Assignment P1

Due Date: June 1

Purpose
Yippee! It’s time for you to try out some basic Python in a short, but exciting (‽), program.

Problem
Pam Python owns a lovely 1987 Ford Escort with a stereo cassette tape player. She can’t bear to miss out on her John Denver and ABBA recordings while she’s driving. The trouble is, her recordings are on 8-track tapes, which don’t fit into the car’s stereo. She is always annoyed when she wants to tape songs from her 8-tracks onto cassette tapes. The time of each song is given on the label, but adding up times to make sure it fits onto a cassette is a bit annoying. She decides that it would be a fun thing to write a program in Python to solve this problem.

Input
The program should prompt the user for a two sets of time, each broken down into three numbers representing hours, minutes, and seconds. Thus, the user will input six numbers in total. Each number will be entered as two-digits.

For example, one input time set might be 02, 33, and 09, and the second set of values might be 00, 34, and 14, representing hours, minutes, and seconds, respectively for each time duration. Note that the user will type each of the values on its own line. If hours, minutes, or seconds are 0, that will be typed in as 00; single digit values will have a leading 0. Thus, 00, 00, and 00 is valid input for one of the time sets. You may assume that the total number of days will not exceed 365 and that all of the input durations will be valid positive integers (or zero).

Output
The output should be an aligned table that shows the input hours and minutes plus the sum of the two values. All values should be right justified.¹ No hour or minute value should ever be higher than 60. For example, for the values shown above, the output might be (you do not have to follow this example exactly):

<table>
<thead>
<tr>
<th>Days</th>
<th>Hours</th>
<th>Minutes</th>
<th>Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time 1:</td>
<td>0</td>
<td>2</td>
<td>33</td>
</tr>
<tr>
<td>Time 2:</td>
<td>0</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>Total time:</td>
<td>0</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

As shown above, it is fine to have 0 as a table entry.

Specifics

- Use named global variables for any constant values used in the program. All other variables should be within the main module.

- Include a good introductory comment, as described in class.

¹Right justified means that the values should line up by their decimal points, even if no decimal points are shown.
• Comment all variables; that is, write a comment describing the purpose of a variable the first time it is used.

• Use good (mnemonic) identifiers; camelCase is preferred.

• The program should have appropriate and consistent spacing and indentation.

Notes

• Be sure to adequately test your program.

• Unless explicitly conveyed, you never have to use a Python idea that was not covered in class at the time the project was handed out. Thus, do not use IF statements in any part of this program.

• In a comment at the top or very bottom, type and electronically sign the Wheaton Honor Code Pledge on what you turn in: “I have abided by the Wheaton College Honor Code in this work.”

• The name of your source code file (the one that ends in .py) should be your first initial, last name, and project name with no spaces. For example, my project would be named mgousieP1.py.

• To turn in your project, send your source code as an attachment to mgousie@wheatoncollege.edu by 11:59:59 PM on the due date.

• The time “stamp” on your project is the time that you turn in your last version. This is what is used to determine if your project is turned in on time. I will grade only the last version submitted.

It's almost too easy.
– Dr. Schulz, my Calculus III professor

2gousie.mike@wheatoncollege.edu works, too.