Adventures in Scientific Computing: Python Programming, Paradigms, and Pizzazz



Johnny Wei-Bing Lin

University of Washington Bothell and
North Park University

Photo: Pexels.com, Pixabay. Thanks for conversations with Murat Ayhan, Stephanie Cheung, and Yusuf Pisan.



Photo: Pexels.com, Pixabay.

Preview

- **Overcoming challenges** in learning scientific programming.
- The importance of **programming paradigms.**
- Initial explorations in **vibe coding.**

My funny career journey

- B.S. Mech. Engr., M.S. Civil Engr.-Water Res. (6 yrs)
- Hydrologist (2 yrs)
- Ph.D. Atmospheric Sciences (6.5 yrs)
- Two postdocs (5 yrs)
- Faculty: Physics (9 yrs)
- Faculty: Computing and Software Systems (12 yrs)
- Number of programming courses after high school: 1.



Photo: Pexels.com, Markus Spiske.



Photo: Pexels.com, Pixabay.

Programming is hard, especially when you first learn it

The “Rainfall Problem”: “Write a program that repeatedly reads in positive integers, until it reads the integer 99999. After seeing 99999, it should print out the average.”

Soloway et al. (1983), described in Guzdial (2010)

Student Group	% Correct Using Raw Pascal
CS 1	14%
CS 2	36%
Systems Course (Juniors and Seniors)	69%

Photo: Pexels.com, Pixabay.

Substantial numbers of students do not pass CS 1

- Meta-analysis of studies of CS 1 courses from 1979-2013 (Watson & Li 2014).
  - **Overall global passing rate: 67%.**
  - No statistically significant variation over time.
  - No statistically significant variation amongst the 4 countries making up 80% of the sample.
- Bi-modal distribution commonly observed in classes (Guzdial 2010).

Photo: Pexels.com, Pixabay.

It's tough to be a novice scientist programmer

- Most scientific computing resources assume you already can program.
- Introductory CS resources often focus on CS puzzles.
- **We do not know of one simple, effective, scalable process to learn programming.**

Photo: Pexels.com, Pixabay.

Tip #1: Learn how to break apart a task

- **Do not start with writing code.**
- Outline the steps of the problem in normal English or Chinese.
- Write the outline by hand.
- Make the outline have sub-levels.
- Avoid assuming there's a single function to solve the problem.

Photo: Pexels.com, Pixabay.



Tip #2: Don't assume programming makes sense

- Variables behave differently than in math.
- Defining and calling a function are different things.
- Beginning students don't get what a function return value is.
- **Be patient with yourself.**



Photo: Pexels.com, Pixabay.

Tip #3: Practice line-by-line code reading

- Ask what **each expression** in a line of code returns.
- For **each line** of code, ask:
  - What are the pre-states.
  - What are the post-states.
  - What did the line change.
- Learn how to make a **handwalk table** of variables.

Photo: Pexels.com, Pixabay.



```
1  def add_up(in_list):
2      if len(in_list) == 1:
3          return in_list[0]
4      else:
5          return in_list[0] + \
6                 add_up(in_list[1:])
7
8  print(add_up([2, 4, 6]))
```

Call Level	Line #	`in_list`	Returns
0	1	`[2, 4, 6]`	N/A
	4	Select `else` option	
	5-6	`[2, 4, 6]`	`2 + add_up([4, 6])`
1	1	`[4, 6]`	N/A
	4	Select `else` option	
	5-6	`[4, 6]`	`4 + add_up([6])`
... continued ...			



The importance of programming and software engineering paradigms

Photo: Pexels.com, SpaceX.

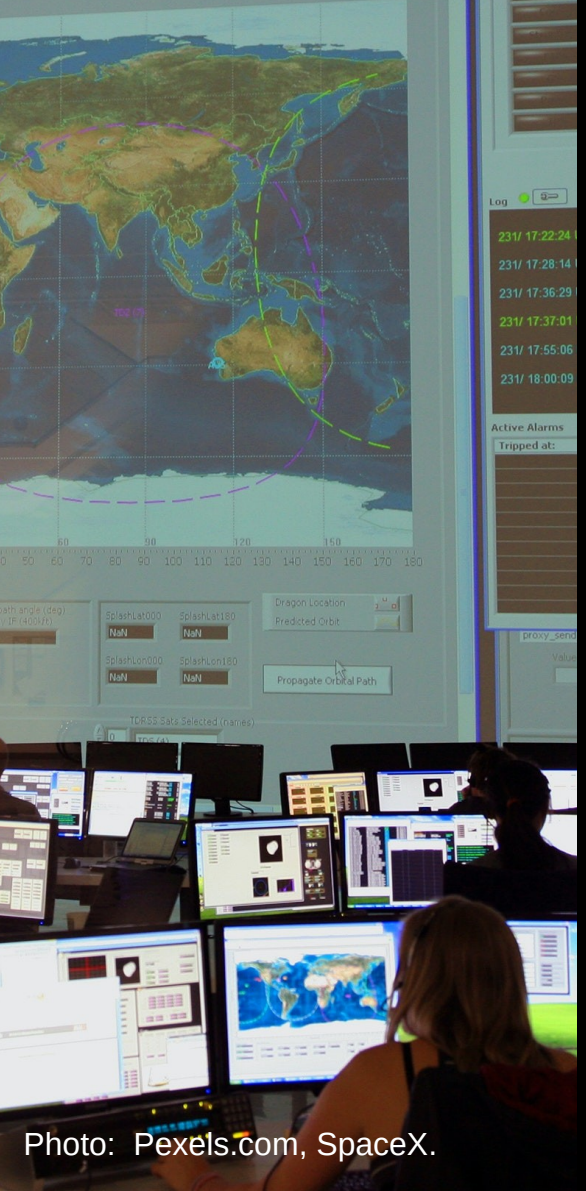


Photo: Pexels.com, SpaceX.

