In [1]:

```
#CITATION :- DATA SCIENCE FROM SCRATCH CHAPTER 1 EXAMPLES
users = [
{ "id": 0, "name": "Hero" },
{ "id": 1, "name": "Dunn" },
{ "id": 2, "name": "Sue" },
{ "id": 3, "name": "Chi" },
{ "id": 4, "name": "Thor" },
{ "id": 5, "name": "Clive" },
{ "id": 6, "name": "Hicks" },
{ "id": 7, "name": "Devin" },
{ "id": 8, "name": "Kate" },
{ "id": 9, "name": "Klein" }
]
```

In [2]:

```
friendships = [(0, 1), (0, 2), (1, 2), (1, 3), (2, 3), (3, 4),
(4, 5), (5, 6), (5, 7), (6, 8), (7, 8), (8, 9)]
```

In [5]:

```
for user in users:
    user["friends"] = []
for userid,friend in friendships:
    users[userid]["friends"].append(friend)
    users[friend]["friends"].append(userid)
```

In [6]:

```
user
```

Out[6]:

```
{'friends': [8], 'id': 9, 'name': 'Klein'}
```

In [7]:

```
users
```

Out[7]:

```
[{'friends': [1, 2], 'id': 0, 'name': 'Hero'},
 {'friends': [0, 2, 3], 'id': 1, 'name': 'Dunn'},
 {'friends': [0, 1, 3], 'id': 2, 'name': 'Sue'},
 {'friends': [1, 2, 4], 'id': 3, 'name': 'Chi'},
 {'friends': [3, 5], 'id': 4, 'name': 'Thor'},
 {'friends': [4, 6, 7], 'id': 5, 'name': 'Clive'},
 {'friends': [5, 8], 'id': 6, 'name': 'Hicks'},
 {'friends': [5, 8], 'id': 7, 'name': 'Devin'},
 {'friends': [6, 7, 9], 'id': 8, 'name': 'Kate'},
 {'friends': [8], 'id': 9, 'name': 'Klein'}]
```

In [9]:

```
def total_num_friends(user):
    return len(user["friends"])

total_num_conn = sum(total_num_friends(user) for user in users)
```

In [10]:

```
total_num_conn
```

Out[10]:

```
24
```

In [11]:

```
num_user = len(users)
```

In [12]:
```
avg_conn = total_num_conn/num_user
```

In [13]:
```
avg_conn
```

Out[13]:
```
2.4
```

In [15]:
```
user_list = [(user["id"], total_num_friends(user)) for user in users]
```

In [16]:
```
user_list
```

Out[16]:
```
[(0, 2),
 (1, 3),
 (2, 3),
 (3, 3),
 (4, 2),
 (5, 3),
 (6, 2),
 (7, 2),
 (8, 3),
 (9, 1)]
```

In [50]:
```
def friend_of_friend_bad(user):
    lst = list()
    for friend in user["friends"] :
        for fof in users[friend]["friends"]:
            lst.append(fof)
    return lst
```

In [52]:
```
friend_of_friend_bad(users[0])
```

Out[52]:
```
[0, 2, 3, 0, 1, 3]
```

In [43]:
```
users[2]
```

Out[43]:
```
{'friends': [0, 1, 3], 'id': 2, 'name': 'Sue'}
```

In [60]:
```
def friend_of_friend(user):
    lst = list()
    for friend in user["friends"] :
        for fof in users[friend]["friends"]:
            if not fof in user["friends"] and not fof == user["id"] and
not fof in lst:
                lst.append(fof)
    return lst
```

