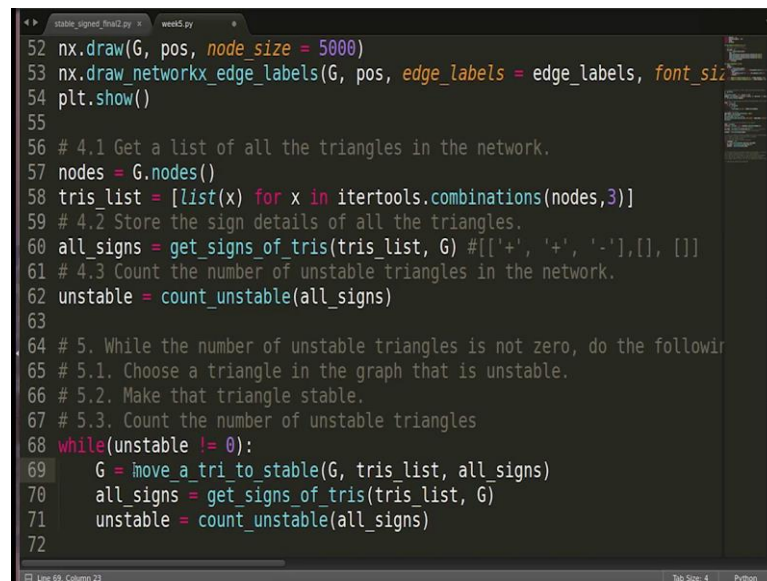**Social Networks**
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Homophily (Continued) & Positive and Negative Relationships
Lecture – 71
Moving a network from an unstable to stable state

(Refer Slide Time: 00:05)


```python
52 nx.draw(G, pos, node_size = 5000)
53 nx.draw_networkx_edge_labels(G, pos, edge_labels = edge_labels, font_si
54 plt.show()
55
56 # 4.1 Get a list of all the triangles in the network.
57 nodes = G.nodes()
58 tris_list = [list(x) for x in itertools.combinations(nodes,3)]
59 # 4.2 Store the sign details of all the triangles.
60 all_signs = get_signs_of_tris(tris_list, G) #[['+', '+', '-'],[], []]
61 # 4.3 Count the number of unstable triangles in the network.
62 unstable = count_unstable(all_signs)
63
64 # 5. While the number of unstable triangles is not zero, do the followir
65 # 5.1. Choose a triangle in the graph that is unstable.
66 # 5.2. Make that triangle stable.
67 # 5.3. Count the number of unstable triangles
68 while(unstable != 0):
69     G = move_a_tri_to_stable(G, tris_list, all_signs)
70     all_signs = get_signs_of_tris(tris_list, G)
71     unstable = count_unstable(all_signs)
72
```

In the previous video we created a network on countries; we also stored the positive and negative signs across the edges. We also computed the total number of unstable triangles in the network. Our next aim is to move this network to a stable state for that what we will do is we will choose one triangle which is unstable, and we will move it to a stable state. We will keep doing this until the complete network become unstable.

So, we have to see how the number of unstable triangles fluctuates and how it changes with respect to each iteration. So, let us get started on that. We have this variable unstable which keeps track of the number of unstable triangles. So, we can start the loop here while this unstable become 0. So, we can write while unstable is not equal to 0 we have to keep repeating this. Now, what do we have to keep repeating?

Since we have to repeat this it will be better if you we create a function for this particular task. So, let me create a function here maybe move a triangle to stable state, we need to pass the graph here because that is what we will get changed during this task. And we

also need to pass all the triangles list we might also need all the signs there. So, let us pass this and in this function, we will make some changes in the graph with respect to the signs of one of the triangles.

So, we too want the updated graph back. So, this is the function we will just implement. Once we get the updated graph that is once one of the triangles has moved to a stable state; the signs of the adjacent triangles might also get affected. So, we need to recompute the signs of the adjacent triangles. So, we can do one thing we can do one of the two things. We can either recompute the signs of the adjacent triangles only or we can recompute the signs of all the triangles.

Here for simplicity I am recomputing the signs of all the triangles. So, basically, I am going to call this function once again; get signs of triangles which we have already created. So, I am going to call this once again on the updated graph G here; also once we get all the signs we need to check how many unstable triangles are there now right. So, we need to get the updated count here. So, we will call this function count unstable once again.

