is the most important is the node list.

So, this list will contain the nodes which have to be colored in one color. So, here we want the nodes which are there in the first list this list to be colored in the same colors. So, you will write node list is equal to first we can specify the color that you want node color is equal to say red. And we can also give the size node size is equal to say 5000; so, that was the displaying the nodes in the first list.

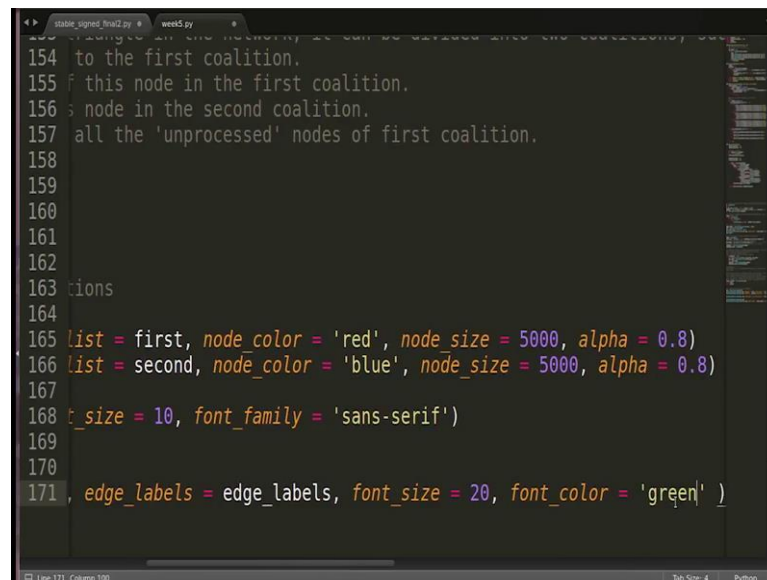
Similarly, we have to display the nodes in the second list we will write `nx.draw_networkx` nodes G and then pos then node list is equal to second. Now node color is equal to say we will change the color here say blue and node size is equal to 1 second (Refer Time: 02:33) node size is equal to 5000, there is one more parameters which we can use that is alpha which basically gives the transparency value for the for the node color.

So, we can keep it and we can change it later if we require; so, as of now I am giving it 0.8. So, now we have drawn the nodes since we have not made use of `nx.draw` the rest of the things will not be drawn by default we have only; so we are manually doing it in a way. So, we have drawn the nodes we have not drawn the edges, we have not drawn the the edge labels, we have not drawn the node label; so we are going to do that one by one ok.

So, let us first write the labels of the nodes, so we will write `nx.draw_networkx` labels. So, again the parameters will be G and the layout and then the font size, font size is equal to 10. And then font we can give the font as well by this parameter font family is equal to say sans serif; I am sorry we will put hyphen here.

Now, we have drawn the nodes and the labels on the nodes; now next thing is the edges. So, we will write `nx.draw_networkx` edges; again, the parameters will be G and pos you can give more parameters as well; as of now I think this is enough. Now we have not yet drawn the labels on the edges; so for that we will use `nx.draw_networkx` edge labels; so this is the command, I think we used it at here as well. So, the parameters again will be G and then the layout and then edge labels is equal to edge labels; we are yet to retrieve the labels firstly, we are going to do that after writing this command.

(Refer Slide Time: 04:47)

A screenshot of a Python code editor window titled 'stable\_signed\_final2.py'. The code is a Jupyter notebook cell, showing lines 154 through 171. The code defines two coalitions, 'first' and 'second', and visualizes them as a network graph. Nodes are colored red and blue, with a size of 5000 and an alpha of 0.8. Edges are labeled with 'plus' or 'minus' signs, with a font size of 20 and a green color. The code uses the 'nx' library for network operations and 'plt' for visualization.

```
154 to the first coalition.
155 if this node in the first coalition.
156 if node in the second coalition.
157 all the 'unprocessed' nodes of first coalition.
158
159
160
161
162
163 tions
164
165 list = first, node_color = 'red', node_size = 5000, alpha = 0.8)
166 list = second, node_color = 'blue', node_size = 5000, alpha = 0.8)
167
168 t_size = 10, font_family = 'sans-serif')
169
170
171 , edge_labels = edge_labels, font_size = 20, font_color = 'green' )
```

So, as we did in the previous representation of the graph; we have to retrieve the labels. If you do not do that by default it displays the attribute name as well for example, sign is equal to plus, sign is equal to minus we do not want that, we only want to display plus and minus there. So, we are going to retrieve the edge labels in the next command and apart from that we can display the font size as well; font size is equal to say 20 and in font color is equal to say green maybe.

