# CMSC 398z
# Effective use of AI Coding Assistants and Agents

Bill Pugh and Derek Willis

Sept 12th, 2025

# Moving through material and leveling up

We didn't know what range of experience students would have with Python and AI coding tools

We have students across a range of experience, but a lot of students with significant experience

We are going to be leveling up, will get to more challenging tasks

- Have a bigger project planned for next week, can drop coaching mode

Students who are still just getting started with Python and AI coding tools are encouraged to spend some additional time playing with them, have the option to do that rather than reading for next week

# longer
# Instructor
# Introductions

# Disclaimers

Unlike many courses, we aren't teaching fundamental truths or knowledge

We are teaching you how to use technologies developed by large tech companies and open source projects, and having you read work by pundits and bloggers.

All of whom have a vested interest in getting you to adopt their technologies and follow them.

Nothing we say in this course should be interpreted as saying

"This is the best tool to use, or the best blogger to follow"

Just want we think will be useful in this class at this point in time

# Staying on top of the field

You might prefer to generally avoid Twitter/X, but it is a great source of information.

We've created a twitter/X list if you just want to follow the people we recommend, including

- Simon WIllison (AI coding tools)
- Ethan Mollick (Impact of AI on workspace and higher education)
- Yan LeCunn (AI researcher @ Meta,  LLM as AGI skeptic)
- Gary Marcus (AI Skeptic)

Some of them also post on BlueSky, Threads and/or LinkedIn

# TechMeme

TechMeme is a tech news aggregator

Followed by a lot of tech CEO's, pretty much everyone in Silicon Valley

Not just AI news, covers all of tech

Just one page of links to stories, visit once per day is plenty

Founded and funded in 2005 by Gabe Rivera, UMD CS PhD '01

TechMeme podcast, now branded as TechBrew Ride Home, it a great way to stay informed

# Hardfork podcast from NY Times

Good and entertaining podcast from the NY Times

> *Each week, journalists Kevin Roose and Casey Newton explore and make sense of the rapidly changing world of tech.*

Covers all tech news, but a fair bit of coverage on AI

Not a lot on AI coding tools

# Coders vs. Software engineers

The strategy at the UW's [CS department] is to graduate nimble problem-solvers who understand computing fundamentals,..

“Coding, or the translation of a precise design into software instructions, is dead, AI can do that. We have never graduated coders. We have always graduated software engineers.”

Geekwire, July 10, 2025 - In next week's reading

A software engineer is someone who understands how to build reliable systems that solve problems with software. – Bill

# Unit tests are important

You will hear us say over and over again that unit tests are important

They have *always* been important for real software development.

Unit tests allow you to:

- to localize bugs
- test functionality before you have enough complete to do a system test

They are even more important in the age of AI coding tools

Why? Discuss

# Google Engineering’s Beyoncé Rule

- Google has a huge monorepo
  - containing many projects and utilities
- They regularly update and refactor code
  - including widely used libraries
- When a change is proposed for the monorepo,
  - they rerun all unit tests that call the changed code
- If the change would break any unit test,
  - it needs to be resolved before the change is checked in
- If the change only breaks your system tests, that's on you

# Why unit tests are important in the age of AI coding tools

Unit test protect your code after it is written

Software experiences change after you "*complete*" it.

- either directly, perhaps by other people,
- or through libraries that are updated

With AI coding tools, you have someone changing the code out from under you, sometimes making mistakes

# Notes from using Copilot

It will often mess code up when you ask it to make file-wide edits, such as changing formatting or refactoring

- Run unit tests often.
- Rewind conversation or use git to restore mistakes

# Unit tests in Python

You can put simple unit tests directly in the docstring for a function

```
uv run python -m doctest myFile.py
```

You can also define separate files that hold unit tests

```
uv run python -m unittest myTests.py
```

AI coding tools are good at turning the cases you come up with into test cases, and decent at coming up with test cases (but will miss some)

There have been horror stories about AI coding tools deleting tests in order to get tests to pass.

# Making sure your tests test what matters

- If a function returns a list of results, does the order matter?
  - If you are doing something like enumerating the values in a hash table, order can easily change
  -
- Are you comparing textual output?
  - Does the exact output format matter?
  - Does the order of items matter?
  - Are the tests looking at logging output?

