Abstraction is simplification. Writing code can get very convoluted and long. With abstraction, we think “There’s a lot of code here. What can I condense? What can I put into its own function/class/method/program so that things can work together and look nicer?” It’s important to start with a plan when coding, but it’s still likely that your code can get overwhelming. Planning with abstraction in mind allows us to mentally break down the code before it’s even written and figure out pieces and how they will work together.

A huge benefit of abstraction is that it can make code much easier to read. Breaking it into pieces and simplifying it makes it more readable. It also helps with runtime, allowing your code to be processed and run faster.

A screenshot of a computer program

Description automatically generated with medium confidence Here is a piece of code from my journal program:

This is my prompt generator class, which is part of the bigger journal program. Instead of generating prompts in my main class, I chose to create a new class that would hold the prompts and choose one at random for the user’s new entry. This makes my main class much easier to read and less full of code.

With proper code planning, abstraction makes code not only easier to read, but easier to write, too. It is more organized and has a better flow. Instead of scrolling up and down, searching for the code you just wrote, it is nicely organized and separated into smaller pieces. Abstraction is useful in any program. In a previous class, I had to make a budget program. Abstraction was a key part of the program, as I created different functions for each part of the budget, similar to the Journal.