Python Exercise 4

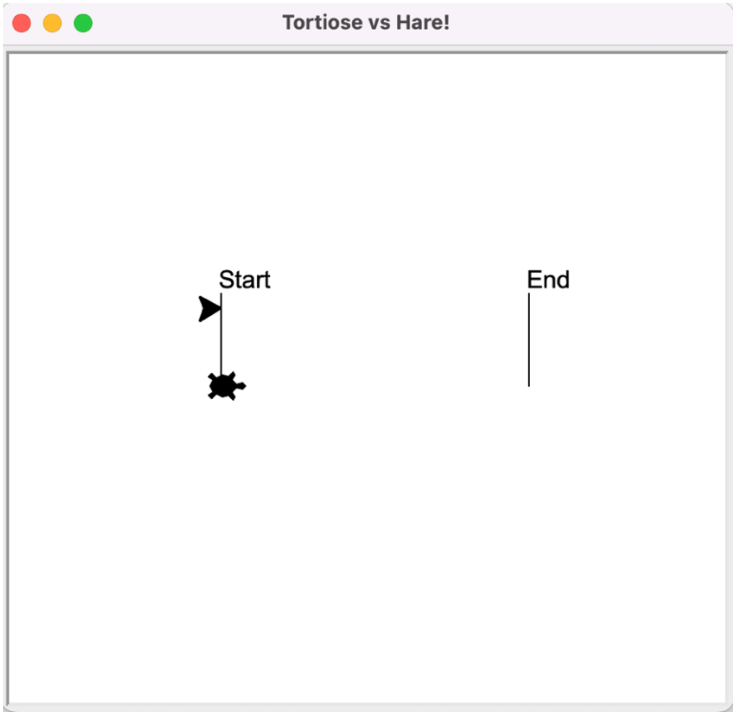
Note: Due to Hurricane Ian, this assignment will count twice (for assignments 4 and 5).

Tortoise vs. Hare (60 pts)

Write a program that simulates the classical race between the tortoise and the hare (with a couple modifications). You will use the turtle module to provide an aerial view of the race!

The tortoise and hare will race in a 'course' of 200 pixels. Each animal starts at the 'starting line' which is position x = -100 in the turtle window. The finish line is at position x = 100 and the first animal to reach the finish line wins the race and the prize (a pail of fresh organic lettuce and carrots). In this program, the course is up the side of a slippery mountain, so the animals can occasionally lose ground.

The tortoise's starting position will be at (-100, 0) and the hare's starting position will be at (-100, 50). To ensure that each animal remains in their own lane, the y coordinate should not change for the duration of the race. (This also prevents the animals from using dirty tactics, like biting or kicking.)



(Note: Diagram shows the tortoise, represented by the turtle cursor, and the hare represented by the classic cursor at position x = -100 (indicated by the line called start). The cursor has been resized in the code, to make it easier to see during the race. Despite the graphics, the tortoise does not actually have a head start!)

Each animal has specific move types that correspond to their movement in the course. The animal, move type, probability of move type and actual move is shown in the table below:

Animal	Move Type	Probability of Move	Actual Move
Tortoise	Fast Plod	50%	3 spaces forward
	Slip	20%	5 spaces backward
	Slow Plod	30%	1 space forward
Hare	Sleep	20%	No movement
	Big Hop	20%	7 spaces forward
	Big Slip	10%	10 spaces backward
	Small Hop	30%	1 space forward
	Small Slip	20%	2 spaces backward

The race is managed by a 'clock' (loop) that ticks once per 'second' (iteration). While the race has not been won, your program should adjust the position of the tortoise and the hare on the course, with each 'tick of the clock' (i.e., iteration in the loop). The race (i.e., loop) ends when one or both of the animals reaches position x = 100 in the window.

Programming tasks

1. Create two value returning functions to keep track of the hare and tortoise's positions. (between x = -100 and x = 100). The function should accept one parameter and return one value.
    a. The parameter represents an animal's initial and current x-positions, and the value returned would be the animal's new x-position in the race.

    b. The body of the functions should include a **random number generator** (RNG) to simulate the movements passed on their probability. A random number should be generated for each animal within the range $1 \leq num \leq 10$. This RNG is essential for determining the animals' movements in the track.

    Example: For the tortoise, a random integer in the range $1 \leq num \leq 5$ should result in a fast plod (as that movement will occur 50% of the time). Similarly, a random integer in the range $6 \leq num \leq 7$ should result in a slip, and a random integer in the range $8 \leq num \leq 10$ should produce a slow plod.

    c. Occasionally, the RNG can cause the animal to slip beyond the starting and ending positions. Use if statements to check the boundaries of the race. If an animal slips past x = -100, reset their position to -100. Similarly, if the animal's position does beyond x = 100, reset the position to 100.

