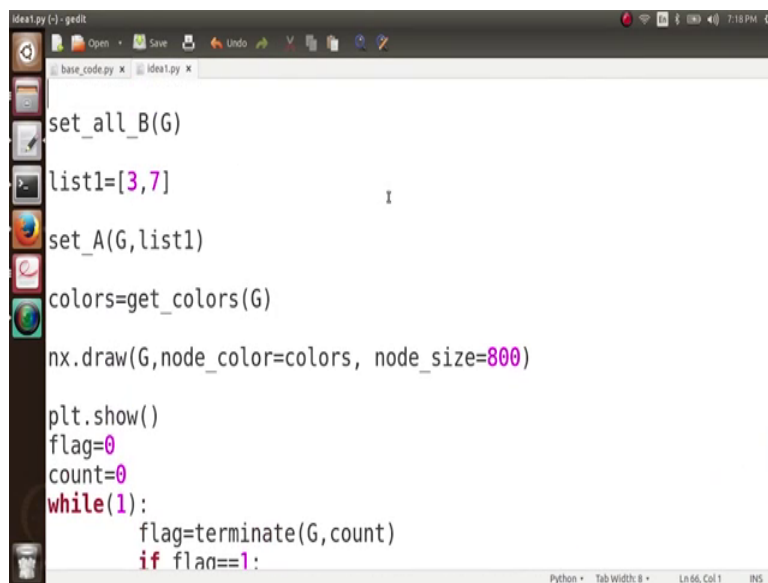**Social Networks**
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Cascading Behavior in Networks
Lecture - 97
Coding the First Big Idea - Increasing the Payoff
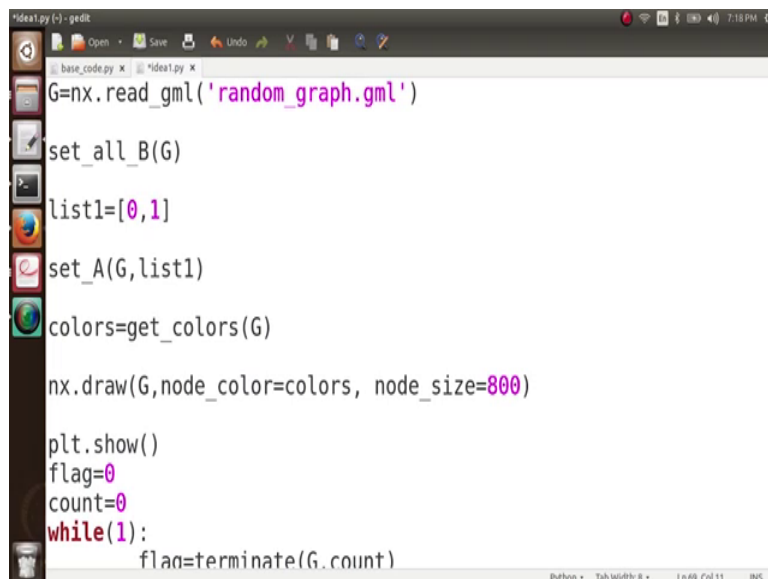
(Refer Slide Time: 00:05)

So, now let us experiment with this code a little bit. Everything is ready now.

(Refer Slide Time: 00:11)

So, we have a system where we can define, who are the initial adopters of the action a and then, we can run an run the code and see how this action a is cascading in the network whether it dies away, or whether it persist in the network or what happens. So, for the time being what I am going to do is, I am going to set the initial adopters to be 0 and 1 in the network and I am actually, also going to change our payoffs associated with a and b.

(Refer Slide Time: 00:34)

So, let me keep the payoffs associated with a to be equal to 6 and with b to be equal to 5.

(Refer Slide Time: 00:46)

And let us execute this code and see what is going to happen.

(Refer Slide Time: 00:47)

