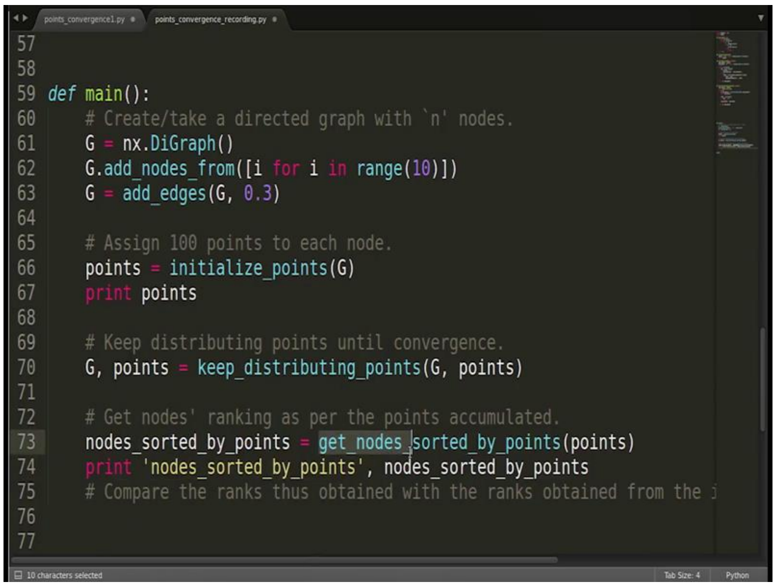
Social Networks
Prof. S. R. S. Iyengar
Prof. Anamika Chhabra
Department of Computer Science
Indian Institute of Technology, Ropar
Link Analysis

Lecture - 82
Implementing Page Rank Using Points Distribution Method – 3

(Refer Slide Time: 00:07)

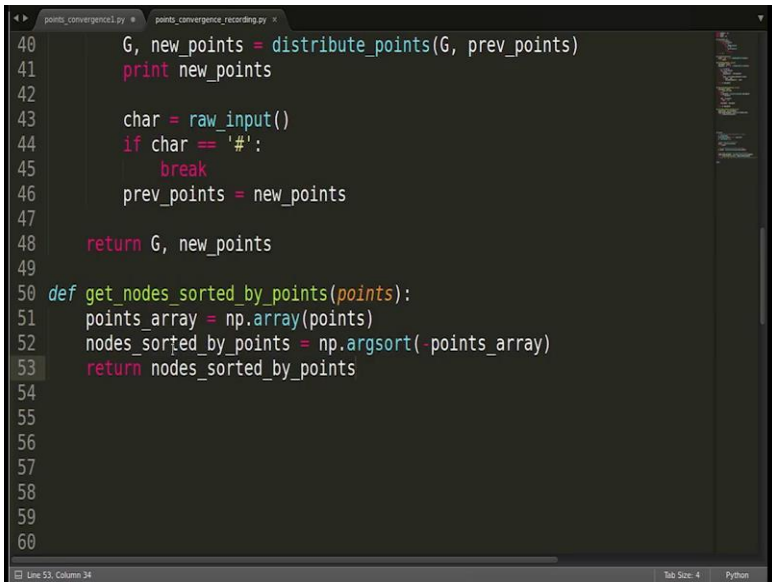


```
def main():
    # Create/take a directed graph with `n' nodes.
    G = nx.DiGraph()
    G.add_nodes_from([i for i in range(10)])
    G = add_edges(G, 0.3)

    # Assign 100 points to each node.
    points = initialize_points(G)
    print points

    # Keep distributing points until convergence.
    G, points = keep_distributing_points(G, points)

    # Get nodes' ranking as per the points accumulated.
    nodes_sorted_by_points = get_nodes_sorted_by_points(points)
    print 'nodes_sorted_by_points', nodes_sorted_by_points
    # Compare the ranks thus obtained with the ranks obtained from the i
```

In the previous video, we saw the conversions taking place when it comes to the points that the nodes are able gather in every iteration. So, we saw that the points are converging. Now, the next step is how we can rank the nodes based on the points that they have gathered. Now, let us create a function to sort the nodes based on points. So, let me create get nodes sorted by points, I am going to pass the points list and it should return me the list of nodes. I am sorry I am sorry ok.

So, I will write nodes_sorted_by_points = get_nodes_sorted_by_points which is a function which I am going to create. And, then let me print the nodes that we will be getting. So, we will just be printing the nodes that we are getting, I should print the initial points as well. So, that we know how the points are changing. So, I am printing the initial points here ok, getting back to the function that you have to create get_nodes_sorted_by_points. Let us create this function here. Sorry ok.