Paradigms are crucial for learning scientific computing

- A paradigm is the **way we think about a system:** what it consists of, how it behaves, how we interact with it.
- Ex. from physics: Newtonian paradigm for physical systems:
  - Newton's Laws of Motion.
  - Conservation laws (energy, momentum, mass).
- Some programming paradigms: **Variables and data structures.**
- Some software engineering paradigms: **Incremental and iterative development.**

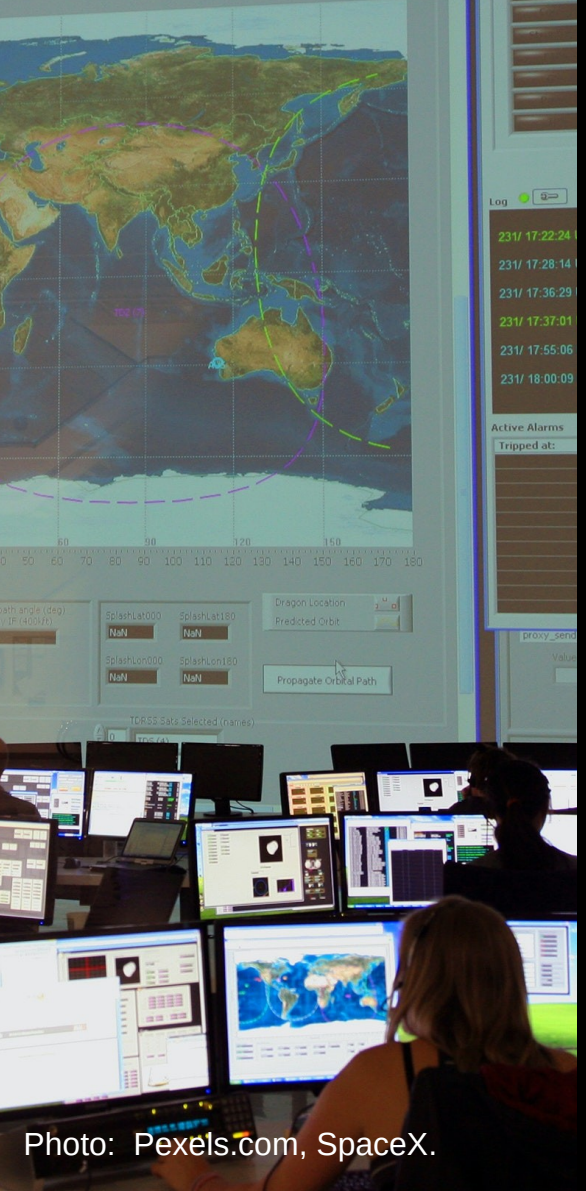


Photo: Pexels.com, SpaceX.

Programming paradigms tell you how to store and refer to data

- Variables
- Arrays: Element-wise handling, array syntax handling
- Hash map/dictionaries
- DataFrames: Hash map/dictionaries + arrays
- **Choosing the right programming paradigm for your problem** can make your code faster, more concise, and more readable.

Live code examples

- Variables
- Arrays: Element-wise handling, array syntax handling

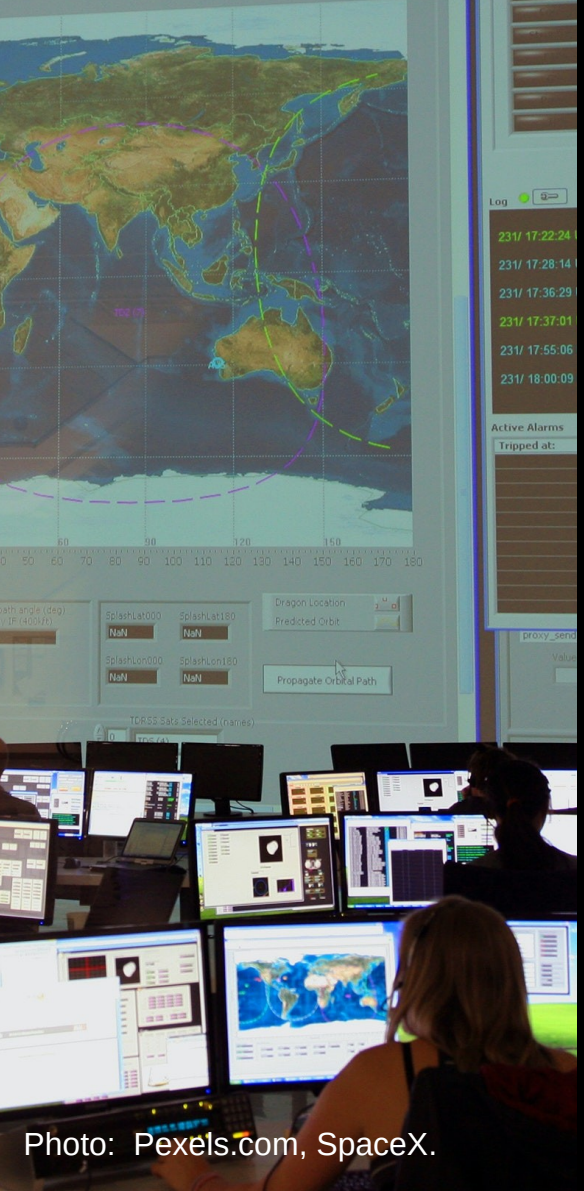


Photo: Pexels.com, SpaceX.

