# Milestone 2

due at midnight on Sun Feb 19 (125 points)

Compose a cart-pole agent that uses a the Perceptron class to learn how to balance the pole. Instead of using the standard perceptron delta rule, implement a **hill-climbing** technique, where it randomly tweaks the weights by some small amount, but only keeps the tweaked weights if they perform better than the previously best-performing weights.

There are several parameters to play with:

- By how much can it randomly alter the weights after each trial? (We can think of this as the learning rate.)
- Should it alter all the weights at the same time, or just one (randomly-chosen) weight?
- Should it perform more than one trial (and calculate the *average* reward) before it adjusts the weights? How many trials?

Try your implementation several times with different parameters. In your repository's `README.md` file, document the perceptron's weights that produced the best performance you were able to achieve, as well as the choices you made with respect to the above parameters.

⇒ **Please name your file `cartm2.py`.**