CIS 41B - Lab assignment 1: iterables, callables

Write a program that works with a list of countries of the world. The program reads each country data from a file and lets the user search for countries based on their data.

**Input file**

The input text file is lab1in.csv (It is a subset of: [www.statvision.com/webinars/countries%20of%20the%20world.xls)](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=2ahUKEwiVjtG4vbzhAhXZHTQIHeqMAOkQFjABegQIAhAC&url=http%3A%2F%2Fwww.statvision.com%2Fwebinars%2Fcountries%2520of%2520the%2520world.xls&usg=AOvVaw0OZrBrPVcFVi9Zhcw9tJ-l)

Each line of the file is one country record, with 4 comma separated fields. The fields are:

country name, continent, population density (per sq. mi.), literacy rate (% population)

Example of one line of the input file: Argentina ,LATIN AMER. & CARIB ,14.4,97.1

There are irregularities in some or all lines of the file that your program should handle:

* All country names have an extra space at the end, make sure to remove the space since it's not part of the name.
* All continent names have multiple extra spaces at the end, make sure to remove these spaces
* Some country names are double quoted and contain a comma, such as "Bahamas, The"  
  Change the comma to a semicolon and remove the double quotes from the name, such as: Bahamas; The
* Some countries don't have literacy rate data. Don't let it be an error in your program.

**Program requirements**

The application is made of 2 source files: lab1.py and country.py

It is recommended that you write code for the green portion first for both files. The green portion is the base code for the lab. Then write the black portion second, it is the extra features that should be added only after the base code works.

country.py

Contains the Country class, which holds information for one country, and has:

* A constructor:
  + Accepts *one line of the input file*, parses (separates) the line into 4 data values and stores them in 4 instance variables. You should do the parsing *without having to use loops*.
  + Make sure to fix the 4 irregularities in the line that are described in the input file section above: space at the end of country name and continent name, double quotes and comma in the name, missing literacy rate.
* A method such that when print(countryObj) is called, the country name is printed.
* Any other access methods as needed.

lab1.py

This file imports the Country class in order to work with it, but lab1.py is *not an OO program*.

lab1.py is a review of procedural programming, and country.py is a review of OO programming.

The file has the following functions.

1. A getData function that will:
   * Accept a filename or use the default filename lab1in.csv
   * If the file open is not successful, print a descriptive error message and end the program
   * Read each line from the input file and create a Country object
   * Store all the Country objects in a list. You should not have to use a Python loop to create the list.
   * Print the number of countries that are read in.
2. A printAll function:

* Loop to print a counting number and the name of each Country object on one line.
* The number is from a counting number sequence: 1 for 1st Country object, 2 for 2nd Country object, etc.
* To print the name of the Country object, only use the Python print function (you should not have to use a Country method, and definitely do not access the Country object's name directly)

1. A getChoice function that will get a choice from the user. The function will:

* Print a menu of 3 choices: l. print countries in descending order of literacy rate

d. print countries based on population density

q. quit

* Keep prompting the user until there is a valid choice and then return the choice.
* +1pt EC: use one loop condition and don't use a break statement or an if statement  
  +1pt EC: the one loop condition doesn't use 'and' or 'or' operator

1. A population density function that will:

* Calculate the population density of each part of the world
* There are 7 parts of the world or continents in the input file: Africa, Asia, Europe, Near East, North America, Oceania, South America.
* To calculate the population density of a continent, average out all the population density of the countries in that continent.
* Print the continent name and the population density in column format, with 1 digit after the decimal point for the density.
* Return the highest and lowest population density values.

1. A generator that will:

* Yield one list of Country objects at a time, in descending range of literacy rate.
* The ranges of literacy rates are: above 90 to 100, above 80 to 90, above 70 to 80, above 60 to 70... and ending with "no literacy data" range.
* Use the sorted function to sort the list of Country objects in descending literacy rate. No need to write your own sort function.
* You should not have to create any list other than the list of Country objects that is yielded.

