In [ ]:

```
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['figure.dpi'] = 100
```

In [ ]:

```
def pref_attachment(T):
    """
    T: number of total residents
    Build a community using the preferential attachment model. Returns the
    networkx graph and the graph degree.
    """
    community = nx.Graph()
    community.add_edge(0, 1)
    for newcomer in range(2, T):
        resident_degrees = list(map(list, list(zip(*community.degree()))))
        residents = np.array(resident_degrees[0])
        degrees = np.array(resident_degrees[1])
        probability = degrees / sum(degrees)
        chosen_neighbor = np.random.choice(residents, p=probability)
        community.add_edge(newcomer, chosen_neighbor)
    return community, community.degree

def config_model(deg_distr):
    """
    deg_distr: The degree of each node
    Build a community using the configuration model and a list of the degrees of
    each node. Returns the networkx graph and the graph degree.
    """
    connections = np.array([])
    for i in range(len(deg_distr)):
        connections = np.append(connections, [i] * deg_distr[i])
    connections = np.random.permutation(connections)

    community = nx.Graph()
    for i in range(0, len(connections), 2):
        community.add_edge(connections[i], connections[i+1])

    return community, community.degree


def ccdf(data):
    '''Takes an array with random samples from a
    distribution, and creates an approximate CCDF
    (complementary CDF) of points. Returns a tuple of
    two vectors x, y where y_i = P(data > x_i)'''
    N = len(data)
    sorted = np.sort(data)
    y = np.linspace((N-1)/N, 0, N)
    return (sorted, y)
```

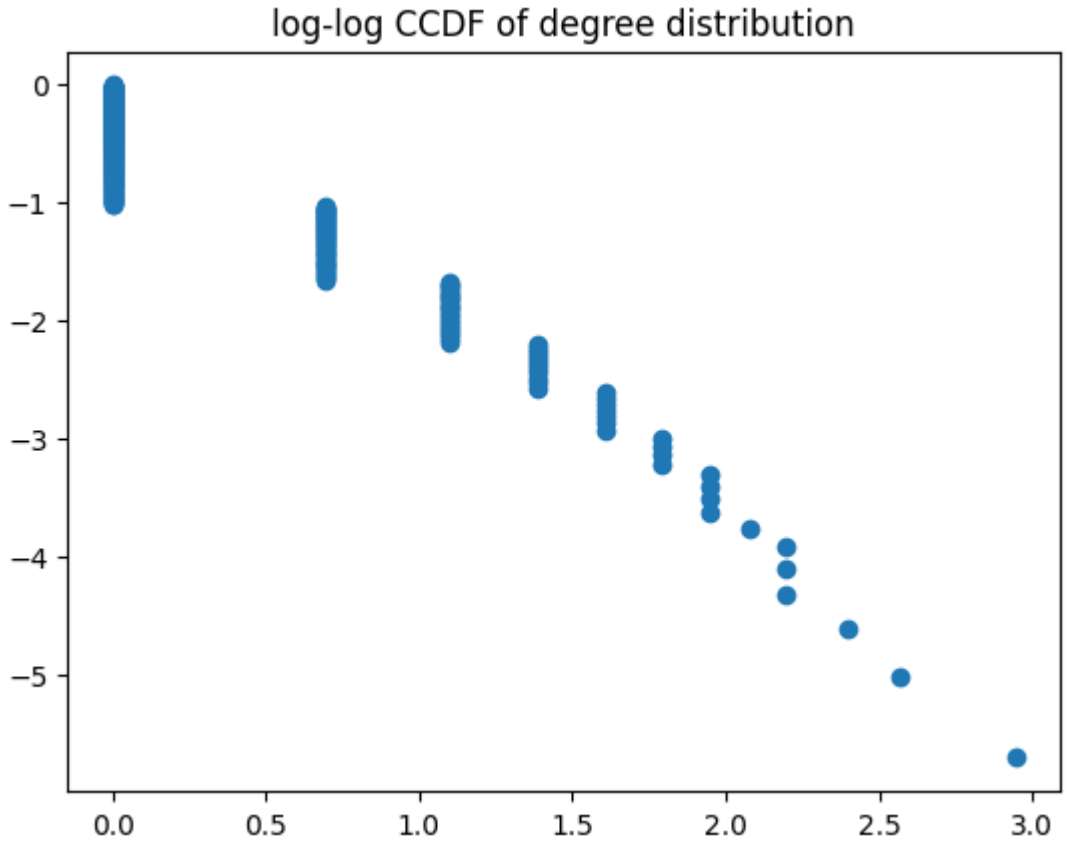
In [ ]:

```
pref_community, pref_community_deg = pref_attachment(300)

degrees = list(map(lambda node: node[1], list(pref_community_deg)))
distribution = ccdf(degrees)
```

```
plt.scatter(np.log(distribution[0]), np.log(distribution[1]))
plt.title("log-log CCDF of degree distribution")
```

Out[ ]:
```
<ipython-input-124-8bd1d4a5bce8>:5: RuntimeWarning: divide by zero encountered in log
  plt.scatter(np.log(distribution[0]), np.log(distribution[1]))
Text(0.5, 1.0, 'log-log CCDF of degree distribution')
```



In [ ]:
```
config_community, config_community_deg = config_model(degrees)
```

In [ ]:
```
plt.figure()
nx.draw(pref_community, pos=nx.spring_layout(pref_community), node_size=100)

plt.figure()
nx.draw(config_community, pos=nx.spring_layout(config_community), node_size=100)
```