**Step 1: Get an informal list of the responsibilities of your objects.**

The following responsibilities are mentioned in the problem statement:

*name*

*population*

*area*

There is a hidden responsibility as well. Need to be able to find out the population of the list of countries.

*get population density.*

**Step 2: Specify the public interface.**

We need to supply parameter variables and determine which methods are accessors and mutators.

To get the country with the largest population

*def largest\_population(countries):*

To get the country with the largest area

*def largest\_area(countries):*

To get the country with the largest area from a list of countries.

Finally, we have

*def get\_density(self):*

*def largest\_adensity(countries):*

Now, we move on to the constructor. The constructor should accept name, population and area of the country.

Here is the complete public interface:

* Constructor

*Def \_\_init\_\_(self, name, population, erea):*

* Mutators

*def largest\_population(countries):*

*def largest\_area(countries):*

* Accessors

*def get\_density(self):*

*def largest\_adensity(countries):*

**Step 3: Document the public interface.**

## A country has name, population, and area can be computed the population density.

#

class Country:

## Constructs a country with name, population, and area.

# @param name the name of the country

# @param population the population of the country

# @param area the area of the country

#

*def \_\_init\_\_(self, name, population, area):*

## Gets the country with the largest population from a list of countries.

# @param countries a list of Country

# @return the Country with the largest population

#

*def largest\_population(countries):*

## Gets the country with the largest area from a list of countries.

# @param countries a list of Country

# @return the Country with the largest area

#

*def largest\_area(countries):*

## Gets the population density of the country.

# @return the population density

#

*def get\_density(self):*

## Gets the country with the largest population density from a list of countries.

# @param countries a list of Country

# @return the Country with the largest population density

#

*def largest\_density(countries):*

**Step 4: Determine instance variables.**

self.name = name

self.population = population

self.area = area

**Step 5: Implement the constructor and methods.**

*def \_\_init\_\_(self, name, population, area):*

*self.name = name*

*self.population = population*

*self.area = area*

**Step 6: Implement the methods.**

*def largest\_by\_population(countries):*

return max(countries, key=attrgetter('population'))

*def largest\_area(countries):*

return max(countries, key=attrgetter('area'))

*def get\_density(self):*

return self.population / self.area

*def largest\_density(countries):*

return max(countries, key=methodcaller('get\_density'))

**Step 7: Test your class.**

Here is a tester program that exercises all methods.

parser = argparse.ArgumentParser(description="Process some countries.")

parser.add\_argument("--country", action="append", required=True, help="Country name.")

parser.add\_argument("--population", action="append", type=int, required=True, help="Population of the country.")

parser.add\_argument("--area", action="append", type=float, required=True, help="Area of the country in square kilometers.")

args = parser.parse\_args()

countries = [Country(name, pop, area) for name, pop, area in zip(args.country, args.population, args.area)]

print(f"Largest Area: {Country.largest\_area(countries).name}")

print(f"Largest Population: {Country.largest\_population(countries).name}")

print(f"Largest Density: {Country.largest\_density(countries).name}")

**Program Run**

Largest Area: AAA

Largest Population: BBB

Largest Density: BBB