2. The main portion of the code must **use functions from the turtle module** to construct the course and display the race. The course is created by using additional 'turtle' objects to draw lines, (i.e. the starting and ending lines) or write messages as well as the serve as the placeholders for the racing animals (one for the tortoise and one for the hare.).

    a. Use the function **Turtle()** to create additional turtle objects to use. You can assign names to the turtle to distinguish between the objects. (E.g. writing bob = turtle.Turtle() allows you to use bob to call some of the other functions from the module, such as bob.pendown().) You can create as many objects that you need to draw the course or display messages for your race.

    b. Use the **shape()** function to determine which cursors to use to represent the tortoise of the hare. Optional: You can resize the cursor by using the **shapesize(w,l, outline)** function

    c. Notice that the race only shows two turtle objects. Use the **hideturtle()** function to make some turtles invisible. Use these objects to draw the start and end lines (using **forward(pixels)**, **backward(pixels)**, **left(angle)** or **right(angle)**). A turtle should be used to display messages with the **write()** function in the window.

    d. Use the **setpos(x,y)** function to move turtles to positions in your window. Don't forget to use **penup()** and **pendown()** to determine when the objects should draw while moving or not.

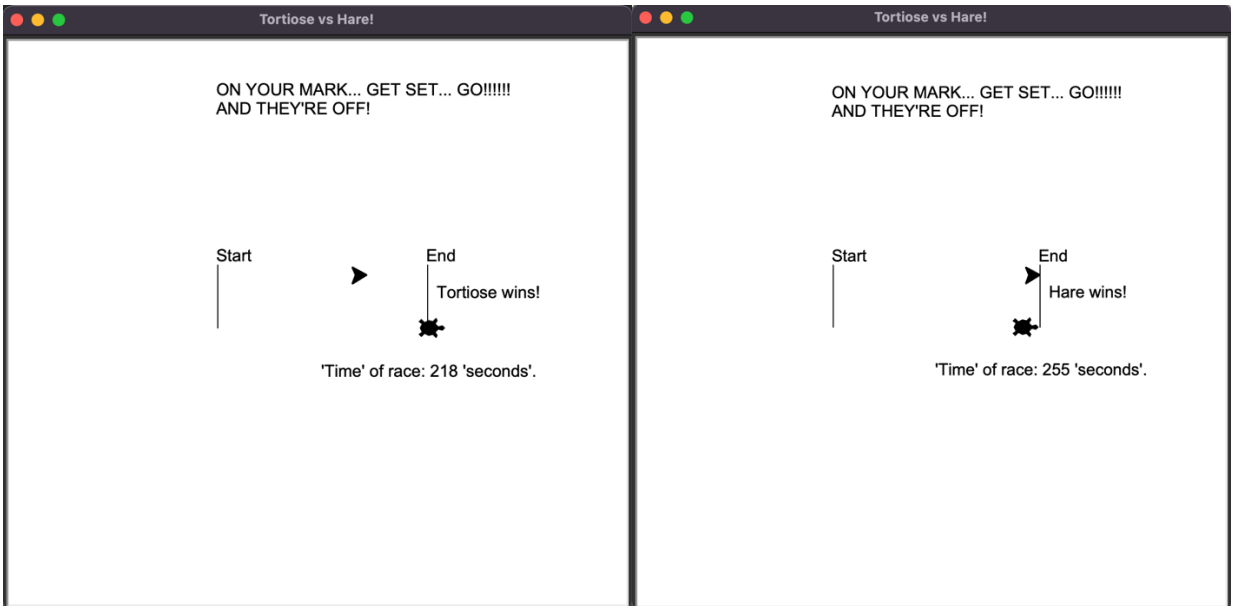
    e. Optional: You can use the **title()** function to label your window.

3. The main portion of your code must also include a while loop to run the race. Create a variable that will serve as the clock for the race, and add 1 to it for each iteration of the loop. The loop is controlled by the variables that represent both the hare's and the tortoises x-position (starting at -100 for each). <u>The loop will run until one or both animals' value reaches x = 100.</u>

4. Once the loop stops, determine the winner of the race. If the value in Tortoise's x-position is greater than or equal to the value in the Hare's x-position, then the Tortoise wins. (Yes, the Tortoise wins in the event of a tie, as the Hare is convinced that a tie is statistically unlikely.) Otherwise, the Hare wins. In each case, print out a message to indicate the winner of the race, as well as the number of iterations it took to complete the race.

Sample output

The outputs below show possible outcomes of the race. Your results may vary.



Tips:

- Keep in mind that a second is represented by an iteration and is not an actual second.
- You will need at least two functions:
    - A function to control the tortoise's movements
    - A function to control the hare's movements
- A main function is not required. You are welcome to create one if you wish, and you can create more functions if you think it will help with organization.
- Using colors for your window background or turtles' objects are optional.