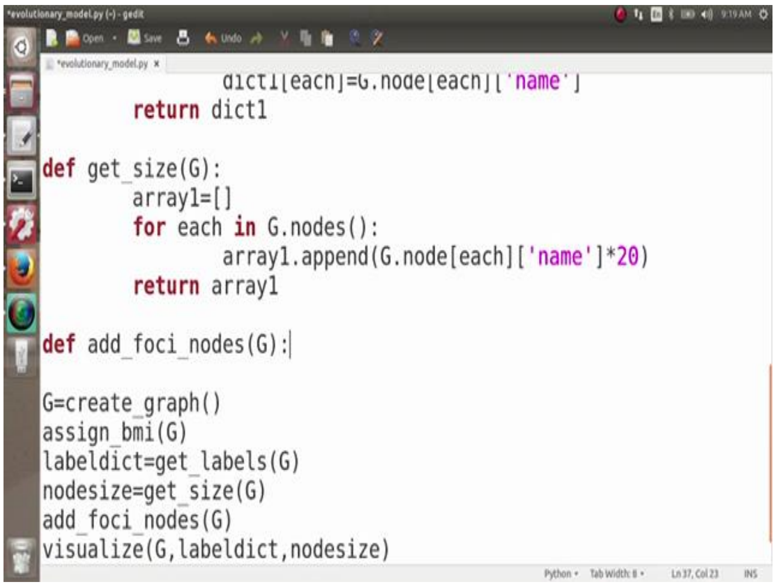
Social Networks
Prof. S. R. S. Iyengar
Department of Computer Science
Indian Institute of Technology, Ropar

Lecture - 48
Strong and Weak Relationships (Continued) and Homophily
Fatman Evolutionary Model - The Base Code (Adding Social Foci)

(Refer Slide Time: 00:05)

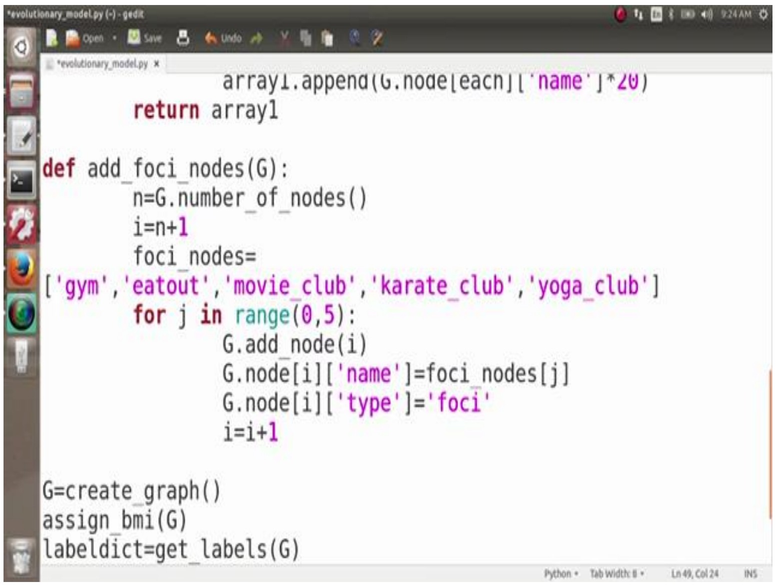


```
        dict1[each]=G.node[each]['name']
        return dict1

def get_size(G):
        array1=[]
        for each in G.nodes():
                array1.append(G.node[each]['name']*20)
        return array1

def add_foci_nodes(G):

G=create_graph()
assign_bmi(G)
labeldict=get_labels(G)
nodesize=get_size(G)
add_foci_nodes(G)
visualize(G,labeldict,nodesize)
```

Next, what we want to do is; we want to add foci_nodes here. So, we have now all the people node here, we want different-different foci like the gym, eat out, movies, yoga etcetera. So, let us see; how do we add foci_nodes here, so as before we are writing a function for everything. So, I create a new function here and I name the function as at foci_nodes and I pass my graph G here. So, in my graph G, I have to add foci_nodes, I define here; define add foci_nodes and I pass the function G here. Now, what I have to do is as discussed before we have 5 foci, so we have to create 5 new nodes in this graph. There, the simplest way to do is we know that the nodes in our graph till now, are numbered from 1 to 100.

So, we create 5 nodes; 101, 102, 103, 104 and 105, but I do not want to do it this way, what if my graph was having 200 nodes or 300 nodes. I might not know; what is the number of nodes in my graph and how should I proceed.

(Refer Slide Time: 01:18)



```
                array1.append(G.node[each]['name']*20)
        return array1

def add_foci_nodes(G):
        n=G.number_of_nodes()
        i=n+1
        foci_nodes=
['gym','eatout','movie_club','karate_club','yoga_club']
        for j in range(0,5):
                G.add_node(i)
                G.node[i]['name']=foci_nodes[j]
                G.node[i]['type']='foci'
                i=i+1

G=create_graph()
assign_bmi(G)
labeldict=get_labels(G)
```

So, what will we do it; we will get an iterator, let us say i or let us say a number n. So what is n? N is G.number_of_nodes. So, it returns as the number of nodes in the graph, so the number of nodes was 100. So, enhance this value 100 unit and now we want the new node to get started from 101, it means that we have to start adding nodes from the number n plus 1.

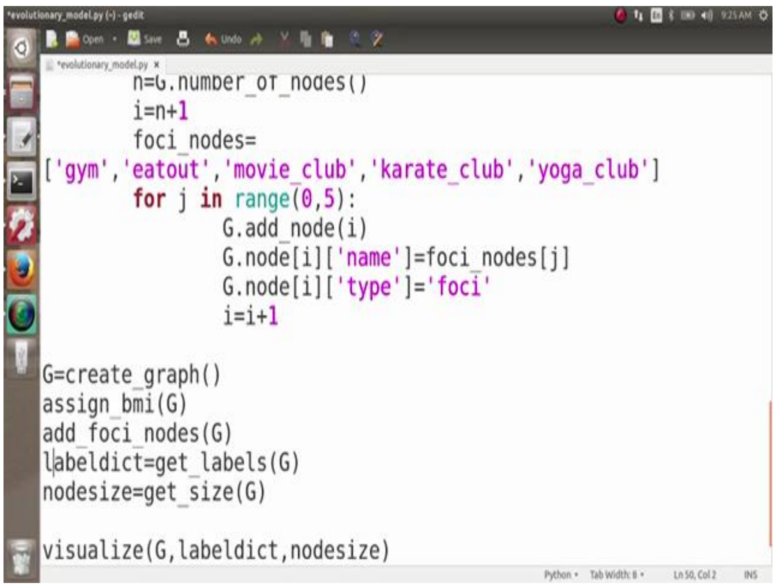
Now, you might want to note here; here the numbering of our nodes, it started from 1 to 100. If the numbering of the nodes was from 0 to 100; sorry was from 0 to 99, so still these were the 100 nodes, but the numbering and deduct 99. So, there we would have to start adding the nodes from the number n itself because n is 100 but provided now, we have the nodes from number 1 to 100. We have to start creating the new nodes from the number n + 1; so, I take the value of i = n + 1.