So, initially 0 and 1 in this network I have adopted the idea and then, you see it dies away. So, seems like the payoff 6 is not enough for this action a to cascade on this network right. So, what I am going to do is, I am going to increase the payoff and I am going to keep this payoff of to be 7. Let us see what happens now.
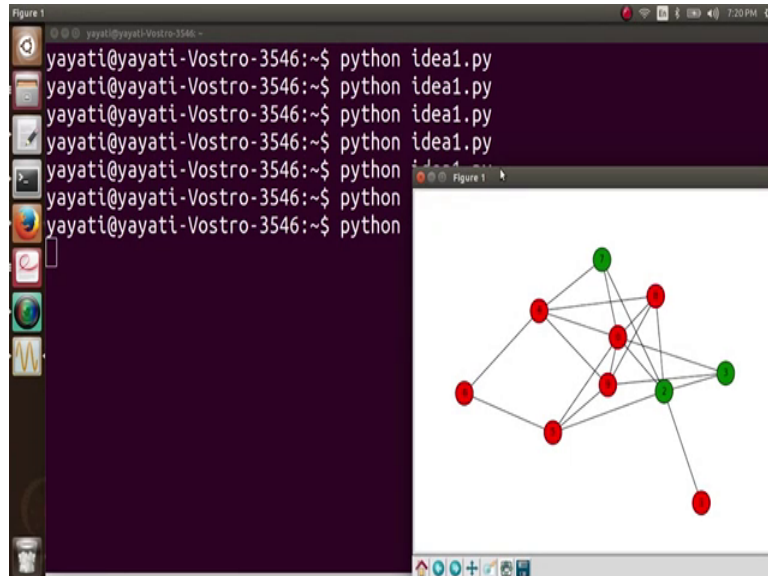
(Refer Slide Time: 01:10)
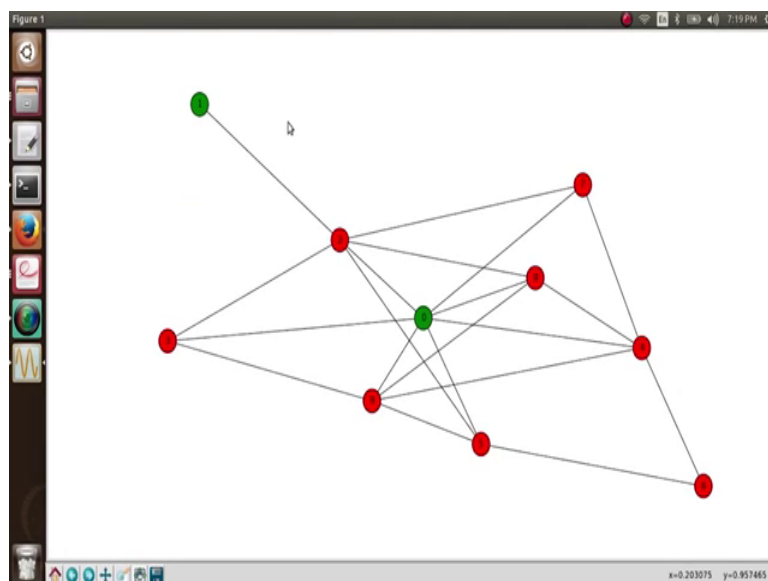
So, when I keep this payoff of to be 7 initially, 1 and 0 and then again, it dies away. So, the payoff 7 is also not working. Let us make it 8. I am make it 8 and then we have 0 and
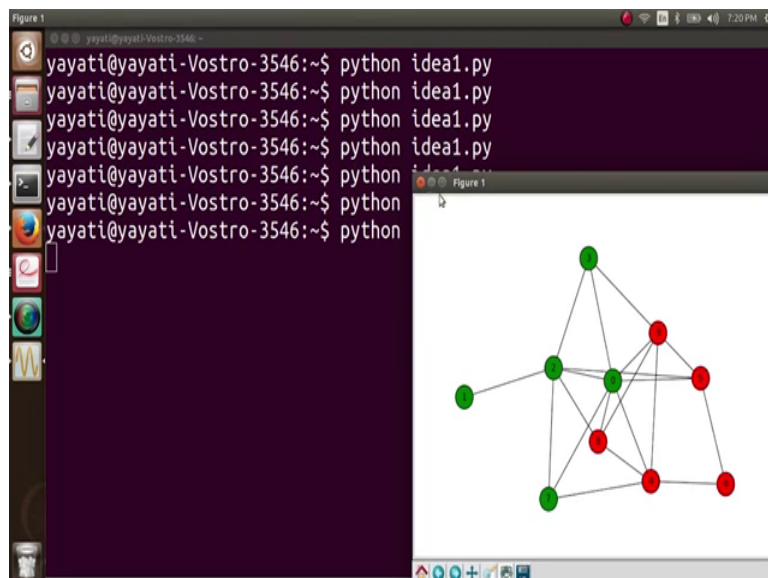
1, in 8 also it dies away ok. Let us make it 9 dies away. And let us make it 10 and then you see when I make it 10.
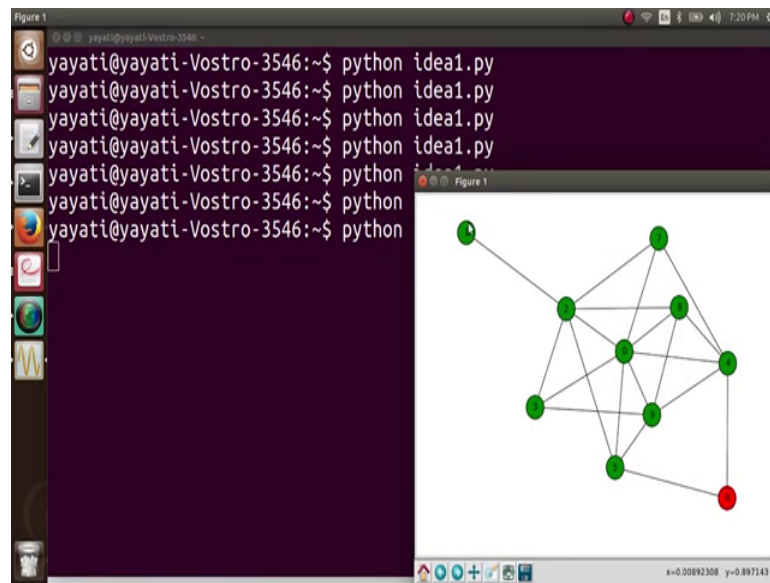
(Refer Slide Time: 01:43)