So, now up this variable unstable has the updated count of the number of unstable triangles. So, after that the control will go back here if this is not equal to 0 it will go back again inside; where it will go inside this function; where it will choose one of the one of the triangles which is unstable and we will move it to the stable state and so on. So, this loop will keep running until the entire network become unstable. So, we are done with this part we are just we just have to implement this function or move a triangle to stable.

```
19  able = 0
20  stable = 0
21  r i in range(len(all_signs)):
22    if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23        stable += 1
24    elif all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
25        unstable += 1
26
27  int 'Number of stable triangles out of ', stable+unstable, 'are', stable
28  int 'Number of unstable triangles out of ', stable+unstable, 'are', unst
29
30  ve_a_tri_to_stable(G, tris_list, all_signs):
31  und_unstable = False
32  ile(found_unstable == False):
33    index = random.randint(0, len(tris_list)-1)
34    if all_signs[index].count('+') == 2 or all_signs[index].count('+') ==
35        found_unstable = True
36    else:
37        continue
38
39  ove the unstable triangle to a stable state
```

So, let us see how we can do that; let me copy the syntax let go up and let us create this function. Now we have the graph G, we have the list of all the triangles, we have the list of all the signs on these triangles. Our aim is to choose one triangle which is unstable, now how do we do that? So, what we can do here is; we can choose one triangle out of this list randomly and then we can check whether this triangle is unstable or not. If this triangle is unstable, we have founded a triangle which we have to work on, if this triangle is not stable. I am sorry if this triangle is stable we will check another triangle.

So, we have to keep checking the triangles until we get an unstable triangle. So, how do we do that? Let us create a flag maybe found unstable = false. So, initially we have not found any unstable triangle we will keep running this loop until we find one. So, write while found unstable; while found unstable = false we have to keep running this. We have to keep running what basically we have to choose one triangle randomly out of it.
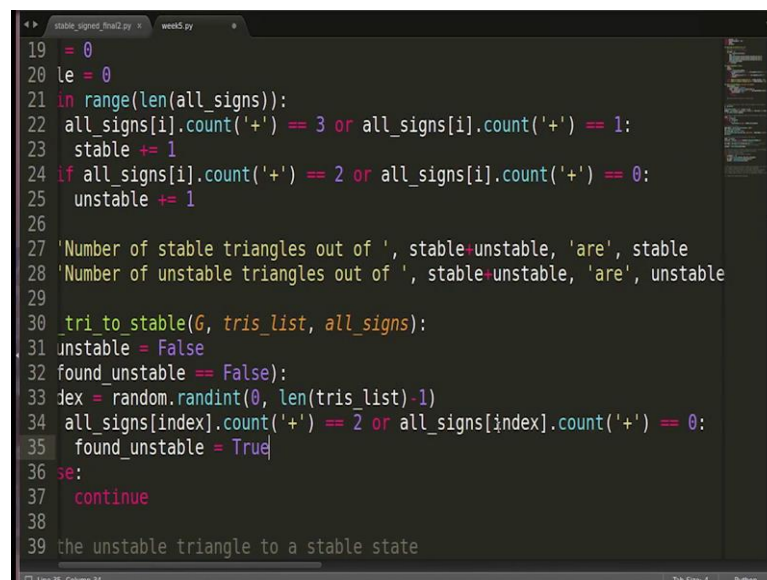
So, what we can do is; we can make use of a function run in which is there in random package. So, maybe index is the variable that will store the index of the triangle. So, so we can write random.randint. So, we can start from 0 length of triangles list minus 1. So, basically what we are doing is this triangle this tris list has some length say 0 to x. We have to choose one index out of 0 to x randomly, right.

So, what we are doing here random.randint this randint function will provide a random integer out of 0 to len(tris_list) - 1. We are doing minus 1 because the next is starting

from 0. So, that index will be stored here; now we have to check whether the triangle which is stored at this index is unstable or not. Now how do we how can we check that? We have this list all signs, which is storing all the signs for every triangle. So, we can check whether the triangle which is existing at this index has 0 pluses 1 pluses 2 pluses 3 pluses. You know what I what I mean. We will count the number of plus their positive sign is there.

So, accordingly we will judge whether the triangle is unstable or not let us see how we do that. So, what we will do if all signs the triangle which is available at this index all signs index.count; we have to count the number of plus there.

(Refer Slide Time: 06:51)


```
19  = 0
20  le = 0
21  in range(len(all_signs)):
22   all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23     stable += 1
24  if all_signs[i].count('+') == 2 or all_signs[i].count('+') == 0:
25     unstable += 1
26
27  'Number of stable triangles out of ', stable+unstable, 'are', stable
28  'Number of unstable triangles out of ', stable+unstable, 'are', unstable
29
30  _tri_to_stable(G, tris_list, all_signs):
31  unstable = False
32  found_unstable == False):
33  dex = random.randint(0, len(tris_list)-1)
34   all_signs[index].count('+') == 2 or all_signs[index].count('+') == 0:
35     found_unstable = True
36  se:
37     continue
38
39  the unstable triangle to a stable state
```

If this = 2 or if this = 0; that means the triangle is unstable. So, we can write flag we have this found_unstable = true; which means we have found an unstable triangle.

In case the number of plus is not equal to 2 or 0, which means the triangle that we found was a stable one so we will continue with our loop. Now that we have found an unstable triangle, now that we have found an unstable triangle, we have to move it to a stable state. Now how do we do that? Firstly, let me write it here move the unstable triangle to a stable state ok.

