CSE 231 **Fall 2015**

Computer Project #1

Assignment Overview

This assignment focuses on the design, implementation and testing of a Python program to calculate values related to Aesop's fable about the Tortoise and the Hare (see below).

It is worth 15 points (1.5% of course grade) and must be completed no later than 11:59 PM on Monday, September 14.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj01.py` – your source code program

Be sure to use the specified file name and to submit it for grading via the **handin system** before the project deadline.

Assignment Background

One of Aesop's Fables is known as The Tortoise and the Hare. A variation of the fable is:

Tired of being ridiculed by the hare for being slow, the tortoise challenges the hare to a race. Confident of winning, the hare takes his time: first running and then resting, then again running and resting, at fixed intervals, until crossing the finish line. The tortoise moves immediately and steadily, albeit more slowly, to the finish line.

Assignment Specifications

You will develop a program to calculate the time it takes for the tortoise and the hare to reach the finish line under different assumptions about the distance of the race, their speeds, and the resting and running times of the hare.

Your program will prompt the user for the following five inputs:

1. The distance to the finish line (in miles).
2. The speed of the tortoise (in inches per minute).
3. The speed of the hare (in miles per hour).
4. The time that the hare rests after running (in minutes).
5. The time that the hare runs before resting (in minutes).

Your program will then calculate and display the times it takes both the tortoise and the hare to complete the race.

Assignment Notes

1. To clarify the project specifications, sample output is appended to the end of this document.

2. The coding standard for CSE 231 is posted on the course website:

   http://www.cse.msu.edu/~cse231/General/coding.standard.html

Items 1-7 of the Coding Standard will be enforced for this project.

3. The `input` function is used to accept a response from the user. The function accepts a string (a sequence of characters between quotes) as a prompt to display to the user. It then waits until the user types a response (terminated by the user touching the Enter key). Finally, the function returns the user's response as a string.

If the user's response is supposed to be processed as a numeric value, the returned string must be converted into a number. When working with floating point values, a string is converted into a floating point number using the `float` function. The function accepts a string as its argument and returns the floating point number which the string represents. For example:

```
num_str = input( "Please enter a number: " )
num_float = float( num_str )
```

4. The `print` function is used to display any combination of variables, values and strings in the output window. Each item to be displayed must be separated from another item by a comma. All the items will be displayed together, separated by a white space, and followed by a new line. For example:

```
print( num_int, "times two is", num_int*2 )
```

Three items will be displayed when the `print` function is called: the value of the variable `num_int`, the string `"times two is"`, and the result of the calculation.

Assuming that the value of the variable `num_int` is 26, then the output will be:

```
26 times two is 52
```

You are probably already familiar with the quotient and remainder operations for integers, but you may not have seen them used before with floating point (real) numbers. For example, `10//3` is the quotient of `10` divided by 3, which is `3`; and `10%3` is the remainder of `10` divided by `3`, which is `1`.

These operations are also referred to as the floor-division and modulus operations, and they also work with floating point values. For example, `3.5//1.5` is the floor-division (quotient) of `3.5` divided by `1.5`, which is `2.0`; and `3.5%1.5` is the modulus (remainder) of `3.5` divided by `1.5`, which is `0.5`.

You will use these operations to calculate the time that the hare takes to complete the race because it does not run continuously. Instead, the hare alternates between running and resting. For example, if the distance to the finish line is 3.5 miles and the hare runs 1.5 miles each time before resting, then the hare will rest twice (once after 1.5 miles and once after 3.0 miles) and run three times to make it to the finish line (the hare will cover 1.5 miles on each of the first two runs and the remaining 0.5 miles on the last).

The `ceil` function in the `math` module might be useful to calculate the number of times that the hare runs. The number of times that the hare rests is always one less than the number of times that the hare runs (the hare rests between runs). For example, if the distance that the hare runs before resting is 0.4 miles and the race is 1 mile long, then the number of times that the hare will run is 3, which is `math.ceil( 1 / 0.4 )`: the hare will run 0.4 miles and rest, then run another 0.4 miles and rest, and finally run the last 0.2 miles to the finish line.

Suggested Procedure

- *Solve the problem using pencil and paper first.* You cannot write a program until you have figured out how to solve the problem. This first step can be done collaboratively with another student. However, once the discussion turns to Python specifics and the subsequent writing of Python, you must work on your own.

- Use Spyder to create a new program:

    - Use the required file name (`proj01.py`).
    - If you are in a CSE lab, select the H: drive as the location to store your file.

- Write a simple version of the program. Run the program and track down any errors.

- Use the **handin** system to turn in the first version of your program.

- Cycle through the steps to incrementally develop your program:

    - Edit your program to add new capabilities.
    - Run the program and fix any errors.

- Use the **handin** system to submit your final version.

- Be sure to log out when you leave the room, if you're working in a public lab.

Be sure to save a copy of your completed program in your CSE file space (H: drive on the lab machines) ***before*** *the project deadline. If you write your program at home and turn it in from home, you should copy it to your CSE file space* ***before*** *the deadline. In case of problems with electronic submission, an archived copy in the CSE file space is the only acceptable evidence of completion.*

Sample Output

```
In [1]: runfile('.../proj01.py', wdir='...')
 How many miles will the tortoise and hare race? 1
 How many inches can the tortoise cover in one minute? 30
 The tortoise takes 35.199999999999996 hours to finish the race.
 How many miles can the hare run in one hour? 5
 How long does the hare rest (in min)? 60
 How long does the hare run at a time (in min)? 10
 The hare takes 1.2 hours to finish the race.

 In [2]: runfile('.../proj01.py', wdir='...')
 How many miles will the tortoise and hare race? 3.5
 How many inches can the tortoise cover in one minute? 100
 The tortoise takes 36.96 hours to finish the race.
 How many miles can the hare run in one hour? 1.5
 How long does the hare rest (in min)? 5
 How long does the hare run at a time (in min)? 60
 The hare takes 2.5 hours to finish the race.

 In [3]:
```

Note: In both of the sample interactions, the user entered the numeric text (highlighted in green) after each of the questions; the program generated the remaining text (using the `input` and `print` functions).