Software engineering paradigms enable you develop programs that work

- Writing code is only one part of developing software.
- How do you know your changes didn't break your program? Use **version control.**
- How do you know it works? You have to **test it.**
- How do you collaborate with other developers? Use a **remote repository.**
- How do you successfully merge your contributions with those of others? Use a **continuous integration (CI)** pipeline.

What is version control, a local repository, and a remote repository

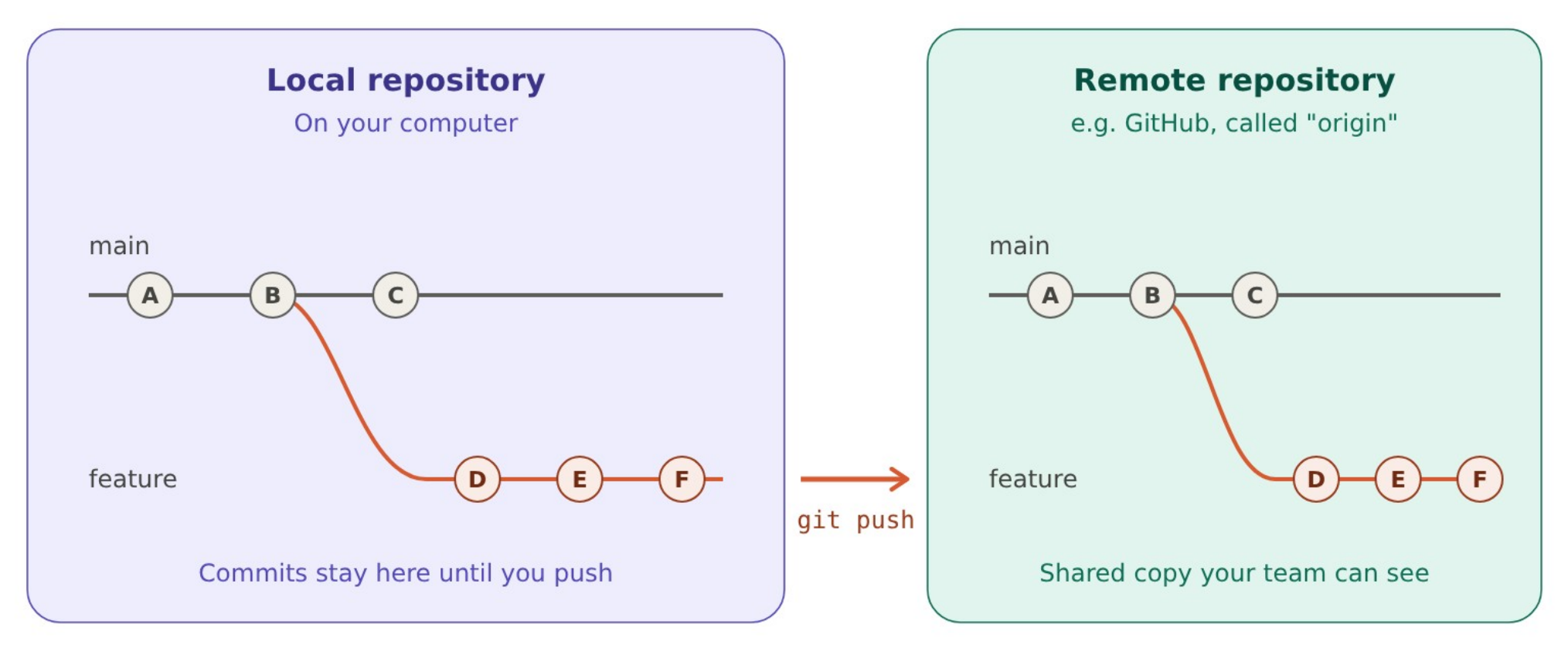


Image by Claude Opus 5.5.

A basic software development workflow

- Write in small chunks you store in a version control system (git).
- Write unit tests that check your code works correctly.
- After all tests pass, request your code be merged into remote repository that contains the contributions of the entire team (github).
- Check the team’s tests all pass on your code via the CI pipeline.
- Accept your changes into the remote repository if tests all pass.