Now, how do we do this? Firstly, the point to be noted is that there are two kinds of unstable triangles; one kind of unstable triangles are the ones where the number of

positive edges is two. And the other kinds of unstable triangles are the ones where the number of positive edges is 0. So, we are going to take different action in both the cases.
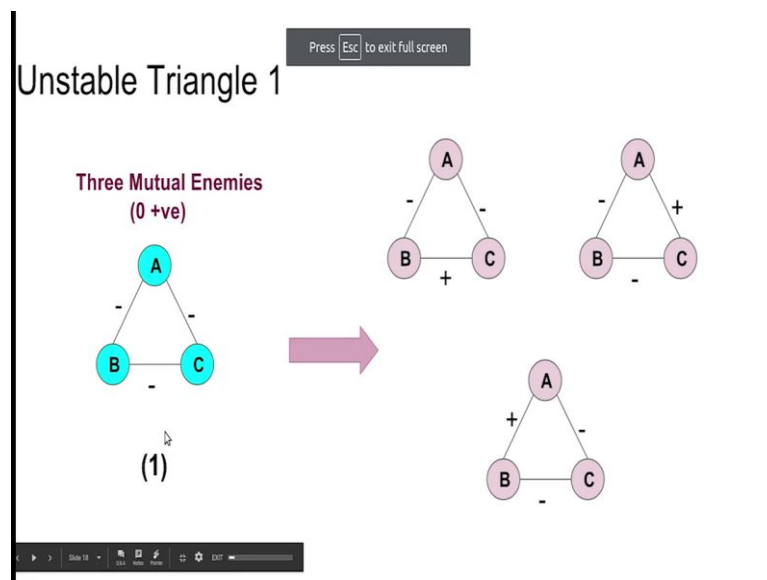
(Refer Slide Time: 08:08)

So, we are going to check in the given triangle the number of positive edges; based on that only we will move it to the appropriate state.

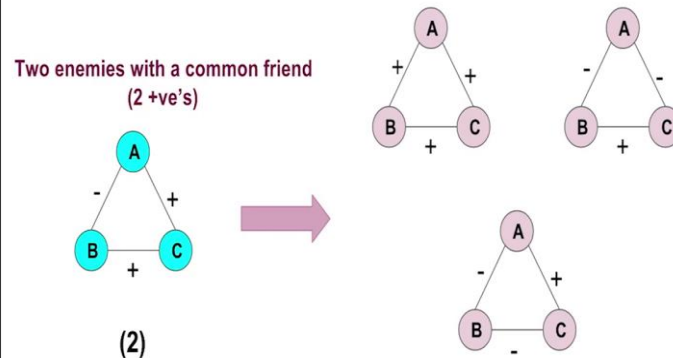(Refer Slide Time: 08:22)

Now, if you remember from one of the previous videos, I had explained the two cases of unstable triangles let me show you once again. So, here you see there are two kinds of unstable triangles unstable triangle 1.

Unstable triangle 2 let us consider this case first. Now this is an unstable triangle where the number of plus is 2; in this case what happens now this state can be moved to any one of these three states. Now, what are these states? In the first case what happens this negative sign between A to B this negative sign has become positive and the rest signs remain the same.

In the second case this minus and plus remain the same, but this plus sign becomes negative this is the second case. In the third case this minus and this plus remain the same here and this plus sign becomes negative.
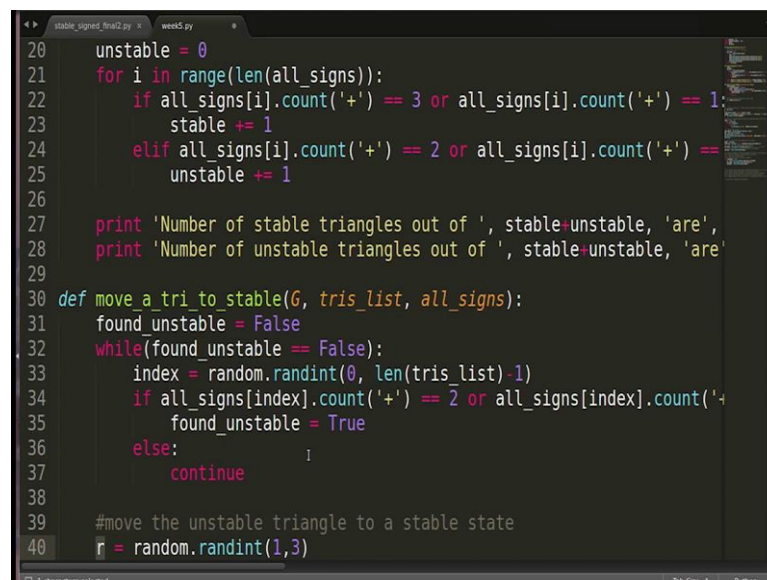
So, basically you see what is happening here in the first case this one this sign is getting inverted. In the second case the sign of the second edge is getting inverted. In the third case the sign of the third edge is getting inverted. So, basically if you take this triangle and choose any one of the edge and invert its sign; whatever you will get that will be a stable state right. So, this is the case of a triangle which has 2 positive edges. If you go back and see the case of the triangle which has 0 positive edges the scenario is a little different here.