Now, although 0 and 1 they have adopted the behaviour b now, but these three nodes 2, 3 and 7, they have adopted the behaviour a. So, the behaviour is persisting.
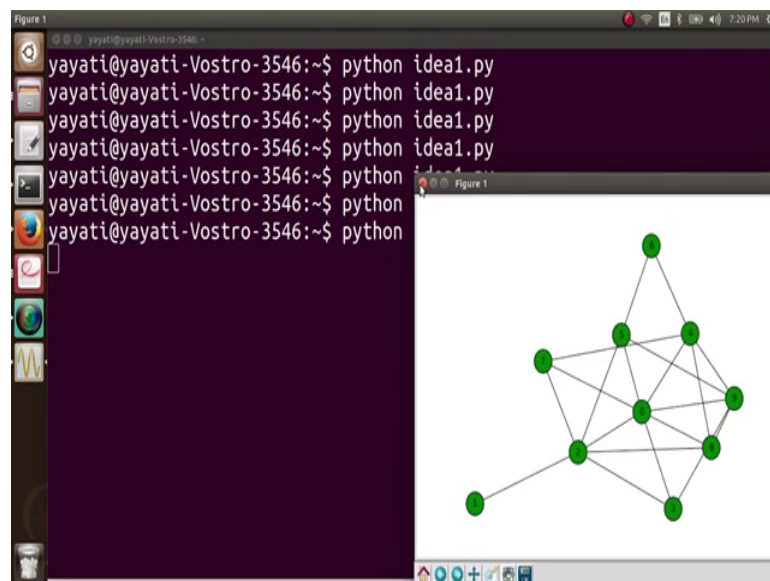
(Refer Slide Time: 02:01)

Let us see in the next iteration. In the next iteration now, 0 and 1 also get back the behaviour a. So, now, you can see this behaviour a cascading on this network.
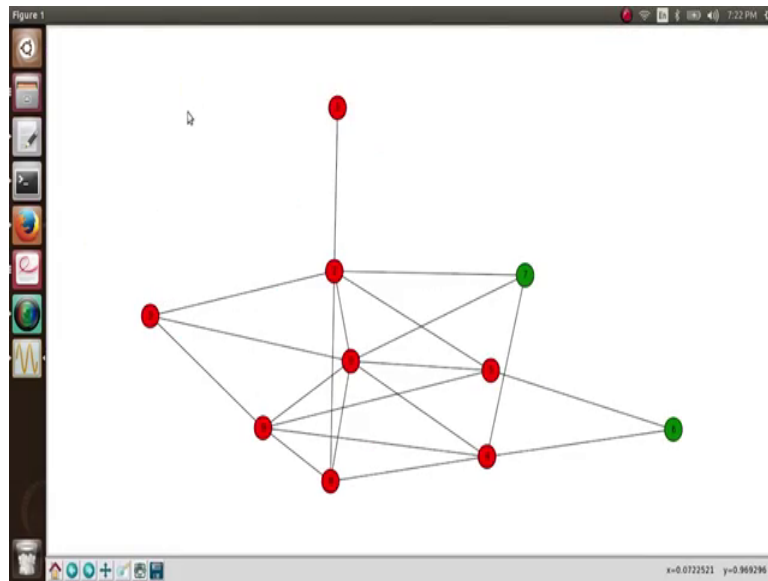
(Refer Slide Time: 02:10)

And then you see some more nodes adopt the behaviour a.

(Refer Slide Time: 02:14)

And then finally, all the nodes adopt this behaviour a and the process ends. So have you seen just now what happened? The, we have started with a payoff 6. So, when the payoff was 6 nothing happened 7, 8, 9, the behaviour ultimately died away, but when you made its payoff to be equal to 10, everybody in this network adopted the behaviour a. Let us try it something else also. So, let us make it 6 and 5.