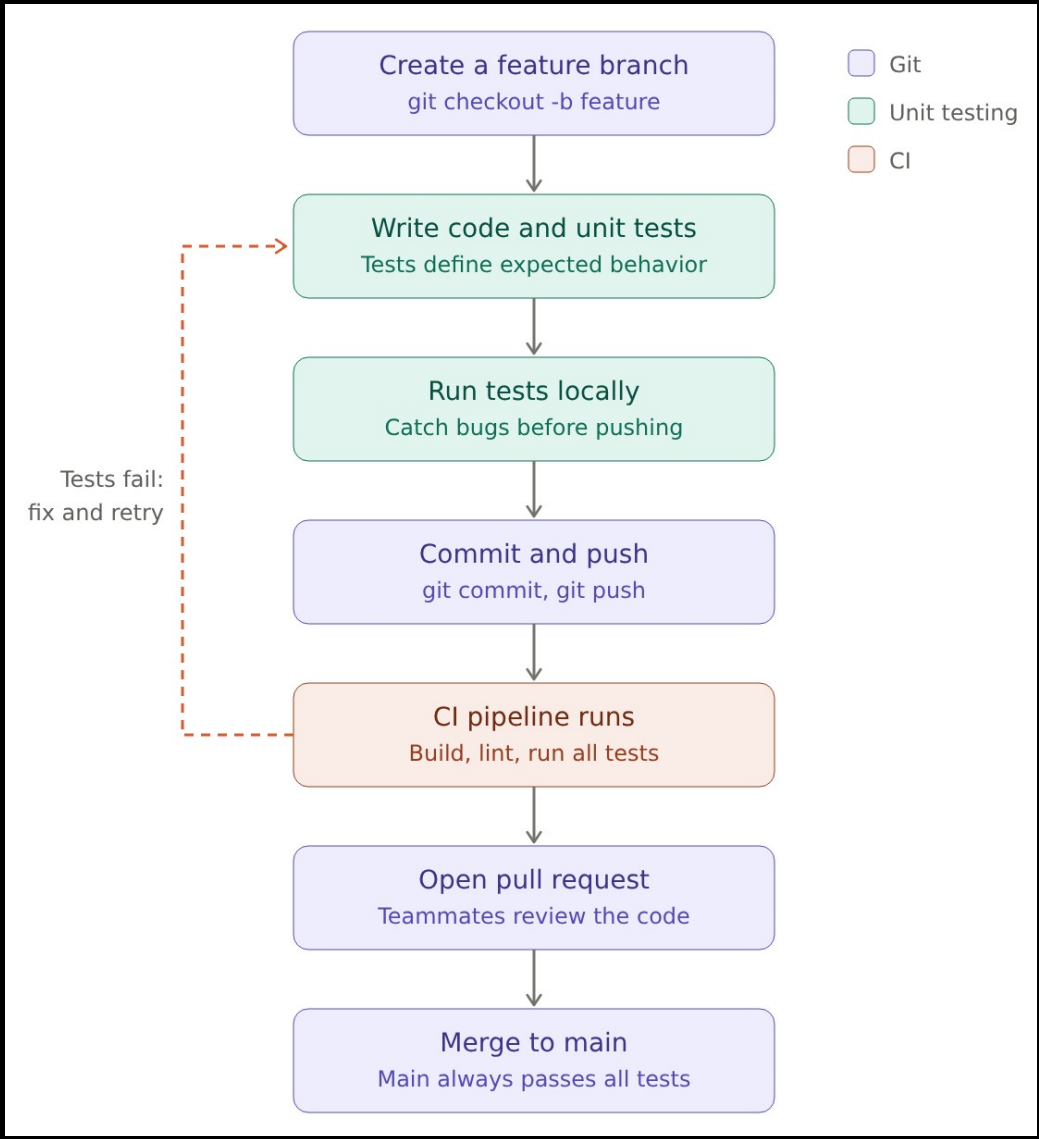


Image by Claude Opus 5.5.



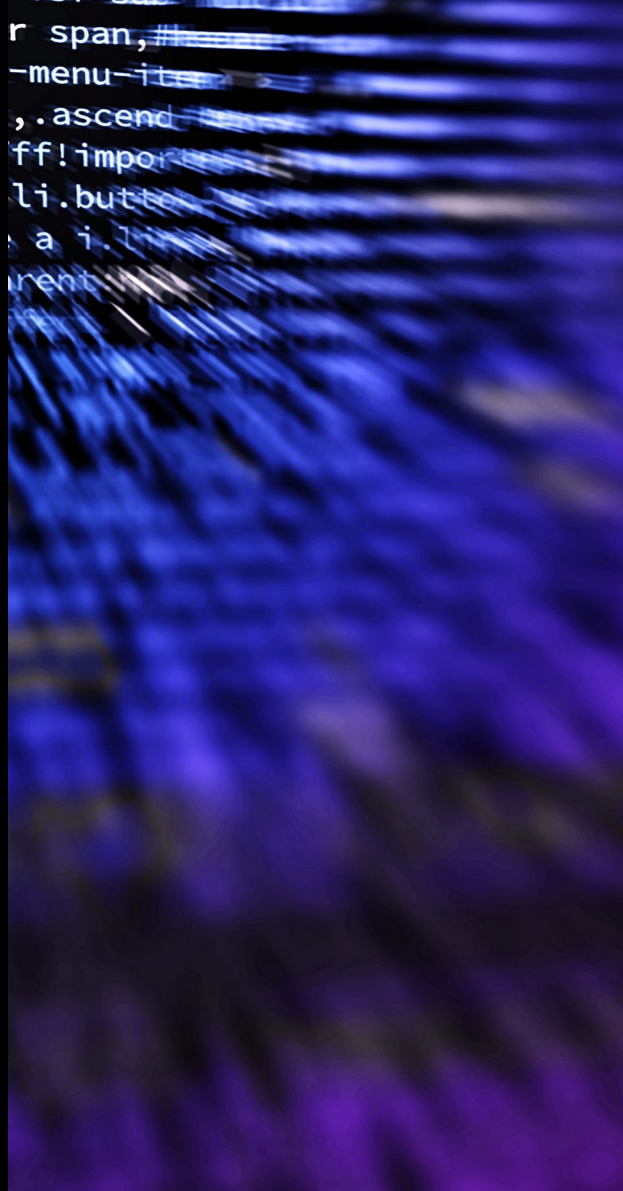
Initial Explorations Into Vibe Coding

Photo: Pexels.com, Negative Space.

Using Claude Code to write pblearn

- Vibe coding = using an AI to program specs written in a natural language (Wikipedia)
- Program to help students demonstrate learning under a proficiency-based model.
- Claude is a *far* better programmer and designer than me.
- I can do in 1 day what it would have taken me up to 2-4 weeks to do.
- *But,* it still needs your help. Paradigm: You're a senior developer and Claude Code is your junior developer. It knows a lot but you do too. Work together.

Photo: Pexels.com, Negative Space.



How do you help Claude Code help you

- Ask it to develop small, discrete chunks.
- Practice “documentation-driven development”.
- Discuss things with Claude:
  - Describe the situation, ask for its advice, and ask for a proposal.
  - If it gives you a choice, ask it for pros/cons of each option.
  - This helps you plan better.
- Ask Claude to test your program’s functionality. It does some testing but it can’t think of everything.