So, accordingly we are going to implement as well so let us try to understand this. In this case this triangle which is unstable has 0 positive edges that is all edges are negative. Now these three states are the ones which are stable states which can emerge out of this state. Now what is happening in all these states? In the first state this minus these two

states are these two edges are same and the third edge becomes positive from negative. In the second state this edge becomes positive in the third state A to B edge becomes positive.

So, basically what is happening here is you choose any one of the edges and make it positive; whatever state you get will be a stable state. So, this is how we are going to implement as well. So, you see the moving of unstable state to a stable state is different in both the cases when we have 0 positives or when you have 2 positives. So, we have to keep this thing in mind while we are implementing it. So, let us go back here.

(Refer Slide Time: 11:11)


```python
20      unstable = 0
21      for i in range(len(all_signs)):
22          if all_signs[i].count('+') == 3 or all_signs[i].count('+') == 1:
23              stable += 1
24          elif all_signs[i].count('+') == 2 or all_signs[i].count('+') ==
25              unstable += 1
26
27      print 'Number of stable triangles out of ', stable+unstable, 'are',
28      print 'Number of unstable triangles out of ', stable+unstable, 'are'
29
30  def move_a_tri_to_stable(G, tris_list, all_signs):
31      found_unstable = False
32      while(found_unstable == False):
33          index = random.randint(0, len(tris_list)-1)
34          if all_signs[index].count('+') == 2 or all_signs[index].count('+
35              found_unstable = True
36          else:
37              continue
38
39      #move the unstable triangle to a stable state
40      r = random.randint(1,3)
```

Another thing to note here is that there are three stable states corresponding to one unstable state. So, we will randomly assign one of the three stable states here. So, we will choose a random number here. So, you again use random package random.randint. So, we want a random integer from 1 to 3. So, any one of these three integers can come in r. So, if r = 1 we will move it to the first stable state if r = 2 we will move it to the second stable state and so on.

So, let us handle the first case where the number of a positive signs = 2; let us handle this case first. I am copying pasting here. So, let us see what we do if all_signs[index].count('plus') = 2. So, as I explained you the 3 stable states that can correspond that can come out of this state. So, the strategy that we are going to follow here is that if r = 1; we will invert the sign of the first edge.

If r = 2 we will invert the sign of the second edge if r = 3 we will invert the sign of the third edge. So, that is the approach that we are going to follow here. Now how do we invert the sign of a given edge for that we should know what the sign of that edge is. How do we get to know what is the sign of a given edge that detail is stored in G basically G has the detail of the sign of a given edge? And how do we access the edges? These edges we will access from tris list.

So, let us see how we can do this. So, write if G so we will pass two parameters here and this the third parameter will be sign as usual simple syntax. So, what is going to the first parameter? First parameter is going to be the node the node of the first edge. How do we access the node of the first edge? All the nodes of the triangle are available in tris list. So, we are going to write tris list which index the index the index that we computed here over here. So, tris list index first node and in the second parenthesis what will come I am pasting here.

Tris list index first a value that is the first index basically means the second value ok. Let me explain it here if you find it is difficult. So, tris list is a list of lists and every list is like this ok. So, 1 will be accessed by the 0th index, 2 will be accessed by the first index and 3 will be accessed by the second by the index two which we are going to use in the next sentence. So, this will basically give us 1 and this will basically give us 2.

So, we are going to pass 1 and 2 to this G to get the sign of that G. So, so we will check whether that is positive or not; if that is positive will make it negative then that is negative and make it positive. So, we will just copy paste here. So, this was positive so we will make it negative ok. Now the same has to be done with the rest of the edges as well. So, I will copy paste I remove this; now this is for the second edge.

The second edge will be accessed with index 1 and 2 here again index 1 and 2 ok; so again, if that is positive this one catch here. So, see what we are doing here is; if the sign of the first edge was positive, we are making it negative, but if the sign of the first edge is negative, we have to make it positive as well. So, this thing we are going to do in the next statement.

So, here we are just copy pasting this. So, let us check this if the sign of the first edge was positively made it negative and if it was negative sorry negative. We will make it positive. So, we are done with the first edge and we will handle the first edge if the value
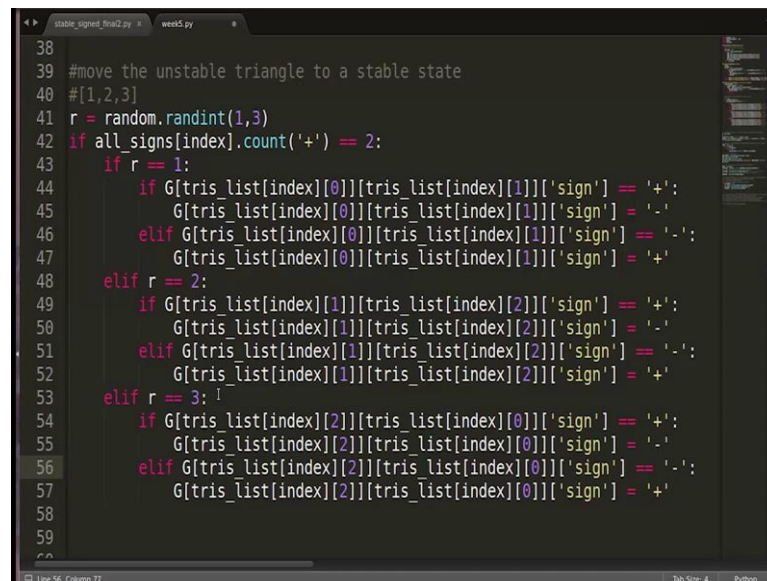
of r was equal to 1. What is the value of r = 2? Then we have to choose the second edge. So, I will write el else if r = 2 we will now handle the second edge.