Next, I create an array for all of my foci_nodes; so I say foci_nodes equals to and this is an array and what all values this array has, it has a gym and then it has eat out and then it has a movie club and then it has a karate club and it also has a yoga club, so you have all these value in our foci_nodes. Now, I want to add 5 nodes in my graph; what we can do is four let us take a new variable; j in range 0 to 5, so it goes from 0 to 4. So, for j in range 0 to 5; what we have to do is, G dot add node and what should be the index of this node, index of this node should be i and then we increment i with i + 1. So, these 2 lines 0 to add node i and i = i + 1; add a new node in the graph.

Now, this new node as before should have 2 properties; the first property is the name which is the attribute of this node, which will tell you which of these 5 social foci it is. So, you see here now; so, to maintain a consistency in the person node also we have taken this attribute to be named. So when you look at a person, the main feature of a person is; his BMI which we depict as name and if you look at a social foci node its main feature is what so foci node it is; that is name. So, that is why we have used the attribute name in both the cases, so we need here 2 attributes; one is the name, and another is the type; so, that we can distinguish it from the person node.

So, what we do is; G.node[i] and what should be its name; its name should be chosen from our array. So, its name should be foci_nodes and we pass the value j here because j ranges from 0 to 4. So, it gets a name here and then we have to add a type; we do G.node[i] and then we have a type and this type should be equal to foci node, that it tell us that it is a foci node.

(Refer Slide Time: 05:57)

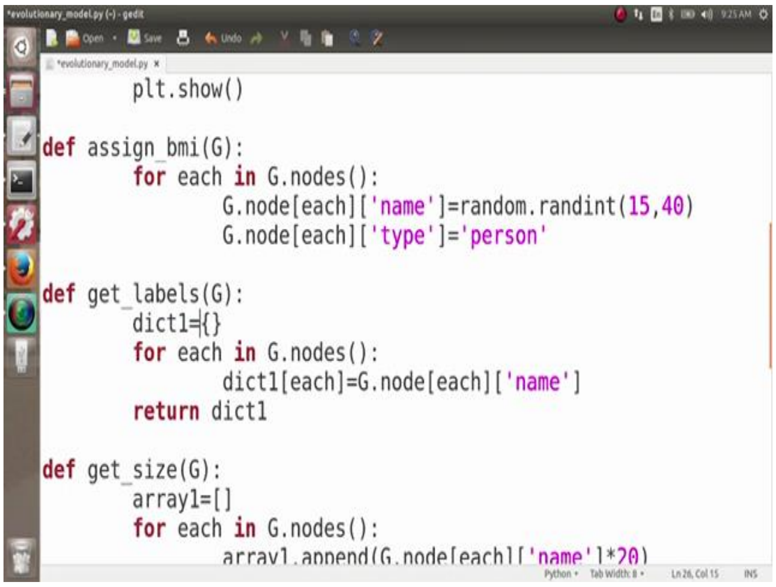


```
        n=G.number_of_nodes()
        i=n+1
        foci_nodes=
['gym','eatout','movie_club','karate_club','yoga_club']
        for j in range(0,5):
                G.add_node(i)
                G.node[i]['name']=foci_nodes[j]
                G.node[i]['type']='foci'
                i=i+1

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)

visualize(G,labeldict,nodesize)
```

So, we have added 5 foci_nodes to our graph G and then for visualizing it as you see that; so, one thing we have to take care where we have to keep this function call is should be after assigning BMI.

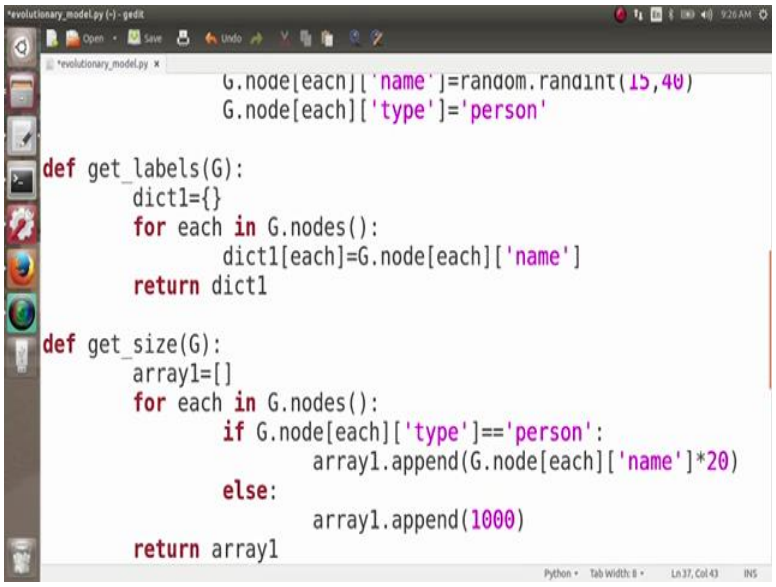
So, once we assign BMIs to the people then we add the foci_nodes, then next we get the labels and as we know labels for every node was its attribute name; so, we need not do any change here. Another was this function get under score size, so here we will have to do a little bit of modification. So, if you see at this function get labels; we do not need any modification here.

