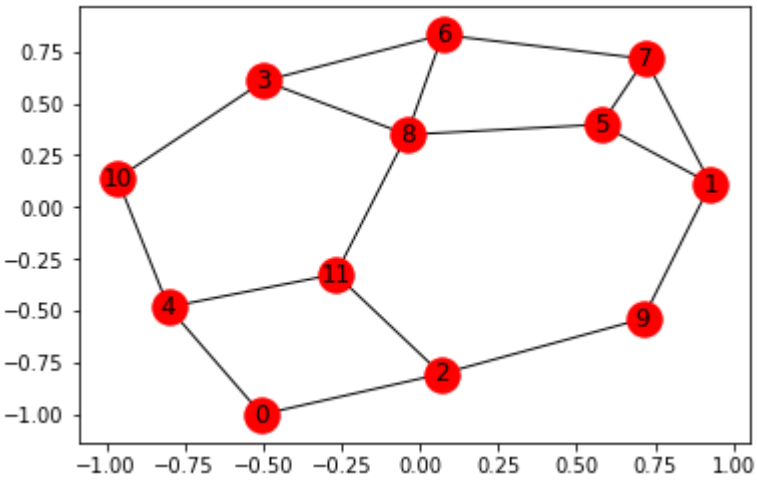
2. Basic Graph Operations

Author : Dongsoo S. Kim (dskim@iupui.edu (mailto:dskim@iupui.edu))
Date : Feb. 8, 2016

In [1]:
```
import networkx as nx
import matplotlib.pylab as plt
import random
#%matplotlib inline
```

In [2]:
```
G = nx.read_gexf('data/us-att.gexf', node_type=int)
print ('|V| =', G.number_of_nodes())
print ('|E| =', G.number_of_edges())
print ('connected?', nx.is_connected(G))
print (G.nodes())
nx.draw_networkx(G)
plt.show()
```

```
|V| = 12
|E| = 17
connected? True
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```



In [3]:
```
1 G.nodes(data=True)
```

Out[3]:
```
NodeDataView({0: {'latitude': 40.712756, 'abbr': 'nwy', 'name': 'New York, NY',
'longitude': -74.006047, 'population': 8175133, 'label': '0'}, 1: {'latitude':
33.94352, 'abbr': 'lax', 'name': 'Los Angeles, CA', 'longitude': -118.40866, 'p
opulation': 3792621, 'label': '1'}, 2: {'latitude': 41.878247, 'abbr': 'chi',
'name': 'Chicago, IL', 'longitude': -87.629767, 'population': 2695598, 'label':
'2'}, 3: {'latitude': 29.76429, 'abbr': 'hst', 'name': 'Houston, TX', 'longitud
e': -95.3837, 'population': 2099451, 'label': '3'}, 4: {'latitude': 39.952622,
'abbr': 'phl', 'name': 'Philadelphia, PA', 'longitude': -75.165708, 'populatio
n': 1526006, 'label': '4'}, 5: {'latitude': 33.445412, 'abbr': 'phx', 'name':
'Phoenix, AR', 'longitude': -112.073961, 'population': 1445632, 'label': '5'},
6: {'latitude': 29.42373, 'abbr': 'san', 'name': 'San Antonio, TX', 'longitud
e': -98.49438, 'population': 1327407, 'label': '6'}, 7: {'latitude': 32.715786,
'abbr': 'sdg', 'name': 'San Diego, CA', 'longitude': -117.15834, 'population':
