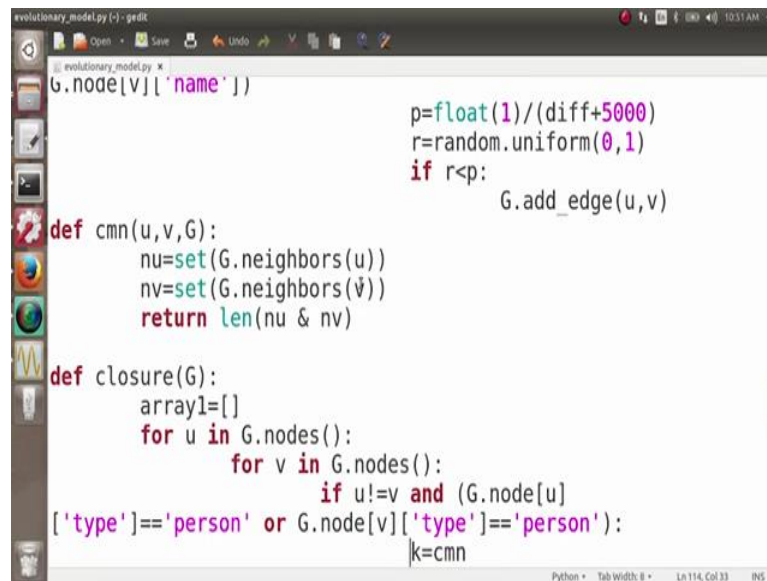


Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture - 53
Strong and Weak Relationships (Continued) and Homophily
Fatman Evolutionary Model - Storing and analyzing longitudinal data

(Refer Slide Time: 00:05)

A screenshot of a Python IDE window titled 'evolutionary_model.py - edit'. The code defines a graph structure and functions for analyzing it. The code is as follows:

```
G.node[v]['name'])

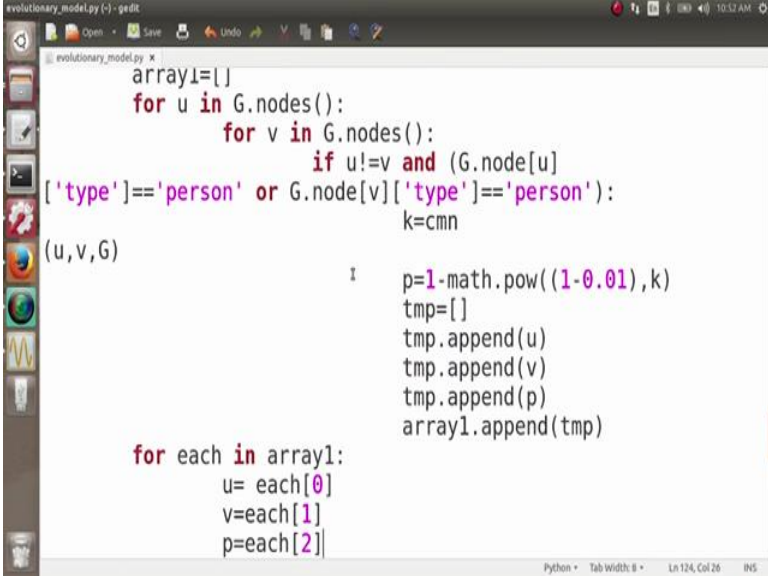
p=float(1)/(diff+5000)
r=random.uniform(0,1)
if r<p:
    G.add_edge(u,v)

def cmn(u,v,G):
    nu=set(G.neighbors(u))
    nv=set(G.neighbors(v))
    return len(nu & nv)

def closure(G):
    array1=[]
    for u in G.nodes():
        for v in G.nodes():
            if u!=v and (G.node[u]
['type']=='person' or G.node[v]['type']=='person'):
                k=cmn
```

Next what we want to do assume we want to analyze what all is happening in this graph. So, it is not actually possible to analyze it very nicely by visualizing the graph and looking it as a figure every time. So, some nicely written functions can help us in the analysis of this graph.

