Finally. Agent Loops Clearly Explained.

14 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=EUZYHZB0VBI
https://www.youtube.com/watch?v=EuzYhzB0vbI

SUMMARY

The video discusses the concept of "agent loops" in AI, emphasizing their importance over traditional prompting methods for coding agents. It explains how designing loops—comprising triggers, actions, and stop conditions—can enhance productivity by allowing AI to self-iterate towards a goal, while also stressing the need for clear definitions of success and verification methods.

- *Agent loops consist of a trigger, action, and stop condition, allowing for more efficient AI interactions.*
- *Loop engineering replaces the need for constant human prompting by allowing agents to infer and execute tasks autonomously.*
- *The two key pillars of effective loops are defining a clear goal and establishing verification methods to determine when the goal is achieved.*
- *Not all tasks require complex loops; simpler workflows can often suffice, especially for verification purposes.*
- *Effective loops can significantly improve output quality through iterative feedback and adjustment.*
- *The verification process is crucial; it ensures agents assess their outputs and make necessary adjustments.*
- *The video highlights that the application of agent loops varies by individual needs and contexts, and not every approach will suit every user.*
- *The creator shares personal experiences with loops, emphasizing that while they can be powerful, they should be tailored to specific tasks and goals.*

Right here, I've got four different agents that are looping, calling other sub agents, and writing all these prompts for me, and designing systems for me. But, is this actually productive, or is that just a cool demo? Here's your monthly reminder that you shouldn't be prompting coding agents anymore. You should be designing loops that prompt your agents. Boris Cherny and Peter Steinberg publicly said they no longer prompt their coding agents. They write loops. A loop is three things: a trigger, an action, and a stop condition. If you're still writing loops that prompt coding agents, you're falling behind. You need to build a meta agent that infers what loops you would have wanted based on your vibe, and then write those loops. We're seeing a ton of talk about agent loops, loop engineering, whatever you want to call it. So, I wanted to make a video to clear up what that actually means.

Because I think that everyone kind of has their own spin and a different definition of what this is, and it applies to everyone very, very differently. I think that this definition sums it up pretty well. Loop engineering is replacing yourself as the person who prompts the agent. You design the system that does that instead. A loop here can be thought of as a recursive goal, where you define a purpose, and the AI iterates until complete. And there's really two most important pillars of that in my mind, which are the goal. What is the actual objective? Something typically that's objective, not subjective. And then verification. How does the agent know what that stop condition is? How does it check and iterate? So, anyways, if you take all that advice, and then you start

doing stuff like this, and designing swarms and fleets of agents that constantly run 24/7, then you need to think about what are you actually doing here? And is this actually moving the needle? So, first of all, I thought to myself, how do I actually use agent loops? Because when you read some of those tweets that I just showed earlier in this video, you kind of think to yourself, okay, if I'm not having five agents that are continuously around the clock orchestrating five of their own agents, then I'm falling behind, or I'm not using my cloud subscription in the best way. And I think that that's very false. Because if you don't understand what you're doing, then you're probably just going to scale problems, and you're going to have a ton of bugs and a ton of things that you're going to have to fix later. And also, not all of us are in a scenario where having agents work 24/7 around the clock actually benefits us.

For example, I don't. I have agents that do things on a certain cadence and I have agents that do things based on certain event actions, but just having them do 24/7 work for me isn't helpful. I think if I was working with a team on a codebase and we were building a product and we were constantly iterating and pulling in different things, then it would maybe make more sense, but for me, that doesn't apply. So, I just wanted to come in here, explain this as simple as I can, and hopefully shed some light on where you guys can start applying loops into your workflows and why and how. So, I actually built an agent loop for this HTML that we're going to look at today, and it basically went through a ton of different sources. It checked 45 sources, whether that was articles, YouTube video transcripts, X posts. It looked through a ton of stuff, and then it kept looping on this until it had a good idea of what to build. And then once it built this HTML, this wasn't V1.

This was probably V7. It had to keep checking, screenshotting, reviewing, iterating, and then it finally said, "Okay, we're done. This is what we got." So, let me walk through this with you guys. An agent loop is just an AI that reasons on what to do, acts on what to do, starts implementing, and then it observes the result. And it will do that over and over and over until some sort of goal is met, until it knows we've hit the stop criteria, this is good, I'm going to stop now. And a really simple visual that I like to think about is AI is never perfect, right? It's never going to one-shot something and you just accept that final output. And so, if we have attempts on the x-axis and we have quality on the y-axis, let's think about this. On attempt one, if you are just giving your agent some sort of simple task, maybe you get to like, let's just say attempt one, you get to 50%, and then you look at that and say, "Okay, here are some changes to make." And then by attempt two, maybe you bump up another five or 10%. And every time that you give more feedback and iterate, you just kind of keep moving up on quality until you hit somewhere where you're okay with that, 90, 95%. And so, the whole idea is why don't we outsource this part, this feedback and iteration loop, to an agent rather than having the human do that? Cuz this is going to happen either way. So, if we have an agent do that instead of a human, then what might happen is on attempt one, we will go straight up to here. And then we can give a little bit more feedback. And then by attempt, you know, three or four, we're already so much higher than where we would have been without sort of that agent verification loop right there. And that's why a lot of people are explaining this in a different way, where some people have the think act see, you know, we basically like reason act observe reason act observe. Some people have the model just going back and forth with tools back and forth back and forth. Some people have just, you know, a goal that runs completely unattended. And some people are using these like fleets of agents with managers prompting other agents prompting other agents. And it's just like, you know, those Russian nesting dolls. So that's why I wanted to put this into kind of the main pillars, which I think are reason act observe.

Think of this like a smart intern that you don't micromanage. You hand them a goal, they figure out what to do next, they check their own work, and they go again, and then they only come back to you and say, "Hey, I'm done." After they probably checked it a few times and made some changes. So you would say, "Okay, Claude code, here's what I want you to do." We as humans are really, really good at defining what we want. We're really good at defining an end goal. And then on top of that, we have to say, "Okay, how do you know when that is done?" So when you're making a cake, you stick the fork in it, and when it comes out and it doesn't have batter all over it, that means it's done. How do you tell your agent something as objective as possible, what is the stop criteria, what is the definition of done? And so what it will do is it will reason, it will plan out, and then it will start to implement. After it implemented, it will observe. So maybe that's visual verification, maybe that's running an actual code test. Whatever it means to verify, it has to verify. And then after it looks at the results, it will say, "Okay, did I meet this done criteria? If no, I'm going to act again, then observe again, and then reason. Otherwise, I'm going to stop, and I'm going to say, "Okay, Mr. or Mrs. Human, I am done."

And what's really interesting is that the majority of tasks don't need loops. What I've started doing is for the majority of my tasks, I will build some sort of loop, but it's just because of the verification, right? This piece is so important, the verification loop. But a lot of times, you don't need some sort of massive agent architecture in order to run this sort of like dynamic looping workflow. You You just get it done with one simple terminal session and a good prompt. You can have the speed just a solo loop, which is what I'm typically doing the most. One agent that's reasoning, that's acting, observing, and repeating. And I'll show you guys some examples of what these loops might look like in just a sec. You can have a maker checker, where you have one agent that does the thing and then one agent that grades the thing and gives feedback. Or you can have this sort of manager with a bunch of helpers. And then as as long as you've got one main agent that's orchestrating the whole thing, then you can build these loops in so many different ways. So, let me just show you guys a few examples that I pulled. These first two that I'm going to show you were actually from this loop library that Matthew Berman published. He created this loop library, which is a list of agent loops that you can use, and people can go submit their own. So, kind of cool to just go in here and play around with and see what's available.

