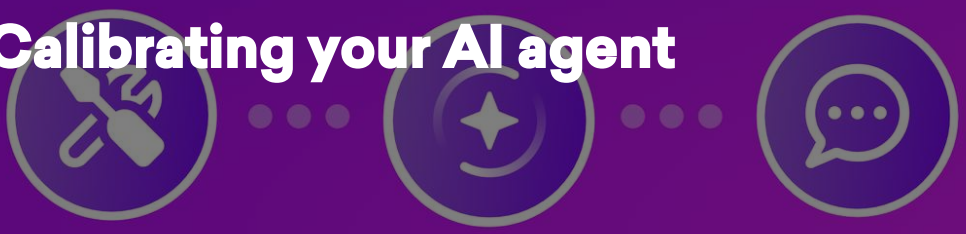
Unit 2 - Calibrating your AI agent



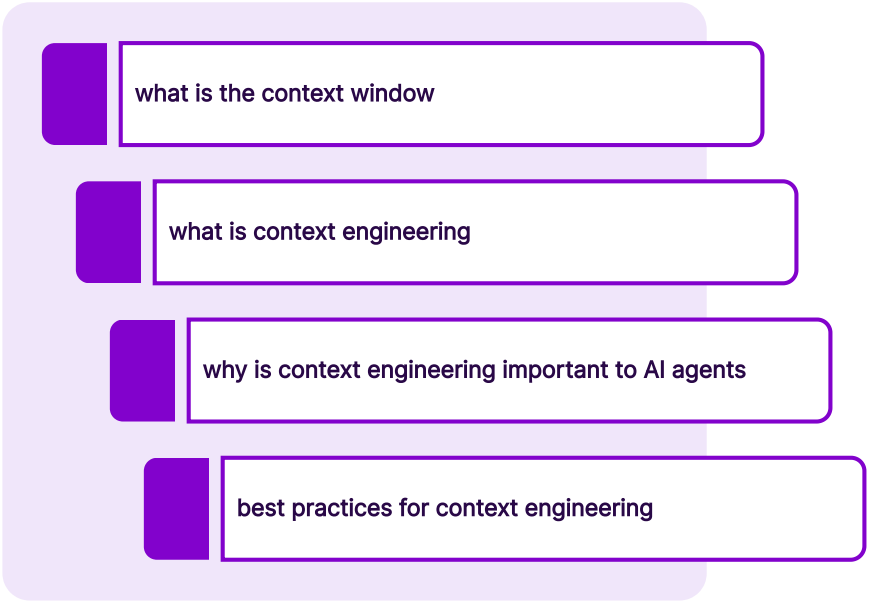
UNIT 2 - CALIBRATING YOUR AI AGENT

Unit 2
Calibrating your AI agent

2.1 Unit Introduction

Welcome to the second unit of the Context Engineering course!

In this unit, you will cover:



Let's dive in!

Continue to 2.2: Context window

2.2 Context window

In this unit, you'll learn how to fine-tune your AI agent by managing the information it processes.

Before you explore some techniques to calibrate your AI agent's context, you need to understand some essential background that affects every AI agent you build: the **context window**.

📌 Definition

The **context window** is the maximum amount of information an AI model can process at once, which is measured in tokens. This limit varies depending on which AI model you use, some handle more information than others.

The context window holds everything your AI agent needs to work: your **system prompt**, **context files**, **tool definitions**, **tool inputs and outputs**, and **conversation history**.

Managing the context window is critical because it determines whether your AI agent will work reliably and operate smoothly even when performing complex tasks.

Here are some reasons **why you need to actively manage your AI agent's context window:**

Context accumulates automatically

There's always a limit on how much context you can give your AI agent. Every interaction adds to the total context. Without active management strategies, your AI agent will exceed capacity limits.

It constrains your design decisions

The context window size determines how much you can pre-load, how long conversations can be, and how many tools you can define. Smaller limits require different strategies than larger ones. Design decisions must account for this constraint from the start.

Overflow of context impacts AI agent reliability

When context exceeds capacity, the system loses information unpredictably, which means even crucial data gets cut off. The AI agent cannot behave coherently when the system discards critical data.

model you use, the challenge is the same: finding the right amount of information to include.

