Zelle 3e Chapter 2 Coding Assignment

General Instructions

My expectations for your work on coding assignment exercises will grow as we progress through the course. In addition to applying any new programming techniques that have been covered in the current chapter, I will be expecting you to follow all of the good programming practices that we have adopted in the preceding weeks. Here is a quick summary of good practices that we have covered so far:

- Include a Python Docstring that describes the intent of the program.
- Place your highest-level code in a function named `main`.
- Include a final line of code in the program that executes the `main` function.
- Follow all PEP-8 Python coding style guidelines enforced by the PyCharm Editor. For example, place two blank lines between the code making up a function and the code surrounding that function.
- Choose names for your variables that are properly descriptive.
- Define CONSTANT_VALUES and use them in place of "magic numbers".
- Model your solution after the code that I demonstrate in the tutorial videos.
- Remember to test your program thoroughly before submitting your work.
- Make sure that your test input/output matches the sample provided.

Exercise 1 (Regular)
Create a program named *simple_sentence_builder.py*. It should meet the following requirements:

1. Prompt the user for 6 character strings to be used as words in a sentence. Use the approach demonstrated in the sample output below.

2. Print the six words followed by a period to form a printed sentence. Don't worry about the capitalization of the first character of the sentence. We will let the user decide if the first word will begin with a capital letter.

3. Make sure that there are spaces (But, no line breaks) between the words of the printed sentence. Make sure that there is no space between the last word and the period.

When running your test, you should expect the following input/output on your console:

```
Please enter the first word: Wet
Please enter the second word: birds
Please enter the third word: never
Please enter the fourth word: fly
Please enter the fifth word: at
Please enter the sixth word: night
Wet birds never fly at night.
```

Exercise 2 (Regular)
Create a program named *flavor_list_printer.py*. This program prints the items in a list of strings. It should meet the following requirements:

1. Simulate user input by defining a list of strings that contains names of flavors of ice cream. Place the following flavor names in the list:

    - Chocolate
    - Green tea
    - Grape
    - Mango
    - Vanilla
    - Mint chocolate chip

2. print the list title “FLAVOR LIST” on a line by itself.

3. Print each flavor name in the list on a separate line.

When running your test, you should expect the following output on your console:

```
FLAVOR LIST

Chocolate
Green tea
Grape
Mango
Vanilla
Mint chocolate chip
```

Exercise 3 (Regular)
Create a program named *pounds_to_stones.py*. This program converts a measurement in pounds to a measurement in stones. It should meet the following requirements:

1. Prompt the user for a measurement in pounds.

2. Convert the measurement into stones.

3. Print the converted measurement.

4. This program should use the following conversion factor:

    1 pound equals 0.0714286 stones

Don't worry about controlling the number of decimal places in the output. We will learn that skill in a future chapter.

When testing, be sure to test your work against calculations made separately on a calculator.

When running your test, you should expect the following input/output on your console:

```
Please enter a measurement in pounds: 95
Measurement in stones: 6.785716999999999
```

Exercise 4 (Challenge)
Create a program named *travel_bucks.py*. This program converts an amount in U.S. Dollars to an equivalent amount in a selected list of other currencies. It should meet the following requirements:

1. Prompt the user for the amount in U.S. Dollars.

2. Convert the amount to the following currencies based on the exchange rate provided in the table below:

Currency	Exchange Rate Factor
Euros	0.8815
China Yuan	6.3385
India Rupees	74.3827
UK Pounds	0.7376
Canada Dollars	1.2581

3. Print the converted amounts in a report with a heading (see sample output below).

Don't worry about controlling the number of decimal places in the output. We will learn that skill in a future chapter.

When testing, be sure to test your work against calculations made separately on a calculator.

When running your test, you should expect the following input/output on your console:

```
Please enter the amount in U.S. Dollars: 999.99

999.99 in U.S. Dollars may be exchanged for:
881.491185 Euros
6338.436615 China Yuan
74381.956173 India Rupees
737.592624 UK Pounds
1258.087419 Canada Dollars
```

Tools
Use PyCharm to create and test all Python programs.

Submission Method
Follow the process that I demonstrated in the tutorial video on submitting your work. This involves:

- Locating the properly named directory associated with your project in the file system.
- Compressing that directory into a single .ZIP file using a utility program.
- Submitting the properly named zip file to the submission activity for this assignment.

File and Directory Naming
Please name your Python program files as instructed in each exercise. Please use the following naming scheme for naming your PyCharm project:

```
surname_givenname_exercises_zelle_3e_chapter_02
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_02
```

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_exercises_zelle_3e_chapter_02.zip
```

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

```
trainor_kevin_exercises_zelle_3e_chapter_02.zip
```

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

Last Revised
2022-05-22