And I grabbed two for these first two demos. So, this was the first one right here. It was a {slash} goal prompt in Cloud Code to make me a thumbnail. So, I told it basically what to use to make them. It says, "Make 10 thumbnail concepts and score each one against Mr. Beast YouTube thumbnails using a rubric. Clarity at small size, curiosity, emotional pull, visual contrast." Stuff like that. And after it makes those 10, it selects the top three, it identifies the weakest part of each concept, it improves them, rescores them, and then it continues iterating on on the strongest concept until it's satisfied.

So, that's one of the issues with this prompt here is that the definition of done was "until you're satisfied." And sometimes you have to have these subjective sort of grading criteria, but you want to get it objective as objective as possible. The best agent loops are where you literally say, "Keep iterating until X metric equals Y result." You can see right here what it did is it created 10. We've got number one, we've got number two, number three, number four, number five. It ended up choosing that number one was one of the top contenders, number two, and so was number eight. So, then it iterated on these. You can see here's number one original, here's number one V2, here's number two original, here's number two V2, and here's number eight original, and here's number eight V2.

And what it did is after those version twos of all of it, it said, "Okay, number eight's the best. So, here is number eight V3." And so, this is the final thumbnail that we got after we ran this goal, which took Claude Code 27 minutes right there. So that's one quick example of the loop. You can see it was it was scoring each of these, and that's how it decided on the winner. But the one thing here is that these scores were subjective. So if we wanted to improve this flow, we would try to figure out how do we make this scoring more objective? And maybe what we would want to do is create a separate sub agent that was a dedicated scorer, and we would prompt that scoring agent and run that through a bunch of evaluations so that we could feel more confident about its scoring ability. Anyways, let's take a look at the next one. So the next one was another slash goal, as you can see right here. This one took 37 minutes.

And the prompt for this one was right here, straight from Matthew Berman's Loop Library. I'm not going to read this whole thing. You guys can pause it right there if you want to see. But it was basically supposed to make a plane using 3.js. So I'll open that up right here. We can see this is the spinning plane that it made. We can sort of zoom in. We can move it around. And that is what we got. Now from a looping perspective, what it had to do was it had to build it and then verify. Open up the browser, spin it around, see if it works, see if it's rendering properly, and then it kept iterating until we finally got this version. And as you can see still, like it's not perfect. There's some things we want to change. I think it was supposed to be see-through like this so we could like actually go look inside. But this is so much better than it would have been if I didn't give it that slash goal with the criteria, and I just said build me a 3D plane with, you know, 3.js. So that's one of the key takeaways here.

Agent loops and goals are not supposed to give you 100% perfect output. They're supposed to help you get much closer on the first try. And here's another great example of that with the whole subjectivity thing. Here's the last one I did, which is a a prompt that I had Claude Code make, a slash goal. It was looking at this famous picture of the Beatles Abbey Road. And then what I told it to do was recreate this without using image generation. So just recreating this using like HTML or CSS or whatever it wants to do. And then it goes through and it creates, you know, version one, version two, version three. And it ended up stopping after version seven. You can see the prompt here says, "If the average is above nine or equal to nine, then stop." And that's when you end. The other thing it said is hard cap on eight passes. So, it was getting near that cap either way. But, these images are not very good. What we can see though is that it did its verification. So, each time it went through and created the HTML, it had to actually put it in a browser and then it would take a screenshot of it. You can see here's the screenshot for number one, here's version two, here's version three, here's version four. So, we can see it in real time getting better and better with each version, with each iteration.

But still, this is the one that it gave me at the end and obviously that looks nothing like the picture. We've got the car here, we've got the trees, we've got the road, we've got yellow, black, gray, light blue, just like the actual image. If I go back here, did I say yellow? I meant to say white. White, black, dark gray, light blue. And so, obviously it's nothing like it. If we would have done this with image generation, it could have been probably much closer. But I just wanted to try how that would work with pure code. The point being, it had the verification checks, it had the ability to take screenshots and look through each of its iterations, understand how did this still not look like the reference image, and what changes do we need to make each time?