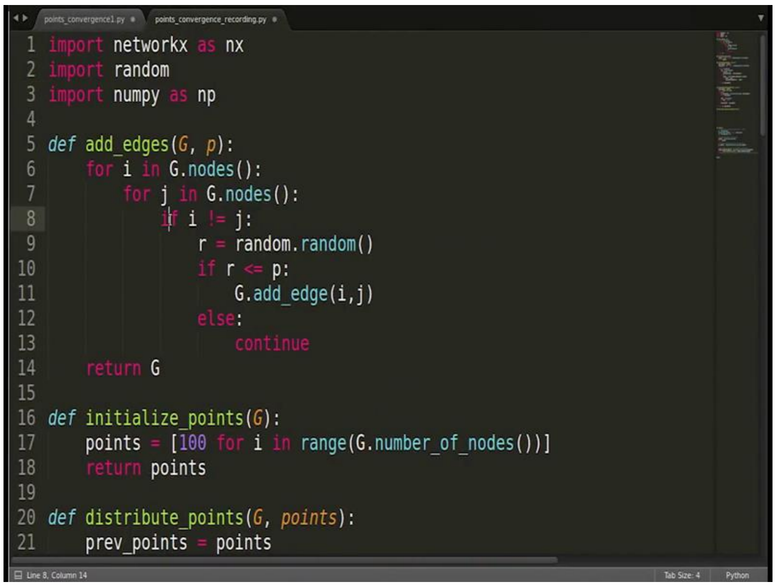
(Refer Slide Time: 01:27)


```
        G, new_points = distribute_points(G, prev_points)
        print new_points

        char = raw_input()
        if char == '#':
            break
        prev_points = new_points

    return G, new_points

def get_nodes_sorted_by_points(points):
    points_array = np.array(points)
    nodes_sorted_by_points = np.argsort(-points_array)
    return nodes_sorted_by_points
```

Now, what should this function do? This function has as an input a list which is points which is a list having the points for every node. And, what should this function do? This function should return a list of the nodes sorted by their points. Now, what are the nodes? How are the nodes labeled? The nodes are basically from as you can see here from 0 to 9 right; that is how the nodes are numbered.

So, we can just simply get the index of the nodes from this points list. So, basically what we have to do is, we have to sort the index of the values which are there in the points right. We can use again several methods for that, one method which might be very quick is we can make use of the package numpy. So, numpy has function argsort which basically returns the index, the indices sorted by the values of a list. So, we are going to use that function argsort that function is there in numpy.

(Refer Slide Time: 02:49)

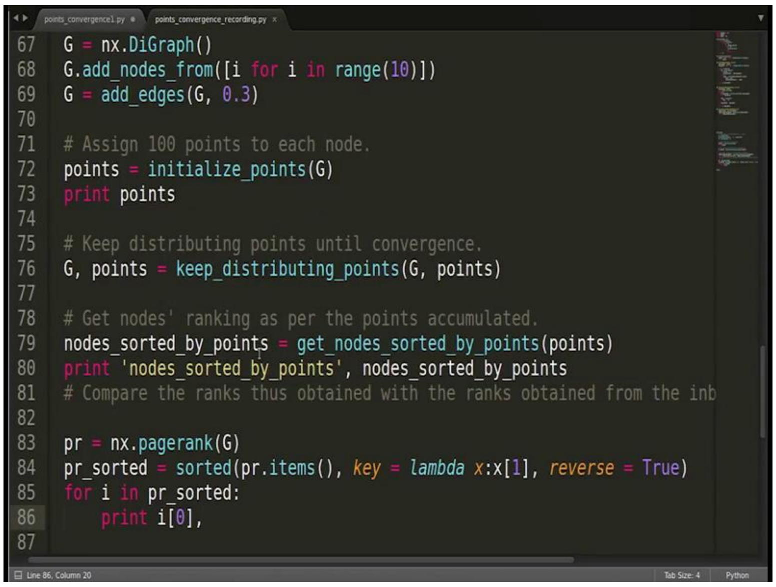

```
import networkx as nx
import random
import numpy as np

def add_edges(G, p):
    for i in G.nodes():
        for j in G.nodes():
            if i != j:
                r = random.random()
                if r <= p:
                    G.add_edge(i,j)
                else:
                    continue
    return G

def initialize_points(G):
    points = [100 for i in range(G.number_of_nodes())]
    return points

def distribute_points(G, points):
    prev_points = points
```

So, let me import that package. So, I will write import numpy as np ok. So, the input to wherever we yeah here so, the input to that function is basically an array which is a numpy array and here as an input we have a points which is a list. So, we have to convert that list into a numpy array. Now, how we do that? So, I will write points array is equal to np numpy.array. So, so this array is the function which converts a list into an numpy array ok. Now, points array is a numpy array and we can apply the function argsort on it. And, what is that function do?

It returns the indices sorted by the values of the list. So, write np numpy.argsort points array ok. What is it return? It returns the indices. So, indices basically are the nodes in our case. So, maybe I can write nodes_sorted_by_points equal to this thing ok. Now, one thing that has to be noted is by default argsort sorts in the ascending order right. And, which order do you want? We want descending order that is the node which has high points should be the first to be ranked ok; so, simple thing that we can do here just put a minus here.

By putting a minus all the values are convert converted to negative and the ordering that we will be getting will be negative. So, as simple as that we can work with that simple method. So, this nodes_sorted_by_points is going to have the nodes sorted by of course, the points in descending order right. So, the first value will be the node which is having highest ranking and hence the highest page rank ok. Let us return this nodes_sorted_by_points, this should work.