(Refer Slide Time: 00:25)

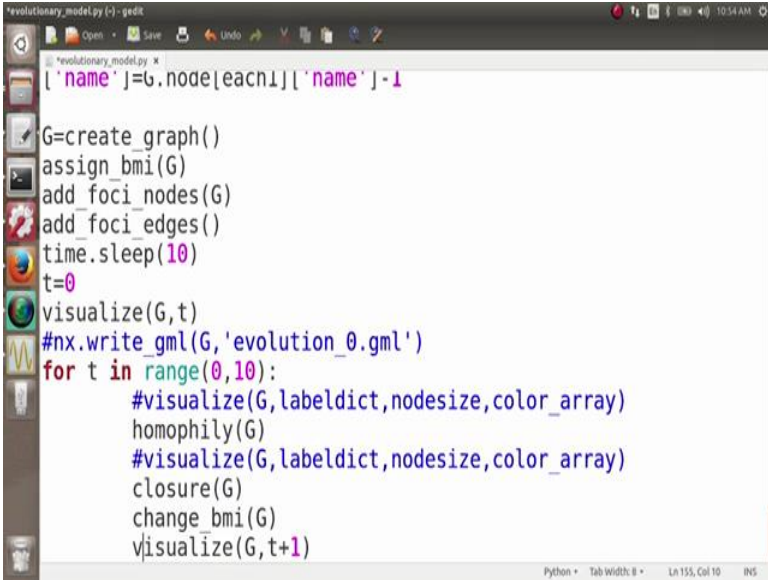


```
evolutionary_model.py - gedit
evolutionary_model.py x
array1=[]
for u in G.nodes():
    for v in G.nodes():
        if u!=v and (G.node[u]
['type']=='person' or G.node[v]['type']=='person'):
            k=cmn
            (u,v,G)
            p=1-math.pow((1-0.01),k)
            tmp=[]
            tmp.append(u)
            tmp.append(v)
            tmp.append(p)
            array1.append(tmp)

for each in array1:
    u= each[0]
    v=each[1]
    p=each[2]
```

So, for that what we will do is, first of all we will store whatever is happening in this graph as different snapshots. So, we will have different network store for different snapshots, next we will look at these networks one by one and look at how this network is changing over time.

(Refer Slide Time: 00:41)



```
evolutionary_model.py - gedit
evolutionary_model.py x
['name']=G.node[each1]['name']-1

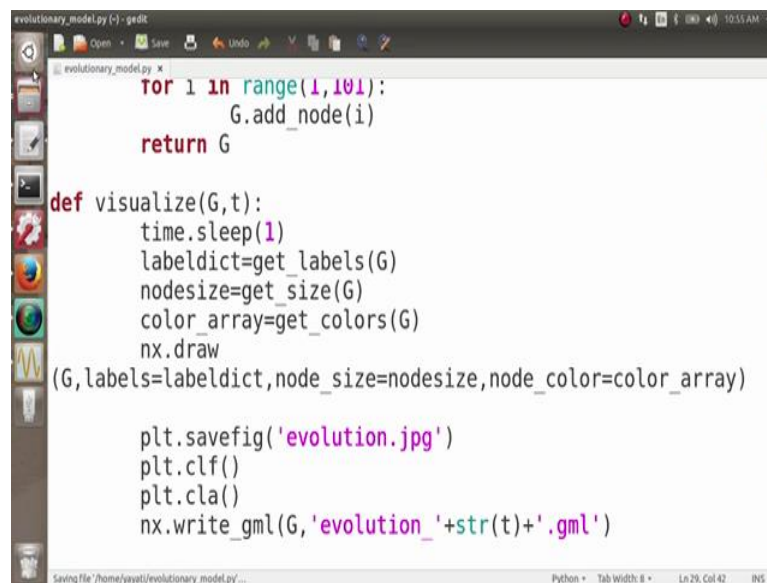
G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
add_foci_edges()
time.sleep(10)
t=0
visualize(G,t)
#nx.write_gml(G,'evolution_0.gml')
for t in range(0,10):
    #visualize(G,labeldict,nodesize,color_array)
    homophily(G)
    #visualize(G,labeldict,nodesize,color_array)
    closure(G)
    change_bmi(G)
    visualize(G,t+1)
```

So, what we do here is along with visualizing G we try to store this network. So, we call a function here let us say store G, and this store G takes a time value with itself let us say its t or let us say the time value is i. So, initially this i is 0 when nothing is done. So, we

call store G , i, and a better option is when we visualize this graph that time only, we can store this network what do we do is. So, for the very first of all outside the loop we store it manually. So, for storing it we use we stored it in a gml format. So, for storing it in a gml format we use the function nx.write_gml as you know we have done it before nx.write_gml(G) and let us say the file name is gain evolution_0.gml. So, this is a first file name.

