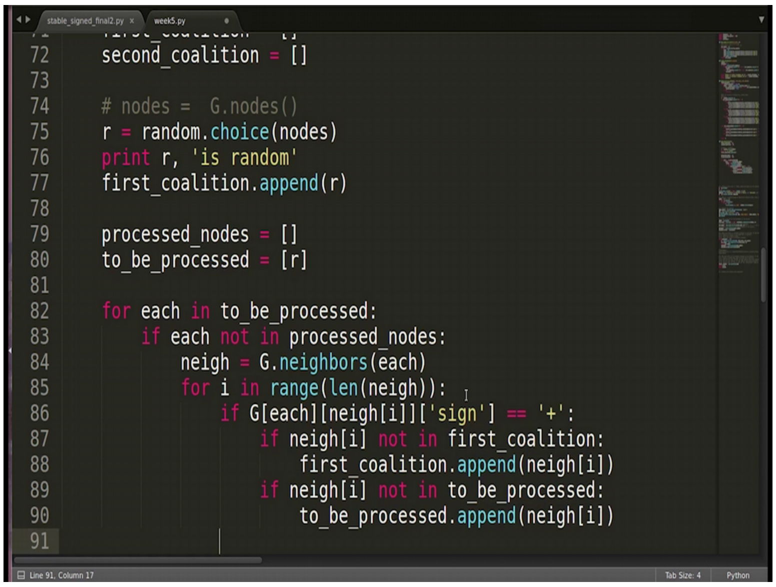
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Homophily (Continued) & Positive and Negative Relationships
Lecture – 73
Forming two coalitions (Continued)

(Refer Slide Time: 00:05)



```python
    second_coalition = []

    # nodes =  G.nodes()
    r = random.choice(nodes)
    print r, 'is random'
    first_coalition.append(r)

    processed_nodes = []
    to_be_processed = [r]

    for each in to_be_processed:
        if each not in processed_nodes:
            neigh = G.neighbors(each)
            for i in range(len(neigh)):
                if G[each][neigh[i]]['sign'] == '+':
                    if neigh[i] not in first_coalition:
                        first_coalition.append(neigh[i])
                    if neigh[i] not in to_be_processed:
                        to_be_processed.append(neigh[i])
```

We will take nodes from this list to be processed and we will keep checking their neighbors. So, we will start a loop for each in to be processed, if each not in process node. So, if this node has not already been processed, if each not in processed node then we will look at his neighbors; so let us have this list neigh is equal to G dot neighbors we have to look at the neighbors of each. So, we will write like this.

Now we have take these neighbors one by one and we have to see whether these neighbors are friends or enemies. If they are friends they will be added to the first list, if they are enemies they will be added to the second list. Apart from that if they are friends and they are being added to the first list, they have to be processed afterwards. So, they will be added to the list to be processed.

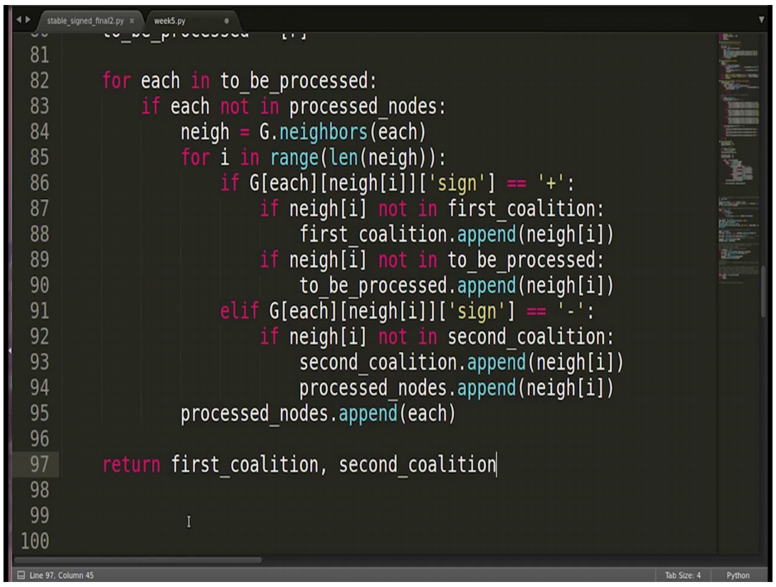
And if they are enemies and they are being added to the second list, then they will not be processed further. So, they will be added to the list process nodes. So, we will write for i in range, length of this list because to get to know how many neighbors are there of course, in this particular example there are n minus 1 precisely n minus 1 neighbors, but we can generalize this and we can write up to length of neighbors.

