Unit 10

Apply what you know

0. Study the programs in the *Learn something new* section until you can write them yourself from scratch without relying on this document or any other source of information. Here are the programs:

    0.1. A game world is populated by players who encounter one another at random. When an encounter occurs, each of the two players must decide whether to be 'nice' or 'nasty'. If both are nice, they work together cooperatively to earn points and receive 30 each. If both are nasty, they earn nothing. But if one is nice and the other is nasty, the nasty player takes advantage and receives 50 points, while the nice one *loses* 70 points.

    Write a program that creates a population of nine friendly players who are always nice and nine mean players who are always nasty. After 2,000 encounters between randomly selected pairs of players, the program should report the scores and produce a new generation by creating two new players of the same type as each of the ones that scored in the top half of the current generation. The program should continue in this way through five generations.

    0.2. Add a 'Mirror' player type to Program 10.0.1 and rerun it starting with an initial population consisting of six players of each type: friendly, mean and mirror. A mirror player is nice in its first encounter with another player and subsequently behaves in each encounter the way the other player behaved in the previous encounter.

1. Add a new 'Probing' player type to Program 10.0.2 and rerun it starting with an initial population of five players of each type: friendly, mean, mirror and probing. A probing player attempts to identify the type of the other players. If playing against a mean player or a nice one, it's always nasty, protecting itself in the first case and exploiting the vulnerable in the second. If playing against a mirror player or another probing player, it's always nice, since the cooperative result is the best one possible.

    To determine the type of the other player, the prober is always nasty in its first encounter and nice in the second. The corresponding actions of the other player are always enough to positively identify it. If the other player is nice twice in a row, it must be a friendly player. If it is nasty twice, it must be a mean player. If it is nice and then nasty, it must be a mirror player. And if it is nasty and then nice, it must be another prober.

2. Rerun the experiment of Program 10.0.2, but cut the number of encounters in each generation from 2,000 to 500. Observe the result and attempt to explain it. [*Hint*: On average, what percent of a player's encounters will be with 'strangers', players it is

encountering for the first time? Modify your program to report data relevant to this question.]

3. In a natural setting, individuals tend mostly to encounter neighbors rather than just any other individual from the population. Rerun the last experiment in the 'Learn something new' section, but restrict encounters to players at most three apart in the `allPlayers` list. For example, the player at index 7 should only encounter players at indices 4, 5, 6, 8, 9 or 10.

4. A bank is open from 9 a.m. to 5 p.m. There is one teller. Customers arrive at random times, averaging 20 per hour. Each needs a random amount of time with the teller, ranging from one to five minutes. When one customer is with the teller, others wait on a line. Write a simulation program to find out, on average, how long the line at the bank will be. Then rerun the simulation with an average of 25 customers arriving per hour.

   You may find the following function useful. It returns `True` with probability `p`:

   ```
   def chance(p):
     return random.random()<p
   ```

   Suppose, for example, that we pass in .3. The `random.random` function picks a number at random between 0 and 1. There is a 30 percent chance that this number will happen to be lower than .3, so there is a 30 percent chance that the `chance` function will return `True`.

   We use the function this way:

   ```
   if chance(.3):
   ```

   In this case, there is a 30 percent chance that the 'something' in the body of the `if` statement will happen.