Welcome to CS126
Course Website

https://courses.grainger.illinois.edu/cs126/sp2022/info/

1. Info - general course info
2. Assignments - are posted here
3. Office Hours
4. Hall of Fame
5. Notes
6. Semester calendar (coming soon)
   a. Lecture topics
   b. Lecture notes
   c. In-class activities
Topics

- Online vs In-person
- About me
- What this class is about and how it is probably different from other classes you have taken.
- What a week looks like in this class
- Plagiarism and Academic Integrity
Online vs In-Person

This course is listed as an in-person course.

A new online CRN is being created to allow those who were not able to come to campus to complete the course.

I will make every effort to allow you to satisfy the course remotely.

- Starting next week, I plan to physically be in the intended lecture location (CIF)
- Lectures will be recorded
- Discussion sections are all remote

*City Scholars section is online and requires that you be enrolled in the City Scholars program
About Me

Over 20 years in game development, specifically flight simulators.
● MS Flight Simulator and subLOGIC Flight Sim 2
● Mission Studios Jet Fighter 3 and 4

Key lessons:
● Test, test, test
● Humans introduce errors
● Automate your processes
● Do not expect raw data to be correct
● Some bugs come from undocumented “limitations”
   ○ (they may not be your fault)

Headhunter (Third Party Recruiter)
Department Of Computer Science - Clinical Assistant Professor
What a week looks like in this class

Monday:
  Lecture
  PrairieLearn assignment

Tuesday:
  assignment for coding
  11:59 previous week’s assignment due

Wednesday:
  Lecture

Thursday:
  Code reviews

Friday -
  Lecture
  Code reviews
  In-class activity
  Ask-Me-Anything -
    staff answers your questions on current assignment
From Prof. Evans (previous instructor)

- Very few programs are written completely from scratch.
  - Most rely heavily on libraries, APIs, and frameworks

- Existing code is human-made and arbitrary
  - No one inherently knows how to interface to it. Which means you will have to work at learning to use someone else’s code (library, API, etc)
  - Need to be able to read documentation
    - Google and StackOverflow are your friends

- In this class, we’ll encourage you to help yourself
  - But we work hard to make sure you have the support you need
Learning to Code is Unique

Coding is a skill or talent that can be learned like any other. Many analogies may apply but none seems to fit perfectly.

Two from my experience seem to apply:

- Learning to play a musical instrument
- Sports (individual)
Metacognition

“Regulation of cognition” contains three essential skills:

**Planning:** appropriate selection of strategies and the correct allocation of resources that affect task performance.

**Monitoring:** refers to one's awareness of comprehension and task performance.

**Evaluating:** refers to appraising the final product of a task and the efficiency at which the task was performed. This can include re-evaluating strategies that were used.
What Will We Do This Semester?

- **style, refactoring, code reviews**
  - layout, commenting, variable usage and naming, control structures
- **test-driven development, testing frameworks, test coverage**
  - defensive programming, assertions, exception handling
- **design, design of routines, object-oriented frameworks**
  - design patterns, event-driven programming, MVC
- **tools: IDEs, source control, debugging, logging, Unix**
- **user interface design, prototyping, user testing**
Code Reviews

- Groups of $\leq 6$ students + 1 “moderator”
- Meets for up to hours/week (arranged time)
- Present code that you’ve written in the past week
  - Get feedback on your style & design
  - See other possible designs (pick up ideas)
  - Ask questions about the code of others
  - Practice presentation & verbal communication skills
Java is a relatively verbose language
Having a good tool accelerates routine drudgery.
Git

- **Version control systems (VCS):**
  - A practice that tracks and provides control over changes to a collection of documents/files.
  - Allows access to any prior version.
  - Facilitates collaboration between multiple developers.

- **Git: an industry-standard distributed VCS**
  - You’ll use this for developing/submitting your code.

