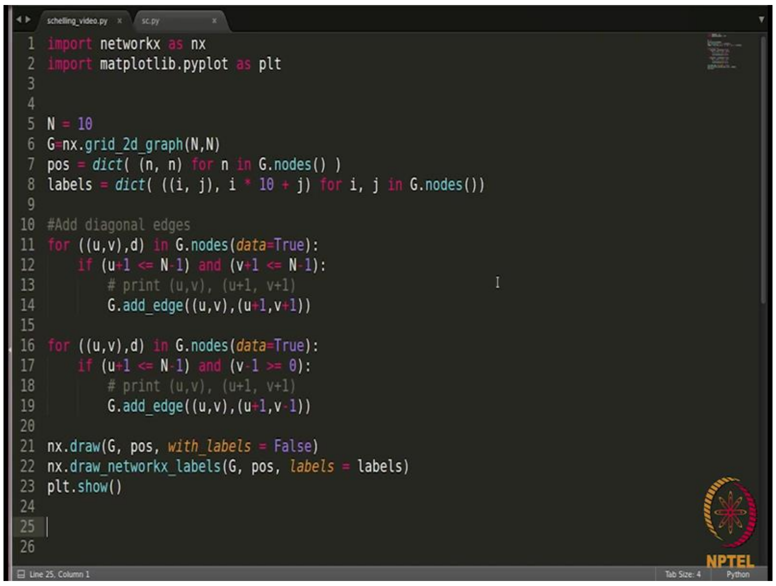
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture – 59
Homophily (Continued) & Positive and Negative Relationships
Schelling Model Implementation – Visualization and Getting a list of boundary & Internal nodes

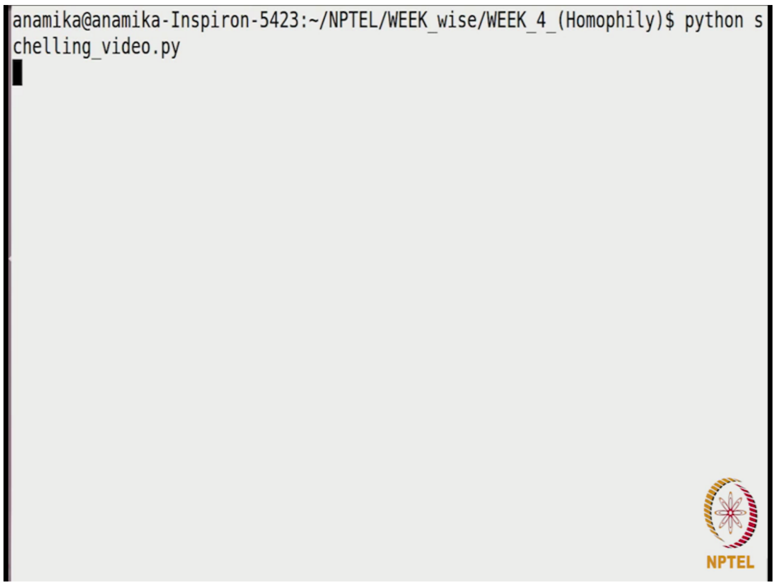
(Refer Slide Time: 00:05)



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


N = 10
G=nx.grid_2d_graph(N,N)
pos = dict( (n, n) for n in G.nodes() )
labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())

#Add diagonal edges
for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v+1 <= N-1):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v+1))

for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v-1 >= 0):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v-1))

nx.draw(G, pos, with_labels = False)
nx.draw_networkx_labels(G, pos, labels = labels)
plt.show()
```

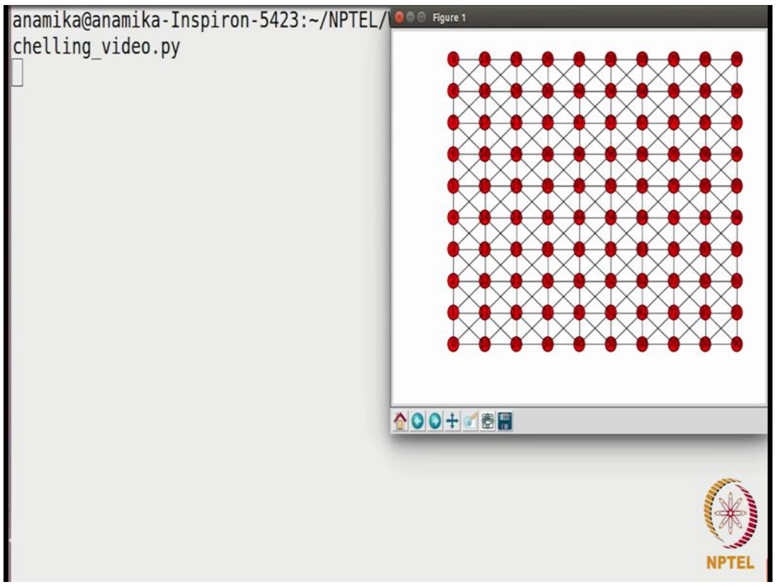
So, I have created this file and whatever we did in ipython console, I copied all that all those commands. So, here; so, there is nothing new here.

(Refer Slide Time: 00:20)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
```

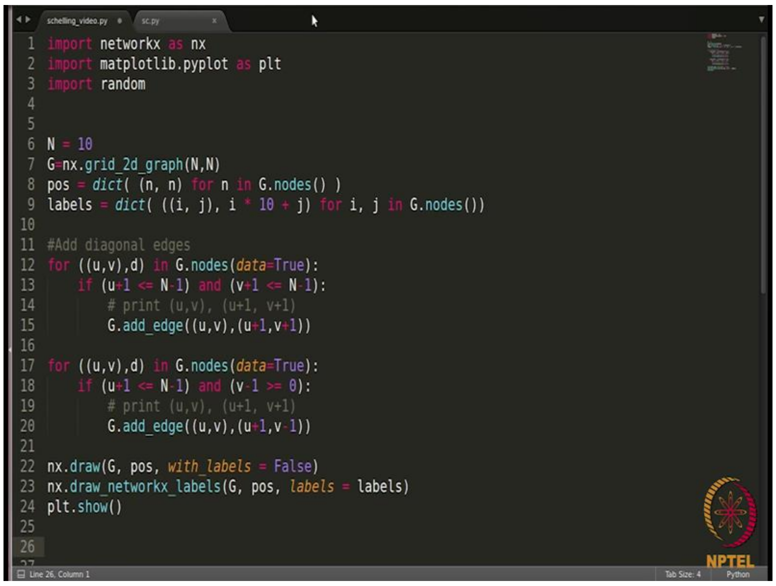
(Refer Slide Time: 00:27)



```
anamika@anamika-Inspiron-5423:~/NPTEL/
chelling_video.py
```

Let me show you the output of this. So, this is where we left of in the previous video and this is the graph that we were getting now we will continue with from here after having created the network we have to assign people to the nodes. So, as I explained we have to assign people of 2 kinds people of type one people of type 2 and there will be some nodes which will be empty. So, what we can do is we can make use of package random.

