CSE 101: Computer Science Principles (Fall 2019)

Homework #3

Assignment Due: Tuesday, December 10, 2019, by 11:59 PM

Getting Started

This assignment requires you to write Python code to solve a moderately-complex computational problem. To help you get started, we have given you a basic starter file with a few tests you can use to verify that your code seems to be correct (**note that these test cases are just examples; we will use different test inputs to grade your work!**). Complete this assignment as directed by the instructions on the following pages. **Do not, under any circumstances, change the name of the primary/required function or its parameter list.** The automated grading system will be looking for a function with that exact names and parameter list; if you change the function header, the grading program will reject your solution and mark it as incorrect.

Directions

This assignment is worth a total of 70 points.

- ⚠ The starter file has a comment block at the top with various important information. Be sure to add your information (name, ID number, and NetID) to the first three comment lines, so that we can easily identify your work. **If you leave this information out, you WILL NOT receive credit for your work!**
- ⚠ Submit your work as a single `.py` file. **DO NOT** zip or otherwise compress your file. **If you do so, you will receive a 0 for this assignment!**
- ⚠ Your primary/required function **MUST** use the name and parameter lists specified in the directions that follow and in the starter code file. Submissions that have the wrong function name (or whose function contains the wrong number of parameters) can't be graded by our automated grading system, and may receive a grading penalty (or may not be graded at all).
- ⚠ Your required function must explicitly ***return*** its final answer; the grading program ignores anything printed by your code. **DO NOT** use `input()` anywhere before the `if __name__ == "__main__":` statement in your files; every function should receive any data it needs from its parameters. Programs that crash will likely receive a failing grade, so test your code **thoroughly** before submitting it.
- ⚠ Blackboard will provide information on how to submit this assignment. You **MUST** submit your completed work as directed by the indicated due date and time. We will not accept (or grade) any work that is submitted after the due date and time, or that is submitted before you have signed the course's Academic Honesty agreement.
- ⚠ **ALL** of the solution code that you submit **MUST** be your own work! You may not receive assistance from or share any materials with anyone else, except for the instructor and the (current) course TAs.

Assignment: A Multi-Purpose Value Conversion Tool

(Place your answer to this problem in the "multiconverter.py" file)

Function Summary

Parameters: A string representing a unit type, and a number representing a quantity measured in that unit.

Return Type: A list of numbers, representing the original value converted into one or more other units, in a specific order.

For this assignment, you will develop a program that allows users to perform three types of unit conversions. Depending on the original unit type, the `multiconverter()` function will convert the stating quantity into one or more other units of that type (for example, if the original unit was a measure of length, the function returns the starting value represented using other units of length (e.g., a value in meters might be translated into feet, centimeters, and furlongs).

Your function should take two arguments. The first argument is a (lowercase) string representing a unit of measure, and the second argument is a (positive) number representing a quantity in that type of units. For example, the function call `multiconverter("feet",5)` represents a starting measure of 5 feet. The function returns a list of numbers: the second argument converted into several other units of measure, in a specific order.

Your function should accept units in the three categories below:

1. **Type Size (20 points)**

   For this conversion, your function should accept the following unit types:

   pica, point, pixel

   Use the following guidelines for your conversions:

   - 1 pica is equal to 12 points
   - 1 pica is equal to 16 pixels

   Depending on the original type of unit, your function should return a list containing the original value's equivalent in the all of the units above, in the order given above (for example, if the starting unit is "point", the final list should contain its equivalent in picas, then points, then pixels, **in that order**).

2. **Temperature (25 points)**

   For this conversion, your function should accept the following unit types:

   kelvin, celsius, fahrenheit, rankine, réaumur

   Conversion formulas for each of these units are available from http://www.csgnetwork.com/temp2conv.html

   Depending on the original type of unit, your function should return a list containing the original value's equivalent in all of the units above, in the order given above (for example, if the starting unit is "kelvin", the final list should contain its equivalent value in kelvin, celsius, fahrenheit, rankine, and réaumur degrees, **in that order**).

3. **Length (25 points)**

   For this conversion, your function should accept the following unit types:

   feet, meters, rods, fathoms, canas

   Use the following guidelines for your conversions (don't worry about excessive precision in terms of decimal points):

   - 1 meter is equal to 3.2808 feet
   - 1 rod is equal to 16.5 feet
   - 1 fathom is equal to 1.8288 meters, or 6 feet
   - 1 cana is equal to 1.57 meters, or 5.1509 feet

   Depending on the original type of unit, your function should return a list containing the original value's equivalent in the all of the units above, in the order given above (for example, if the starting unit is "fathom", the final list should contain its equivalent in feet, meters, rods, fathoms, and canas, **in that order**).