Flaky tests:

- Tests that sometimes pass, sometimes fail
- Big problem in large projects
- Submit server flags them

# Some variations on unit tests

It can be challenging and time consuming to come up with good unit tests

Regression tests:

- Run functions or the system on sample data, record output
- Regression tests involves rerunning code, seeing if anything changed
    - Change might be an improvement

Differential testing:

- Implement two versions of an algorithm: An elegant efficient algorithm, and a simple brute force algorithm.
- Compare them on bunch of inputs, including randomly generated inputs

# Fuzz testing

Generate lots of random input data

- maybe complete noise
- maybe shaped to be plausible

See what happens when you run code against lots of test data

Use runtime checking tools such as valgrind if applicable

If you generate a memory error, you may have found something that can be turned into an exploit

# Using a verifier as a test oracle

On some problems, it might be very hard to find a solution

- For example, for an NP-hard problem

But if a solution is proposed, much easier to verify that it is valid

You can use a verifier as a test Oracle

- Whenever running any unit tests
- Including on random test cases and fuzzed data

# Code coverage

Various tools that instrument your code, report what parts of your code were covered (executed) during the run.

- Normally, just statement coverage, although branch and path coverage is also possible

One measure of the quality of your tests is their code coverage

- Anything that isn't covered isn't tested, so the tests don't tell you if that code is correct
- But you can have 100% code coverage and still have significant gaps

# AI Coding tools and test generation

AI tools pretty good at coming up with brute force implementations and verifiers

- Even if you wouldn't trust them with the primary task, at least not without careful review
- Having these is also helpful in checking test cases generated by AI tools
  - which can be wrong

Visual Studio 2026 Preview provides agent with access to code coverage information

- Not available yet on MacOS

# Reviewing submissions

We looked at a number of the project submissions, provided code reviews for some of them

- You don't need to provide responses to the code reviews if you got one

Found patterns worth discussing in class

If anyone would really like us to review their code, we will.

Side note: I knew prepping for a new class was a lot of work, but wow….

# Observations from looking at submissions for playWordle

- Some people had only a main method, much harder to test
- Some people had a analyze_guess function that returned a colored word
  - with escape characters provided by termcolor
- Having a function that return a textual description of how to color a guess makes it easy to test (e.g., 'YYBGB' or ['Y', 'Y', 'B', 'G', 'B']
  - have separate function that takes guess and description of how to color it, returns colored text
- One person didn't use termcolor at all, AI chat provided the escape characters to use
- Several students didn't take the answer word from the command line
  - Not a bad wordle implementation, but not the one asked for

# Providing context

I suspect that some of the issues I saw results from the students just describing the task to the agent, and the agent never saw the instructions in the README.md file

Approaches:

- Provide files as context to the agent. It doesn't always figure it out automatically
- Alternatively, read instructions closely and instruct the agent
- Also, review the code that is generated and check that is accomplished what is required.

