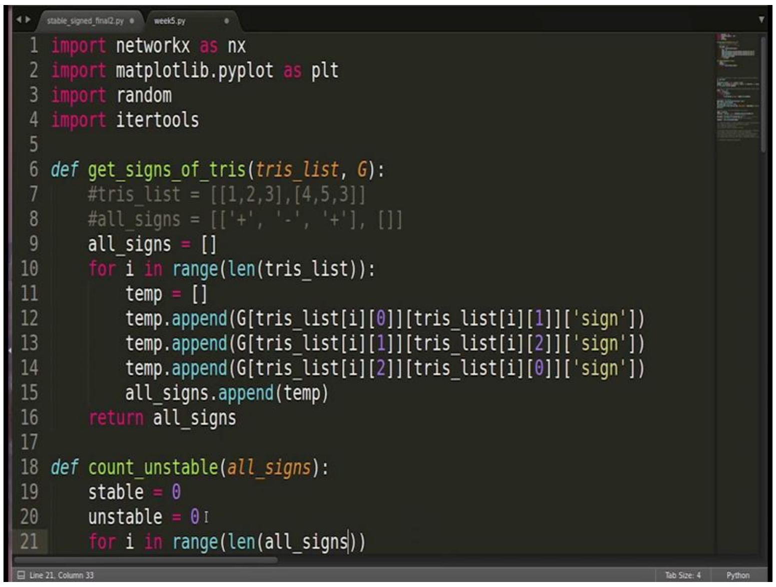
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Homophily (Continued) & Positive and Negative Relationships
Lecture - 70
Creating graph, displaying it and counting unstable triangles

(Refer Slide Time: 00:05)



```python
import networkx as nx
import matplotlib.pyplot as plt
import random
import itertools

def get_signs_of_tris(tris_list, G):
    #tris_list = [[1,2,3],[4,5,3]]
    #all_signs = [['+', '-', '+'], []]
    all_signs = []
    for i in range(len(tris_list)):
        temp = []
        temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
        temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
        temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
        all_signs.append(temp)
    return all_signs

def count_unstable(all_signs):
    stable = 0
    unstable = 0
    for i in range(len(all_signs))
```

Let us get started with the implementation. So, for your convenience I have adding all the steps so here so that we will pick these steps one by one and we will implement them ok. So, initially let me import the packages that we are going to need, since we are going to work with networkx.

Let me import networkx. We are also going to display the network. So, we might need matplotlib so I will import that. We will be randomly choosing the triangles out of all possible triangles. So, we will need the random package. And to choose all possible triangles we might need to use combinations function from itertools package we used it in previous video as well. So, I will import itertools. So, these were the packages.

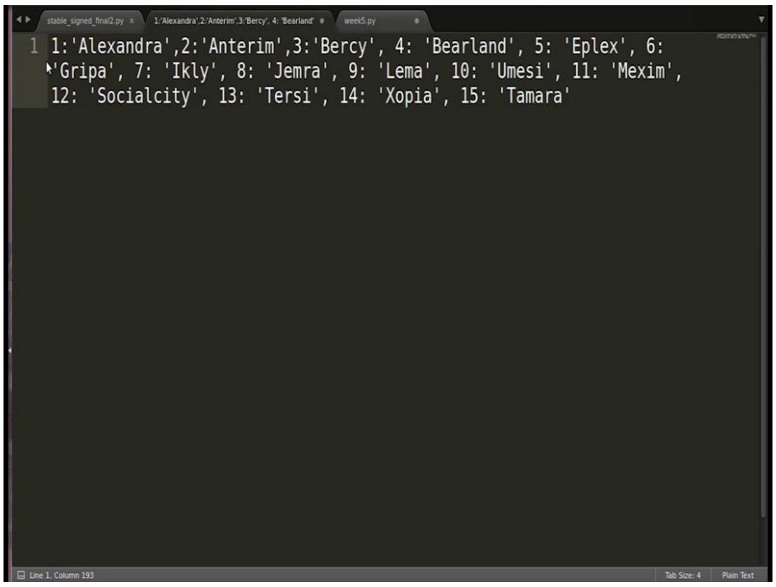
Now let us look at the first step we have to create a graph with n nodes, where the nodes are the countries. So, let me create graph this is an empty graph as of now. So, let me add the nodes to this graph let us initialize this n to 5 and we can change it later. So, we are just going to start with a simple network, where there are 5 countries here increase that later.

