


# Generative AI Applications with AI Agents

# WORKSHOP DETAILS

- Instructor: Tanya Khanna
- Email: [tk759@scarletmail.rutgers.edu](mailto:tk759@scarletmail.rutgers.edu)
- Workshop Materials:
  - Github Link:  
<https://github.com/Tanya-Khanna/Data-Science-Workshop---Spring-2025---NBL->
  - Workshop Recordings: [Libguides](#) ● ● ●

# SCHEDULE

Introduction to Python Programming	February 3, 2025; 2 – 3:30 PM
Mastering Data Analysis: Pandas and Numpy	February 10, 2025; 2 – 3:30 PM
Introduction to Tableau: Visualizing Data Made Easy	February 17, 2025; 2 – 3:30 PM
Introduction to Machine Learning: Supervised Learning	February 24, 2025; 2 – 3:30 PM
Introduction to Machine Learning: Unsupervised Learning	March 3, 2025; 2 – 3:30 PM
Data-Driven Decision Making:  A/B Testing and Statistical Hypothesis Testing	March 10, 2025; 2 – 3:30 PM
Demystifying Generative AI	March 24, 2025; 2 – 3:30 PM
Large Language Models: From Theory to Implementation	March 31, 2025; 2 – 3:30 PM
Generative AI Applications with AI Agents	April 7, 2025; 2 – 3:30 PM
Building Intelligent Recommendation Systems	April 14, 2025; 2 – 3:30 PM

[Spring 2025 workshop calendar](#)

# TABLE OF CONTENTS

**01**

Generative AI vs Agentic AI

**02**

AI agents: Architecture, Working

**03**

Use Cases of AI agents

**04**

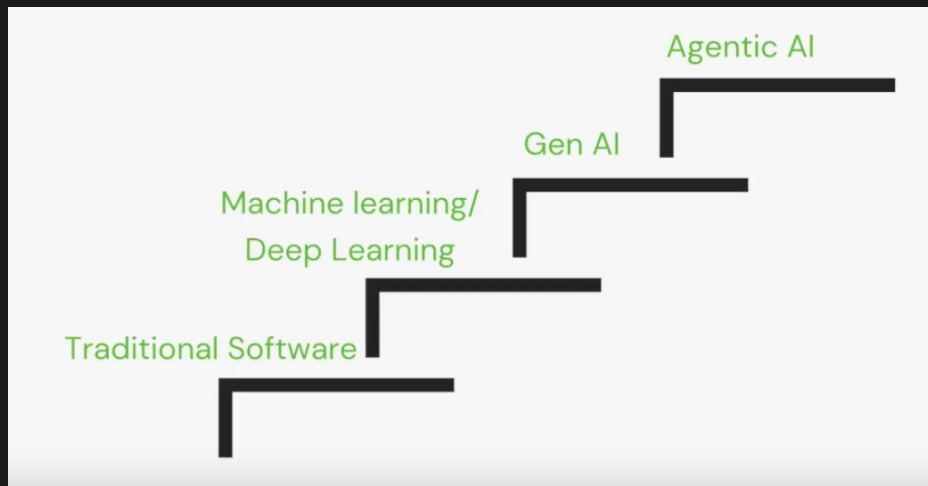
Best Practices for Designing AI Agents

**05**

Practical Session: AI Research Assistant

# Generative AI vs. Agentic AI

- LAST YEAR: Generative AI, Generative AI, Generative AI,
- THIS YEAR: Agentic AI, Agentic AI, Agentic AI



This shift from *Generative AI* to *Agentic AI* is part of a broader wave of innovation in the tech world.

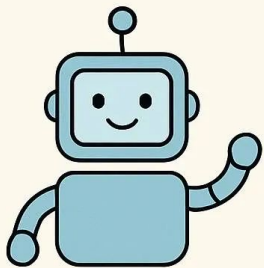
Started with traditional software—rule-based systems, hard coded logic. These systems could only do what we explicitly programmed them to do.

Then, in the late 2010s → Machine Learning and Deep Learning. These models could learn from data, recognize patterns, and make predictions. It was a huge leap from static logic to statistical learning.

Just in the last 3–4 years, we've seen a dramatic rise in Generative AI—models like GPT, DALL·E, and others that can create new content: text, images, even code. Generative AI made machines creative. Now in 2024 and beyond, the conversation is shifting again—this time to **Agentic AI**.

