

# Communities & cliques

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON

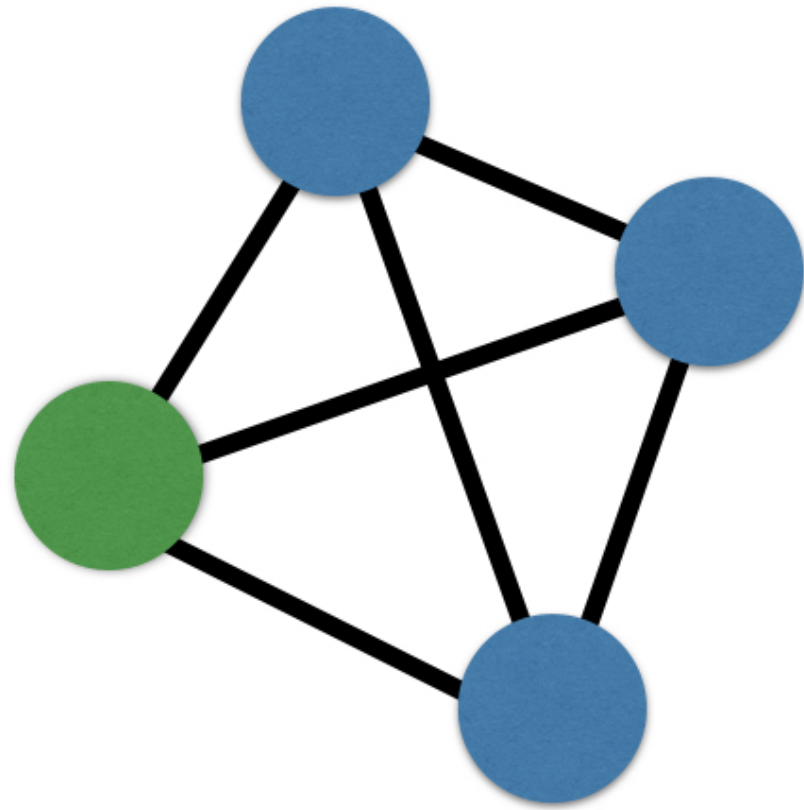


**Eric Ma**

Data Carpentry instructor and author of  
nxviz package

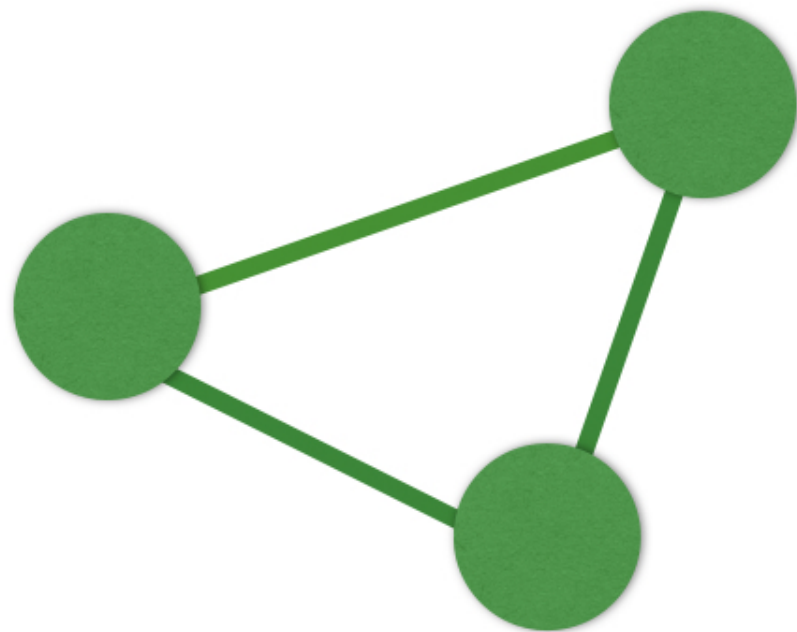
# Cliques

- Social cliques: tightly-knit groups
- Network cliques: completely connected graphs