So, n is equal to 5, we are going to add 5 nodes to it. So, I will write we have to add 5 nodes. So, let me pass a list of 5 nodes over here as a parameter. So, I will write i for i in range; the nodes should be numbered from 1 to n. So, I can write i to n + 1 because range if it starts from 1 to n + 1 it goes from 1 to n. So, this is what we are going to pass here.

Now we have added the nodes to this graph we have to assign the names to these nodes, the country names to these nodes. So, let me use dictionary; we are we are going to use dictionary for this because we have this function a networkx. That is nx.relabel_nodes; what this function does is? It labels the nodes, because as of now the nodes are labeled from 1 to n right that is 1 to 5 in this case. We have to assign the country name as a label to all these nodes. So, they are going to make use of this function relabel nodes from networkx.

The parameters are two: the first one is a dictionary let me name this dictionary is mapping that is what will pass. This dictionary will contain the labels the new labels of the nodes. So, the key will be the old label and the value will be the new label right. The second parameter is the graph itself. So, we are going to make use of this function, for this we first need a dictionary which has the new labels.

(Refer Slide Time: 03:33)



```
1:'Alexandra',2:'Anterim',3:'Bercy', 4: 'Bearland', 5: 'Eplex', 6:
'Gripa', 7: 'Ikly', 8: 'Jemra', 9: 'Lema', 10: 'Umesi', 11: 'Mexim',
12: 'Socialcity', 13: 'Tersi', 14: 'Xopia', 15: 'Tamara'
```

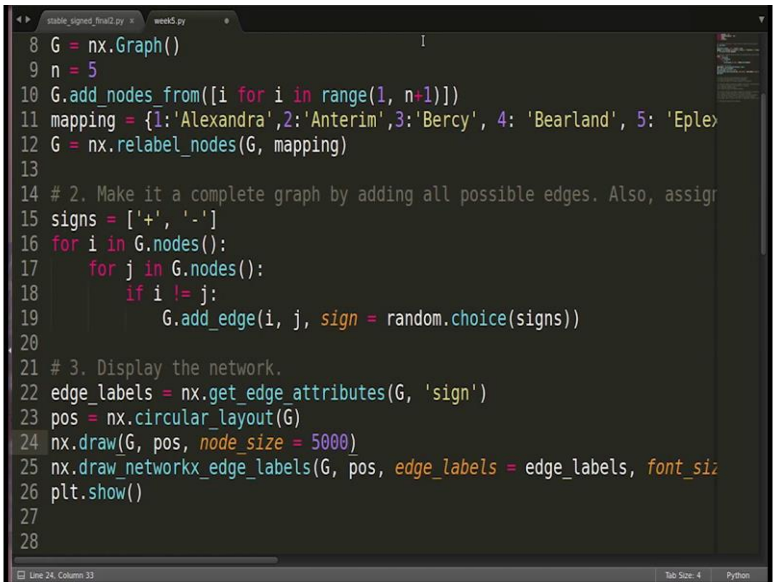
I already have this here; I have some random names of the countries. So, I am going to use these fictitious names for the countries. So, this is our mapping dictionary which we will pass to this graph and the nodes will be relabeled by this function.

So, the first step is over, we have created a graph with n nodes where the nodes are the countries. Let us go to the second step. The second step is we have to add all possible edges to this graph and we also have to assign wait either positive or negative to these edges. So, let us create a list with two weights: positive and negative. So, what we are going to do is: we will randomly choose one of these weights and we will assign to the edges.

Since we have to create a complete graph, we have to add all possible edges. So, how can we do that? I can start the loop here. So, I can write i in G.nodes for j in G.nodes, if i is not equal to j then we will add this edge. So, we will write G.add_edge we have to add the edge between i and j, we also have to assign the weight.

