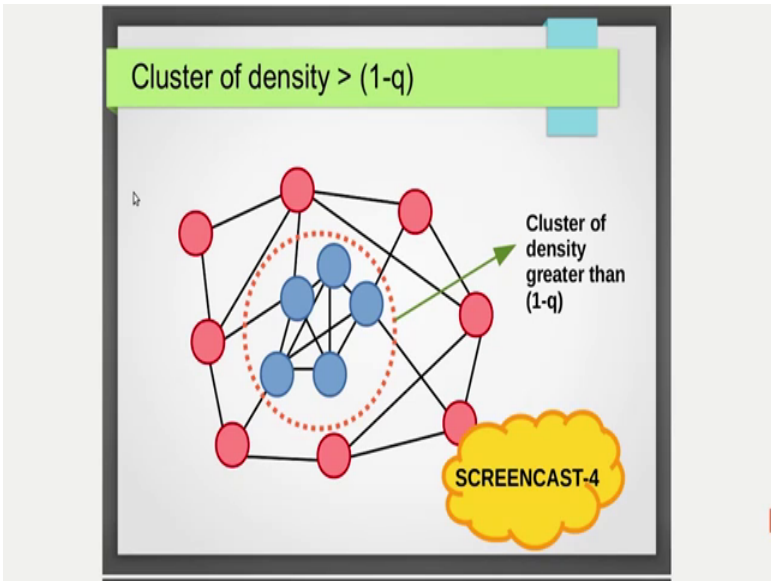
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

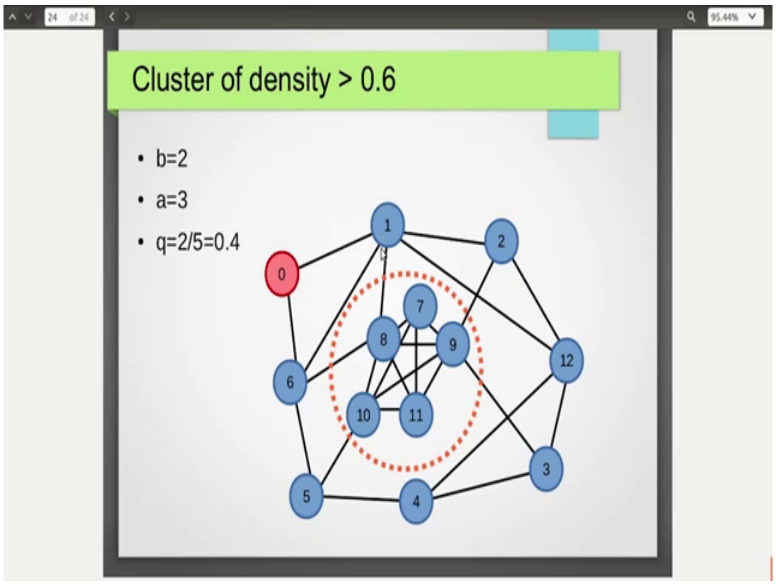
Cascading Behavior in Networks
Lecture - 100
Coding the Fourth Big Idea – Cascades and Clusters

(Refer Slide Time: 00:05)


Cluster of density > (1-q)
Cluster of density greater than (1-q)
SCREENCAST-4

Let us now move on to the last programming screen cast where we are going to validate this theorem experimentally that if there is a cluster of density greater than 1 minus q in your network, then you can never achieve a complete cascade. What we are going to do for this? We need certain parameters.

(Refer Slide Time: 00:25)



Cluster of density > 0.6

- b=2
- a=3
- q=2/5=0.4

So, these are the parameters which we take. Let the payoff associated with your behavior b which was your initial behavior on the network is 2 that the payoff associated with a new behavior which is coming up in your network b 3. So, the value of q become equals to 2 by 5.

So, you remember how did we calculate this q? So, you can actually go back and watch it the value of q is as we calculated was b divided by a plus b which is 2 by 5 here which is 0.4. What is it mean? It mean that in this network where initially everybody has adopted this behavior b and now some people have adopted is behavior a. Now every node will look at their neighbors and if 40 percent of their neighbors have adopted the behavior a that particular node will also adopt the behavior a is what it means. And then we are going to work with a particular graph. So, this is the graph we are going to work with.

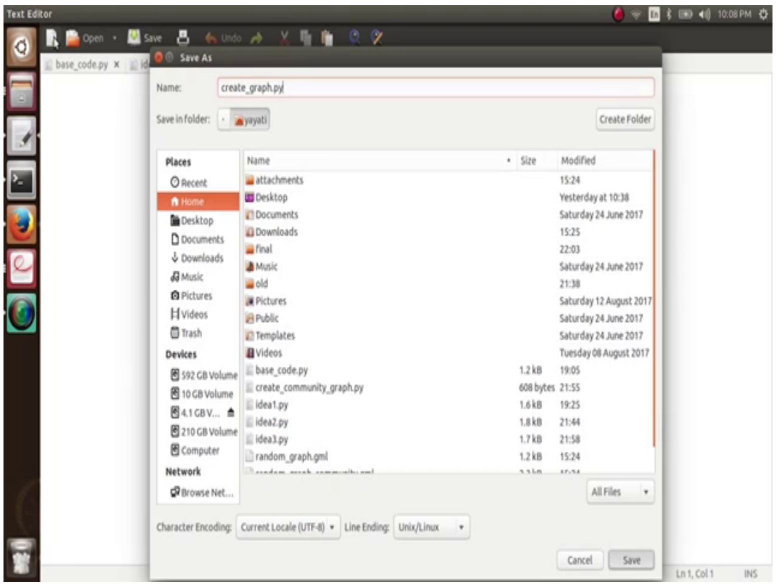
So, as you can see that in this network, here is a cluster and the density of this cluster is greater than 0.6. So, you can also validate it density greater than 0.6 means if you look at every node in the cluster whether it be node 7, 8, 9, 10 or 11; 60 percent of their neighbors should be inside this cluster only. So, for example, this node 8 here has how many neighbors? 1, 2, 3, 4, 5, 6; it has 6 neighbors. And out of the 6 neighbors, 4 belongs inside this community.

So, 4 by 6 is 2 by 3 and inside this it is own community which means greater than 60 percent. Node 7 has all its nodes inside the same cluster, 11 has all its nodes neighbors inside the same cluster and nodes 9 and 10 also have 60 percent of their neighbors inside this cluster.

So, we are going to take this network and after taking this network, we will you experimentally validate that no matter from which ever nodes you start your cascade outside this cluster. So, your cascade should start outside this cluster will start our cascade from some set of nodes which lie outside this cluster. And then we will see that you can never achieve a complete cascade. Your cascade will not be able to impact any of these nodes number from 7 to 11. So, we now code it and see.

So, for coding we are going to make this graph manually. We want to work with this graph because here we clearly know that we have this cluster. This is the set this set as a density greater than 0.6.

(Refer Slide Time: 03:06)



Name: create_graph.py

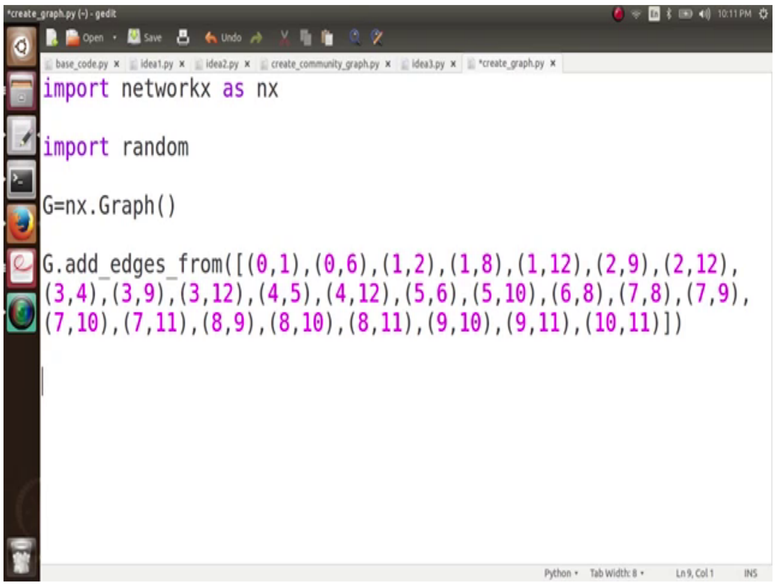
Save in folder: yayati

Name	Size	Modified
attachments		15:24
Desktop		Yesterday at 10:38
Documents		Saturday 24 June 2017
Downloads		15:25
final		22:03
Music		Saturday 24 June 2017
old		21:38
Pictures		Saturday 12 August 2017
Public		Saturday 24 June 2017
Templates		Saturday 24 June 2017
Videos		Tuesday 08 August 2017
base_code.py	1.2 kB	19:05
create_community_graph.py	608 bytes	21:55
idea1.py	1.6 kB	19:25
idea2.py	1.8 kB	21:44
idea3.py	1.7 kB	21:58
random_graph.gml	1.2 kB	15:24

Character Encoding: Current Locale (UTF-8) Line Ending: Unix/Linux

So, let us make a graph first. So, I will create a file here let say create underscore graph dot py.

