import networkx as nx  
import matplotlib.pyplot as plt  
import seaborn as sns  
import random  
import numpy as np  
  
  
print ("start")  
n=2  
NumberOfChance=10  
numberOfMembers= 100  
result = [ [ 0 for i in range(NumberOfChance) ] for j in range(n) ]  
numOfIterations=0  
  
  
for iz in range (10):  
 print ("iteraion i = ", iz ," % " , (iz+1)/10 )  
 result[0][iz]= (iz+1)/10  
 gr2 = nx.gnp\_random\_graph(numberOfMembers, (iz+1)/10 , seed=354, directed=False)  
 '''nx.draw(gr2, with\_labels=True)'''  
 pos = nx.spring\_layout(gr2)  
  
 labels = {}  
 for x in range(len(gr2)):  
 labels[x] = random.choice(['A', 'B', 'C'])  
 print("Labels :", labels)  
  
 neighbors = gr2.edges  
 print("the neighbors: ", neighbors)  
  
 NumberOfOpnions=3  
 votes = [ [ 0 for i in range(NumberOfOpnions) ] for j in range(numberOfMembers) ]  
 votes2 = [ [ 0 for i in range(NumberOfOpnions) ] for j in range(numberOfMembers) ]  
 arrayOfNeighbors = [ [ 0 for i in range(numberOfMembers) ] for j in range(numberOfMembers) ]  
 opinion = ['A', 'B', 'C']  
 threshold = 0.55  
 numOfIterations=0  
 change = True  
  
  
 for i in range(numberOfMembers):  
 placeInCol = 0  
 for j in range(len(neighbors)):  
 res = list(neighbors.keys())[j]  
 if (res[0] == i):  
 arrayOfNeighbors[i][placeInCol] = res[1]  
 placeInCol = placeInCol + 1  
 if (res[1] == i):  
 arrayOfNeighbors[i][placeInCol] = res[0]  
 placeInCol = placeInCol + 1  
 if( (j == (len(neighbors)-1)) and placeInCol == 0 ):  
 arrayOfNeighbors[i][placeInCol] = -1  
  
  
 print("arrayOfNeighbors" , arrayOfNeighbors)  
 print()  
 OneOfNeighbors = 0  
 while (change):  
 numOfIterations = numOfIterations + 1  
 change = False  
 for a in range(len(arrayOfNeighbors)):  
 counter = 0  
 votes = [[0 for i in range(NumberOfOpnions)] for j in range(numberOfMembers)]  
 for b in range(numberOfMembers):  
 OneOfNeighbors = arrayOfNeighbors[a][b]  
 if ((OneOfNeighbors != 0 and OneOfNeighbors!= (-1)) or (OneOfNeighbors == 0 and b ==0) ):  
 for k in range (len(opinion)):  
 if (labels[OneOfNeighbors] == opinion[k]):  
 votes[a][k] = votes[a][k] + 1  
 counter = counter + 1  
 for l in range (len(opinion)):  
 if(counter > 0):  
 votes2[a][l] = votes[a][l]/counter  
 if(votes2[a][l] >= threshold and labels[a] != opinion[l] ):  
 change = True  
 '''print("votes2 to change : " , votes2)  
 print(" we want to change a : ", a, " labels[a] = ", labels[a], "l", l, "opinion[l] = ", opinion[l])  
 labels[a] = opinion[l]  
 print("labels after change :", labels)  
 print(" ")'''  
 ''' print(" we not change a : ", a, " labels[a] = ", labels[a], "l", l, "opinion[l] = ", opinion[l])  
 print("votes2 : " , votes2)  
 print(" ")'''  
  
 print(" ")  
 print("votes : ", votes)  
 print("votes2 : ", votes2)  
 print("numOfIterations : ", numOfIterations)  
 print("End Iteration")  
 print(" ")  
  
 result[1][iz] = numOfIterations  
  
 '''nx.draw\_networkx(gr2)  
 plt.show()'''  
  
  
print ()  
print("result = ", result)  
  