Let us call the attribute sign; sign is equal to. So, we have to randomly choose the sign out of these. So, the function that we can use is random.choice. What this function does is: it chooses, so the parameter is a list that is the signs here out of this the values of this list it randomly chooses one value. So, out of plus and minus it will randomly choose either plus or minus and we will assign to this edge.

(Refer Slide Time: 05:22)



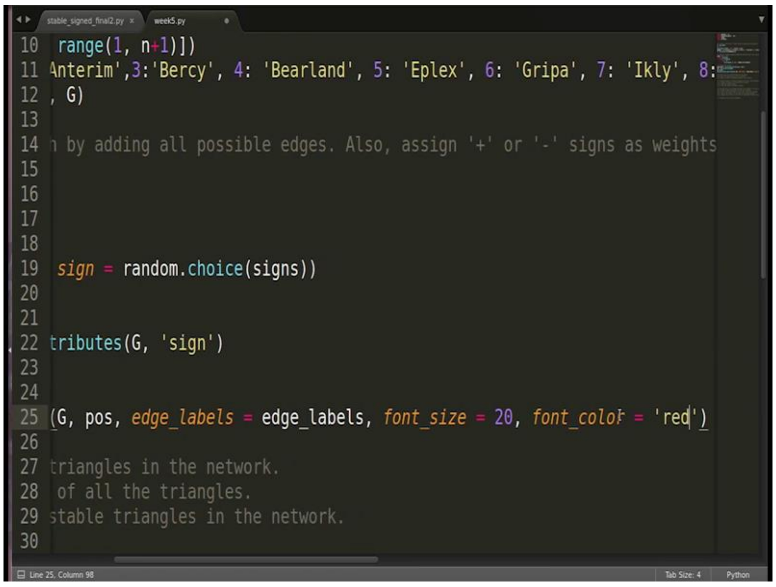
```python
G = nx.Graph()
n = 5
G.add_nodes_from([i for i in range(1, n+1)])
mapping = {1:'Alexandra',2:'Anterim',3:'Bercy', 4: 'Bearland', 5: 'Eplex
G = nx.relabel_nodes(G, mapping)

# 2. Make it a complete graph by adding all possible edges. Also, assign
signs = ['+', '-']
for i in G.nodes():
    for j in G.nodes():
        if i != j:
            G.add_edge(i, j, sign = random.choice(signs))

# 3. Display the network.
edge_labels = nx.get_edge_attributes(G, 'sign')
pos = nx.circular_layout(G)
nx.draw(G, pos, node_size = 5000)
nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_siz
plt.show()
```

So, now we are done with the edges. Next step is to display the network; the command to display the network is nx.draw(G). However, we want to display number of other things on the network. So, we are going to use some extra features of drawing functions from networkx. For example, we can use a layout let me use circular layout so that the nodes are nicely visible. So, I will write nx.circular_layout.

Now, by default the edge labels are not displayed. So, we will have to use an extra function nx.draw networkx edge labels. So, we are going to use this function and I will tell you the parameters to this. Another thing is that when we use this function the labels will be displayed. However, the labels will be like sign is equal to plus sign is equal to minus. We do not want that to be displayed like this, you just want plus we just want minus, we just want the signs not sign is equal to plus. So, for that we will have to use one more extra function. We will write edge labels is equal to nx.get_edge_attributes from G we must take the attributes and the attribute whose values we have to take is sign. Now these edge labels we are going to pass here the first parameter is G itself, the second parameter is the layout that we are using and then we will pass the edge labels.

Edge labels is equal to edge labels which means the values that we are getting out of this function those values only should be displayed here. If you want, you can change the font size you can change the font color.