(Refer Slide Time: 03:19)

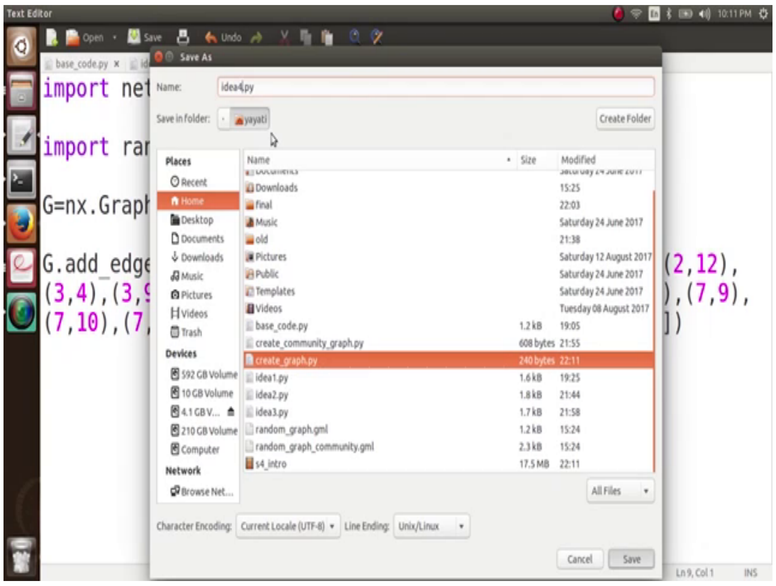


```
import networkx as nx

import random

G=nx.Graph()

G.add_edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])
```

And you can you can whatever edges I am going to type here, you can actually go back and match it with the figure which we have seen with the graph which we have seen previously. So, import networkx as nx and then import random and then G equals to nx dot graph and then I am going to put here edges and these edges are going to be absolutely the edges which where there in the figure.

So, these are the edges which we have added in this graph you can go back and match it with the figure which we have been previously. So, here we are getting this graph, we have this graph. Here only in this file only we are actually going to implement a third idea.

(Refer Slide Time: 04:02)

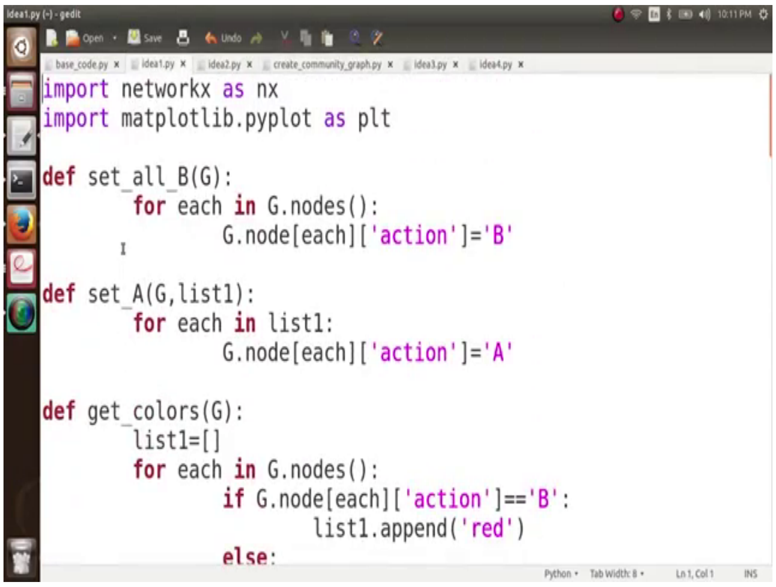


```
import net
import ran
G=nx.Graph
G.add_edge                                   (2,12),
(3,4),(3,                                    ),(7,9),
(7,10),(7,                                   ])
```

```
Save As
Name: idea4.py
Save in folder: yayati

Name                          Size        Modified
Downloads                                 15:25
final                                     22:03
Music                                     Saturday 24 June 2017
old                                       21:38
Pictures                                  Saturday 12 August 2017
Public                                    Saturday 24 June 2017
Templates                                 Saturday 24 June 2017
Videos                                    Tuesday 08 August 2017
base_code.py                  1.2 kB      19:05
create_community_graph.py     608 bytes   21:55
create_graph.py               240 bytes   22:11
idea1.py                      1.6 kB      19:25
idea2.py                      1.8 kB      21:44
idea3.py                      1.7 kB      21:58
random_graph.gml              1.2 kB      15:24
random_graph_community.gml    2.3 kB      15:24
s4_intro                      17.5 MB     22:11

Character Encoding: Current Locale (UTF-8)   Line Ending: Unix/Linux
```

So, let us actually save it as from create graph instead we should save it as let us say idea 4 dot py. We are going to a forth idea here. So, we have the graph which we wanted.