Next when the time runs from 0 to 10 what we are doing is we are calling these three functions and we are visualizing this network. So, let us pass the value $t + 1$ here. So, why $t + 1$? t was here t starts from 0, but we want the indices of this network from 1. So, that is evolution_1.gml, evolution_2.gml. So, we have here visualize (G , $t + 1$) and actually we should be doing it here also because the format of the function is same. So, we need to pass the same parameters here. So, let us first have $t = 0$ and then we call visualize(G , t) you can actually come and take and then our t then we called visualized (G , $t + 1$).

(Refer Slide Time: 02:56)



```
evolutionary_model.py - . gedit
evolutionary_model.py x
for i in range(1,101):
    G.add_node(i)
return G

def visualize(G,t):
    time.sleep(1)
    labeldict=get_labels(G)
    nodesize=get_size(G)
    color_array=get_colors(G)
    nx.draw
    (G,labels=labeldict,node_size=nodesize,node_color=color_array)

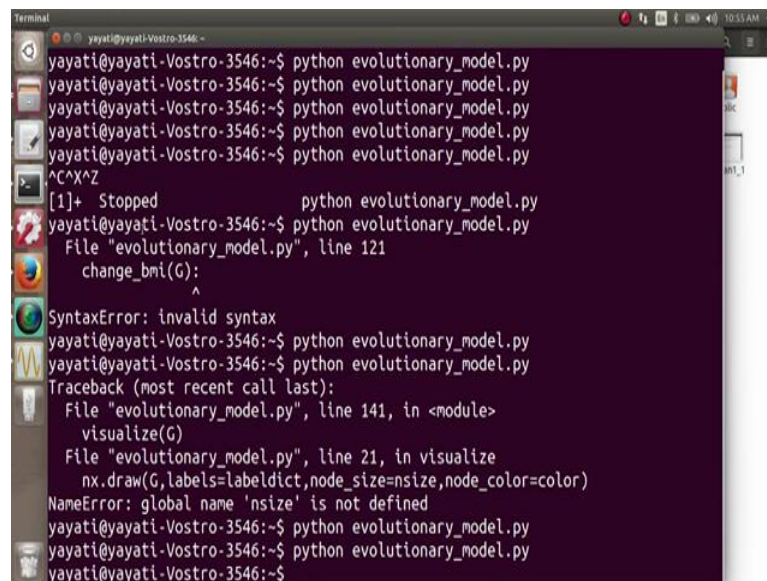
    plt.savefig('evolution.jpg')
    plt.clf()
    plt.cla()
    nx.write_gml(G,'evolution_'+str(t)+'.gml')
```

And what happens here coming back to our visualization function. So, it now it is taking an extra parameter which is the t at which the time t the graph is to be stored. So, what we do here is to store the graph after doing all this, we have a function here nx.write_gml and then we write this graph G as what should be its name its name should be evolution underscore, and then we have this value of t here. So, we do a string append operation

and what do we append to this string is the value of t as a string. So, it is evolution_0 then evolution_1 and we append.gml to it.

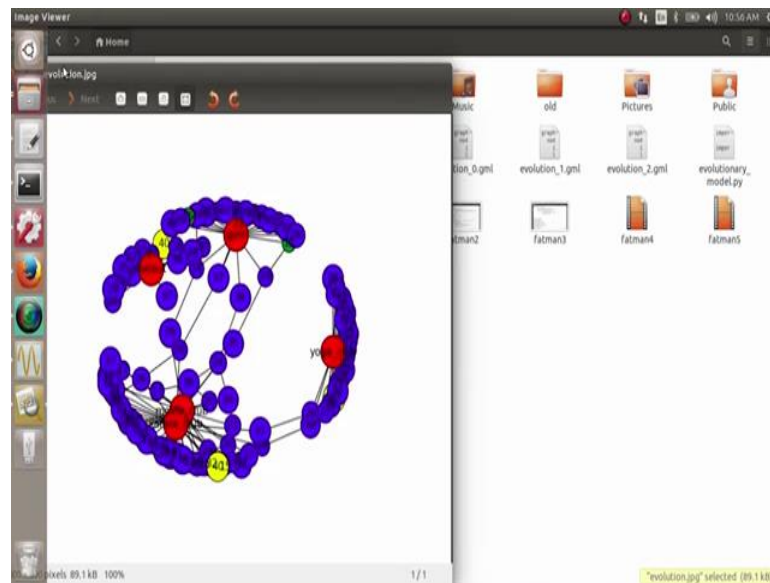
So, how this graph gets stored is G evolution_0.gml, evolution_1.gml and so on and then will be loading this graphs and we look at some properties of this evolution of this completes evolution. So, let us see let us try to run it.