(Refer Slide Time: 07:25)


stable_signed_final2.py × week5.py

```
 range(1, n+1)])
Anterim',3:'Bercy', 4: 'Bearland', 5: 'Eplex', 6: 'Gripa', 7: 'Ikly', 8:
, G)

n by adding all possible edges. Also, assign '+' or '-' signs as weights




 sign = random.choice(signs))


tributes(G, 'sign')


(G, pos, edge_labels = edge_labels, font_size = 20, font_color = 'red')

triangles in the network.
 of all the triangles.
stable triangles in the network.
```

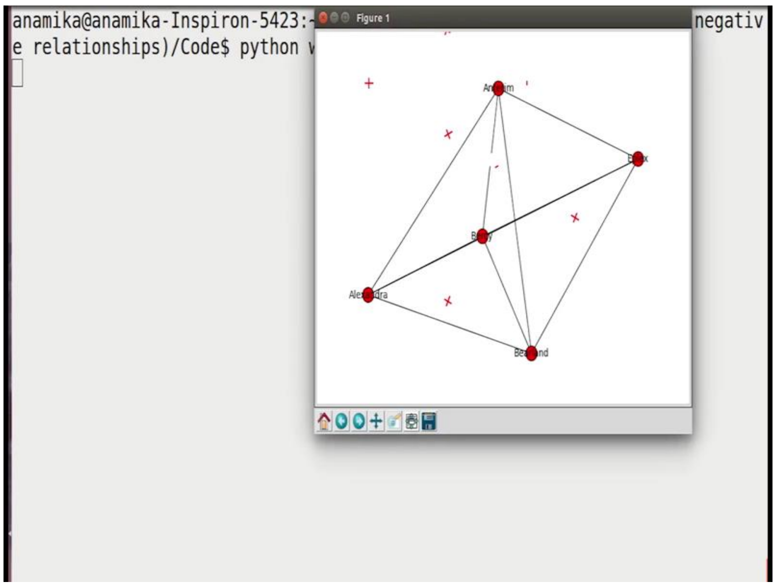
Line 25, Column 98 Tab Size: 4 Python

So, let us use this font size is equal to say 20, you can also change the font color let us do that is well let us make it red. So, these are just some notification parameters you can use them if you want. So, just to show you how they work and going to show then in this video.

I think this should be sufficient. So, to display the network we have to use plt.show. So, to sum up we first added the nodes here, and we really build; oh, I think first we pass the graph and then we pass the dictionary ok. So, first we added the nodes, we relabeled the nodes, then we added the edges, and we assigned weight to these edges, and now we are just going to display this.

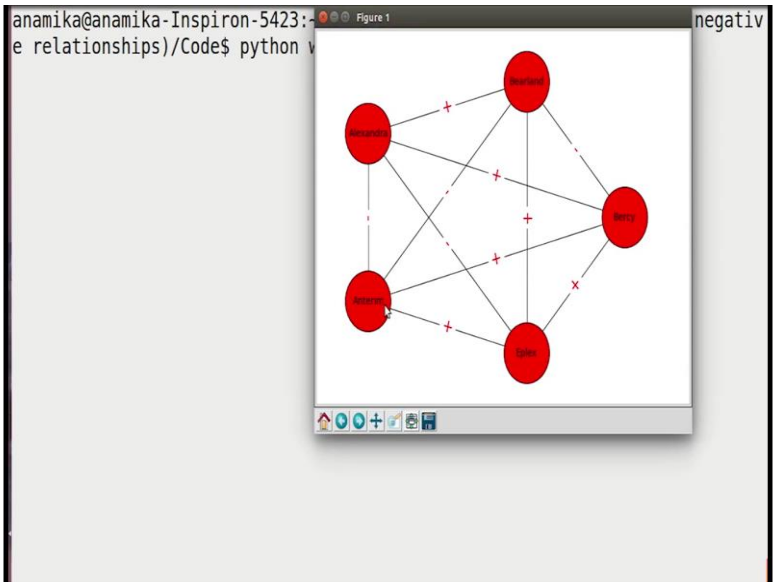
So, let us try executing this, ok.

(Refer Slide Time: 08:18)


```
anamika@anamika-Inspiron-5423:~                                    negativ
e relationships)/Code$ python v
```

So, we go back here and let us execute this ok. So, this is not coming in a circular layout, let us go back here ok. We have not passed this pos here while we are drawing, so we can pass this. Another thing that we can do is we can change the node; the node size is well since you saw that the country names are not visible nicely in the nodes. So, we can change the size. Let me do that node size you can specify any size I am just giving this, ok.