So, once we take the neighbor, we have to check the relationship of this neighbor with the node that is each. So, we will write G each to neigh of i and how do we check the relation by using this attribute sign. So, we will start this if this is equal to plus which means its a friend, as I told you what has to be done it has to be added to the list of first coalition.

So, if it is not already there. So, we will check if neigh i not in first coalition, then it will be added there. So, you write first coalition dot append neigh i done. And second thing that has to be done is it has to be added to the list of to be process nodes if it is not all be there. You will write if neigh i not in to be processed and it will appended there to be processed dot append neigh i alright

And in the other case when the neighbor is an enemy, what we are going to do is.

(Refer Slide Time: 02:55)



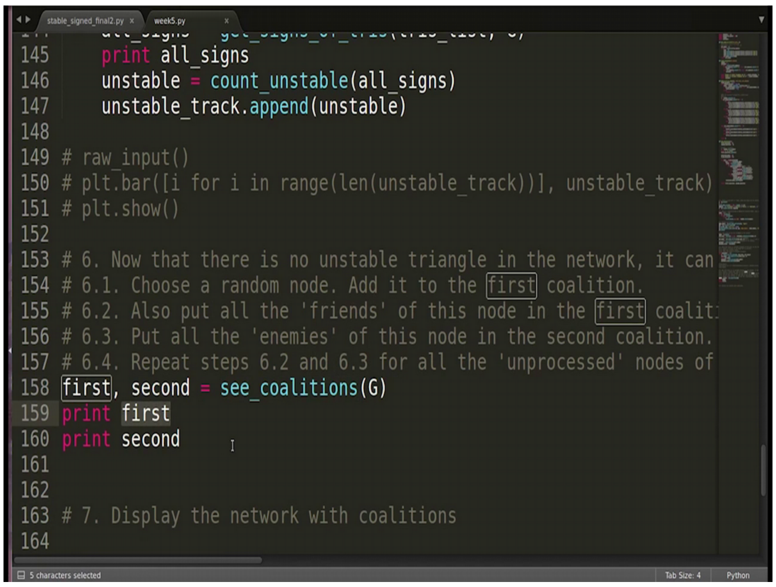
```
    for each in to_be_processed:
        if each not in processed_nodes:
            neigh = G.neighbors(each)
            for i in range(len(neigh)):
                if G[each][neigh[i]]['sign'] == '+':
                    if neigh[i] not in first_coalition:
                        first_coalition.append(neigh[i])
                    if neigh[i] not in to_be_processed:
                        to_be_processed.append(neigh[i])
                elif G[each][neigh[i]]['sign'] == '-':
                    if neigh[i] not in second_coalition:
                        second_coalition.append(neigh[i])
                        processed_nodes.append(neigh[i])
            processed_nodes.append(each)

    return first_coalition, second_coalition
```

So, will write elif and this thing which means this is minus, so its an enemy. What has to be done? It has to be added to the second coalition list if it is not already there. So, we will write if neigh i not in second coalition, it will be appended neigh of i ok. Second in that has to be done is as I told you it does not have to be processed again. So, it has to be added to the list of processed nodes.

So, we can directly just add it there, processed nodes dot append neigh of i. Next thing that has to be done is this each the node which is called each has been process now. So, it has to be added to the list of process nodes. So, we will write here processed nodes dot append each ok. So, that is done

So, there this loop should basically divide the nodes in the two lists that is first coalition and second coalition and that is what this function will return. So, we are going to return this. So, we will write return first coalition comma second coalition this should work.

