# Prompting 101 | Code w/ Claude

24 MIN · YOUTUBE · HTTPS://WWW.YOUTUBE.COM/WATCH?V=YSPBXH0LPIE
https://www.youtube.com/watch?v=ysPbXH0LpIE

SUMMARY

*Hannah and Christian from Anthropic present a workshop on prompt engineering, focusing on best practices for creating effective prompts for language models like Claude. They illustrate their points through a real-world scenario involving car accident reports, emphasizing the importance of clear instructions, context, and iterative refinement in prompt design.*

- *Prompt engineering is the practice of crafting clear instructions to communicate effectively with language models.*
- *A structured prompt should include a task description, content, detailed instructions, examples, and a summary of critical information.*
- *Iterative testing and refinement of prompts are essential for improving model responses.*
- *Providing context about the task and expected tone helps the model generate more accurate outputs.*
- *Using structured formats (like XML or JSON) can enhance clarity and organization in model responses.*
- *Including examples and conversation history can enrich the context and improve the model's performance.*
- *Output formatting guidelines help ensure that the model's responses are concise and usable for applications.*
- *Extended thinking features in models allow for deeper reasoning, which can be leveraged to enhance prompt effectiveness.*

Hi everyone. Thank you for joining us today for
prompting 101. Uh my name is Hannah. I'm part of the applied AI team here at Anthropic. And with me is Christian, also part of the applied AI team. And what we're going to do today is take you through a little bit of prompting best practices. And we're going to use a real world scenario and build up a prompt together. Uh so a little bit about what prompt engineering is. uh prompt engineering. You're all probably a little bit familiar with this. This is the way that we communicate with a language model and try to get it to do what we want. So, this is the practice of writing clear instructions for the model, giving the model the context that it needs to complete the task, and thinking through how we want to arrange that information in order to get the best result. Um, so there's a lot of detail here, a lot of different ways you might want to think about building out a prompt. Um, and as always, the best way to learn this is just to practice doing it. Um, so today we're going to go through a hands-on scenario. Uh, we're going to use an example inspired by a real customer that we worked with. So, we've modified what the actual customer asked us to do, but this is a really interesting case of trying to analyze some images and get uh factual information out of the images and have Claude make a judgment about what content it finds there. And I actually do not speak the language that this content is in, but luckily Christian and Claude both do. Uh so I'm going to pass it over to Christian to talk about the scenario and the content. So for this example that we have here, it's uh intended so so to set the stage, imagine

you're working for a Swedish insurance company and you deal with uh car insurance claims on a daily manner. Um and the purpose of this is that you have two pieces of information. Um we're going to these in detail as well, but visually you can see on the left hand side we have a car accident report form. um just detailing out what transpired before the action accident actually took place. And then finally, we have a sort of human drawn um sketch of how the accident took place as well. So these two pieces of information is what we're going to try to pass on to cloud. And to begin with, we could just take these two and throw them into a console and just see what what happens. So if we transition over to console as well, we can actually do this in a real manner. And in this case here, you can see we have our shiny beautiful entropic console. We're using the new claw for solid model as well. In this case, setting temperature zero and having a a huge max token budget as well. Just helping us make sure that there's no limitations to what CL can do. In this case, you can see I have a very simple prompt just setting the stage of what Cloud's supposed to do. in this case mentioning that this is um intend to review a an accident report form uh and eventually also determine um what happened in an accident and who's at fault. So you can see here with this very simple prompt if I just run this let me go to preview. Uh we can see here that Claude thinks that this is in relation to a skiing accident that happened on a street called Chappangan. It's a very common street in Sweden.

Um and in many ways you can sort of understand this innocent mistake in the sense that in our prompt we actually haven't done anything to set the stage on what is actually taking place here. So this sort of first guess is not too bad but we still notice a lot of intuition that we can bake into cloud. So if we switch back to the slides you can see here that um in many ways prompt engineering is a very iterative empirical science. Uh in this case here, we could almost have a test case where Claude is supposed to make sure it understands it's in a car or vehicular environment, nothing to do with skiing. Uh and in that way, you iteratively build upon your prompt to make sure it's actually tackling the problem you're intending to solve. Um and to do so, we'll go through some best practices of how we we at Anthropic break this down internally and how we recommend others to do so as well. So, we're going to talk about some best practices for developing a great prompt. Uh, first we want to talk a little bit about what a great prompt structure looks like. So you might be familiar with kind of interacting with a chatbot with Claude going back and forth having a more kind of conversational style interaction. When we're working with a task like this, we're probably using the API and we kind of want to send one single message to Claude and have it nail the task the first time around without needing to uh kind of move back and forth.