Smaller context windows force you to be more selective. Every token counts, so you must prioritize carefully or you'll run out of space before including everything the AI agent needs.

Larger context windows give you more breathing room, but they introduce new challenges. With more space available, it's tempting to include too much information which leads to some failures you'll learn about soon.

When processing your request, AI models use **attention** to understand text. You covered this in the AI fundamentals course, so feel free to go back if you need a refresher. **Attention means the model calculates how every word relates to every other word, and which words are important for understanding other words**.

The context window determines the maximum number of words the model can attend to, and **as this number grows, the model must track more relationships, making it harder to focus on what truly matters**.

More information creates more difficulty for the model to process effectively. This is why strategic context management matters, you need to give the model only what it truly needs so it can focus its attention on the right information.

Context window management isn't about the size of your window, it's about finding the right balance of information within it:

🟡Too little context	🔴Too large context	🟢Optimal context
Lacks information	Overwhelmed	Right information
Incomplete guidance	Higher hallucination	Well organized
Makes uninformed decisions	Confuses information	Context strategically selected

Continue to 2.2.1: Reasons for context window failures

2.2.1 Reasons for context window failures

As the context window size grows, several error types begin to happen or increase in frequency.

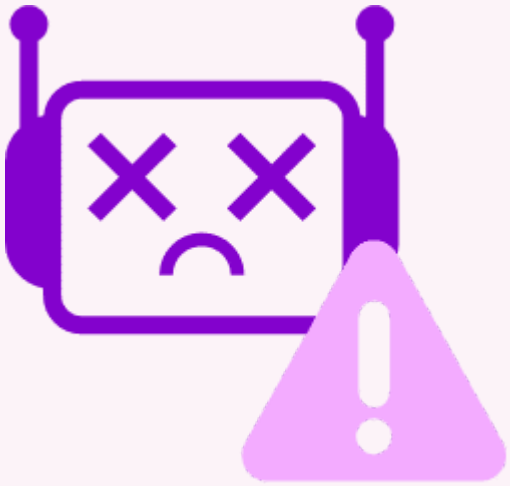
These failures are:

1. Context poisoning
2. Context distraction
3. Context confusion
4. Context clash

Let's discover each one.

1

Context poisoning



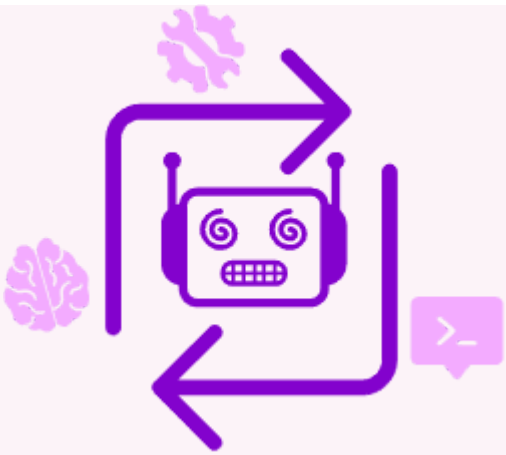
Context poisoning occurs when incorrect or irrelevant information in the context leads the AI agent to produce wrong outputs. With larger context windows, you can fit more information, but this also makes it easier for irrelevant or outdated data to contaminate the AI agent's decision making.

An AI agent helping with a refund policy question **has access to multiple context files**: the **current refund policy** (correct), an **outdated policy version from two years ago** (incorrect), and **unrelated shipping policies** (irrelevant). With a larger context window, all three files fit.

The AI agent may blend information from all sources. This produces incorrect guidance that mixes old and new policies.

2

Context distraction



Context distraction happens when too much context accumulates, such as **conversation history** or **tool results**. This causes the AI agent to repeat past actions rather than reasoning through the current request.