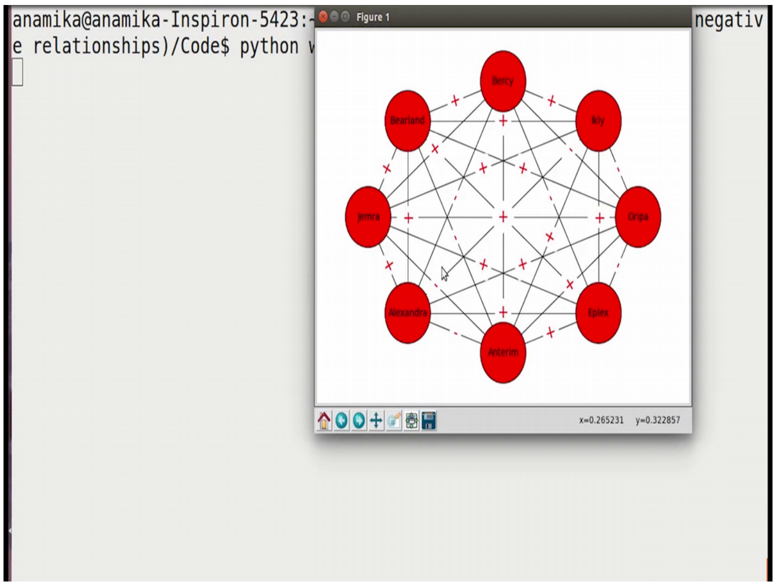
(Refer Slide Time: 04:25)



```
    print all_signs
    unstable = count_unstable(all_signs)
    unstable_track.append(unstable)

# raw_input()
# plt.bar([i for i in range(len(unstable_track))], unstable_track)
# plt.show()

# 6. Now that there is no unstable triangle in the network, it can
# 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 coalit
# 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, second = see_coalitions(G)
print first
print second


# 7. Display the network with coalitions

```

Let us go down and see this is where we call this function and after it returned the two lists, we are just spending those two lists. So, let us see how it works ok.

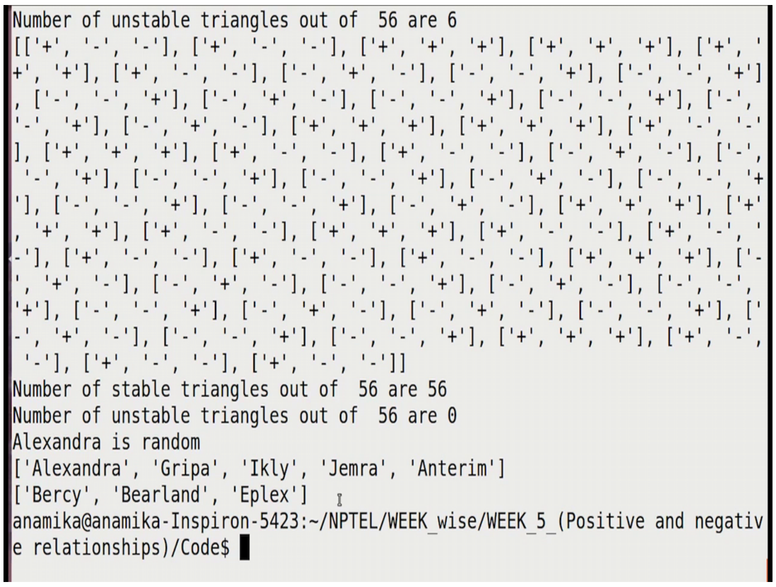
(Refer Slide Time: 04:32)



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

These are the air cities and you see the relationships amongst them as well positive and negative relationships I am closing it.

(Refer Slide Time: 04:41)



```
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
Alexandra is random
['Alexandra', 'Gripa', 'Ikly', 'Jemra', 'Anterim']
['Bercy', 'Bearland', 'Eplex']
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_5_(Positive and negativ
e relationships)/Code$ █
```

So, it has processed and you see these are the two coming the two coalitions formed in the first one there are 5 countries and in the second one there are 3 countries. So, this is the division

In the next video we are going to visualize this network and we are going to compare it with the initial configuration that was entered initially.