(Refer Slide Time: 01:00)

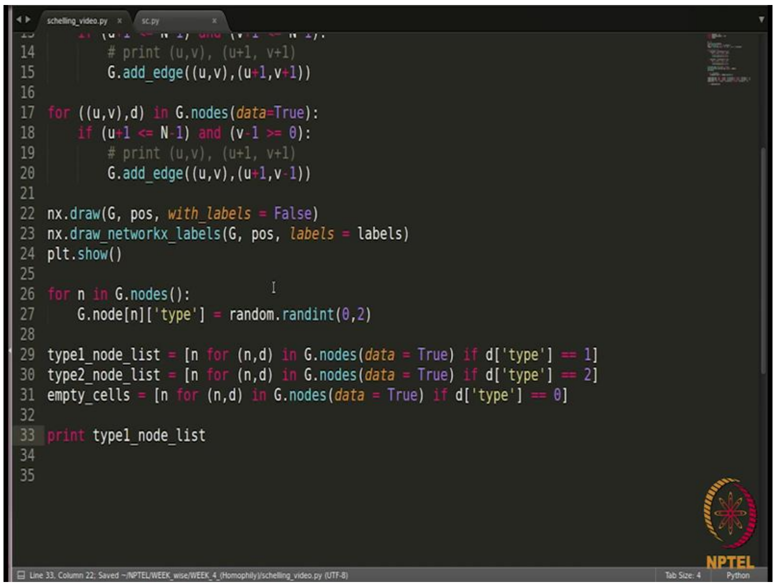


```
import networkx as nx
import matplotlib.pyplot as plt
import random


N = 10
G=nx.grid_2d_graph(N,N)
pos = dict( (n, n) for n in G.nodes() )
labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())

#Add diagonal edges
for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v+1 <= N-1):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v+1))

for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v-1 >= 0):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v-1))

nx.draw(G, pos, with_labels = False)
nx.draw_networkx_labels(G, pos, labels = labels)
plt.show()
```

So, I will write both random to assign people to nodes what we can do here is that we can assign and attribute that is type to every node in the network.

So, this attribute type will have values from 0 1 or 2 if the type is equal to 0 the node will be considered empty if the type is equal to 1, the node will be considered a person of type 1 if the type = 2 the node will be considered a person of type 2. So, we will be following this convention.

(Refer Slide Time: 01:41)



```
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v+1))

for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v-1 >= 0):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v-1))

nx.draw(G, pos, with_labels = False)
nx.draw_networkx_labels(G, pos, labels = labels)
plt.show()

for n in G.nodes():
    G.node[n]['type'] = random.randint(0,2)

type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]

print type1_node_list
```

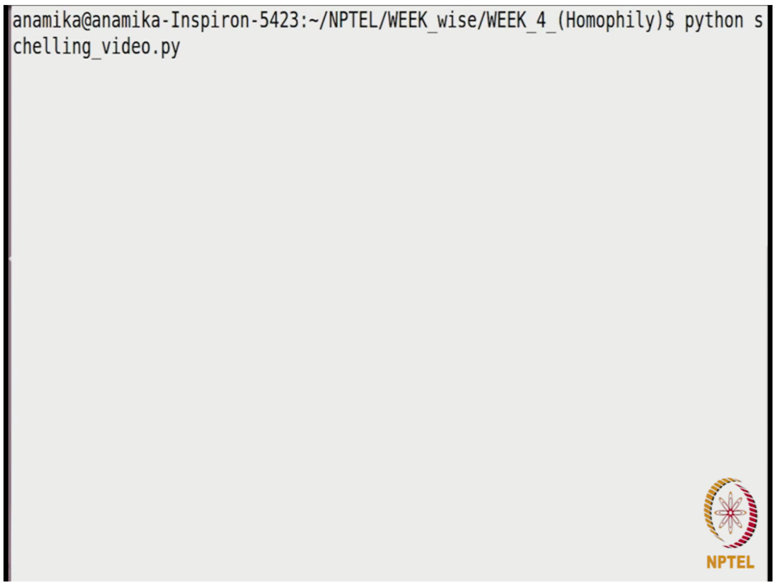
Let me assign the type to the nodes. So, what we can do is for n in G.nodes, G.node. So, they must assign a type to n. So, I will write type I want to assign a type randomly. So, I will write random.randint and this should re this should go from 0 to 2. So, 0 1 2 any random value could be assigned to the type of to the type of node n. Now after we have assign the type to every node, we should maintain a list of type one people 2 people and the list of empty nodes or you can say empty cells.

So, we will create three lists for that purpose. So, I will write type one node list sorry is equal to. So, this has to be list and what we can do here is n for (n, d) now I will explain what this means G.nodes (data = true) if the type of t = 1. So, let me tell you what I have done here. So, when you when you can G.nodes you get only a list of nodes, but in case we have assigned an attribute to the nodes and still we write G.nodes still we get only the list of nodes we do not get the get the tie we do not get the value of that attitude in case you want the value that attribute you have to explicitly write data = true.

So, when you write G.nodes (data = true) you get n and d where n is a node and d is a dictionary which will contain the attribute and its value. So, since we want to get the type of every node we will write G.nodes (data = true) and that will write us (n, d) where n is a node and d will give us the type. So, we have written if d['type'] = 1. So, if d['type'] = 1 the node will be of type 1 sorry please pattern by back though out. So, this is for type one node list similarly we will get the type 2 list it just copies paste I am sorry. So, I will write type 2 node list is equal to n for (n, d) in G.nodes that as will true if d['type'] = 2, right.

