CS1026: Assignment 4

Weight: 14%

Learning Outcomes:
By completing this assignment, you will gain skills relating to

- Strings and text files
- Writing your own classes
- Testing code
- Using complex data structures (i.e., sets and dictionaries)

In this assignment you will create a complete program that uses classes to store, search, sort, remove, and filter country data. The two major tasks are outlined below.

1. **Implement a class Country**
   - Instance Variables:
     i. Name: string
     ii. Population: integer
     iii. Area: float
     iv. Continent: string
   - Methods:
     i. constructor
     ii. Getter Methods: getName, getPopulation, getArea, getContinent,
     iii. getPopDensity: This calculates and returns the population density for the country. Population density is the population divided by the area.
     iv. Setter Methods: setPopulation
     v. **def** __repr__(self): generate a string representation for the class

        Name in Continent with a population density of value
        e.g China is in Asia with a population density of 4.56

```
class Country :
  def __init__(self, name, pop, area, continent) :
  def __repr__(self):
  def setPopulation(self, pop):
  def getName(self) :
  def getArea(self) :
  def getPopulation(self) :
  def getContinent(self):
  def getPopDensity(self) :
```

Test all your classes and methods before moving to the next section. Feel free to create other helper methods if necessary.

2. Implement a class called **CountryCatalogue**
   This class has two instance variable **catalogue (***that is a set or dictionary or list of countries***)**

and **cDictionary** (a dictionary)
This class will have the following methods.

- **Constructor:** this method will open the specified file and then first create the cDictionary and then create countries and add them to the second data constructor.
  Major Steps
  1. **Fill the Dictionary:** open the continent.txt file and fill cDictionary. The key is the country name, while the continent is the value.
  2. **Fill the Catalogue:** Open the file (the file name is passed into the constructor). Then read each line of the file and from that create a country and add it catalogue.

  A sample data file has been included data.txt *(Note that both files have headers)*

- **addCountry:** Give the user the option to add a new country to the set.
  Using user input the user must specify the name, population, area and continent. If the country they entered already exists, let them know they can't add a country already in the catalogue and keep on prompting for input until a unique name has been entered. In addition to adding the newly created country to the catalogue, make sure you add the continent to the cDictionary. Include a confirmation message to let the user know whether the operation was successful or not.

- **deleteCountry:** allows the user to enter a specific countryName, if this country exists then it should be deleted from the **catalogue** instance variable. Include a confirmation message to let the user know whether or not it was deleted.

- **findCountry:** allows the user to enter a specific countryName, if this country exists then print all of the country's information to the output screen. If the country does not exist send a message as well.

- **filterCountriesByContinent:** allow the user to enter a specific continent (assume valid continent), print to the screen all the countries (just their names) in the catalogue that are in that continent.

- **printCountryCatalogue:** print the whole catalogue to the screen, one country object as a time using the overloaded repr method for the Country class.

- **setPopulationOfASelectedCountry**: Ask the user for a country name and new population and then set the population of the country (if it is in the catalogue) to the value. Print the new population density value for that country to the screen (don't worry about units).

- **findCountryWithLargestPop**: find and display the name of the country with the largest population to the screen

- **findCountryWithSmallestArea:** find and display the name of the country with the smallest area to the screen.

- **filterCountriesByPopDensity:** ask the user to enter the lower bound and upper bound for a population density range and then find all countries that have a population density that falls within the range. Assume valid input of integers.

- **findMostPopulousContinent**: find and display the name of the continent with the most number of people living in it. Also display the number of people living in the continent.

- **saveCountryCatalogue:** allow the user to save all the countries to a file. Countries should be sorted alphabetically by name prior to saving. Population density should be rounded up to two decimal places.

  Format:
  Name|Continent|Population|PopulationDensity
  For example: `China|Asia|1200000000|14.56`

  After saving all the countries print a confirmatory message to the screen to let the user know that it was saved to the specified file.

```
class CountryCatalogue:
  def __init__(self, filename):
  def filterCountriesByContinent(self):
  def printCountryCatalogue(self):
  def findCountry(self):
  def deleteCountry(self):
  def addCountry(self):
  def setPopulationOfASelectedCountry(self):
  def saveCountryCatalogue(self, filename):
  def findCountryWithLargestPop(self):
  def findCountryWithSmallestArea(self):
  def findMostPopulousContinent(self):
  def filterCountriesByPopDensity(self):
```

Test your countryCatalogueClass to make sure all functionality is working.

Use the main.py file provided as the interface to your program. The TA will use a similar main.py to grade your assignment. DO NOT EDIT THIS FILE AT ALL

User input means that you should get input from the user. This is not passed in as function parameters. For output to both the console and file you do not need to format numbers to include commas. (e.g 1000 does NOT need to be written as 1,000). ***You may assume all the data is correct*** and there are no errors relating to the data file *(so don't worry about Exceptions or validating input except where otherwise noted for this assignment).*

Non-functional Specifications:

1. Include brief comments in your code identifying yourself, describing the program, and describing key portions of the code.
2. Assignments are to be done individually and must be your own work. Software may be used to detect cheating.
3. Use Python coding conventions and good programming techniques, for example:
   i. Meaningful variable names
   ii. Conventions for naming variables and constants
   iii. Use of constants where appropriate
   iv. Readability: indentation, white space, consistency

Submit the file in which your classes are defined. The name of the file should be **countryGalore.py**. Make sure you attach your python file to your assignment; DO NOT put the code inline in the textbox.

Make sure that you develop your code with Python 3.5 as the interpreter. TAs will not endeavor to fix code that uses earlier versions of Python.

What You Will Be Marked On:

1. Functional specifications:
   - Does the program behave according to specifications?
   - Does it run with the main program provided?
   - Are your classes created properly?
   - Are you using appropriate data structures?
   - Is the output according to specifications?
2. Non-functional specifications: as described above
3. Assignment submission: via OWL assignment submission