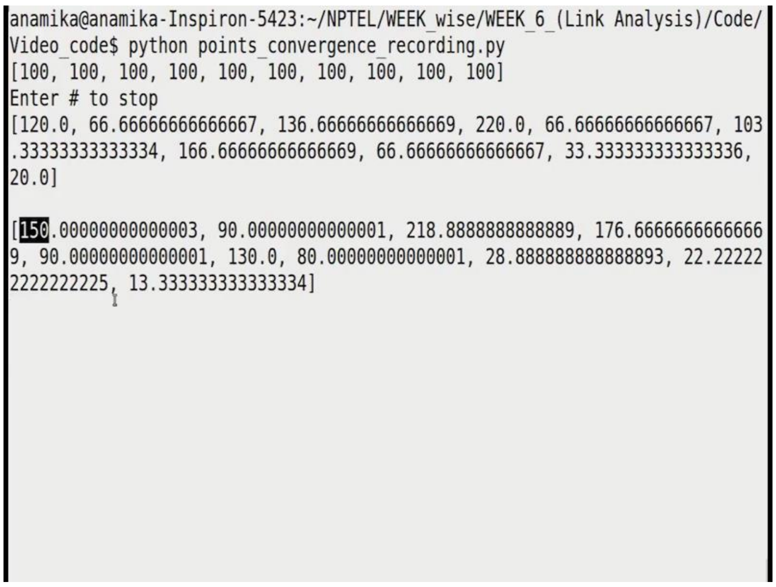
(Refer Slide Time: 05:07)



```python
G = nx.DiGraph()
G.add_nodes_from([i for i in range(10)])
G = add_edges(G, 0.3)

# Assign 100 points to each node.
points = initialize_points(G)
print points

# Keep distributing points until convergence.
G, points = keep_distributing_points(G, points)

# Get nodes' ranking as per the points accumulated.
nodes_sorted_by_points = get_nodes_sorted_by_points(points)
print 'nodes_sorted_by_points', nodes_sorted_by_points
# Compare the ranks thus obtained with the ranks obtained from the inb

pr = nx.pagerank(G)
pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
for i in pr_sorted:
    print i[0],
```

So, let us go back to main and let us see is anything pending. So, we had already called this function and we had already printed the values. So, let us see whether this function works fine or not ok.

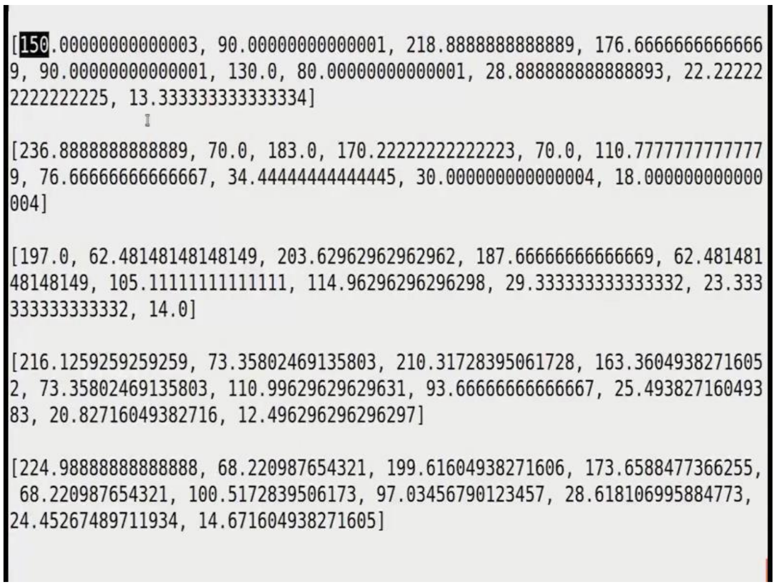
(Refer Slide Time: 05:18)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[120.0, 66.66666666666667, 136.66666666666669, 220.0, 66.66666666666667, 103
.33333333333334, 166.66666666666669, 66.66666666666667, 33.333333333333336,
20.0]

[150.00000000000003, 90.00000000000001, 218.8888888888889, 176.6666666666666
9, 90.00000000000001, 130.0, 80.00000000000001, 28.888888888888893, 22.22222
2222222225, 13.333333333333334]
```

So, let us call this function, let us let us see how the code works. So, let me execute this ok. So, these are the initial points I am sorry these are the initial points 100 each, as we assigned initially. And then we started the iterations, after the first iteration as you can see the first node which was having 100 points is now having 120 points and second node the points for second node have reduced and so on. So, that that keeps changing; this is the scenario after first iteration. I press enter and this is this scenario after second iteration. So, the points have 150 for the first node, I keep on pressing enter.

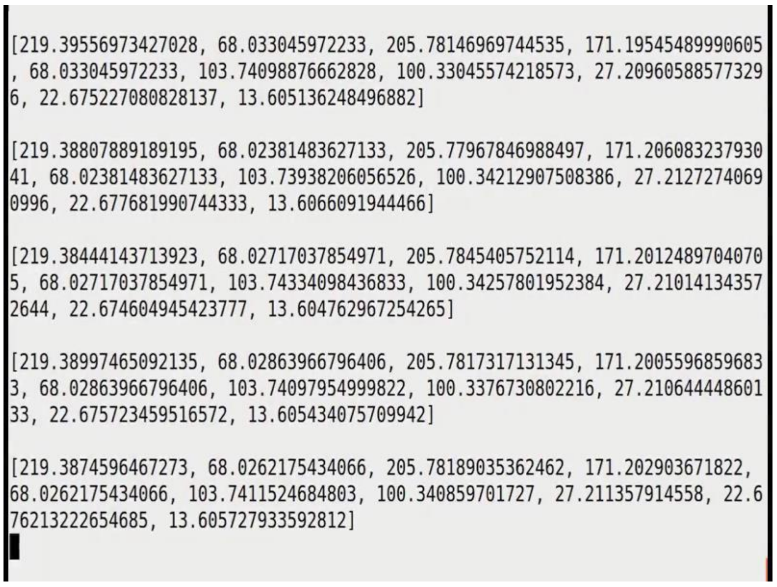
(Refer Slide Time: 06:01)