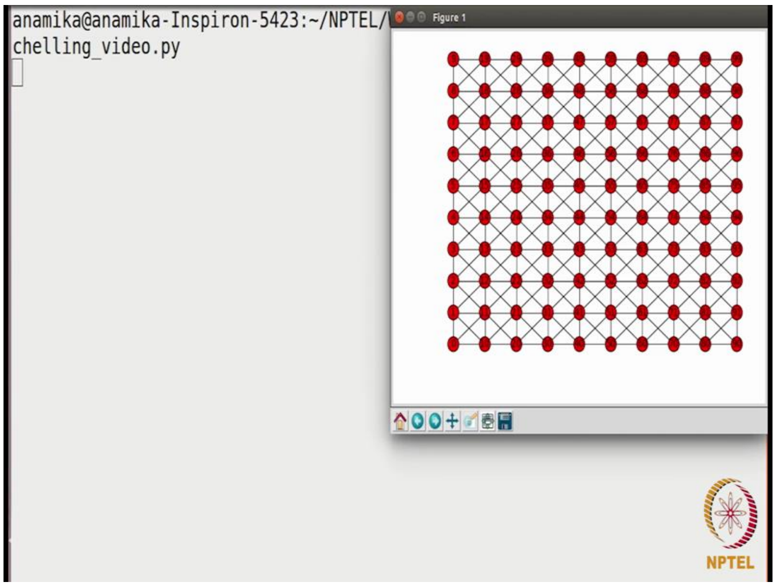
So, similarly we can get a list of empty nodes. So, we can write empty cells is equal to, I will just copy paste where the type = 0. So, they should work we can also verify let me try printing the type one node list type one node list let me check whether it works find.

(Refer Slide Time: 05:07)



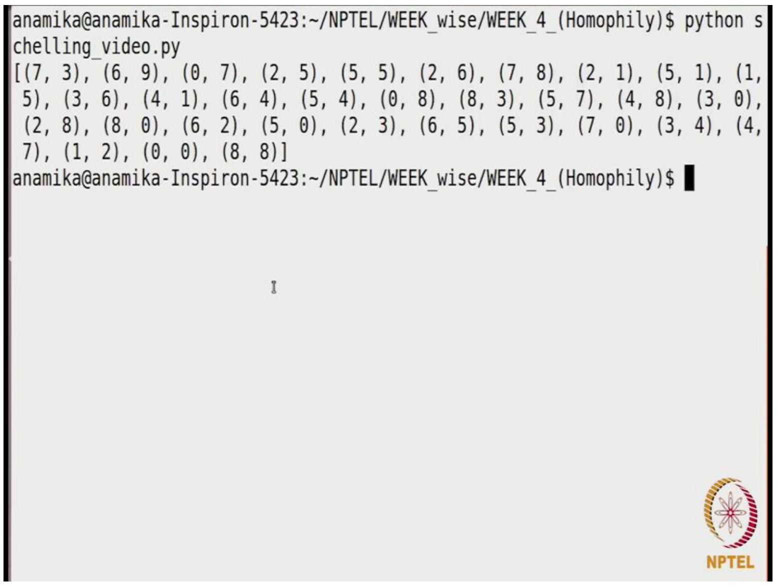
```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
```

(Refer Slide Time: 05:10)



```
anamika@anamika-Inspiron-5423:~/NPTEL/
chelling_video.py
```

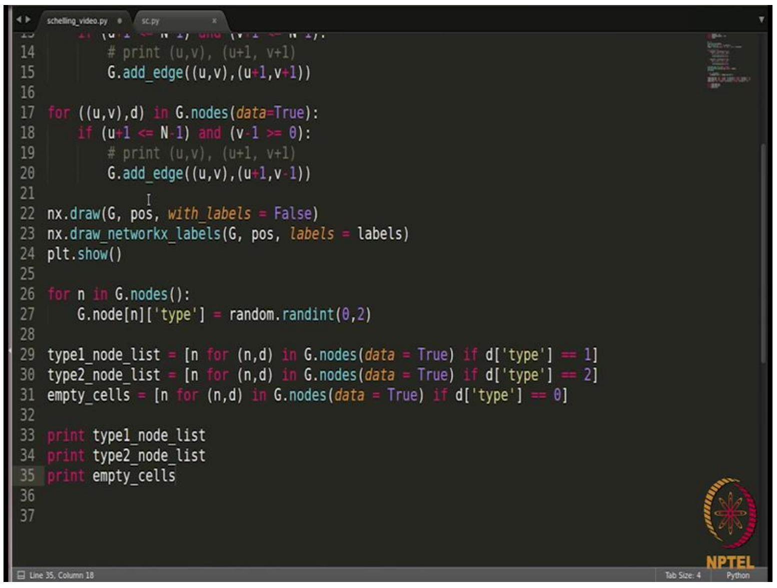
(Refer Slide Time: 05:11)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(7, 3), (6, 9), (0, 7), (2, 5), (5, 5), (2, 6), (7, 8), (2, 1), (5, 1), (1,
 5), (3, 6), (4, 1), (6, 4), (5, 4), (0, 8), (8, 3), (5, 7), (4, 8), (3, 0),
 (2, 8), (8, 0), (6, 2), (5, 0), (2, 3), (6, 5), (5, 3), (7, 0), (3, 4), (4,
 7), (1, 2), (0, 0), (8, 8)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

So, this is the list of the list of nodes which are n type 1 similarly we can print the other lists as well let me just verify and then we will continue empty cells yeah.