A customer reports a missing package to a support AI agent. After discussing delivery delays and addresses, the customer sends a message saying: **It just arrived, but it's damaged!**

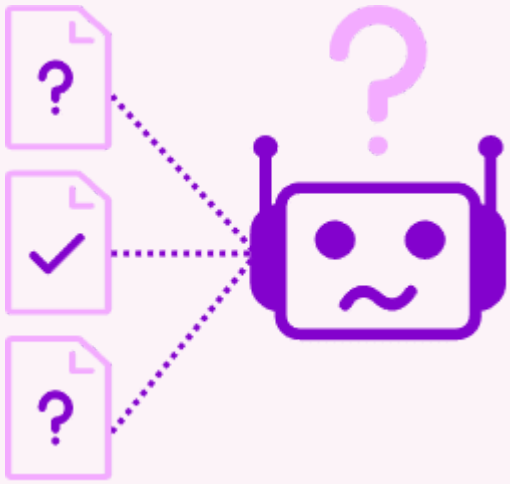
Since the conversation history is full of delivery discussions, the AI agent replies: **Let me check your shipping address**.

The **AI agent got distracted by the delivery pattern in context** and missed that the problem changed to the package being damaged.

3: Context confusion

3

Context confusion



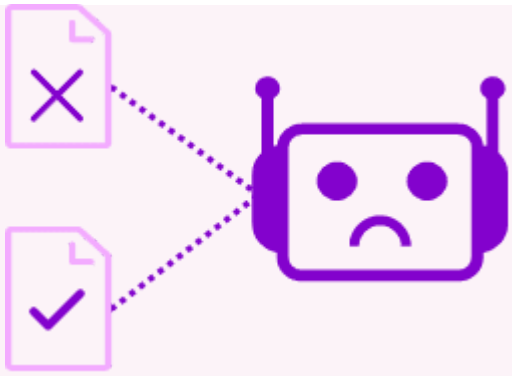
Context confusion occurs when irrelevant information clutters the context. This can cause the AI agent to pick the wrong tool or follow the wrong instructions.

A code review AI agent has **context files** containing **coding standards for multiple programming languages**. When reviewing code, the **AI agent references standards from the wrong language** because all files were present in context. The **irrelevant files cluttered the context**, causing the AI agent to apply incorrect guidelines.

4: Context clash

4

4. Context clash



Context clash happens when contradictory information in the context confuses the AI agent, causing it to produce inconsistent or incorrect outputs.

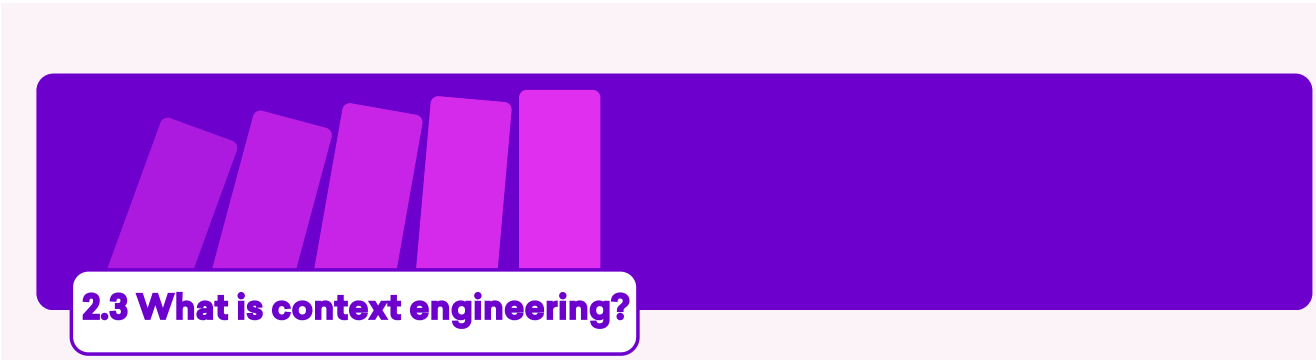
An HR assistant AI agent has a rule in its system prompt: **Vacation needs two weeks notice**. During conversation, a manager adds: **This week only, allow same-day requests**. The AI agent now has conflicting instructions: the permanent two week rule versus the temporary same-day exception. **It gives inconsistent answers**, either applying the exception as permanent policy or ignoring it entirely.

Context accumulates automatically with every interaction, and without deliberate strategies, your AI agent will either hit capacity limits or suffer

from context poisoning, distraction, confusion, or clash. The goal is to optimize what information you include.

Now that you understand the context window, let's explore context engineering, the practice that helps you calibrate your AI agent to work effectively within these limits.

Continue to 2.3: What is context engineering?



2.3 What is context engineering?

Now it's time to learn how to calibrate your AI agent through context engineering.

📌 Definition

Context engineering is the practice of selecting and managing the right information for your AI agent to guide its performance.

In the last unit, you covered prompt engineering and crafting effective prompts. You also learned what context is and its components: prompts, tool definitions, context files, tools inputs and outputs, and call history.

Context engineering is the extension of prompt engineering.

While prompt engineering focuses on writing good instructions, context engineering takes a broader approach that considers how the AI agent and all its components work together.

However, context has limits. **When an AI agent receives a request, it processes the available context information to understand the task.** The AI agent accumulates context while performing its task, which means the AI agent must handle increasingly larger amounts of information as the task progresses. Therefore, you need a strategy for managing it. The quality of context you provide determines the quality of outcome you receive.

Context engineering is the practice of making deliberate decisions about:

1. **What to include:** only add information the AI agent needs. Irrelevant information wastes space and can create unnecessary noise for the AI agent.

2. **Where to put information:** not all information belongs in the same place. Decide where each piece should be stored based on how often it's needed and how it's used.

3. **How to keep context manageable:** remove old and irrelevant information and make room for new. Context keeps growing and will eventually overflow if not actively managed.

When AI agents fail, you may focus on what's most visible: the prompt. You revise instructions, add more details, try different wording. But even a well-written prompt fails when you don't manage the surrounding context properly.

Think of it like an iceberg. The prompt is the tip, visible and easy to focus on. But beneath the surface lie the real issues that cause AI agent failures:

- **Running out of space:** The AI agent stops mid-task or cuts off critical information, producing incomplete or incorrect responses.
- **Poorly organized context:** The AI agent can't locate the right information quickly, so it uses the wrong data which leads to inaccurate answers.
- **Relevant information buried in noise:** The AI agent processes irrelevant details and may miss the key information it needs, resulting in responses that don't address the actual problem.

- **Too much information at once:** The AI agent becomes overwhelmed and produces confused or contradictory outputs, mixing information that shouldn't be combined or hallucinating details.

Several problems are involved in context management: limits, poor organization, conflicting data, and too much information. These issues are what determine whether your AI agent works or not. This is why context engineering is essential for building reliable AI agents.

Continue to 2.3.1: Why does context engineering matter?

2.3.1 Why does context engineering matter?

Let's explore some reasons why context engineering matters:

🎯 **Context shapes AI agent behavior**

Different context produces different results, so you must engineer it deliberately to get the behavior you want.

📦 **Context capacity is limited**

AI agents can only handle so much data at once. You need to choose what to include and what to skip. Poor choices mean either missing important information or filling up with useless data, both cause failures.

Complex tasks need constant updates

If you have tasks that follow multiple steps, you need to adapt the context as the AI agent operates. If you don't remove old information and add new relevant data, context accumulates, making the AI agent fail.

Clear context allows your AI agent to make better choices

When context is well-organized with clear descriptions, AI agents make correct decisions independently. Messy or unclear context causes agents to make mistakes.

You now understand what context engineering is and why it matters. Now let's get practical. Let's explore the context engineering best practices you can apply to calibrate your AI agent's context.