(Refer Slide Time: 02:42)

And let us change my initial adopters to be let us say 4 and 1 and let us see what happens now. 4 and 1 and then, 6 and then, dies away ok. Let us increase the payoff to let us say 8, 4 and 1, 6, dies away and say 9, 4 and 1 and 6 and then, dies away and then 10 ok.

(Refer Slide Time: 03:06)

Do you see here what is happening now in this case?

(Refer Slide Time: 03:36)

So, this was what I wanted to show you. In some situations what can happen, you see 7, 6 and then, 7 and 6 in fact 4. And then, 4 in fact 7 and 6 and then, 4 and then, this and this, and this, code goes into an infinite loop. And it is going to run a 100 times. It is going to run a 100 times ok.

(Refer Slide Time: 04:09)
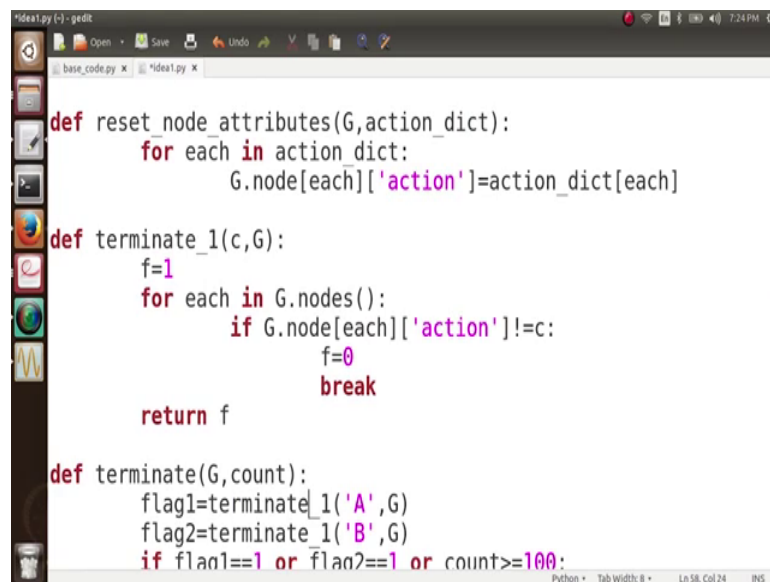
So, what I want to do is instead, of seeing this code to run for 100 times, I actually want to know just a simple thing. I want to know whether my cascade is complete or not.

What do I mean by my cascade is complete is, everything at the end has adopted this behaviour which is behaviour a shown in green colour or not.

So, that is my only aim. So, I do not need, I do not want to see these intermediary graphs. So, I just want to see my graph once at the starting, once at the end and in addition I want to know whether my cascade is complete or not. So, we can quickly see whether my cascade is complete or not. So, we call a function c here sorry, variable c here which holds the value is my cascade complete or not; is complete cascade. So, whether this cascade which has occurred on my graph it is complete or not? And how do we know it is complete?