(Refer Slide Time: 04:03)

A terminal window with a dark purple background and white text. The window title is 'Terminal'. The prompt is 'yayati@yayati-Vostro-3546:~'. The user enters 'python evolutionary_model.py' five times. The first three times, the command is accepted. The fourth time, the user enters '^C^X^Z' (Ctrl-C). The prompt returns to '\$'. The fifth time, the user enters 'python evolutionary_model.py'. The terminal shows a syntax error: 'SyntaxError: invalid syntax' at line 121, column 12, in the function 'change_bmi(G)'. The user enters 'python evolutionary_model.py' again, and the terminal shows a traceback error: 'NameError: global name 'nsize' is not defined' at line 21, in the function 'visualize(G)'. The user enters 'python evolutionary_model.py' a third time, and the terminal shows the same 'NameError'.

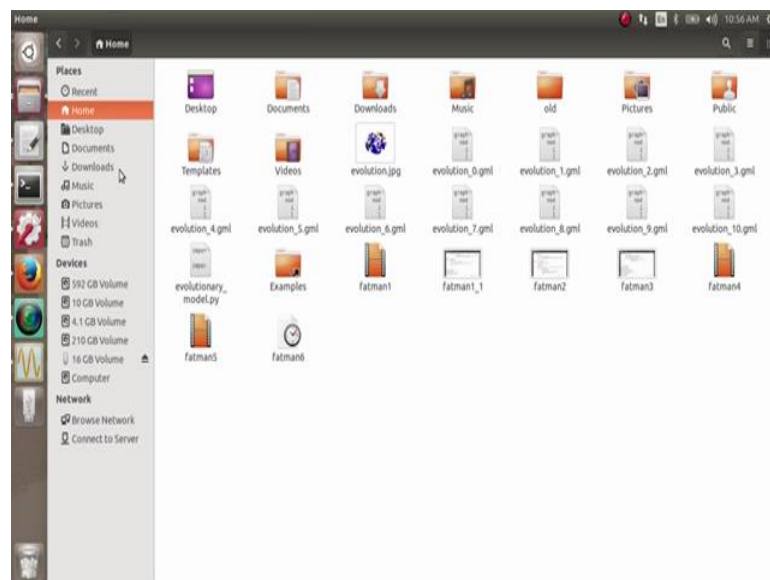
So, let us see how this graph gets generated here we let us (Refer Time: 04:06) python evolutionary_model.py and let see what happens here. So, we have seen as before this file evolution.jpg is going to change with time it will keep changing.

(Refer Slide Time: 04:16)



So, and if you see here let us just let us just close it. So, you see here that we have here these different gml files for me evolution_0.gml, evolution_1.gml, 2.gml.

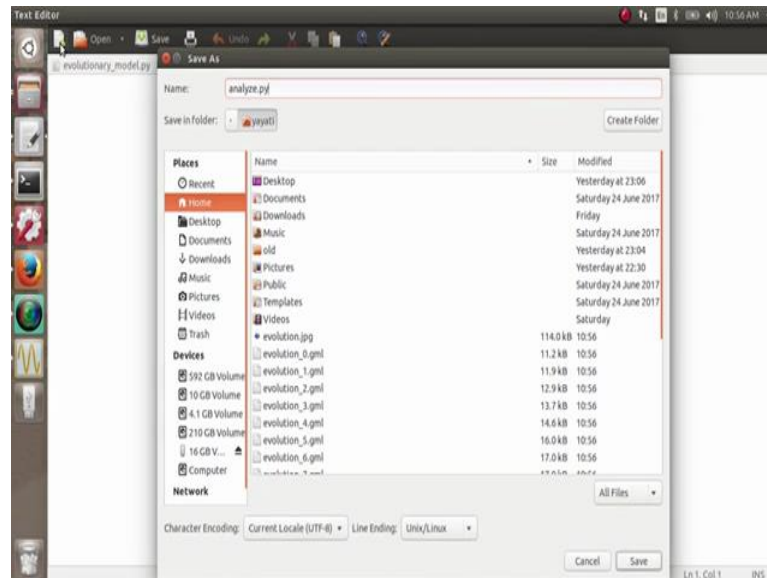
(Refer Slide Time: 04:24)



So, we have done it over some ten snapshots. So, you can do it over as many number of snapshots as you want. So, here we have 10 networks here I do not say ten networks it is the same network over 10 times stumps. So, it is my initial network there is some homophily some closure etcetera are happening on this network and then we have these different gml files coming up here.

