Jupyter Notebook Assignment
Instructions

Before Starting

Please play the *Preparing for Jupyter Notebook Assignment* tutorial video before starting this assignment. You will find it in the weekly schedule.

Important Note: Do NOT use the Python *numpy* or *pandas* packages

It is possible that you are already aware of the Python *numpy* and *pandas* packages. In this course, we do NOT cover either of these packages. Instead, we are exploring how to create program solutions without them. So, please do NOT include *numpy* or *pandas* in your solutions to these exercises. When grading this assignment, we will be making point deductions for solutions that use either of these packages.

Overview

This assignment follows a different pattern than previous assignments in this course. In this new pattern, you will be expected to do the work in the tutorial videos provided for the assignment. The work that you submit for the assignment will be a combination of work where you followed along in the tutorial and other work that you will complete on your own.

In this assignment, we will be creating a Jupyter notebook within a PyCharm project. When complete, the notebook will be an example of how to use a Jupyter notebook for a data cleaning task.

Tools

I am expecting you to use the tools that are demonstrated in the tutorial videos: Anaconda, PyCharm, and Jupyter.

Tool Versions

The tutorial videos for this assignment were created in a prior semester when we were using Python 3.8. In the current semester, I am expecting you to use Python 3.10.

Starter Files

One starter file will be available for download from the Weekly Schedule:

- raw_data.txt

Tutorial Parts

This is a 3-part tutorial.

Part 1 – Create Notebook, Add Requirements and Overview
In this part of the tutorial, we create the workbook and add documentation cells to the top of the workbook. These documentation cells include:

- Title
- Requirements
- Overview

Part 2 – Raw Data, Configure, Analyze City Name Values, Analyze State Name Values
In this part of the tutorial, we create the *Raw Data* portion of the workbook. This includes the *Configure* section. You will need to download the raw_data.txt starter file and add it to your project.

You will follow the tutorial to create the *Analyze City Name Values* portion of the workbook. To do this, you will need to create the *analyze_city_name_values.py* program in your PyCharm project. Then, you will need to import that program and run it from within the notebook.

Using this same approach, you will do the next portion of this assignment on your own. This will include creating the *analyze_state_name_values.py* program in your PyCharm project, importing it into the notebook, and running it.

Part 3 – Correct Data Coding Errors, Configure, Run Correction Program, Cleaned Data, Analyze City Name Values, Analyze State Name Values
In this part of tutorial, we will create the C*orrect Data Coding* Errors portion of the notebook. This will include creating notebook cells for titles and configuration. We will work together to create the *correct_data_coding_errors.py* program. This initial version of the program will include code to correct the problems with city names. We will then continue on to create the C*leaned Data* portion of the notebook. This will include another An*alyze City Name Values* run.

Having worked together to correct the city names and demonstrate the correction, you will continue on to do the same activities for state names.

When complete, your notebook should match the notebook version that I show at the very end of the Part 3 tutorial video.

Exercises

1. **Exercise 1 (Regular)**
   Follow Parts 1 through 3 of the tutorial instructions exactly.

2. **Exercise 2 (Challenge)**
   The goal of this exercise is to make a new, more verbose version of the data correction program. To do so, use the following procedure:

   a. Copy the *correct_data_coding_errors.*py program to create a new program named *correct_data_coding_errors_verbose.py.*

   b. Modify the program so that it prints a list of records that were changed on the console. Keep a count of these changed records so that a total can be printed at the end of processing.

   c. Unit test this new verbose program version using PyCharm.

   d. Do not change the Jupyter Notebook file created in Exercise 1. Instead, create a copy of this file named *data_cleaning_workflow_example_with_challenge.ipynb*.

   e. Modify this new notebook file so that it imports and runs *correct_data_coding_errors_verbose.py*. Test the notebook to make sure that this new challenge version works as expected.

   When you run the *correct_data_coding_errors_verbose.py* program using the fully populated raw data file, the printed output should be as follows:

```
Record 1: Clinton,PENNSYLVANIA,518 ---> Clinton,Pennsylvania,518
Record 8: ARLINGTON,California,517 ---> Arlington,California,517
Record 12: Franklin,texas,749 ---> Franklin,Texas,749
Record 14: Lebanon,New York ,235 ---> Lebanon,New York,235
Record 15: Dayton,Michigan ,347 ---> Dayton,Michigan,347
Record 17: GEORGETOWN,Pennsylvania,123 ---> Georgetown,Pennsylvania,123
Record 20: Centerville,NEW YORK,878 ---> Centerville,New York,878
Record 21: Winchester,GEORGIA,973 ---> Winchester,Georgia,973
Record 23: chester,Florida,977 ---> Chester,Florida,977
Record 24: salem,Michigan,990 ---> Salem,Michigan,990


**** MANY LINES HAVE BEEN OMITTED TO SAVE SPACE ****


Record 955: Bristol ,North Carolina,345 ---> Bristol,North Carolina,345
Record 966: madison,ILLINOIS,460 ---> Madison,Illinois,460
Record 967: Winchester,ILLINOIS,151 ---> Winchester,Illinois,151
Record 971: Washington ,California,212 ---> Washington,California,212
```

Record 974: Newport,Illinois ,636 ---> Newport,Illinois,636
Record 977: fairview,Georgia,695 ---> Fairview,Georgia,695
Record 998: newport,Michigan ,788 ---> Newport,Michigan,788
Record 1000: Arlington,CALIFORNIA,610 ---> Arlington,California,610

Input file: data/raw_data.txt
Output file: data/cleaned_data.txt
1000 records were processed.
189 records were changed.

Important – Check your work for these issues before submitting your assignment

The Python programs that we create during this assignment will be imported and run from the Jupyter notebook. They should be created using a special programming style that is suited to this Jupyter notebook use case. Here are some of the program style issues that you should check before submitting:

1. **Programs should not run when imported.**
   The Python programs that we are creating in this assignment should not run when imported into the Jupyter notebook. They should only run when explicitly called. So, remember to include the following in your code:

   ```
   if __name__ == '__main__':
       main()
   ```

2. **Programs should not prompt the user for input.**
   The user interface approach that is appropriate for Jupyter notebooks does not include prompting users from Python programs using the *input()* function. Instead, any values that are needed to configure the Python program should be passed as parameters. This allows the parameter values to be set explicitly in the Jupyter notebook. Setting parameter values in the notebook provides the explicit documentation that the notebook users need to see. Then, the notebook reader sees all the configuration values that contributed to the workflow.

3. **Unit test should be done using PyCharm. The Jupyter notebook is a production run.**
   Use PyCharm to do unit testing of a Python program that will eventually be called from a Jupyter Notebook. If you need to run more than one unit test scenario, then just change the parameter values in the *main()* function and run another test. This would, for instance, allow you to test a program twice – once with a populated input file and once with an empty input file. Remember that there is no need to repeat this unit testing in the Jupyter notebook. The call from the Jupyter notebook represents a production run. So, just run it with whatever data is needed to accomplish the production workflow.

Code Deliverables

You are expected to submit a properly organized PyCharm project that is ready to be tested using Anaconda, PyCharm, and Jupyter. Please refer to my tutorial video for details.

Non-Code Deliverables

Your PyCharm project must also include a data subdirectory that contains the raw_data.txt file, and the cleaned_data.txt file.

File and Directory Naming

Please name your Python program files as instructed in each tutorial video. Please use the following naming scheme for naming your PyCharm project:

surname_givenname_jupyter_notebook_assignment

If this were my own project, I would name my PyCharm project as follows:

trainor_kevin_jupyter_notebook_assignment

Use a zip utility to create one zip file that contain the PyCharm project directory. The zip file should be named according to the following scheme:

surname_givenname_jupyter_notebook_assignment

If this were my own project, I would name the zip file as follows:

trainor_kevin_exercises_jupyter_notebook_assignment

Due By

Please submit this assignment by the date and time shown in the Weekly Schedule.

Last Revised
2023-03-22