(Refer Slide Time: 05:21)
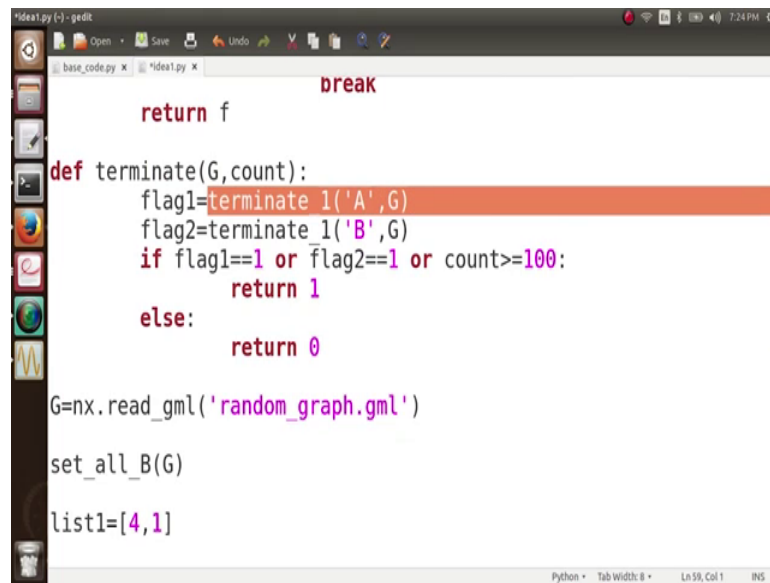
So you see, we have used this function here, terminate 1, A, G. So, this function it used to return 1. When everything in this graph has adopted the behaviour a, so, instead of using a new function, we can use the same function.
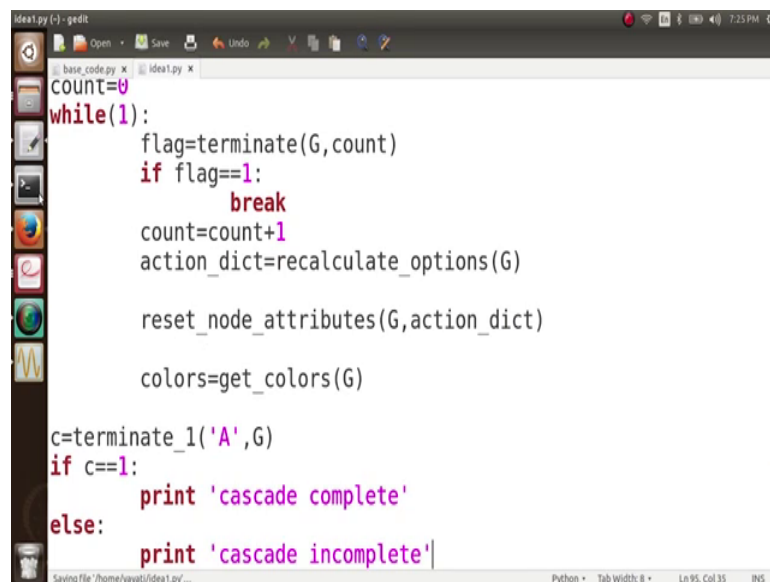
(Refer Slide Time: 05:35)

So, we can use this function terminate 1 A, G, which tells me whether all the nodes in my graph has adopted the behaviour a or not right.
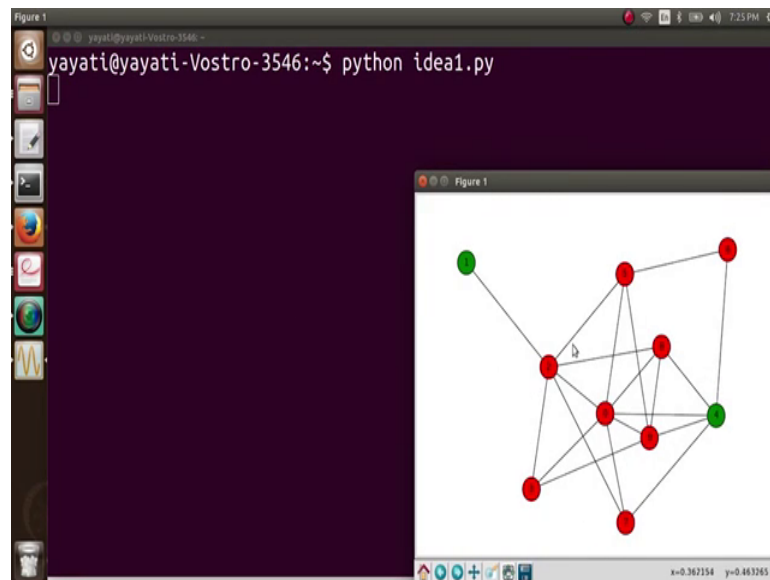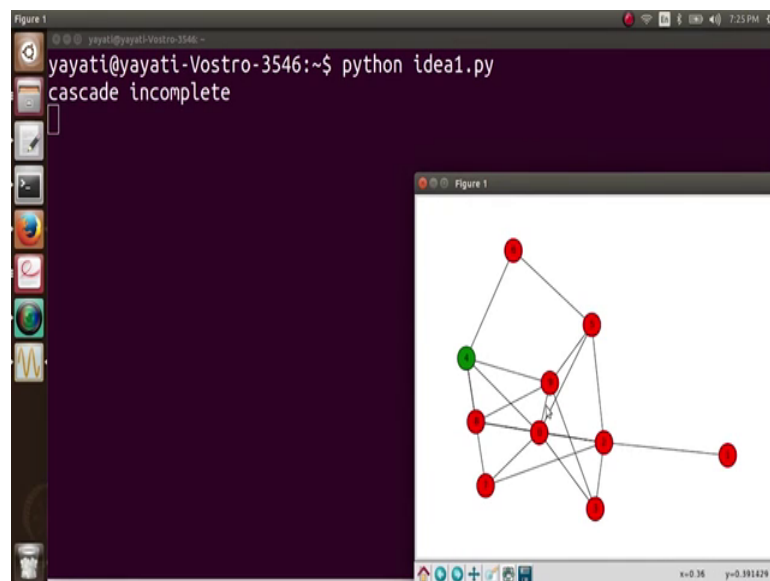
(Refer Slide Time: 05:40)

What I will do is, c equals to terminate 1 A, G and what I want to see is, I want to print. So, if your c equals to equals to 1 then, you print cascade is complete else you print cascade is incomplete. Let us run it and see now; idea1 dot py.

