In [ ]:
```
import networkx as nx
import numpy as np
import random
from matplotlib import pyplot as plt
random.seed()
```

In [ ]:
```
# n = 1000000
# c = 2m/n = 2log2
# m = nc/2 = nlog2 = 100*log2
G = nx.Graph()
```

We want to create a network of n nodes (one million). This is fixed. Given this network, from the constraint on the mean degree produce, we know how many total edges we need. So we just need to randomlt allocate these edges (could do while loop)

In [ ]:
```
# start with 100 nodes only
n = 100
iter = 0
while iter <= n:
    print(iter)
    name = 'Node_' + str(iter)
    print(name)
    G.add_node(name)
    iter += 1
```

```
0
Node_0
1
Node_1
2
Node_2
3
Node_3
4
Node_4
5
Node_5
6
Node_6
7
Node_7
8
Node_8
9
Node_9
10
Node_10
11
Node_11
12
Node_12
13
Node_13
```

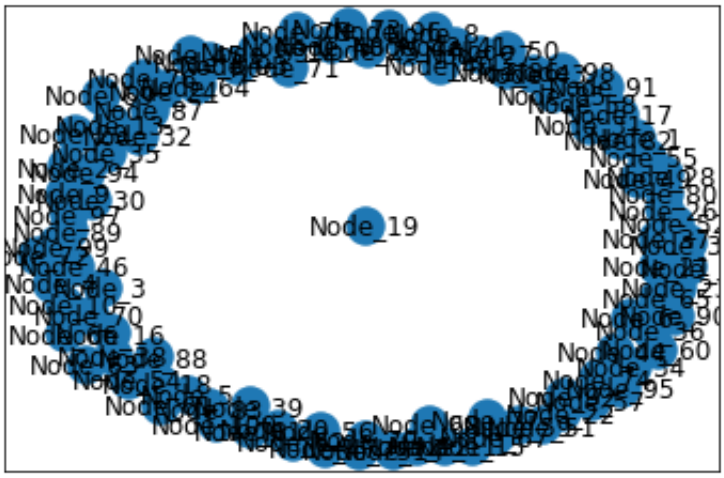
14
Node_14
15
Node_15
16
Node_16
17
Node_17
18
Node_18
19
Node_19
20
Node_20
21
Node_21
22
Node_22
23
Node_23
24
Node_24
25
Node_25
26
Node_26
27
Node_27
28
Node_28
29
Node_29
30
Node_30
31
Node_31
32
Node_32
33
Node_33
34
Node_34
35
Node_35
36
Node_36
37
Node_37
38
Node_38
39
Node_39
40
Node_40
41
Node_41
42

Node_42
43
Node_43
44
Node_44
45
Node_45
46
Node_46
47
Node_47
48
Node_48
49
Node_49
50
Node_50
51
Node_51
52
Node_52
53
Node_53
54
Node_54
55
Node_55
56
Node_56
57
Node_57
58
Node_58
59
Node_59
60
Node_60
61
Node_61
62
Node_62
63
Node_63
64
Node_64
65
Node_65
66
Node_66
67
Node_67
68
Node_68
69
Node_69
70
Node_70

71
Node_71
72
Node_72
73
Node_73
74
Node_74
75
Node_75
76
Node_76
77
Node_77
78
Node_78
79
Node_79
80
Node_80
81
Node_81
82
Node_82
83
Node_83
84
Node_84
85
Node_85
86
Node_86
87
Node_87
88
Node_88
89
Node_89
90
Node_90
91
Node_91
92
Node_92
93
Node_93
94
Node_94
95
Node_95
96
Node_96
97
Node_97
98
Node_98
99

```
Node_99
100
Node_100
```

In [ ]:
```
# print the graph that has no edges
nx.draw_networkx(G)
```



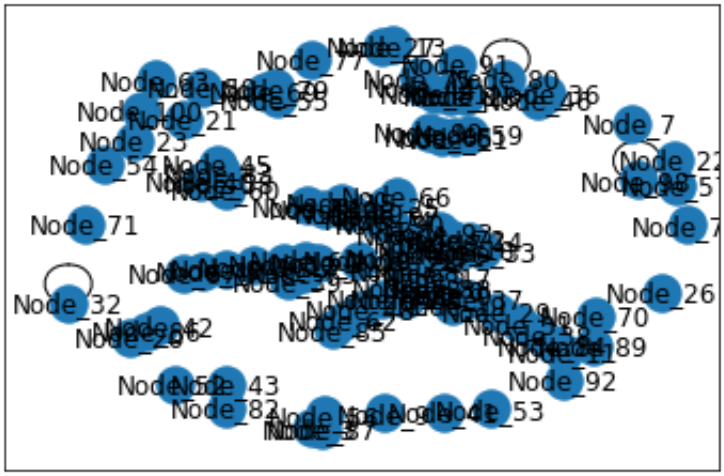
In [ ]:
```
tot_egdes = n * np.log(2)
print(tot_egdes)
```

```
69.31471805599453
```

In [ ]:
```
# randomly add 69 edges
iter = 0
while iter < tot_egdes:
    print(iter)
    node1 = "Node_" + str(random.randint(0, n))
    node2 = "Node_" + str(random.randint(0, n))
    if not G.has_edge(node1,node2):
        G.add_edge(node1,node2)
        iter += 1
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
16
17
18
```

In [ ]:
```
nx.draw_networkx(G)
```



In [ ]:
```
degrees = [val for (node, val) in G.degree()]

sum_of_edges = sum(degrees)
avg_degree = sum_of_edges / tot_egdes

print(2*np.log(2))
print(avg_degree)
```

```
1.3862943611198906
2.019773057244549
```

In [ ]:
```
Gcc = sorted(nx.connected_components(G), key=len, reverse=True)
G0 = G.subgraph(Gcc[0])
```

In [ ]:
```
print("full network node count is " + str(G.number_of_nodes()))
print("largest compent node count is " + str(G0.number_of_nodes()))
nx.draw_networkx(G0)
plt.show()
```

```
full network node count is 101
largest compent node count is 48
```