(Refer Slide Time: 06:31)



```
        plt.show()

def assign_bmi(G):
        for each in G.nodes():
                G.node[each]['name']=random.randint(15,40)
                G.node[each]['type']='person'

def get_labels(G):
        dict1={}
        for each in G.nodes():
                dict1[each]=G.node[each]['name']
        return dict1

def get_size(G):
        array1=[]
        for each in G.nodes():
                array1.append(G.node[each]['name']*20)
```

(Refer Slide Time: 06:38)



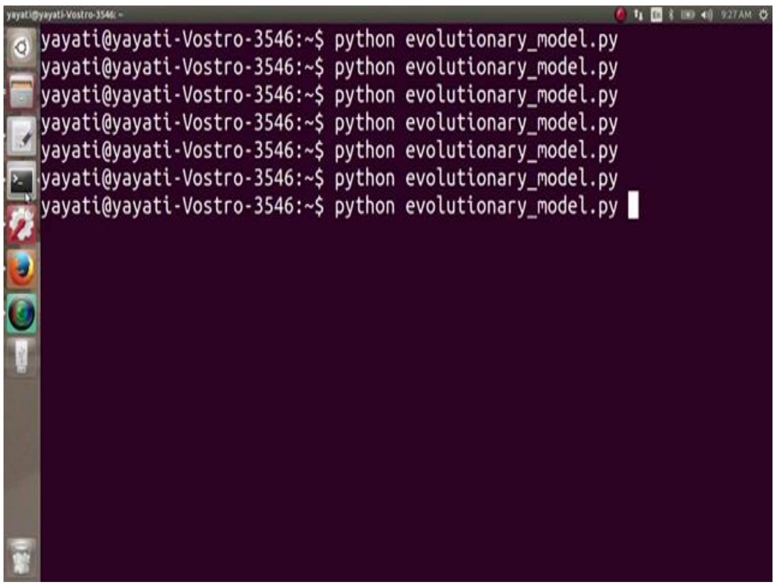
```
                G.node[each]['name']=random.randint(15,40)
                G.node[each]['type']='person'

def get_labels(G):
        dict1={}
        for each in G.nodes():
                dict1[each]=G.node[each]['name']
        return dict1

def get_size(G):
        array1=[]
        for each in G.nodes():
                if G.node[each]['type']=='person':
                        array1.append(G.node[each]['name']*20)
                else:
                        array1.append(1000)
        return array1
```

Because it is simply returning the name of every node, which is well defined for a person node as well as for a foci node, but if we look at these get size here. So, as we know that in our graph now, if we look at this name attribute for nodes. So, for the people node this name attribute returns the BMI, which is a number.

But, if we look at the social foci node; its name attribute is a string and multiplying the string by a 20 or using it in the array of size makes no sense. So, what we will do it; for each in G.nodes, if G dot each; if G.node[each][‘type’]. So, if its type == person; then what we do is, we go ahead and have this value, but else if its type is not of person; it is a foci node. Let us have some well-defined value fix here, so what we append here; array l.append and let us append a value 1000 here.

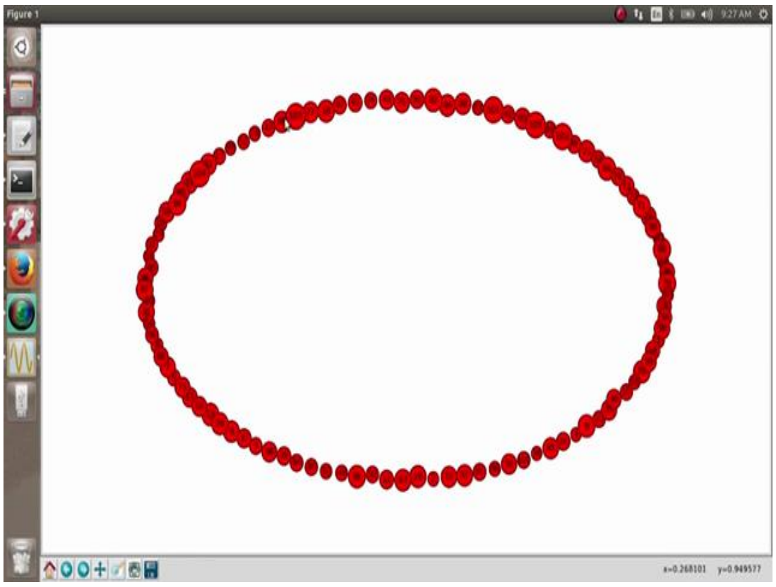
Now, if it is a person node; the size of the node is proportional to its BMI and if it is a foci node, we have a fixed size of that node which is 1000 and as before we then visualize this graph.

(Refer Slide Time: 08:02)



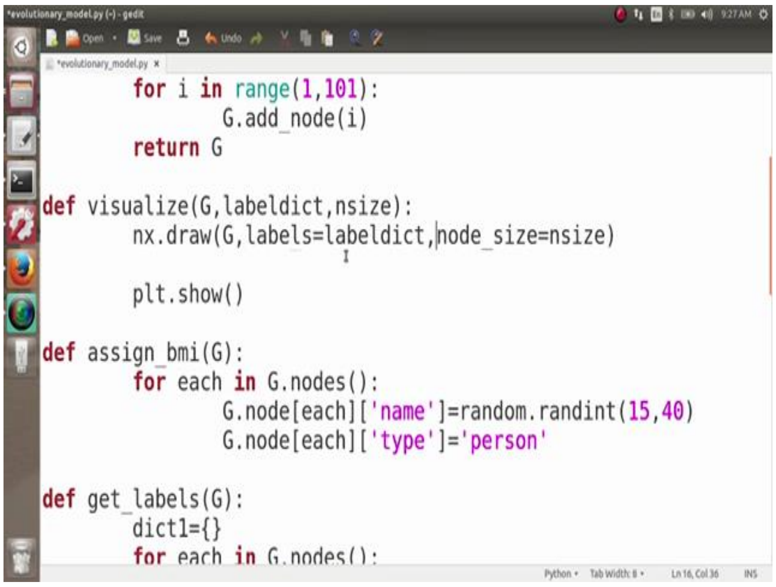
```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 08:05)



Let us execute this code, so we see this graph here; so, what is the problem here? We see all the nodes, but we do not see the names. So, what we had done previously is; we have to use this labeldict, so we were getting the label for every node; we have also passed this label and by visualizing the graph; we have actually removed that statement from here.

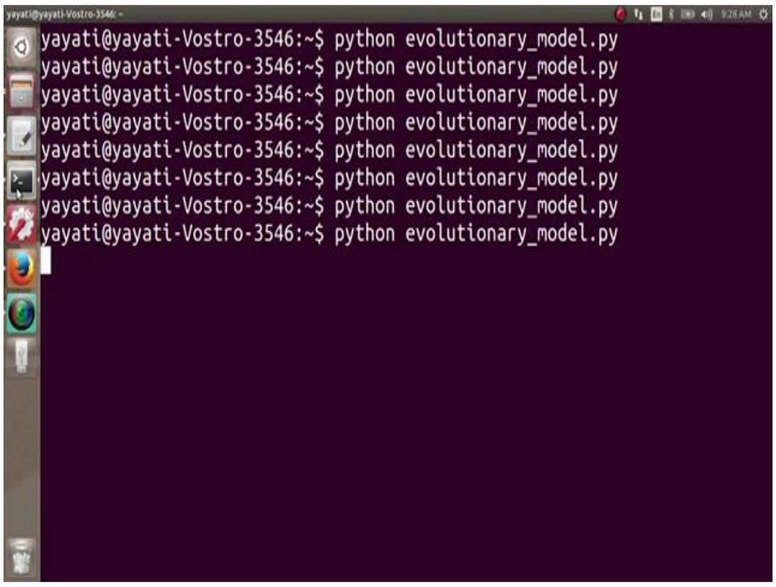
(Refer Slide Time: 08:28)