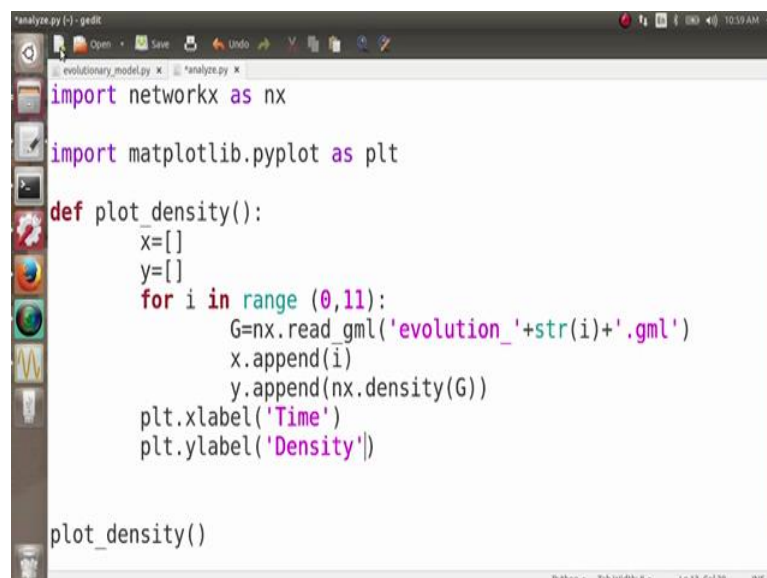
Now, I want to analyze these files why I want to analyze how this how the change in this graph is happening. So, to analyze this file we create a new file.

(Refer Slide Time: 05:10)



And let us name it analyze.py, and what I want to do in analyze.py see I want to analyze three things I want to see how the number of obese people in the network change then I want to see how the number of edges between these obese people change, and then I want to look at the change in density on this network. Let us keep it simple for the timing let just look at 2 things.

(Refer Slide Time: 05:44)

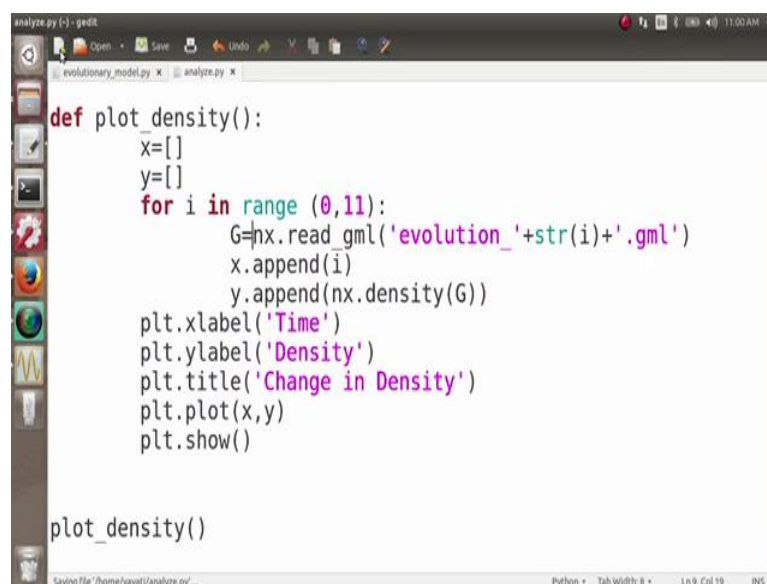


Let us look at how the number of obese people change and how the density of these networks changes. So, we need to import network x as nx and we also need to import matplotlib.pyplot as plt.

And next what we want to do is let us say our function is let us have our function plot density. So, what plot density does? It plots how the density in a network is changing. So, we defined here plot underscore density and what this function is going to do is. So, we need have. So, you know that for plotting we need an x axis, we need a y axis; x axis here is the iterations and y axis here is the density of these different networks. So, our x axis will be the iterations and our y axis will be the density.

So, what we do is we know that for i in range we have 10 snapshots. So, we take from range 0 to 11 we need a graph here. So, what is our graph G = nx.read_gml and what is the file name? File name we know is evolution under score and whatever is the value of i plus and we have here gml. When we read this file and then we want to look at the density of this network. So, what was in the x axis? x.append its very simple. So, we append i here and what do we appending y is the density of this network. So, we have a direct function for density let us call that nx.density(G). In y we append the density of this graph. So, we have an x axis, we have a y axis and at the end we plot both of these or we can actually put the titles for the plot plt.xlabel and let us see it is the time and then we can have plt.ylabel.

