What is this class about? You might think that writing code is just about getting a computer to do what you want it to do, but that misses a really big part of the story.

**Source Code has two "audiences":**

- Compiler and CPU
- Humans (including your future self)

**Programs have to be written with two goals in mind:**

- communicating with the computer.
    - syntactically correct and type-correct (first persuading the compiler that your program is sensible)
    - logically correct (gives the right results at runtime)
- communicating with other people. Making the program easy to understand, so that when somebody has to fix it, improve it, or adapt it in the future, they can do so.

**This class is about these THREE MAIN THEMES - writing code that is:**

- **Safe from bugs**. Correctness (correct behavior right now) and defensiveness (correct behavior in the future - easy to modify without accidentally creating bugs) are required in any software we build.
- **Easy to understand**. The code has to communicate to future programmers who need to understand it and make changes in it (fixing bugs or adding new features). That future programmer might be you, months or years from now. You'll be surprised how much you forget if you don't write it down, and how much it helps your own future self to have a good design.
- **Ready for change.** Software always changes. Some designs make it easy to make changes; others require throwing away and rewriting a lot of code.

**Hacking vs. Engineering**

Hacking is often marked by unbridled optimism. What is bad:

- writing lots of code before testing any of it
- keeping all the details in your head, assuming you'll remember them forever, instead of writing them down in your code
- assuming that bugs will be nonexistent or else easy to find and fix

Software engineering is not hacking. Engineers are pessimists, so we:

- write a little bit at a time, testing as you go. Later on, we'll talk about test-first programming.
- document the assumptions that your code depends on
- defend your code against carelessness – especially your own! Static checking helps with that.