```
[150.00000000000003, 90.00000000000001, 218.8888888888889, 176.6666666666666
9, 90.00000000000001, 130.0, 80.00000000000001, 28.888888888888893, 22.22222
2222222225, 13.333333333333334]

[236.8888888888889, 70.0, 183.0, 170.22222222222223, 70.0, 110.7777777777777
9, 76.66666666666667, 34.44444444444445, 30.000000000000004, 18.000000000000
004]

[197.0, 62.48148148148149, 203.62962962962962, 187.66666666666669, 62.481481
48148149, 105.11111111111111, 114.96296296296298, 29.333333333333332, 23.333
333333333332, 14.0]

[216.1259259259259, 73.35802469135803, 210.31728395061728, 163.360493827160
52, 73.35802469135803, 110.99629629629631, 93.66666666666667, 25.493827160493
83, 20.82716049382716, 12.496296296296297]

[224.98888888888888, 68.220987654321, 199.61604938271606, 173.6588477366255,
 68.220987654321, 100.5172839506173, 97.03456790123457, 28.618106995884773,
24.45267489711934, 14.671604938271605]
```

And I keep on seeing the values changing as you can see. So, let us concentrate on the first node as of now. So, 224 I keep pressing enter because it is keeping on changing it is 2 219.

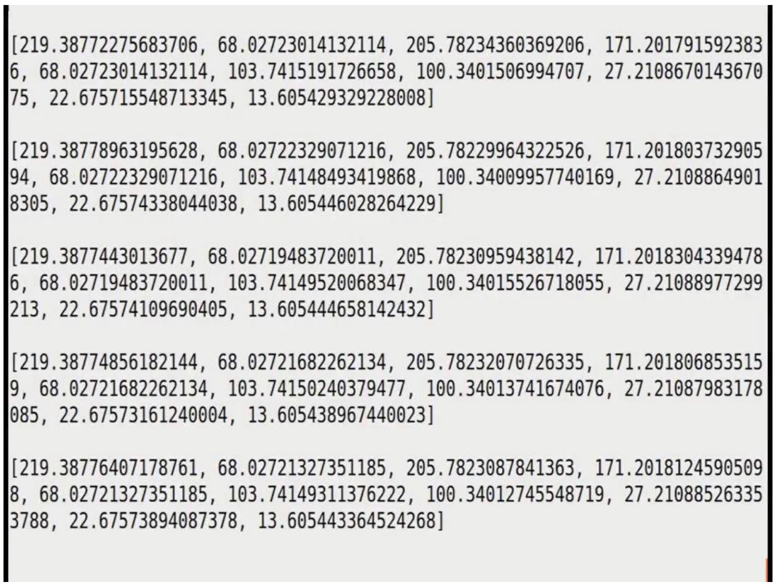
(Refer Slide Time: 06:14)



```
[219.39556973427028, 68.033045972233, 205.78146969744535, 171.19545489990605
, 68.033045972233, 103.74098876662828, 100.33045574218573, 27.20960588577329
6, 22.675227080828137, 13.605136248496882]

[219.38807889189195, 68.02381483627133, 205.77967846988497, 171.206083237930
41, 68.02381483627133, 103.73938206056526, 100.34212907508386, 27.2127274069
0996, 22.677681990744333, 13.6066091944466]

[219.38444143713923, 68.02717037854971, 205.7845405752114, 171.2012489704070
5, 68.02717037854971, 103.74334098436833, 100.34257801952384, 27.2101413435
72644, 22.674604945423777, 13.604762967254265]

[219.38997465092135, 68.02863966796406, 205.7817317131345, 171.2005596859683
3, 68.02863966796406, 103.74097954999822, 100.3376730802216, 27.210644448601
33, 22.675723459516572, 13.605434075709942]

[219.387459646727 3, 68.0262175434066, 205.78189035362462, 171.202903671822,
68.0262175434066, 103.7411524684803, 100.340859701727, 27.211357914558, 22.6
76213222654685, 13.605727933592812]
█
```

And only the decimal values are changing ok.

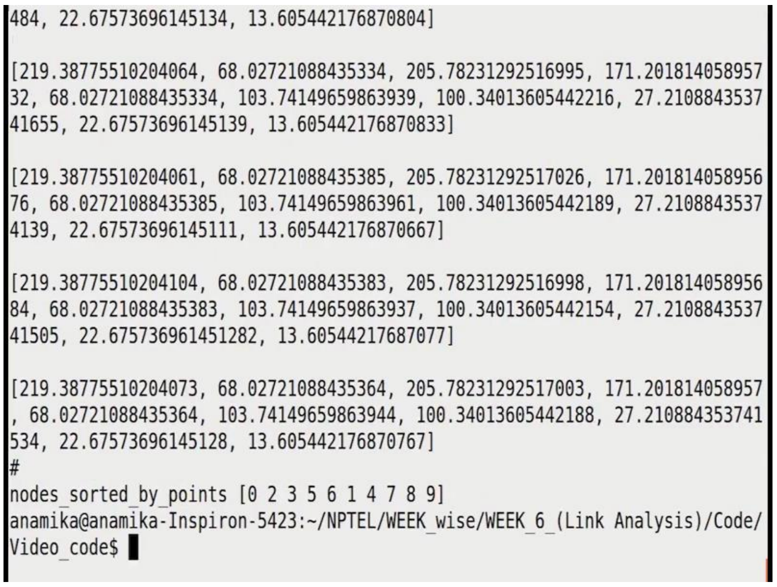
(Refer Slide Time: 06:20)