Now, again we will check the sign of the second edge we just copy paste things here. If the sign of the first edge was positive, we will make it negative and otherwise I will just copy paste again. So, I will write elif if it was negative, we will make it positive. Since we are repeating things here you can also create a function over here. So, that you pass the values and it makes the changes I am just copy pasting here. And the third thing is if I if the value of r = 3 we will choose the third edge and we will invert its sign.

So, again I will copy paste the entire thing. So, I will write if r = 3 now we have to access the third edge. So, we will write 2 comma 0 or you can write 0 comma 2; write 2 0 like here like here.

(Refer Slide Time: 17:49)
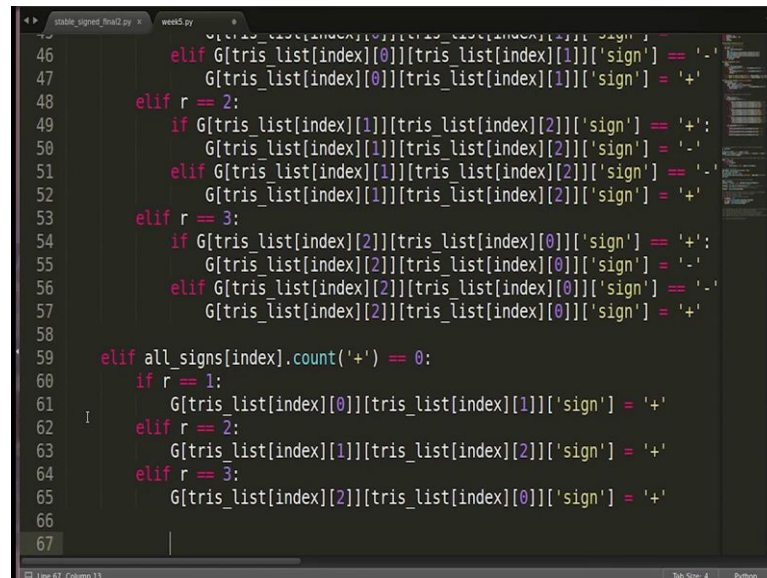

```python
38
39  #move the unstable triangle to a stable state
40  #[1,2,3]
41  r = random.randint(1,3)
42  if all_signs[index].count('+') == 2:
43      if r == 1:
44          if G[tris_list[index][0]][tris_list[index][1]]['sign'] == '+':
45              G[tris_list[index][0]][tris_list[index][1]]['sign'] = '-'
46          elif G[tris_list[index][0]][tris_list[index][1]]['sign'] == '-':
47              G[tris_list[index][0]][tris_list[index][1]]['sign'] = '+'
48      elif r == 2:
49          if G[tris_list[index][1]][tris_list[index][2]]['sign'] == '+':
50              G[tris_list[index][1]][tris_list[index][2]]['sign'] = '-'
51          elif G[tris_list[index][1]][tris_list[index][2]]['sign'] == '-':
52              G[tris_list[index][1]][tris_list[index][2]]['sign'] = '+'
53      elif r == 3:
54          if G[tris_list[index][2]][tris_list[index][0]]['sign'] == '+':
55              G[tris_list[index][2]][tris_list[index][0]]['sign'] = '-'
56          elif G[tris_list[index][2]][tris_list[index][0]]['sign'] == '-':
57              G[tris_list[index][2]][tris_list[index][0]]['sign'] = '+'
58
59
```

So, if it was positive, we made it negative if it is negative, we made it positive. So, all set; now this was the case of the triangles where the number of positive edges were 2.

Next, we have to handle the case where the number of positive edges = 0. In that case as I told you what we will do we will choose any of the edge since all the edges are negative already, we will choose any one of the edges and we will make it positive. So, the resultant state is a positive is a stable state. So, what I will do is; we will write in the else part, if I know this if ok.

If the number of a positives edge = 0; what we are going to do ok. Again, there can be three cases here based on the value of r we will choose the edge. So, we will write if r = 1 we will choose the first edge and I we will do that else if r = 2 we will choose the second edge else if r = 3 we will choose a third edge. Now how do we choose the edges the same way that we have done. So, like you can choose like this.

Now, there is no checking here you just have to make the sign positive; because we know that already the sign is negative. So, I will write it here the sign of the first edge we are making to be positive ok. If r = 2 we will choose the second edge and second edge will be accessed by a 1 and 2; here we will make it positive. And if r = 3 we will choose the third edge and that will be accessed by 2 and 0 here ok.