```
	for i in range(1,101):
		G.add_node(i)
	return G

def visualize(G,labeldict,nsize):
	nx.draw(G,labels=labeldict,node_size=nsize)

	plt.show()

def assign_bmi(G):
	for each in G.nodes():
		G.node[each]['name']=random.randint(15,40)
		G.node[each]['type']='person'

def get_labels(G):
	dict1={}
	for each in G.nodes():
```

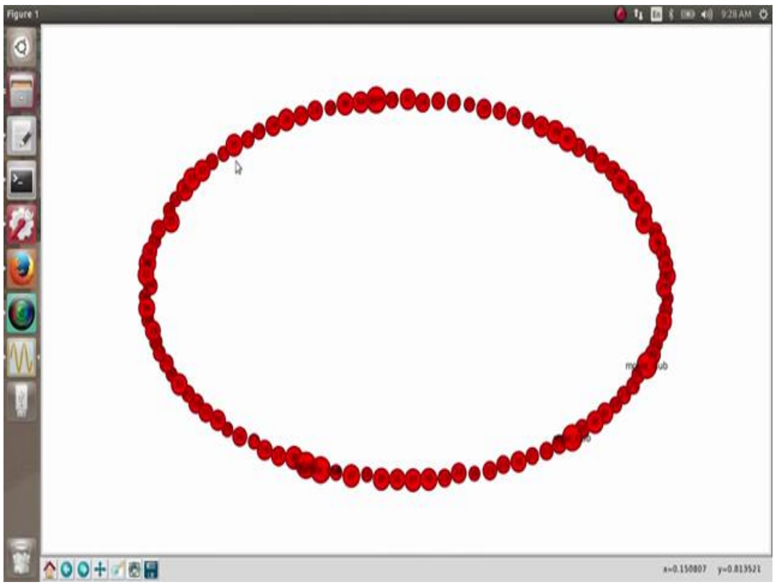
So, what we want is; we want the labels and the labels should be equal to labeldict so that we can see the labels for all the nodes, we want the labels; so, we keep = labeldict.

(Refer Slide Time: 08:47)



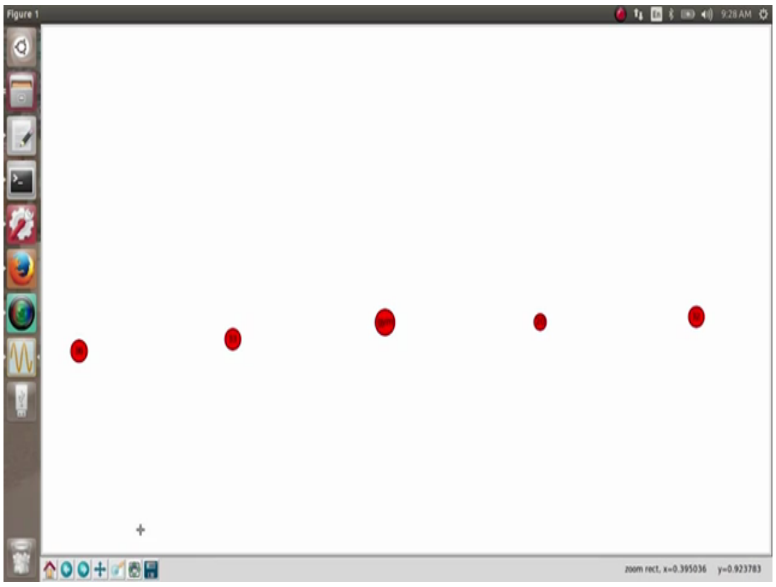
```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 08:52)

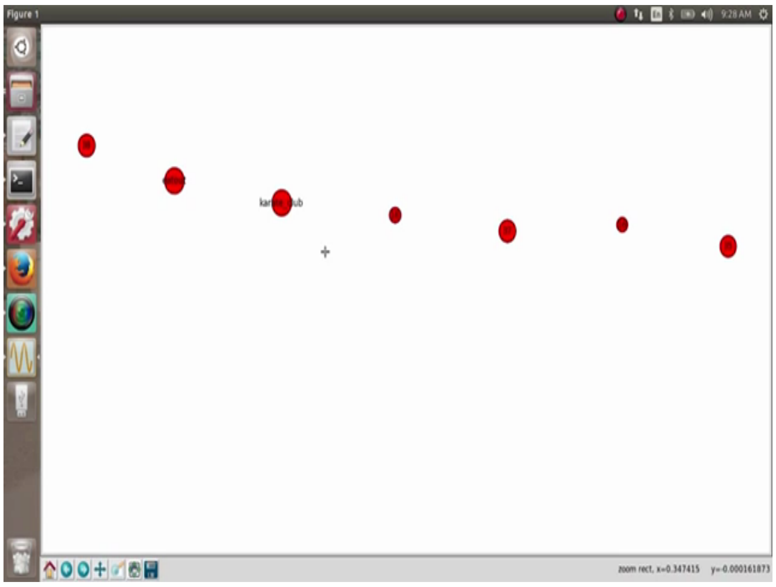


We go back then execute this code, now if we look at this graph; you can see that we have all different nodes labeled with their BMI, with their size, but here we have 5 social foci_nodes as well.

