In [21]:
```
def create_database(filepath):
    """
    Write a function that takes in a csv filepath and creates a list of
    dictionaries, where each dictionary represents 1 observation of dat

    args:
        filepath(str): filepath to csv containing data
    returns:
        a list of dictionaries, each representing 1 observation.

    >>> create_database('data/datasource.csv')
    [{'name': 'a', 'quality': '4', 'source': 'dsc'},
 {'name': 'b', 'quality': '10', 'source': 'lign'},
 {'name': 'c', 'quality': '2', 'source': 'dsc'},
 {'name': 'd', 'quality': '5', 'source': 'dsc'}]
    """
    output = []
    with open(filepath, 'r') as f:
        column_names = f.readline().strip().split(',')
        for line in f:
            line = line.strip().split(',')
            output.append({column_names[i]:line[i] for i in range(len(
    return output
database = create_database('data/datasource.csv')
database
```

Out[21]:
```
[{'name': 'a', 'quality': '4', 'source': 'dsc'},
 {'name': 'b', 'quality': '10', 'source': 'lign'},
 {'name': 'c', 'quality': '2', 'source': 'dsc'},
 {'name': 'd', 'quality': '5', 'source': 'dsc'}]
```

In [30]:

```
def inject_value(filepath, new_entry):
    """
    Write a function that takes in a filepath and adds an entry
    to the data file.

    args:
        filepath(str): filepath to csv containing data to be updated
        new_entry(dict): dictionary of values in the same format to be

    >>> new_entry = {'name': 'e', 'quality': '12', 'source': 'cse'}
    >>> inject_value('data/datasource.csv', new_entry)
    >>> create_database('data/datasource.csv')
    [{'name': 'a', 'quality': '4', 'source': 'dsc'},
 {'name': 'b', 'quality': '10', 'source': 'lign'},
 {'name': 'c', 'quality': '2', 'source': 'dsc'},
 {'name': 'd', 'quality': '5', 'source': 'dsc'},
 {'name': 'e', 'quality': '12', 'source': 'cse'}]
    """
    injection_data = ','.join([x for x in new_entry.values()])
    with open(filepath, 'a') as f:
        f.write('\n')
        f.write(injection_data)
new_entry = {'name': 'e', 'quality': '12', 'source': 'cse'}
inject_value('data/datasource.csv', new_entry)
create_database('data/datasource.csv')
```

Out[30]:

```
[{'name': 'a', 'quality': '4', 'source': 'dsc'},
 {'name': 'b', 'quality': '10', 'source': 'lign'},
 {'name': 'c', 'quality': '2', 'source': 'dsc'},
```

```
{'name': 'd', 'quality': '5', 'source': 'dsc'},
{'name': 'e', 'quality': '12', 'source': 'cse'}]
```

In [9]:
```
def query_data_source(database, possible_sources):
    """
    Write a function that takes in a database and uses map/filter to
    return a list of sources within possible_sources.

    args:
        database(list): list of dictionaries representing a csv output
        possible_sources(list): list of strings that represent sources
    returns:
        a list of sources from database that exist inside possible_sour
    >>> database = create_database('data/datasource.csv')
    >>> query_data_source(database, ['lign', 'cse', 'mgt', 'econ'])
    ['cse', 'lign']
    """
    valid_entries = filter(lambda x: x['source'] in possible_sources, d
    return list(set(map(lambda x: x['source'],valid_entries)))
query_data_source(database, ['lign', 'cse', 'mgt', 'econ'])
```

Out[9]:
```
['cse', 'lign']
```

In [20]:

```
def query_data_quality_avg(database, min_quality):
    """
    Write a function that takes in a database and uses map/filter to
    return the average quality of entries above min_quality.

    args:
        database(list): list of dictionaries representing a csv output
        min_quality(int): minimum quality value
    returns:
        the average quality of entries above min_quality.
    >>> database = create_database('data/datasource.csv')
    >>> query_data_quality_avg(database, 4)
    9.0
    """
    valid_entries = filter(lambda x: int(x['quality']) > min_quality, d
    quality_values = list(map(lambda x: int(x['quality']),valid_entries
    return sum(quality_values) / len(quality_values)
query_data_quality_avg(database, 4)
```

Out[20]: 9.0