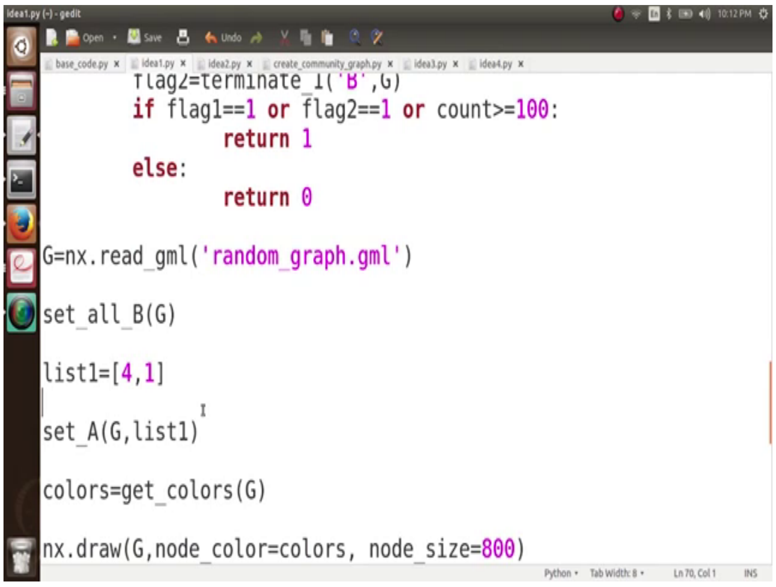
(Refer Slide Time: 04:22)



```
import networkx as nx
import matplotlib.pyplot as plt

def set_all_B(G):
        for each in G.nodes():
                G.node[each]['action']='B'

def set_A(G,list1):
        for each in list1:
                G.node[each]['action']='A'

def get_colors(G):
        list1=[]
        for each in G.nodes():
                if G.node[each]['action']=='B':
                        list1.append('red')
                else:
```

And next what we want to do is our all the functions are actually going to be the same which were here. So, I might consider copy pasting these functions here. So, just give me 1 minute. These are all the basic functions which we have implemented previously ok.

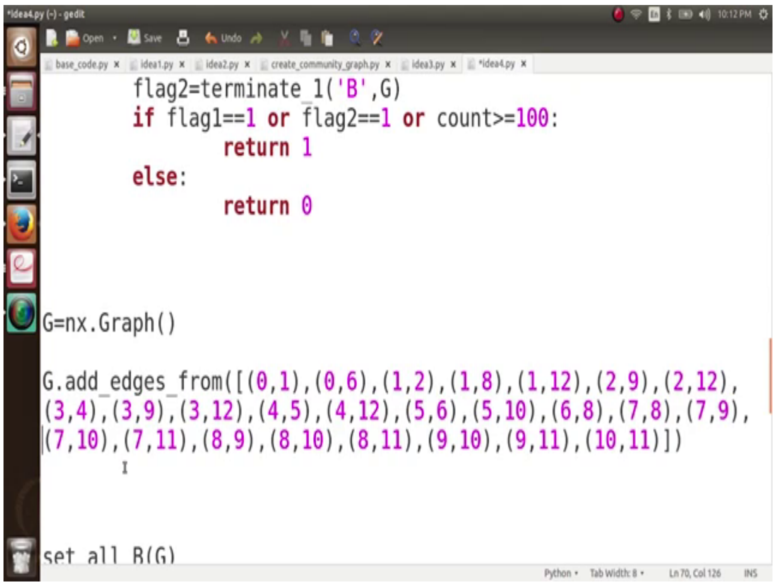
(Refer Slide Time: 04:55)



```
        flag2=terminate_1('B',G)
        if flag1==1 or flag2==1 or count>=100:
                return 1
        else:
                return 0

G=nx.read_gml('random_graph.gml')

set_all_B(G)

list1=[4,1]

set_A(G,list1)

colors=get_colors(G)

nx.draw(G,node_color=colors, node_size=800)
```

And what else do we need from this first code is let us take this code also. It is going to help us and we have this code here.

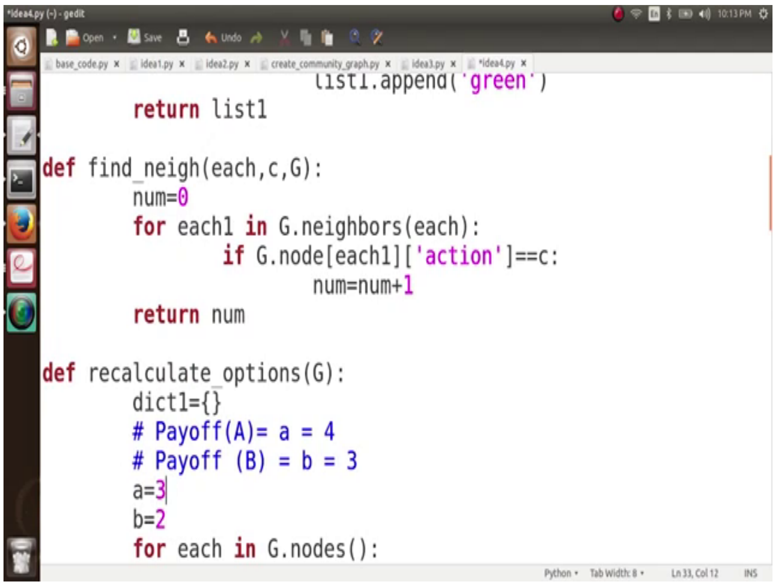
(Refer Slide Time: 05:11)



```
        flag2=terminate_1('B',G)
        if flag1==1 or flag2==1 or count>=100:
                return 1
        else:
                return 0


G=nx.Graph()

G.add_edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])


set_all_B(G)
```

So, here we have this graph and we have all the necessary functions here. What we have to do now is as we decided we know that the payoff associated with this action b.