(Refer Slide Time: 09:02)

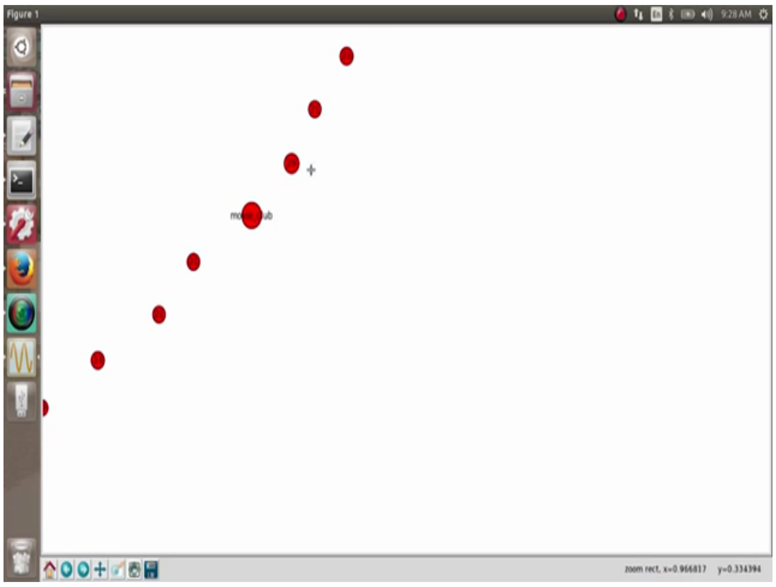


(Refer Slide Time: 09:07)



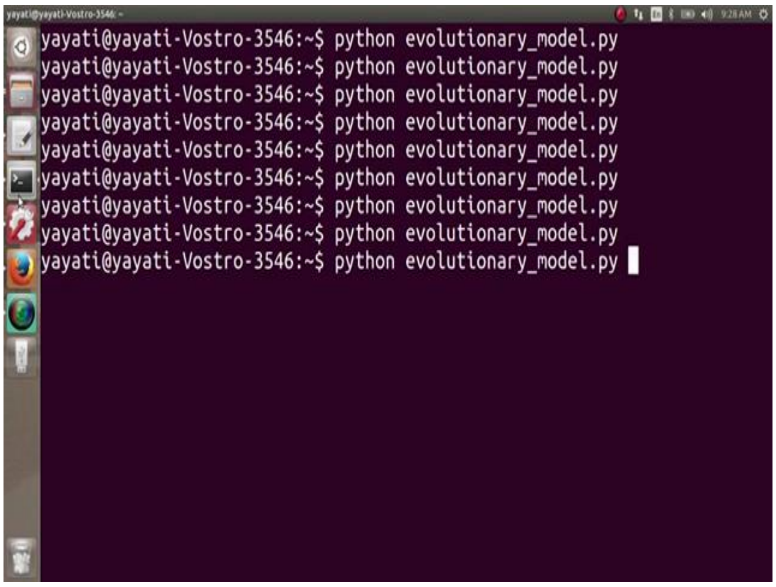
So, here we have gym node and then here we have eat out and we have karate club and here we have movie club.

(Refer Slide Time: 09:11)



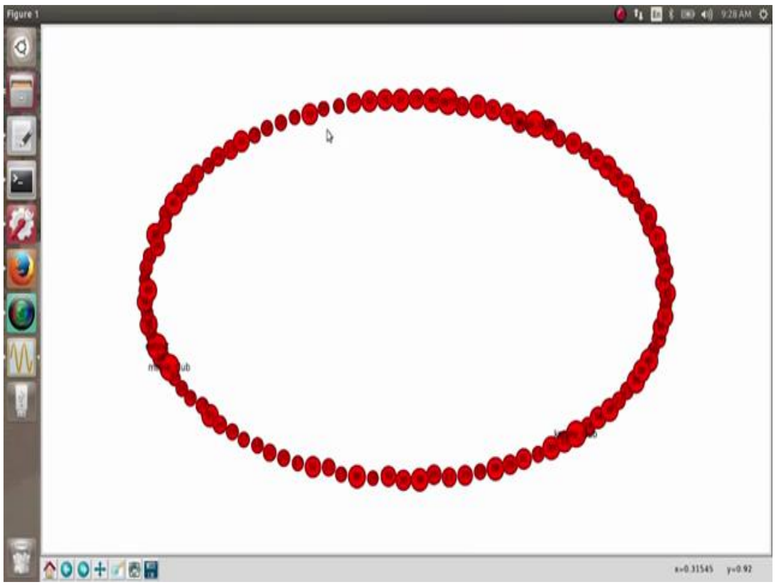
So, we have all the social foci_nodes also here in this graph; so, we have 100 nodes for people and 5 nodes for social foci. What is more interesting to do next, so something which is coming next is all the way more interesting; what do I want?

(Refer Slide Time: 09:37)