(Refer Slide Time: 08:55)



```
anamika@anamika-Inspiron-5423:~                                    negativ
e relationships)/Code$ python v
```

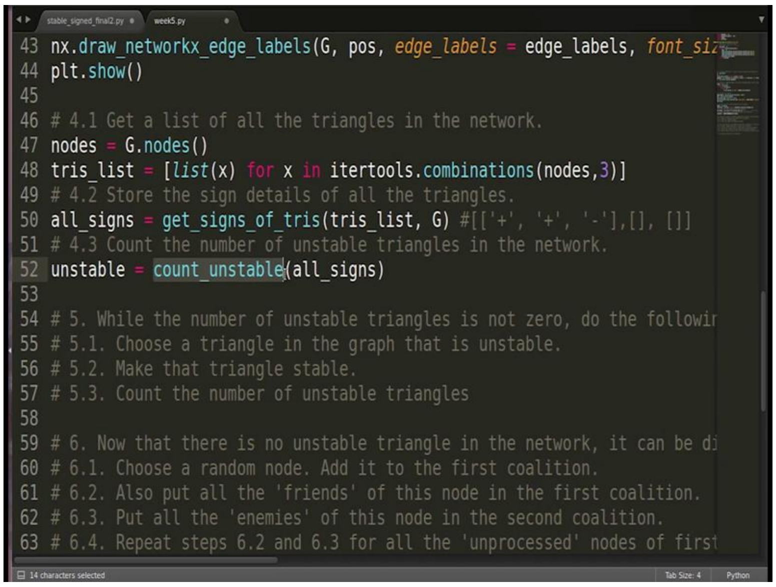
So, let us go back here and try to run it: oh, now this looks better. So, as you can see here. So, these are the 5 countries just for simplicity we have kept a smaller number of countries and will increase them later. So, the edge labels indicate whether they are friends or enemies to each other.

Now, you can see there are some unstable triangles here. For example, Alexandra, Bear land and Bercy you can see that they are all enemies to each other. So, we see that this kind of these kind of relationships do not tend to stay for a long time in the network, they tend to go towards the stable state. That is what we will show in our implementation. So, that is one unstable triangle. Another unstable triangle let us see Alexandra, Anterim, and Bearland you see they are all again enemies to each other that is another unstable triangle.

If you look at Anterim, Eplex and Bercy they are all friends to each other. So, this kind of relationship should stay as it is. So, this is the first three steps completed: we have the network with us, now we have to make it evolve from one state to the other.

Let us close it and go back to our program.

(Refer Slide Time: 10:15)



```python
nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_siz
plt.show()

# 4.1 Get a list of all the triangles in the network.
nodes = G.nodes()
tris_list = [list(x) for x in itertools.combinations(nodes,3)]
# 4.2 Store the sign details of all the triangles.
all_signs = get_signs_of_tris(tris_list, G) #[['+', '+', '-'],[], []]
# 4.3 Count the number of unstable triangles in the network.
unstable = count_unstable(all_signs)

# 5. While the number of unstable triangles is not zero, do the followin
# 5.1. Choose a triangle in the graph that is unstable.
# 5.2. Make that triangle stable.
# 5.3. Count the number of unstable triangles

# 6. Now that there is no unstable triangle in the network, it can be di
# 6.1. Choose a random node. Add it to the first coalition.
# 6.2. Also put all the 'friends' of this node in the first coalition.
# 6.3. Put all the 'enemies' of this node in the second coalition.
# 6.4. Repeat steps 6.2 and 6.3 for all the 'unprocessed' nodes of first
```

So, let us go to the fourth step now.

Now, we have the network with us and we have to observe the triangles. So, let us execute this 4.1 that is get a list of all the triangles in the network. How do we do that? To get all the triangles all the possible triangles we will have to choose three nodes from the list of nodes right, all possible combinations of three nodes. So, for that we first need list of all the nodes.

So, I am going to get that nodes is equal to G.nodes. So, we got a list of all the nodes. Now we have to out of this list we have to choose three nodes at a time. So, let us create a list: tris list which will keep which will keep all the triangles in the network. So, how do we do that? We will create a list and we will make use of combinations function here.