(Refer Slide Time: 06:24)

And then, this is the initial graph where 4 and 1 are infecting.

(Refer Slide Time: 06:29)

And this keeps running. So, it shows a result at the end after 100 iterations and it shows you that the cascade is incomplete. So, this is the only result we are interested in and we need not see all the intermediate cascades. We have just introduce this here because, we will be using it in the coming up programming screen casts.
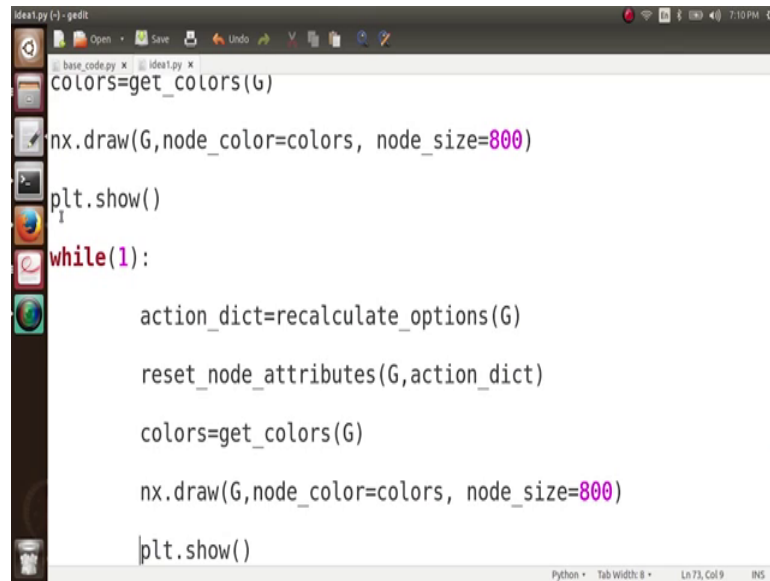
(Refer Slide Time: 06:50)

Let us get started with the first idea where we have to increase the payoff and see what happen. Well basically, take this code and I will just paste it in another file.

(Refer Slide Time: 06:57)
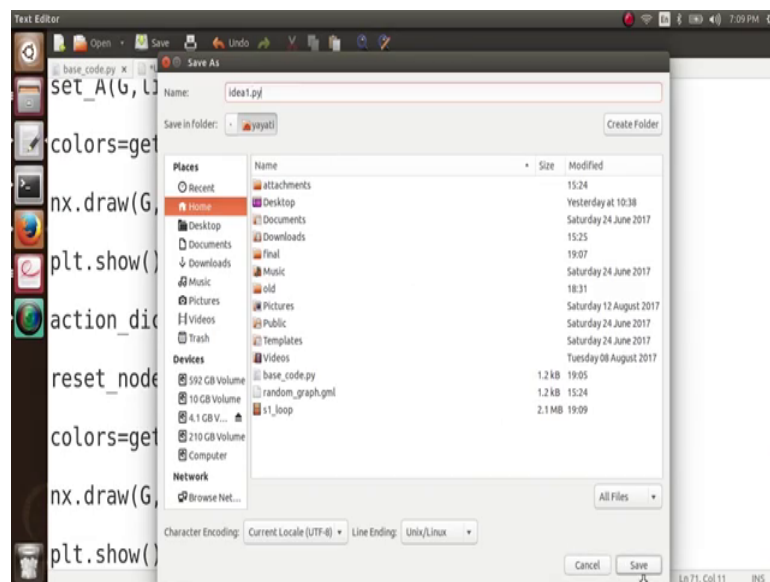
 And let me name this atom, let me name it as idea1 dot or idea1 dot py ok.
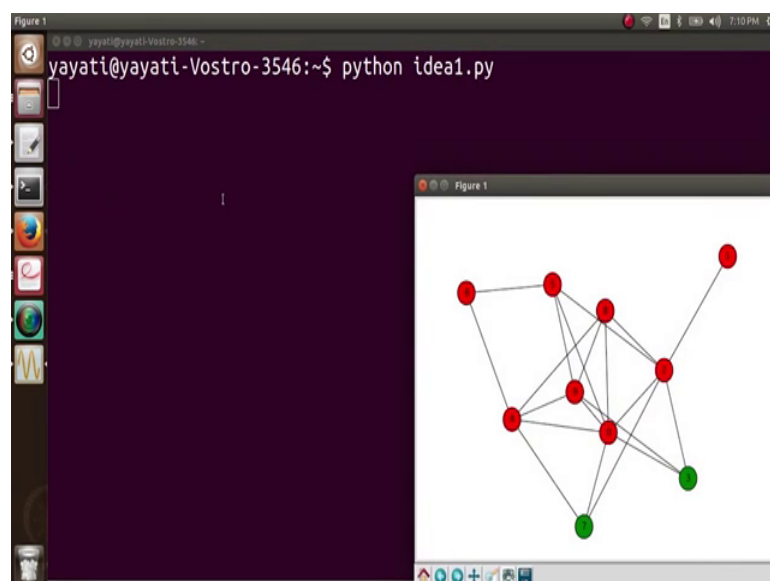
Now, one thing here, you see here currently we are running this process manually. So, this our initial graph right, nx dot draw where we set 2 nodes to have, where we set this 2 nodes 3 and 7 to have this initial, to have this behaviour a and then, we draw this graph and then we recalculate the options and then we again draw the graph. So, let us automate this process. So, what we will do if we can actually put this thing inside a loop right.