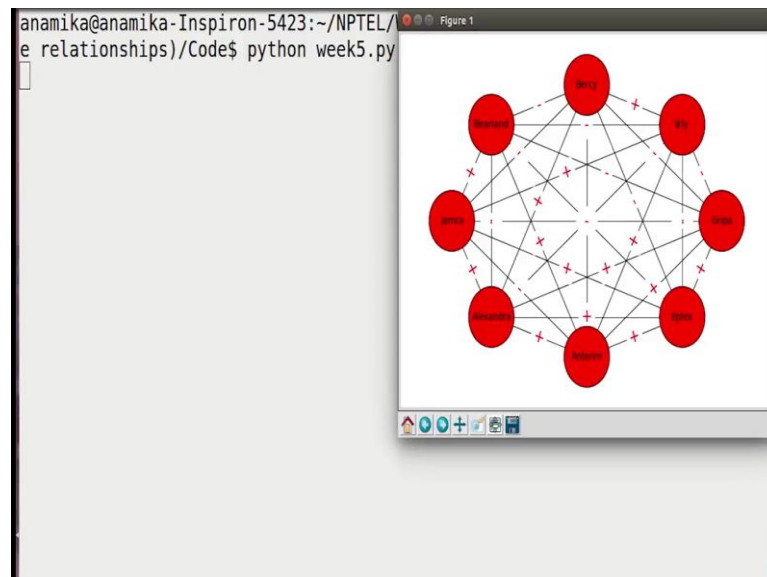
So, the only thing left is retrieving the edge labels ok; for that we will go up here, we will write this command edge labels is equal to `nx.get_edge_attributes`. And the column address will be the graph and the attribute for which we want to retrieve the values. So, this should fetch the latest attributes of sign for all the edges. So, I think we have drawn pretty much everything, and it should work.

(Refer Slide Time: 05:56)

```
157 # 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of
158 first, second = see_coalitions(G)
159 print first
160 print second
161
162 raw_input()
163
164 edge_labels = nx.get_edge_attributes(G, 'sign')
165 # 7. Display the network with coalitions
166 pos = nx.circular_layout(G)
167 nx.draw_networkx_nodes(G, pos, nodelist = first, node_color = 'red')
168 nx.draw_networkx_nodes(G, pos, nodelist = second, node_color = 'blue')
169
170 nx.draw_networkx_labels(G, pos, font_size = 10, font_family = 'sans-serif')
171
172 nx.draw_networkx_edges(G, pos)
173 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_size = 8)
174
175 plt.show()
```

So, we can write `plt.show` let me also add one more thing; let me add a function here `draw_input`. Now what this function does is; it will ask you for an enter before it proceeds. So, we want it to pos for a few moments and then we will press enter and then will display the graph the network ok. Let us save it and we will check here.

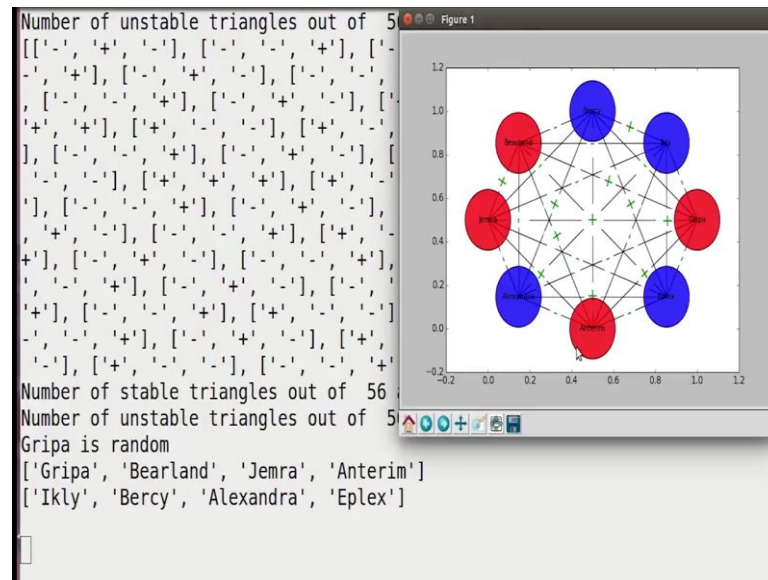
(Refer Slide Time: 06:23)