Uh, so the kind of structure that we recommend is setting the task description up front. So telling Claude, "What are you here to do? What's your role? What task are you trying to accomplish today?" Then we provide content. So in this case, it's the images that Christian was showing, the form and the drawing of the accident and how they occurred. That's our dynamic content. This might also be something you're retrieving from another system, depending on what your use case is. We're going to give some detailed instructions to Claude, so almost like a step-by-step list of how we want Claude to go through the task and how

we want it to um tackle the reasoning. We may give some examples to Claude. Here's an example of some piece of content you might receive. Here's how you should respond when given that content. And at the end, we usually recommend repeating anything that's really important for Claude to understand about this task. Kind of uh reviewing the information with Claude, emphasizing things that are extra critical and then telling Claude, "Okay, go ahead and do your work." So, here's another view. This has a little bit more detail, a little bit more of a breakdown, and we're going to walk through each of these 10 points individually and show you how we build this up, um, in the console. So, the first couple things, um, Christian's going to talk about the task context and the tone context. Perfect. So, yeah, if we begin with the task context, as you realized when I went through a little demo there, um, we didn't have much elaborating what what scenario Chlo was actually working within. And because of that, you can also tell that claw doesn't necessarily need to guess a lot more on what you actually want from it. So in our case, we really want to break that down, make sure we can give more clear-cut instructions. Um, and also make sure we understand what's the task that we're asking Claw to do. Um, secondly, as well, we also make sure we add a little bit of tone into it all. Um, key thing here is we want Claw to stay factual and to stay confident. So if uh, Claw can't understand what it's looking at, we don't want to guess and just sort of mislead us. We want to make sure that any assessment and in our case we want to make sure that we can understand who's at fault here. We want to make sure that assessment is as clear and as confident as possible. If not, we're sort of losing track of what we're doing. So if we transition back to the the console, um we can jump to a V2 that we have here. So I'll just navigate to V2. And you can see here um I'll also just illustrate the data because we didn't really do that last time around just to really highlight what we're looking at. So, what we're seeing here, this is the car accident report form, and it's just 17 different checkboxes going through what actually happened. You can see there's a vehicle A and vehicle B, both on the left and right hand side. And the main purpose of this is that we want to make sure that Claude can understand this manually generated data to assess what's actually going on. And that is uh corroborated by if I navigate back here to this sketch that we can highlight here as well. In this case, the form is just a different um data point for the same scenario. Um and in this case here, I want to bake in more information into our version two. Uh and by doing so, I'm actually elaborating a lot more on what's going on. So, you can see here I'm specifying that uh this AI assistant is supposed to help a human's claim claims adjuster that's reviewing car accident report forms in Swedish as well. Um, you can see here we're also elaborating that it's a human-driven sketch of the incident and that you should not um make an assessment if it's not actually fully confident.

And that's really key because if we run this,
you'll see that and you can see it's the same settings as well. Clo my new shiny model zero temperature as well. If we run this, we can see here what actually happens in this case. Um, CL is able to pick up that uh now it's relating to car accidents, not skiing accidents, which is great. We can see it's able to pick up that vehicle A was marked on on checkbox one and then vehicle B was on 12. Um, and if we scroll down though, we can still tell that there's some information missing for claw to make a fully confident determination of who's at fault here. And this is great. This is pertaining to a task set. Make sure you don't make anything any claims that aren't um uh factual

and make sure you you only sort of assess things when you're when you're confident. But there's a lot of information we're still missing here.

um regarding the form uh what the form actually
entails and a lot of that information is what we want to want to bake into this LM application as well and the best way of doing so is actually adding it to the system prompt which Hannah will elaborate on. Um so back in the slides uh we have the next item we're going to add to the prompt and this is um background detail data documents and images and here as Christian was saying we actually know a lot about this form. the form is going to be the same every single time. The form will never change. And so this is a really great type of information to provide to Claude to tell Claude, here's the structure of the form you'll be looking at. We know that will not ever alter between different queries. The way the form is filled out will change, but the form itself is not going to change. And so this is a great type of um information to put into the system prompt. Also a great thing to use prompt caching for if you're considering using prompt caching. This will always be the same. And what this will help Claude do is spend less time trying to figure out what the form is the first time it sees the form each time. And it's going to do a better job of reading the form because it already knows um what to expect there.

