```
##
# Assignment 4. (Country Classes)
# Author: Patrick Mihalcea. Student number: 251023246
# The purpose of this code is to use classes to store, search,
sort, remove, and filter country data.

from country import Country

class CountryCatalogue():
     """Representation of a catalogue of 'Country' objects."""

     def __init__(self, data, continent):
          """Initialize attributes of the catalogue of
countries."""
          # Start by reading in the two files of data and storing
them.
          self._data  = open(data,"r",encoding="utf8")
          # Creating cDictionary.
          self._cDictionary = {}
          self._continentLines = self._continent.readlines()
          for line in self._continentLines[1:]: # Skip header
line.
               line = line.split(',')
               self._cDictionary[line[0]] =
line[1].strip().title() # Clean up string and format as title.
          # Creating countryCat
          self._countryCat = {}
          self._dataLines = self._data.readlines()
          for line in self._dataLines[1:]: # Skip header line.
               line = line.split('|')
               # Clean up the floating values to rid of commas.
               line[1] = line[1].replace(',', '')
               line[2] = line[2].replace(',','')
               # Add every object to the dictionary
               self._countryCat[line[0]] = Country(line[0],
int(line[1]), float(line[2].strip()), self._cDictionary[line[0]])

     def findCountry(self, country):
          """Find a country in the catalogue."""
          if country.title() in self._countryCat.keys():
               result = self._countryCat[country]
               # Return country object.
               return result
          else:
               return None
```

```
    def setPopulationOfCountry(self, country, population):
        """Set the population of a country in the catalogue."""
        # Format input of country for proper reference.
        country = country.title()
        if country in self._countryCat.keys():

self._countryCat[country].setPopulation(population)
            return True
        else:
            return False

    def setAreaOfCountry(self, country, area):
        """Set the area of a country in the catalogue."""
        country = country.title()
        if country in self._countryCat.keys():
            self._countryCat[country].setArea(area)
            return True
        else:
            return False

    def addCountry(self, country, population, area, continent):
        """Add a new country to the dictionary and catalogue of
countries."""
        # Check if the country already exists in the dictionary
and catalogue.
        country = country.title()
        if country not in self._cDictionary.keys():
            # Add the country.
            continent = continent.title()
            self._cDictionary[country] = continent
            self._countryCat[country] = Country(country,
int(population), float(area), continent)
            return True
        else:
            return False

    def deleteCountry(self, country):
        """Delete an existing country from the dictionary and
catalogue."""
        country = country.title()
        if country in self._cDictionary.keys():
            del self._cDictionary[country]
            del self._countryCat[country]
    def printCountryCatalogue(self):
        """Print the entire catalogue."""
        for country in self._countryCat.keys():
            print(self._countryCat[country])

    def getCountriesByContinent(self, continent):
        """Return a list of countries in the catalogue that are 
in a certain continent."""
        continent = continent.title()
        result = []
        for country in self._countryCat.keys():
            # Check the continent of each country in the 
catalogue.
            if self._countryCat[country].getContinent() == 
continent:
                # Add country to list if in desired 
continent.
                result.append(self._countryCat[country])
        return result

    def getCountriesByPopulation(self, continent = ""):
        """Return a list of countries in descending order of 
population."""
        result = []
        if continent != "":
            selected_countries = 
countries.getCountriesByContinent(continent)
            for country in selected_countries:
                result.append((country.getName(), 
country.getPopulation()))
        else:
            for country in self._countryCat.keys():

    result.append((self._countryCat[country].getName(), 
self._countryCat[country].getPopulation()))
        # Organize countries in descending order of population.
        result.sort(key=lambda tup: tup[1], reverse=True)
        return result

    def getCountriesByArea(self, continent = ""):
        """Return a list of countries in descending order of 
area."""
        result = []
        if continent != "":
            selected_countries = 
countries.getCountriesByContinent(continent)
               for country in selected_countries:
                    result.append((country.getName(),
country.getArea()))
          else:
               for country in self._countryCat.keys():

     result.append((self._countryCat[country].getName(),
self._countryCat[country].getArea()))
          # Organize countries in descending order of area.
          result.sort(key=lambda tup: tup[1], reverse=True)
          return result

     def findMostPopulousContinent(self):
          """Retrun a tuple of a the most populous continent and
its population."""
          # Make a dictionary of continents and their total
populations.
          continents = {}
          for country in self._countryCat.keys():
               continent =
self._countryCat[country].getContinent()
               # Add each new continent to the dictionary.
               if continent not in continents.keys():
                    continents[continent] =
self._countryCat[country].getPopulation()
               # Keep running total population of every country
in each continent.
               elif continent in continents.keys():
                    continents[continent] = continents[continent]
+ self._countryCat[country].getPopulation()
          # Creates list of (key, value) pairs.
          results = sorted(continents.items(), key=lambda x:
x[1])
          # Take result with greatest value.
          result = results[-1]
          return result

     def filterCountriesByPopDensity(self, low, high):
          """Return a list of (country, pop. density) pairs that
fall within a given range."""
          result = []
          for country in self._countryCat.keys():
               popDensity =
self._countryCat[country].getPopulation()/self._countryCat[countr
y].getArea()
                if popDensity >= low and popDensity <= high:

     result.append((self._countryCat[country].getName(),
popDensity))
          # Organize countries in descending order of pop.
density.
          result.sort(key=lambda tup: tup[1], reverse=True)

          return result

     def saveCountryCatalogue(self, filename):
          """Write the catalogue to a file alphabetically by
country."""
          # Open a file to save catalogue to. Check that it
exists.
          file_object = open(filename, 'w')
          # Make a list of the countries sorted alphabetically.
          countryList = []
          for country in self._countryCat.keys():
               countryList.append(country)
          countryList = sorted(countryList)
          # Make item counter to count items written.
          count = 0
          # Write the catalogue to the file.
          for country in countryList:
               countryName = self._countryCat[country].getName()
               population =
self._countryCat[country].getPopulation()
               area = round(self._countryCat[country].getArea(),
2)
               continent =
self._countryCat[country].getContinent()
               popDensity = round(population/area, 2)


file_object.write(countryName+'|'+continent+'|'+str(population)
+'|'+str(area)+'|'+str(popDensity)+"\n")
               count += 1
          # Close the file and return count.
          file_object.close()
          return count
```