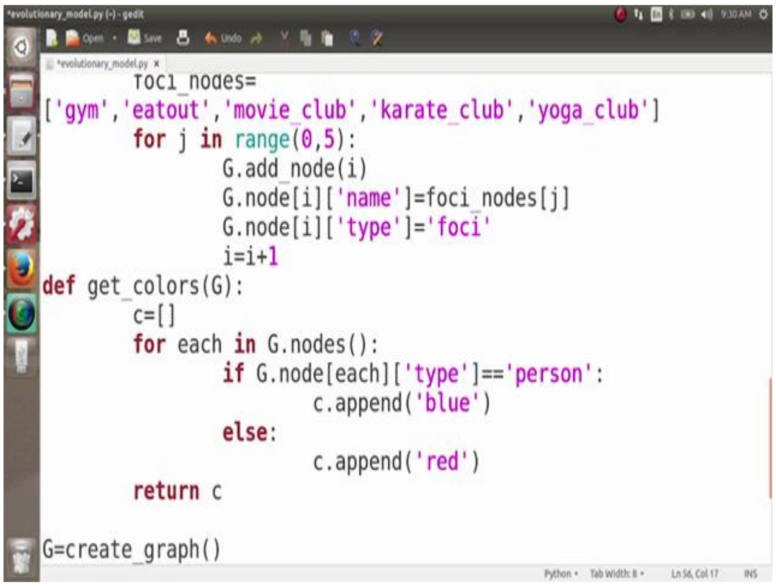
```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 09:39)



So you have seen that when we looked at this graph here; everything comes everything is in a red color. So, it makes difficult for me to identify the social foci_nodes here; I have to search for them. Next, what we will do is; we write a piece of code, so that the social foci_nodes appear in red color, but all the other nodes that is the people nodes; they appear in blue color and this is very simple. So, like we did for the labels and for the node size, we also do a similar thing.

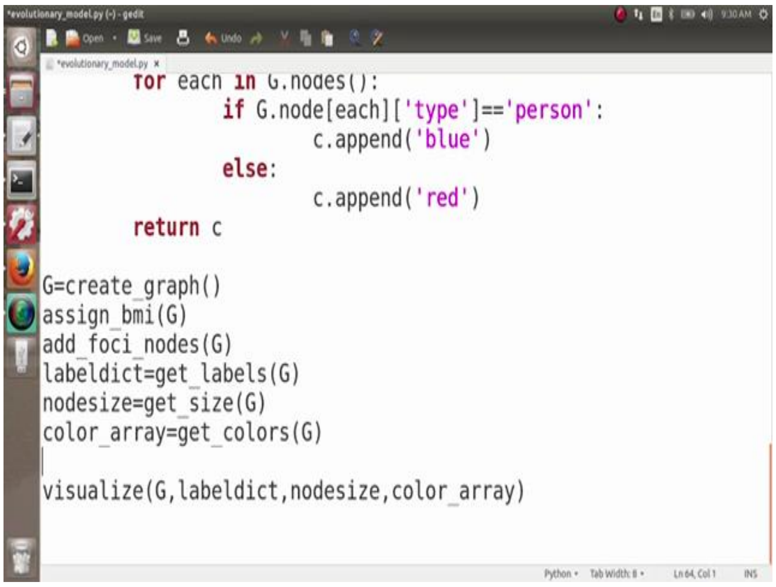
(Refer Slide Time: 10:06)



```
        foci_nodes=
['gym','eatout','movie_club','karate_club','yoga_club']
        for j in range(0,5):
                G.add_node(i)
                G.node[i]['name']=foci_nodes[j]
                G.node[i]['type']='foci'
                i=i+1
def get_colors(G):
        c=[]
        for each in G.nodes():
                if G.node[each]['type']=='person':
                        c.append('blue')
                else:
                        c.append('red')
        return c

G=create_graph()
```

Here, we make an array; color array and get this array from the function get colors G and then we define this function get colors G here; define get colors G and what do we do here is, we have an array let us say c, then for each in G.nodes; if G.node each, type, so if its type == person, What do we do? We append a blue in this array else we append a red in this array and then we return this array.

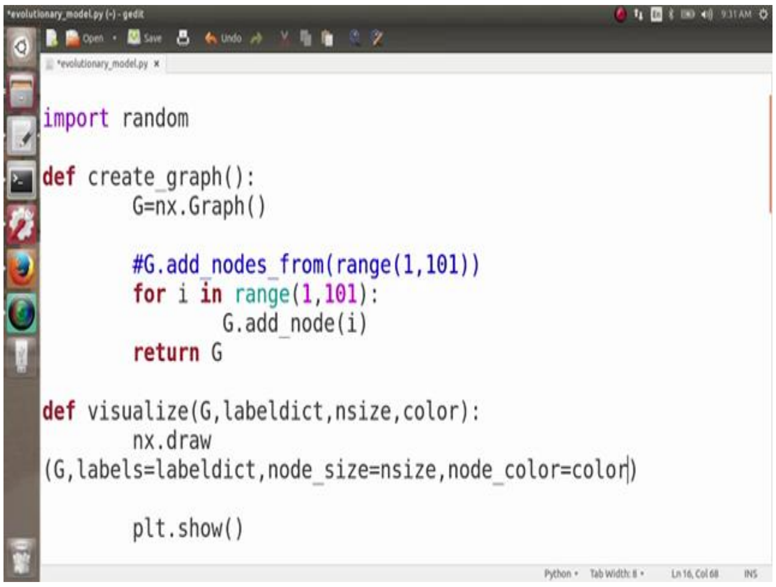
(Refer Slide Time: 11:05)



```
        for each in G.nodes():
                if G.node[each]['type']=='person':
                        c.append('blue')
                else:
                        c.append('red')
        return c

G=create_graph()
assign_bmi(G)
add_foci_nodes(G)
labeldict=get_labels(G)
nodesize=get_size(G)
color_array=get_colors(G)

visualize(G,labeldict,nodesize,color_array)
```

So, we get here color array, so this color array has a blue color if a node is a person and it has a red color, if the node is a social foci and what do we do here is we pass this color array here and the visualize function.

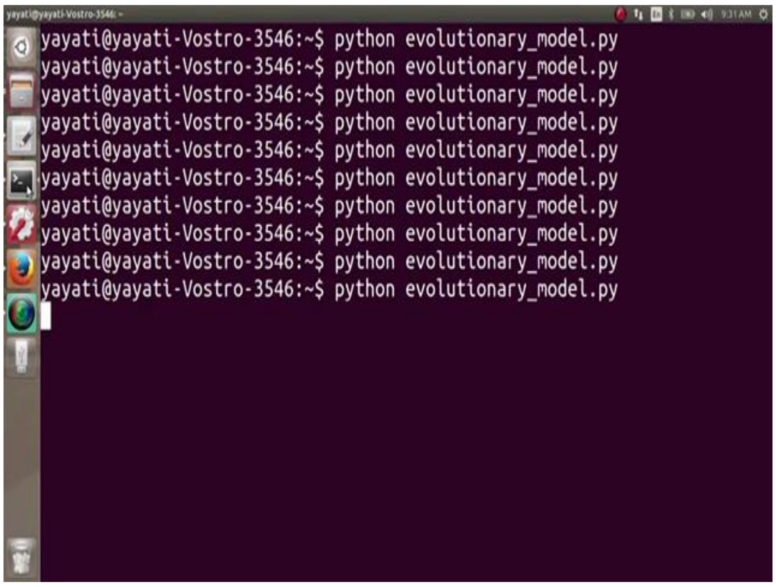
(Refer Slide Time: 11:27)



```
import random

def create_graph():
	G=nx.Graph()