```
[219.38772275683706, 68.02723014132114, 205.78234360369206, 171.201791592383
6, 68.02723014132114, 103.7415191726658, 100.3401506994707, 27.2108670143670
75, 22.675715548713345, 13.605429329228008]

[219.38778963195628, 68.02722329071216, 205.78229964322526, 171.201803732905
94, 68.02722329071216, 103.74148493419868, 100.34009957740169, 27.2108864901
8305, 22.67574338044038, 13.605446028264229]

[219.3877443013677, 68.02719483720011, 205.78230959438142, 171.2018304339478
6, 68.02719483720011, 103.74149520068347, 100.34015526718055, 27.21088977299
213, 22.67574109690405, 13.605444658142432]

[219.38774856182144, 68.02721682262134, 205.7823207072633 5, 171.201806853515
9, 68.02721682262134, 103.74150240379477, 100.34013741674076, 27.21087983178
085, 22.67573161240004, 13.605438967440023]

[219.38776407178761, 68.02721327351185, 205.7823087841363, 171.2018124590509
8, 68.02721327351185, 103.74149311376222, 100.34012745548719, 27.21088526335
3788, 22.67573894087378, 13.605443364524268]
```

Let us let us look at any other node if the values are changing drastically. So, if the values are not at all changing, we will stop of course, if the values are changing slightly, we can either keep going on and on or we can stop there. So, that that should be enough and that is what I am mean to say; let us see if any other value is changing. I am giving on pressing enter and the values are only slightly changing. So, I think I can stop at any moment, let me press hash here.

(Refer Slide Time: 06:46)



```
484, 22.67573696145134, 13.605442176870804]

[219.38775510204064, 68.02721088435334, 205.78231292516995, 171.201814058957
32, 68.02721088435334, 103.74149659863939, 100.34013605442216, 27.2108843537
41655, 22.67573696145139, 13.605442176870833]

[219.38775510204061, 68.02721088435385, 205.78231292517026, 171.201814058956
76, 68.02721088435385, 103.74149659863961, 100.34013605442189, 27.2108843537
4139, 22.67573696145111, 13.605442176870667]

[219.38775510204104, 68.02721088435383, 205.78231292516998, 171.201814058956
84, 68.02721088435383, 103.74149659863937, 100.34013605442154, 27.2108843537
41505, 22.675736961451282, 13.60544217687077]

[219.38775510204073, 68.02721088435364, 205.78231292517003, 171.201814058957
, 68.02721088435364, 103.74149659863944, 100.34013605442188, 27.210884353741
534, 22.67573696145128, 13.605442176870767]
#
nodes_sorted_by_points [0 2 3 5 6 1 4 7 8 9]
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ 
```

So, our loops stops ok, this function is looking fine. So, I getting the nodes_sorted_by_points, these are the sorted nodes that we are getting. Basically, these are rankings; this is the ranking of the nodes. So, the node 0 is having the highest page rank node 2 is on the second number and so on. So, this is the ranking that we obtained. Now, let us see whether this ranking is correct or not as per the page rank; by third that is there in networkx. So, let us get back get to the last step which is compare the ranks thus, obtained the ranks obtained with the inbuilt page rank method.

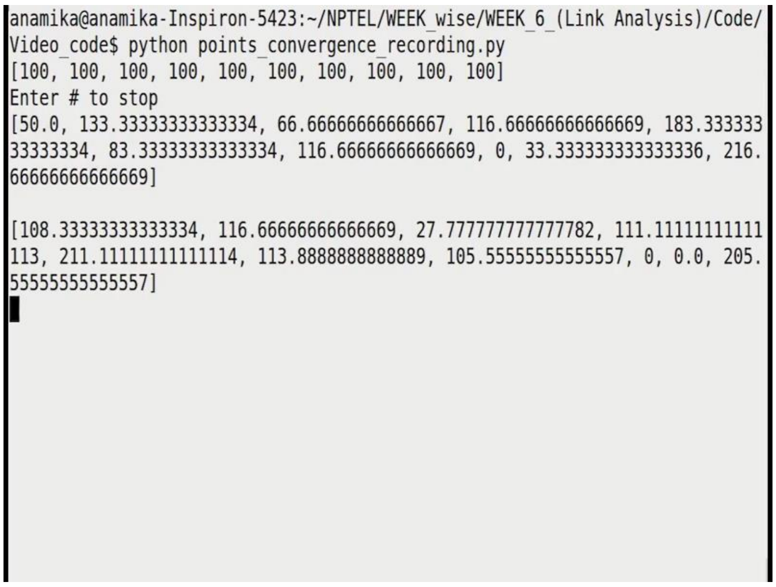
Now, let me tell you there is an inbuilt function which we can use to compute the page rank values for the nodes for a given graph. We have ourselves implemented it over here, there are several other methods as well let me tell you. We have implemented one of the methods which is using the distribution of the points. We might look at other methods as well maybe in the subsequent videos; let us check the ranking that we obtained from page rank. So, the function is nx.pagerank and we will pass the graph here ok.

Let me tell you what it returns, it basically returns a dictionary where the keys are the nodes and the values are the page ranks of the nodes. So, basically it does not tell the ranking of the nodes, it tells us the page ranks page rank values of each node. So, we would have to rank them ourselves. So, how can we do that? Let me sort this dictionary ok, we cannot sort the dictionary because the dictionary does not contain any sorting order. So, we can create a list of the tuples from the dictionary as we have been doing in the previous videos as well.

So, how can we do that? Let us see so, this is going to be a list of the sorted tuples based on the values. So, the function that we can use is sorted, I write pr.items and same old method that we use key is equal to lambda. How can how do we want to sort it? We want to sort it based on the value which is the second thing in the tuple which is x1. So, we will write x : x1 and by default it is a ascending right and we want it to be descending.

So, we will write reverse is equal to True. So, pr sorted is a list of the tuples ok. What we want? We want the nodes; we want the nodes sorted by the page rank value only. So, maybe what we can do is, we can only print the first value from each tuple. So, write for i in pr_sorted ok, I am going to only print the first value for every i ok. So, I want all the values in same line so, I have put a comma here ok. Let us check the working of the code. So, basically what we have to check is the ranking that we were getting r by r method and the ranking that we are getting by this method are the same or not.

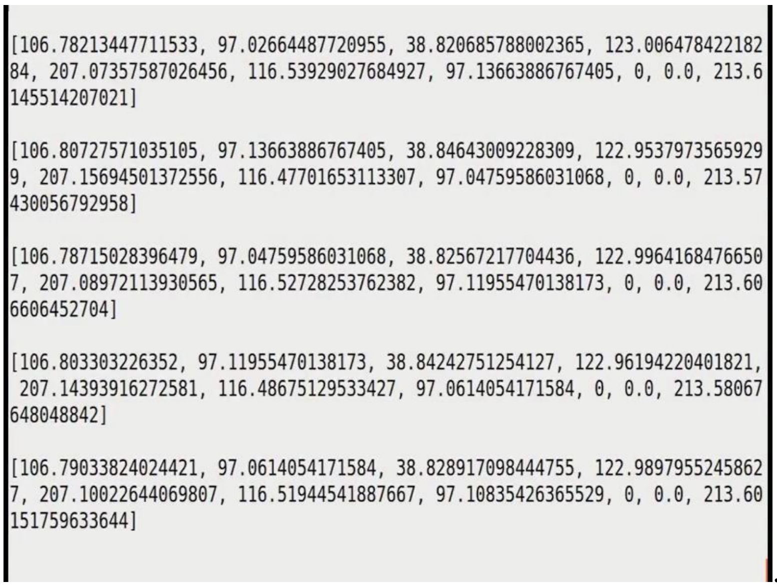
(Refer Slide Time: 10:20)