Very sophisticated tool; we’ll only use a subset of features.
VCS concepts

- **Repository:** A collection of files under version control, along with all of their previous (committed) versions.
- **Checkout (verb):** To make a working copy on your local machine for editing/testing.
- **Commit (verb):** To take a set of file modifications and add them to the repository, usually with a descriptive message.
- **Commit (noun):** The set of changes (a “diff”) along with its descriptive message resulting from a commit (verb).
Git Concepts

- **Local repository vs. remote repository:**
  - Git lets you have as multiple related repositories on different machines.

- **Clone:**
  - Make a local repository from a remote repository.

- **Staged:**
  - Files whose changes are to be committed.

- **Push:**
  - Copying local commit to remote repository.

- **Pull:**
  - Bringing changes from remote to local repository. Implemented by a “fetch” then a “merge”.
Version Control, why do we care?

Single Developer:
- Most things worth doing are too big to do all at once.
- Break large projects into small steps
- *Design, implement, test, debug, commit each step*
- Have access to every working version through VCS
- If things stop working
  - inspect the differences between current and last working versions.
  - revert back to last working version (e.g., throw away changes)

Multiple Developers:
Allows multiple developers to coordinate edits to a shared set of files
What this class is NOT about

This is NOT a ‘Computer Science’ class

● This is a programming class
● The goal is to develop skills and habits
● Don’t hate CS even if you hate this class
● Do not overly concern yourself with execution efficiency (why not!?!?!)
  ○ It’s a matter of priority
  ○ Multiple things affect execution speed

This class SHOULD help you in your ‘Computer Science’ classes

● Alleviate the low-level programming struggles
● Focus your attention on the big ideas!
“The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds his castles in the air, from air, creating by exertion of the imagination. Few media of creation are so flexible, so easy to polish and rework, so readily capable of realizing grand conceptual structures....

Yet the program construct, unlike the poet's words, is real in the sense that it moves and works, producing visible outputs separate from the construct itself. […] The magic of myth and legend has come true in our time. One types the correct incantation on a keyboard, and a display screen comes to life, showing things that never were nor could be.”

— Fred Brooks
Pragmatically, programming is the tool that computer scientists use to collect, analyze, and visualize data, automate tasks, make products, mechanically prove theorems, and build tools. As lawyers write prose and architects build models, programming is the underlying tool of the computer scientist.
“The gap between the best software engineering practice and the average practice is very wide—perhaps wider than in any other engineering discipline”

— Fred Brooks

“The original study that found huge variations in individual programming productivity … studied professional programmers with an average of 7 years' experience and found that the ratio of initial coding time between the best and worst programmers was about 20 to 1; the ratio of debugging times over 25 to 1; of program size 5 to 1; and of program execution speed about 10 to 1.”

— Steve McConnell (Code Complete, 2nd Edition)
A solid, concise book on software construction.

Less than $30 on Amazon prime (or $16 for a Kindle version).

Check out reviews on Amazon about how good this book is.
1. Your friend offers to send you the code for the MP so you can hang out.
2. Your friend is having difficulties starting the assignment, so you give them a general overview of what you did.
3. Your friend is having difficulties starting the assignment, so you show them the first few methods you wrote.
4. Your friend is having difficulties with the assignment, so you help them debug their code.
5. Your friend just does NOT have time this week for CS 126 and they ask you to give them your code. Don't worry they'll change it up a little.
6. You get stuck on how to do something on an MP, so you google it and copy ~30 lines from the very helpful first StackOverflow post.
7. You get stuck on how to do something on an MP, so you google it and add a comment citing the code you took from StackOverflow. You can explain what the code does in code review.
8. You get stuck on how to do something on an MP, so you google it and use code from a past student's incredibly handy GitHub repository! That student's intro programming assignment surely looks impressive to recruiters!
9. You think you're smarter than course staff, so you "borrow" your friend's code and attempt to modify it enough so that you don't get caught.