So, let us call it x for x in itertools.combinations out of wish list we have to choose we have to choose out of nodes list. How many choose nodes we have to choose we have to choose three nodes. So, basically what we are doing we are taking out all the combinations of three nodes from this list nodes. And since that is in the form of a tuple, we are going to store that in the form of a list and that list we will store in this list that is tris list.

So, basically this tris list will be a list of list where every item of this list will be a list having the three nodes from the network. These three nodes will indicate a triangle. Now that we have the triangles, we have to see whether these triangles are stable or not, and which ones of these are unstable so that we can you can we can implement moving them towards a stable state.

So, for that we have to see what the weights on the edges on these triangles are. We are going to create a function for that. Before that we will create a list that we will keep track of all the signs of these edges. So, I will create a list and we will call a function we will just create that. We will pass the triangles list in that and we will pass the graph. So, this function we will just create: what this function we will do is it will take the triangles list and the graph as the parameter and it will return a list where the list where this list is a list of list. And every element of this list will have the signing details of the triangles let me write it as a comment here.

For example, this would be a list and this list will have number of other lists like this, and these lists will have the signing details of every edge. For example, this will have plus minus and so on. So, this will be list of lists this all signs which will be return by this functional will just create this function.

Now, go to the next step. Once we have gotten all the sign details of the triangles. We have to check how many unstable triangles are there. So, let me create variable here and let me create a function also: count unstable. We will pass this all signs list here so that this function can tell us how many unstable triangles in all these in all this list are, ok.

So, once these two functions are created, we are will be done with the four steps. Now, let us go back and implement this function first, get signs of triangles. So, we will go up and we will create a function. The parameters as we passed here. We basically need a triangle list and we need the graph as well, so we will pass them here. So, what do we have to do in this function as an input we have tris list? Let me show you the format of tris list. So, it’s going to be list with element lists in it and this list will have the nodes; for example, 4, 5, 3. So, this is an example.

And the output has to be the sign list: like this call this one. So, basically, we have to the triangle is the one which is comprising of the nodes 1, node 2, node 3. So, we have to see the sign between 1 and 2 and we have to store it here, we have to see the sign between 2 and 3 we have to store it here, and we have to see the sign between 2 and 3 and we have to store it here right. So, the output will be something like this for example.

So, this plus indicates that the sign between first and second node of the triangle is plus. This minus indicates that the sign between second and third node of the triangle is this, and this indicates that the sign between first and third node of the triangle is this. So, this is how we are going to create the all signs list and that is what we will return from this function. Similarly, the format of the signal list and all the subsequent lists will be like this only.

So, we will create this entry list that we are going to return. Now for every list inside this what we have to do? We have to do something, so we are going to start a loop for every element list inside this list; so for i in range length of tris list. Now how do we get the sign details of these edges? The sign details are stored in the graph.

So, we are going to take the signs from the graph only. So, all the signs is an empty list it should be returned as this list. So, we have to add a list as an element here. So, I will create our temporary list and I will keep adding values to this list and ultimately we will add this list to all signs. So, how can we get the sign? The sign we will get from G and tris list i-th elements 0th entry and the first entry. This will be the first node.

And the second node will be tris list i-th element first element; I am sorry one will be there ok. So, what we are doing is; we are passing first parameter here which is tris list i-th elements 0th value. So, if this is the i-th element the 0th value will be 1. And second parameter is tris list i-th elements first entry; so i-th elements first entry which is 2 here. So, we want to get the sign of the edge between 1 and 2. So, the third parameter is the sign that we will pass here.

Now whatever sign we get out of here we have to store that in temp. So, I will write temp.append this whole thing. Similarly, we have to get the sign between 2 and 3 and we have to get the sign between 1 and 3. So, I am just going to copy paste this. So, initially this is the sign between 0 and 1, the second will be the sign between 1 and first and second entry and the third will be sign between second and 0th entry right or 0th were second entry this is undirected. So, you can use it either way.

So, by this time we have gotten this temp list which we have to store in all signs. So, I am going to write all signs.append this temp right. So, we have done it for all the all the lists in the tris list, in the end of this list we can in the end of this far loop we can return all signs. This was the implementation of the first function that is get signs of triangle.