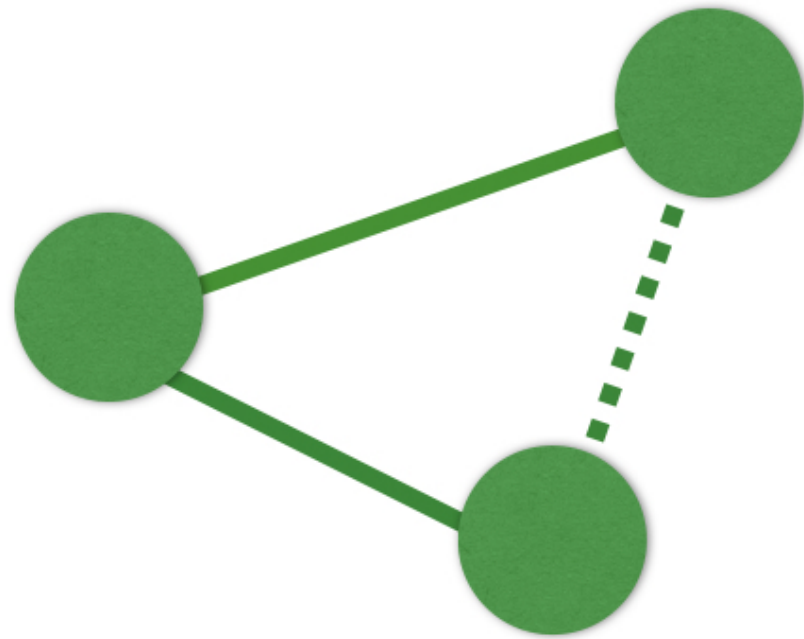
# Cliques

- Simplest complex clique: a triangle



# Triangle Applications

- Friend recommendation systems



# Clique Code

```
G
```

```
<networkx.classes.graph.Graph at 0x10c99ecf8>
```

```
from itertools import combinations
for n1, n2 in combinations(G.nodes(), 2):
    print(n1, n2)
```

```
0 1
0 2
0 3
0 4
0 5
...
```

