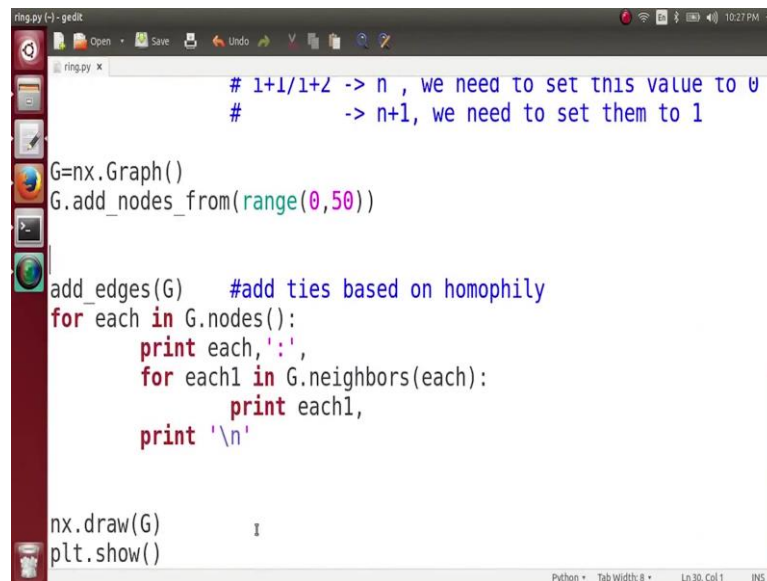


**Social Networks**  
**Prof. S. R. S. Iyengar**  
**Department of Computer Science**  
**Indian Institute of Technology, Ropar**

**How to Go Viral on Web**  
**Lecture - 153**  
**Adding Weak Ties**

(Refer Slide Time: 00:05)



```
ring.py (-) - gedit
# 1+1/1+2 -> n , we need to set this value to 0
#         -> n+1, we need to set them to 1

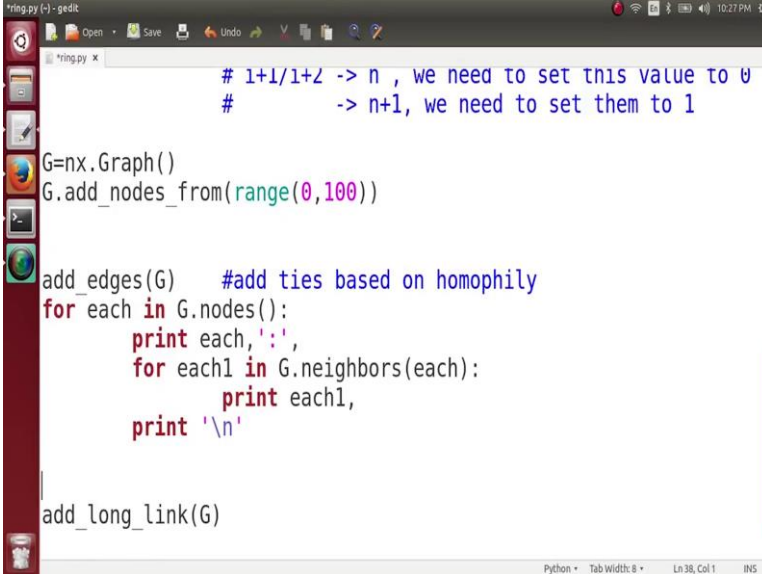
G=nx.Graph()
G.add_nodes_from(range(0,50))

add_edges(G) #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'

nx.draw(G)
plt.show()
```

So, now we have a network which is having 50 nodes, and which is having homophily based contexts where every node is connecting to 2 nodes towards its left and 2 nodes towards its right ok.

