In [ ]:

```
#create a k-regular graph
```

In [ ]:

```
import networkx as nx #all libraraies are included
import numpy as np
import random
```

In [ ]:

```
n=1000 #number of node is n and dgeree is k
k=10
```

In [ ]:

```
def k_regular_graph(k,n):  #fucntion for creating k regular graph
  G1=nx.Graph()
  for i in range(0,n):
    G1.add_node(i)
  for i in range(n):
    for j in range(k//2):
      G1.add_edge(i,(i+j+1)%n)  # here we need to connect node by nearest neighbor
  return G1
```

In [ ]:

```
G_regular=k_regular_graph(k,n) #initial G regular graph
```

In [ ]:

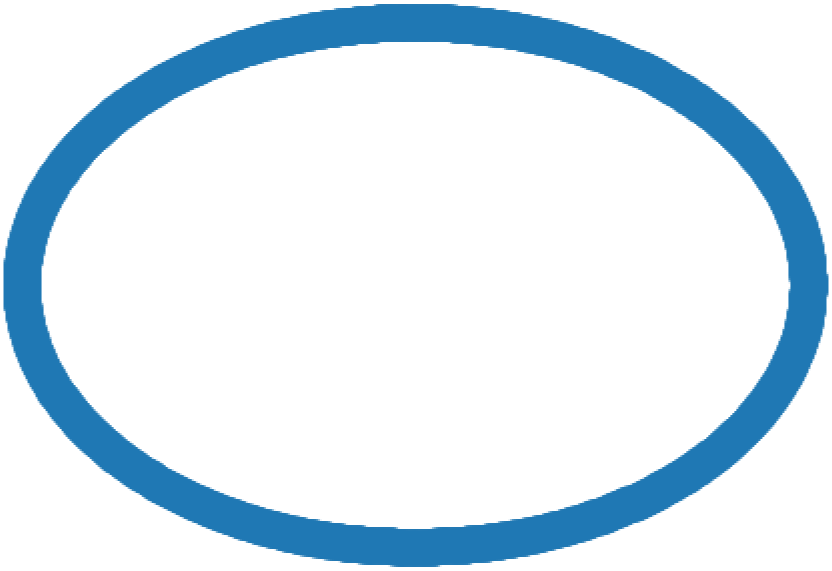
```
L_0=nx.average_shortest_path_length(G_regular) # initial avg shortest path length of initial graph
```

In [ ]:

```
C_0=nx.average_clustering(G_regular) #avg initial clustering coefficient of initial graph
```

In [ ]:

```
nx.draw_circular(G_regular)  #drwa of initial graph
```



In [ ]:

```
# edge_list
```

In [ ]:
```
def Graph_rewire(n,k,p):   #rewiring of the graph
  G2=k_regular_graph(k,n)   #generate k regular graph
  for i in range(int(k/2)):  #run loop k/2 times
    for j in range(0,n):     # run loop n times
        p_random=np.random.uniform(0.0,1.0)    #generate random prob
        lo=[]
        if p_random<p:  # checking the condition for prob
          for l in range(0,n):
            if l!=j and G2.has_edge(j,l)==False: # check for edges and node
              lo.append(l)
          random_node=np.random.choice(lo)  # random choice for nodes
          # print("lo:",j,lo)
          # print("random_node:",random_node)
          G2.add_edge(j,random_node)  # add the edges
          if G2.has_edge(j,(j+i+1)%n):  #
            G2.remove_edge(j,(j+i+1)%n)
  return G2
```

In [ ]:
```
# Graph_rewire(20,4,0.012)
```

In [ ]:
```
# nx.draw_circular(G_regular)
```

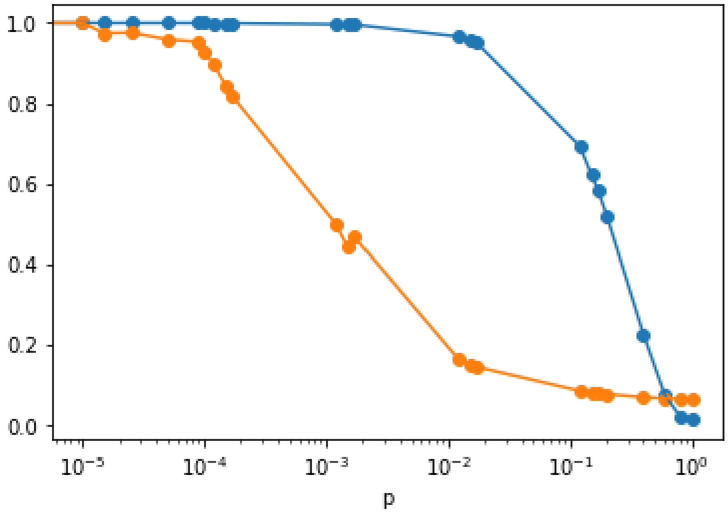
In [ ]:
```
cl=[]
pl=[]
iter=20
# p=14points
for p in [0,0.00001,0.000015,0.000025,0.000050,0.000090,0.0001, 0.00012, 0.000
15, 0.00017,0.0012, 0.0015, 0.0017, 0.012, 0.015,0.017,0.12,0.15,0.17,0.2,0.4,
0.6,0.8,1.0]:
  cl_n=[]
  pl_n=[]
  # print("p:",p)
  for i in range(iter):



    G3=Graph_rewire(n,k,p)
    # if if nx.is_connected(G):

    pl_n.append(nx.average_shortest_path_length(G3))
    cl_n.append(nx.average_clustering(G3))

  cl.append((sum(cl_n)/iter)/C_0)
  pl.append((sum(pl_n)/iter)/L_0)
```

In [ ]:
```
# cl
```

In [ ]:
```
# pl
```

In [ ]:

```
# importing two required module
import numpy as np
import matplotlib.pyplot as plt
X=[0,0.00001,0.000015,0.000025,0.000050,0.000090,0.0001, 0.00012, 0.00015, 0.00017,0.0012, 0.0015, 0.0017, 0.012, 0.015,0.017,0.12,0.15,0.17,0.2,0.4,0.6,0.8,1.0]
plt.scatter(X,cl)
plt.scatter(X,pl)
plt.plot(X, cl,label=" path length")
plt.plot(X, pl,label=" clustering coeff")
plt.xlabel("p")=
plt.xscale("log")
plt.show()
```



In [3]:

```
print("Details")
print("1.blue line showing cluster coeffients")
print("2.orange line showing averge path length")
print("3.y axis is degree")
```

```
Details
1.blue line showing cluster coeffients
2.orange line showing averge path length
3.y axis is degree
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: