AMS 209 Homework#5: Problem #2

Christina Findley,\textsuperscript{,1*}

\textsuperscript{1}Department of Applied Mathematics and Statistics, University of California Santa Cruz, AMS 209 with Dr. Dongwook Lee (Due: November 9 2016)

*To whom correspondence should be addressed; E-mail: cfindley@ucsc.edu.

November 9, 2016

Implementation:

The conversion\_factors.py starts by importing the sys module to access the exit() function later. My function length\_conversion takes inputs for the length and the units provided by the user. The code starts by using the dict() method to convert a list of tuples into dictionaries for the length and SI units. The first and second elements of the tuples are keys and values of the dictionary. Next, I grab the value corresponding to the key ‘unit\_type’ taken from the user to be used as the unity factor in the conversions. For every key in the convert\_factors dictionary, compute the conversions according to

\[
\text{conversion}_{\text{length}} = \text{convert\_factor} \times \text{length}_{\text{unity\_factor}}
\]  

(1)

Similarly, for every key in the convert\_SI dictionary, compute the conversions according to

\[
\text{conversion}_{\text{SI}} = \text{convert\_factors} \times \text{convert\_factors}[\text{‘meter’}]
\]

(2)

The value after the \times takes the value for the key ‘meter’ in the dictionary convert\_factors.

The inputs from the user are taken in singular form (discussed later). Once the conversions were calculated, I wanted to use plural form. For example, instead of saying 368.00010058402745: ‘inch, I wanted to say 368.00010058402745: ‘inches. Dictionaries are immutable; therefore, it is not possible to change any of the keys directly from singular to plural form. I overcome this obstacle by popping out (ie: “deleting”) a key in the singular form (for example, meter) and fill it with the plural form of the key
(for example, meters). As a result of making the keys plural, the unit_type representing the unit obtained from the user must also be plural. To do this, I use a series of if, else if (elif), and else statements to change the unit_type for each plural form of a unit because not all of the plural forms require merely the addition of the suffix ’s’. For instance, the plural of foot is feet. Next, I use the pop() method again on the convert_factors dictionary to pop out the conversion requested by the user because it is trivial. Clearly, if the user enters 10 yards and the program converts it to 10 yards, it need not be printed. I proceed to print a newline and tell the user that conversions will be done using simple print statements and the escape sequence \n is used to make the new line.

For each key and value in the convert_factors.items() dictionary, print the key and the value of the dictionary after the conversions. This displays to the computed conversions to the user on the screen. The items() method is used to return a copy of the (key,value) pairs of the dictionary. Similarly, a newline is printed along with a message that tells the user that the conversions for IS units will be done next. Using the items() method again and the for loop as before, but this time for the convert_SI dictionary, the conversions will be displayed to the user as keys and values in the same format as before. Under the command if __name__="__main__"; the function will be ran for a variety of inputs. The user is asked for a length using input() to deal with number and the units using raw_input to deal with a string. If the user entered length is negative and/or the units are not in the specified format, an error will be displayed and the program will exit using sys. Lastly, call the function to run the conversions.

Examples:

I have considered a variety of examples for conversion_factors.py. Please refer to README.txt in the problem2 subdirectory of the directory homework5 in bitbucket for more information on these examples, how to run the code, and implementations. First, if the user enters any incorrect inputs, an error will be displayed to the user and the program will terminate as shown below:

![Example Output]

The left case has an invalid unit (league) and the right case has a negative length values. They return appropriate errors. The following two examples were obtained to verify that the same numbers were obtained as in two cases for the explanation of problem 1:
Indeed, the numbers match those in the explanation. I have changed the output slightly to print each conversion on a new line because I think it is more legible this way.

**Comments:**

It should be noted that my function length_conversion is only one possible solution to Problem 2. I have implemented my function by using the dictionaries convert_factors and convert_SI to hold the information for the conversion factors for 1 meter in typical length units and SI units, respectively. An alternative method would be to define convert_factors and convert_SI as lists. There needs to be two lists for each; one list for the keys of the dictionaries, and one list for the values. Dictionaries are not the same as lists in Python so a list implementation would change the syntax in the code. For example, I would have to use the index() method to obtain the unity_factor that will be used in the conversion formula. Assume for example, I have a list called my_list This method my_list.index(unit_type) returns the lowest index in which the object obj for example. Because Python is an object-oriented language, it is a very powerful tool for data structures (strings, sets, dictionaries, lists, and tuples).

Another comment to consider is that the code is limited. It can be improved upon to include more units, such as leagues and decimeter (dm), for both outputs/conversions and inputs from the user. The code also does not consider the case that if any value does equal one, then the plural forms of the keys would need to be singular again. It is not likely that the user will ever find a value near one, but it would be nice to switch the key from the plural form back to singular form if a certain threshold is low enough. For instance, if the threshold is low enough, the code should print a conversion like `1.0000000002302844: 'mile'` instead of `1.0000000002302844: 'miles'` because `1.0000000002302844` is close to 1.0.