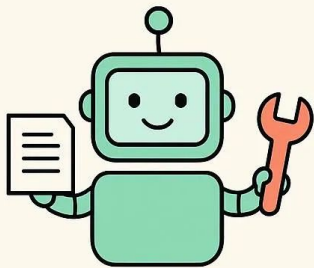
# Generative AI vs. Agentic AI

## Generative AI



Understands  
and generates  
content

## Agentic AI



Understands,  
generates,  
and performs  
actions

Generative AI models can understand and generate text, images, videos, and more, but they can't complete tasks end-to-end.

By **2024**, people wanted more. Talking was cool — but what if the AI could actually do things?

Sure, AI can suggest a travel itinerary — but can it actually book the tickets? Can it remember preferences and personalize things?

It can tell employees how many leaves they have — but can it apply for leave on their behalf?

The real question is: *can AI plan and execute tasks end-to-end using existing APIs and software instead of just giving answers?*

# Enter AI Agents!

This is where we move from passive AI to **active, autonomous AI**. An AI Agent is more than just a chatbot. It can **PERCEIVE** the environment (input from user, web, tools), **REASON** (make decisions, plan steps), **ACT** (use tools, run code, interact with APIs).

## Key Components of an AI Agent:

### 🎯 Goal:

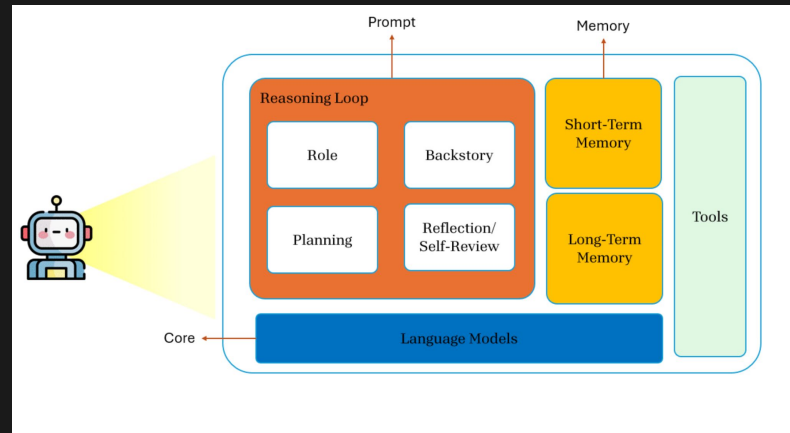
A high-level task you'd give to a smart assistant — like:

“Book me a vacation under \$1,000”

“Email everyone on this list and schedule a call”

These aren't just questions. They're *missions* — and agents are built to carry them out.

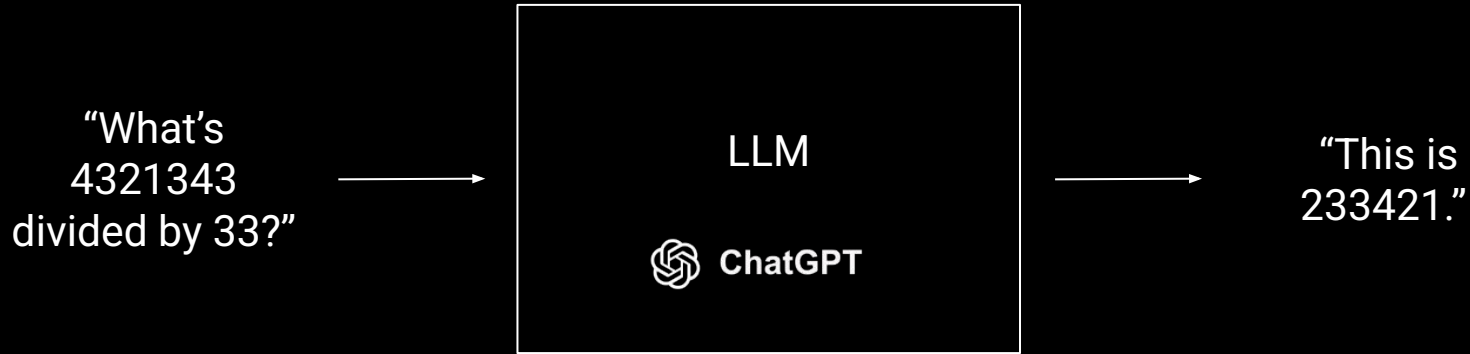
*Under the hood* of an AI agent:



- 🧠 **Core (LLM):** The brain. It understands language, reasons through the request, and makes decisions.
- 🔄 **Reasoning Loop:** The decision-maker. It figures out *what role to play*, *what to do next*, and *how to reflect* if things go wrong.
- 🧠 **Memory:** *Short-Term:* Remembers the current conversation; *Long-Term:* Remembers *you* — your preferences, past interactions, and context.
- 🛠️ **Tools:** These are the hands. The agent uses them to *act* — like checking your calendar, calling APIs, or calculating results. Without tools, an agent just talks. With tools, it acts.
- 📝 **Prompt:** This is your ask — the spark that sets everything in motion. Together, these parts let agents go from *just talking* to *actually doing*.

## A large language model (LLM)

- “ZIP file” of the entire internet - trained to understand and generate human-like text.
- Text input, text output - the LLM is the language brain—it's what helps the agent understand and talk to us. But on its own, it doesn't act, plan, or use tools. The agent wraps around the LLM and gives it the power to act. So while the LLM is the language brain, the agent is the whole body—with memory, reasoning, decision-making, and hands (aka tools).





One of its biggest limitations of LLM: **hallucinations**. This "zip file" is built through lossy compression—it doesn't remember everything exactly. It learns patterns from massive amounts of text, but it doesn't have a calculator inside it. It tries to guess the answer—not calculate it.

✅ It sounds correct; But it's completely wrong ❌. This is what we call a hallucination—when the model gives a response that sounds plausible, but is factually incorrect or made-up. This is why tools are important. To overcome this, we don't just want LLMs that talk—we want agents that act, using calculators, search engines, APIs, and more to ground their answers in reality. And that's where we're headed next: *How do we go from talking to doing?*

"What's  
4321343  
divided by 33?"



~~"This is  
233421."~~

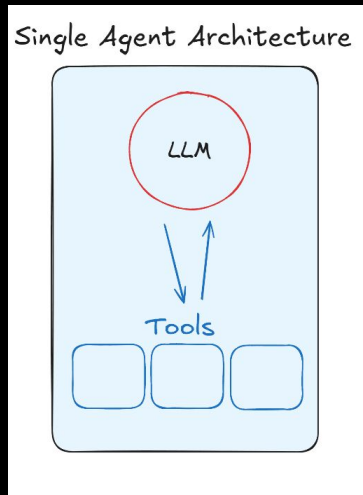
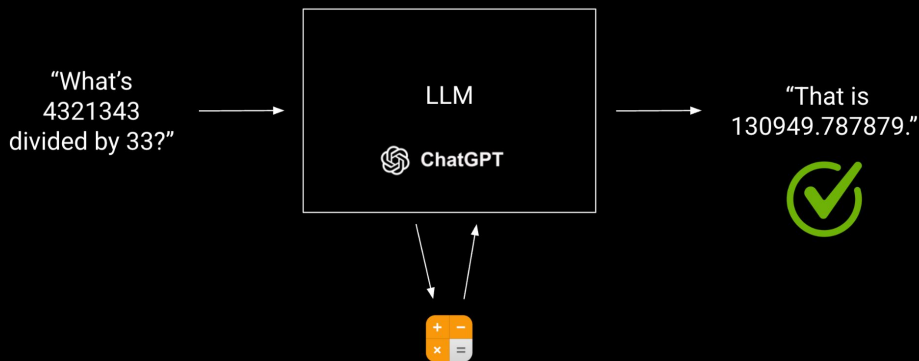
This is a so-called  
hallucination