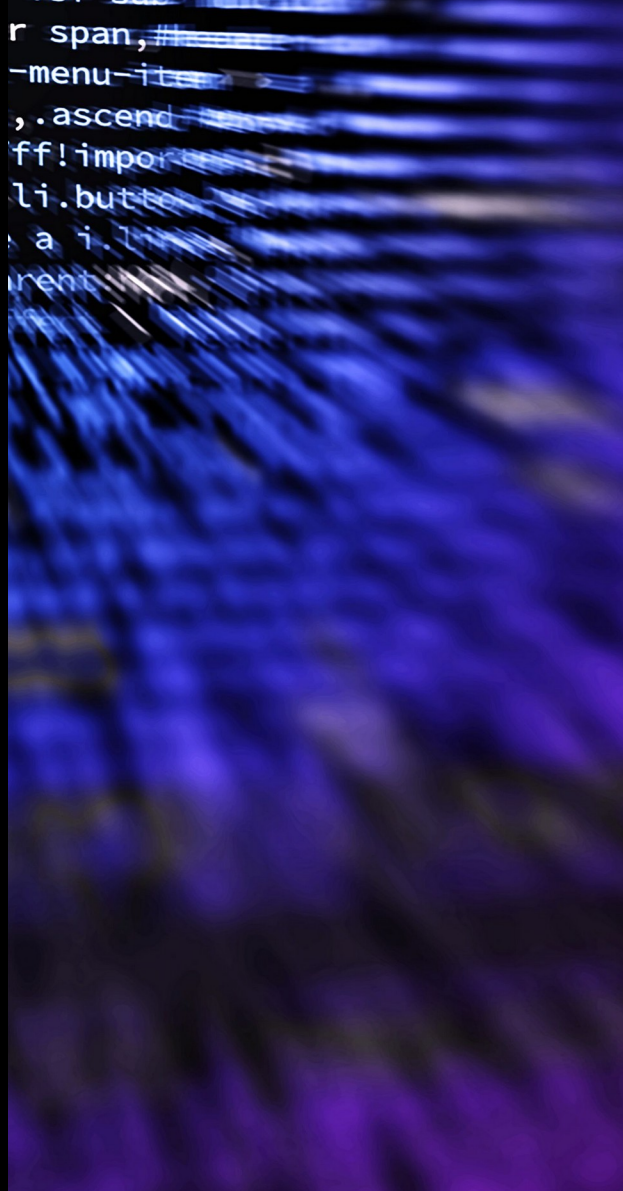


Photo: Pexels.com, Negative Space.

How do you help Claude Code help you (cont.)

- You don’t have to test each line of Claude’s code but you need to test the functionality of what Claude produces.
- Ask it to enforce your style preferences (e.g., docstrings, number of characters per line, etc.)
- Write your instructions out in clear language. Be unambiguous but not overly wordy.
- Ask Claude to create an LLM Wiki for the project where Claude documents the development process. This gives its thinking more persistence. Here are the specs by Karpathy: `https://tinyurl.com/llm-wiki`

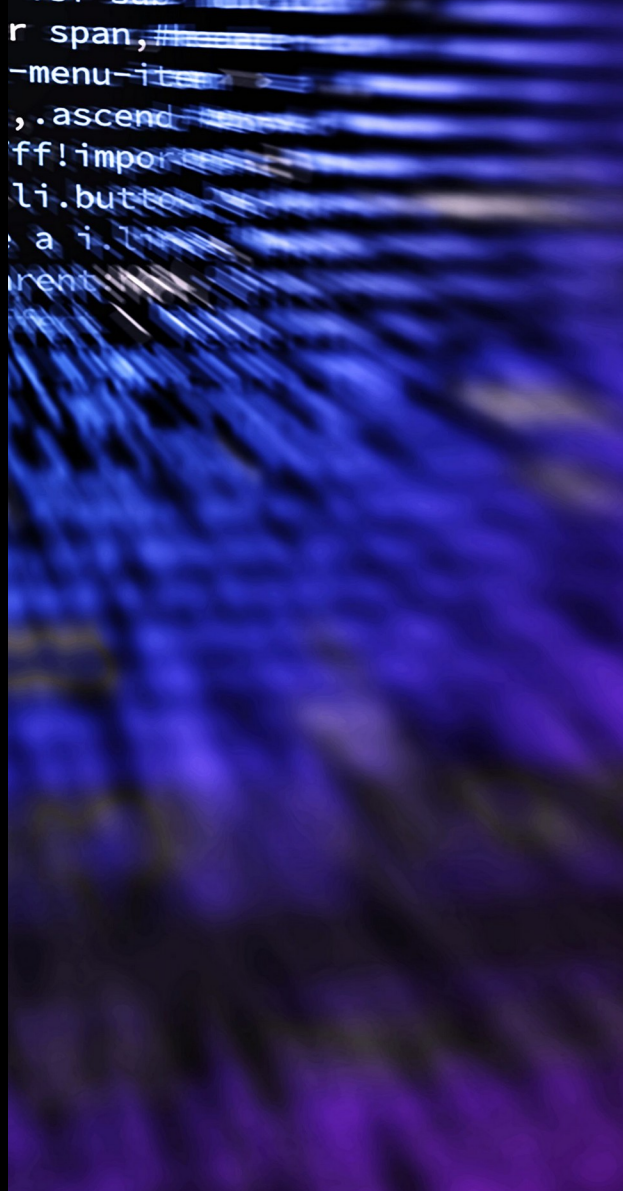
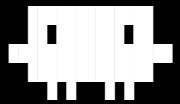


Photo: Pexels.com, Negative Space.

The project description I gave Claude ...