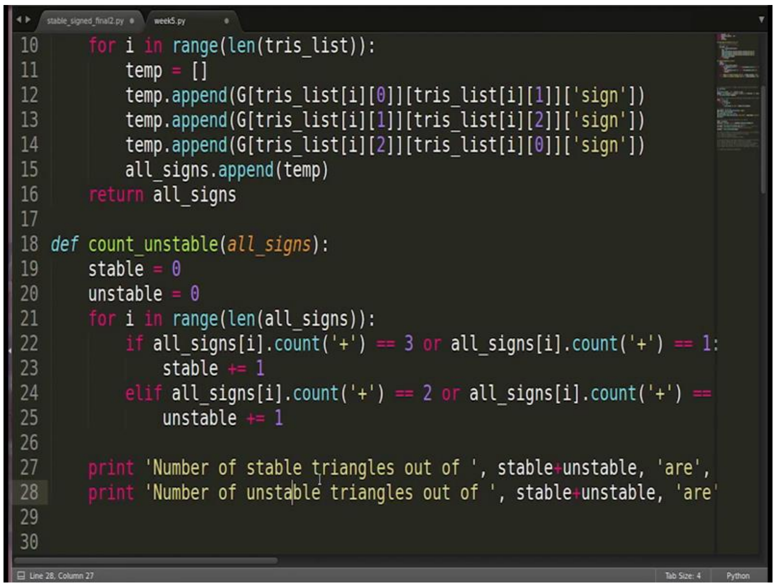
So, we have now stored the signing details of all the triangles. And now let us get back here the second function that we have been using and I am we are yet to implement is count unstable.

So, now, we have all the sign details in this list, all the signs which has been returned by this function. This list we will pass to the next function which will count the number of plus in every list and we will tell us whether it is stable or not and it also we will keep a track of the total number of unstable triangles.

So, let us now implement this function count unstable. Let us go up here, the parameter was all signs right and it is returning variable which is the unstable. So, we will write all signs here and it is returning a variable unstable. Let us also keep a track of stable triangles. So, initialize to 0, Now we have all signs as a parameter, and you know that the format of all signs is like this where every element list has the number of causes plus and minus given in it.

So, we can simply count the number of plus and minus, and we can get to know whether the triangle is stable or not. So, we can start loop here length off all signs I am sorry, length off all signs.

(Refer Slide Time: 21:14)



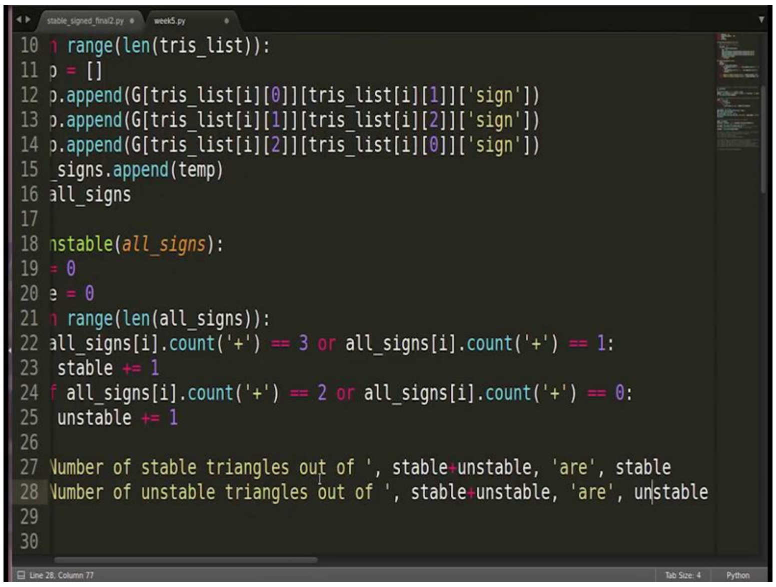
```
    for i in range(len(tris_list)):
        temp = []
        temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
        temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
        temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
        all_signs.append(temp)
    return all_signs

def count_unstable(all_signs):
    stable = 0
    unstable = 0
    for i in range(len(all_signs)):
        if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
            stable += 1
        elif all_signs[i].count('+') == 2 or all_signs[i].count('+') ==
            unstable += 1

    print 'Number of stable triangles out of ', stable+unstable, 'are',
    print 'Number of unstable triangles out of ', stable+unstable, 'are'
```

So, for every element we have to pick one element list from all the signs, and we have to count the number of plus minus in that. Or we can simply count the number of plus as we discussed in the previous video.