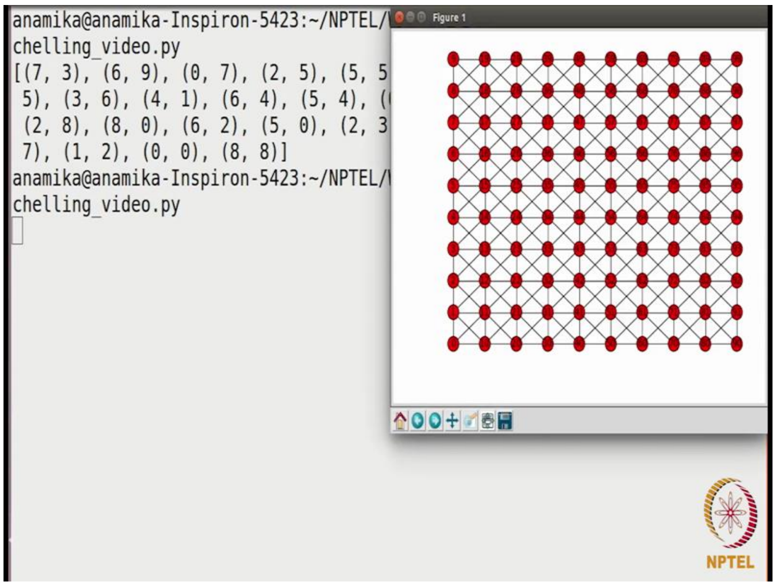
(Refer Slide Time: 05:20)



```
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v+1))

for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v-1 >= 0):
        # print (u,v), (u+1, v+1)
        G.add_edge((u,v),(u+1,v-1))

nx.draw(G, pos, with_labels = False)
nx.draw_networkx_labels(G, pos, labels = labels)
plt.show()

for n in G.nodes():
    G.node[n]['type'] = random.randint(0,2)

type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]

print type1_node_list
print type2_node_list
print empty_cells
```

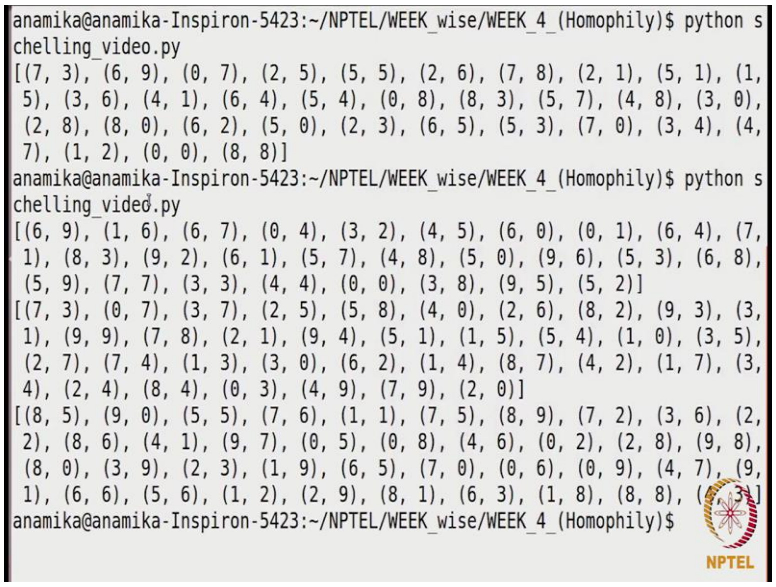
So, it is giving all the three lists we are going to make use of these lists. So, I am going to comment this because we do not need to print.

(Refer Slide Time: 05:32)



```
anamika@anamika-Inspiron-5423:~/NPTEL/\
chelling_video.py
[(7, 3), (6, 9), (0, 7), (2, 5), (5, 5
 5), (3, 6), (4, 1), (6, 4), (5, 4), (
 (2, 8), (8, 0), (6, 2), (5, 0), (2, 3
 7), (1, 2), (0, 0), (8, 8)]
anamika@anamika-Inspiron-5423:~/NPTEL/\
chelling_video.py
```

(Refer Slide Time: 05:34)

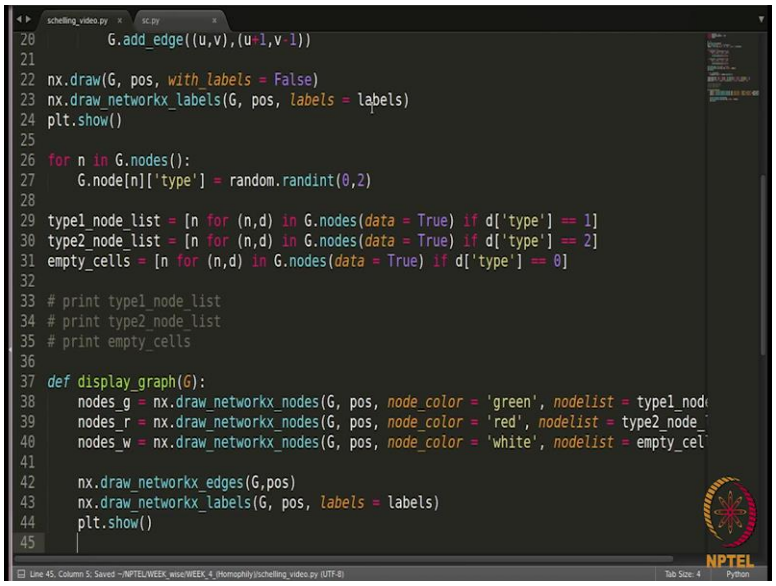


```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(7, 3), (6, 9), (0, 7), (2, 5), (5, 5), (2, 6), (7, 8), (2, 1), (5, 1), (1,
 5), (3, 6), (4, 1), (6, 4), (5, 4), (0, 8), (8, 3), (5, 7), (4, 8), (3, 0),
 (2, 8), (8, 0), (6, 2), (5, 0), (2, 3), (6, 5), (5, 3), (7, 0), (3, 4), (4,
 7), (1, 2), (0, 0), (8, 8)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(6, 9), (1, 6), (6, 7), (0, 4), (3, 2), (4, 5), (6, 0), (0, 1), (6, 4), (7,
 1), (8, 3), (9, 2), (6, 1), (5, 7), (4, 8), (5, 0), (9, 6), (5, 3), (6, 8),
 (5, 9), (7, 7), (3, 3), (4, 4), (0, 0), (3, 8), (9, 5), (5, 2)]
[(7, 3), (0, 7), (3, 7), (2, 5), (5, 8), (4, 0), (2, 6), (8, 2), (9, 3), (3,
 1), (9, 9), (7, 8), (2, 1), (9, 4), (5, 1), (1, 5), (5, 4), (1, 0), (3, 5),
 (2, 7), (7, 4), (1, 3), (3, 0), (6, 2), (1, 4), (8, 7), (4, 2), (1, 7), (3,
 4), (2, 4), (8, 4), (0, 3), (4, 9), (7, 9), (2, 0)]
[(8, 5), (9, 0), (5, 5), (7, 6), (1, 1), (7, 5), (8, 9), (7, 2), (3, 6), (2,
 2), (8, 6), (4, 1), (9, 7), (0, 5), (0, 8), (4, 6), (0, 2), (2, 8), (9, 8),
 (8, 0), (3, 9), (2, 3), (1, 9), (6, 5), (7, 0), (0, 6), (0, 9), (4, 7), (9,
 1), (6, 6), (5, 6), (1, 2), (2, 9), (8, 1), (6, 3), (1, 8), (8, 8), (4, 3)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

Now, what we can do is now that we have assigned type to every node. So, way we have 2 different kinds of people in our network it will be nice if he represents them with a different colour.

So, that it is easy to visualise the kind of people that are present in a given cell.

(Refer Slide Time: 06:09)