# Readings done this week

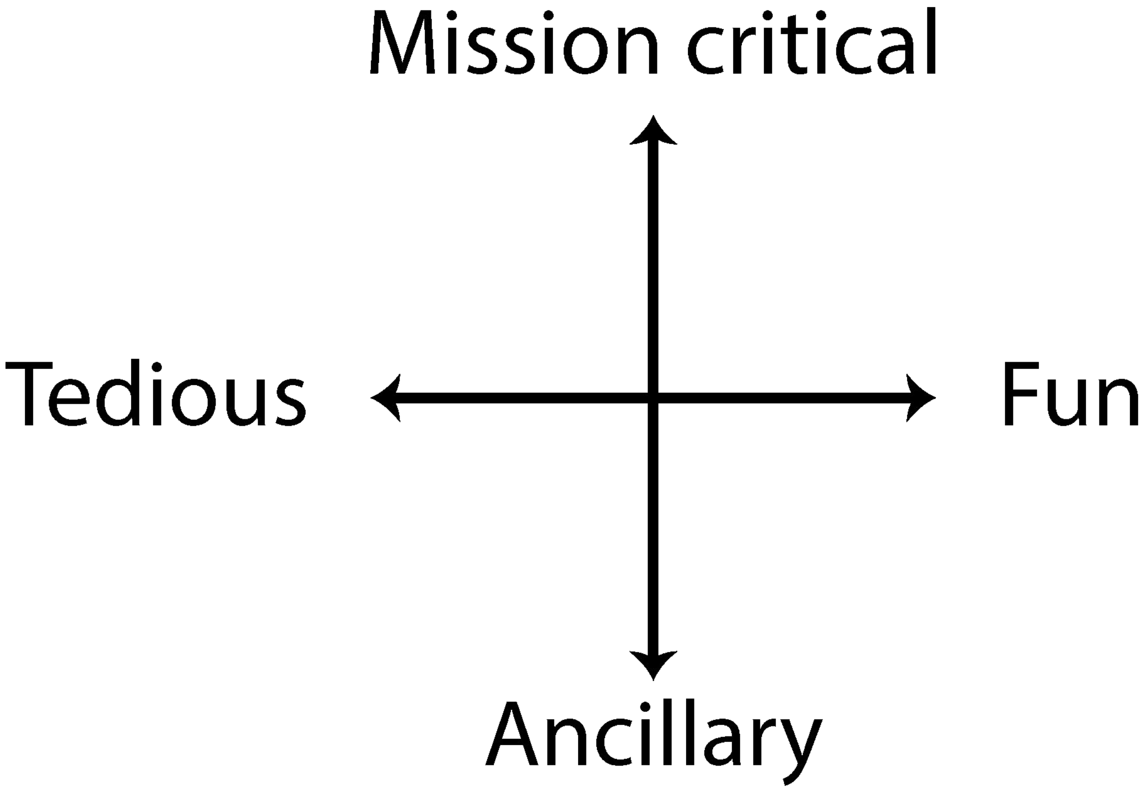
Plan for discussion

- discuss around each table, 5 minutes per paper
- discuss items from survey or anything you want to discuss
- Every table should come up with a thought or question they want to share
- You can either raise your hand and one of us will come to you
- Or at the 3 minute mark, Derek and Bill will start coming to tables and asking you for your statement/question

# NYTimes How Do You Teach Computer Science in the A.I. Era?

# The Fly Blog - My AI Skeptic Friends Are All Nuts

The Fly Blog - My AI Skeptic Friends Are All Nuts



Mission critical

Tedious

Fun

Ancillary

# Using AI in each of those categories

Mission-critical / tedious:

- Humans are not good writing tedious code
- "There should be a nice reliable deterministic tool for this"
- LLM's can often do well, but need careful review

Mission-critical / fun:

- Places where humans are most useful, and most willing
- LLMs can be useful partners, help checking plan and code, writing test cases

# Ancillary cases

For Ancillary code, "Vibe coding" might be more appealing

- not having to spend a lot of time closely reviewing code
- testing still important
- For fun/ancillary components, can enjoyable to have LLM as a partner
    - But perhaps delegating many of the details to the LLM

# Harper Reed - An LLM Codegen Hero's Journey

# Coding project for today

Markov chain text generation

crude but simple predecessor to LLMs

Train on some source material

Start predicting some text

Based on the last-n words, choose randomly from the words that follow next in the training data.

In Alice's Adventures in Wonderland, "a sort of" is followed by mixed, circle and knot

# Completely implemented for you

And provided AlicesAdventures.txt and Sherlock.txt

But absolutely no documentation

You have to provide docstrings for the class MarkovModel and every function

Also document every field/attribute of MarkovModel

Document any if statement where it isn't obvious when it would fire and why it is there

- There is at least one such if statement that you will likely have to use the debugger to figure out

# Still have coaching mode in place

You can ask the agent what the magic word is. This is a way to verify that the agent has read the custom instructions and has them in context

We will likely be dropping the coaching mode instructions next week, if you are tired of it, keep using it this week

# Additional projects to work on during class

**Word Grid** - Project Derek and Bill started on

In week 1 directory, both starter project and a completed project

**Wordle helper** - help you figure out what words are still valid in wordle

In week 2 directory

Using an old version of the wordle word list. This current NYTimes one is not public and they continue to add words to it.

You can take this to several different levels, described in README.md

Use/adapt code from week 1

# Start coding in pairs

Different pairs than last week.