	#G.add_nodes_from(range(1,101))
	for i in range(1,101):
		G.add_node(i)
	return G

def visualize(G,labeldict,nsize,color):
	nx.draw
(G,labels=labeldict,node_size=nsize,node_color=color)

	plt.show()
```

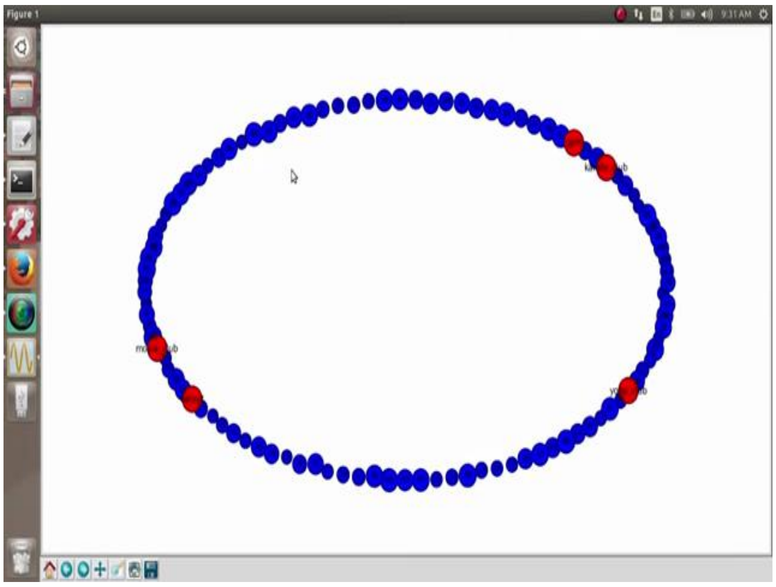
We go back to the definition of the visualize function; we have an extra array color here let us say color. So, what we do here is we add one more parameter which is node underscore color and from where every node takes its color is our array color, so we are done; go back.

(Refer Slide Time: 11:45)



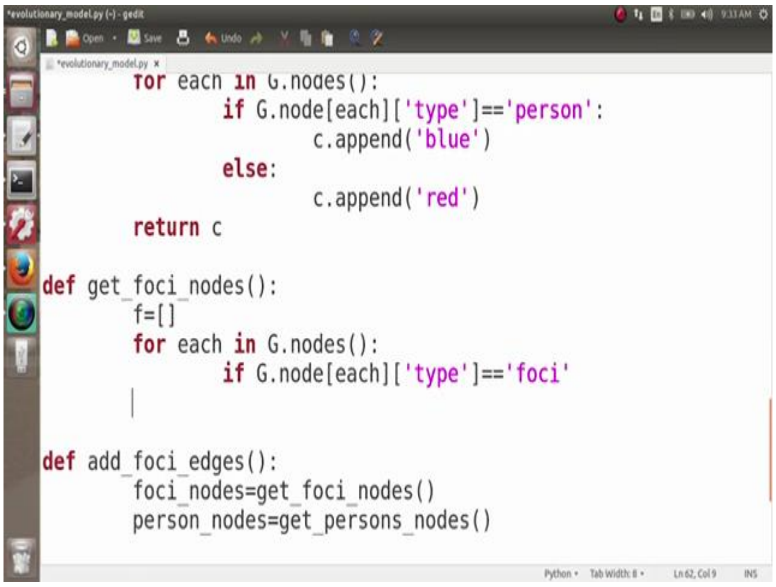
```
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 11:51)



Execute our code and you can see now; every social foci node comes in a red color and all the people node, it comes in a blue color; so far, so good. Now, we have a city and the city has 100 people and the city has 5 social foci. Next, what we want to do is in the beginning every person should be a part of exactly one social foci. So, now we must add edges between the people and the social foci.

(Refer Slide Time: 12:26)

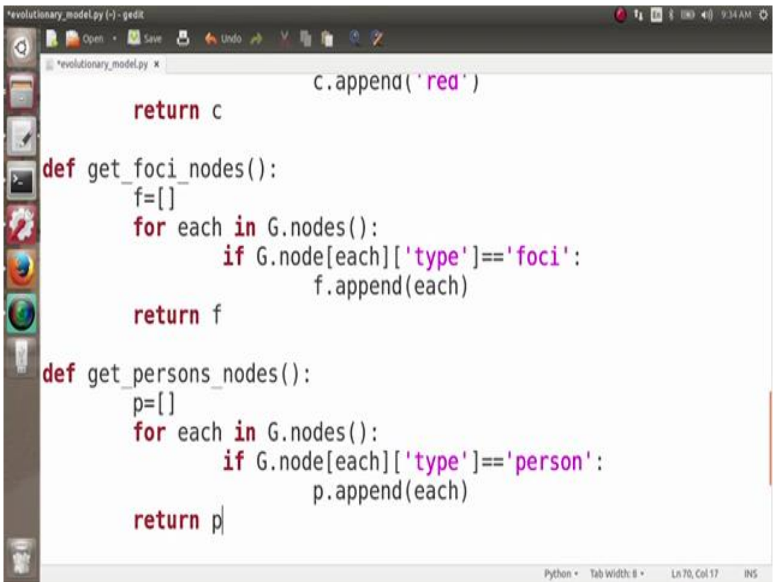


```
    for each in G.nodes():
        if G.node[each]['type']=='person':
            c.append('blue')
        else:
            c.append('red')
    return c

def get_foci_nodes():
    f=[]
    for each in G.nodes():
        if G.node[each]['type']=='foci'


def add_foci_edges():
    foci_nodes=get_foci_nodes()
    person_nodes=get_persons_nodes()
```

So, again we write a function for that and let the name of the function be add_foci_edges. So, we want to add some edges from every person to a social foci; so, we have here define add; foci edges and now what we want to do is, we chose one person one by one and from this person, we add an edge to every foci node. So, for that first of all we need all the nodes which are the foci_nodes at one place, so that we know to which nodes we have to make connections to. Although, we know that the foci_nodes are from 101 to 105, but that might not be the case with every graph given to you. We have made this graph this way, so let us try to get an array which consists of all the nodes, which are the foci_nodes.

So, we make an array foci_nodes equals to and then and let us make an array; people node. Rather, a better approach is to use functions for both of these; so, for foci_nodes equals to get. So, the array foci_nodes = get_foci_nodes and the array person_nodes = get_persons_nodes and then we quickly define both of this functions here; define get_foci_nodes. So, what do we do is; we take an array f and then few of we want to make the foci_nodes for each in G.nodes; if G.node each and if its type is equal to equal to what? If its type == ‘foci’, then what do we do?