# Let's practice!

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON

# Maximal cliques

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON

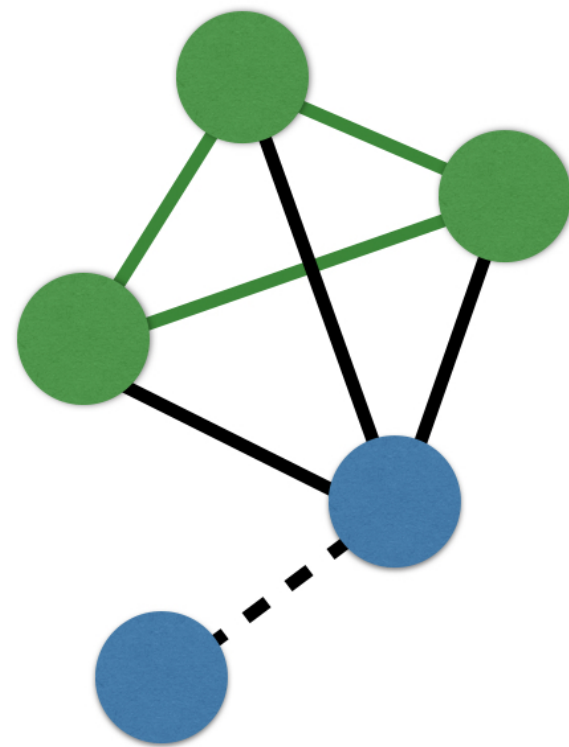


**Eric Ma**

Data Carpentry instructor and author of  
nxviz package

# Maximal cliques

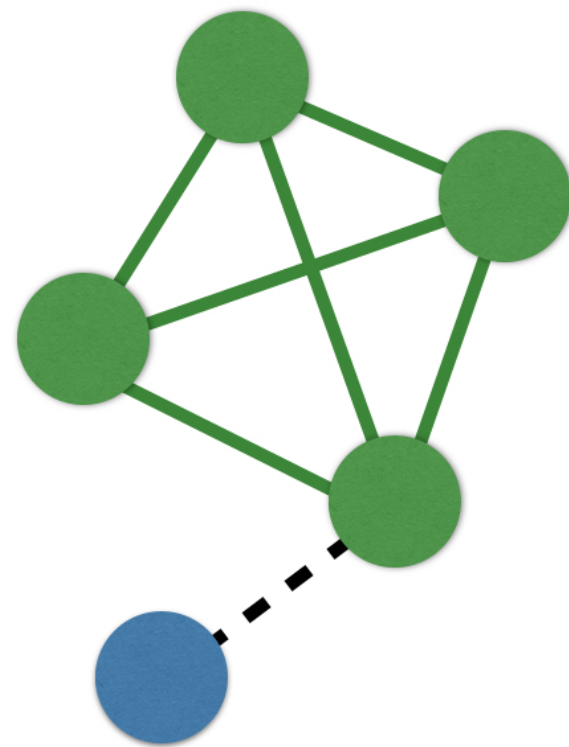
- Definition: a clique that, when extended by one node is no longer a clique