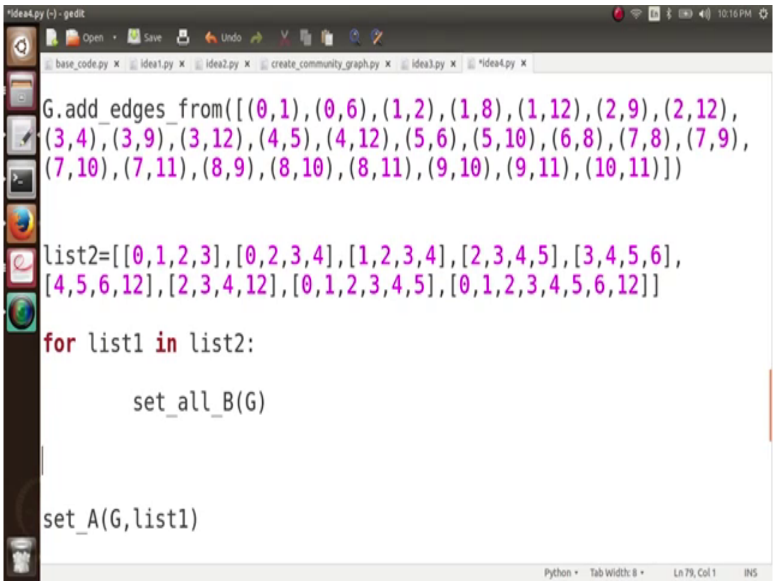
(Refer Slide Time: 05:32)



```
                list1.append('green')
        return list1

def find_neigh(each,c,G):
        num=0
        for each1 in G.neighbors(each):
                if G.node[each1]['action']==c:
                        num=num+1
        return num

def recalculate_options(G):
        dict1={}
        # Payoff(A)= a = 4
        # Payoff (B) = b = 3
        a=3
        b=2
        for each in G.nodes():
```

Here is going to be 2 and a here is going to be 3.

(Refer Slide Time: 05:41)



```
G.add_edges_from([(0,1),(0,6),(1,2),(1,8),(1,12),(2,9),(2,12),
(3,4),(3,9),(3,12),(4,5),(4,12),(5,6),(5,10),(6,8),(7,8),(7,9),
(7,10),(7,11),(8,9),(8,10),(8,11),(9,10),(9,11),(10,11)])


list2=[[0,1,2,3],[0,2,3,4],[1,2,3,4],[2,3,4,5],[3,4,5,6],
[4,5,6,12],[2,3,4,12],[0,1,2,3,4,5],[0,1,2,3,4,5,6,12]]

for list1 in list2:

        set_all_B(G)



set_A(G,list1)
```

And then so, there were how many nodes in the network? If you remember there was 0 to 12 nodes. So, total 13 nodes in the network and out of this 13 networks denotes inside our cluster were nodes from 7 to 11. So, we can start our cascade from any node from any set of nodes except the node 7 to 11.

What I am going to do is there are actually a lot of possible sets from this list from 0 to 6 and then node number 12. So, how many nodes? 0 to 6; 7 plus 8; so, there are 8 nodes.

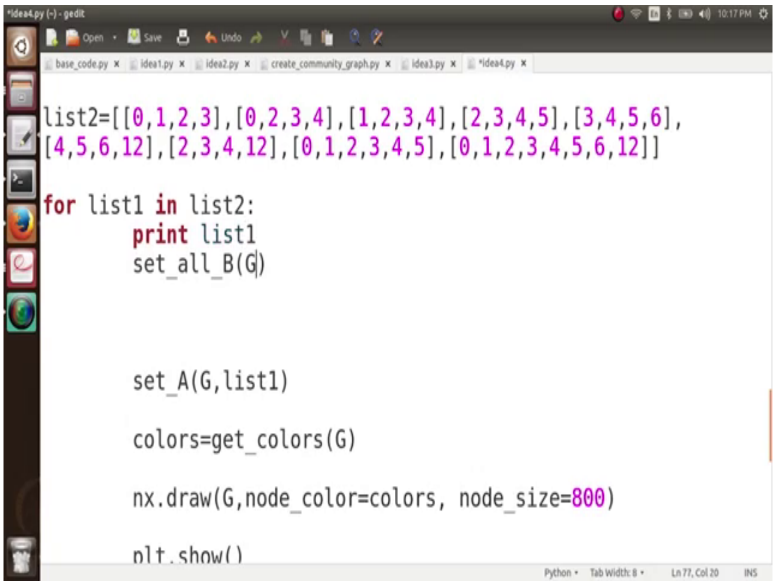
So, there are how many possible sets on in 8 nodes is 2 to the power 8. So, I am not actually going to take all the 2 raise to the 2 to the power 8 sets and show you the result. I will take some particular sets here and show you the result. So, I create a list here let us say list 2 and what is this list 2? This list 2 is the list of lists. So, I will having some lists here.