```
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[50.0, 133.33333333333334, 66.66666666666667, 116.66666666666669, 183.333333
33333334, 83.33333333333334, 116.66666666666669, 0, 33.333333333333336, 216.
66666666666669]

[108.33333333333334, 116.66666666666669, 27.777777777777782, 111.1111111111
113, 211.11111111111114, 113.8888888888889, 105.55555555555557, 0, 0.0, 205.
55555555555557]
```

Let us check alright. So, these are the initial values 100 each and after first iteration these are the values that we get. We got 50 points for the first node now, I am pressing enter and the values are changing ok. So, the values are changing.

(Refer Slide Time: 10:41)

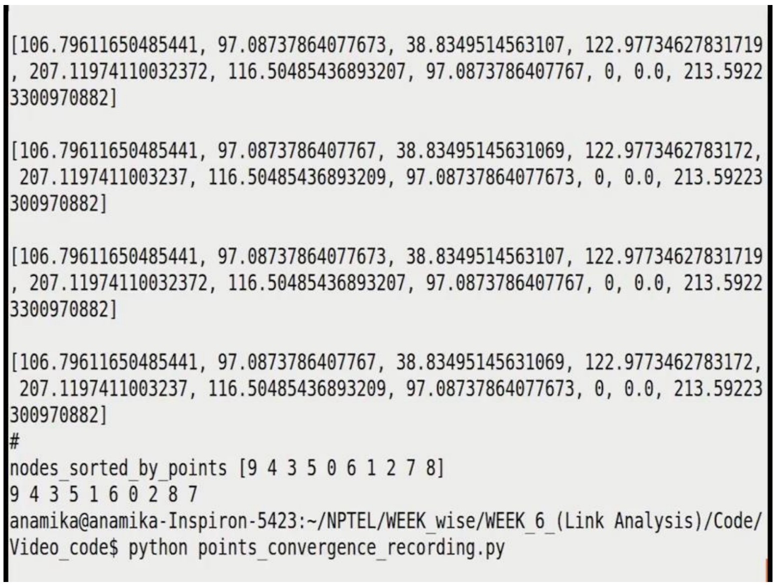


```
[106.78213447711533, 97.02664487720955, 38.820685788002365, 123.006478422182
84, 207.07357587026456, 116.53929027684927, 97.13663886767405, 0, 0.0, 213.6
145514207021]

[106.80727571035105, 97.13663886767405, 38.84643009228309, 122.9537973565929
9, 207.15694501372556, 116.47701653113307, 97.04759586031068, 0, 0.0, 213.57
430056792958]

[106.78715028396479, 97.04759586031068, 38.82567217704436, 122.9964168476650
7, 207.08972113930565, 116.52728253762382, 97.11955470138173, 0, 0.0, 213.60
6606452704]

[106.803303226352, 97.11955470138173, 38.84242751254127, 122.96194220401821,
 207.14393916272581, 116.48675129533427, 97.0614054171584, 0, 0.0, 213.58067
648048842]

