Hey, what's going on Internet, If you've been using Cloud four over the last couple of days and you've seen that some prompts give you amazing results and some prompts give you kind of average results, you're in the right place.

Cloud Four and many of these advanced reasoning models such as O3, Gemini 2.5 Pro, etcetera, there's a new type of prompting needed to get the most out of these models.

And these are completely different from the older models and even sometimes contradictory to what people used to do for prompting those models.

And the interesting thing is most people don't know about these ideas and how to apply them.

In this video, I'm going to show you the exact prompting techniques to get the most out of cloud four and many other these advanced reasoning models.

With that being said, let's get into it.

As always, we have a beautiful image given to us from Open AI.

Here we have a representation of Cloud 4 as a cat.

All right, so there's a lot that I want to walk you through.

And if we zoom out, there's 12345 macro points that are specific to cloud four to get the most out of the model in certain use cases.

And we'll start over here, which I feel is probably the most important point.

But before we do that, most of the content I pulled is from this blog post from Anthropic.

So these are techniques that Anthropic recommends people use when prompting their models.

And this is coming directly from the research team that created these models.

All right, so the First things first is being specific.

Now, this is very similar to many of the other advanced models and specificity is key and will likely continue to be key going forward.

Reason being, as many of these models that we're using today, they're very, very, very good at following instructions, which means they're going to do exactly as you say.

So being specific in your word usage and and the words you choose is critical.

And that's what I'm calling out down here.

More specifically, specifically, get it?

All right.

So be specific on behavior, be specific on instructions, and be specific on features that you ask for.

And in the end, words matter.

So the words that you use in your prompts matter more today than they have ever in the past.

Because often times in the past, older models, they would need to be reminded consistently throughout a prompt and you'd have to scream things at the model and you'd have to put things in all caps.

You'd even have to sometimes threaten the model to get it to do what you want it to do.

But in this case, you no longer need to do that because these models do exactly as you ask based on the words you've provided.

Now, I'm going to give you a few examples of what this looks like.

We'll start on the left and work our way right.

So the first one here is modifiers.

So with modifiers, you can think of these as extending a statement in different ways to add specificity to what you're trying to get from the model.

What does this mean?

So a basic prompt.

So this is kind of a like a bad example, so don't do this.

But a basic prompt is create an analytical dashboard.

So this is something we've asked the model to do.

A better way of saying this would be to create analytical dashboard, include as many relevant features and interactions as possible.

Go beyond the basics to create a fully featured implementation.

Now we can break this prompt down and pull out the modifiers that were used in this example.

So the modifiers used in this example, starting with the scope modifier.

So the scope modifier is saying include as many relevant features and interactions as possible.

So we're saying interactions and features, and we're doing as many as you can.

So we're expanding the scope as broad as possible for this analytical dashboard.

The next modifier here is depth.

So how deep do we want it to go?

And here we can see we're saying going go beyond the basics.

So don't just give me a basic dashboard, go beyond that and try to get as creative as possible.

And then the last one here is completeness.

And we're saying give me a fully featured implementation.

So I want all the different bells and whistles associated to a dashboard.

These are different modifiers that we can tack onto our prompt when we're giving our model a task to get a higher quality output from it.

So this is the first one as modifiers.

Now our next example here is examples.

So examples matter.

Now, what do I mean by this?

Well, when you're giving a prompt to an AI, esecially if you're building something out that's more sohisticated and you have a system prompt in the background, oftentimes people use this concept of few shot learning.

And if you shot learning basically means that in your prompt you're going to bake in a series of examples.

So you're going to say, do this task, do it in this way, you know, give me this output format, etcetera, etcetera.

And at the end of that you're going to say, here's some examples to get you started.

And those examples you can have between probably 1 and 100 depending on the size of the example.

But just imagine you have all these examples here, and you would bundle these at usually probably near the end of your prompt, depending on the model you're using.