So, this first list here is the first my set of initial adopter for the first experiment. This list is the set of initial adopters for the second experiment and I show you that for all these sets. So, there are different different initial adopters. So, for all of these sets, your cascade is not complete. It is unable to hit any of the node inside the cluster.

So, let us make here list let say that the first list is 0, 1, 2, 3. So, I can take any number of elements here outs any number of elements here except the elements from 7 to 11 which are inside my cluster. And let us take 0, 2, 3, 4 and let us also take sorry 1, 2, 3, and 4 and 2, 3, 4, 5. Let us take many many sets 3, 4, 5, 6 4, 5, 6, 12. Let say 2, 3, 4, 12. Let us take many many nodes, let us take 6 node 0, 1, 2, 3, 4, 5. It seems that if you start your cascade from so many nodes from 6 node the cascade should be complete. And rather let us taken extreme case, let us take all the nodes except the nodes in the cluster 0, 1, 2, 3, 4, 5, 6 and 12.

So, except all the nodes in the particular cluster, we are talking about I am taking all the nodes here and we see that even in this case your cascade is unable to become complete. So, now, what we are going to do for list 1 in list 2, we are going to repeat this experiment ok.

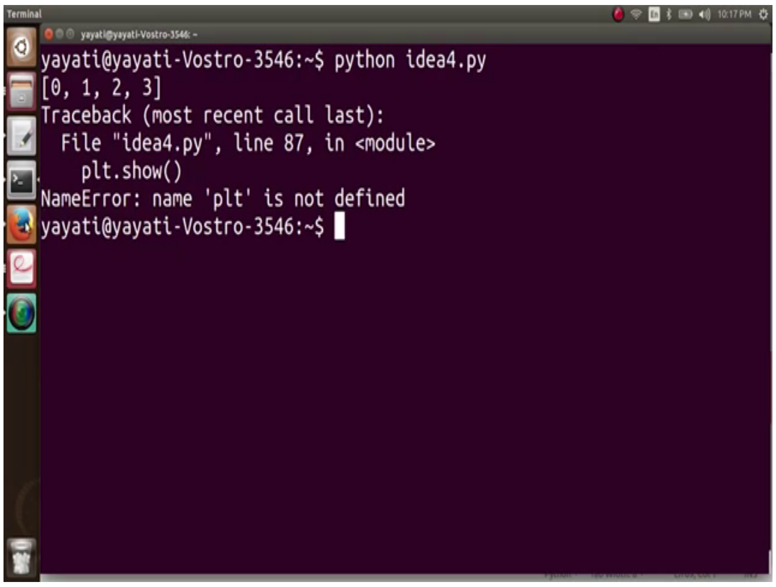
(Refer Slide Time: 09:30)



```
list2=[[0,1,2,3],[0,2,3,4],[1,2,3,4],[2,3,4,5],[3,4,5,6],
[4,5,6,12],[2,3,4,12],[0,1,2,3,4,5],[0,1,2,3,4,5,6,12]]

for list1 in list2:
        print list1
        set_all_B(G)



        set_A(G,list1)

        colors=get_colors(G)

        nx.draw(G,node_color=colors, node_size=800)

        plt.show()
```

So, for every possible set of initial adopters here, what will be seeing is that list and then will be seeing the final graph and then will be seen whether the cascade is complete or not. So, there is nothing much we just need to execute it and see what is happening here.

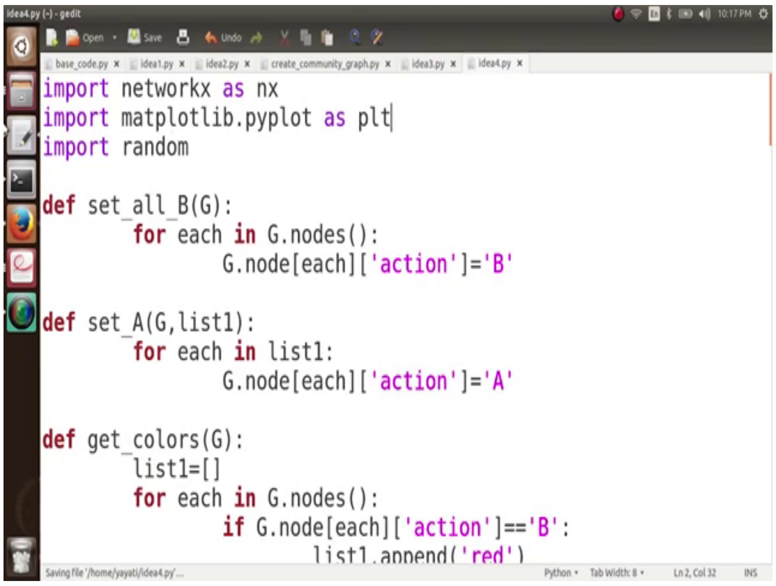
(Refer Slide Time: 09:48)



```
yayati@yayati-Vostro-3546:~$ python idea4.py
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <module>
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$
```

python idea4 dot py.