(Refer Slide Time: 08:13)

A screenshot of a text editor window titled 'analyze.py - gedit'. The editor contains a Python function 'def plot_density():' which initializes two empty lists 'x' and 'y'. It then enters a 'for' loop with 'i' ranging from 0 to 11. Inside the loop, it reads a GML file named 'evolution_' followed by the iteration number 'i' and '.gml' using 'G=nx.read_gml()'. It then appends 'i' to the 'x' list and 'nx.density(G)' to the 'y' list. After the loop, it sets the x-axis label to 'Time', the y-axis label to 'Density', and the plot title to 'Change in Density'. Finally, it calls 'plt.plot(x,y)' and 'plt.show()'. Below the function definition, the function is called 'plot_density()'. The status bar at the bottom indicates 'Python', 'Tab Width: 8', 'Ln 9, Col 19', and 'RHS'.

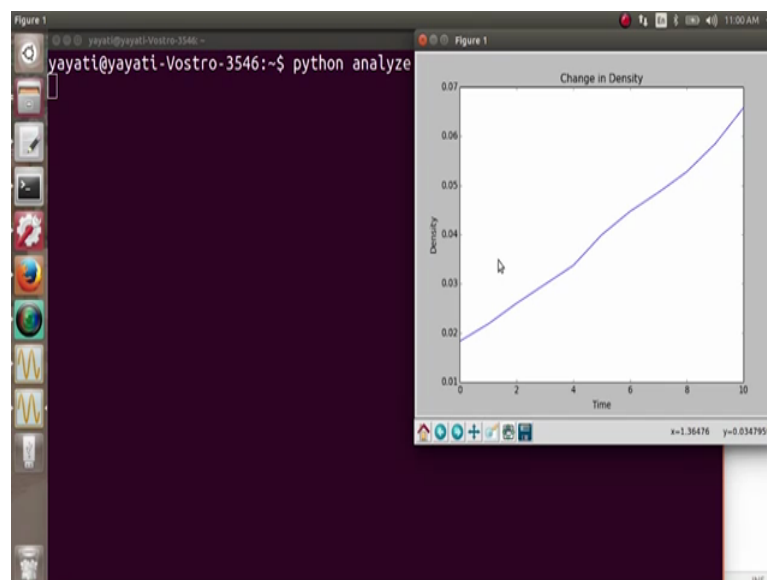
And let us have it as density, and then we have a title for this plot will it change in density; and then what we do is `plt.plot(x, y)` as we do, and then we do a `plt.show`.

So, this is simple right we have these 10 gml graphs we read each of these graph one by one look at their density and then we clued their density let us execute it for.

(Refer Slide Time: 08:47).

```
Terminal
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
^C^X^Z
[1]+  Stopped                  python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
File "evolutionary_model.py", line 121
    change_bmi(G):
    ^
SyntaxError: invalid syntax
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
Traceback (most recent call last):
  File "evolutionary_model.py", line 141, in <module>
    visualize(G)
  File "evolutionary_model.py", line 21, in visualize
    nx.draw(G, labels=labeldict, node_size=ns, node_color=color)
NameError: global name 'ns' is not defined
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ clear
```

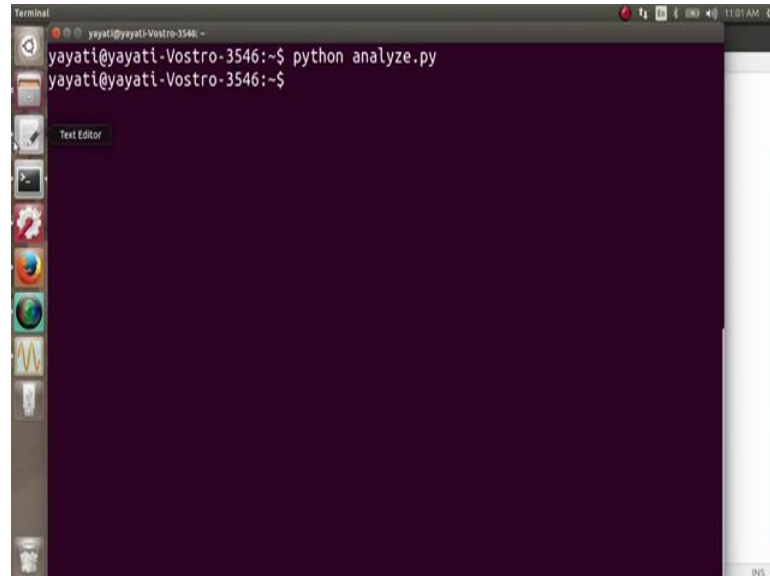
(Refer Slide Time: 08:50)