So, by doing this we have handled both the cases of unstable triangles and we have. So, basically what we have done in this function we have chosen 1 unstable triangle over here ok. We have found the index of a triangle which is unstable then we are checking whether; what is the kind of that unstable triangle; based on that we are taking the action. Let us go back sorry based on that we are taking action if the signs are 2 we are executing this if the signs are 0 positive signs is 0 we are executing this.

(Refer Slide Time: 20:24)

So, we are done here and then we will return the updated graph here.

(Refer Slide Time: 20:32)

Let us go back here and see if everything works fine. So, here we are calling this function which we have just now implemented and after calling this function, we are again computing all the signs, and we are counting the number of unstable triangles. This function is returning this variable unstable we can also keep track of this variable.
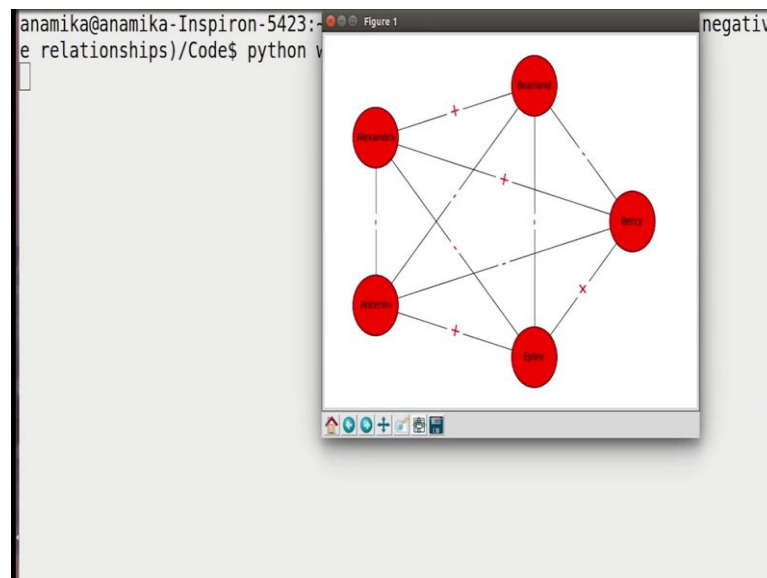
(Refer Slide Time: 20:57)


```python
10  for i in range(len(tris_list)):
11      temp = []
12      temp.append(G[tris_list[i][0]][tris_list[i][1]]['sign'])
13      temp.append(G[tris_list[i][1]][tris_list[i][2]]['sign'])
14      temp.append(G[tris_list[i][2]][tris_list[i][0]]['sign'])
15      all_signs.append(temp)
16  return all_signs
17
18  count_unstable(all_signs):
19  stable = 0
20  unstable = 0
21  for i in range(len(all_signs)):
22      if all_signs[i].count('+') == 3 or all_signs[i].count('+') ==
23          stable += 1
24      elif all_signs[i].count('+') == 2 or all_signs[i].count('+') ==
25          unstable += 1
26
27  print 'Number of stable triangles out of ', stable+unstable, 'are',
28  print 'Number of unstable triangles out of ', stable+unstable, 'are
29  return unstable
```

This one statement missing here I am sorry; we did not return this variable.

We have to return unstable here.

(Refer Slide Time: 21:19)

So let us check, let us verify our code we can check this. So, so this is the network that we had plotted that we had drawn earlier.

(Refer Slide Time: 21:28)

This is the output you see initially the number of unstable triangles was 4 and then it became 3 and then it became 0. So, this was very small network. So it happened it converged quite quickly.
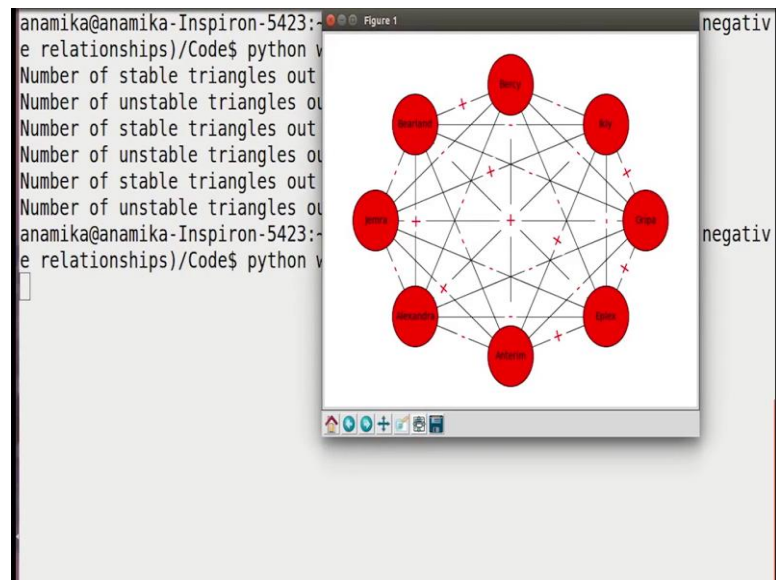
(Refer Slide Time: 21:47)

