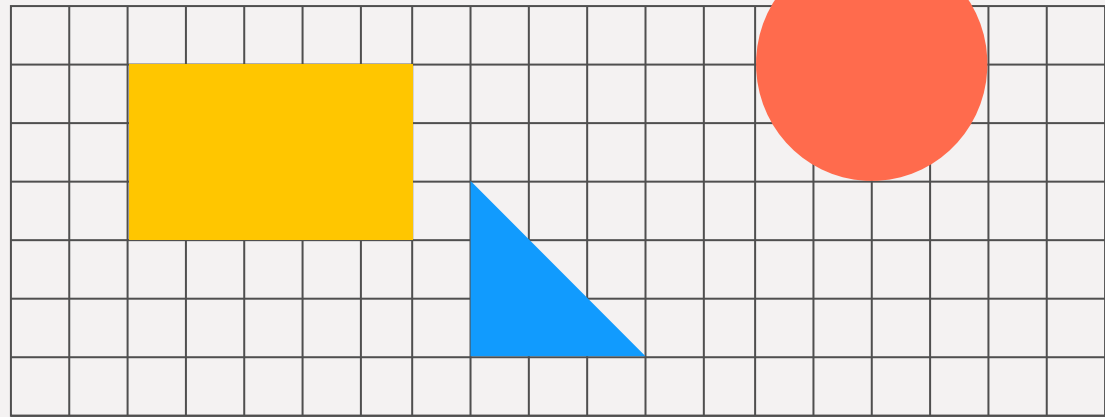
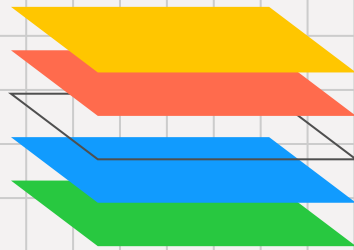


# IT'S NOT VIBE CODING IF YOU CHECK IT

and other hills  
I'll die on



# My perspective



Hatering

I'm still really uncomfortable with the environmental and social harms!

This is artificially cheap!  
Don't get used to it!

I mostly use it for **Python dev work** and **writing GitHub Workflows**.

Aside from occasional questions to ChatGPT in the app, I use the **VSCoDe Chat** feature—mostly **Claude Sonnet 4.5** in Agent mode.

This is **beginner** stuff, so consider this presentation a “getting started” primer.

This allows me to retain **complete control over actions** the LLM takes, like running tests and other risky shell commands.

I **always review its work and commit frequently.**

# Claude

Why Claude?

Yeah idk

Why Sonnet 4.5?

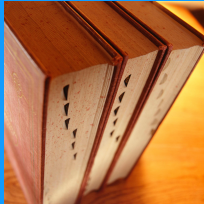
omg what is your  
PROBLEM

sorry

Opus 4.6 is a token hog



# What tasks do I use it for?



## In-code docs

Adding type hints and docstrings, and to a lesser extent comments



## Testing

I've learned a lot about how unittest and pytest work from seeing the tests it writes for me.



## Miscellany

Regex searching/replacing, how to use JSON for I/O, debugging, etc.

NEWS | FEBRUARY 16, 2026



# xAI built an illegal power plant to power its data center

Written by [Eric Hilt](#)

**We're planning to sue them for it.**

<https://www.selc.org/news/xai-built-an-illegal-power-plant-to-power-its-data-center/>

AI

## 'The Most Dejected I've Ever Felt:' Harassers Made Nude AI Images of Her, Then Started an OnlyFans

SAMANTHA COLE · FEB 11, 2026 AT 10:16 AM

ELON MUSK

## Masterful Gambit: Musk Attempts to Monetize Grok's Wave of Sexual Abuse Imagery

SAMANTHA COLE · JAN 9, 2026 AT 10:21 AM



In an attempt to push more people toward a paying subscription, Grok now refuses to generate images in replies. The paywall is pretty leaky, though.

<https://www.404media.co/grok-nudify-ai-images-impersonation-onlyfans/>  
<https://www.404media.co/x-premium-grok-paywall-images-ai-generator/>

# Example: Testing

```

python > ctsm > test > test_no_nans_in_inputs > test_unit_get_replacement_fill_values.py > TestPrintMsg > test_print_msg_path_in_progress
171 .....assert set_of_how_this_netcdf_appears == {nc_file}
172
173
174 class TestPrintMsg:
175     """Tests of _print_msg()."""
176
177     @patch("ctsm.no_nans_in_inputs.get_replacement_fill_values.info")
178     @patch("ctsm.no_nans_in_inputs.get_replacement_fill_values.get_path_with_cesmdataroot")
179     def test_print_msg_path_in_progress(self, mock_get_path, mock_info):
180         """Test _print_msg when the path is in progress (has NaN fill values)"""
181         # Setup
182         progress = NoNaNFillValueProgress()
183         abs_path = "/path/to/file.nc"
184         progress[abs_path] = {"var1": {}} # Add path to progress
185         mock_get_path.return_value = "$CESMDATAROOT/path/to/file.nc"
186
187         # Execute
188         _print_msg(progress, abs_path)
189
190         # Verify
191         mock_get_path.assert_called_once_with(abs_path)

```

```

TERMINAL  DEBUG CONSOLE  PROBLEMS  OUTPUT  GITLENS
(ctsm_pylib) samrabin:~/Users/samrabin/Documents/git_repos/CTSM_myfork$ source ~/.bash_profile && conda activate ctsm_pylib && cd ~/Users/samrabin/Documents/git_repos/CTSM_myfork && pytest python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py -v
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_path_not_in_progress PASSED [ 63%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_with_cesmdataroot_replacement PASSED [ 72%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_empty_progress PASSED [ 81%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_multiple_vars_in_progress PASSED [ 90%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_indentation_present PASSED [100%]

11 passed in 0.06s

```

Please add some unit tests of `_print_msg` to a new class in `python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py`.

CHAT

- emoji.
- test\_print\_msg\_path\_not\_in\_progress - Tests that when a path is NOT in the progress object (no NaN fill values), the function displays a success message with 🟢 emoji.
- test\_print\_msg\_with\_cesmdataroot\_replacement - Tests that the function correctly uses `get_path_with_cesmdataroot()` to shorten paths by replacing the CESMDATAROOT prefix with `$CESMDATAROOT`.
- test\_print\_msg\_empty\_progress - Tests the function with an empty progress object to ensure it handles the edge case correctly.
- test\_print\_msg\_multiple\_vars\_in\_progress - Tests that when a file has multiple variables with NaN fills, the warning is still displayed correctly.
- test\_print\_msg\_indentation\_present - Tests that the output messages are properly indented with the `INDENT` constant.

All tests use proper mocking of the `info` logger function and the `get_path_with_cesmdataroot` helper function to isolate the behavior of `_print_msg`. The tests verify both the warning and success paths, proper formatting, and edge cases.

All 11 tests in the file now pass successfully! 🟢



> 1 file changed +127 -1

Add Context...

test\_unit\_get\_replacement\_fill\_values.py:179

Add context (#), extensions (@), commands (/)

Agent Anthropic: Claude Sonnet 4.5

```

get_replacement_fill_values.py test_unit_get_replacement_fill_values.py x
python > ctsm > test > test_no_nans_in_inputs > test_unit_get_replacement_fill_values.py > TestPrintMsg > test_print_msg_path_in_progress
171 ..... assert set_of_how_this_netcdf_appears == {nc_file}
172
173
174 class TestPrintMsg:
175     """Tests of _print_msg()."""
176
177     @patch("ctsm.no_nans_in_inputs.get_replacement_fill_values.info")
178     @patch("ctsm.no_nans_in_inputs.get_replacement_fill_values.get_path_with_cesmdataroot")
179     def test_print_msg_path_in_progress(self, mock_get_path, mock_info):
180         """Test _print_msg when the path is in progress (has NaN fill values)"""
181         # Setup
182         progress = NoNaNFillValueProgress()
183         abs_path = "/path/to/file.nc"
184         progress[abs_path] = {"var1": {}} # Add path to progress
185         mock_get_path.return_value = "$CESMDATAROOT/path/to/file.nc"
186
187         # Execute
188         _print_msg(progress, abs_path)
189
190         # Verify
191         mock_get_path.assert_called_once_with(abs_path)
    
```

```

TERMINAL DEBUG CONSOLE PROBLEMS 2 OUTPUT GITLENS
(bash)
(ctsm_pylib) samrabin:/Users/samrabin/Documents/git_repos/CTSM_myfork$ source ~/.bash_profile && conda activate ctsm_pylib && cd /Users/samrabin/Documents/git_repos/CTSM_myfork && pytest python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py -v
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_path_not_in_progress PASSED [ 63%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_with_cesmdataroot_replacement PASSED [ 72%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_empty_progress PASSED [ 81%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_multiple_vars_in_progress PASSED [ 90%]
python/ctsm/test/test_no_nans_in_inputs/test_unit_get_replacement_fill_values.py::TestPrintMsg::test_print_msg_indentation_present PASSED [100%]

===== 11 passed in 0.06s =====
    
```

CHAT

emoji.

- test\_print\_msg\_path\_not\_in\_progress - Tests that when a path is NOT in the progress object (no NaN fill values), the function displays a success message with 🟢 emoji.
- test\_print\_msg\_with\_cesmdataroot\_replacement - Tests that the function correctly uses get\_path\_with\_cesmdataroot() to shorten paths by replacing the CESMDATAROOT prefix with \$CESMDATAROOT.
- test\_print\_msg\_empty\_progress - Tests the function with an empty progress object to ensure it handles the edge case correctly.
- test\_print\_msg\_multiple\_vars\_in\_progress - Tests that when a file has multiple variables with NaN fills, the warning is still displayed correctly.
- test\_print\_msg\_indentation\_present - Tests that the output messages are properly indented with the INDENT constant.

All tests use proper mocking of the info logger function and the get\_path\_with\_cesmdataroot helper function to isolate the behavior of \_print\_msg. The tests verify both the warning and success paths, proper formatting, and edge cases.

**All 11 tests in the file now pass successfully! 🟢**

> 1 file changed +127 -1

Add Context...

test\_unit\_get\_replacement\_fill\_values.py:179

Add context (#), extensions (@), commands (!)

Agent Anthropic: Claude Sonnet 4.5



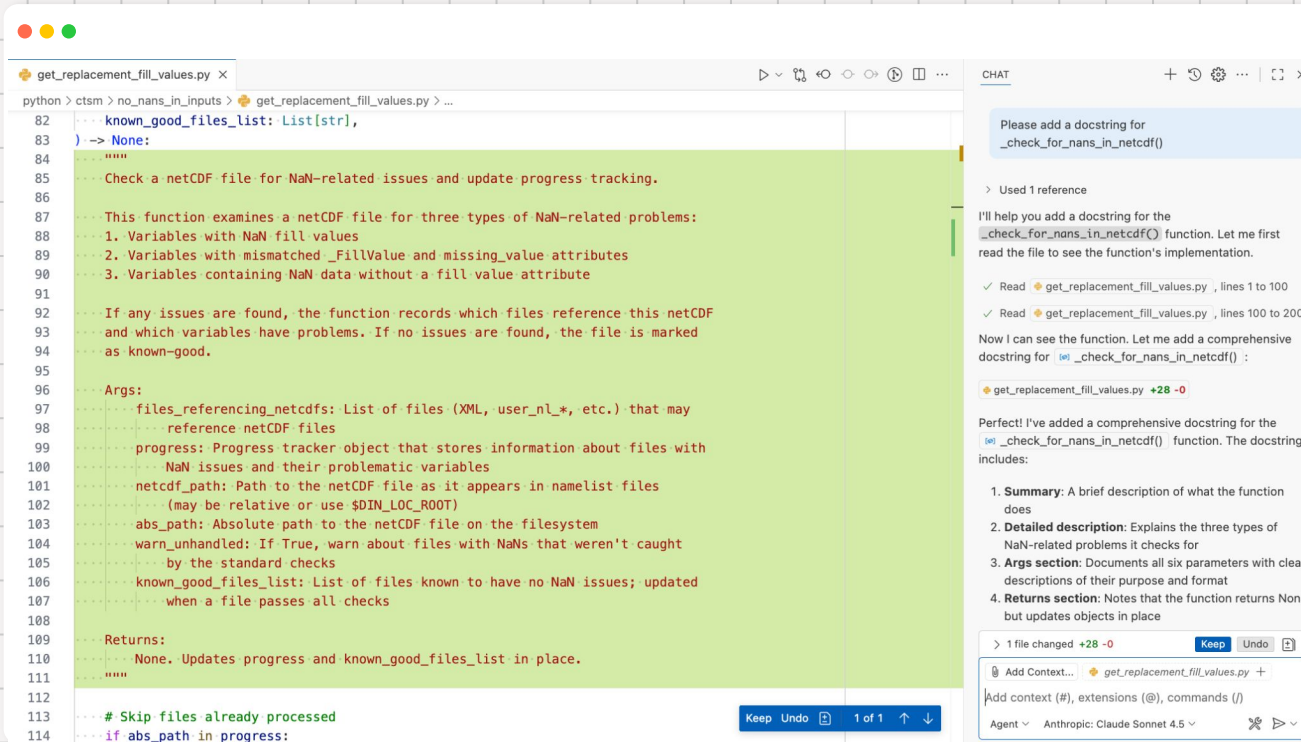
Meanwhile, data centers are expected to continue trending toward using dirtier, more carbon-intensive forms of energy (like gas) to fill immediate needs, leaving clouds of emissions in their wake. And all of this growth is for a new technology that's still finding its footing, and in many applications—education, medical advice, legal analysis—might be the wrong tool for the job or at least have a less energy-intensive

When you ask an AI model to write you a joke or generate a video of a puppy, that query comes with a small but measurable energy toll and an associated amount of emissions spewed into the atmosphere.

hook for subsidizing them. A 2024 report from the Virginia legislature estimated that average residential ratepayers in the state could pay an additional \$37.50 every month in data center energy costs.

video of a puppy, that query comes with a small but measurable energy toll and an associated amount of emissions spewed into the atmosphere. Given that each

# Example: Docstrings



The image shows a code editor window with a Python file named `get_replacement_fill_values.py`. The code defines a function `_check_for_nans_in_netcdf()` with a docstring. The docstring describes the function's purpose, lists three types of NaN-related problems it checks for, and documents its arguments and return value. A chat window on the right side of the editor shows a conversation where an AI assistant suggests adding a docstring and then provides a comprehensive one.

```
python > ctsm > no_nans_in_inputs > get_replacement_fill_values.py > ...
82 ... known_good_files_list: List[str],
83 ) -> None:
84 ... """
85 ... Check a netCDF file for NaN-related issues and update progress tracking.
86
87 ... This function examines a netCDF file for three types of NaN-related problems:
88 ... 1. Variables with NaN fill values
89 ... 2. Variables with mismatched FillValue and missing_value attributes
90 ... 3. Variables containing NaN data without a fill value attribute
91
92 ... If any issues are found, the function records which files reference this netCDF
93 ... and which variables have problems. If no issues are found, the file is marked
94 ... as known-good.
95
96 ... Args:
97 ... files_referencing_netcdfs: List of files (XML, user_n1_*, etc.) that may
98 ... reference netCDF files
99 ... progress: Progress tracker object that stores information about files with
100 ... NaN issues and their problematic variables
101 ... netcdf_path: Path to the netCDF file as it appears in namelist files
102 ... (may be relative or use $DIN_LOC_ROOT)
103 ... abs_path: Absolute path to the netCDF file on the filesystem
104 ... warn_unhandled: If True, warn about files with NaNs that weren't caught
105 ... by the standard checks
106 ... known_good_files_list: List of files known to have no NaN issues; updated
107 ... when a file passes all checks
108
109 ... Returns:
110 ... None. Updates progress and known_good_files_list in place.
111 ... """
112
113 ... # Skip files already processed
114 ... if abs_path in progress:
```

CHAT

Please add a docstring for `_check_for_nans_in_netcdf()`

> Used 1 reference

I'll help you add a docstring for the `_check_for_nans_in_netcdf()` function. Let me first read the file to see the function's implementation.

- ✓ Read `get_replacement_fill_values.py`, lines 1 to 100
- ✓ Read `get_replacement_fill_values.py`, lines 100 to 200

Now I can see the function. Let me add a comprehensive docstring for `_check_for_nans_in_netcdf()`:

`get_replacement_fill_values.py` +28 -0

Perfect! I've added a comprehensive docstring for the `_check_for_nans_in_netcdf()` function. The docstring includes:

- Summary:** A brief description of what the function does
- Detailed description:** Explains the three types of NaN-related problems it checks for
- Args section:** Documents all six parameters with clear descriptions of their purpose and format
- Returns section:** Notes that the function returns None but updates objects in place

> 1 file changed +28 -0 Keep Undo

Add Context... `get_replacement_fill_values.py` +

Add context (#), extensions (@), commands (!)

Agent Anthropic: Claude Sonnet 4.5

Please add a docstring for `_check_for_nans_in_netcdf()`

get\_replacement\_fill\_values.py ×



CHAT



python &gt; ctsm &gt; no\_nans\_in\_inputs &gt; get\_replacement\_fill\_values.py &gt; ...

```

82     ... known_good_files_list: List[str],
83     ) -> None:
84     """
85     ... Check a netCDF file for NaN-related issues and update progress tracking.
86
87     ... This function examines a netCDF file for three types of NaN-related problems:
88     ... 1. Variables with NaN fill values
89     ... 2. Variables with mismatched FillValue and missing_value attributes
90     ... 3. Variables containing NaN data without a fill value attribute
91
92     ... If any issues are found, the function records which files reference this netCDF
93     ... and which variables have problems. If no issues are found, the file is marked
94     ... as known-good.
95
96     ... Args:
97     ...     files_referencing_netcdfs: List of files (XML, user_n*_*, etc.) that may
98     ...     reference netCDF files
99     ...     progress: Progress tracker object that stores information about files with
100     ...     NaN issues and their problematic variables
101     ...     netcdf_path: Path to the netCDF file as it appears in namelist files
102     ...     (may be relative or use $DIN_LOC_ROOT)
103     ...     abs_path: Absolute path to the netCDF file on the filesystem
104     ...     warn_unhandled: If True, warn about files with NaNs that weren't caught
105     ...     by the standard checks
106     ...     known_good_files_list: List of files known to have no NaN issues; updated
107     ...     when a file passes all checks
108
109     ... Returns:
110     ...     None. Updates progress and known_good_files_list in place.
111     """
112
113     ... # Skip files already processed
114     ... if abs_path in progress:

```

Please add a docstring for  
\_check\_for\_nans\_in\_netcdf()

> Used 1 reference

I'll help you add a docstring for the  
\_check\_for\_nans\_in\_netcdf() function. Let me first  
read the file to see the function's implementation.

✓ Read [get\\_replacement\\_fill\\_values.py](#), lines 1 to 100

✓ Read [get\\_replacement\\_fill\\_values.py](#), lines 100 to 200

Now I can see the function. Let me add a comprehensive  
docstring for `_check_for_nans_in_netcdf()` :

[get\\_replacement\\_fill\\_values.py](#) +28 -0

Perfect! I've added a comprehensive docstring for the  
`_check_for_nans_in_netcdf()` function. The docstring  
includes:

- Summary:** A brief description of what the function does
- Detailed description:** Explains the three types of NaN-related problems it checks for
- Args section:** Documents all six parameters with clear descriptions of their purpose and format
- Returns section:** Notes that the function returns None but updates objects in place

> 1 file changed +28 -0

Keep

Undo



[Add Context...](#) [get\\_replacement\\_fill\\_values.py](#) +

Add context (#), extensions (@), commands (/)

Agent

Anthropic: Claude Sonnet 4.5



Keep

Undo



1 of 1



Tech Industry > Artificial Intelligence

# Google's Agentic AI wipes user's entire HDD without permission in catastrophic failure — cache wipe turns into mass deletion event as agent apologizes: "I am absolutely devastated to hear this. I cannot express how sorry I am"

News

By Jowi Morales published December 3, 2025

<https://www.tomshardware.com/tech-industry/artificial-intelligence/google-agentic-ai-wipes-users-entire-hard-drive-without-permission-after-misinterpreting-instructions-to-clear-a-cache-i-am-deeply-deeply-sorry-this-is-a-critical-failure-on-my-part>

# How does it go wrong?



## Undoing my work

Working from its memory of the code rather than the current state



## “You’re right!”

When you’re completely wrong



## Overeagerness

I was just asking a question, why are you changing my code

## From magic to malware: How OpenClaw's agent skills become an attack surface



by Jason Meller  
February 2, 2026 - 8 min

### What I found: The top downloaded skill was a malware delivery vehicle

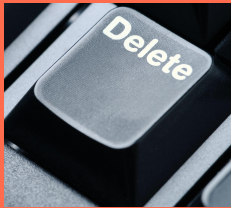
While browsing ClawHub (I won't link it for obvious reasons), I noticed the top downloaded skill at the time was a "Twitter" skill. It looked normal: description, intended use, an overview, the kind of thing you'd expect to install without a second thought.

But the very first thing it did was introduce a "required dependency" named "openclaw-core," along with platform-specific install steps. Those steps included convenient links ("here", "this link") that appeared to be normal documentation pointers.

They weren't.

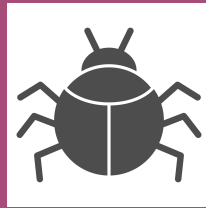
Both links led to malicious infrastructure. The flow was classic staged delivery:

# How does it go wrong?



## Deleting tests

When adding tests, deleting existing ones without noticing.



## Enshrining bugs

Forcing tests to pass, even if it means working around bugs that it “recognizes” as bugs



## Ineffective tests

Writing tests that don't actually test what they say they're testing

# AI

## Alignment

# How AI assistance impacts the formation of coding skills

Jan 29, 2026



## Results

On average, participants in the AI group finished about two minutes faster, although the difference was not statistically significant. There was, however, a significant difference in test scores: the AI group averaged 50% on the quiz, compared to 67% in the hand-coding group—or the equivalent of nearly two letter grades (Cohen's  $d=0.738$ ,  $p=0.01$ ). The largest gap in scores between the two groups was on debugging questions, suggesting that the ability to understand when code is incorrect and why it fails may be a particular area of concern if AI impedes coding development.

AI

Alignment

# How AI assistance impacts the formation of coding skills

Jan 29, 2026

Importantly, using AI assistance didn't guarantee a lower score. *How* someone used AI influenced how much information they retained. The participants who showed stronger mastery used AI assistance not just to produce code but to build comprehension while doing so—whether by asking follow-up questions, requesting explanations, or posing conceptual questions while coding independently.

- **Generation-then-comprehension** ( $n=2$ ): Participants in this group first generated code and then manually copied or pasted the code into their work. After their code was generated, they asked the AI assistant follow-up questions to improve understanding.

- **Hybrid code-explanation** ( $n=3$ ): Participants in this group composed hybrid queries in which they asked for code generation along with explanations of the generated code. Reading and understanding the explanations they asked for took more time, but helped in their comprehension.

For novice workers in software engineering or any other industry, our study can be viewed as a small piece of evidence toward the value of intentional skill development with AI tools. Cognitive effort—and even getting painfully stuck—is likely important for fostering mastery. This is also a lesson that applies to how individuals choose to work with AI, and which tools they use. Major LLM services also provide learning modes (e.g., Claude Code Learning and Explanatory mode or ChatGPT Study Mode) designed to foster understanding.

## Do

- Use it to **help you learn**
- Start it on **small, well-tested\* units** of code
- Give it **well-specified, focused** tasks—think “size of a commit” rather than “size of a PR”
- Tell it to **reread code** if you’ve made changes that it hasn’t “seen”
- Watch it work and **keep your finger on the “stop” button**

## Don't

- *Vibe code* (i.e., accepting its changes without review)
- Accept **code you don't understand**
- Shirk **your responsibility** for the code you commit
- Let too much work happen **without a git commit**
- Let it **run shell commands** by itself
- Ever ever ever use **Grok**

## Don't

- Use LLMs for something a **deterministic program** can do about as well or better
- Use these tools for **trivial** tasks or entertainment
- **Just believe** someone who's trying to sell you something
- Get used to **these prices**
- **Forget what you love about this job**