Continue to 2.4: Context engineering best practices

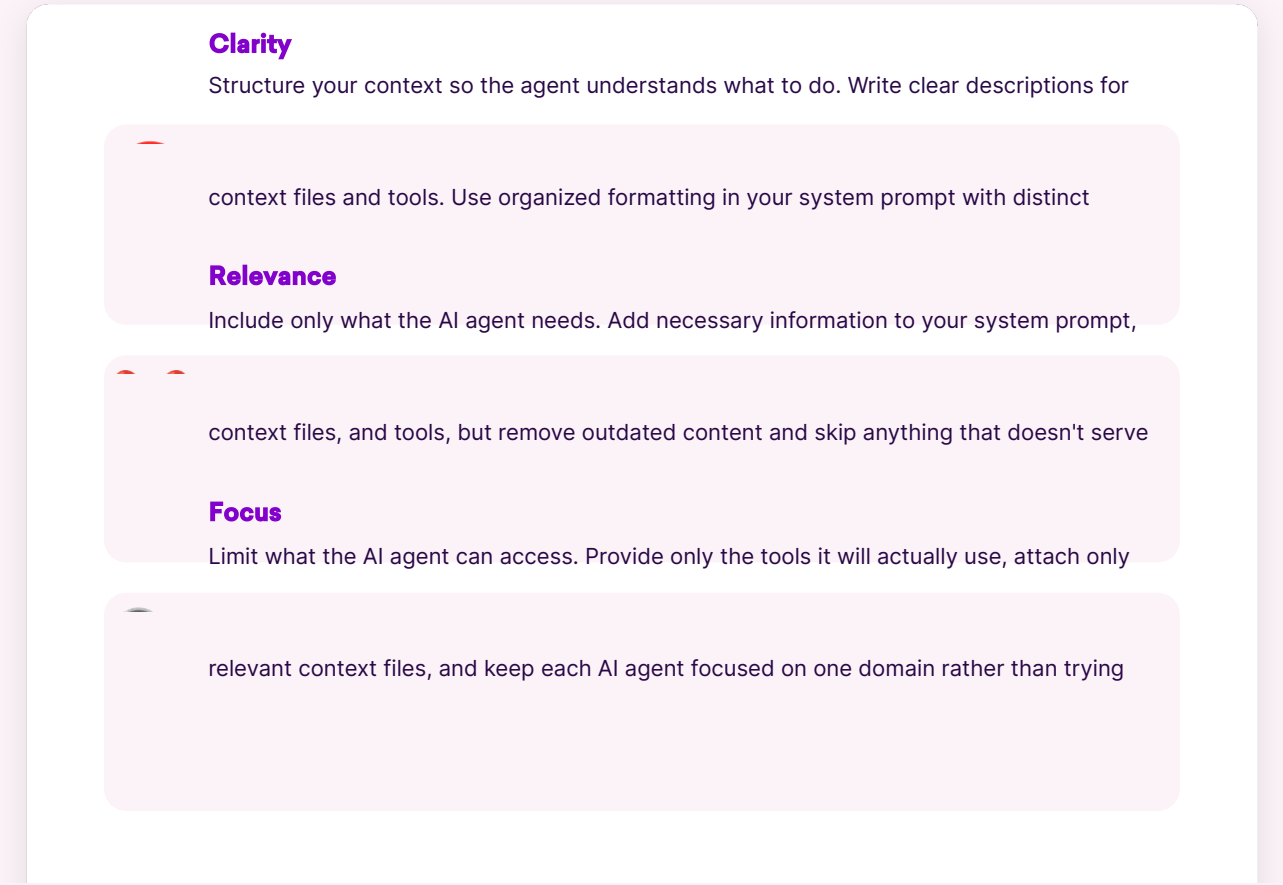
2.4 Context engineering best practices

Context engineering changes how you build AI agents. It's not just about writing better prompts, it's about managing all the information your AI agent processes within its limited space.

The practices you've learned, providing clear descriptions, using specialized tools, limiting tool access, and removing irrelevant information, all follow the same principle: **give your AI agent only the relevant information it needs, at the time it needs it**.

Context engineering best practices revolve around three core principles: **clarity**, **relevance**, and **focus**.

Here's how each one impacts AI agents:



Here's a list of some **best practices you can apply when context engineering your AI agent**:

1. Write clear descriptions for every context element
2. Break complex tasks into specialized tools
3. Only give tools the AI agent actually needs
4. Remove old or irrelevant information

Let's discover each of them in more detail.

1: Write clear descriptions for every context element

1

✏️ Write clear descriptions for every context element

The way you describe your context elements directly impacts your AI agent's ability to use them correctly because they will choose them based on their descriptions.

In unit 1, you learned the importance of writing clear prompts and tool definitions. This same principle applies to all context elements you provide to your AI agent. Without explicit descriptions, the AI agent may guess or ignore information entirely.

Context files are also a part of the AI agent's context and require careful description. When you attach a context file to your AI agent, you need to **describe its purpose and usage conditions** so the AI agent **knows when to consult it**.

A customer support AI agent has two context elements:

- **Context file: Refund Policy** with description: **Covers eligibility by purchase date (30-90 days) and customer type**.
- **Tool: Order Tracker** with description: **Retrieves order status and delivery info for a given order ID**.