This complete thing can be put inside a loop. So, what will happen if I put it inside a loop? So, while 1, while 1, I just keep repeating this thing. So, every time my, so, this process is now become iterative. So, we start with some initial configuration and then every node recalculates its options and then it keeps drawing the graph again and again. So, I will just quickly show you.

(Refer Slide Time: 08:24)
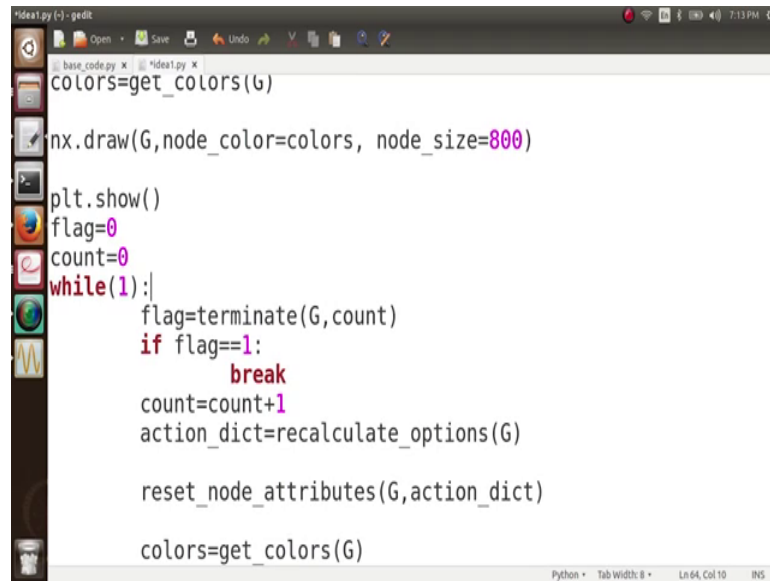
(Refer Slide Time: 08:26)

So, if I implement this here, python idea1 dot py, so you see what? So, this is our initial graph and then you see it will keep drawing it again, and again, and again right. Now, we can see how the cascade is growing in the network. So, currently the cascade dies at the first step itself. So, we are unable to see anything ok.

```python
colors=get_colors(G)

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

plt.show()
flag=0
count=0
while(1):
        flag=terminate(G,count)
        if flag==1:
                break
        count=count+1
        action_dict=recalculate_options(G)

        reset_node_attributes(G,action_dict)

        colors=get_colors(G)
```
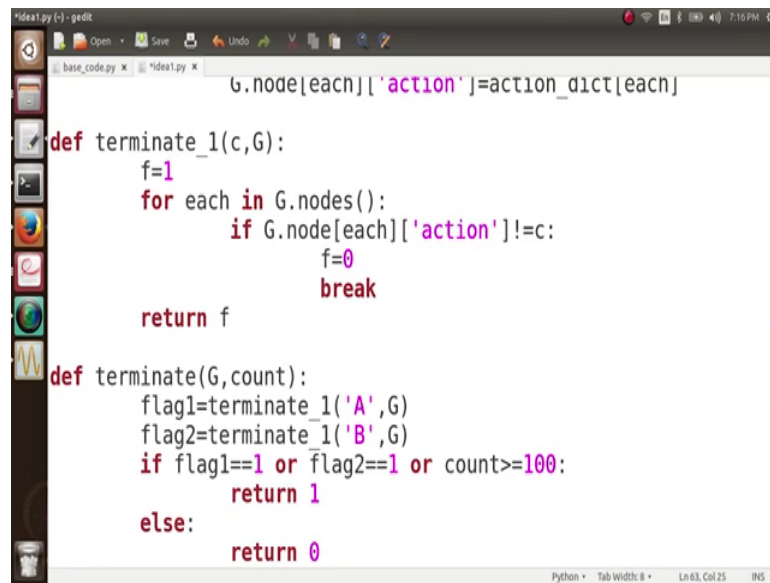
So what will do is, we want to put a terminating condition here. When should my code terminate? So, for that when should my code terminate? When should this process end? So, the process should end when a complete cascade occurs whether for the behaviour a or b. So, if everything has adopted the behaviour b, behaviour a dies away and if everything, every node adopts behaviour a, behaviour b dies away.