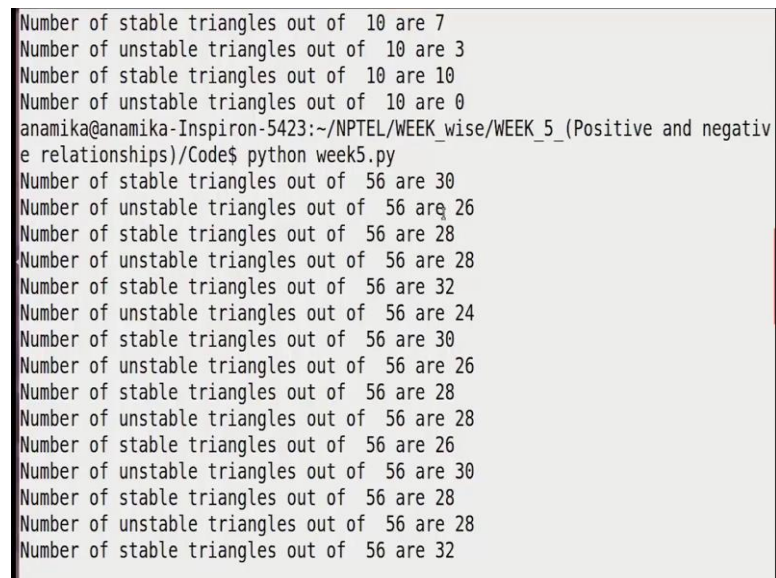Let us for example, increase this number of countries to say 8 and see how it works.

(Refer Slide Time: 21:53)

So, this is the network that gets created out of 8 countries and these are the edges I am closing it.

(Refer Slide Time: 22:05)

Here you see; you go up the number of unstable triangles was the 26 then it became 28 you see. After converting 1 unstable triangle to a stable one; the number of unstable triangles increased to 28 then it reduced then it increased, increased, increased and then it decreased. So, as you can see it kept fluctuating.

```
Number of unstable triangles out of  56 are 22
Number of stable triangles out of  56 are 34
Number of unstable triangles out of  56 are 22
Number of stable triangles out of  56 are 32
Number of unstable triangles out of  56 are 24
Number of stable triangles out of  56 are 36
Number of unstable triangles out of  56 are 20
Number of stable triangles out of  56 are 40
Number of unstable triangles out of  56 are 16
Number of stable triangles out of  56 are 40
Number of unstable triangles out of  56 are 16
Number of stable triangles out of  56 are 44
Number of unstable triangles out of  56 are 12
Number of stable triangles out of  56 are 46
Number of unstable triangles out of  56 are 10
Number of stable triangles out of  56 are 50
Number of unstable triangles out of  56 are 6
Number of stable triangles out of  56 are 56
Number of unstable triangles out of  56 are 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negativ
e relationships)/Code$
```

So, it keeps fluctuating, and, in the end, it became 10 and then quickly 0 and then quickly it became 0.

So, if you want you can also plot this let me show you quickly how we can do that. So, basically, we want to see how the number of unstable triangles is changing when we keep moving 1 unstable triangle to a stable state in each iteration. So, this is the count of the unstable triangles; let us create a list and keep storing the number of unstable triangles there. So, let me create a list here unstable track and initially let it contain the number of the initial number of unstable triangles.

And as and when in every iteration we are getting the number of unstable triangles we will keep appending this number unstable to this list alright. So, we will write unstable track sorry unstable track.append unstable. So, this unstable track is a list which has all the number of unstable triangles with respect to each iteration and if you want to plot you can plot that as well.

(Refer Slide Time: 23:46)


```python
101  unstable = count_unstable(all_signs)
102  unstable_track = [unstable]
103
104  # 5. While the number of unstable triangles is not zero, do the fol
105  # 5.1. Choose a triangle in the graph that is unstable.
106  # 5.2. Make that triangle stable.
107  # 5.3. Count the number of unstable triangles
108  while(unstable != 0):
109      G = move_a_tri_to_stable(G, tris_list, all_signs)
110      all_signs = get_signs_of_tris(tris_list, G)
111      unstable = count_unstable(all_signs)
112      unstable_track.append(unstable)
113
114  raw_input()
115  plt.bar([i for i in range(len(unstable_track))], unstable_track)
116  plt.show()
117
118
119  # 6. Now that there is no unstable triangle in the network, it can
120  # 6.1. Choose a random node. Add it to the first coalition.
```

So, we can do plt.plot if you want to bar chart you can do that also.

So, the function is plt.bar and it requires 2 inputs the first will be the values on it on the x axis. So, we can have the values to start from 0 on the x axis. So, maybe we can pass the list here I can write I for I in range we can just pass the length of this list here so that the number of parameters match. So, this was our x axis and the y axis will of course be unstable track and then we can show it plt.show.