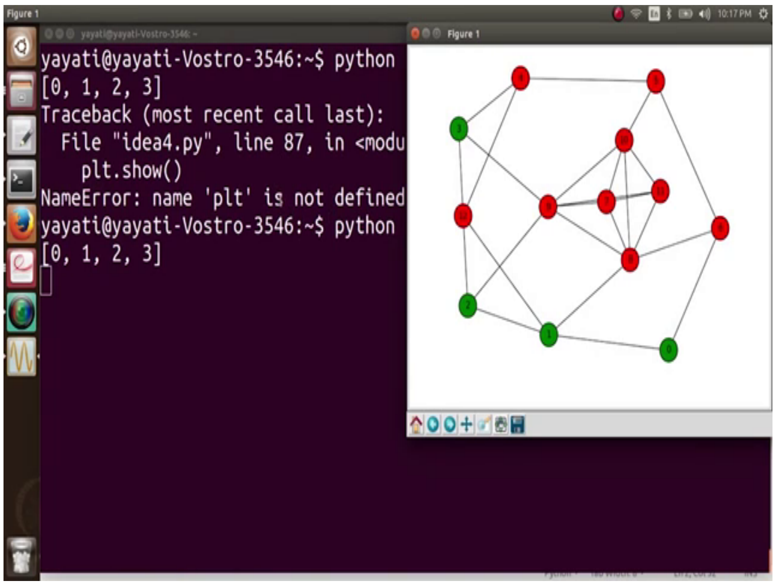
(Refer Slide Time: 10:02)



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

def set_all_B(G):
        for each in G.nodes():
                G.node[each]['action']='B'

def set_A(G,list1):
        for each in list1:
                G.node[each]['action']='A'

def get_colors(G):
        list1=[]
        for each in G.nodes():
                if G.node[each]['action']=='B':
                        list1.append('red')
```

So we have to import matplotlib dot pyplot as plt.

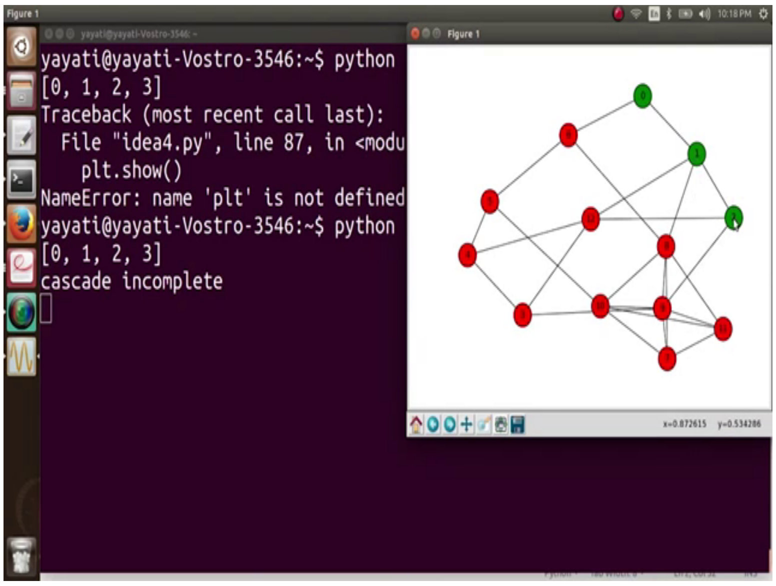
(Refer Slide Time: 10:18)



```
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <modu
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
```

So, here is a case when our initial adopter are 0, 1, 2, 3. So, this is the cluster we were talking about. So, this is a cluster having a density greater than 60 percent and you will see that in none of the cases this cascade will be able to touch this cluster.

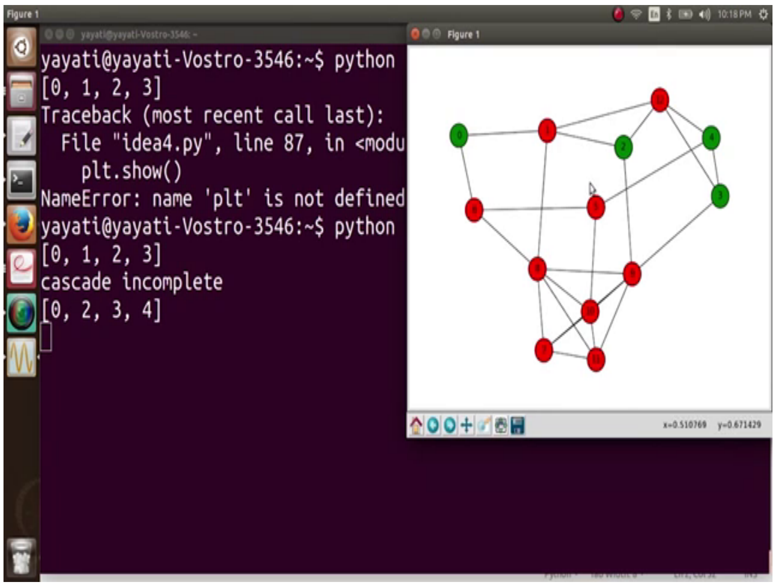
(Refer Slide Time: 10:37)