So, in both of these conditions the process should come to an end; however, there is one more condition where it does not converge to either a, or b. The node keeps flipping. So, it might be, so, it goes into an infinite loop. So, what do we do for that condition is we keep a maximum iteration limit for 100. So, we do not want to see this process beyond 100 iterations. 100 is actually a big number. So, we put a terminating condition here. First of all we take a variable flag here, flag equals to 0.

In while 1, we recalculate this flag. It will call a function terminate G which will decide whether we terminate or not. And if our flag equals to equals to 1 then we break. So, this function terminate it runs some code and if the process has to be terminated, it should set flag to be equal to 1. Let us also take a variable count here, count equals to 0 and we pass count also here. And count should increase every time ok.

```python
                G.node[each]['action']=action_dict[each]

def terminate_1(c,G):
        f=1
        for each in G.nodes():
                if G.node[each]['action']!=c:
                        f=0
                        break
        return f

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

So, let us now define this function terminate G and count, define terminate G count. What is this function going to do is, if all the nodes have adopted the behaviour a or adopted the behaviour b we terminate it. So, we take further two values flag 1, flag 1 is terminate underscore 1, a new function and it checks whether, so, flag 1 is going to check whether all the nodes in this network have adopted the behaviour a. All the nodes in this behaviour in this network have adopted the behaviour a and it is actually very easy to check.

So, we define terminate underscore 1 here and here we have A sorry, here we have a character c and the graph G. So, what we are going to do is, we take again indicator variable flag. What we do is for each in G dot nodes, for everyone should have one behaviour c. So, as soon as we find a node, if G dot node each; as soon as we find a node whose action is not equal to c, but something else. It means that all the nodes in this network does not have this behaviour. We set f equals to 0 and we break. And we return f here.

So, you see f is going to return a 1 only if all the nodes in this network have this behaviour character c. So, flag 1 tells me whether all the nodes in this network have this behaviour a and similarly, flag 2 tells me whether all the nodes in this network have this behaviour b. These are 2 conditions and then, we have a count also. So, count should be less than 100. So, what do we do is if your, so, we have 3 conditions.

If our flag 1 is 1 or our flag 2 is 1 or our count value is greater than 100. Now either greater than or equals to 100, what do we do is we return 1 that, you should terminate else we return 0 ok.

(Refer Slide Time: 13:50)

And now also ok so, first of all let us execute and see what is happening.

(Refer Slide Time: 14:12)
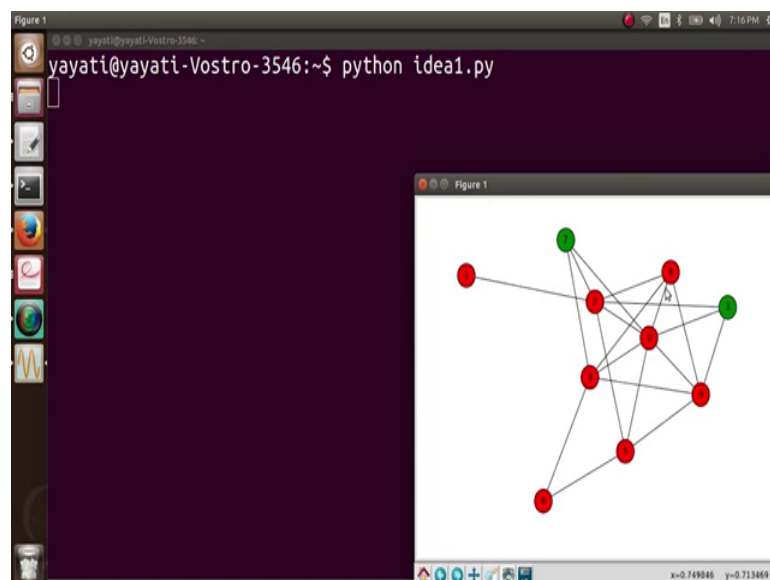
So, we execute it. So, we have this network where initially 7 and 3 have adopted the idea and then, this idea goes away. And you see what happened: 7 and 3 are infected and then this idea goes away. Everybody here has adopted b. So, the process stops.