I want to develop a software tool using Python called pblearn that will enable students to work on auto-graded exercises and assessments using a proficiency-based (a.k.a., mastery-based) learning pedagogy. While such software currently exists (both commercial and open-source), the tool will be unique in the following ways: (1) it will enable teachers to require students to submit handwritten work to the tool, and (2) it will be able to automatically generate a properly scaffolded tree of assessment dependencies given user-chosen set(s) of questions, thus enabling students and instructors to combine learning resources from multiple sources. The pblearn system will consist of two components. The first component is the pblearn program itself which will be open-source (under a BSD or similar type of license) and designed to run on the student's computer (i.e., not as a web application). The second component is the question modules for the assessments. I plan to include one set of modules with the program that will also be released using an open-source license, but the program will be able to incorporate question modules designed by others that may be copyrighted and/or distributed via the web (whether via a paid subscription or not). The first modules will be written for introductory programming courses, but I plan to utilize packages and/or frameworks (e.g., Markdown, reStructuredText, etc.) that will enable the program to support courses in other disciplines.

… and what Claude gave back to me



```
Claude Code v2.1.278
Sonnet 5 · Claude Pro
~/unison_shared/uwb/work/prog/python/pblearn

❯ @project_descrip.txt build a working prototype of this
  ⎿  Read project_descrip.txt (2 lines)

  Read 2 files, ran 2 shell commands (ctrl+o to expand)

● Now the sample modules.

  Ran 2 shell commands (ctrl+o to expand)

● I built a working prototype of pblearn in the project directory. It uses only
  the Python standard library and needs Python 3.8 or newer. The 8 unit tests
  pass. I also ran goal add, plan and next from the command line; I haven't run
  an interactive practice session by hand.

  [... JWL snip of Claude’s description of what was and was not built …]

✻ Baked for 2m 58s · done Friday, Sep 18, 9:28 PM
```

An example dialogue with Claude about a possible code change

```
❯ the problem i'm thinking about is what happens if i download another author's
  module.  if it is also a programming module, it probably has similar topic
  names (e.g., for_loops).  is there a way i can add another author's module to
  work with my existing modules?  is there a tool i could create that would
  help this merge?  maybe if the dependency tree had certain properties?

  Read 1 file (ctrl+o to expand)

● Yes to the first question — nothing stops you from loading two
  independently-authored modules together (that's what --modules/modules.path
  are for); the real risk is concept-name collisions, not question-id ones,
  since concepts are deliberately unnamespaced (single lowercase words like
  for_loops, output) so modules can interoperate without coordination — that's
  the same mechanism trace_practice uses to require functions from intro_python.
  So "two modules teach the same concept name" isn't inherently a bug; pool
  selection (2 of output) relies on exactly that. The problem is you can't tell
  from the name alone whether an overlap is intentional reuse or an accidental
  collision between two unrelated authors.
```

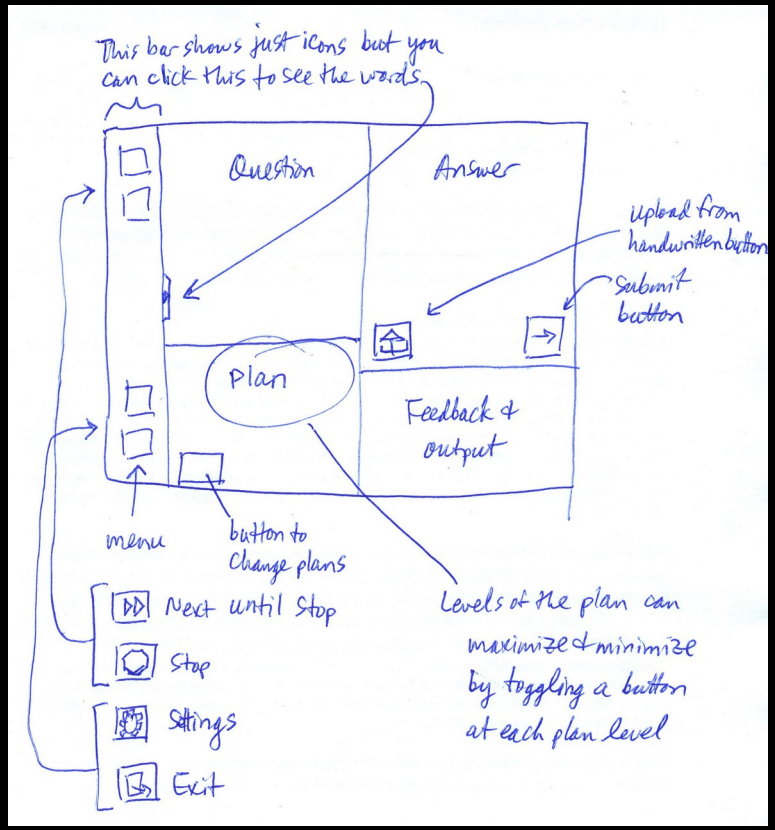
An example dialogue with Claude about a possible code change (cont.)