[106.79033824024421, 97.0614054171584, 38.828917098444755, 122.9897955245862
7, 207.10022644069807, 116.51944541887667, 97.10835426365529, 0, 0.0, 213.60
151759633644]
```

.

Let us the, for first node has 106, only the decimal values are changing. And, let us check the second one, second one is also slightly changing third, fourth I think all the nodes are only slightly changing. So, we can stop at this moment or maybe you can go ahead a little more ok. I am just keeping on pressing enter; it is just changing very slightly. It might take number of iterations to stop. So, maybe we can press hash and stop at this point.

(Refer Slide Time: 11:15)



```
[106.79611650485441, 97.08737864077673, 38.8349514563107, 122.97734627831719
, 207.11974110032372, 116.50485436893207, 97.0873786407767, 0, 0.0, 213.5922
3300970882]

[106.79611650485441, 97.0873786407767, 38.83495145631069, 122.9773462783172,
 207.1197411003237, 116.50485436893209, 97.08737864077673, 0, 0.0, 213.59223
300970882]

[106.79611650485441, 97.08737864077673, 38.8349514563107, 122.97734627831719
, 207.11974110032372, 116.50485436893207, 97.0873786407767, 0, 0.0, 213.5922
3300970882]

[106.79611650485441, 97.0873786407767, 38.83495145631069, 122.9773462783172,
 207.1197411003237, 116.50485436893209, 97.08737864077673, 0, 0.0, 213.59223
300970882]
#
nodes_sorted_by_points [9 4 3 5 0 6 1 2 7 8]
9 4 3 5 1 6 0 2 8 7
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
```

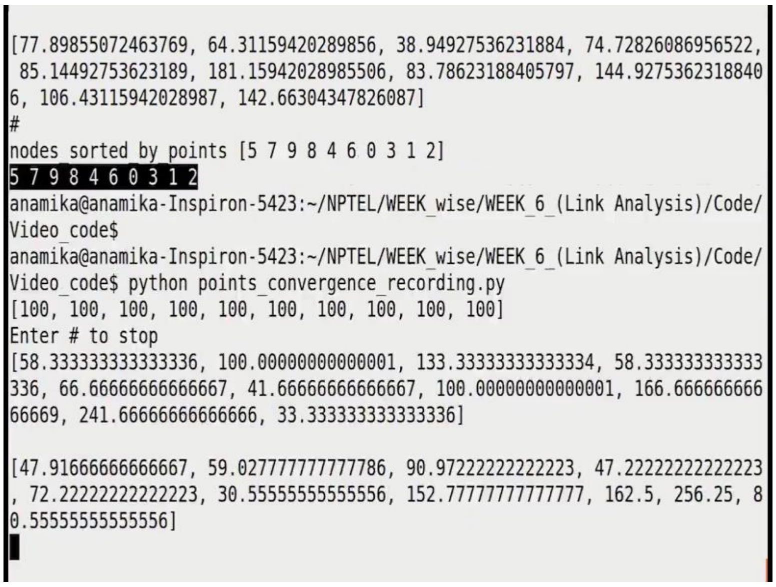
This is something that I want to tell you. So, this is the first thing that you can see is the ranking that we got by our method ok. And, the second row is the values that we got by the page rank method from networkx. Now, let us compare 9 4 3 5 is correct after that 0 6 1 2 7 8. So, we can we see some changes here.

This is what I wanted to show you, why these changes are coming and how we can handle that ok. So, can you imagine what can happen if there is a node which has no out link. So, as we also implemented if a node has no out link, this node does not distribute its points. So, in every iteration, what happens it keeps on changing; it keeps on getting points ok. It keeps on getting point from other nodes and it never distributes its own points. So, what happens is that the point sink happens.

Sink basically all the points are getting accumulated at 1 node or a set of nodes. So, this example I gave for 1 node, it may also happen that in the graph that we generated there are a bunch of nodes which are not distributing any points outside. So, basically there is a link from 1 node to the other and there is a link from second node to the first node, but there is no link outside. So, these 2 nodes will keep accumulating all the points and that is how the ranking that we will get will not be accurate.

So, for that we need to take some measures, we must handle that case. The networkx function is basically handling those cases whereas; our function is not handling those cases; that is why we are getting these differences in the ranking. Now, we are going to handle these cases. Let me tell you it will not happen in all the cases, because in all the cases your graph might not be having such sinks right. Let us check one more example, if we are getting the same. So, let us check for the convergence first.

(Refer Slide Time: 13:31)

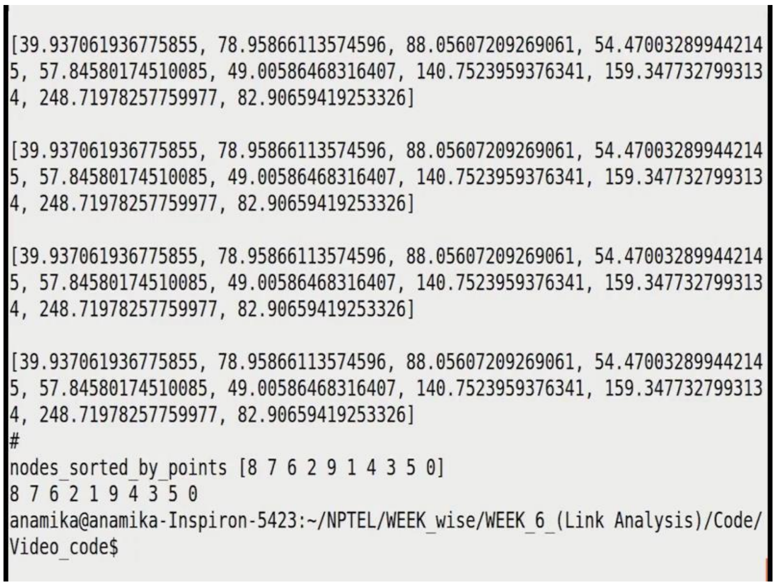


