```python
#!/usr/bin/python
'''
BLG 517E - Modelling and Performance Analysis of Networks
Random Graphs Assignment
Tugrul Yatagan 504161551
'''
import igraph as ig
import numpy as np
import scipy as sp
import scipy.stats
import matplotlib.pyplot as plt

N=50
repeat=5
P_list=[0.05, 0.2, 0.4, 0.6, 0.8]
confidence=0.95

edge_list=[[0 for x in range(repeat)] for y in range(len(P_list))]
giant_size_list=[[0 for x in range(repeat)] for y in range(len(P_list))]
diameter_list=[[0 for x in range(repeat)] for y in range(len(P_list))]
avg_degree_list=[[0 for x in range(repeat)] for y in range(len(P_list))]

def confidence_interval(data, conf):
    a = 1.0*np.array(data)
    n = len(a)
    m, se = np.mean(a), sp.stats.sem(a)
    h = se * sp.stats.t._ppf((1+conf)/2., n-1)
    #return m, m-h, m+h, h
    return m, h


for k, P in enumerate(P_list):
    for i in range(repeat):
        g = ig.Graph.Erdos_Renyi(n=N, p=P)
        ig.plot(g, 'graph_p{0}_{1}.png'.format(P, i+1))
        print "Random Graph #{0}, n={1}, p={2}".format(i+1, N, P)
        edge_list[k][i] = g.ecount()
        print "  Number of edge = {0}".format(edge_list[k][i])
        giant_size_list[k][i] = max(g.components().sizes())
        print "  Giant component size = {0}".format(giant_size_list[k][i])
        diameter_list[k][i] = g.diameter()
        print "  Diameter = {0}".format(diameter_list[k][i])
        avg_degree_list[k][i] = np.mean(g.degree())
        print "  Average degree = {0}".format(avg_degree_list[k][i])
        print g.degree_distribution()
    print "***"

print edge_list
print giant_size_list
print diameter_list
print avg_degree_list


avg_degree_mean_list=[confidence_interval(x, confidence)[0] for x in avg_degree_list]
avg_degree_confidence_list=[confidence_interval(x, confidence)[1] * 2 for x in avg_degree_list]
plt.bar(range(len(P_list)), avg_degree_mean_list, yerr=avg_degree_confidence_list, alpha=0.2, align='center', width=0.4)
plt.xticks(range(len(P_list)), [str(p) for p in P_list])
for m, c, k in zip(avg_degree_mean_list, avg_degree_confidence_list, range(len(P_list))):
        plt.annotate("{0:.1f} +- {1:.1f}".format(m, c/2), (k, m+2), va='bottom', ha='center', fontsize=10)
plt.grid()
plt.title("Average degree")
plt.ylabel("Average degree")
plt.xlabel("p")
plt.savefig('avg_degree.png')
plt.gcf().clear()

edge_mean_list=[confidence_interval(x, confidence)[0] for x in edge_list]
edge_confidence_list=[confidence_interval(x, confidence)[1] * 2 for x in edge_list]
plt.bar(range(len(P_list)), edge_mean_list, yerr=edge_confidence_list, alpha=0.2, align='center', width=0.4)
```

```
plt.xticks(range(len(P list)), [str(p) for p in P list])
for m, c, k in zip(edge mean list, edge confidence list, range(len(P list))):
        plt.annotate("{0:.1f} +- {1:.1f}".format(m, c/2), (k, m+10), va='bottom', ha='center', fontsize=10)
plt.grid()
plt.title("Number of edge")
plt.ylabel("Number of edge")
plt.xlabel("p")
plt.savefig('edge.png')
plt.gcf().clear()

giant size mean list=[confidence interval(x, confidence)[0] for x in giant size list]
giant size confidence list=[confidence interval(x, confidence)[1] * 2 for x in giant size list]
plt.bar(range(len(P list)), giant size mean list, yerr=giant size confidence list, alpha=0.2, align='center', width=0.4)
plt.xticks(range(len(P list)), [str(p) for p in P list])
for m, c, k in zip(giant size mean list, giant size confidence list, range(len(P list))):
        plt.annotate("{0:.1f} +- {1:.1f}".format(m, c/2), (k, m+0), va='bottom', ha='center', fontsize=10)
plt.grid()
plt.title("Giant component size")
plt.ylabel("Giant component size")
plt.xlabel("p")
plt.savefig('giant.png')
plt.gcf().clear()

diameter mean list=[confidence interval(x, confidence)[0] for x in diameter list]
diameter confidence list=[confidence interval(x, confidence)[1] * 2 for x in diameter list]
plt.bar(range(len(P list)), diameter mean list, yerr=diameter confidence list, alpha=0.2, align='center', width=0.4)
plt.xticks(range(len(P list)), [str(p) for p in P list])
for m, c, k in zip(diameter mean list, diameter confidence list, range(len(P list))):
        plt.annotate("{0:.1f} +- {1:.1f}".format(m, c/2), (k, m+0.2), va='bottom', ha='center', fontsize=10)
plt.grid()
plt.title("Diameter")
plt.ylabel("Diameter")
plt.xlabel("p")
plt.savefig('diameter.png')
plt.gcf().clear()
```