So, we have here python and we have analyze dot p y. So, you can see here how the density of this graph is changing. So, we trailed over ten iterations the density is increasing very fast, not we cannot say very fast densities increasing almost linearly this

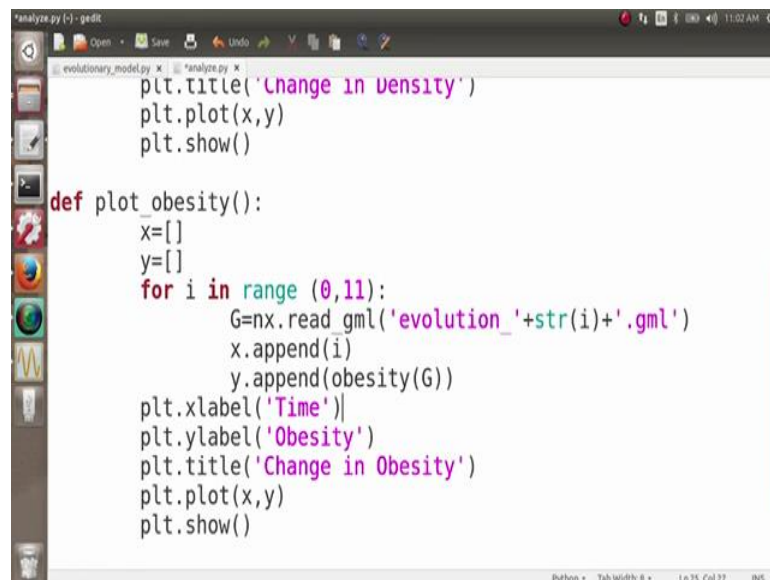
is about the density of this network more and more links are being formed because of homophily and because of all these three different kinds of closures.

(Refer Slide Time: 09:17)



So, you can actually see whatever you want to see across this evolution with the while loading this graphs and looking at them.

(Refer Slide Time: 09:35)

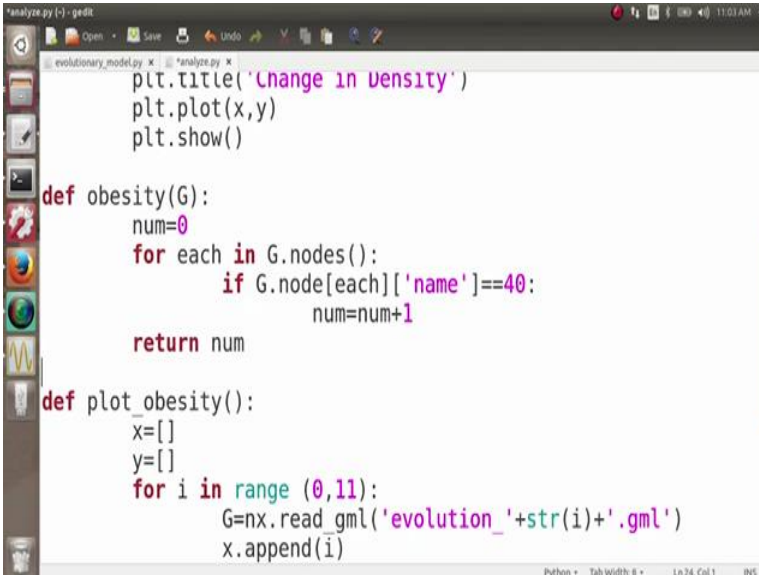


Let us say next I want to plot the obesity in this network and how do I plot the obesity by looking at the number of obese people in this network. So, we can write a similar code for this. So, we have defined plot underscore obesity and actually it is going to be almost

the same. So, let us just copy paste we use copy paste again and again in computer science. Define plot no (Refer Time: 10:10) computer science in coding define plot underscore obesity and we have this till here it is perfect.

Now, the problem is only here y dot append we have appended here the density of graph G, but what we want to append here is let us call our function obese and let us name this function as obesity. So, obesity is a function which will take as input your graph G and return you the number of obese people in this graph, and we just remove the density here by obesity, but yes we need to define this function obesity G.