And the critical part here is ensuring that when you're using examples that they're aligned with the behavior you want from that model, and you want to encourage and minimize behaviors that you'd want to avoid.

And the reason this is important is that if the model falls and follows instructions very effectively, the examples you provide are critical because if you provide an example you think is in line with the behavior you're seeking, but it's not because there's some small nuance that you've missed in that statement or example, then the model is going to follow that and go in the wrong way.

So that means that you're going to have to be very vigilant about the examples that you provide to the model.

And you're going to have to be very specific and, and look into those and, and be scrutinizing every single one example you provide.

So that's another important year.

So these additional two that I've listed here in purple, the reason I've listed these in purple is both of these examples are common for other advanced reasoning models.

So the ones in orange are one specific to cloud four.

And these are common across all advanced reasoning models.

So our first one here is XML tags.

So these are informal speak.

These are basically called delimiters and a delimiter.

I have no idea if I spelled this right, but we'll hope that we did.

A delimiter is basically a way to segment a prompt to allow the AI to know what part of the prompt is where.

So say we have say this is our prompt, right?

These these lines.

And in here we've, we've separated these out in segments.

So say these are three segments.

So this is segment 1-2 and three.

And we want the prompt to read the system prompt in that way.

So we wanted to see this is a segment one, This is segment 2.

And the way we delineate those and and and separate them is through delimiters.

So we can use XML tags to separate out these different segments based off of what we want the AI to understand.

In addition to delineating a system prompt, we can also, as the examples listed here, give the AIA prompt stating that it should use XML tags to separate parts of its thought process.

So this is the, the first point I've just given you.

So this example here, this one is what we give the AI and then this one, this, this alternative example is what we're getting from the AI, but we're forcing the AI to use these tags to think in different ways.

So in this case, we're asking it to write its prose section in your, in your response and a smooth flowing prose paragraph tags.

So we're basically stating that when you respond to me, the prose you provide should be inside of those tags.

If there's any additional commentary or things around that, it should be outside the tags.

And this is used for for especially if you're using AP is and programmatic ways of using these models, because you can then go into that response we've gotten from the AI and just extract out what's inside the tags instead of getting the additional commentary around it.

And this is this applies to all advanced racing models.

And then the next one is what to do.

So this is very similar across other models as well, where in the past when we used to use these models, we used to yell at them and say never to do something.

So this is this is common.

And I think GPT four O 4.1 actually does does well with knots.

So if you do a negating term of saying never do something or don't do something or not, etcetera, these negating terms, it does OK with that, but it's better to state what you want it to do.

So a bad example here is do not use markdown in your response.

It's a bad example, but since these models can follow instructions very effectively, then that means that today, instead of doing a knot, because this is what we used to do for the old models, we can actually state what we want instead of what we don't want.

And in this case, we can say your response should be composed of smoothly flowing pros and paragraphs.

So we're not saying anything about markdown, we're just saying exactly what we want from the model.

And since the model follows instructions so effectively, it's going to do a good job at doing this for us.

And then I think this might be our last.

That's pretty intense.

All right, our last one here is lead by example.

And I thought this is pretty interesting because this is a very cloud specific example or a cloud specific tactic for prompting.

And what do we mean by this?

It's actually a good tactic when writing your prompt to the AI model is writing in the way that you want the AI to respond to you.

So that means that your prompt, it's likely going to influence how the model responds to you, not just what you say, but how you say it, how it's structured, the tone associated to it, all of these things.

It's very similar to how a child mimics their parent.

So a parent may tell a kid not to use their phone, but if the kid sees their parent using the phone all the time, then they're going to use their phone as well.

So it's important that you lead by example instead of just saying what you want.

You have to act as if you kind of behave the way you want the model to behave as well.

And that's exactly what this tactic is stating as if you come down here, you can see Claude is often matching your prompt style and your desired output style.

And it's important to, for example, if you wanted to remove markdown, make sure that in your prompt, you reduce the volume of markdown in the output.