1307402, 'label': '7'}, 8: {'latitude': 32.803468, 'abbr': 'dal', 'name': 'Dall
as, TX', 'longitude': -96.769879, 'population': 1197816, 'label': '8'}, 9: {'la
titude': 37.339458, 'abbr': 'sjs', 'name': 'San Jose, CA', 'longitude': -121.89
5022, 'population': 945942, 'label': '9'}, 10: {'latitude': 30.332428, 'abbr':
'jkv', 'name': 'Jacksonville, FL', 'longitude': -81.656165, 'population': 82178
4, 'label': '10'}, 11: {'latitude': 39.768663, 'abbr': 'ind', 'name': 'Indianap
olis, IN', 'longitude': -86.159855, 'population': 820445, 'label': '11'}})
```

In [4]:
```
1 G.edges(data=True)
```

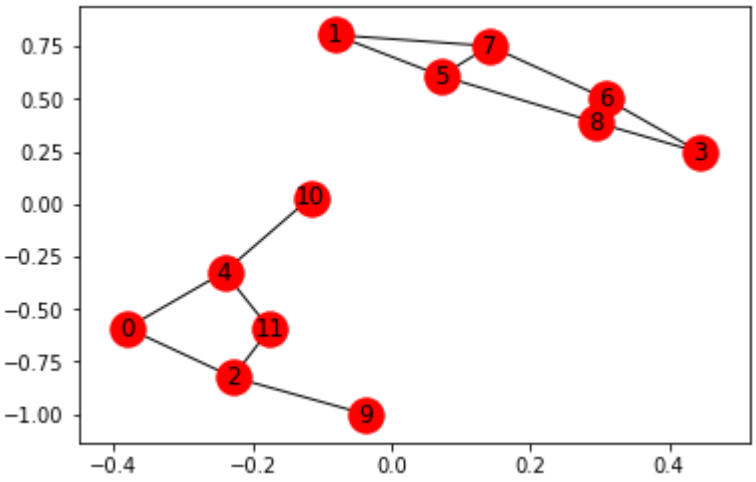
Out[4]:
```
EdgeDataView([(0, 2, {'distance': 1264, 'capacity': 100, 'id': '0'}), (0, 4,
{'distance': 152, 'capacity': 100, 'id': '1'}), (1, 9, {'distance': 544, 'capac
ity': 100, 'id': '2'}), (1, 5, {'distance': 432, 'capacity': 100, 'id': '3'}),
(1, 7, {'distance': 192, 'capacity': 100, 'id': '4'}), (2, 9, {'distance': 346
4, 'capacity': 100, 'id': '5'}), (2, 11, {'distance': 296, 'capacity': 100, 'i
d': '6'}), (3, 8, {'distance': 384, 'capacity': 100, 'id': '7'}), (3, 10, {'dis
tance': 1392, 'capacity': 100, 'id': '8'}), (3, 6, {'distance': 320, 'capacit
y': 100, 'id': '9'}), (4, 10, {'distance': 1360, 'capacity': 100, 'id': '10'}),
(4, 11, {'distance': 1032, 'capacity': 100, 'id': '11'}), (5, 8, {'distance': 1
704, 'capacity': 100, 'id': '12'}), (5, 7, {'distance': 568, 'capacity': 100,
'id': '13'}), (6, 8, {'distance': 440, 'capacity': 100, 'id': '14'}), (6, 7,
{'distance': 2040, 'capacity': 100, 'id': '15'}), (8, 11, {'distance': 1440, 'c
apacity': 100, 'id': '16'})])
```

In [5]:
```
edge_cut = [(10,3),(11,8),(9,1)]
G.remove_edges_from(edge_cut)