```
[77.89855072463769, 64.31159420289856, 38.94927536231884, 74.72826086956522,
 85.14492753623189, 181.15942028985506, 83.78623188405797, 144.9275362318840
6, 106.43115942028987, 142.66304347826087]
#
nodes sorted by points [5 7 9 8 4 6 0 3 1 2]
5 7 9 8 4 6 0 3 1 2
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$ python points_convergence_recording.py
[100, 100, 100, 100, 100, 100, 100, 100, 100, 100]
Enter # to stop
[58.333333333333336, 100.00000000000001, 133.33333333333334, 58.333333333333
336, 66.66666666666667, 41.66666666666667, 100.00000000000001, 166.666666666
66669, 241.66666666666666, 33.333333333333336]

[47.91666666666667, 59.027777777777786, 90.97222222222223, 47.22222222222223
, 72.22222222222223, 30.55555555555556, 152.77777777777777, 162.5, 256.25, 8
0.55555555555556]
```

Again the values are keeping changing ok, now I am pressing enter and it is not going dead basically the values are not at all changing. So, it is completely stopped, there is no change now I am keeping on pressing enter. So, we can just press hash, let us see the ranking that we getting we are getting 5 7 9 8 4 6 0 3 1 2 ok, that is again another thing I want to show. So, this might be a graph where there is no such problem as I just explained you there is no sink.

There is no 1 node or a set of nodes which are keeping on accumulating all the points and they are not distributing that case is not here and that is why we are getting the same thing here. So, our method is accurate just in case there is no sink there in the graph ok. Let me just take another case and I am sure there will be some problem here. I actually want some problem to happen so, that the next video we can cover that because, we are going to handle such cases.

(Refer Slide Time: 14:28)

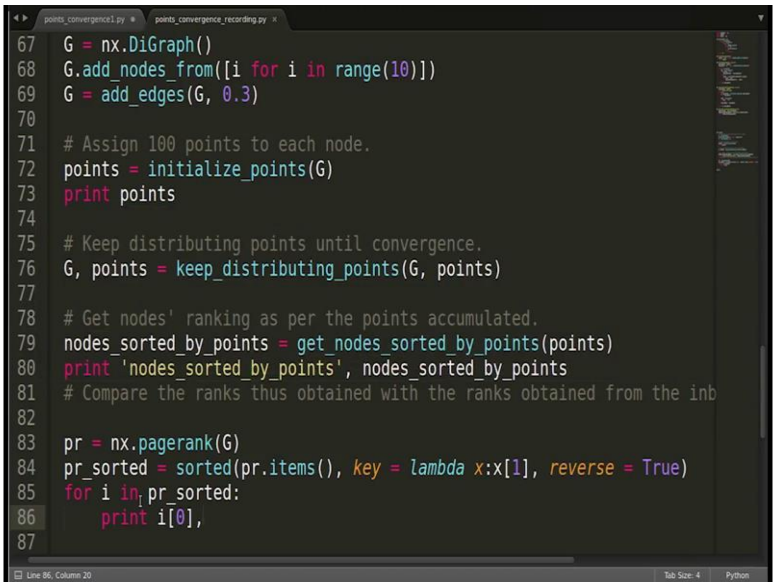


```
[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]

[39.937061936775855, 78.95866113574596, 88.05607209269061, 54.47003289944214
5, 57.84580174510085, 49.00586468316407, 140.7523959376341, 159.347732799313
4, 248.71978257759977, 82.90659419253326]
#
nodes_sorted_by_points [8 7 6 2 9 1 4 3 5 0]
8 7 6 2 1 9 4 3 5 0
anamika@anamika-Inspiron-5423:~/NPTEL/WEEK_wise/WEEK_6_(Link Analysis)/Code/
Video_code$
```

I am again pressing the enter and it is not going ahead, it is stopped. So, 8 7 6 2 1 you pick 2 1. Yeah, we got the difference nice. So, 2 9 1 and 2 1 9 so, we got a difference right; let me tell you one more thing sometimes there might be a slight you know variation for another reason as well. And, that another reason could be that 2 nodes are having exactly same page rank values right.

So, in that case your method can rank them in a different order and page rank method from networkx for example, can rank them in a different order. So, that sort of difference can happen even if your method is completely working fine and taking care of the sink cases as well. So, that you can just ignore, but as of now we know that our code is not taking care of the sink cases. So, we need to do that ok. So, let us get back to our code.

(Refer Slide Time: 15:30)



points_convergence1.py   points_convergence_recording.py

```
G = nx.DiGraph()
G.add_nodes_from([i for i in range(10)])
G = add_edges(G, 0.3)

# Assign 100 points to each node.
points = initialize_points(G)
print points

# Keep distributing points until convergence.
G, points = keep_distributing_points(G, points)

# Get nodes' ranking as per the points accumulated.
nodes_sorted_by_points = get_nodes_sorted_by_points(points)
print 'nodes_sorted_by_points', nodes_sorted_by_points
# Compare the ranks thus obtained with the ranks obtained from the inb

pr = nx.pagerank(G)
pr_sorted = sorted(pr.items(), key = lambda x:x[1], reverse = True)
for i in pr_sorted:
    print i[0],

```

Line 86, Column 20   Tab Size: 4   Python

And in the next video we will be handling that case.