Cutting Through the Slop

Lessons Learned from a Year of Claude Code

PYTEXAS MEETUP · AUGUST 2026

Mason Egger

Howdy

- Mason Egger, Solutions Architect at Temporal
- PyTexas Foundation President
- PSF Fellow
- Author, Educator
- Previously DevRel at DigitalOcean, SRE at Vrbo, SDE at Forcepoint
- **AI Realist**

Using Claude Code for Over a Year

- Started at PyCon US 2025
- First use case: tutorial writing
- Asked Moshe to sit me down and teach me

Where I Use It

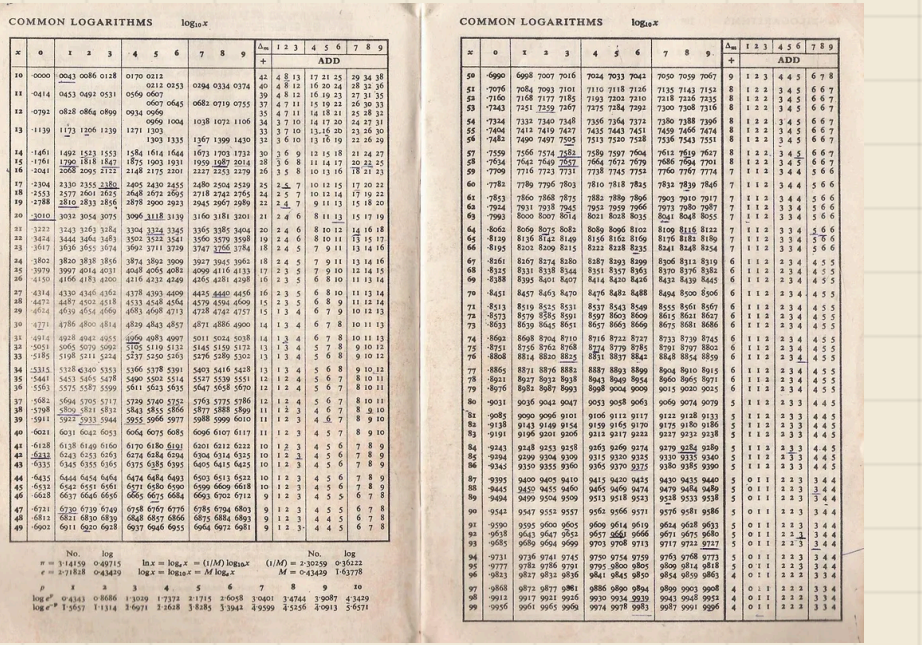
- Health planning
- Small business planning
- Executive assistant: my calendar, my tasks, my notes
- PyTexas: the websites, the automation, the schedules
- Technical writing
- Software development

History Repeats Itself

The World is Changing



A Relic of a Different Time



COMMON LOGARITHMS $\log_{10} x$

x	0	1	2	3	4	5	6	7	8	9	Δ_m	1	2	3	4	5	6	7	8	9
											+					ADD				
10	0000	0043	0086	0128	0170	0212					42	4	8	13	17	21	25	29	34	38
						0212	0253			0294	0334	0374	40	4	8	12	16	20	24	28	32	36
11	0414	0453	0492	0531	0569	0607					39	4	8	12	16	19	23	27	31	35
						0607	0645			0682	0719	0755	37	4	7	11	15	19	22	26	30	33
12	0792	0828	0864	0899	0934	0969					35	4	7	11	14	18	21	25	28	32
						0969	1004			1038	1072	1106	34	3	7	10	14	17	20	24	27	31
13	1139	1173	1206	1239	1271	1303					33	3	7	10	13	16	20	23	26	30
						1303	1335			1367	1399	1430	32	3	6	10	13	16	19	22	26	29
14	1461	1492	1523	1553	1584	1614	1644			1673	1703	1732	30	3	6	9	12	15	18	21	24	27
15	1761	1790	1818	1847	1875	1903	1931			1959	1987	2014	28	3	6	8	11	14	17	20	22	25
16	2041	2068	2095	2122	2148	2175	2201			2227	2253	2279	26	3	5	8	10	13	16	18	21	23
17	2304	2330	2355	2380	2405	2430	2455			2480	2504	2529	25	2	5	7	10	12	15	17	20	22
18	2553	2577	2601	2625	2648	2672	2695			2718	2742	2765	24	2	5	7	10	12	14	17	19	22
19	2788	2810	2833	2856	2878	2900	2923			2945	2967	2989	22	2	4	7	9	11	13	15	18	20
20	3010	3032	3054	3075	3096	3118	3139			3160	3181	3201	21	2	4	6	8	11	13	15	17	19
21	3222	3243	3263	3284	3304	3324	3345			3365	3385	3404	20	2	4	6	8	10	12	14	16	18
22	3424	3444	3464	3483	3502	3522	3541			3560	3579	3598	19	2	4	6	8	10	11	13	15	17
23	3617	3636	3655	3674	3692	3711	3729			3747	3766	3784	18	2	4	5	7	9	11	13	14	16
24	3802	3820	3838	3856	3874	3892	3909			3927	3945	3962	18	2	4	5	7	9	11	13	14	16
25	3979	3997	4014	4031	4048	4065	4082			4099	4116	4133	17	2	3	5	7	9	10	12	14	15
26	4150	4166	4183	4200	4216	4232	4249			4265	4281	4298	16	2	3	5	6	8	10	11	13	14
27	4314	4330	4346	4362	4378	4393	4409			4425	4440	4456	16	2	3	5	6	8	10	11	13	14
28	4472	4487	4502	4518	4533	4548	4564			4579	4594	4609	15	2	3	5	6	8	9	11	12	14
29	4624	4639	4654	4669	4683	4698	4713			4728	4742	4757	15	1	3	4	6	7	9	10	12	13
30	4771	4786	4800	4814	4829	4843	4857			4871	4886	4900	14	1	3	4	6	7	8	10	11	13
31	4914	4928	4942	4955	4969	4983	4997			5011	5024	5038	14	1	3	4	6	7	8	10	11	13
32	5051	5065	5079	5092	5105	5119	5132			5145	5159	5172	13	1	3	4	5	7	8	9	10	12
33	5185	5198	5211	5224	5237	5250	5263			5276	5289	5302	13	1	3	4	5	6	8	9	10	12
34	5315	5328	5340	5353	5366	5378	5391			5403	5416	5428	13	1	3	4	5	6	8	9	10	12
35	5441	5453	5465	5478	5490	5502	5514			5527	5539	5551	12	1	2	4	5	6	7	8	10	11
36	5563	5575	5587	5599	5611	5623	5635			5647	5658	5670	12	1	2	4	5	6	7	8	10	11
37	5682	5694	5705	5717	5729	5740	5752			5763	5775	5786	12	1	2	4	5	6	7	8	10	11
38	5798	5809	5821	5832	5843	5855	5866			5877	5888	5899	11	1	2	3	4	6	7	8	9	10
39	5911	5922	5933	5944	5955	5966	5977			5988	5999	6010	11	1	2	3	4	6	7	8	9	10
40	6021	6031	6042	6053	6064	6075	6085			6096	6107	6117	11	1	2	3	4	5	7	8	9	10
41	6128	6138	6149	6160	6170	6180	6191			6201	6212	6222	10	1	2	3	4	5	6	7	8	9
42	6232	6243	6253	6263	6274	6284	6294			6304	6314	6325	10	1	2	3	4	5	6	7	8	9
43	6335	6345	6355	6365	6375	6385	6395			6405	6415	6425	10	1	2	3	4	5	6	7	8	9
44	6435	6444	6454	6464	6474	6484	6493			6503	6513	6522	10	1	2	3	4	5	6	7	8	9
45	6532	6542	6551	6561	6571	6580	6590			6599	6609	6618	10	1	2	3	4	5	6	7	8	9
46	6628	6637	6646	6656	6665	6675	6684			6693	6702	6712	9	1	2	3	4	5	5	6	7	8
47	6721	6730	6739	6749	6758	6767	6776			6785	6794	6803	9	1	2	3	4	5	5	6	7	8
48	6812	6821	6830	6839	6848	6857	6866			6875	6884	6893	9	1	2	3	4	4	5	6	7	8
49	6902	6911	6920	6928	6937	6946	6955			6964	6972	6981	9	1	2	3	4	4	5	6	7	8

	No.	log
$\pi =$	3·14159	0·49715
$e =$	2·71828	0·43429

$\ln x = \log_e x = (1/M)\log_{10} x$ $\quad (1/M) = 2{\cdot}30259 \quad 0{\cdot}36222$

$\log x = \log_{10} x = M \log_e x$ $\quad M = 0{\cdot}43429 \quad 1{\cdot}63778$

p	1	2	3	4	5	6	7	8	9	10
$\log e^p$	0·4343	0·8686	1·3029	1·7372	2·1715	2·6058	3·0401	3·4744	3·9087	4·3429
$\log \pi^p$	1·5657	1·1314	2·6971	2·2628	3·8285	3·3942	4·9599	4·5256	4·0913	5·6571

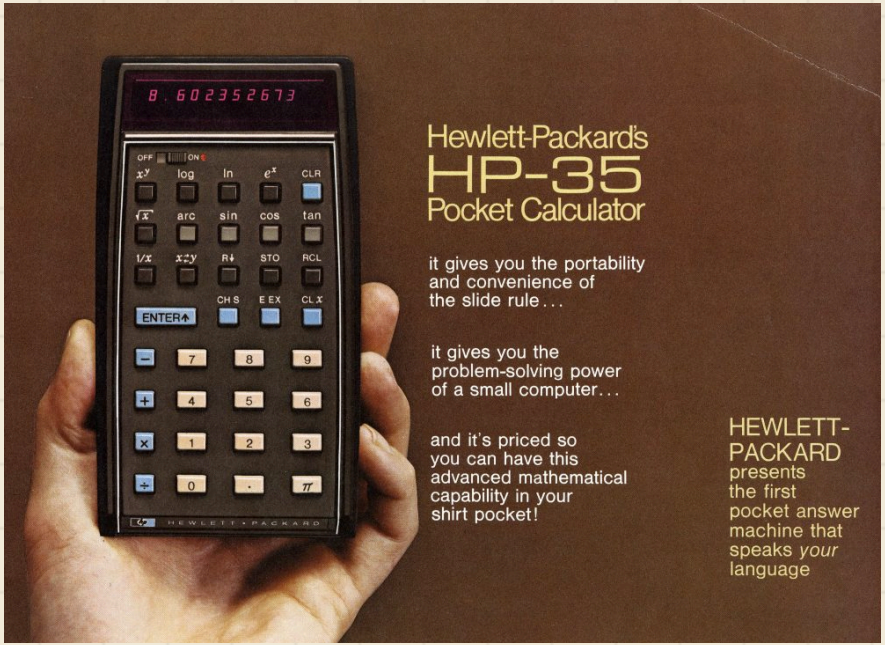
COMMON LOGARITHMS $\log_{10} x$

x	0	1	2	3	4	5	6	7	8	9	Δ_m	1	2	3	4	5	6	7	8	9
											+					ADD				
50	6990	6998	7007	7016	7024	7033	7042	7050	7059	7067	9	1	2	3	4	4	5	6	7	8
51	7076	7084	7093	7101	7110	7118	7126	7135	7143	7152	8	1	2	2	3	4	5	6	6	7
52	7160	7168	7177	7185	7193	7202	7210	7218	7226	7235	8	1	2	2	3	4	5	6	6	7
53	7243	7251	7259	7267	7275	7284	7292	7300	7308	7316	8	1	2	2	3	4	5	6	6	7
54	7324	7332	7340	7348	7356	7364	7372	7380	7388	7396	8	1	2	2	3	4	5	6	6	7
55	7404	7412	7419	7427	7435	7443	7451	7459	7466	7474	8	1	2	2	3	4	5	6	6	7
56	7482	7490	7497	7505	7513	7520	7528	7536	7543	7551	8	1	2	2	3	4	5	6	6	7
57	7559	7566	7574	7582	7589	7597	7604	7612	7619	7627	8	1	2	2	3	4	5	6	6	7
58	7634	7642	7649	7657	7664	7672	7679	7686	7694	7701	8	1	2	2	3	4	5	6	6	7
59	7709	7716	7723	7731	7738	7745	7752	7760	7767	7774	7	1	1	2	3	4	4	5	6	6
60	7782	7789	7796	7803	7810	7818	7825	7832	7839	7846	7	1	1	2	3	4	4	5	6	6
61	7853	7860	7868	7875	7882	7889	7896	7903	7910	7917	7	1	1	2	3	4	4	5	6	6
62	7924	7931	7938	7945	7952	7959	7966	7973	7980	7987	7	1	1	2	3	3	4	5	6	6
63	7993	8000	8007	8014	8021	8028	8035	8041	8048	8055	7	1	1	2	3	3	4	5	6	6
64	8062	8069	8075	8082	8089	8096	8102	8109	8116	8122	7	1	1	2	3	3	4	5	6	6
65	8129	8136	8142	8149	8156	8162	8169	8176	8182	8189	7	1	1	2	3	3	4	5	6	6
66	8195	8202	8209	8215	8222	8228	8235	8241	8248	8254	7	1	1	2	3	3	4	5	6	6
67	8261	8267	8274	8280	8287	8293	8299	8306	8312	8319	6	1	1	2	2	3	4	4	5	5
68	8325	8331	8338	8344	8351	8357	8363	8370	8376	8382	6	1	1	2	2	3	4	4	5	5
69	8388	8395	8401	8407	8414	8420	8426	8432	8439	8445	6	1	1	2	2	3	4	4	5	5
70	8451	8457	8463	8470	8476	8482	8488	8494	8500	8506	6	1	1	2	2	3	4	4	5	5
71	8513	8519	8525	8531	8537	8543	8549	8555	8561	8567	6	1	1	2	2	3	4	4	5	5
72	8573	8579	8585	8591	8597	8603	8609	8615	8621	8627	6	1	1	2	2	3	4	4	5	5
73	8633	8639	8645	8651	8657	8663	8669	8675	8681	8686	6	1	1	2	2	3	4	4	5	5
74	8692	8698	8704	8710	8716	8722	8727	8733	8739	8745	6	1	1	2	2	3	4	4	5	5
75	8751	8756	8762	8768	8774	8779	8785	8791	8797	8802	6	1	1	2	2	3	4	4	5	5
76	8808	8814	8820	8825	8831	8837	8842	8848	8854	8859	6	1	1	2	2	3	4	4	5	5
77	8865	8871	8876	8882	8887	8893	8899	8904	8910	8915	6	1	1	2	2	3	4	4	5	5
78	8921	8927	8932	8938	8943	8949	8954	8960	8965	8971	6	1	1	2	2	3	4	4	5	5
79	8976	8982	8987	8993	8998	9004	9009	9015	9020	9025	6	1	1	2	2	3	4	4	5	5
80	9031	9036	9042	9047	9053	9058	9063	9069	9074	9079	5	1	1	2	2	3	3	4	4	5
81	9085	9090	9096	9101	9106	9112	9117	9122	9128	9133	5	1	1	2	2	3	3	4	4	5
82	9138	9143	9149	9154	9159	9165	9170	9175	9180	9186	5	1	1	2	2	3	3	4	4	5
83	9191	9196	9201	9206	9212	9217	9222	9227	9232	9238	5	1	1	2	2	3	3	4	4	5
84	9243	9248	9253	9258	9263	9269	9274	9279	9284	9289	5	1	1	2	2	3	3	4	4	5
85	9294	9299	9304	9309	9315	9320	9325	9330	9335	9340	5	1	1	2	2	3	3	4	4	5
86	9345	9350	9355	9360	9365	9370	9375	9380	9385	9390	5	1	1	2	2	3	3	4	4	5
87	9395	9400	9405	9410	9415	9420	9425	9430	9435	9440	5	0	1	1	2	2	3	3	4	4
88	9445	9450	9455	9460	9465	9469	9474	9479	9484	9489	5	0	1	1	2	2	3	3	4	4
89	9494	9499	9504	9509	9513	9518	9523	9528	9533	9538	5	0	1	1	2	2	3	3	4	4
90	9542	9547	9552	9557	9562	9566	9571	9576	9581	9586	5	0	1	1	2	2	3	3	4	4
91	9590	9595	9600	9605	9609	9614	9619	9624	9628	9633	5	0	1	1	2	2	3	3	4	4
92	9638	9643	9647	9652	9657	9661	9666	9671	9675	9680	5	0	1	1	2	2	3	3	4	4
93	9685	9689	9694	9699	9703	9708	9713	9717	9722	9727	5	0	1	1	2	2	3	3	4	4
94	9731	9736	9741	9745	9750	9754	9759	9763	9768	9773	5	0	1	1	2	2	3	3	4	4
95	9777	9782	9786	9791	9795	9800	9805	9809	9814	9818	5	0	1	1	2	2	3	3	4	4
96	9823	9827	9832	9836	9841	9845	9850	9854	9859	9863	4	0	1	1	2	2	3	3	4	4
97	9868	9872	9877	9881	9886	9890	9894	9899	9903	9908	4	0	1	1	2	2	3	3	4	4
98	9912	9917	9921	9926	9930	9934	9939	9943	9948	9952	4	0	1	1	2	2	3	3	4	4
99	9956	9961	9965	9969	9974	9978	9983	9987	9991	9996	4	0	1	1	2	2	3	3	4	4

Somebody's Whole Job

- Roads, bridges, and dams were designed with this book
- People memorized the logarithms they used most
- The "high performer" could probably recall from memory (or calculate) them instantly
- Being fast became an advantage, because everyone else had to flip pages
- *Rote memorization became a competitive advantage*

Thirty-Five Keys



An Example Closer to Home



What Persists

- Every time, we adapted
- Expertise is the value, implementation never was
- Every abstraction moved the work up a level, not away
- Learning a new tool has overhead
  - A tool of this caliber requires a lot of learning
  - Software engineers are used to learning new tools
- The alternative is to be left behind

Reality Check

AI Isn’t Going Away

- Revolutionizing everything it touches
- Tasks that took weeks now take minutes
- Insights spread across hundreds of sources now compiled during your lunch break
- **Code is cheaper than it has ever been, and it will never be this expensive again**

We Have Heard This Before

- **Clifford Stoll, 1995.** Twenty years online, wrote the book on catching a hacker across ARPANET: "No online database will replace your daily newspaper"
- **Robert Metcalfe, 1995.** Invented Ethernet: "I predict the Internet will soon go spectacularly supernova and in 1996 catastrophically collapse"
- **Paul Krugman, 1998**, in an article titled *Why most economists' predictions are wrong*: "By 2005 or so, it will become clear that the Internet's impact on the economy has been no greater than the fax machine's"

Same Song, Different Decade

- **"It’s all slop"**
  - The slop is real. It’s also an input problem, not a tool defect
- **"It’s a bubble. It pops and this all goes away"**
  - There will be a bubble. The technology doesn’t disappear when it pops
- **"Fine for toys. It can’t do real engineering"**
  - People ship production systems with it every week

The Game Is Yours to Lose

- Nobody is asking whether you like it
- It’s as dumb now as it ever will be
- "Let’s just wait and see what happens" is over
- Every day you wait, someone else pulls ahead of you
- So what do you do?

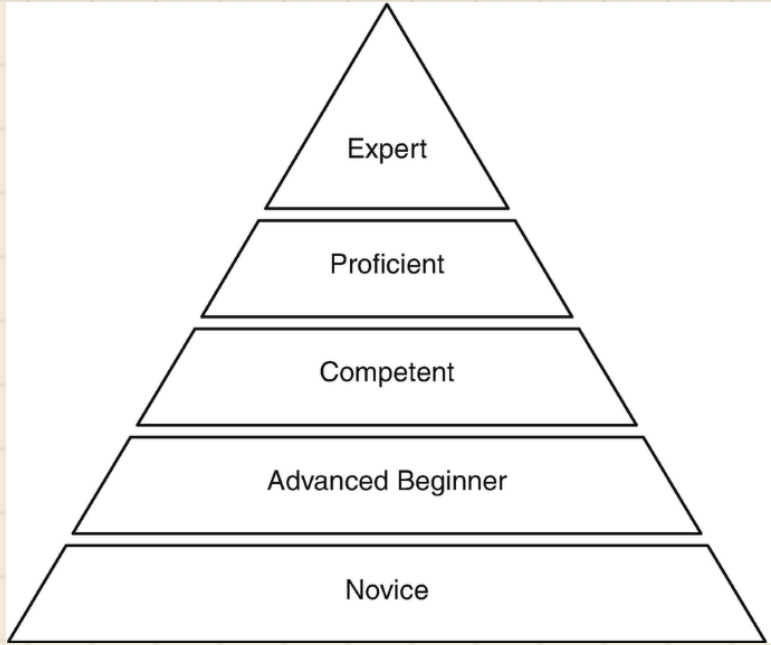
Tip 1: Use the Damn Thing

Watching from the Sidelines Isn't Learning

- Adults learn by doing
- "Oh I'm an auditory/visual/telepathic learner"
  - No you're not. You're all of them. This is junk science, was debunked years ago.
- Reading about it and watching others do it **feels** like progress
  - It's not, you're fooling yourself
- **Nothing you consume becomes a skill until you do something with it**

You Are a Novice Again

- Every new skill starts you at the bottom
- Most developers never get past advanced beginner, because they keep starting over
- Twenty years of experience doesn’t exempt you. It only makes the novice stage shorter
- Rejoice! Enjoy the thrill of being new again



"But There Was That Study"

- A lot of people like to quote a recent study about AI and productivity
  - A randomized controlled trial.
  - 16 experienced open-source developers
  - 246 real tasks, in mature repos they had worked in for an average of five years
  - Ran February to June 2025, on Claude 3.5 and 3.7 Sonnet
- Results *that were quoted*
  - Allowing AI tooling made them **19% slower**
  - The developers forecast AI would make them **24% faster**
  - After finishing, having just been slower, they estimated they had been **20% faster**
- As always, people don’t read

The Data Point Nobody Screenshotted

- METR couldn’t rule out learning effects past **fifty hours** with the tool
- The one developer who had that much time on it was **faster**, not slower
- A second one later reported over a hundred hours. He was slower. The bucket nets to zero
- Two people isn’t a finding. It’s the only look we get at the far side of the curve

Did You Hear About the Follow Up to that Study?

- In February 2026, METR redesigned the experiment
- Between **30% and 50%** of developers wouldn't hand over their tasks at all
- The ones they held back were the ones they didn't want to do without AI
- So the study kept measuring the work where AI helps least
- METR's own read now: developers are **likely more sped up** than that first estimate

Oh Look, We're Learning!

- You will be slower at first, and you won't be able to feel it
- The slop you got in week one was the curve, not a verdict on the tool
- Developers under six months with these tools are the likeliest to have a low opinion of them
- A lot of them tried it, it didn't match expectations, and they stopped

Signals to Ignore

- **Model of the month.** It's a Formula 1 race. The lead changes constantly and you aren't driving
- **Feature of the month.** Adopt a feature when you have the need, not when it ships
- **What everyone online is doing.** More than half of them are lying
- The only variable you actually control is you

Force Yourself to Use It

When Do I Reach for It?

- If a human has to do it, a human does it
- Everything else comes down to two questions
- Do I want to do it?
- Do I have the time to do it?
- Both true and I do it myself. Anything else gets an AI pass

```
do_it_myself = human_must_do or (want and time)
```

"But What Should I Build?"

- F**king **EVERYTHING**!
- Need a library? Build it
- Custom CLI Tool? Ship it
- App for monitoring the moon phases on Jupiter? Why not?
- **Just build something!**

Tip 2: Codify Yourself

Developers Don’t Trust AI Output

- The 2025 Stack Overflow survey, about thirty-three thousand developers answered this
- **84%** are using these tools or planning to
- **33%** trust the accuracy of what comes out
- **46%** actively distrust it
- Favorable opinion fell from over 70% to **60%** in a year

Why Don’t They?

- **It doesn’t look like the code you write!**
- An agent with no context writes the average of every codebase it has ever seen
- The average is mediocre
- It has no idea what your team decided in 2023 and never wrote down
- It can’t read the room, because you never gave it one

So Turn Yourself Into a Markdown File

- Everything that makes the code yours is in your head
- None of it is in the training data. It never was
- AI *cannot* read your mind (yet), so stop making it guess
- Write it down. Once. It's markdown in a folder

Like a Horcrux, But You Didn't Have to 💀 Anyone

- A Skill is what correct looks like to you, plus the tools to get there
- It loads only when the work calls for it, so you stop paying for context you don't need
- It's templates, examples, and scripts it can actually run
- Write it once and any compatible agent can use it
- Every project you finish makes the next one start further along

What Goes in the Skill

- **Your standards and anti-patterns.** What good looks like to you, and the specific things you refuse to ship
- **Your toolchain and definition of done.** How it builds, how it tests, how it lints, when you call it finished
- **The reference material it can’t see.** The samples repo, last year’s version, the SDK docs
- **Your decisions and the reasons behind them.** The rule plus the why, so it can generalize
- Write down how you work, and it writes like you

You Are Not Locked In

- Skills are an open standard. Anthropic wrote the format, then gave it away
- Same folder, any compatible agent. You aren't betting on a vendor
- `AGENTS.md` is the same practice with nobody's logo on it either
- The filename has already changed twice. It will change again
- Writing the rules down is the part that survives the change

Tip 3: Bad Output Is a Post-Mortem

You Can’t Just Say "Build Me Facebook"

- You can’t open an empty repo, type "build me Facebook", walk away, and expect production code
- Walk into a room of engineers and try the same thing
- It was never going to work either time
- The agent can do damn near anything you ask it to
- It failed because **you** didn’t give it the context or the guardrails

You’re Not Alone

- Their number one frustration, in their own words: **"AI solutions that are almost right, but not quite"**
- **66%** picked it off the list. Nothing else came close
- **45%** also said debugging AI-generated code is more time-consuming than writing it themselves
- If that’s your week, every week, walking away is the rational call
- It’s also the bill for skipping this lesson

Two Reactions Are Available

- It hands you something wrong. What happens next?
- **A verdict.** "This thing writes garbage." Close the terminal. Tell people at lunch
- **An incident.** Something in my system produced a bad output. Find out what
- One of those ends the story. The other one compounds



We Already Know How to Do This

> *"A blamelessly written postmortem assumes that everyone involved in an incident had good intentions and did the right thing with the information they had."*

– Google SRE Book, Postmortem Culture



Don’t Fix It. Ask It.

- Your first instinct is to fix it yourself. You’re an engineer, that’s the job
- Stop. That’s where the lesson gets thrown away
- `Did you follow the specification we agreed on for this?`
- The answer names which one you got wrong: context, or guardrails
- `What did I not provide, and what can I give you next time?`

A Real Example I Encountered

- I see a plan step add code to `__init__.py`
- Verdict reading: "It writes bad Python"
- Post-mortem reading: I never specified it shouldn’t
- The output was correct given what it knew. What it knew was wrong
- Update my Skill. Otherwise it happens in the next forty steps too

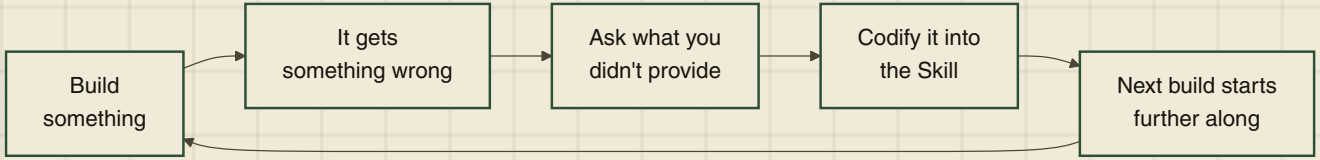
You Cannot Post-Mortem Code You Didn’t Read

- Review everything. **If I don’t understand it, it doesn’t get committed**
- Unread output can’t teach you anything, because you never see the failure
- Let it run unwatched and you understand less of your own project every week
- That’s cognitive debt, and the name fits

It Is a Mirror

- AI is an **amplifier**
- It magnifies an organization's existing strengths and its existing dysfunction
- Strong practices get stronger. Weak practices ship bad work faster
- Which means your bad output is a true statement about your system

The Wheel Keeps Turning



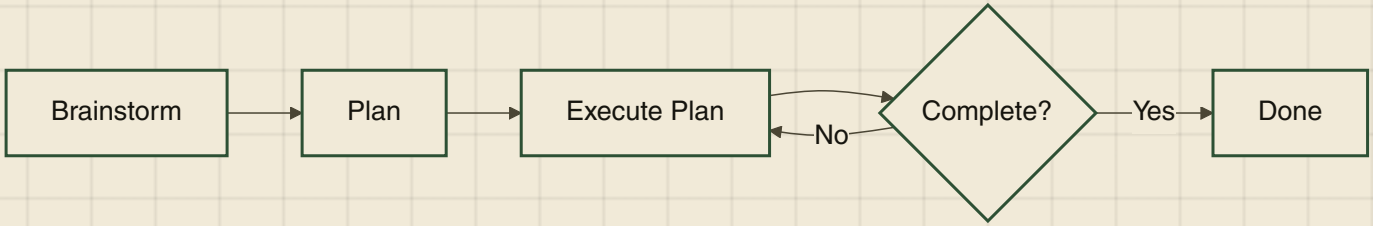
- Every project feeds back into the Skills
- The more I build, the better the Skills get
- The better the Skills get, the faster I build
- The faster I build, the more I build

What I Actually Do

The Loop I Actually Run

- **Brainstorm.** Iterative Q&A until the spec says what I actually meant
  - Produces `spec.md`
- **Plan.** The spec becomes right-sized steps, each one structured around tests
  - Ingest `spec.md`, create `plan.md` and `todo.md`
- **Execute.** One step at a time, following the plan exactly
  - Read `plan.md` and `todo.md` and implement
- **Review and record.** Recap the session, keep the lessons for next time
- The spec is the artifact. The code is the output

Oooh Pretty Diagram



Ask Claude How Claude Would Have Claude Do It

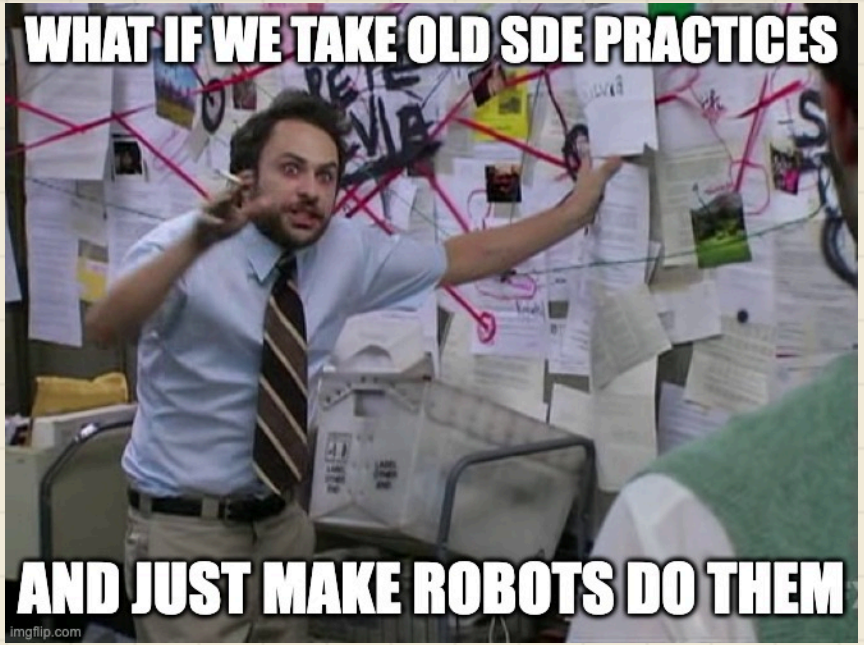
- Moshe, day one: "Claude is a better prompt engineer than you. Ask Claude how Claude would have Claude do it"
- The plan stage does exactly that. It writes the prompts it will later run
- `spec.md` is **what** to build
- `plan.md` is **how** to build it
- `todo.md` is what actually got built

Deterministic Tools Drive the Non-Deterministic M

- Tests, linters, formatters, type checkers
- The agent can run all of those. It can’t run your taste
- So make it practice Test Driven Development
- My success rate doubled when I switched, and the code came out tested

Waterfall FTW

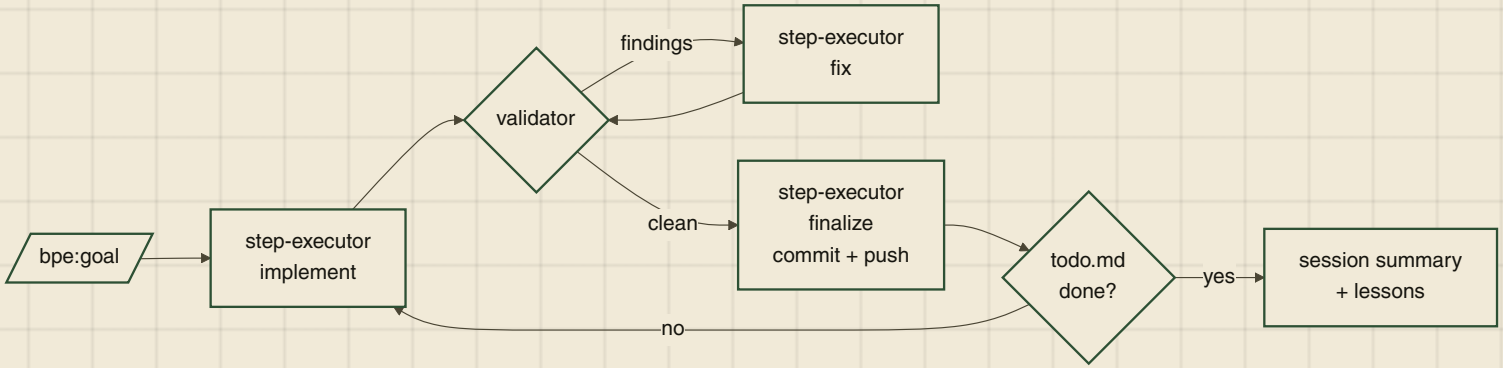
- Waterfall got its reputation because iterating on a spec was slow and expensive
- Updating a spec now takes minutes. Regenerating a plan takes minutes
- The objection didn’t survive the tooling
- Every other engineering discipline never stopped working this way



Look Ma! No Hands!

- `/bpe:goal` hands the plan to an autonomous run
- One step per dispatch, and every dispatch is a fresh context
- No compaction, no `/clear`, no watching the context window fill up
- The commit is the only durable unit. An interrupted step is reset and redone
- It refuses to run on `main`. Ask me how I learned that one

The Goal Loop





Three Agents, Three Jobs

- **step-executor** writes the code. One step, fresh context, nothing committed until it passes
- **validator** reads the diff. Read-only. It can't edit, commit, or run the tests
- **cheap-research** looks things up, fast, because most lookups aren't hard
- The agent that wrote the code isn't the agent that approves it
- Each role gets the model the role needs, not whatever you happened to be running

And It Isn’t Just Code



It Runs My Week

- 8:30 every weekday. Fifteen minutes, and it runs the same way every time
- It reschedules yesterday's misses, or makes me kill them
- Punted the same task three times? It stops and makes me pick: keep, icebox, delete
- It sets priority, duration, and a label. Reclaim does the actual scheduling
- Drop a big rock and I have to type why. It lands in an append-only log

It Talks to What I Already Use

- Reads Todoist for today, the next seven days, and everything tagged strategic
- Reads Reclaim for the calendar, so it knows a meeting wall when it sees one
- Reads this week’s plan file for the big rocks it has to defend
- Writes back priorities, durations and labels, and opens today’s Obsidian note
- A cron job at 7:47 triages the inbox before I’m awake

Nothing Ships Until I Approve It

- Every plan renders as an HTML page I approve in a browser
- The skill emits a validated JSON contract. A script renders the page
- The markup lives in a template, not in the prompt, so it can’t drift
- Missing data is a loud error, not a silent skip
- Same renderer that shows me my specs

One for Python. One for Me.

- The Python skill: type hints everywhere, no single-letter names, config blocks pasted verbatim
- Every rule carries its reason, so it holds up in the case I didn’t think of
- The writing skill came out of a hundred questions I answered about myself
- It knows my banned words, my tells, and the jokes I won’t make
- And it’s forbidden from writing for me. It edits. My name is on the post

What That Looks Like: Python

````
---
name: python
description: Python development standards and toolchain preferences. Use when
  writing ANY Python code, configuring tooling, or reviewing Python.
---

## Core Requirements

1. **Type hints everywhere** - all functions, all parameters, all return
   types. No `Any`.

6. **Descriptive variable names** - single-letter names (`i`, `j`, `k`, `m`,
   `x`, etc.) are NEVER allowed. Use names that convey meaning:
   `line_index`, `inner_index`, `label_match`.

9. **Canonical config blocks only** - the config blocks in
   `references/toolchain.md` are pasted verbatim into projects. Composing
   per-project ruff or mypy config is prohibited; per-repo improvised
   configs drift.
````

What That Looks Like: Teaching

```
## Core Opinions

- **Place the learner on the Dreyfus Model first.** Novices want step-by-step
  recipes; experts work on intuition and are hard to teach. Most developers
  never move past Advanced Beginner, so design for Novice by default.
- **Adults need the "why" up front.** Orient toward tasks and problems,
  respect existing experience, front-load the value.
- **A competency is something the learner does, not knows.** Single action
  verb, observable and measurable. "Understand X" is not a competency.
- **Backwards design.** Write the performance assessment before the
  learning activities.
- **Course design is expensive.** Budget ~233 hours of prep per 1 hour of
  instruction. Scope effort before committing to a course.
```

So What? What Have You Built?

PyTexas Runs on It

- The meetup site, the conference site, the CFP, the runbooks
- `pytexas/infrastructure`: the droplet, the compose projects, the sops secrets, the ansible deploys
- The monthly meetup turnover, which used to be an evening of copy-paste
- My name is on every commit. I reviewed every one of them
- If it breaks, it is my problem, not the model's

The Meetup Automation

- One recurring Todoist task is the checklist of record. The skill mirrors it
- The speaker comes out of the CFP sheet in Drive, the booked date out of Gmail
- It opens the website PR, builds the Canva card, creates the run of show and the attendance form
- Every artifact is template-driven. It fills slots, it doesn’t improvise prose
- Then it posts to Discord: the assets to marketing, the summary to the organizers

A Temporal SDK. In Perl.

- `masonegger/sdk-perl`. Client, worker, activities, workflows, signals, queries, updates, timers, child workflows, Nexus
- It drives the Rust core over a C ABI, with a deterministic scheduler for workflow code
- 859 tests across 197 files, verified live against a dev server
- `masonegger/samples-perl`: 65 runnable samples, each one a port of the canonical cross-language example
- Nobody asked for this. Nobody wanted it
- I was seeing how far I could push Fable, and it went all the way

Code Is Cheap, Know Your Worth

The $500 Button

- A machine goes down. The plant calls in an expert
- She looks at it for a minute and presses one button. The machine starts running
- "That will be five hundred dollars"
- "Five hundred? You were here five minutes and you pressed one button!"
- "You don’t pay me to push the button, you pay me to know which button to push"

You Were Never Paid to Type

- We convinced ourselves, and worse, our employers, that the button-pushing was the hard part
- The job was never to write code. It was to solve problems
- Code was the way we solved them. That changed. The problems didn’t
- People are scared right now because they valued the implementation over the design

Congratulations, You Are an Architect

- The work moves up: design, integration points, product sense, verification
- Verification is the new center of gravity. You have to define what correct means
- The theory stops being academic. Algorithms, failure modes, consensus, all of it earns its keep
- You only get the speed-up if you already know what you’re doing, because that’s how you spot the nonsense

It’s Still My Name on It

- I review everything. If I don’t understand it, it doesn’t get committed
- My name is on every commit, and I mean that both ways
- If it breaks, it’s my problem. Not the model’s
- The bar didn’t move because the typing got faster
- **"I engineered it. Claude did the implementation."**

What I Think Is Coming

- Software is **not** dead. It’s finally becoming a real engineering discipline
- Big tech keeps downsizing as it gets more efficient. That’s already happening
- The constriction goes too far, then they back off and re-hire
- The correction period is going to be **rough**, and that doesn’t help you if you’re on the layoff list
- We have to reconcile the false promise: "learn to code and you’ll have a job"

Finish Things

> *"The world belongs to those who finish things."*

- Michael W. Lucas
- Most people who read the advice never run it
- Code is cheaper than it has ever been, and it will never be this expensive again
- Go build the thing you have been putting off

Where to Find All of This

- **The long version**: `Brain Dump Ramblings on AI`, five parts, at `mason.dev`
- **The loop**: `github.com/MasonEgger/bpe-claude-code-plugin`
- **Me**: `@masonegger`

Thank You

Questions?