So another thing I want to touch on here is how we
like to organize information in prompts. So Claude really loves structure, loves organization. That's why we recommend following kind of a standard structure in your prompts. And there's a couple other tools you can use to help Claude understand the information better. I also just want to mention all of this is in our docs with a lot of really great examples. So definitely take pictures, but if you forget to take a picture, don't worry. All of this content is online with lots of examples and definitely encourage you guys to check it out there too. Um anyway the uh so some things you can use delimiters like XML tags also markdown is pretty useful to Claude but XML tags are nice because you can actually specify what's inside those tags. So we can tell Claude here's here's user preferences. Now you're going to read some content and these XML tags are letting you know that everything wrapped in those tags is related to the user's preferences and it helps Claude refer back to that information maybe at later points in the prompt. Um, so I want to show in the back in the console how we actually do this in this case. And Christian's going to pull up our version three. So we're keeping everything about the other part of the user prompt the same. And we've decided in this case to put this information in the system prompt. You could try this different ways. Uh, we're doing it in the system prompt here. And we're going to tell Claude everything it needs to know about this form. So this is a Swedish car accident form. The form will be in Swedish. It'll have this title. It'll have two columns. The columns represent different vehicles. We'll tell Claude about each of the 17 rows and what they mean. You might have noticed when we ran it before, Claude was reading individually each of the lines to understand what they are. We can provide all of that information up front. And we're also going to give Claude a little bit of information about how this form should be filled out. This is also really useful for Claude. We can tell it things like, you know, humans are filling this form out basically. So, it's not going to be perfect.

People might put a circle. They might scribble.
They might not put an X in the box. There could be many types of markings that you need to look for when you're reading this form. Uh we can also give Claude a little bit of information about how to interpret this or what the purpose or meaning of this form is. And all of this is context that is hopefully really going to help Claude um do a better job analyzing the form. So if we run it, everything else is still the same.

So we've kept the same user prompt down here.
Oh, your scroll is backwards from mine. Uh, the we have the same user prompt here. Still asking Claude to do the same task, same context. And we'll see here that it's spending less time. It's kind of narrating to us a little bit less about what the form is because it already knows what that is. And it's not concerned with kind of bringing us that information back. It's going to give us a whole list of what it found to be checked, what the sketch shows. And here Claude is now becoming much more confident with this additional context that we gave to Claude. Claude now feels it's appropriate to say vehicle B was at fault in this case based on this drawing and based on this sketch. So already we're seeing some improvement in the way Claude is analyzing these. I think we could probably all agree if we looked at the drawing and at the list that vehicle B is at fault. Um so we'd like to see that.

Uh so we're going to go back to the slides and
talk about a couple of other items that we're not really using in this prompt um but can be really helpful to building up uh building up your prompt and making it work better. Exactly. I think um one thing that we really highlight is examples. I think examples or few shot is a mechanism that really is powerful in steering cloud. So you can imagine this um in in quite a non-trivial way as well. So imagine you have scenarios, situations, even in this case concrete accidents that have happened that are um tricky for claw to get right.

But you with your human intuition and your human
label data um is able to actually get to the right conclusion. Then you can bake that information into the system problem itself by having clear-cut examples of a the data that that it's supposed to look at. So you can have visual examples. you can just base 64 encode a a an image and have that as part of the data that you're passing along into the examples and then on top of that you can have the sort of depiction or description rather of how to break that down and understand it. This is something we really highlight and and emphasize in how you can sort of push the limits of your LLM application is by baking in these examples into system prompt. And this again is sort of the empirical science of prompt engineering that you sort of always want to push the limits of your application and get that feedback loop in where it's going wrong and try to add that into system prompt so that next time when example that sort of mimics that u takes place it's able to actually reference it in its example set. You can see here as well, this is just a little example of how we do this. Again, really emphasizing the sort of XML structure that we we um we enjoy. It's it gives a lot of structure to the clone. It's what it's been fine-tuned on as well. Um and it works perfectly well for this example. And in our case, we're not doing this just because it's a simple demo,

but you can realistically imagine if you were building this for an insurance company, you'd have tens, maybe even hundreds of examples are quite difficult, maybe in the gray, that you'd like to make sure that Claude actually has some basis in to make the verdict next time. Um, another topic we really want to highlight, which we're not doing in this demo, is conversation history. It's in the same vein as examples. uh we use this to make sure that the enough context rich information is at close disposal when it when when closing on on on your behalf. Um in our case now this isn't really a userfacing LLM application. It's more something happening in the background. You can imagine for this insurance company they have this automated system some data is generated out of this and then you might have a human in the loop at towards the end. If you were have to build something much more userf facing where you'd have a long conversation history that would be um relevant to bring in this is a perfect place in the system prompt to include because it enriches the context that Claude works within. Um in our case we haven't done so but what we do is and the next step is try to make sure we give a concrete reminder of the task at hand.

So, now we're going to build out the
final part of this prompt for Claude, and that's coming back to the reminder of what
the immediate task is and giving Claude a reminder about any important guidelines that we want it to follow. Some reasons that we may do this are a preventing hallucinations. Um, so we want Claude to uh not invent details that it's not finding in this prompt, right? Or not finding in the data.
If Claude can't tell which form is checked, we don't want Claude to take its best guess or
invent the idea that a box might be checked when it's not. If the sketch is unintelligible, the person did a really bad job drawing this drawing and even a human would not be able to figure it out. We want Claude to be able to say that. And so these are some of the things we'll include in this final reminder and kind of wrap up step for Claude. Uh remind it to do things like answer only if it's very confident. We could even ask it to refer back to what it has seen in the form anytime it's making a factual claim. So if it wants to say vehicle B turned right, it should say I know this based on the fact that box two is clearly checked or whatever it might be. We can kind of give Claude some guidelines about that. So if we go back to the console, we can see the next version of the prompt and we're going to keep uh we're going to keep everything the same here in the system prompt. So, we're not changing any of that background context that we gave to Claude about the form, about how it's going to fill everything out. We're not changing anything else about the context and the role. We're just adding this detailed list of tasks. And this is how we want
Claude to go about analyzing this. And a really key thing that we found here as we were building this demo and when we were working on the customer example is that the order in which Claude analyzes this information is very important. And this is analogous to way you might think about doing this.
If you were a human, you would probably not look at the drawing first and try to understand what was going on, right? It's pretty unclear. It's a bunch of boxes and lines. We don't really know
what that drawing is supposed to mean without any additional context. But if we have the form and we can read the form first and understand that we're talking about a car accident and that we're seeing some checkboxes that indicate what vehicles we're doing at certain times, then we know a little bit more about how to understand what might be in the drawing. And so that's the kind of detail that we're going to give Claude here is to say, "Hey, first go look at the form. Look at it very

carefully. Make sure you can tell what boxes are checked. Make sure you're not missing anything here. Um, make a list for yourself of what you see in that. And then move on to the sketch. So after you've kind of confidently gotten information out of the form and you can say what's factually true, then you can go on and think about what you can gain from that sketch. keeping in mind your understanding of the accident so far. So, whatever you've learned from the form, you're trying to match that up with the sketch. And that's how you're going to arrive um at your final uh at your final assessment of the form. And we'll run it.

And here you can see one behavior that
this produced for Claude because I told it to very carefully examine the form. It's
showing me its work as it does that. So, it's telling me each individual box. Is the box
checked? Is it not checked? And so, this is one thing you'll notice as you do prompt engineering. In our previous prompts, we were kind of letting claw decide how much it wanted to tell us about what it saw on the form here. Because I've told it carefully examine each and every box, it's very carefully examining each and every box. And that might not be what we want in the end. So, that's something we might change. Um, but it's also going to give me these other things that I asked for in XML tags. So, a nice analysis of the form, the accident summary so far. It's going to give me a sketch analysis, and it's going to continue to say that vehicle B appears to be clearly at fault. In this in this example, it's pretty simple example with more complicated drawings, more uh less clarity in the forms. This kind of step-by-step thinking for Claude is really impactful in
its ability to make a correct assessment here.