```
        G.add_edge((u,v),(u+1,v-1))

nx.draw(G, pos, with_labels = False)
nx.draw_networkx_labels(G, pos, labels = labels)
plt.show()

for n in G.nodes():
    G.node[n]['type'] = random.randint(0,2)

type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]

# print type1_node_list
# print type2_node_list
# print empty_cells

def display_graph(G):
    nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_nod
    nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node_
    nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cel

    nx.draw_networkx_edges(G,pos)
    nx.draw_networkx_labels(G, pos, labels = labels)
    plt.show()
    
```

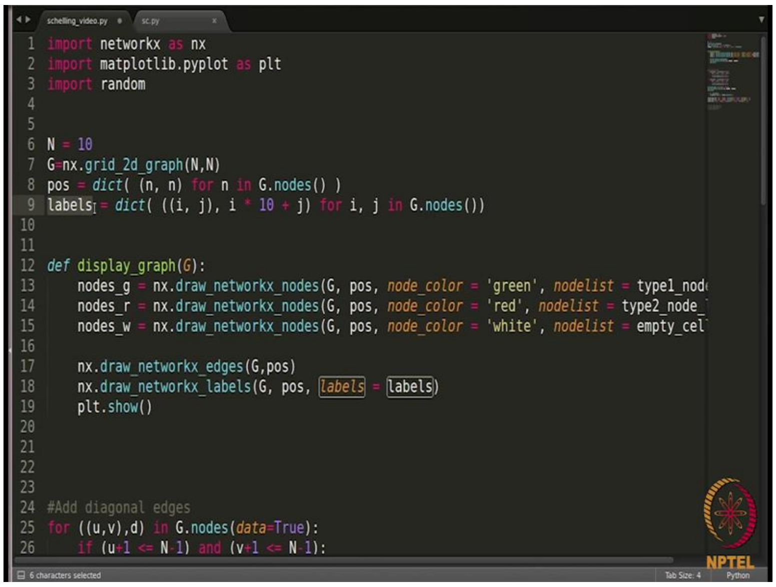
```
Line 45, Column 5; Saved ~/NPTEL/WEEK_wise/WEEK_4_(Homophily)/schelling_video.py (UTF-8)          Tab Size: 4     Python
```

So, let me create a function for that I will create a function say display graph. So, I am just going to decorate the graph here it is up to you how much you want to do here how much you want t, how you want to display how much you want to decorate it is up to you can skip some of the a steps here. If you want who is a what I want is type one node should be displayed in say green colour type 2 nodes should be displayed in say red colour and empty cell should be say display displayed in white colour. So, the function that we will use and it is not draw network x nodes where we can specify a list of nodes that should be displayed by a given colour. So, what we can do is nx.draw networkx nodes the parameters will be G and this pos parameter that we posed already.

And what is to be want I everyone the node colours. So, I will write node colour is equal to green I wanted to be green and then what should be the nodes for which we want to as for whom we want to assign the pink colour. So, I will write node list is equal to type one node list right and going to copy the next kind of nodes we want in red colour. So, I will write nodes are I want in red colour and then type to we are doing this all this here because it will be nice to visualise the kind of transitions that take place. So, all though this is optional step it will be in it will help us to visualise the things I have just to go out the yeah. So, we want empty cells here I will write empty cells and I want why it here.

So, now we have drawn the nodes and, but the edges have not been drawn. So, we have to draw the edges. So, we will write nx.draw networkx edges and there again the 2 parameters we will pos G and pos the positioning since we are since we are manually draw in the nodes and edges labels we will not come. So, again we have to manually drop the labels as well. So, you write nx.draw_networkx labels and again we I think we used at already. So, yeah this one we can just copy this, they should work and anything that we skipped let us see plt.show. So, now, we have to call this function let me put this function on the top so that we can call it.

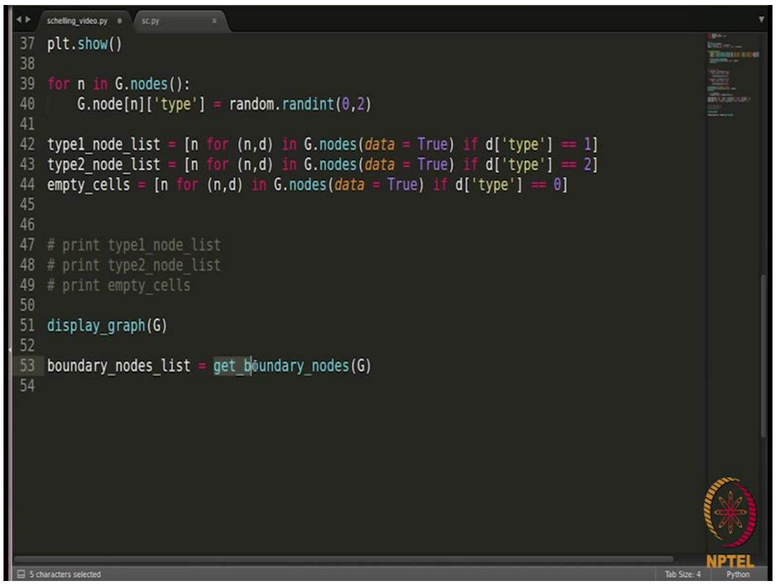
(Refer Slide Time: 09:37)



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


N = 10
G=nx.grid_2d_graph(N,N)
pos = dict( (n, n) for n in G.nodes() )
labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())


def display_graph(G):
    nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_nod
    nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node_
    nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cel

    nx.draw_networkx_edges(G,pos)
    nx.draw_networkx_labels(G, pos, labels = labels)
    plt.show()




#Add diagonal edges
for ((u,v),d) in G.nodes(data=True):
    if (u+1 <= N-1) and (v+1 <= N-1):
```

So, I will cut it from here and I will right it here this is because we want to make use of labels. So, I want to define the labels dictionary before I declare this function.

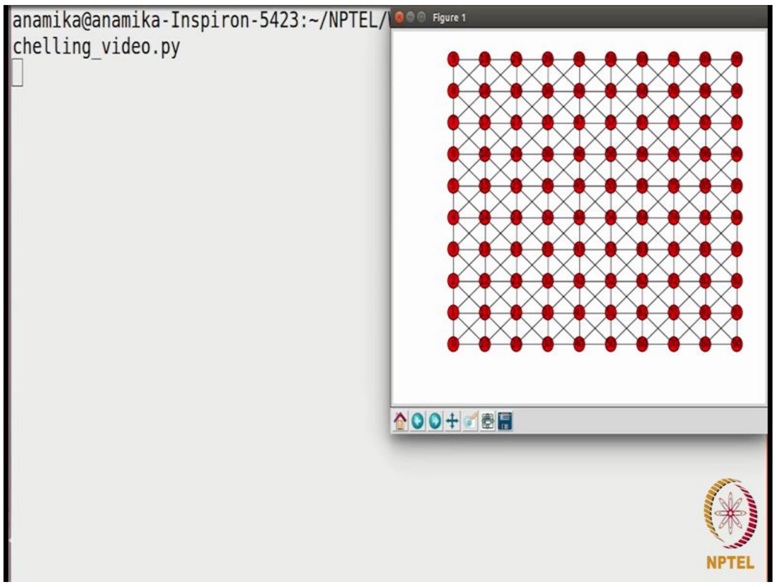
(Refer Slide Time: 09:54)



```
plt.show()

for n in G.nodes():
    G.node[n]['type'] = random.randint(0,2)

type1_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 1]
type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]


# print type1_node_list
# print type2_node_list
# print empty_cells

display_graph(G)

boundary_nodes_list = get_boundary_nodes(G)
```

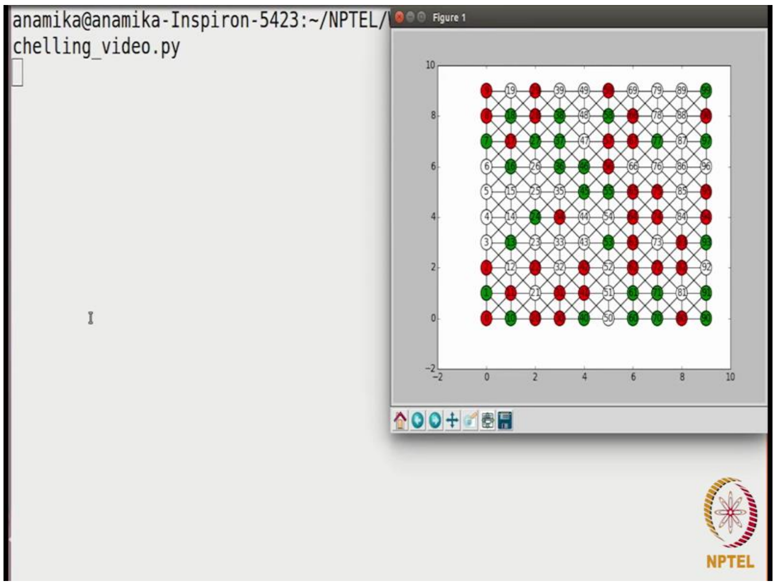
So, I will call this function here a write display graph G let me execute it.

(Refer Slide Time: 10:01)



```
anamika@anamika-Inspiron-5423:~/NPTEL/
chelling_video.py
```

(Refer Slide Time: 10:05)

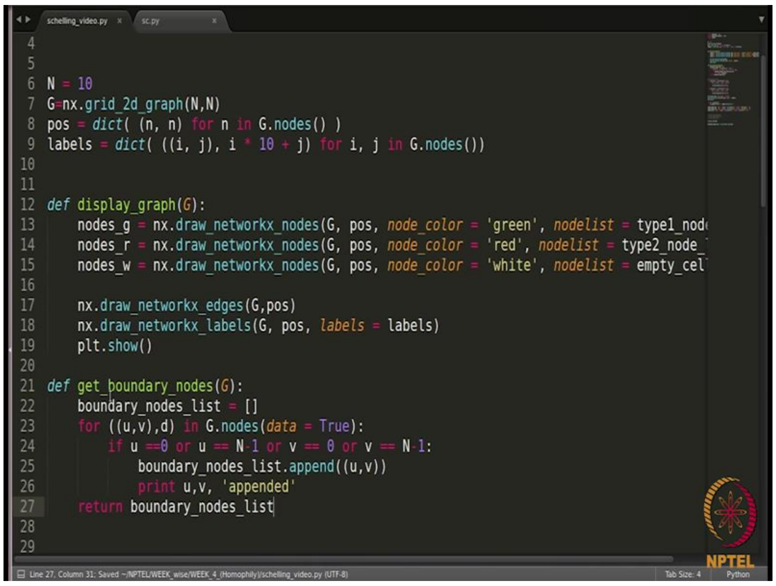

```
anamika@anamika-Inspiron-5423:~/NPTEL/\
chelling_video.py
```

This is the graph that we initially displayed in this is the graph that now we are displaying which looks quite better and we are able to see which nodes belong to which type.

So, it looks pretty good if you want you can change the font size you can change the node size so that all you can play around. So, what is the next step we have the network we have the people. The next step is to identify the people were unsatisfied now how do if how do you find which people are unsatisfied we should be able to know for a given node which are its neighbour right it only then we will be able to you know tell whether this node is satisfied or unsatisfied. Now to be able to know which the neighbours of a are given node we should know whether the node is an internal node or is it a boundary node. So, that we must find out. So, for that matter it will be helpful if we create a list of boundary nodes as well as a list of internal nodes.

So, maybe we can create a function for that to get the list of boundary nodes. So, what we can write is get boundary nodes lets pos the graph here boundary nodes list is equal to this. So, basically we are going to put all the nodes which are aligning in the boundary into this list through this function which is get boundary nodes let us create this functional we will see how we can do that.

(Refer Slide Time: 11:58)



schelling_video.py   sc.py

```python


N = 10
G=nx.grid_2d_graph(N,N)
pos = dict( (n, n) for n in G.nodes() )
labels = dict( ((i, j), i * 10 + j) for i, j in G.nodes())


def display_graph(G):
    nodes_g = nx.draw_networkx_nodes(G, pos, node_color = 'green', nodelist = type1_nod
    nodes_r = nx.draw_networkx_nodes(G, pos, node_color = 'red', nodelist = type2_node_
    nodes_w = nx.draw_networkx_nodes(G, pos, node_color = 'white', nodelist = empty_cel

    nx.draw_networkx_edges(G,pos)
    nx.draw_networkx_labels(G, pos, labels = labels)
    plt.show()

def get_boundary_nodes(G):
    boundary_nodes_list = []
    for ((u,v),d) in G.nodes(data = True):
        if u ==0 or u == N-1 or v == 0 or v == N-1:
            boundary_nodes_list.append((u,v))
            print u,v, 'appended'
    return boundary_nodes_list


```

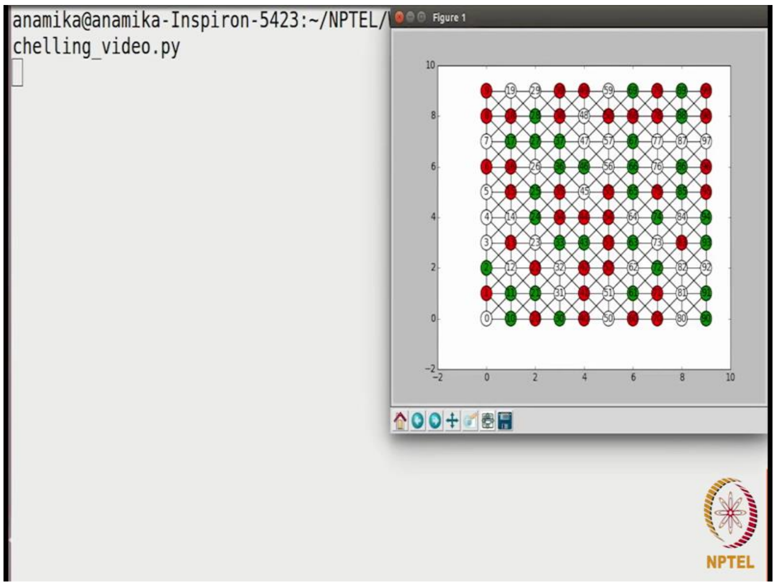
Line 27, Column 31; Saved ~/NPTEL/WEEK_wise/WEEK_4_(Homophily)/schelling_video.py (UTF-8)   Tab Size: 4   Python

So, I will create this function here now how to do this we should know which nodes are boundary nodes if you remember the kind of grid that we were getting you will know that the nodes which have either u or v if (u, v) is a node the nodes which have either u as 0 or v as 0 is a boundary node also the nodes which have either u as n - 1 or v as n minus one they are also boundary nodes.

So, you can check the grid and see how it how it comes out to be. So, let us write this function lets create of lists which we will be writing. So, I will write boundary nodes list is equal to empty. So, I will write for u, v what else is there t which is a type we can actually skip the type here because we do not need this lets see in G.nodes(data = true). So, what I was saying was data is not needed for these particular steps. So, we can skip that is one and we can keep that is one. So, and choosing to keep it just like the now as I told you if u = 0 or u = n - 1 or v = 0 or v = n - 1 then the node is a boundary node.

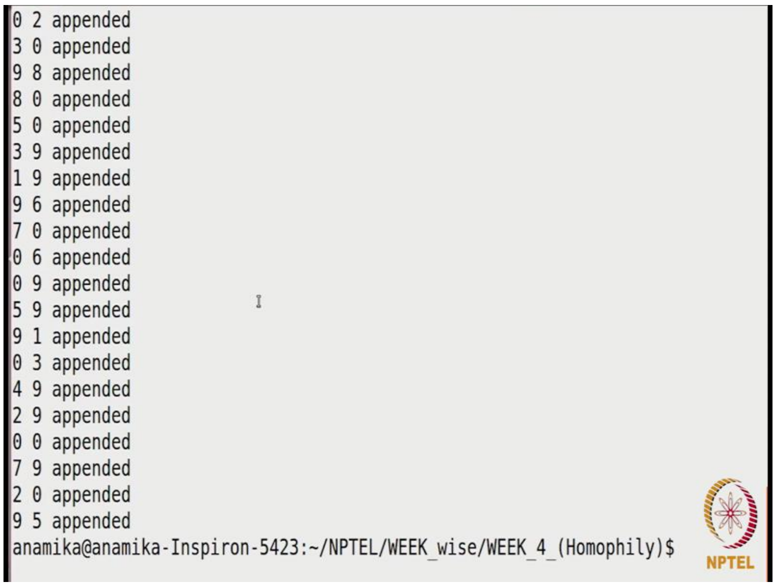
So, what we will do is the boundary nodes lists.append we are going to append u v that yeah we should go let me print also it is to cross check whether things are fine or not. So, let us print this whenever a node is appended to the list there will be is this state. So, once that is we are going to written this list. So, we have already called this function yes let us call let us execute. Before that let me remove this displaying of the graph because now we are because now we have created a function and we are getting a better visualizations.

(Refer Slide Time: 14:35)



```
anamika@anamika-Inspiron-5423:~/NPTEL/
chelling_video.py
```

(Refer Slide Time: 14:38)

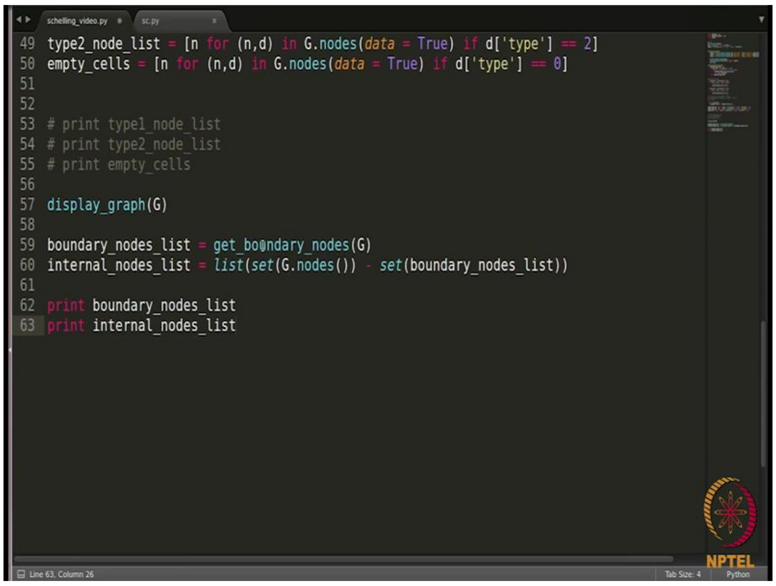


```
0 2 appended
3 0 appended
9 8 appended
8 0 appended
5 0 appended
3 9 appended
1 9 appended
9 6 appended
7 0 appended
0 6 appended
0 9 appended
5 9 appended
9 1 appended
0 3 appended
4 9 appended
2 9 appended
0 0 appended
7 9 appended
2 0 appended
9 5 appended
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

So, I have commented on that. So, this is the graph I am closing it yeah. So, we are we are getting things working fine these are the nodes which are appended to the list.

Now, the second step is to get the list of the internal nodes right. So, what we can do is if we subtract the boundary nodes from the total nodes; obviously, whatever remains is the list of internal nodes.

(Refer Slide Time: 15:05)

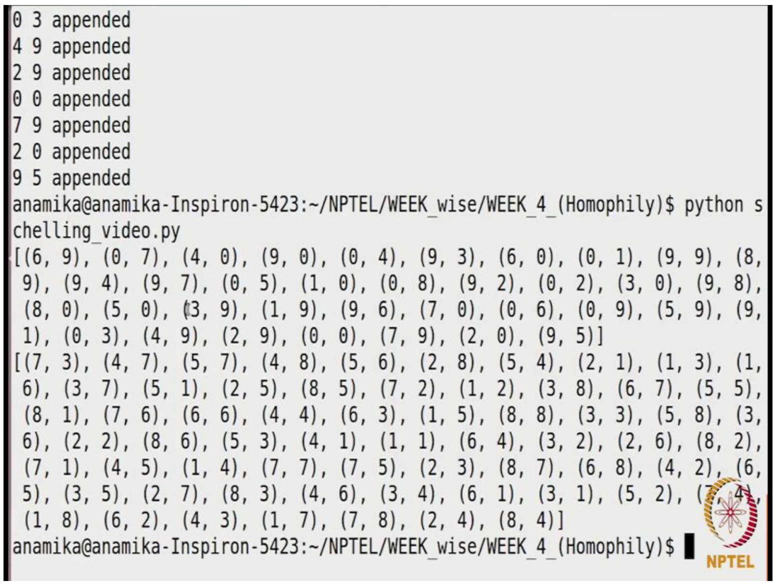


```
type2_node_list = [n for (n,d) in G.nodes(data = True) if d['type'] == 2]
empty_cells = [n for (n,d) in G.nodes(data = True) if d['type'] == 0]


# print type1_node_list
# print type2_node_list
# print empty_cells

display_graph(G)

boundary_nodes_list = get_boundary_nodes(G)
internal_nodes_list = list(set(G.nodes()) - set(boundary_nodes_list))

print boundary_nodes_list
print internal_nodes_list
```

So, what we can do is I will create a list for internal nodes internal nodes list is equal to; now what is this I told you this technique if you want to subtract one list from the other you can use is set function. So, I am going to convert list of all the nodes into a set. So, I will write G.nodes; yeah minus set or should I write boundary nodes list yeah this work. So, let me try printing these 2 lists just to cross it.

So, I will write print boundary nodes list and print internal nodes list yeah let me command that appending. So, that we get only what we want to print (Refer Time: 16:08) that yeah in this function I am going to commentary this let's check yeah. So, we are getting 2 lists here first lists are boundary node list and second list is internal nodes. So, we are doing well.

(Refer Slide Time: 16:20)



```
0 3 appended
4 9 appended
2 9 appended
0 0 appended
7 9 appended
2 0 appended
9 5 appended
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$ python s
chelling_video.py
[(6, 9), (0, 7), (4, 0), (9, 0), (0, 4), (9, 3), (6, 0), (0, 1), (9, 9), (8,
 9), (9, 4), (9, 7), (0, 5), (1, 0), (0, 8), (9, 2), (0, 2), (3, 0), (9, 8),
 (8, 0), (5, 0), (3, 9), (1, 9), (9, 6), (7, 0), (0, 6), (0, 9), (5, 9), (9,
 1), (0, 3), (4, 9), (2, 9), (0, 0), (7, 9), (2, 0), (9, 5)]
[(7, 3), (4, 7), (5, 7), (4, 8), (5, 6), (2, 8), (5, 4), (2, 1), (1, 3), (1,
 6), (3, 7), (5, 1), (2, 5), (8, 5), (7, 2), (1, 2), (3, 8), (6, 7), (5, 5),
 (8, 1), (7, 6), (6, 6), (4, 4), (6, 3), (1, 5), (8, 8), (3, 3), (5, 8), (3,
 6), (2, 2), (8, 6), (5, 3), (4, 1), (1, 1), (6, 4), (3, 2), (2, 6), (8, 2),
 (7, 1), (4, 5), (1, 4), (7, 7), (7, 5), (2, 3), (8, 7), (6, 8), (4, 2), (6,
 5), (3, 5), (2, 7), (8, 3), (4, 6), (3, 4), (6, 1), (3, 1), (5, 2), (7, 4),
 (1, 8), (6, 2), (4, 3), (1, 7), (7, 8), (2, 4), (8, 4)]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_4_(Homophily)$
```

So, now we have a list of boundary nodes internal nodes what is the next step the next step is to identify the unsatisfied nodes and then moving them to a new location we will do that in the next part of the video.