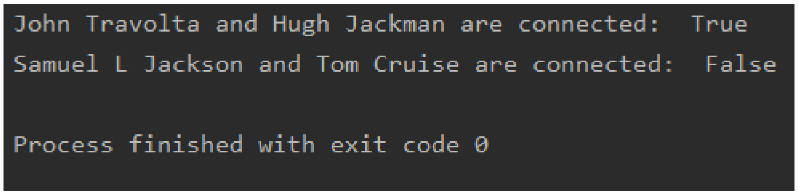
Andrew Breslauer, Coding Assignment A2: Degrees of Separation

```
Connections:
  Characters:
    Samuel L Jackson: SLJ
    John Travolta: JT
    Uma Thurman: UT
    Bruce Willis: BW
    Taron Egerton: TE
    Colin Firth: CF
    Hugh Jackman: HJ
    Christopher Walken: CW
    Sir Patrick Stewart: SPS
    Sir Ian McKellen: SIM
    George Clooney: GC
    Brad Pitt: BP
    Matt Damon: MD
    Franka Potente: FP
    Chris Cooper: CC
    Tom Cruise: TC
    Emmanuelle Beart: EB
    Henry Czerny: HC
  Movies:
    Pulp Fiction: SLJ, JT, UT, BW
    Kingsman: TE, SLJ, CF
    Eddie the Eagle: TE, HJ, CW
    X-Men: HJ, SPS, SIM
    Ocean's 11: GC, BP, MD
    Bourne Identity: MD, FP, CC
    Mission Impossible: TC, EB, HC
```

```
import pprint
from collections import defaultdict


class Graph(object):
    """ Graph data structure, undirected by default. """

    def __init__(self, connections, directed=False):
        self._graph = defaultdict(set)
        self._directed = directed
        self.add_connections(connections)

    def add_connections(self, connections):
        """ Add connections (list of tuple pairs) to graph """
        for node1, node2 in connections:
            self.add(node1, node2)

    def add(self, node1, node2):
        """ Add connection between node1 and node2 """
        self._graph[node1].add(node2)
        if not self._directed:
            self._graph[node2].add(node1)
    def remove(self, node):
        """ Remove all references to node """
        for n, cxns in self._graph.iteritems():
            try:
                cxns.remove(node)
            except KeyError:
                pass

            try:
                del self._graph[node]
            except KeyError:
                pass

    def is_connected(self, node1, node2):
        """ Is node1 directly connected to node2 """
        return node1 in self._graph and node2 in self._graph[node1]

    def find_path(self, node1, node2, path=[]):
        """ Find any path between node1 and node2 (may not be shortest) """
        path = path + [node1]
        if node1 == node2:
            return path
        if node1 not in self._graph:
            return None
        for node in self._graph[node1]:
            if node not in path:
                new_path = self.find_path(node, node2, path)
                if new_path:
                    return new_path
        return None

    def isConnected(self, node1, node2):
        if self.find_path(node1, node2) is not None:
            return True
        return False

    def __str__(self):
        return '{}({})'.format(self.__class__.__name__, dict(self._graph))


connections = [('SLJ', 'JT'), ('SLJ', 'UT'), ('SLJ', 'BW'), ('JT', 'UT'), ('JT',
'BW'), ('UT', 'BW'), ('TE', 'SLJ'),
               ('TE', 'CF'), ('SLJ', 'CF'), ('TE', 'HJ'), ('HJ', 'CW'), ('TE', 'CW'),
('HJ', 'SPS'), ('HJ', 'SIM'),
               ('SPS', 'SIM'), ('GC', 'BP'), ('GC', 'MD'), ('BP', 'MD'), ('MD',
'FP'), ('MD', 'CC'), ('FP', 'CC'),
               ('TC', 'EB'), ('TC', 'HC'), ('EB', 'HC')]

g = Graph(connections, directed=False)
print(g._graph)
print("John Travolta and Hugh Jackman are connected: ", g.isConnected('SLJ', 'HJ'))
print("Samuel L Jackson and Tom Cruise are connected: ", g.isConnected('SLJ', 'TC'))
```



```
John Travolta and Hugh Jackman are connected:  True
Samuel L Jackson and Tom Cruise are connected:  False

Process finished with exit code 0
```