Getting Started with Microsoft Visual Studio Code

This document briefly explains how to use the non-web-based version of Microsoft Visual Studio (VS) Code as your primary IDE to work through the chapters in *Problem Solving with Python: Using Computational Thinking in Everyday Life*. This guide assumes that you are familiar with Unix shell commands and the basic operation of VS Code.

Tool setup

As you are installing this IDE on your local machine, you don't need any login account beyond the one you use to log into your own computer. You will, however, need to download VS Code, which you can get from this link.

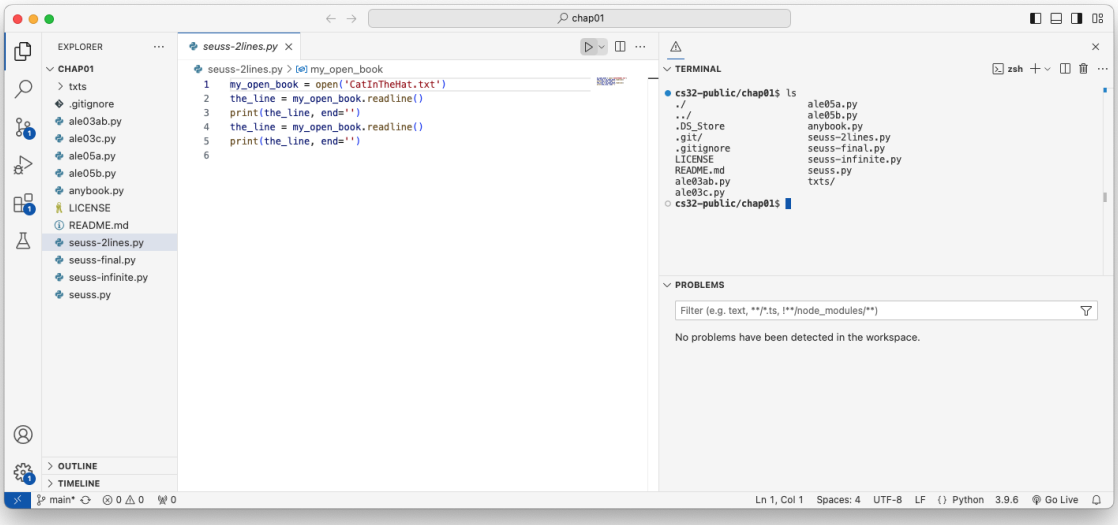
You'll also need to download and install a Python interpreter and its most popular libraries, which you can read about at this link.

If you're not experienced in setting up programming environments, I strongly suggest that you enlist the help of someone who is.

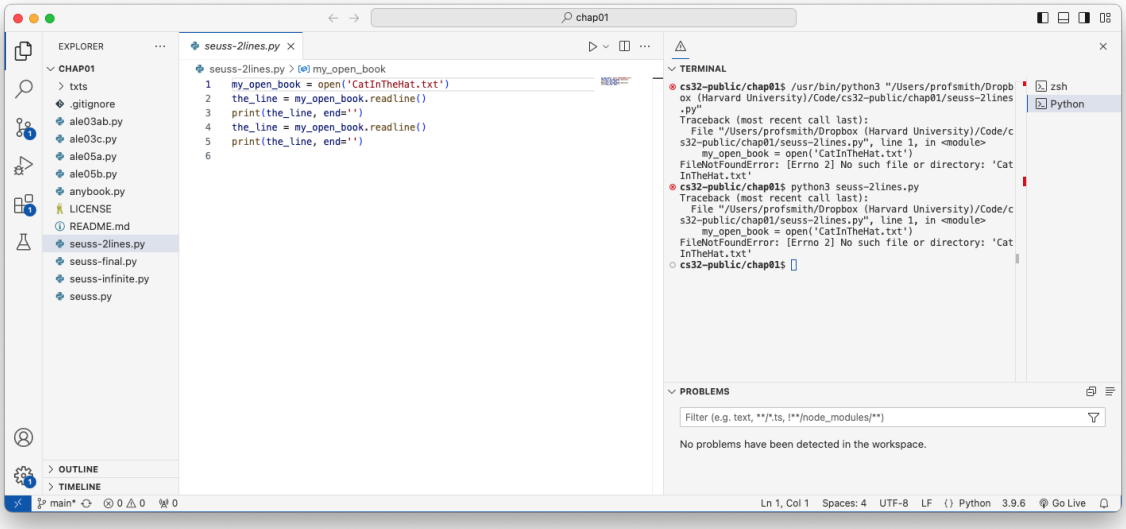
Mapping VS Code's IDE to the book's generic IDE

The book's generic IDE highlights two bars, three panels, and one button all in one IDE window. In the screenshot below of VS Code running on my machine:

- The menu bar is on the far left.
- The file explorer panel is next from the left, and it contains the files associated with the book's Chapter 1.
- The code editor panel is the middle panel, and it displays the contents of `seuss-2lines.py`.
- The console panel is on the right and is open to the Terminal, in which we can interact with my machine's shell program. The panels are movable, and the Terminal panel may appear below the editor panel.
- The run button is in the upper left corner of the editor panel.