So the prompt you give it, you don't include markdown.

So it's interesting how lead by example is a very specific thing for Claude, but I'm sure this will become more prevalent in other models as well.

So these are the big components to being specific.

If you had to break it down into elements.

The next example I wanted to give you here is improved visuals.

So historically, Clot has probably been one of the better models when it comes to creating UI designs for different applications and things like that.

Well, we can improve that even more now with these models through 2 tactics.

1 is encouragement, which is kind of like strange and funny.

And the other one is modifiers, again, similar to what I've talked about in the past.

So if you look at the encouragement side, you can see that one of the examples that Anthropic gave was encouragement stating that don't hold back, give it your all.

Now this has been a tactic people have used in the past, but my guess is that it's more effective now because the model follows instructions more effectively.

And then the other one is modifiers that we've already talked about.

What I'll give you some examples for UI specifically.

So these modifiers for UI is this first one is include as many relevant features and interactions as possible.

So again, we're talking about scope here.

So we're saying as many relevant features and interactions.

Another one here is add thoughtful details like Hoover states, transitions and microinteractions.

So we're giving specifics on what we want.

So again, this is specific specificity.

We're saying exactly what we want from the model to provide us.

And then last is applied applied design principles, hierarchy, contrast, balance and movement.

Again, these are specifics.

So these are different modifiers you can add to your prompt to get more improved visuals back from it.

OK, so our next is cleaning up bread crumbs.

So this is a very cloud four specific thing.

I'm sure other models will do this in the future, but I'd not for my what I've seen and don't do this today.

So what happens with cloud four, especially in an agentic use case?

So I'm let me call that out specifically.

So this is often for agentic coding use cases, but it can manifest in other use cases that are more agentic in nature.

And what we mean by agentic is autonomous.

So the model is going off and doing a series of things autonomously without you intervening in its actions.

And when it goes through these actions autonomously, sometimes it'll create new files just for testing and iteration purposes.

So you can think of these files as a kind of placeholders as it progresses.

So this is basically a journey.

As the AI is going on this journey, it's it's creating documents along the way.

And these documents are useful because it can reference back to the document to see what has happened and what it's worked on for that specific phase of the project.

O it's giving itself a plan and also kind of bread crumbs to refer back to, to see what it's done so far and where it needs to go next.

And when it creates these documents, these documents act as temorary scratch ads before saving its final outut.

So it's final output will be the end code that it writes, but it'll have also often markdown files as it's coding things up.

And I've been building out probably for the last week or so with Cloud Four, and I've gotten a lot of experience in seeing this in action where the AI writes a bunch of markdown files summarizing what it's done so far and what it has to go.

And then it'll refer back to those as it's coding.

This is useful, entertaining, but also it's frustrating if you have 55 markdown files inside of your project, all of which were relevant for a certain point and are no longer relevant today because you've already built those parts of the project out.

So in that case, you can add a prompt to your model, or in this case, you could add it to your rules.

So if you're using cursor, you can add it to a global rule or project rule, or if you're using Windsor, if you can do the same there and also an aider.

So if you're using any of these for coding purposes and you want to clean it up, you can add this into your project rules where you can say if you create any temporary new files, scripts or help helper files for iteration, clean up these files by removing them at the end of the task.

And from my experience, this works pretty effectively.

So once it's done and it's achieved its task and it's passed its test and it's been validated, it'll then go back and delete those files so you don't have them sitting in your repository.

So it's just one prompting tactic to clean up those bread crumbs.

All right.

Our next one is iterative thinking.

So for iterative thinking, this is something that O3 does very well and that's why I've added a little opening eye logo to the ears of these robots or antennas, however you want to see it.

So O3 has done this historically.

And O3 is kind of one of the first models to introduce this to the world through its researching capabilities where it, when it researches things, it'll go off and research something.

