# SI 330 - Homework 1: Python Basics with Delimited Text Files

## Due Date: Friday, Jan 13, 2017 @ 11:59pm

**Objectives:**

- Get experience with basic Python programming constructs
- Practice manipulation of real datasets via delimited text files

**Submission Instructions:**

After completing this homework, you will turn in **two** files via Canvas -> Assignments.

1. Your Python script, named `si330-hw1_`***`YOUR_UNIQUE_NAME`***`.py`.
2. Your output file, named `world_bank_output_`***`YOUR_UNIQUE_NAME`***`.csv`, which must be a text file in Comma Separated Values (CSV) format. **This should be the file produced by your Python script.**
3. A plain text file with answers to the written questions in 5b, named `si330-hw1_`***`YOUR_UNIQUE_NAME`***`.txt`.

---

This homework involves multiple steps. You should review all steps before starting, to think about how you're going to structure your code that can be used for both the earlier and later steps. You should then complete the steps one by one and verify that you've completed each correctly before moving on to the next one.

**Note: To get started, you can use the code example provided in Lab 1 that uses the csv module and the DictReader and DictWriter classes to read the input file and write the output file. You can do Steps 2, 3, and 4, by adding some additional code to this initial script.**

To check your final results, a sample file with the first 100 rows of desired output is provided in the `world_bank_output_result_first100.csv` file included in the .zip file.

**1. Download Datasets**

Download the following files from the folder `Canvas->Files->Datasets`:

`world_bank_country_data.txt`

`world_bank_regions.txt`

`world_bank_output_results_first100.csv`

The `.txt` files are in tab-separated (also known as tab-delimited) format, which means that the tab character (ASCII code 09) is used to separate the column data for each row. This is a standard exchange format for simple tabular data. For each file, the first row is a header row that contains the column names, but is not actually part of the data to be used/analyzed.

The file `world_bank_country_data.txt` is a summary dataset from the World Bank in tab-delimited format, containing 18 'indicators' (statistics) from over 200 countries, collected for each year in the period 2000-2010. (This is a subset of a larger dataset with 331 indicators spanning 1960-present: more information about each indicator and those not included here can be found at: http://data.worldbank.org/indicator (Links to an external site.) ) The file's first line is the header row containing the names of the various fields.

A. The file `world_bank_country_data.txt` has 20 columns per line, which are:

| *Column Number* | *Column Name* |
|---|---|
| 0 | Country Name |
| 1 | Date |
| 2 | Transit: Railways, (million passenger-km) |
| 3 | Transit: Passenger cars (per 1,000 people) |
| 4 | Business: Mobile phone subscribers |
| 5 | Business: Internet users (per 100 people) |
| 6 | Health: Mortality, under-5 (per 1,000 live births) |
| 7 | Health: Health expenditure per capita (current US$) |

| | |
|---|---|
| 8 | Health: Health expenditure, total (% GDP) |
| 9 | Population: Total (count) |
| 10 | Population: Urban (count) |
| 11 | Population:: Birth rate, crude (per 1,000) |
| 12 | Health: Life expectancy at birth, female (years) |
| 13 | Health: Life expectancy at birth, male (years) |
| 14 | Health: Life expectancy at birth, total (years) |
| 15 | Population: Ages 0-14 (% of total) |
| 16 | Population: Ages 15-64 (% of total) |
| 17 | Population: Ages 65+ (% of total) |
| 18 | Finance: GDP (current US$) |
| 19 | Finance: GDP per capita (current US$) |

B. The `world_bank_regions.txt` file gives the names of the region and subregion that each country belongs to, and has 3 columns per line:

| *Column Number* | *Column Name* |
|---|---|
| 0 | Region |
| 1 | Subregion |
| 2 | Country Name |

Your code will load these two files and do some manipulation on them to get the desired output.

## 2. Add a new "Mobile users per capita" column

Add code to compute and add a new column to the output file called "Mobile users per capita". This will take several (small) steps, for each row:

(a) Compute mobile users per capita as a floating point number by dividing the total mobile subscribers for a country (the "Business: Mobile phone subscribers" column in the input file) by its total population (the "Population: Total (count)" column in the input file). Note that to do this, you'll first have to convert those two input fields, which are strings, to floats. This will take two steps:

- Use the replace method for strings to remove commas and double-quote characters from the input strings
- Use float(.) to convert the resulting strings to floating point numbers
- Compute mobile users per capita for that row

(b) Next, for output purposes you will convert the resulting per capita statistic back to a string that is formatted such that there are exactly 2 digits after the decimal point (e.g. 0.04). Read about format specifications at http://docs.python.org/3/library/string.html#formatspec (Links to an external site.)
The output "row" object is a dictionary, so to add the value for the new column, store the formatted string into the output "row" object by setting the appropriate key-value pair in the row variable.

(c) Next, one convention in data formatting is to use "NA" for the value of a field if the data for that field is missing. If the "Business: Mobile phone subscribers" field is empty in the input file, set the Mobile users per capita in the output file to "NA".

(d) Last, you'll see in the lab sample code that the csv DictWriter class takes an initialization argument called `fieldnames` that is a list of strings that define the names of the columns to be written to the output file. (The `extrasaction = 'ignore'` argument tells the DictWriter class to ignore and not write out any columns in a row that are not mentioned in the `fieldnames=` list.) Add the new column name "Mobile users per capita" to the fieldnames list, inserting after Country Name, so that the DictWriter adds that value to the output csv file.

## 3. Add a new "Region" column

Next, we're going to add another new column to the output file called "Region" that has the global region associated with each row's country. We'll get the mapping from country to region using that other `world_bank_regions.txt` file you downloaded.

(a) First, write a function that takes a filename as its input parameter, and returns a dictionary that maps a country name (the dictionary key)  to its corresponding region (the key's value). The function will look something like this:

```
def read_region_file(filename):
   region_mapping = dict()  # create an empty dictionary
   with open(filename, 'r', newline='') as input_file:
       region_reader = csv.DictReader(input_file, delimiter='\t', quotechar ='"')
       for row in region_reader:
            # ADD YOUR CODE here to get the "Region" and "Country" input columns
            # and set the appropriate key-value pair in the region_mapping dictionary
            # for each input row
   return region_mapping
```

(b) Then, add a line to your main() function in the appropriate place that calls the `read_region_file` function, passing in the region file's name and saving the resulting dictionary in a variable.

(c) With this dictionary, add code in the loop that reads the input file rows to set the "Region" column's value in the row variable. Again, just as with the new "mobile users per capita" field, since the output "row" object is a dictionary, to add the value for the new column, you store the formatted string into the output "row" object by setting an appropriate key-value pair in the row variable..

(d) Add code to handle the case where if a country does not have an entry in the regions file, then set the region name to "NA". (There are a handful: for example, the Channel Islands)

(e) Just as you did with new "mobile users per capita" field above, you need to add the new column name "Region" to the `fieldnames` list so that the DictWriter adds that value to the output csv file.  Insert the "Region" string in the `fieldnames` list before "Country Name".

(f) Make any other changes to the order of the column names in the `fieldnames` list so that the output columns are, in order:

```
Region
Country Name
Mobile users per capita
Population
Year
```

**4. Filter rows by year**

Add code to the loop that reads the input rows so that it checks the year field and only writes out input rows that are from **either** the year 2000 **or** the year 2010.

## 5. Finalize and examine program output

(a) Make sure you've changed the output filename string in the code to
`world_bank_output_`***`YOUR_UNIQUE_NAME`***`.csv`
When you run your program, if everything is correct, you should see:

```
Done! Wrote a total of 428 rows!

Process finished with exit code 0
```

and it should have created a .csv file called `world_bank_output_`***`YOUR_UNIQUE_NAME`***`.csv` in the same directory.

(b) Open your new csv output file in your favorite spreadsheet program (or any other number-crunching environment that lets you sort tables, group by field), and sort the rows by year (primary sort key) and region (secondary sort key). Compute the median Mobile users per capita for each region, by year.

- Which region had the largest growth in mobile users per capita between 2000 and 2010, and by how much?
- Which region had the smallest growth, and by how much?

Submit written answers in a plain text file named `si330-hw1_`***`YOUR_UNIQUE_NAME`***`.txt`.

**Useful links:**

Python 3.x csv File Reader documentation

Python 3.x Data Structures (Dictionaries)