When a customer asks **"My order #12345 is late. Can I get a refund?"**, the AI agent reads these descriptions, uses the **Order Tracker** to check the order status, then references the **Refund Policy** to determine eligibility, and answers based on both sources.

2: Break complex tasks into specialized tools

> **🔧Break complex tasks into specialized tools**
>
> Give your AI agent specific tools for specific jobs, not one tool that tries to do everything.

When your AI agent needs to handle multiple operations, **create a separate tool for each one**. Each tool should do one thing clearly, with a description that explains exactly what it does.

You need an AI agent to handle customer order issues. Instead of one vague "manage orders" tool, you create three specialized tools:

- **Order Status Checker:** retrieves tracking and delivery information.
- **Refund Processor:** initiates refunds up to $500.
- **Shipping Address Updater:** modifies delivery address for orders.

Each tool has one clear job, making the AI agent's decision straightforward instead of trying to figure out how to handle every order operation with one generic capability.

3: Only give tools the AI agent actually needs

3

🎯 Only give tools the AI agent actually needs

Don't give your AI agent access to tools just in case. Only connect tools it will actually use for its specific job.

Too many tools create confusion. When AI agents have access to significantly more tools than their task requires, they waste processing resources evaluating irrelevant options and increasing the probability of incorrect tool selection.

Giving just the right tools to your AI agent:

- **Reduces decision complexity:** fewer options mean faster, more accurate tool selections.
- **Prevents tool misuse:** eliminating irrelevant options means the AI agent can't choose the wrong tools.

You build an AI agent to schedule team meetings.

What not to do?
Give it access to multiple tools just in case, like product data, calendar, CRM, and analytics.

Why?
The AI agent wastes time evaluating which tool to use for a simple **"schedule a meeting at 2PM"** request.

What works better?
Give it **only calendar tools** (**check availability and create meetings**). When asked to schedule a meeting, the AI agent uses the one relevant tool immediately.

4: Remove old or irrelevant information

4

✂️ Remove old or irrelevant information

Context accumulates as the AI agent operates. You must actively remove what's no longer needed, or the AI agent will fail. This practice is called **context pruning**.

Without management, context eventually exceeds the window limit, causing failures. You need to actively remove completed tasks, summarize old conversation exchanges, and eliminate redundant information.

Context pruning keeps your AI agent's context clean and focused. By removing information that's no longer needed, you achieve **faster responses** and **better accuracy**. Without irrelevant information distracting it, the AI agent focuses on what actually matters for the current task.

Here's how to implement context pruning:

Start new sessions when switching tasks

Remove task-specific information, like conversation history and tool results, when switching tasks so irrelevant data doesn't waste context space or distract the agent from the new task.

An e-commerce customer support AI agent handles one customer's order issue from start to finish. Once that issue is resolved, the AI agent starts a new session for the next customer. This way, the previous customer's order details, conversation history, and resolution steps don't clutter the context when helping a new customer.

So how can you do this with Make?

Leave the Conversation ID empty or create a new one for every interaction.

A **Conversation ID** is a unique identifier for a conversation. All messages with the same Conversation ID share the same context. When you leave it empty, Make automatically generates a unique Conversation ID each time a new customer starts a conversation. Alternatively, you can create a new Conversation ID yourself for each interaction.

Either approach ensures each conversation is independent, and the AI agent doesn't carry over context from previous customers.

So you have learned the best practices for context engineering your AI agents. Let's discover a real-life example that applies them.

Continue to 2.5: Real-life example

2.5 Real-life example

Here's an example of context engineering for a customer support AI agent.

The AI agent's goal is to resolve order fulfillment problems. It identifies issues like late deliveries or lost packages. Then it provides customers with clear next steps based on company policy, like processing a refund, starting a carrier investigation, or updating delivery status.

Now, let's build the AI agent.