(Refer Slide Time: 14:42)

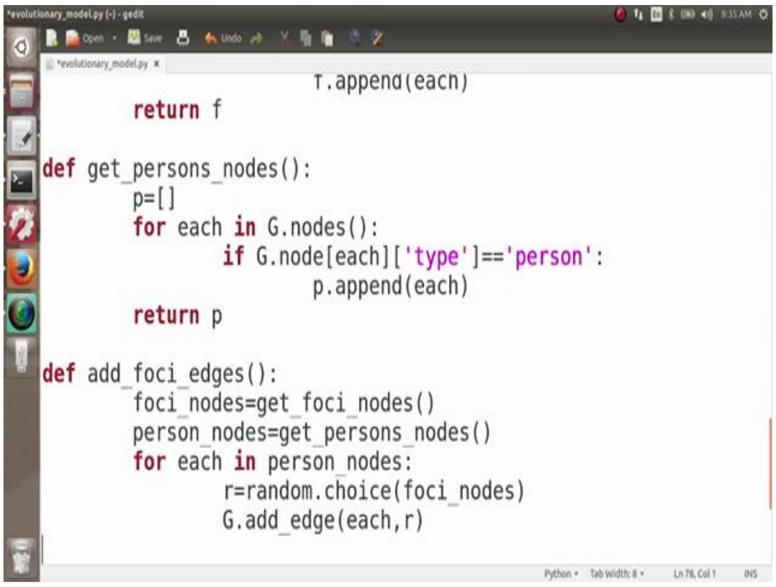


```
                c.append('red')
        return c

def get_foci_nodes():
        f=[]
        for each in G.nodes():
                if G.node[each]['type']=='foci':
                        f.append(each)
        return f

def get_persons_nodes():
        p=[]
        for each in G.nodes():
                if G.node[each]['type']=='person':
                        p.append(each)
        return p
```

Then what do we do is; f.append; f.append each and then you return f. So, this particular function will return you an array or foci_nodes and we do exactly the same thing for the person’s nodes. So, instead of writing the code again; let us just copy paste this code and make the changes. So, get_foci_nodes and here we have get persons nodes and what we do not want to do here is let this array be p. So, for each in G.nodes, if

G.node[each][‘type’] equals to equals to and we have here person, then we append this in array p and then we return p.

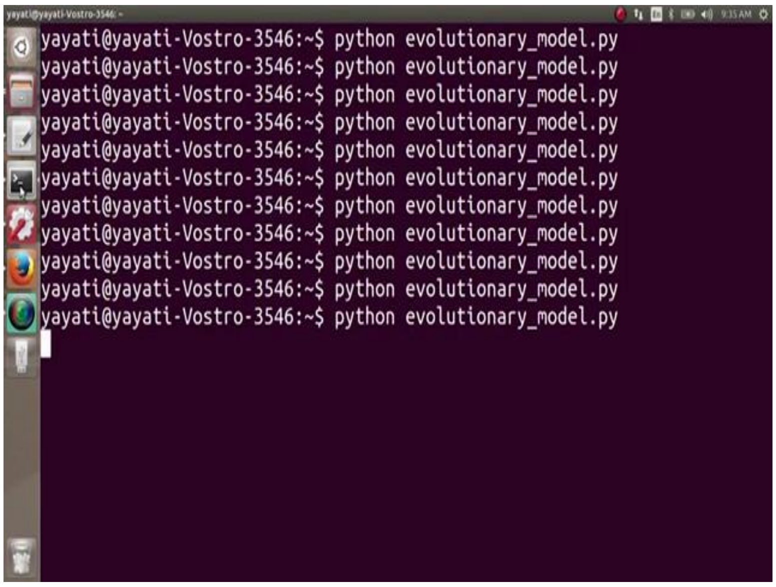
(Refer Slide Time: 15:37)



```
                        f.append(each)
        return f

def get_persons_nodes():
        p=[]
        for each in G.nodes():
                if G.node[each]['type']=='person':
                        p.append(each)
        return p

def add_foci_edges():
        foci_nodes=get_foci_nodes()
        person_nodes=get_persons_nodes()
        for each in person_nodes:
                r=random.choice(foci_nodes)
                G.add_edge(each,r)
```

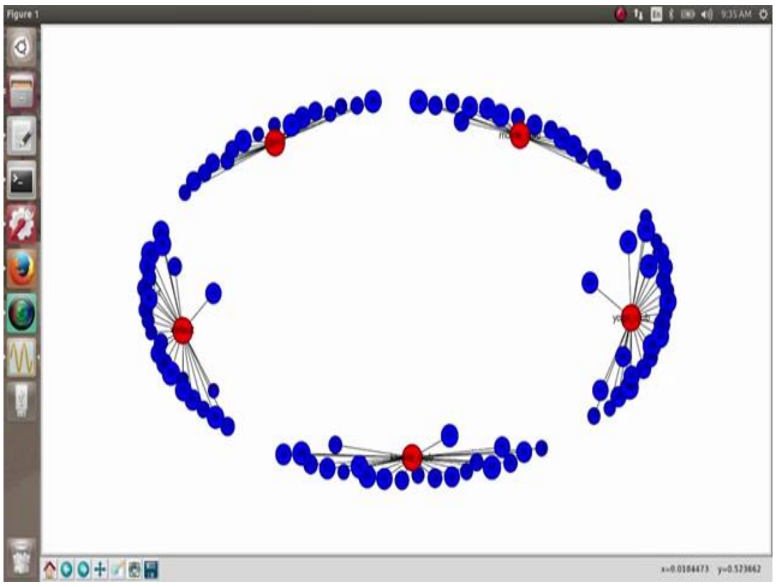
So, here we have got an array one is for the foci_nodes and another is for the person’s node. Now, what we want to do is add an edge, so what do we do for adding an edge is for each in and we want to add an edge from every person nodes. So, for each in person nodes, what we do is, we choose a random node from the foci_nodes. So, r = random.choice, so random.choice is used to get a random value from an array; r equals to random dot choice from foci_nodes and then what we do is G.add_edge and we add edge from each to r. So, for every node in the person’s node, we choose a node randomly from the foci_nodes and add an edge between the 2.

(Refer Slide Time: 16:26)



```
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
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
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
yayati@yayati-Vostro-3546:~$ python evolutionary_model.py
```

(Refer Slide Time: 16:31)



So, let us go back and look at the output; so, you can see here what we have is; every person has chosen here one foci_nodes. So, these people here are part of gym, these people are part of movie club, these people are part of yoga club, karate club and eat out.