Let us run this; so, this is the initial graph; the initial network. Let us take an example of an unstable relationship here, let me see Bearland, Alexandra and Anterim this is a triangle which has two positive edges and one negative edge.

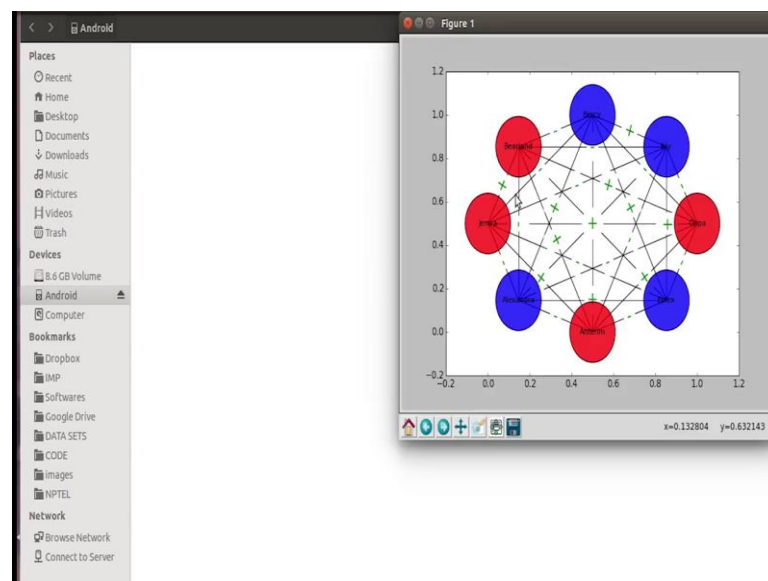
So, this is an unstable relationship this should not stay for a long time in the network. So, let us note down Bearland, Alexandra and Anterim; so, between Bearland and Alexandra there is negative and this two are positive.

(Refer Slide Time: 07:10)



So, let us see what happens to this triangle let me close it and it is now I am pressing enter; this is the final network with the different coalitions colored in a different color.

(Refer Slide Time: 07:19)



So here you see blue color nodes are in the first come in one of the communities I do not I do not remember which community; so, the second coalition nodes are in the red color.

Now, if we look at Bearland, Alexandra and Anterim; this has now become stable by; so, now, there is only one positive edge here.

So, what exactly is happened Alexandra and Anterim which were already friends; they have now turned enemies. So, this is as per our initial configurations that this is the kind of relationship that exists for a long time, but not the previous one. So, that previous one has led to the animosity between the Alexandra and Anterim that is one thing to see. Second thing is, if you look at all the red nodes there are all positive edges amongst them right Bearland and Gripa is positive Gripa and Anterim is positive you can check all the cases.

Similarly, if you took look at all the blue nodes; there are all positive edges amongst them Alexandra, Bercy, Eplex and Icly you see all positive edges are there; they are all friends. However, if you look at the inter edges which is the edges from one coalition to the other coalition; they are all negative which is quite an interesting pattern to see. So, Bercy to Bearland there is negative, Bercy to Gripa is negative Bercy to Jemra is negative; and similarly, Alexandra to Anterim is negative Alexandra to Bearland is negative.

So, precisely we have divided the network into two parts: where people in the first part are all friends, people in the second part are all friends, but these the people in these two parts hate each other; they are all enemies of each other. So, this is a nice and interesting phenomena to observe.