So, we can check whether it works before that we can use this function. So, that it us for an enter before it precedes. So, let us check the function we can.
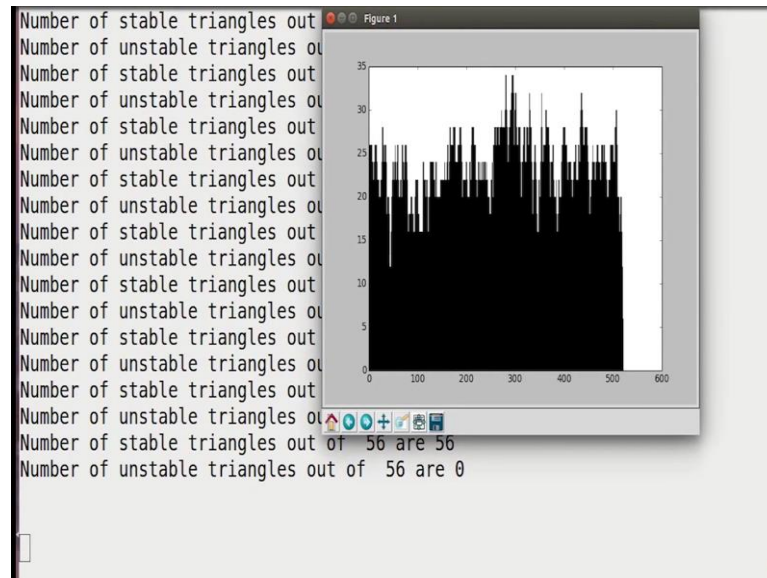
(Refer Slide Time: 24:42)

So, this is the network.

(Refer Slide Time: 24:47)

Let us close it and this is the how the number of unstable triangles changed.

And I am just pressing enter here and this is the plot that we got. So, as you can see the fluctuations that happened in the number of unstable triangles; I have not written anything on the x and y axis, but you can understand the y axis means the number of unstable triangles.

So, it reach reached here somewhere here and then quickly came to 0. So, that is how it happens let me close it now. Let us go back.

Now, if you want you can also keep printing this list which is all signed so that you can see how the signs of the triangles are changing. So, let me quickly show you that I am printing all signs. Since we have already seen the plot. I am just commenting this part and let me increase the number of cities to say 10, alright.

So, basically, we want to see the signs how they are changing from the initial configuration through the last configuration where all the triangles are stable.

(Refer Slide Time: 25:56)
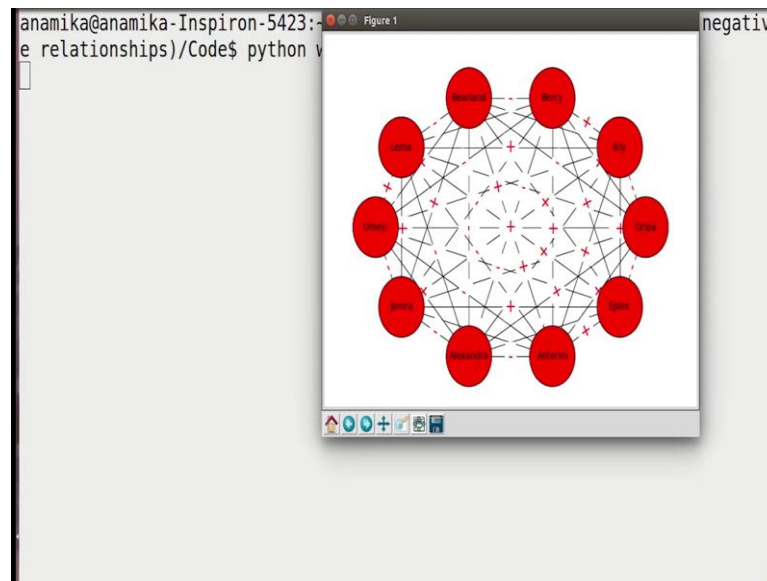
So, let us check this is the network with ten countries. So, it is taking some time as you keep increasing the countries it takes a whole lot of time from 8 to 10; it is taking a quite a lot of time; it is taking some time.

So, as you can see this is the list, which is all signs and once it stops, I will show you some examples here; it is actually taking a lot of time ok. I think I can pause the video and show you the final result ok; I will show you the final result now.

You see this is the final result: in the end out of one twenty triangles we in the end we have all the stable triangles, and if you go back I am just going a little up.

(Refer Slide Time: 26:59)

Here, you can see a number of unstable triangles for example, here there are 2 positives. So, this is one of the unstable triangles and there are 0 positives. Here this is this is another unstable triangle. However, over time it keeps evolving and in the end whatever you get. So, this all list is completely stable there, there is no triangle which has 0 or 2 positive signs. So, this is how it looks like we can go back here.

So, we done with evolving the network from us from an unstable state to a stable state As I told you once a network becomes stable it can be divided into two groups two coalitions I am not calling them communities for a reason because in we did community division in one of the previous videos. So, you should not get confused with that algorithm and this method, because community is basically; community detection is completely different algorithm depending on the number of links. Here the network is complete, here we are concerned with the signs of the edges, so that is why we are not calling them division in two committees recalling them division to coalitions or groups.

So, in the next video we are going to divide this network into two groups given that it has become stable now.