print ('connected?', nx.is_connected(G))
print ('# of components=', nx.number_connected_components(G))
S = nx.connected_component_subgraphs(G)
i = 1
for s in S:
    print ('subgraph #%s has %s nodes' % (i, s.number_of_nodes()))
    i += 1

nx.draw_networkx(G)
plt.show()
```

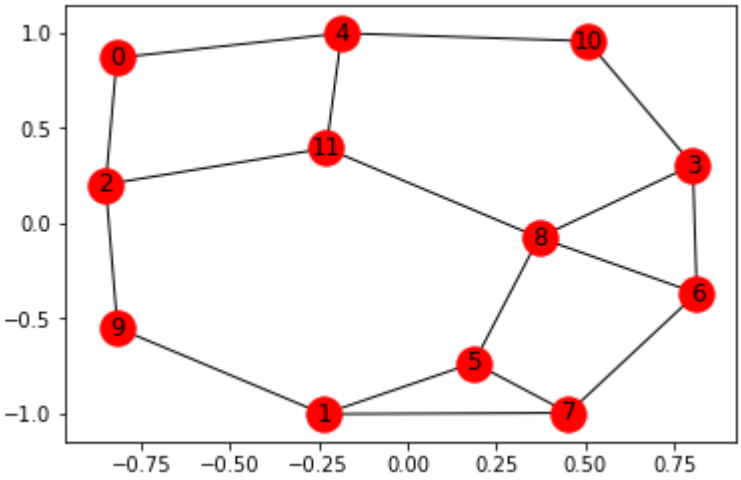
```
connected? False
# of components= 2
subgraph #1 has 6 nodes
subgraph #2 has 6 nodes
```



Persistent Node Placement

Compute the placement first.
Then, give the placement to drawing functions.

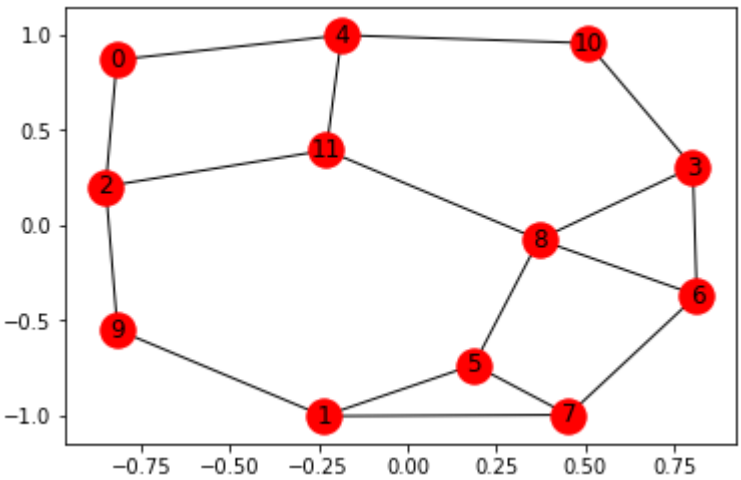
In [6]:
```
# plt.figure(1,figsize=(10,10))
G = nx.read_gexf('data/us-att.gexf', node_type=int)
layout = nx.spring_layout(G)
nx.draw_networkx(G, pos=layout)       # named parameter 'pos'
```



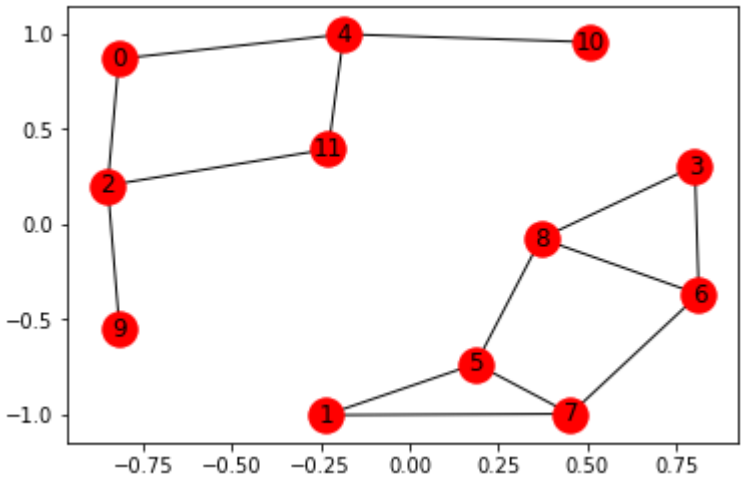
In [7]:
```
layout
```

Out[7]:
```
{0: array([-0.81792564,  0.86610593]),
 1: array([-0.23626672, -1.        ]),
 2: array([-0.84960475,  0.20304583]),
 3: array([0.80309624, 0.30510464]),
 4: array([-0.18510751,  0.9952655 ]),
 5: array([ 0.18672451, -0.73150706]),
 6: array([ 0.81425891, -0.36723768]),
 7: array([ 0.45335502, -0.99370693]),
 8: array([ 0.37009173, -0.07522968]),
 9: array([-0.81619845, -0.54862913]),
 10: array([0.50622161, 0.9539233 ]),
 11: array([-0.22864494,  0.39286527])}
```

In [8]:
```
nx.draw_networkx(G, pos=layout)
```



In [9]:
```
1 G.remove_edges_from(edge_cut)
2 nx.draw_networkx(G, pos=layout)
```



In [10]:
```
1 layout
```

Out[10]:
```
{0: array([-0.81792564,  0.86610593]),
 1: array([-0.23626672, -1.        ]),
 2: array([-0.84960475,  0.20304583]),
 3: array([0.80309624, 0.30510464]),
 4: array([-0.18510751,  0.9952655 ]),
 5: array([ 0.18672451, -0.73150706]),
 6: array([ 0.81425891, -0.36723768]),
 7: array([ 0.45335502, -0.99370693]),
 8: array([ 0.37009173, -0.07522968]),
 9: array([-0.81619845, -0.54862913]),
 10: array([0.50622161, 0.9539233 ]),
 11: array([-0.22864494,  0.39286527])}
```

Format of the layout data

The layout data is a dictionary with

- nodes as *keys* and
- positions as *values* in array format of numpy.

In [ ]:
```
1
```