In our final two-part project, you will use the GDP data from the previous project to create plots on a world map using Pygal. When complete, your code should have graphical functionality similar to this page at the World Bank data site. The primary goal of this assignment is to gain more hands-on experience working with multiple dictionaries as well as searching the web for information about specific features of a package. The assignment will also expose you to some typical issues that arise in cleaning and unifying multiple sets of data.

Preliminaries: Working on the Project

As with the previous project, you should follow the course coding style guidelines. We have also provided you with an Owltest page to receive a preliminary grade and feedback on your project. Remember that each OwlTest link is specific to a particular project and that it does not submit a grade to Coursera.

When you are ready to submit your code to be graded formally, submit your code to the assignment page for this project. You will be prompted to open a tool which will take you to the Coursera LTITest page. Note that the Coursera LTITest page looks similar to the OwlTest page, but they are *not* the same! The CourseraLTI Test page has a **white** background and does submit your grade to Coursera.

Project: Plotting GDP Data on a World Map - Part 1

Provided Files

We will again use the World Bank data from the last week's project for this two-part project. Our version of this data is available here as a CSV file. **Please use this version for your project** since having everyone work from the same data set will make testing your code much easier.

We have provided the following template that you can use to get you started on the project. It includes the signatures (name, parameters, and docstrings) for all of the functions that you will need to write. The code however, simply returns some arbitrary value no matter what the inputs are, so you will need to modify the body of the function to work correctly. You should not change the signature of any of the functions in the template, but you may add any code that you need to.

As with the last project, we will again use a gdpinfo dictionary to specify information about the GDP data file. The gdpinfo dictionary is exactly the same as before and contains the following keys, all of which are strings (the use of these keys will become apparent as you work on the project, you may want to refer back to this information as you work on the different parts of the project):

- "gdpfile": the name of the CSV file that contains GDP data.
- "separator": the delimiter character used in the CSV file.
- "quote": the quote character used in the CSV file.
- "min_year": the oldest year for which there is data in the CSV file.
- "max_year": the most recent year for which there is data in the CSV file.
- "country_name": the name of the column header for the country names.
- "country_code": the name of the column header for the country codes.

If you look in the template file, you will see an example of such an "gdpinfo" dictionary that is used to access the GDP data from the World Bank discussed above.

While we have provide real GDP data, we strongly recommend you write smaller tests and utilize OwlTest to test each function you write. If something goes wrong, you will likely want to write smaller tests to help you understand how your code is working anyway. OwlTest uses smaller files to allow more targeted and understandable testing. You can use those files on your own, as well. Once you have everything working, you should be able to operate on the real data.

Problem 1: Create a dictionary that maps Pygal country codes to World Bank country names

The main task of this project is to process the World Bank GDP data and build a dictionary whose values represented the GDP data for a given year that can be plotted with Pygal. The key step in this process is reconciling Pygal country codes/names with the World Bank country names. So, you first must write a function called **`reconcile_countries_by_name`** which takes **`plot_countries`**, a dictionary mapping country codes used by a plot library (such as Pygal) to the corresponding country name, and **`gdp_countries`**, a dictionary whose keys are the country names used within the GDP data. The values in the **`gdp_countries`** dictionary are irrelevant for this function, but presumably contain GDP data for each country. The **`reconcile_countries_by_name`** function should return a dictionary and a set. The dictionary should map the country codes from **`plot_countries`** to country names that match between **`plot_countries`** and **`gdp_countries`**. It should not contain key-value pairs for the countries within **`plot_countries`** that do not appear in **`gdp_countries`**. The set should contain all of the country codes within **`plot_countries`** that did not match any countries in **`gdp_countries`**, so is effectively the set of countries that the plot library knows about but cannot be found within the GDP data.

Here is the signature of the **`reconcile_countries_by_name`** function:

```python
def reconcile_countries_by_name(plot_countries, gdp_countries):
    """
    Inputs:
      plot_countries - Dictionary whose keys are plot library country codes
                       and values are the corresponding country name
      gdp_countries  - Dictionary whose keys are country names used in GDP data

    Output:
      A tuple containing a dictionary and a set.  The dictionary maps
      country codes from plot_countries to country names from
      gdp_countries The set contains the country codes from
      plot_countries that were not found in gdp_countries.
    """
```

You need to write the code that implements this function.

Hints:

1. Be sure to look at the contents of the returned set. This should give you an understanding of how well the two sets of country names agree.
2. If you code runs correctly, using the Pygal countries and the World Bank data, there should be 30 countries that are not found in the World Bank data. Examine these missing names and consider how to improve the mapping from Pygal country codes to World Bank country names. This topic will be the focus of next week's project.