Uh, so I think we'll go back to the slides and
Christian's going to talk about a last kind of piece that we might add to this um to really make it useful for a real world task. Indeed. Thank you so much. So, as Hannah mentioned, uh, we sort of set the stage in this prompt to make sure that really acting on our behalf in a right manner. Um, and a key step that we also add towards the end of this prompt that I'm going to show you in a second is a simple sort of guidelines or reminder part as well. just strengthening and reinforcing exactly what we want to get out of it. And one important piece is actually output formatting. You can imagine if you're a data engineer working on this LM application, all the sort of fancy preamble is great, but at the end of the day, you want your piece of information to to
be stored in, let's say, your SQL database, wherever you want to store that data. And the rest of it that is necessary for cloud to sort of give its verdict isn't really that necessary for your application. You want the nitty-gritty information for your application. So if we transition back to the console, you'll see here that we just added a simple importance guidelines part. And again, this is just reinforcing the sort of mechanical behavior that we want out of cloud here. Want to make sure that the summary is clear, concise, and accurate. Want to make sure that nothing is sort of impeding in in in Claw's assessment apart from the data it's analyzing. And then finally, when it comes to output formatting, in my case here, I'm just going to ask Claude to wrap its final verdict. All other stuff I'm actually going to ignore for my application and just look at what it's actually assessing. And that is I can I can use this if I want to build some sort of analytics tool afterwards as well.

Or if I just want a clearcut um uh determination,
this is a way I can do so. So if I just run this here, you'll see it's going through the same sort of process that we've seen before. In this case, it's much more succinct because we've asked to be to summarize its findings in a much more straightforward manner. And then finally towards the end you'll see that it'll wrap my output in these final verdict XML tags. So you can see that during this demo we've gone from a skiing accident to sort of unconfident insecure outputs from perhaps a car accident in the second version to now a much more strictly formatted confident output that we can actually build an application around and actually help you know a real world um car insurance company for example.

U finally if we transition back to the um slides
another key way of shaping CL's output is actually putting words in CL's mouth or as we call it pre-filled responses. You could imagine that parsing XML tags is nice and all but maybe you want a structured JSON output to make sure that uh it's JSON serializable and you can use this in a subse subsequent call for example. Um this is quite simple to do. You could just add that um claude needs to begin its output with a certain format. This could be for example a uh open square bracket squarely bracket for example or even in this case that we see in front of us this would be an XML tag for itinerary. In our case it could also be that final verdict XML tag. Um, and this is just a great way of again shaping how Claude is supposed to respond. Um, without all the preamble if you don't want that, even though that is also key in shaping his output to make sure that Claude is reasoning through the steps that we wanted. So in our case here, we would just wrap it in the final verdict and then parse it afterwards. But you can use prefill as well. Now finally one step that I would like to highlight here as well is that both cloud 3.7 and especially cloud 4 of course is a sort of hybrid reasoning model meaning that there's extended thinking at your disposal. Um and this is something we want to highlight because you can use extended thinking as a crutch for your prompt engineering. Basically you can enable this to make sure that Claude actually has time to think. It adds his thinking tags and the scratch pad. Um and the beauty of that is you can actually analyze that transcript to understand how claude is going about that data. So as we mentioned we have these check boxes where it goes through step by step of the scenario that transpired for the accident. And in many ways there you can actually try to help claude in building this into the system prompt itself. It's not only more token efficient but it's a good way of understanding how these intelligent models that don't have our intuition actually go about the data that we provide them. And because of that, it's quite key in actually trying to break down how your system prompt can get a lot better. Um, and with that said, I think uh I'd like to thank all you for coming today. We'll be around as well.

So if you have any questions on prompting, please
uh please go ahead. I know there's a prompting. You want to learn more about prompting in an hour. We have prompting for agents and right now we have an amazing demo of Claude plays Pokemon. So don't go anywhere for that. And as Christian said, we'll be around all day. So, I know we didn't have time for Q&A in this session, but uh please come find us if you want to chat.
And thank you guys for coming. Thank you so much.