Epilogue

Well, that's the whole book. We came a long way, from writing the crow's nest program in chapter 2 to chapter 22's interactive Tic-Tac-Toe game, incorporating a custom class based on named tuples and using type hints. I hope you can see now how much you can do with Python's strings, lists, tuples, dictionaries, sets, and functions. I especially hope I've convinced you that, above all, you should always write programs that are

- *Flexible*, by taking command-line arguments
- *Documented*, by using something like `argparse` to parse your arguments and produce usage statements
- *Tested*, by writing both *unit* tests for your functions and *integration* tests for your program as a whole

The people using your programs will really appreciate knowing how to use your program and how to make it behave differently. They'll also appreciate that you took the time to verify that your program is correct. Let's be honest, though. The person most likely to be using and modifying your programs will be you, several months from now. I've heard it said that "documentation is a love letter to your future self." All this work you put into making your programs good will be very appreciated by you when you come back to your code.

Now that you've worked through all the exercises and seen how to use the tests I've written, I challenge you to go back to the beginning and read the test.py programs. If you intend to adopt test-driven development, you may find that you can steal many ideas and techniques from those programs.

Further, each chapter included suggestions for how to extend the ideas and exercises presented. Go back and think about how you can use ideas you learned later in the book to improve or extend earlier programs. Here are some ideas:

- Chapter 2 (The crow's nest)—Add an option to randomly select a greeting other than "Hello" from a list like "Hello," "Hola," "Salut," and "Ciao."
- Chapter 3 (Going on a picnic)—Allow the program to take one or more options and incorporate those into the output with the correct articles for each item joined on the Oxford comma.
- Chapter 7 (Gashlycrumb)—Download *The Devil's Dictionary* by Ambrose Bierce from Project Gutenberg. Write a program that will look up a word's definition if it appears in the text.
- Chapter 16 (The scrambler)—Use the scrambled text as the basis for encrypting messages. Force the scrambled words to uppercase, remove all the punctuation and spaces, and then format the text into "words" of five characters followed by a space, with no more than five per line. Pad the end so that the text completely fills the last line. Can you make sense of the output?
- new.py—I first wrote a program to create a new program when I was a greenhorn Perl hacker. My new-pl program would add a random quote from the poetry of William Blake (yes, really—I also went through phases with the Brontes and Dickinson). Alter your version of new.py to add a random quote or joke or to customize it in some way for your programs.

I hope you've had as much fun writing the programs as I've had creating and teaching them. I want you to feel you now have dozens of programs and tests with ideas and functions you can steal to create even more programs.

All the best to you in your coding adventures!