```
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <modu
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
cascade incomplete
```

So, in this case when our cascade start from 0, 1, 2, 3, this is the final cascade 0, 1, 2 and it remains incomplete, rather outside cluster also there are a number of nodes which remain unaffected.

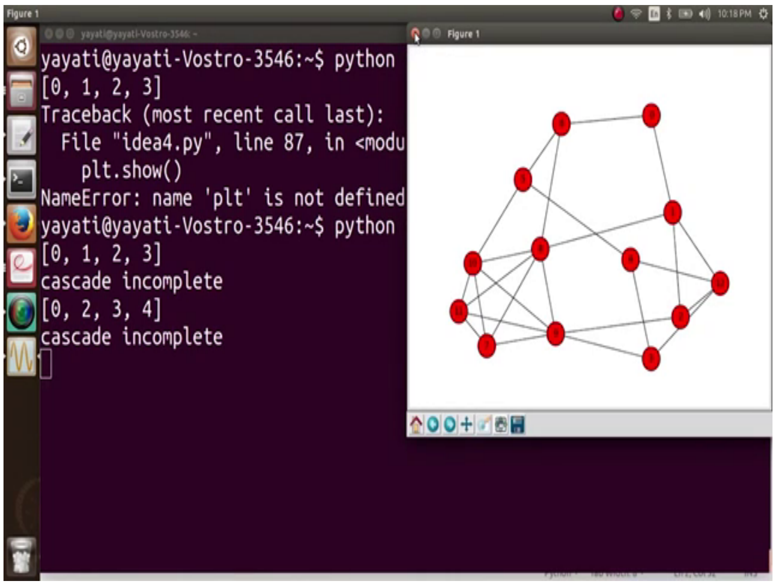
(Refer Slide Time: 10:47)



```
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <modu
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
cascade incomplete
[0, 2, 3, 4]
```

And then we start from 0, 2, 3, 4.

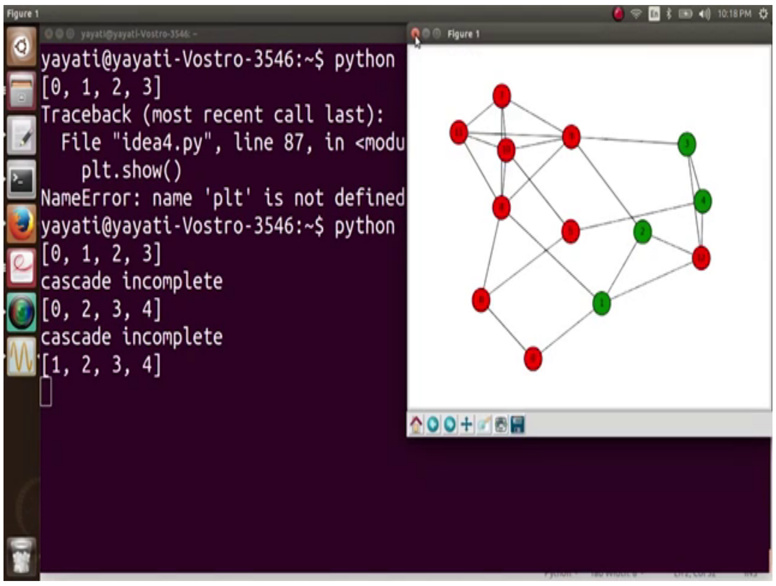
(Refer Slide Time: 10:50)



```
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <modu
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
cascade incomplete
[0, 2, 3, 4]
cascade incomplete
```

And then, you see that cascade is incomplete rather the cascade dies away with time.

(Refer Slide Time: 10:57)



```
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
Traceback (most recent call last):
  File "idea4.py", line 87, in <modu
    plt.show()
NameError: name 'plt' is not defined
yayati@yayati-Vostro-3546:~$ python
[0, 1, 2, 3]
cascade incomplete
[0, 2, 3, 4]
cascade incomplete
[1, 2, 3, 4]
```

And let say we start with 1, 2, 3, 4. And then again the cascade dies away with time 2, 3, 4, 5 dies away 3, 4, 5, 6 and sub having some node infected, but still it is unable to touch any of the nodes in our cluster from 7 to 11. And then let us start from 4, 5, 6, 12 when we start from 4, 5, 6, 12. So, this is the final graph and we see that the cascade remains incomplete then, we start from 2, 3, 4 and 12 and then many nodes are inserted, but again and cluster remains unaffected and then we start from almost 6 node seems like everybody should be in affected at the end, but you can still see that the cluster remains unaffected. And then this was the extreme case we were talking about.

So, we start. So, these many nodes almost 1, 2, 3, 4, 5, 6, 7, 8 nodes in the graph, I have adopted the behavior a. Even then our cascade remains incomplete and our cluster remains intact the cascade is unable to enter this cluster. So, we have validated that if there is a cluster of density greater than 1 minus q in the network of the cascade can never become complete.