You can submit individually or a single shared submission

# Project preview for next week

Do not start coding this, or ask a LLM how to code it, until class next week

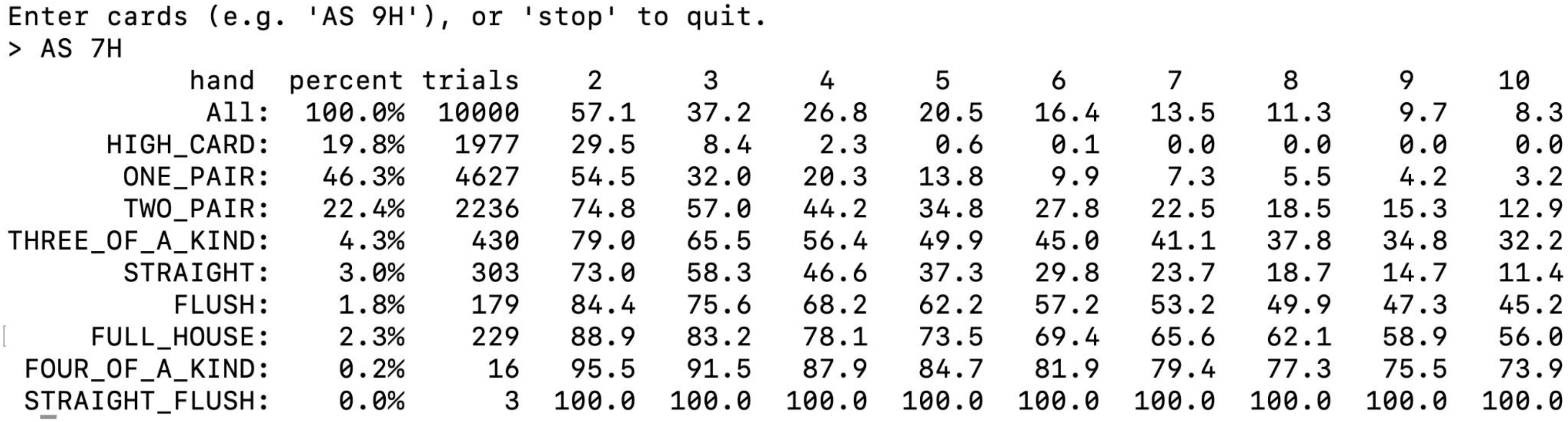
But you will likely find it useful to think about how to approach this problem

Can also do research on required background knowledge

# Texas Hold'em analysis

- Figure out the best poker hand you can make from 7 cards
- Level 2:
- User provides from 0-7 poker cards
  - First two are your hole cards
  - the next 0-5 are community cards
- Run many simulations
- If user provided only the hole cards, random deal 5 more cards from the remaining deck, determine frequency for best hand
- Level 3: also deal to other players, figure your chance of winning

# Level 3 poker analysis



```
Enter cards (e.g. 'AS 9H'), or 'stop' to quit.
> AS 7H
           hand  percent trials    2      3      4      5      6      7      8      9     10
           All:  100.0%  10000   57.1   37.2   26.8   20.5   16.4   13.5   11.3    9.7    8.3
      HIGH_CARD:   19.8%   1977   29.5    8.4    2.3    0.6    0.1    0.0    0.0    0.0    0.0
       ONE_PAIR:   46.3%   4627   54.5   32.0   20.3   13.8    9.9    7.3    5.5    4.2    3.2
       TWO_PAIR:   22.4%   2236   74.8   57.0   44.2   34.8   27.8   22.5   18.5   15.3   12.9
THREE_OF_A_KIND:    4.3%    430   79.0   65.5   56.4   49.9   45.0   41.1   37.8   34.8   32.2
       STRAIGHT:    3.0%    303   73.0   58.3   46.6   37.3   29.8   23.7   18.7   14.7   11.4
          FLUSH:    1.8%    179   84.4   75.6   68.2   62.2   57.2   53.2   49.9   47.3   45.2
     FULL_HOUSE:    2.3%    229   88.9   83.2   78.1   73.5   69.4   65.6   62.1   58.9   56.0
 FOUR_OF_A_KIND:    0.2%     16   95.5   91.5   87.9   84.7   81.9   79.4   77.3   75.5   73.9
 STRAIGHT_FLUSH:    0.0%      3  100.0  100.0  100.0  100.0  100.0  100.0  100.0  100.0  100.0
```