(Refer Slide Time: 00:23)



```
ring.py (-) - gedit
# 1+1/1+2 -> n , we need to set this value to 0
# -> n+1, we need to set them to 1

G=nx.Graph()
G.add_nodes_from(range(0,100))

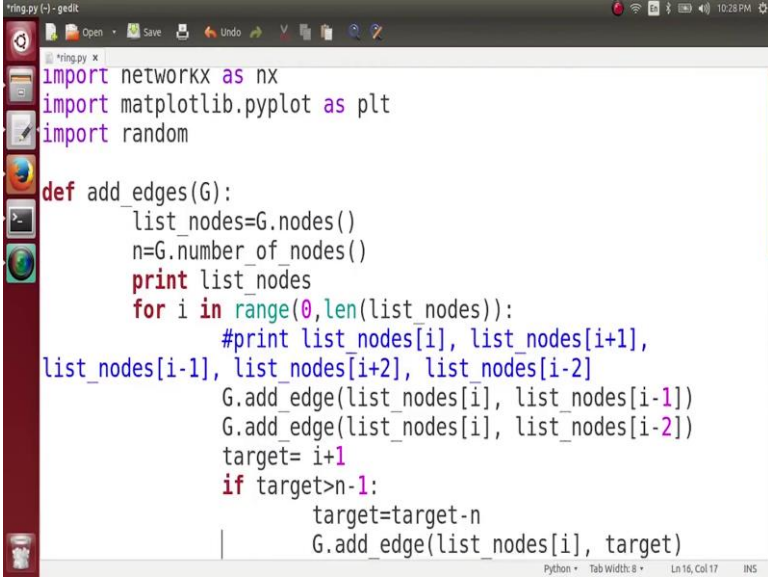
add_edges(G) #add ties based on homophily
for each in G.nodes():
    print each,':',
    for each1 in G.neighbors(each):
        print each1,
    print '\n'

add_long_link(G)
```

Let us take 100 node instead of 50. Next what we want to do is I am going to create a simple function which is going to create a weak tie. So, when we call this function ones it creates one weak tie, we are going only to add these weak ties. And we know that this is very easy, we have done it many times before from going to take a function here let us say add long link its basically add a long range link in G and it is going to be pretty easy, isn't it?

So, we what we have to do? We have to randomly pick 2 nodes in this method and put an edge between them.

(Refer Slide Time: 01:03)



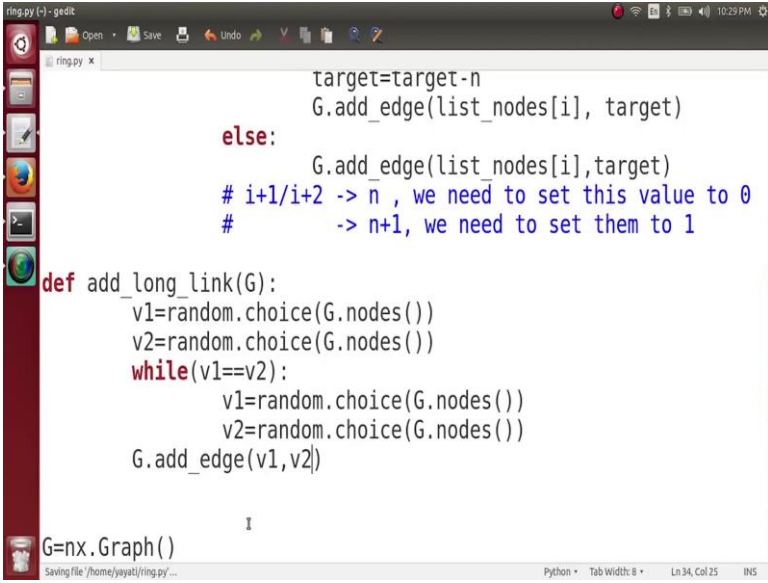
```
ring.py - gedit
ring.py
import networkx as nx
import matplotlib.pyplot as plt
import random

def add_edges(G):
    list_nodes=G.nodes()
    n=G.number_of_nodes()
    print list_nodes
    for i in range(0,len(list_nodes)):
        #print list_nodes[i], list_nodes[i+1],
        list_nodes[i-1], list_nodes[i+2], list_nodes[i-2]
        G.add_edge(list_nodes[i], list_nodes[i-1])
        G.add_edge(list_nodes[i], list_nodes[i-2])
        target= i+1
        if target>n-1:
            target=target-n
            G.add_edge(list_nodes[i], target)
```

Python • Tab Width: 8 • Ln 16, Col 17 • INS

So, we can actually import the function random here.