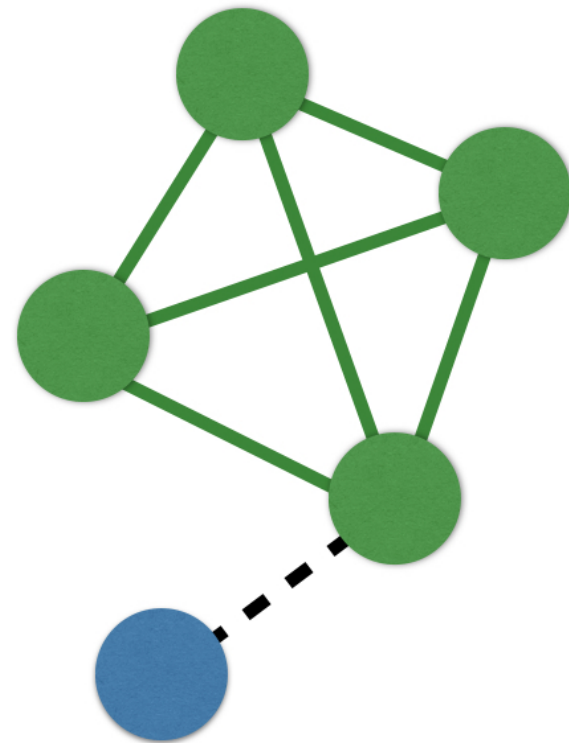
# Maximal cliques

- Definition: a clique that, when extended by one node is no longer a clique



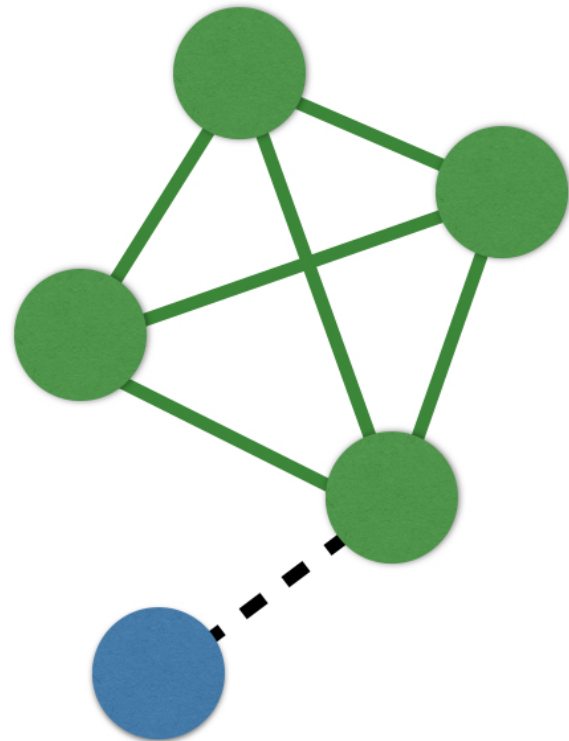
# Maximal cliques

- Applications: community finding



# Communities

- Find cliques
- Find unions of cliques



# NetworkX API

- `find_cliques` finds all maximal cliques

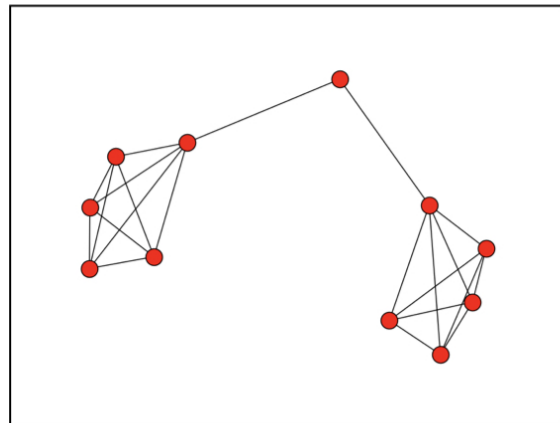
# Maximal cliques

```
import networkx as nx
G = nx.barbell_graph(m1=5, m2=1)
nx.find_cliques(G)
```

```
<generator object find_cliques at 0x1043f1f68>
```

```
list(nx.find_cliques(G))
```

```
[[4, 0, 1, 2, 3], [4, 5], [6, 8, 9, 10, 7], [6, 5]]
```



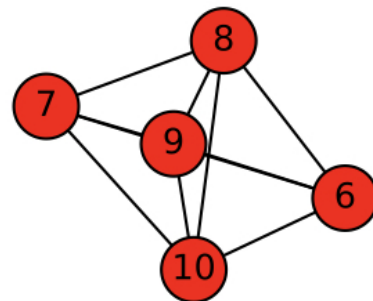
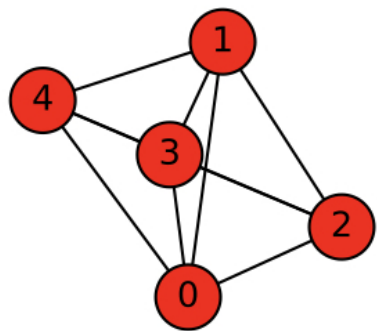
# Maximal cliques

```
import networkx as nx
G = nx.barbell_graph(m1=5, m2=1)
nx.find_cliques(G)
```

```
<generator object find_cliques at 0x1043f1f68>
```

```
list(nx.find_cliques(G))
```

```
[[4, 0, 1, 2, 3], [4, 5], [6, 8, 9, 10, 7], [6, 5]]
```



