Python programming assignment

W. H. Bell

October 18, 2022

1 Overview

The objective of the assignment is to build a program that reads Scottish census population and SIMD data and performs a simple analysis with respect to the multi-mode electoral ward regions. The input data files can be found on the MyPlace page. These files have been downloaded from the Scottish census and SIMD web sites. The census data file has been modified slightly, such that it is easier to read it with a Python program.

A complete solution to the assignment should include a zip file that contains three Python files:

- `simd_age.py` - Including a program, classes and functions.
  - `function skip_lines` - A function to skip number of specified lines in a file, leaving the file pointer referring to the next line.
  - `class CensusData` - A class to load the census data file and provide analysis functions.
  - `class SIMD_Data` - A class to load the SIMD data file and provide analysis functions.
- `test_census_data.py` - A unit test program for testing the `CensusData` class.
- `test_simd_data.py` - A unit test program for testing the `SIMD_Data` class.

These Python files are formed by following the instructions that are given in Section 2. The software that is written should follow the PEP8 coding standard. The software should include appropriate commenting.

2 Instructions

Complete the following steps to construct the program.

1. Create a file named `simd_age.py`.
2. In the `simd_age.py` file, write a function named `skip_lines`. This function should accept a file connection and a number of lines to skip. The function should use the `readline()` function to read lines. If `readline()` returns an empty string, then there are no more lines in the input file. If there are fewer lines in the input file than are requested to be skipped, the function should return `False`. If there are enough lines, the function should return `True`.

3. In the `simd_age.py` file, write a class named `CensusData`. The constructor for the class should accept a file name and assign it to a file name data member. The constructor should also create an empty dictionary data member named `data_dict`.
4. Write a `__repr__` member function for the `CensusData` class that returns its data members within a string, together with their data member names.
5. Write a `load` member function for the `CensusData` class. This function should not accept input arguments. The function should:
   - Check if the input file exists. If it does not, then the function should return `False`.
   - Use `encoding="iso-8859-1"` when opening the input file.
   - Skip the first four lines of the input file.
   - Create a `DictReader` to read from the input file.
   - Read data from the `"Region"`, `"Range"` and `"All people"` columns.
   - Store the total population per range and region, such that the `data_dict` data member has the structure `data_dict[region][range]`, where the values correspond to the population. For example, `data_dict["Region1"]["1"]` should contain the number of children that are one year old in the region `"Region1"`.
   - Return True on success.
6. Write a `regions` member function for the `CensusData` class that returns a list of the available regions that have been loaded.
7. Write a `total_population` member function for the `CensusData` class that returns the total population in a specific region for people an age less than or equal to an input age. The function should:
   - Accept the name of a region and an age value as an integer.
   - Return `0` if the region is not in the available regions.
   - Consider the range `"Under 1"` as `"0"`.
   - Consider the range `"85 to 89"` as `"89"`.
   - Consider the range `"90 to 94"` as `"94"`.
   - Consider the range `"95 and over"` as `"100"`.
   - Convert the range to an age, ignoring the `"All people"` range.
   - Sum the population up to and including the input age, returning the result of the sum.
8. In the `simd_age.py` file, write a class named `SIMD_Data`. The constructor for the class should accept a file name and assign it to a file name data member. The constructor should also create an empty dictionary data member named `data_dict`.
9. Write a `__repr__` member function for the `SIMD_Data` class that returns its data members within a string, together with their data member names.

10. Write a `load` member function for the `SIMD_Data` class. This function should not accept input arguments. The function should:

    - Check if the input file exists. If it does not, then the function should return `False`.
    - Use `encoding="iso-8859-1"` when opening the input file.
    - Create a `DictReader` to read from the input file..
    - Read data from the `"SIMD2020v2_Rank"` and `"MMWname"` columns, which correspond to the SIMD rank and the region name in the census data file.
    - The function should assign the average SIMD rank per region to a dictionary data member `data_dict[region]`, such that `data_dict["Region1"]` contains the average SIMD rank for the region `"Region1"`.
    - Return `True` if the function is successful.

11. Write a `regions` member function for the `SIMD_Data` class that returns a list of the available regions that have been loaded.
12. Write a `lowest_simd` member function for the `SIMD_Data` class that returns the name of the region that has the lowest average SIMD rank.
13. Write a `main` function in the `simd_age.py` file. This function should be called when the `simd_age.py` file is executed. The function should:

    - Load the two input files:

    ```
            census_data = CensusData("DC1117SC.csv")
            if not census_data.load():
                return

            simd_data = SIMD_Data("SIMD_2020v2csv.csv")
            if not simd_data.load():
                return
    ```

    - Print the region with the lowest average SIMD rank.
    - Print the lowest average SIMD rank.
    - Print the total population of 15 and under that are in the region with the lowest SIMD rank.

14. Create a file named `test_census_data.py`. Use `import` to import the `simd_age` module. In the `test_census_data.py` file, create a unit test class named `TestCensusData` that inherits from `unittest.TestCase`. The unit test class should include member functions to:

    - Test the `regions` member function of the `CensusData` class.
    - Test the `total_population` member function of the `CensusData` class.

15. Create a file named `test_simd_data.py`. Use `import` to import the `simd_age` module. In the `test_simd_data.py` file, create a unit test class named `TestSIMD_Data` that inherits from `unittest.TestCase`. The unit test class should include member functions to:

- Test the `regions` member function of the `SIMD_Data` class.
- Test the `test_lowest_simd` member function of the `SIMD_Data` class.

The Python files `simd_age.py`, `test_census_data.py` and `test_simd_data.py` should be added to a zip file and submitted for marking.

Marking Scheme

The assignment is an individual assignment that should be completed within the deadline that is posted on the module's MyPlace page. If sections of the program are found to be copied from another student, then associated marks will be deducted from the total awarded.

- The assignment is marked out of 10.
- Up to 2 marks will be awarded for commenting Python that is written.
  - Marks will be awarded for the level of commenting used.
  - 2 marks will be awarded for appropriate commenting, similar to course examples.
  - 1 mark will be awarded for some appropriate commenting.
- Up to 6 marks will be awarded for the structure of the Python.
  - Marks will be awarded for the Python classes and functions.
  - Marks will be awarded for the Python unit tests.
  - The marks will be split between the Python program and unit testing code, relative to the amount of code written. The unit testing code is expected to be significantly shorter and therefore will be assigned up to 1 mark.
- Up to 2 marks will be awarded for the functionality of the program.
  - 1 mark will be awarded if the program partially works as required.
  - 2 marks will be awarded if the program works as required.