So, if all the signs i-th list.count; what do we have to count we have to count plus if it is equal to 3 or if this is equal to 1; then what we have discussed in that case the list will be stable. So, we can increase the stable count in this case. On the other hand, let us use elif. On the other hand, if the total number of plus is equal to 0 or 2 then to be unstable; so, I am just copying pasting here. So, if the total number of plus is 2 or it is 0, then it will be unstable. So, we will increase the unstable count here. So, I think we are done.

So, by the end of this for loop we would have the total number of stable and unstable triangles in the network, which we want to keep track of to know whether the complete network has become stable or not. So, we can print this information so that we can keep track of it. So, let us print. Number of stable triangles; I let us also within the total number of triangles ok. Similarly, I am just doing it so that while executing the code we can keep track of what is going on; number of unstable triangles out of stable plus unstable are unstable.

(Refer Slide Time: 23:27)

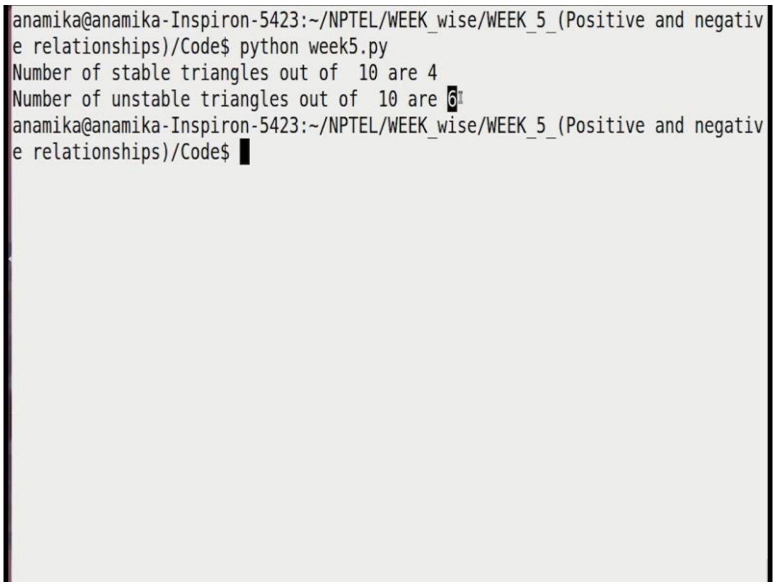


```
ı range(len(tris_list)):
p = []
p.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
p.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
p.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
_signs.append(temp)
all_signs

nstable(all_signs):
= 0
e = 0
ı range(len(all_signs)):
all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
 stable += 1
f all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
 unstable += 1

Number of stable triangles out of ', stable+unstable, 'are', stable
Number of unstable triangles out of ', stable+unstable, 'are', unstable
```

So, I think we are done with these two functions implementation let us go back here and ok.

So, we have implemented this, and we have implemented this. So, it should work fine let us try executing it and see whether it is telling us the stable and unstable triangle count or not. Let us execute this and let us check. This is the graph that we just visualized; I am going to close it ok.

(Refer Slide Time: 23:57)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negativ
e relationships)/Code$ python week5.py
Number of stable triangles out of  10 are 4
Number of unstable triangles out of  10 are 6
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negativ
e relationships)/Code$ 
```

We are getting this number of stable triangles out of 10 or 4. So, total number of triangles are 10, and the stable are 4 and unstable are 6 ok. So, this is working fine, let us go back.

In the next part of the video we will see how we can choose an unstable triangle and how we can move it to a stable state. So, we will do it for all the triangles and the network will eventually become stable.