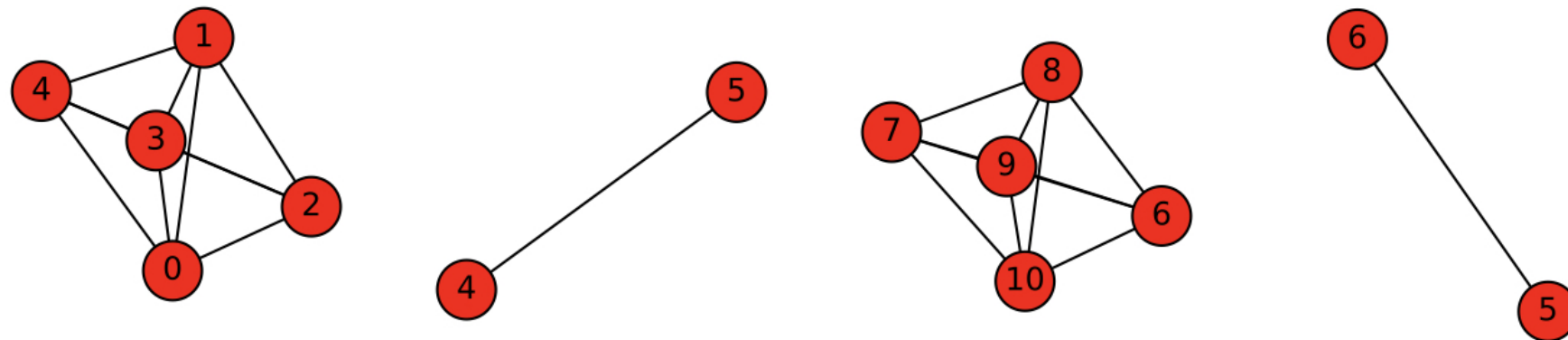
# Maximal cliques

```
import networkx as nx
G = nx.barbell_graph(m1=5, m2=1)
nx.find_cliques(G)
```

```
<generator object find_cliques at 0x1043f1f68>
```

```
list(nx.find_cliques(G))
```

```
[[4, 0, 1, 2, 3], [4, 5], [6, 8, 9, 10, 7], [6, 5]]
```



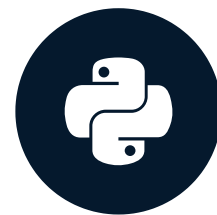
# Let's practice!

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON



# Subgraphs

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON



**Eric Ma**

Data Carpentry instructor and author of  
nxviz package

# Subgraphs

- Visualize portions of a large graph
  - Paths
  - Communities/cliques
  - Degrees of separation from a node

# Subgraphs

```
import networkx as nx
G = nx.erdos_renyi_graph(n=20, p=0.2)
G.nodes()
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```

```
nodes = G.neighbors(8)
nodes
```

```
[2, 3, 4, 10]
```

```
nodes.append(8)
```

# Subgraphs

```
G_eight = G.subgraph(nodes)
G_eight.edges()
```

```
[(8, 2), (8, 3), (8, 4), (8, 10), (2, 10)]
```

```
G_eight
```

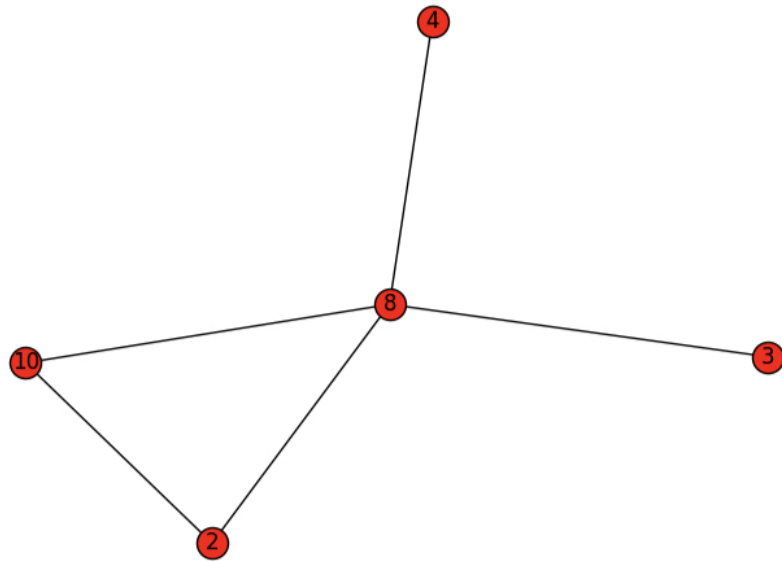
```
<networkx.classes.graph.Graph at 0x10cae39e8>
```

```
G
```

```
<networkx.classes.graph.Graph at 0x10cad1f60>
```

# Subgraphs

```
nx.draw(G_eight, with_labels=True)
```



# Let's practice!

INTRODUCTION TO NETWORK ANALYSIS IN PYTHON