# Plot the data  
dataPlot = plt.plot(result[0], result[1], label='linear')  
  
# Add a legend  
plt.legend()  
  
# Show the plot  
plt.show()

import networkx as nx  
import matplotlib.pyplot as plt  
import seaborn as sns  
import random  
import numpy as np  
  
  
print ("start")  
n=2  
NumberOfChance = 10  
numberOfMembers = 3  
result = [ [ 0 for i in range(NumberOfChance) ] for j in range(n) ]  
numOfIterations = 0  
  
  
for iz in range (10):  
 print ("iteraion i = ", iz ," % " , (iz+1)/10 )  
 result[0][iz]= (iz+1)/10  
 gr2 = nx.gnp\_random\_graph(numberOfMembers, (iz+1)/10 , seed=354, directed=False)  
 '''nx.draw(gr2, with\_labels=True)'''  
 pos = nx.spring\_layout(gr2)  
  
 labels = {}  
 for x in range(len(gr2)):  
 labels[x] = random.choice(['A', 'B', 'C'])  
 print("Labels :", labels)  
  
 neighbors = gr2.edges  
 print("the neighbors: ", neighbors)  
  
 NumberOfOpnions=3  
 votes = [ [ 0 for i in range(NumberOfOpnions) ] for j in range(numberOfMembers) ]  
 votes2 = [ [ 0 for i in range(NumberOfOpnions) ] for j in range(numberOfMembers) ]  
 arrayOfNeighbors = [ [ 0 for i in range(numberOfMembers) ] for j in range(numberOfMembers) ]  
 opinion = ['A', 'B', 'C']  
 threshold = 0.55  
 numOfIterations=0  
 change = True  
  
  
 for i in range(numberOfMembers):  
 placeInCol = 0  
 for j in range(len(neighbors)):  
 res = list(neighbors.keys())[j]  
 if (res[0] == i):  
 arrayOfNeighbors[i][placeInCol] = res[1]  
 placeInCol = placeInCol + 1  
 if (res[1] == i):  
 arrayOfNeighbors[i][placeInCol] = res[0]  
 placeInCol = placeInCol + 1  
 if( (j == (len(neighbors)-1)) and placeInCol == 0 ):  
 arrayOfNeighbors[i][placeInCol] = -1  
  
  
 print("arrayOfNeighbors" , arrayOfNeighbors)  
 print()  
 OneOfNeighbors = 0  
 while (change):  
 numOfIterations = numOfIterations + 1  
 change = False  
 for a in range(len(arrayOfNeighbors)):  
 counter = 0  
 votes = [[0 for i in range(NumberOfOpnions)] for j in range(numberOfMembers)]  
 for b in range(numberOfMembers):  
 OneOfNeighbors = arrayOfNeighbors[a][b]  
 if ((OneOfNeighbors != 0 and OneOfNeighbors!= (-1)) or (OneOfNeighbors == 0 and b ==0) ):  
 for k in range (len(opinion)):  
 if (labels[OneOfNeighbors] == opinion[k]):  
 votes[a][k] = votes[a][k] + 1  
 counter = counter + 1  
 else :  
 break  
 for l in range (len(opinion)):  
 if(counter > 0):  
 votes2[a][l] = votes[a][l]/counter  
 if(votes2[a][l] >= threshold and labels[a] != opinion[l] ):  
 change = True  
 '''print("votes2 to change : " , votes2)  
 print(" we want to change a : ", a, " labels[a] = ", labels[a], "l", l, "opinion[l] = ", opinion[l])  
 labels[a] = opinion[l]  
 print("labels after change :", labels)  
 print(" ")'''  
 ''' print(" we not change a : ", a, " labels[a] = ", labels[a], "l", l, "opinion[l] = ", opinion[l])  
 print("votes2 : " , votes2)  
 print(" ")'''  
  
  
 print(" ")  
 print("votes : ", votes)  
 print("votes2 : ", votes2)  
 print("numOfIterations : ", numOfIterations)  
 print("End Iteration")  
 print(" ")  
  
 result[1][iz] = numOfIterations  
  
 '''nx.draw\_networkx(gr2)  
 plt.show()'''  
  
  
print ()  
print("result = ", result)  
  
# Plot the data  
dataPlot = plt.plot(result[0], result[1], label='linear')  
  
# Add a legend  
plt.legend()  
  
# Show the plot  
plt.show()

result100 = [[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], [5, 4, 1, 3, 1, 1, 1, 1, 1, 1]]