1 Implementations of the function length_conversion()

This code document describes my routine implementation that converts a length in a given unit system into other systems. The problem details is in AMS 209 Homework 5.

To convert a length from one unit system to other systems, I write a Python function named length_conversion(). There are two versions of this function. One uses dictionaries; the other uses lists. The complete python script can be found in the file hw5.py and hw5list.py. The followings discuss the implementations of both functions.

1.1 Arguments

Two versions of the function length_conversion() have the same arguments. The function has two arguments, length and system. Input length is a positive or at least non-negative real number that specifies the length we want to convert. The other input system is the unit system of data type “string”, which is chosen from one of the following five units: meter, mile, inch, foot or yard. Users need to enter these two inputs on the screen when implementing the routine.

1.2 Outputs

After implementing the routine, the dictionary version of the function prints two dictionaries out. The first dictionary has four key:value pairs where the keys are different numerical values of length and values are their associated unit system. Each key:value pair has exactly the same length as the length-system pair specified in the arguments.

The other output is a five-element dictionary that shows the conversion of the requested length in meter into its multiples of the International System of Units (SI), including nanometer (nm), micrometer (um), millimeter (mm), centimeter (cm), and kilometer (km). Its keys are values of length and values are the corresponding SI multiple.

The list version of the function, instead, prints two lists of strings. The first list includes four string elements, and each gives us both value and unit of length information. Similarly, the second list has five different strings that tell us the SI multiples of the input length in meter.
1.3 Usage of dictionaries and lists

For the dictionary version, I first create a dictionary

```python
convert = {'meter': 1., 'mile': 0.000621, 'inch': 39.370079, 'foot': 3.28084, 'yard': 1.093613}.
```

Then given the inputs `length` and `system`, I update its values for each key by

```python
unit_value = convert[system]
convert[key] = (length/unit_value) * convert[key].
```

For example, suppose `length = 10` and `system = yard`, the value associated with the key foot, `convert['foot']`, will be \( (10/1.093613) \times 3.28084 = 30.000009144 \).

To compute SI multiples, I first create a dictionary

```python
SI = {'nm': 10**9, 'um': 10**6, 'mm': 10**3, 'cm': 10**2, 'km': 10**(-3)}.
```

Then for each key, I update its value by

```python
SI[key] = convert['meter'] * SI[key].
```

For example, if now `convert['meter'] = 5`, then `SI['um']` will be \( 5 \times 10^6 = 5000000 \).

For the list version, four lists are created:

```python
Unit = ['mile', 'meter', 'yard', 'foot', 'inch']
Val = [0.000621, 1., 1.093613, 3.28084, 39.370079]
SI = ['nm', 'um', 'mm', 'cm', 'km']
SIval = [10 ** x for x in [9, 6, 3, 2, -3]]
```

Then the goal is to update `Val` and `SIval` element by element using the way of calculation mentioned above. For example, `Val[3]` is \( (10/1.093613) \times 3.28084 = 30.000009144 \). If now `Val[1] = 5`, `SIval[1]` (the value for um unit) is \( 5 \times 10^6 = 5000000 \).

2 Implementation Examples

When we implement the routine, the program will ask us to enter value and unit of length that are used as the arguments of the function `length_conversion()`. The basic output looks like

```
$ python hw5.py (python hw5list.py)
Please enter a length (number only): 10
Please enter an unit system (meter, mile, inch, foot, yard): foot
The conversion result for 10 feet:
{3.04799990246 meter, 304.799990246 cm, 30479.9990246 mm, 3047999.90246 um, 3.04799990246 km}
{30.4799990246 nm, 304.799990246 um, 30479.9990246 mm, 3047999.90246 cm, 30479999.90246 km}
```

\(^1\)I combine `hw5.py` and `hw5list.py`'s output together. When running the command `python hw5list.py` in the paranthesis, we get the output after dashline ----.
$ python hw5.py (python hw5list.py)
Please enter a length (number only): 5
Please enter an unit system (meter, mile, inch, foot, yard): mile
The conversion result for 5 miles:
{8051.529790660225: 'meter', 316989.3639291465: 'inch',
  8805.2576489533: 'yard', 26415.780998389695: 'foot'}
{8051529790660.226: 'nm', 8051529.790660226: 'mm', 8.051529790660226: 'km',
  805152.9790660226: 'cm', 8051529790.660226: 'um'}
-----------------------------------------------------------------------------------
['8051.52979066 meter', '8805.25764895 yard', '26415.7809984 foot', '316989.363929 inch']
['8.05152979066e+12 nm', '8051529790.66 um', '8051529.79066 mm', '805152.979066 cm', '8.05152979066 km']

Notice that the output dictionaries have reverse order of key:value compared to convert and SI.
In addition, when the input number is greater than one, the sentence “The conversion result for 10 feet:” uses plural noun form. Moreover, dictionaries print more digits than lists.

If a user enters a negative number, the program will complian and the user needs to re-enter a number until it is valid, like:

Please enter a length (number only): -5
Length must be non-negative. Enter a number again:

Similarly, the program will ask the user to re-type the unit system until it is valid if any typo happens, like:

Please enter an unit system (meter, mile, inch, foot, yard): yarn
Wrong unit. Type it again (meter, mile, inch, foot, yard):

3 Comments

We can use either dictionary or list to complete the length unit conversion. Both of my code are equally concise. If one would like to focus more on the precision of conversion, using dictionaries may be preferred because the output shows more digits. However, if one cares about ordering by unit, list version will be better because dictionaries have no ordering, and as the output shown above, the lists can be ordered either from the largest unit to the smallest unit (mile-meter-yard-foot-inch) or from the smallest unit to the largest unit (nm-um-mm-cm-km).