In [62]:
```
friend_of_friend(users[3])
```

Out[62]:
```
[0, 5]
```

In [82]:

```python
from collections import Counter
def not_the_same(user, other_user):
    return user["id"] != other_user["id"]

def not_friends(user, other_user):
    return all(not_the_same(friend, other_user)
               for friend in user["friends"])

def friends_of_friend_ids(user):
    return Counter(foaf["id"]
                   for friend in user["friends"]
                   # Probably wrong code in DSS CHAPTER 1 :-  frined["friends"] is  not working
                   for foaf in friend["friends"]
                   if not_the_same(user, foaf)
                   and not_friends(user, foaf))
```

```
In [83]: print (friends_of_friend_ids(users[3]))
```

```
-----------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-83-2b36273aaa21> in <module>()
----> 1 print (friends_of_friend_ids(users[3]))

<ipython-input-82-d51e563acd1c> in friends_of_friend_ids(user)
      9 def friends_of_friend_ids(user):
     10     return Counter(foaf["id"]
---> 11                    for friend in user["friends"] # for each of
 my friends
     12                    for foaf in friend["friends"] # count *their
* friends
     13                    if not_the_same(user, foaf) # who aren't me

/Users/abhisheknigam/anaconda/lib/python3.5/collections/__init__.py in
__init__(*args, **kwds)
    528             raise TypeError('expected at most 1 arguments, got
 %d' % len(args))
    529         super(Counter, self).__init__()
--> 530         self.update(*args, **kwds)
    531
    532     def __missing__(self, key):

/Users/abhisheknigam/anaconda/lib/python3.5/collections/__init__.py in
update(*args, **kwds)
    615                     super(Counter, self).update(iterable) # fas
t path when counter is empty
    616             else:
--> 617                 _count_elements(self, iterable)
    618         if kwds:
    619             self.update(kwds)

<ipython-input-82-d51e563acd1c> in <genexpr>(.0)
     10     return Counter(foaf["id"]
     11                    for friend in user["friends"] # for each of
 my friends
---> 12                    for foaf in friend["friends"] # count *their
* friends
     13                    if not_the_same(user, foaf) # who aren't me
     14                    and not_friends(user, foaf)) # and aren't my
 friends

TypeError: 'int' object is not subscriptable
```

In [66]:

```
interests = [
(0, "Hadoop"), (0, "Big Data"), (0, "HBase"), (0, "Java"),
(0, "Spark"), (0, "Storm"), (0, "Cassandra"),
(1, "NoSQL"), (1, "MongoDB"), (1, "Cassandra"), (1, "HBase"),
(1, "Postgres"), (2, "Python"), (2, "scikit-learn"), (2, "scipy"),
(2, "numpy"), (2, "statsmodels"), (2, "pandas"), (3, "R"), (3,
"Python"),
(3, "statistics"), (3, "regression"), (3, "probability"),
(4, "machine learning"), (4, "regression"), (4, "decision trees"),
(4, "libsvm"), (5, "Python"), (5, "R"), (5, "Java"), (5, "C++"),
(5, "Haskell"), (5, "programming languages"), (6, "statistics"),
(6, "probability"), (6, "mathematics"), (6, "theory"),
(7, "machine learning"), (7, "scikit-learn"), (7, "Mahout"),
(7, "neural networks"), (8, "neural networks"), (8, "deep learning"),
(8, "Big Data"), (8, "artificial intelligence"), (9, "Hadoop"),
(9, "Java"), (9, "MapReduce"), (9, "Big Data")
]
```

In [69]:

```
interestDict = dict();
for user, interest in interests:
    if interest in interestDict:
        interestDict[interest].append(user)
    else :
        interestDict[interest] = [user]
print (interestDict)
```

```
{'Hadoop': [0, 9], 'Big Data': [0, 8, 9], 'Haskell': [5], 'deep learnin
g': [8], 'scipy': [2], 'R': [3, 5], 'neural networks': [7, 8], 'Cassand
ra': [0, 1], 'MongoDB': [1], 'mathematics': [6], 'C++': [5], 'numpy':
 [2], 'probability': [3, 6], 'Postgres': [1], 'regression': [3, 4], 'Py
thon': [2, 3, 5], 'libsvm': [4], 'scikit-learn': [2, 7], 'decision tree
s': [4], 'statsmodels': [2], 'programming languages': [5], 'statistic
s': [3, 6], 'Storm': [0], 'pandas': [2], 'Spark': [0], 'Java': [0, 5,
 9], 'artificial intelligence': [8], 'HBase': [0, 1], 'NoSQL': [1], 'Ma
pReduce': [9], 'theory': [6], 'Mahout': [7], 'machine learning': [4,
 7]}
```