It thinks, researches, thinks, researches, thinks, etcetera.

Now, Cloud Four does this very well, specifically when coding and also researching and doing a variety of other things.

And this for me, I would say is probably one of the biggest jumps in intelligence and usefulness of AI models so far, probably in the last couple, I would say maybe last year or so.

And reason being is that it's enabling the model to not just think about something, but to use a tool, figure out what the output is from that tool, and then reason over that response and then figure out what tool to use next.

So it's really enabling these models to be somewhat agentic and autonomous.

How does this come into prompting specifically?

Well, you can enforce this area of thinking.

So if sometimes the model doesn't do this and you want it to to get improved outputs, you can somewhat enforce the model to do this.

So this is the prompt you can add to the model to enforce that action.

So here we say after receiving tool results, carefully reflect on their quality and determine optimal next steps before proceeding.

Use your thinking to plan and iterate based on these new tool, this new information, and then take the best next action.

So here we're basically saying taking the new information, take the next, next best action after you've used the tools and after you've gotten new information from those tools.

So this is something you can bake into your system prompts to ensure that the AI does this effectively.

And you can do this in a variety of ways.

And you can probably encourage it through projects as well.

So if you're using cloud projects, you can bake that into your, let me spell this correctly, cloud.

If you're using GPT, you can use GPT projects in this way.

I've not tried this yet, but you could probably try it with gems as well with Gemini and add this to your system instruction for that project to see if it actually pulls out that air to thinking that you're seeking to get better responses from, especially if you're doing a research task or a coding task.

OK.

And our last point here is somewhat of an emergent property associated to these models when it comes to their tool calling abilities and their intelligence and how that increases over time.

And this has a little bit to do with the point.

But first, this is a great movie.

You should watch it.

Everything, everything, everywhere all at once.

And how is this connected?

Well, in my research on this topic that I'm going to share with you is superposition is mentioned multiple times and how superposition is a term adapted by ML researchers for this use case.

So what's happening here is our AI is going to call multiple tools all at once, opening up a new dimension of thinking where they shared, where the where the context is shared and its brain like kind of continuously.

So what I mean by this in simple terms, in the past, what models would do?

So say this is our AI in this box, it would call a tool, it would wait for the tool to respond.

And once it got its response, then it would call another tool, it would wait for that response, and then it would keep doing this over and over.

So everything is sequential.

So basically calls the tool, waits, calls the tools, waits, calls the tools, waits.

By doing this, we're wasting a lot of mental space for the AI because it has to segment that space in its brain to to a waiting period before it gets the response.

So this is wasted energy.

What happens now with at least cloud four models is what the model does is it calls all the tools all at once.

And when it calls all the tools all at once, it gets all the responses somewhat simultaneously.

And by getting all the responses simultaneously, it's able to synthesize those responses and share context.

And it's basically in its head throughout all the responses and match and and compare the responses to figure out exactly what it should do next based off of the collective response instead of just one response at a time.

And this has has shown to improve the ability of the model to take effective actions and do more complex tasks overtime.

And yeah, this is basically what I've stated reviously.

And down here is a cool little way that we can try to enforce or yeah, enforce basically this action where this is a prompt that you can add to your system prompt where you state for maximum efficiency, whenever you need to perform multiple independent operations, invoke all relevant tools simultaneously rather than sequentially.

So we're enforcing that it uses a simultaneous call instead of the sequential call to get the best response and most efficient response.

So this is just another way to get the most out of our AIS when using when using tools Internet.

That's the video.

So I hope you enjoyed this.

Hope you found this entertaining, educational.

If you did, please reshare it with your friends.

And if you'd like to work with me, I have a company called Gradient Labs, an organization that helps other companies implement AI to automate different processes, saving time, money and generating revenue.

So if that's at all interesting to you, below is a link where you can book a free discovery call for 30 minutes to see if there's a good match between the two of us.

With that being said, Internet, I'll see you next time.