And so, that's why a loop is only going to be as good as it's done check, as the done criteria. So, there's two things you need to think about before you build your first loop or your goals. What does done mean? And then how will it check? Because let's say you're building an actual game, a game that you can open up on your PC and play. It would have to check that in many ways. It would have to check visually, it would have to check functionally, and it would have to play the levels and see if anything breaks. If you're writing some sort of like script, how does it check? It doesn't need to check visually, it just needs to check flow. It needs to check that it sounds like your tone of voice.

It needs to check in other ways. So, based on what you're building, the verification checks obviously look different and it's your job to make sure that your agents have the right tools in order to do those checks. And then of course, on the other side, what does done mean? Like I mentioned earlier, if you can get as objective as possible with a specific metric, then that's best. But sometimes you can't. Sometimes you have to say until you're 100% confident, right? And so like, my most common use of these loops is when I use hyper frames in Cloud Code to edit videos because I will basically chuck it in, do a slash goal, and it does everything for me. It has to get the transcript, cut out the mistakes and the pauses, it has to make the beats, it has to sync the beats, it has to obviously render them, and then it has a ton of verification on making sure that all of the beats are in bounds and that they line up with the transcript correctly.

And that is how you're able to see a lot of these people say, "Okay, I did this with one shot, with one prompt." Because it was a loop, because it had verification and iteration. So, what makes a loop actually work? A checkable goal, a hard stop, good tools, memory, a separate checker, planning first, logging, and then making it make sense with the cost. Because a lot of times these loops can run for a long time, and especially if you have a pretty hard goal, a goal that might take a lot of iteration, and then if the done criteria is also very hard, where maybe it just can't actually ever hit that, then that thing's going to run for a long time.

So, I've had a couple loops that have gone for 12 hours plus, and they're just not like super useful to me. Most of the time when I'm running loops that run for a while, it's usually more like these. It's usually things that take like 35 minutes or maybe a couple hours, but I don't need a loop that's going to run for 4 days straight. I just don't really need that. So, another kind of message that I'm trying to send here is just because you're seeing someone like Peter Steinberger saying something like this, doesn't actually mean that this applies directly to you and your use case.

Because he's a hardcore coder, he's building agents, he works at OpenAI. This probably makes a lot of sense for the way that he works and has probably 10xed his productivity. And that's the cool thing about AI is that it because it's going to seep into every single vertical and every single role, not everyone will use it the same. So, it's good to stay up to date with what people like Peter Steinberger are saying, but that doesn't mean you have to drop everything right now and go try it. Or maybe it's good to try it, but that doesn't mean you have to fully integrate it into every single Cloud Code session forever.

So, anyways, coming from a non-coding background, coming from a perspective of someone who uses Cloud Code all all time, 24/7, but I use it for knowledge work rather than massive database code base refactors and

building software and building apps every day. That's kind of the way that I feel about these agent loops, and I've been seeing a ton of stuff about them lately, so I felt like I needed to come in here and just share my opinions on it. Some of you guys may disagree with this, but that's the way that I've been using them because I do use them. I just don't go for those fancy runs that run for like 3 days straight. A lot of times if I have a big goal, I will shoot off a nice chunky loop before I go to bed, and I can wake up with something that's ran for maybe 4 or maybe 8 hours, and that is truly very beneficial. But a lot of that stuff is more experimental for me, and then I'm able to take that output I got from the overnight run, and then chuck it back into some more loops or iterate on that myself as a human. So, there's a little bit more detail that was covered in this slide deck as well as this full audit, which is way more wordy and super ugly to look at, but I will attach both of these sources in my free school community if you guys want to check all that out. The link for that is down in the description. You'll hop in the free school community, you'll go to classroom, you'll click on all YouTube resources, and you can find everything in there. But, that's going to do it for today. So, if you guys enjoyed the video or you learned something new, please give it a like. Helps me out a ton. And as always, I appreciate you guys making it to the end of the video, and I'll see you on the next one.

Thanks, guys.