First things first, as you learned in unit 1, you need to write the **system prompt**:

> ROLE
> You are a customer service agent specialized in order fulfillment issues. Your job is to resolve problems with orders, shipping, and delivery.
>
> TONE & STYLE
> Professional, helpful, solution-oriented.
>
> INSTRUCTIONS
> 1. When a customer reports an order problem, check the current status first.
> 2. Determine resolution options based on the order status.
> 3. Issue the refund if it is approved per policy guidelines.
> 4. Provide clear next steps to the customer based on policy requirements.
>
> CONSTRAINTS
> 1. Do not discuss products, inventory, or issues unrelated to order fulfillment.
> 2. Do not process refunds without confirming policy eligibility first.

Even with a good system prompt, the AI agent needs the appropriate context to perform its tasks reliably. Let's see how a well-engineered AI agent looks like.

📝Context files

- **Shipping & refund policy:** Current 2025 policy. Use for delivery issues & refund eligibility. 30-day window applies.
- **Order issue guide:** Step-by-step troubleshooting for common shipping problems. Last updated June 2025.
- ⚠️No past versions of the policies or additional documentation.

🛠️Tools

- **Order Tracker:** Check order status, carrier, tracking number, and estimated delivery date for a specific order ID.
- **Refund Processor:** Processes the refund for an order.
- **Shipping Support:** Create carrier investigation ticket for lost or delayed packages.
- ⚠️No just in case tools.

🗨️Conversation History

- Start a new session for each customer's order issue.
- Focus on the current customer only, no history from previous customers or resolved issues.
- ⚠ Once an issue is resolved, begin a new session for the next customer.

So what's the result? The well-engineered AI agent quickly responds with accurate, policy-compliant guidance. It uses the Order tracker tool to check the shipment status, references the current 30-day refund policy, and performs the task. The customer gets clear next steps without confusion, contradictions, or irrelevant information.

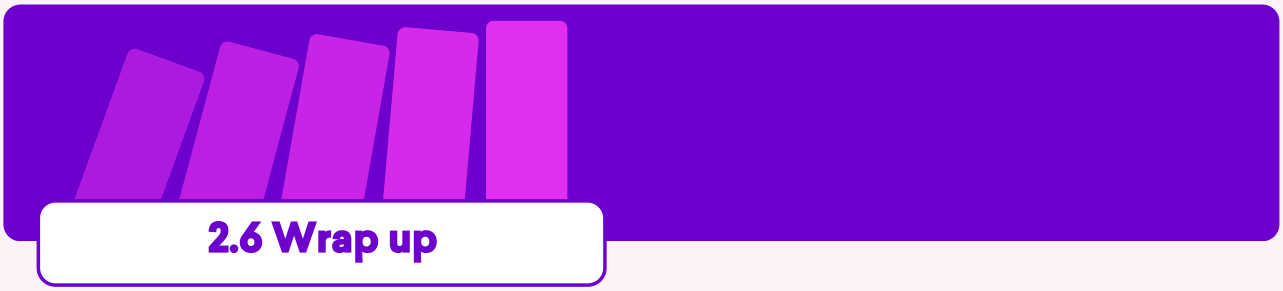
This means:

- **No poisoning:** only current, correct information included (no outdated or irrelevant data).

- **No confusion:** tools match exactly what the instructions specify (no overlapping or ambiguous options).
- **No distraction:** starts a new session for each customer (no conversation history from previous customers or resolved issues).
- **No clash:** single source of truth for policies (no contradictory information about refund timeframes or procedures).

Now that you know about context engineering, you can start applying its best practices to your AI agents.

Continue to complete this unit



2.6 Wrap up

1

The **context window** is the **maximum information your AI agent can process at once**. It fills automatically with every interaction until you actively manage it. Larger windows seem advantageous but actually introduce failures: **context poisoning**, **distraction**, **confusion**, and **clash**.

2

Context engineering is the practice of **selecting and managing the right information for your AI agent to guide its performance**. You make strategic decisions about what information to include, where to place it, how to organize it clearly, and when to remove outdated information.

3

Context engineering best practices follow three principles: **clarity**, **relevance**, and **focus**. Apply these by **providing clear descriptions for every context element**, **breaking complex tasks into specialized tools**, giving your AI agent **only the tools it will actually use**, and actively **removing old or irrelevant information** before it causes failures.

Great job! You completed the last unit of the course.

Now you know why context engineering is crucial when building AI agents. You are ready to apply what you've learned in your own AI agents!

make | academy

Mark this task complete to finish this course.