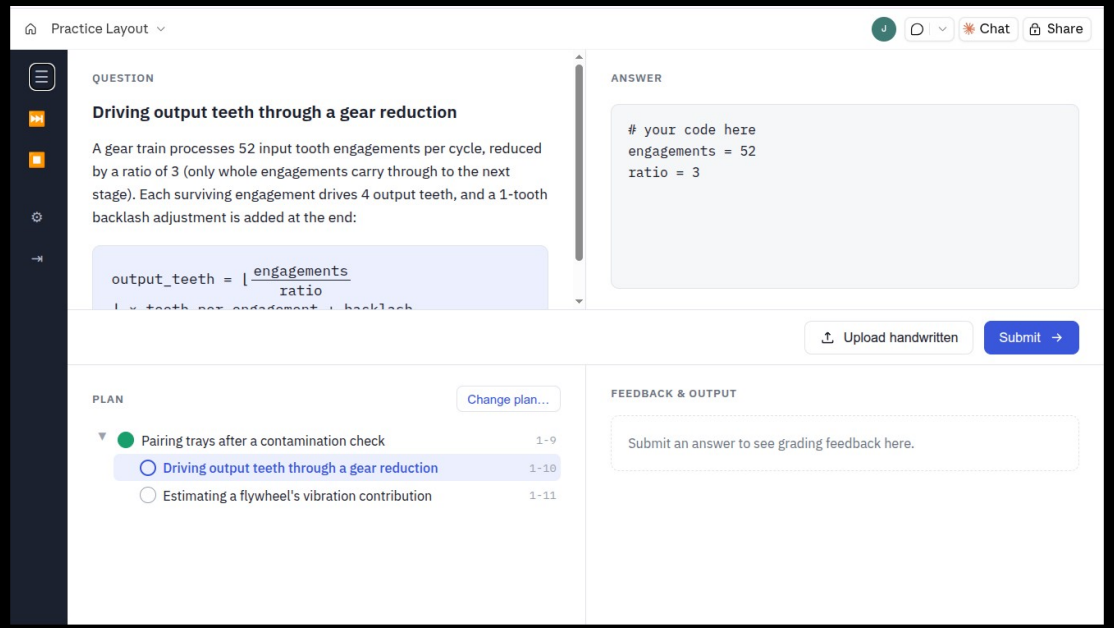
```
  That asymmetry is why I'd aim the tool at detection, not prevention: pblearn
  modules --full already prints each module's teaches: list, and
  Library.concepts (library.py:29) already maps each concept to its provider
  question ids, each with a .module. A small addition — flag any concept whose
  providers span more than one distinct module — would surface exactly the
  "shared concept" cases worth a human glance, while leaving same-module pools
  alone. That's maybe 15-20 lines in cli.py's modules command, no new data
  structures needed. Want me to build that?

✻ Baked for 49s · done Saturday 10:26 PM

❯ yes, build that
```

Claude can interpret a sketch you provide as input





Images by Johnny Lin (left) and a portion of a screenshot of the artifact by Claude Opus 5.5. (right)

Do I still need to learn programming? Yes!

- Use Claude to write the code.
- You still need to know programming and software engineering in order to know how to break up the task into the right pieces.
- CS achievement and writing skill may predict ability to vibe code: See Thorgeirsson et al. (2026).