Problem 2: Transform GDP data for given year into a form suitable for a world map plot

Your next task is to implement the main function that processes GDP data. You must write a function called **`build_map_dict_by_name`** which takes **`gdpinfo`**, a GDP information dictionary (as used in the first project), **`plot_countries`**, a dictionary mapping country codes used by a plot library (such as Pygal) to the corresponding country name, and **`year`**, the year for which to create a GDP map dictionary, expressed as a string. The **`build_map_dict_by_name`** function should return a dictionary and two sets. The dictionary should map the country codes from **`plot_countries`** to the log (base 10) of the GDP for the associated country in the given year. (The logarithmic scaling is chosen to yield a better distribution of color shades in the final plot.) The first set should contain the country codes from **`plot_countries`** that were not found in the GDP data file. The second set contains the country codes from **`plot_countries`** that were found in the GDP data file, but have no GDP information for the specified year.

Here is the signature of the **`build_map_dict_by_name`** function:

```
def build_map_dict_by_name(gdpinfo, plot_countries, year):
    """
    Inputs:
      gdpinfo        - A GDP information dictionary
      plot_countries - Dictionary whose keys are plot library country codes
                       and values are the corresponding country name
      year           - String year to create GDP mapping for

    Output:
      A tuple containing a dictionary and two sets.  The dictionary
      maps country codes from plot_countries to the log (base 10) of
      the GDP value for that country in the specified year.  The first
      set contains the country codes from plot_countries that were not
      found in the GDP data file.  The second set contains the country
      codes from plot_countries that were found in the GDP data file, but
      have no GDP data for the specified year.
    """
```

You need to write the code that implements this function.

Hints:

1. You should be using functions that you have already written in your implementation of this function. In particular, remember that you have already written a function to read a CSV file for the last project!
2. You might find the Python `math` module to be helpful.
3. When the World Bank does not have GDP data for a country for a specific year, the field will be empty. This will be read as an empty string.
4. If you examine the World Bank GDP data downloaded directly from the World Bank site, you will note that the GDP information for Congo uses the number `'0'` to indicate no GDP for the years up until 1991. We have edited the data to replaced `'0'` by the empty string so that missing GDP data is represented uniformly.

Problem 3: Create an SVG image of the GDP data plotted on the world map

As the final part of this project, your task is to write a function that takes the GDP map information computed using `build_map_dict_by_name` and create a world map plot using Pygal. You should write a function called `render_world_map` which takes `gdpinfo`, a GDP information dictionary, `plot_countries`, a dictionary mapping country codes used by Pygal to the corresponding country name, and `year`, the string year for which to create a GDP map dictionary, and `map_file`, the string name of the file to write the output plot to.

Using these inputs, you should use Pygal to plot the logarithmically-scaled GDP data on a world map. Review Pygal's documentation on world maps for more details. Make sure that you plot not only the GDP data, but also the countries which are missing from the GDP data entirely and the countries that are contained within the GDP data, but have no data for the given year.

The output plot should be stored in an SVG file with the name specified by the **`map_file`** input.

Here is the signature of the **`render_world_map`** function:

```
def render_world_map(gdpinfo, plot_countries, year, map_file):
    """
    Inputs:
      gdpinfo        - A GDP information dictionary
      plot_countries - Dictionary whose keys are plot library country codes
                       and values are the corresponding country name
      year           - String year to create GDP mapping for
      map_file       - Name of output file to create

    Output:
      Returns None.

    Action:
      Creates a world map plot of the GDP data for the given year and
      writes it to a file named by map_file.
    """
```

You need to write the code that implements this function.

Hints:

1. Make sure make use of the functions you have previously written!
2. Your function should create a **`pygal.maps.world.World()`** object and use the **`add()`** method to plot both the dictionary containing GDP data and the sets containing missing countries.
3. You should see three different colored countries. One color for countries with GDP data, one for countries that are missing from the GDP data, and one for countries that don't have GDP data for the given year.
4. As was the case for last week's project, you can also render the output directly to your browser using the **`render_in_browser()`** method.
5. The precise labels on your plots are up to you. As a general rule of thumb, your plot should have an informative title and legend entries. See pygal's documentation on how to add such labels to your plot.

VERY IMPORTANT FINAL NOTE: OwlTest does not include tests for the final function **`render_world_map`** since specifying the precise format of the resulting output image is extremely difficult. Therefore, your grade for this project will depend on the correctness of your implementation of the other required functions. You are welcome to self-assess the correctness of your version of the final function **`render_world_map`** based on the sample output images provided below (note that the function **`test_render_world_map`** in the template will call your **`render_world_map`** function appropriately to create these images):

- **`year = "1960"`** produces the world map plot isp_gdp_world_name_1960.svg.
- **`year = "1980"`** produces the world map plot isp_gdp_world_name_1980.svg.
- **`year = "2000"`** produces the world map plot isp_gdp_world_name_2000.svg.
- **`year = "2010"`** produces the world map plot isp_gdp_world_name_2010.svg.

✓ Complete