In [70]:

```
from collections import defaultdict
```

In [76]:

```
interest_by_userid = defaultdict(list)
for user, interest in interests:
    interest_by_userid[user].append(interest);
```

```
In [77]: interest_by_userid
```

```
Out[77]: defaultdict(list,
                     {0: ['Hadoop',
                       'Big Data',
                       'HBase',
                       'Java',
                       'Spark',
                       'Storm',
                       'Cassandra'],
                      1: ['NoSQL', 'MongoDB', 'Cassandra', 'HBase', 'Postgres'],
                      2: ['Python',
                       'scikit-learn',
                       'scipy',
                       'numpy',
                       'statsmodels',
                       'pandas'],
                      3: ['R', 'Python', 'statistics', 'regression', 'probabilit
         y'],
                      4: ['machine learning', 'regression', 'decision trees', 'l
         ibsvm'],
                      5: ['Python',
                       'R',
                       'Java',
                       'C++',
                       'Haskell',
                       'programming languages'],
                      6: ['statistics', 'probability', 'mathematics', 'theory'],
                      7: ['machine learning',
                       'scikit-learn',
                       'Mahout',
                       'neural networks'],
                      8: ['neural networks',
                       'deep learning',
                       'Big Data',
                       'artificial intelligence'],
                      9: ['Hadoop', 'Java', 'MapReduce', 'Big Data']})
```

```
In [84]: def data_scientists_who_like(target_interest):
             return interestDict[target_interest]
```

```
In [89]: def most_common_interest(user):
             return Counter(usr for interest in interest_by_userid[user["id"]]
                     for usr in interestDict[interest]
                     if not user["id"] == usr)

         print(most_common_interest(users[0]))
```

```
         Counter({9: 3, 1: 2, 8: 1, 5: 1})
```

```
In [90]: salaries_and_tenures = [(83000, 8.7), (88000, 8.1),
         (48000, 0.7), (76000, 6),
         (69000, 6.5), (76000, 7.5),
         (60000, 2.5), (83000, 10),
         (48000, 1.9), (63000, 4.2)]
```

In [91]:

```
salary_by_tenure = defaultdict(list)
for salary, tenure in salaries_and_tenures:
    salary_by_tenure[tenure].append(salary)
```

In [92]:

```
average_salary_by_tenure = {
    tenure : sum(salaries) / len(salaries)
    for tenure, salaries in salary_by_tenure.items()
}
```

In [93]:

```
average_salary_by_tenure
```

Out[93]:

```
{0.7: 48000.0,
 1.9: 48000.0,
 2.5: 60000.0,
 4.2: 63000.0,
 6: 76000.0,
 6.5: 69000.0,
 7.5: 76000.0,
 8.1: 88000.0,
 8.7: 83000.0,
 10: 83000.0}
```

In [96]:

```
def tenure_bucket(tenure):
    if tenure < 2:
        return "less than two"
    elif tenure < 5:
        return "between two and five"
    else:
        return "more than five"

salary_by_tenure_bucket = defaultdict(list)
for salary, tenure in salaries_and_tenures:
    bucket = tenure_bucket(tenure)
    salary_by_tenure_bucket[bucket].append(salary)

average_salary_by_bucket = {
    tenure_bucket : sum(salaries) / len(salaries)
    for tenure_bucket, salaries in salary_by_tenure_bucket.items()
}

average_salary_by_bucket
```

Out[96]:

```
{'between two and five': 61500.0,
 'less than two': 48000.0,
 'more than five': 79166.66666666667}
```

In [101]:

```
words_and_counts = Counter(num_users for userIds in interest_by_userid.k
eys()
        for num_users in interest_by_userid[userIds])

print (words_and_counts)
```

```
Counter({'Big Data': 3, 'Python': 3, 'Java': 3, 'Hadoop': 2, 'R': 2, 'n
eural networks': 2, 'Cassandra': 2, 'probability': 2, 'regression': 2,
 'scikit-learn': 2, 'statistics': 2, 'HBase': 2, 'machine learning': 2,
 'Haskell': 1, 'deep learning': 1, 'scipy': 1, 'MongoDB': 1, 'mathemati
cs': 1, 'C++': 1, 'numpy': 1, 'Postgres': 1, 'libsvm': 1, 'decision tre
es': 1, 'statsmodels': 1, 'programming languages': 1, 'Storm': 1, 'pand
as': 1, 'Spark': 1, 'artificial intelligence': 1, 'NoSQL': 1, 'MapReduc
e': 1, 'theory': 1, 'Mahout': 1})
```

In [ ]: