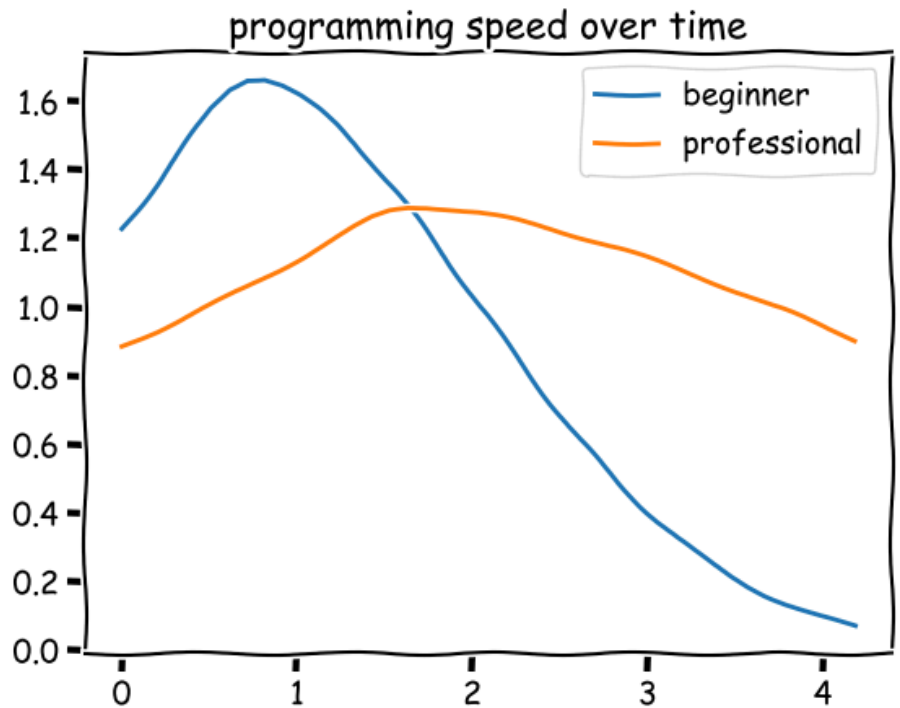
General Introduction

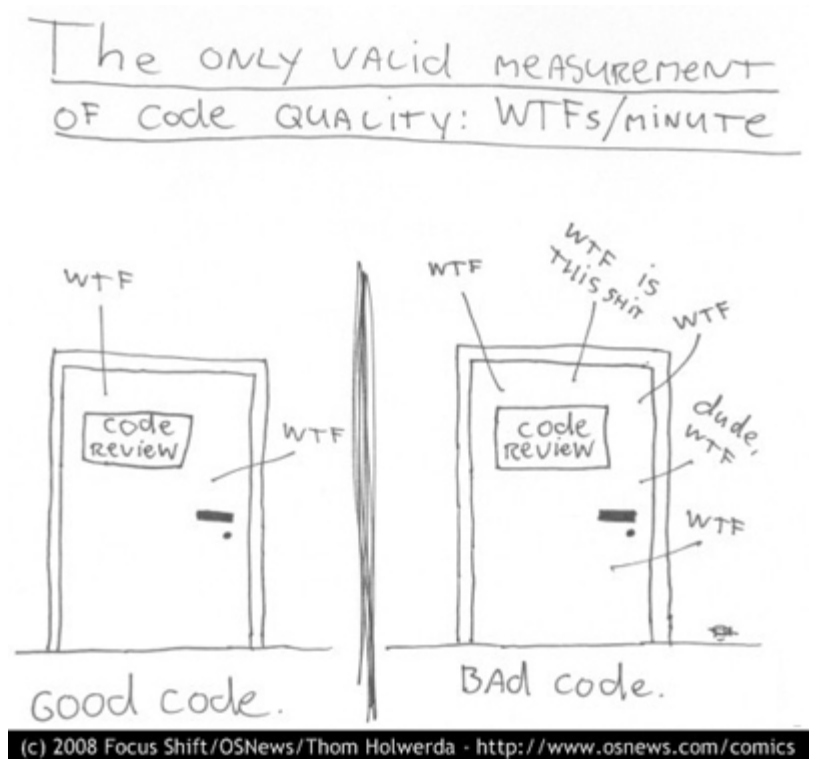
Everything we teach today will slow you down



Slow down to speed up!

Slow and steady wins the race.

$$\texttt{code\_quality} = \frac{1}{\texttt{wtf\_per\_minute}}$$



About software engineering

https://en.wikipedia.org/wiki/Software_engineering

> Software engineering is the systematic application of engineering approaches to the development of software.

> "… the systematic application of scientific and technological knowledge, methods, and experience to the **design**, **implementation**, **testing**, and **documentation** of software."

Titus Winter:

> (Software) engineering is programming over time.

About software engineering, continued

- Programming process itself is as important as writing code itself!
- Use tools and techniques to control and automate parts of the process.
- Regularly rethink your personal programming process.

Topics / Presentations

Presentation: Clean and maintainable code

- After this introduction.

Presentation: Version control systems

- Time machine for your source code.
- `git`, `subversion`, ...
- No need to keep commented out code or version information in file names.
- Retrospective understanding of changes.
- Undo changes.
- Also supports sharing code and distributed development in a team.
- Also "Hands on" during the code clinics session.

Presentation: Automated code testing

- Automates manual tests.
- **NO MAGIC BUTTON**: extra work required.

Presentation: Automated code testing example

- So called *productive code*:

```
def add(a, b):
    return a + b
```

- Add extra code which tests your productive code:

```
def test_add():
    assert add(0, 0) == 0
    assert add(0, 1) == 1
    assert add(3, 1) == 4
    assert add(-1, 1) == 0
```

Presentation: Automated code testing

- **Unit testing**
  - compare small units against expected results
- **Regression testing**
  - run system or parts and compare against a previously recorded correct result
  - important to get started with testing for an existing code base
- **Integration testing**
  - tests full system and how parts play together

Presentation: Automated code testing, continued

- **Continuous integration (CI) testing**
  - used with version control systems (more later)
  - tests run after updating code on remote repository (e.g. github.com)
  - thus tests run on a **different machine**

Presentation: Refactoring

- Techniques to improve your code without changing functionality.

Using AI for Coding

Do I still need to learn all that?

What issues can AI generated code have?

- Produces wrong results
- Code is not maintainable
- … also not by other agents
- Deletes files on your computer -> Sanboxing
- You own the code the AI generates!

Ways to use AI for coding

- Chatbox: One shot, code often wrong
- AI agent (e.g. claude, or in VS Code): multiple shots until goal is reached
- The goal must be clear to the agent! -> Planning is important!
- Ask an agent to use Test Driven Development (TDD) when you have a good plan.

Planning

https://github.com/mattpocock/skills/blob/main/skills/productivity/grilling/SKILL.m

```
Interview me relentlessly about every aspect of this plan until we reach a
shared understanding. Walk down each branch of the design tree, resolving
dependencies between decisions one-by-one. For each question, provide your
recommended answer.

Ask the questions one at a time, waiting for feedback on each question before
continuing. Asking multiple questions at once is bewildering.

If a question can be answered by exploring the codebase, explore the codebase
instead.

Write your findings in a document for future reference.
```

Where AI agents are very helpful

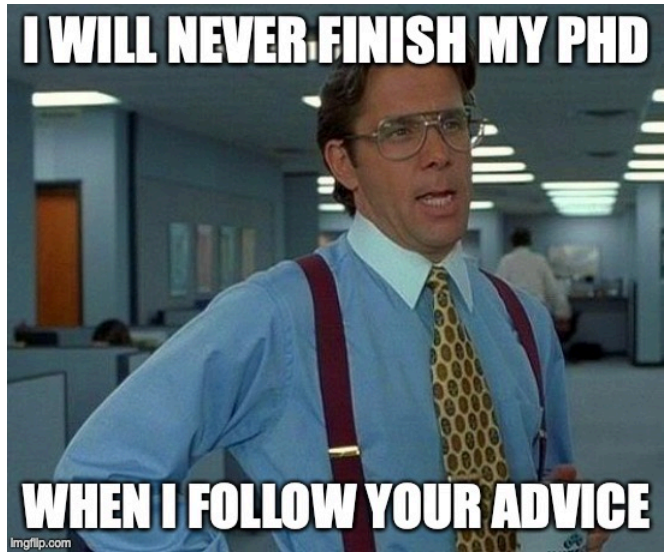
- Let an agent review your code!
- Discuss technical concepts!
- Setup a package.
- Setup and draft or polish documentation.
- Setup and draft automated tests.
- Commit messages!

What to learn in the AI era?

- Agents still use technical language and concepts.
- You need to understand technical concepts to steer an agent.
- You still need to be able to review the code.
- The focus shifts away from the smaller details.
- The future is uncertain (AI capabilites and costs!)

After the workshop

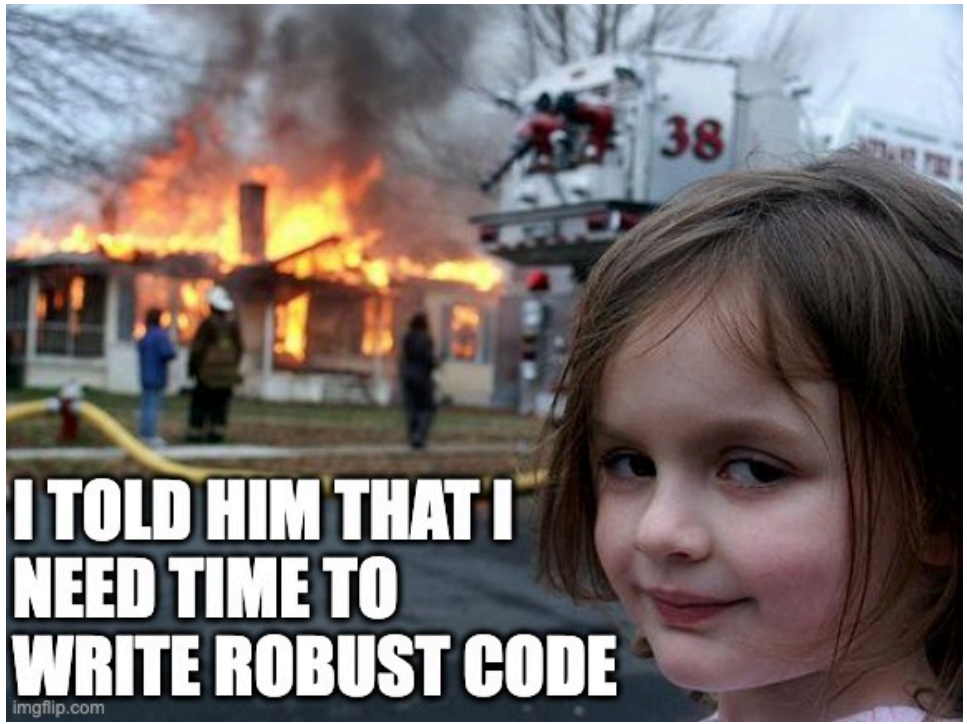
This is so much to learn



source: https://i.invalid/3dl1ks.jpg

Learn best practices incrementally.
Automate techniques and habits step by step.
Start NOW!

My professor won't give me time for the stuff you teach! Communicate risks and future costs, not technicalities.



source: https://imgflip.com/i/9yxs3w.jpg, Credit: Dave Roth.

Questions?