(Refer Slide Time: 01:09)



```
ring.py - gedit
ring.py
        target=target-n
        G.add_edge(list_nodes[i], target)
    else:
        G.add_edge(list_nodes[i],target)
    # i+1/i+2 -> n , we need to set this value to 0
    #       -> n+1, we need to set them to 1

def add_long_link(G):
    v1=random.choice(G.nodes())
    v2=random.choice(G.nodes())
    while(v1==v2):
        v1=random.choice(G.nodes())
        v2=random.choice(G.nodes())
    G.add_edge(v1,v2)

G=nx.Graph()
```

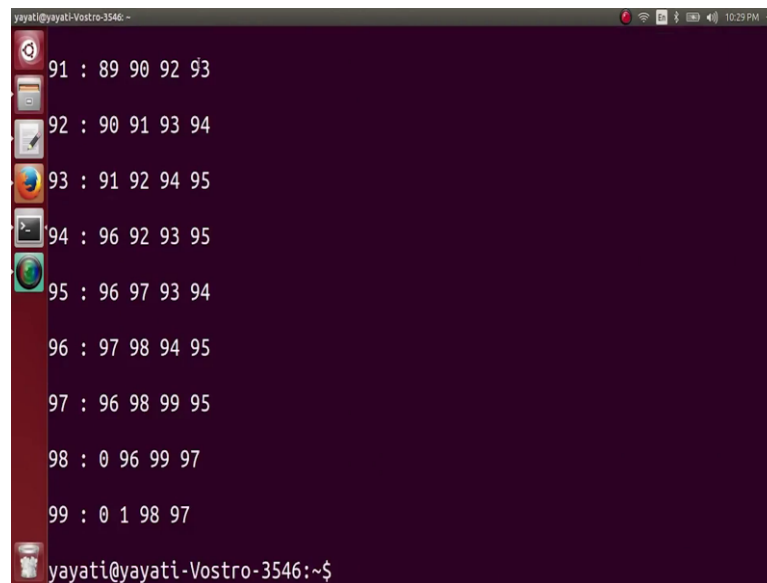
Saving file "/home/jayati/ring.py" ... Python • Tab Width: 8 • Ln 34, Col 25 • INS

And then what we have to do is define add long link G. And what we are going to do is let us take the first vertex it is nothing but random node choice for random element from the list G.nodes, and then v2, it also equal to random node choice G.nodes. And now, these two nodes can be same. We are going to put an edge between them only then these two nodes are different.

So, what I am going to do for that is why  $v1 == v2$ . So, if  $v$  by chance  $v1$  become equal to  $v2$  what I am going to do is I am going to repeat the same process again till I get to different values for  $v1$  and  $v2$ . And as soon as I get to different values for  $v1$  and  $v2$ , I put an edge between them `G.add_edge(v1, v2)`.

So, when I call this function once add long link it adds two edge it adds one edge between 2 randomly picked word this is in my network.

(Refer Slide Time: 02:33)



```
yayati@yayati-Vostro-3546:~$  
91 : 89 90 92 93  
92 : 90 91 93 94  
93 : 91 92 94 95  
94 : 96 92 93 95  
95 : 96 97 93 94  
96 : 97 98 94 95  
97 : 96 98 99 95  
98 : 0 96 99 97  
99 : 0 1 98 97
```

The image shows a terminal window with a dark purple background. On the left side, there is a vertical dock with several application icons. The terminal displays a list of numbers from 91 to 99, each followed by a colon and a list of its neighbors. For example, '91 : 89 90 92 93'. The prompt 'yayati@yayati-Vostro-3546:~\$' is visible at the bottom.

And we just check whether this code running fine or not, there is no error. So, this code is running fine.