In general, there should exist tests to handle

- Small cases
- Normal cases
- Variations on normal cases
- Edge cases

1.

```
assert max_even([]) is None
assert max_even([2]) == 2
assert max_even([1]) is None
assert max_even([1, 2]) == 2
assert max_even([2, 1]) == 2
assert max_even([-2, 1, -4]) == -2
assert max_even([6, 4, 2]) == 6
assert max_even([2, 6, 4]) == 6
assert max_even([4, 2, 6]) == 6
```

2.

```
assert multiply(0, 0) == 0
assert multiply(1, 0) == 0
assert multiply(0, 1) == 0
assert multiply(1, 1) == 1
assert multiply(-1, 1) == -1
assert multiply(1, -1) == -1
assert multiply(-1, -1) == 1
assert multiply(3, 4) == 12
assert multiply(-3, 4) == -12
assert multiply(3, -4) == -12
assert multiply(3, 4) == 12
assert multiply(0.5, 1.5) == 0.75
assert multiply(2.33, -1) == -2.33
assert multiply(0.2, 1) == -0.33
assert multiply(0, 10.76) == 0
```

3.

```
assert mode([]) == None
assert mode([1]) == 1
assert mode([1, 1, 2]) == 1
assert mode([1, 2, 1]) == 1
assert mode([2, 1, 1]) == 1
assert(mode([1, 1, 2, 2]) == 1 or mode([1, 1, 2, 2]) == 2)
assert mode([1, 2, 3, 4, 1, 2, 3, 3]) == 3
```

4.
Overall purpose of our function:
To iterate through a given list of dictionaries and calculate the average population for each states with the same number of electors
Inputs/outputs:
The function needs to work with some data to do the necessary calculations.
For this example, we would want to take a list of dictionaries as an input. Specifically, dictionaries with the following keys: State,Name,Electors,Population

Assume: we are given a dictionary with the following keys: State, Name, Electors, Population
Function name/docstring:
name: calculate_avg_population_per_numb_electors(data)
docstring: Given a list of election data, returns a dictionary mapping the

number of electors to the average population for a state with that many electors.

Tests based on function:
(Test cases will vary from person to person, a few examples are shown below)

```
assert calculate_avg_population_per_numb_electors([]) == {}
test_case = [{"State": "WA", "Name":"Washington", "Electors": 9, "Population": 6000000}]
second_test = [{"State": "WA", "Name":"Washington", "Electors": 9, "Population": 6000000},
{"State": "CA", "Name":"California", "Electors": 9, "Population": 8000000}]
assert calculate_avg_population_per_numb_electors(test_case) == {9:6000000}
assert calculate_avg_population_per_numb_electors(second_test) == {9:7000000}
```

Decide on an implementation:
One implementation is shown below:

```
def calculate_avg_population_per_numb_electors(data):
    internal_dict = {}
    for entries in data:
        electors = entries['Electors']
        if electors in internal_dict.keys():
            internal_dict[electors].append(entries['Population'])
        else:
            internal_dict[electors] = [entries['Population']]
    output_dict = {}
    for values in internal_dict:
        output_dict[values] = sum(internal_dict[values])/len(internal_dict[values])
    return output_dict
print(calculate_avg_population_per_numb_electors(read_csv("2012-electoral-college.csv")))
```