(Refer Slide Time: 10:51)



```
analyze.py (-) - gedit
evolutionary_model.py x  analyze.py x
plt.title('Change in Density')
plt.plot(x,y)
plt.show()

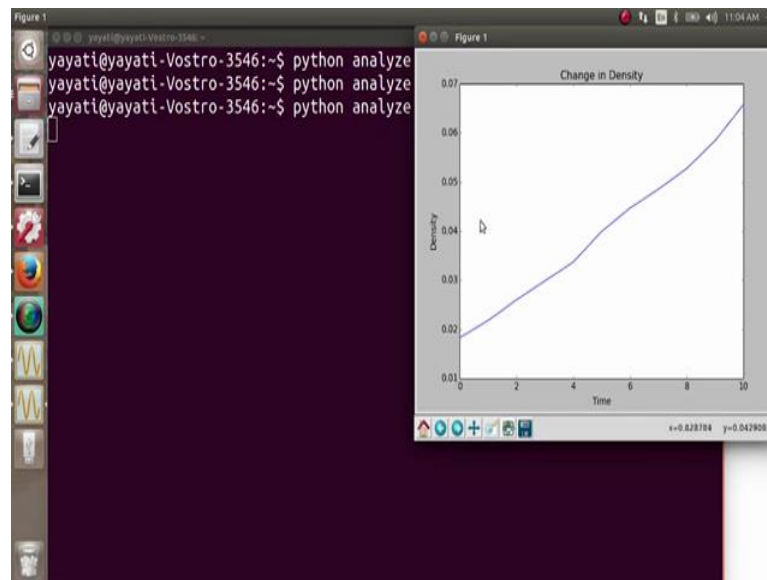
def obesity(G):
    num=0
    for each in G.nodes():
        if G.node[each]['name']==40:
            num=num+1
    return num

def plot_obesity():
    x=[]
    y=[]
    for i in range (0,11):
        G=nx.read_gml('evolution_'+str(i)+'.gml')
        x.append(i)
```

Lets quickly define this function obesity G and it is quite easy to find the number of obese people in this graph. So, let us say the number is 0 then for each in G.nodes we see whether the node is obese or not and if the node is obese, we increase num by 1.

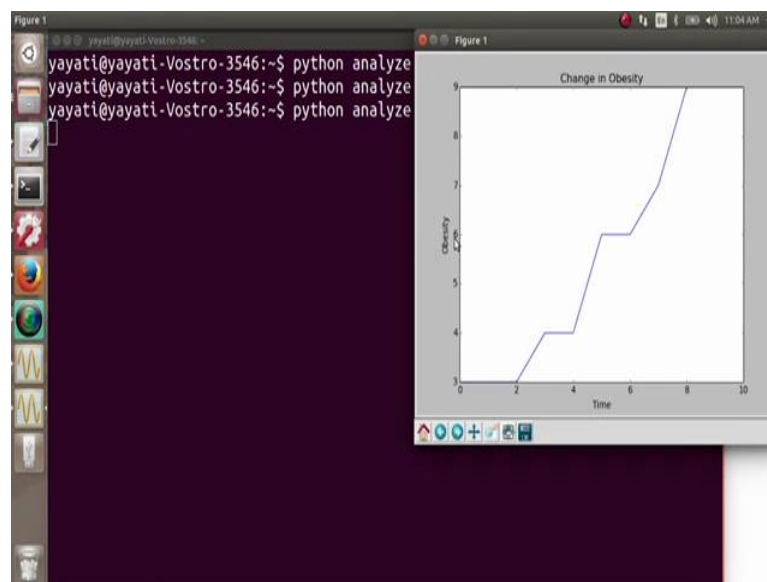
So, for each in G.nodes if G.node[each] and we look at its name right if its name equals to equals to 40 that is if its BMI is 40, we increase num by one and then we return num. So, this is how it goes.

(Refer Slide Time: 11:36)



Let us plot it and let us see it now. So, let us plot it and see. So, here we have python analyze.py. So, this first plot here is for the change in density, which we can see is increasing with time.

(Refer Slide Time: 11:51)



And now let us look at obesity, how the number of obese people in this network increase is. So, you can see here. So, it is an interesting curve you can see that it increases that then it remains constant for some time it then increases, then again remains constant for some time and so on.

We can actually see a lot of the change in a lot of interesting parameters in this graph you can actually see how the number of membership closures is changing in this graph, you can look at the number of triads in a lot of other things. So, that is a little bit of more complicated coding, but; obviously, fun to do if you are interested in any of that coding you can try it yourself and leave your questions on discussion forum will be always happy to help you.

So, I will leave you here with these very 2 simple observations. So, over this complete evolution and you can try your own coding and let us see if there is any problem. I hope you enjoyed this screen cast.