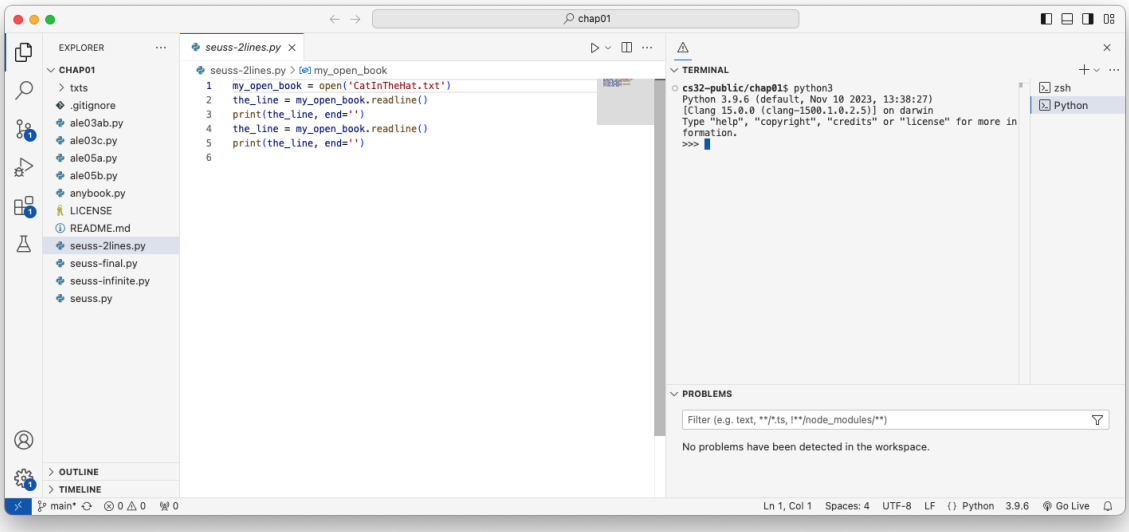
If you click the run button, the IDE runs the Python interpreter on the file in the editor panel and displays the result in the Terminal panel. I don't recommend using the run button because you can do the same thing by typing `python3 program.py` directly in the Terminal panel, as the next screenshot illustrates.



Starting the interactive Python interpreter

Instead of writing a full Python script, we sometimes want to try out a few Python statements. This is best done in what's called the *interactive Python interpreter*. You can tell you're talking to the interactive Python interpreter when the command line prompt looks like three greater-than symbols (>>>).

More generally, we start the interactive Python interpreter in the Terminal panel by typing `python3` at the terminal's command prompt. Yes, it's like running a Python program but not feeding a program file to the Python interpreter.



Chapter 1 contains examples of how the interactive Python interpreter will be useful to us. For now, just know that you type *Ctrl+D* (i.e., simultaneously press your keyboard's Control and D keys) to exit the interactive Python interpreter.

Try starting and exiting the interactive Python interpreter now.

Loading the files for Chapter 1

You're not often going to want to create a file and type its contents. Sometimes you'll want to use files that we've already created. This is how you grab the book's Chapter 1 code.

1. In the Terminal panel, `cd` to the directory where you want to store the folders for all the book's chapters. In the screenshots above, that was my `cs32-public` folder. That's certainly not the name you'll want to use. Pick something that makes sense to you and is located in a sensible spot in your computer's file system.

2. When the shell's focus is in this directory, type the following command at the shell prompt (i.e., after the `$`):

   ```
   git clone https://github.com/pswp-book/chap01
   ```

3. This command downloads all the files you'll need for Chapter 1 from the book's Github repository, and it puts them in a directory called `chap01` you can see in VS Code's file

explorer. While you can open up the folder in the file explorer, you can also ask the shell to show you the files by typing the following at the command prompt and hitting return:

```
ls chap01
```

You'll learn more shell commands in Chapter 13.

4. To open the IDE editor on an existing file, click the file's name in the file explorer.

5. You are ready to start reading Chapter 1. Enjoy!

Loading files for another chapter

It's easy to get the code for another book chapter. Follow the steps in the previous section, except replace `chap01` in the `git` command with another chapter's GitHub repo name. For example, Chapter 8's name is `chap08` and Chapter 11 is `chap11`.

Installing a package

In Chapter 4, you'll start using packages that are not installed by default in the basic Python distribution. In particular, this chapter will use the `requests` package, which you can install by running the Python package manager `pip` in the terminal panel. Specifically, you'd type `pip install requests`, which would install the `requests` package and make it available for you to `import` in your Python programs.

A better approach than installing libraries for all projects is to install them just for a particular project (e.g., like `chap04`). It requires you to be comfortable with a few more terminal commands, and the approach is explained in the chapter README files. See the section about Python virtual environments and the `requirements.txt` file. For more help, ask a generative AI chatbot for more help with these two topics.

[Version 20260227]