1. A literacy rate function that will:

* Loop to let the user press the Enter key to get a list of countries within a descending literacy rate.
* The user enters any other key to stop the loop, or the loop ends with "no more data" when all countries have been printed. (The range 0-10 will have no data, and the countries with no literacy data are not printed)
* Each time the user presses the Enter key, use the generator to get a list of Country objects. Then loop to print each country name and corresponding literacy rate on one line, with a % sign after the literacy rate.

Note that each time the literacy rate function is called and the user presses the Enter key, the descending range continues from the previous time that the function runs.  
Example:   
1. The user runs the literacy rate function for the first time and presses Enter twice. This means the countries within range 90-100 and 80-90 are printed. Then the user quits the function and goes back to the main menu.

2. Later in the same test run, the user runs the literacy rate function again and presses Enter. This means the countries within range 70-80 are printed. This range is a continuation of the previous run because the generator is keeping track of where it is in the sequence of sorted Countries.

1. A decorator called 'retVal' that will:

* Print the return value of the function that it decorates.
* Use this decorator with the population density function so that the highest and lowest population densities (from the return value) are printed.

1. A main function that will:

* Call the getData function
* Call the printAll function
* Create the generator that yields one list of Country objects at a time, in descending range of literacy rate.
* Loop to call the getChoice function and, *without using an if statement*, call the literacy rate or population density function based on the user choice. The loop ends when the user selects 'q'.

**Documentation requirements**

* At the top of each file, put your name and one line description of the code in the file.
* Add a docstring (short one line is okay) to describe each function and public method.

[Note: lab assignments in this class are about implementing advanced concepts and problem solving, so they do take some thinking and some time, but the code itself is generally not very long. My country.py file is around 20 lines of code and my lab1.py is around 80 lines of code (not counting comments and blank lines, but counting import lines). The line counts are to give you a general idea so you can estimate your work, you do not need to match it.]

**Sample output** (user input is shown in green)

The sample output is to show how your output should look like and for you to check your work. You can use your own wording for text strings such as the prompts, no need to duplicate my wording.

Read in 227 countries # print output of getData()

1 Afghanistan # print output of printAll()

2 Albania

3 Algeria

4 American Samoa

<cut to save space>

224 Western Sahara

225 Yemen

226 Zambia

227 Zimbabwe

=========================

l: literacy rate

d: population density

q: quit

Enter your choice: l

Press Enter to see countries and literacy rates, anything else to quit: # Enter key

Andorra: 100.0%

Australia: 100.0%

Denmark: 100.0%

< cut to save space>

Peru: 90.9%

Zimbabwe: 90.7%

Vietnam: 90.3%

Press Enter for next names, anything else to quit: # Enter key

Guadeloupe: 90.0%

Bahrain: 89.1%

Antigua & Barbuda: 89.0%

<cut to save space>

Swaziland: 81.6%

Zambia: 80.6%

El Salvador: 80.2%

Press Enter for next names, anything else to quit: # Enter key

Botswana: 79.8%

Iran: 79.4%

Sao Tome & Principe: 79.3%

<cut to save space>

Tunisia: 74.2%

Guatemala: 70.6%

Rwanda: 70.4%

Press Enter for next names, anything else to quit: a

=========================

l: literacy rate

d: population density

q: quit

Enter your choice: d

Africa 86.7

Asia 1264.8

Europe 952.0

Near East 205.7

North America 260.9

Oceania 131.2

South America 136.2

(16271.5, 0.0) # This tuple is printed by the decorator

=========================

l: literacy rate

d: population density

q: quit

Enter your choice: l

Press Enter to see countries and literacy rates, anything else to quit: # Enter key

Algeria: 70.0% # Note that the generator continues with the next descending range

Uganda: 69.9%

Cambodia: 69.4%

<cut to save space>

Malawi: 62.7%

Sudan: 61.1%

Togo: 60.9%

Press Enter for next name, anything else to quit: d

=========================

l: literacy rate

d: population density

q: quit

Enter your choice: b # Invalid input

Enter your choice: q