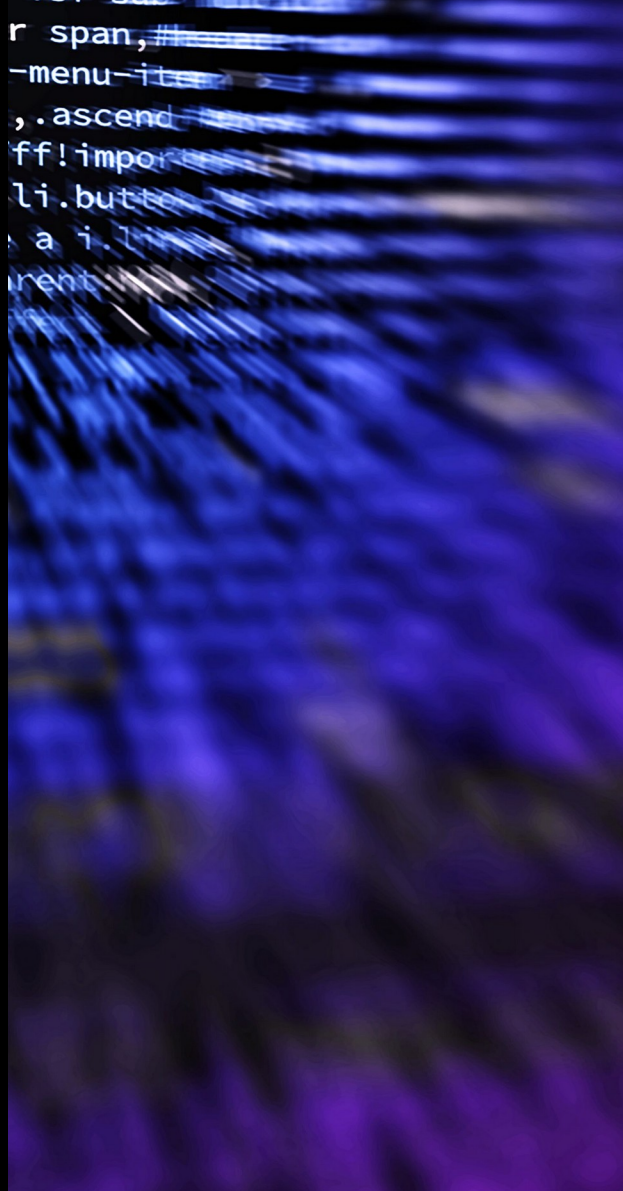


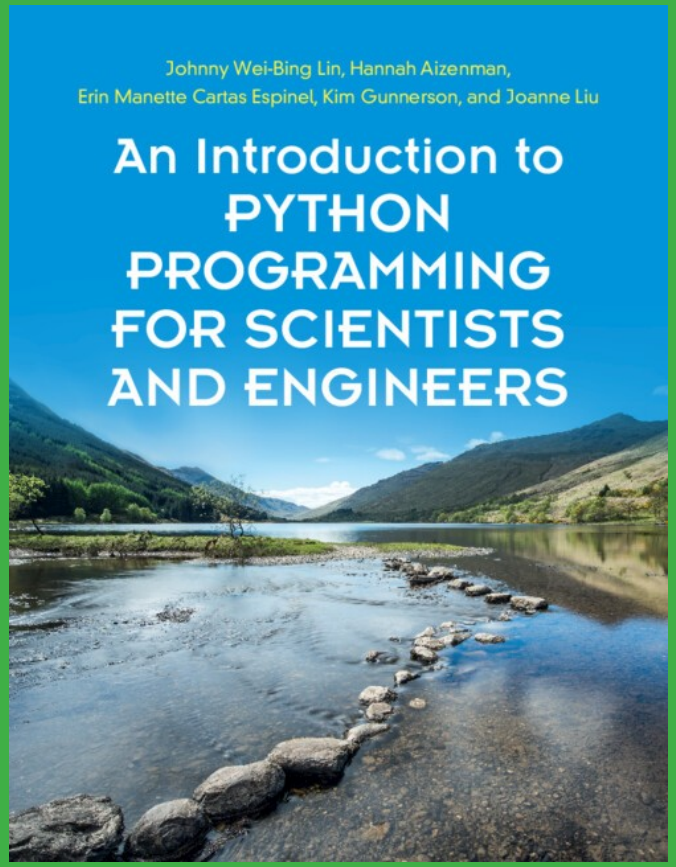
Photo: Pexels.com, Negative Space.



Photo: Pexels.com, Pixabay.

Conclusions

- Learning how to program means learning how to **break down a solution** into tasks a computer can do.
- **Be patient** with yourself as you learn to read code line-by-line.
- The right programming and software engineering **paradigms will help you** write better programs.
- Vibe coding using AI makes you **much more productive,** but you still need to know how to guide it.



Endorsed by
Kari Jordan
(The Carpentries)

Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs

Features

- Science workflow-driven, not syntax-driven.
- Examples and practice problems are from the sciences.
- **Very gentle pacing,** with many practice problems.
- Additional discipline-specific Jupyter notebooks of practice and homework problems in biology, chemistry, and physics.
- **Teaches novices programming.**



Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs

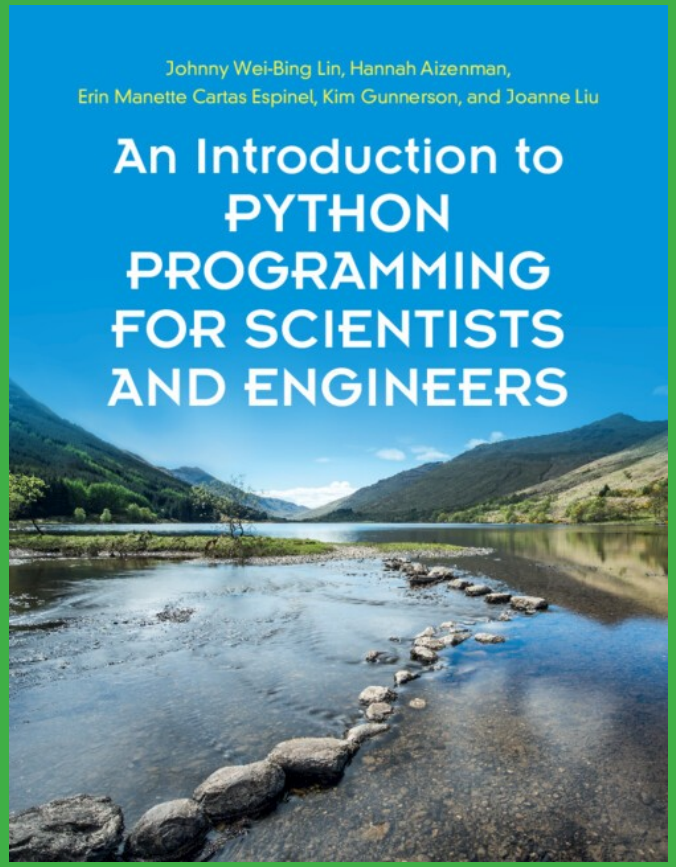
Contents

- Pt. 1: Getting Basic Tasks Done (e.g., Python as a calculator, basic plots, text files).
- Pt. 2: Doing More Complex Tasks (e.g., n-D data analysis, missing data).
- Pt. 3: Advanced Programming Concepts (e.g., inheritance, searching and sorting, other file formats, recursion).
- Pt. 4: Going From a Program Working to Working Well (e.g., documentation, profiling, unit testing).



Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs



Endorsed by
Kari Jordan
(The Carpentries)

Book: `tinyurl.com/pyscibook`
Rules: `tinyurl.com/roaopk`
Email: `johnny@johnny-lin.com`

@Lin_etal_IPySEs

References

- Guzdial, M. (2010): Ch. 7 of Oram, A. and G. Wilson, *Making Software: What Really Works, and Why We Believe It,* O'Reilly Media.
- Soloway, E., J. Bonar, et al., 1983. Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM* **26,** 11, 853-860.
- Thorgeirsson, S., T. B. Weidmann, and Z. Su, 2026: Computer Science Achievement and Writing Skills Predict Vibe Coding Proficiency *arXiv e-prints,* arXiv:2603.14133, https://doi.org/10.48550/arXiv.2603.14133.
- Watson, C. and F. W.B. Li, Failure rates in introductory programming revisited, *ItICSE'14,* June 21-25, 2014, 39-44.
- Wikipedia contributors. (2026, October 16). Vibe coding. In *Wikipedia, The Free Encyclopedia.* Retrieved 15:23, October 22, 2026, from https://en.wikipedia.org/w/index.php?title=Vibe_coding&oldid=1380596186.

TWITTER, TWEET, RETWEET and theTwitter Bird logo are trademarks of Twitter Inc. or its affiliates.