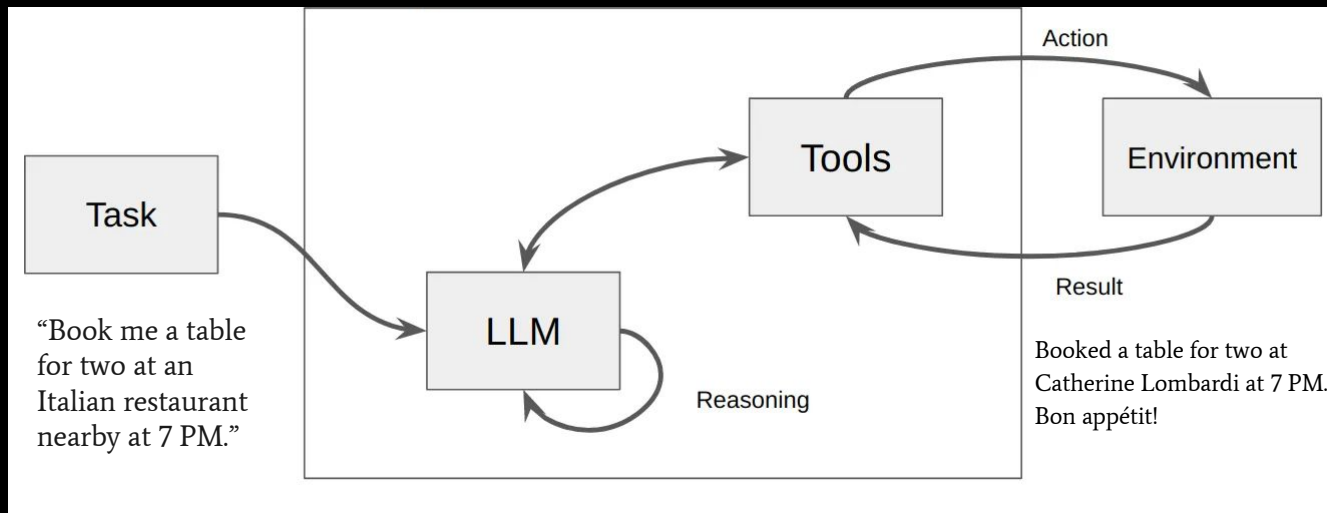
So how do we fix hallucinations? We give the model tools. In this case, the LLM knows it might not be accurate with math. So instead of guessing, it can call a calculator tool to get the correct answer: This is the idea behind tool use — the model doesn't just generate text, it decides when to **delegate**.

It says: *"I'm not sure — better use a calculator for this."*

This is the first major building block of an AI agent: An LLM augmented with tools. And this can scale from basic things like math or weather, to:

Automating customer service, Filing documents, Running code, Booking your next trip. So while a chatbot just talks, 👉 an agent talks, thinks, and acts.





**Task:** The user gives a goal in natural language.


**LLM:** Understands the meaning and intent of the user's request. Extracts relevant entities like number of people = 2, cuisine = Italian, time = 7 PM, location = inferred (Maybe from context, previous memory, or user profile). Decides **what tools** to use to fulfill the task.

Starts a **reasoning loop** : e.g., *"I need to find a suitable restaurant first, then check availability, then book."*

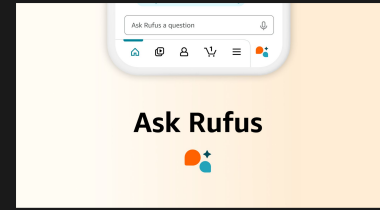
**Tools:** Tools are APIs or software services the LLM can call. A **restaurant search API** (e.g., Yelp, Google Places API); A **booking API** (e.g., OpenTable API or Resy API)

The LLM formulates queries for these tools: *"Search for Italian restaurants in New Brunswick with a 7 PM slot available tonight."*

**Environment:** The **is where the actions take effect** — it includes real-world systems, services, or APIs. Once the tool identifies a restaurant (say Catherine Lombardi), it makes a **real-time call** to: Check availability; Send a reservation request with details like name, time, party size. The booking **actually happens** through. The environment responds with: *"Reservation confirmed for 2 at Catherine Lombardi at 7 PM"*. The agent receives this result.

 **Back to LLM (Reasoning + Confirmation)** LLM reviews the result. Optionally stores the result in memory. Decides next steps: Confirm with user? Offer directions? Add to calendar? The agent responds naturally.

# Use Cases



## 1. Amazon's AI Shopping Assistant – Rufus (Helping customers shop smarter)

Rufus is an AI agent that can answer product questions, compare features, and suggest items across Amazon's massive catalog. Instead of you searching and filtering manually, Rufus understands your need and finds the right product for you—like a personal shopping assistant. It doesn't just chat—it acts: asking questions, fetching filtered product lists, and even remembering your preferences. Impact: Predicted to boost profits by \$700M by 2025.

## 2. Mastercard's Scam Prevention Agent (Blocking fraud before it happens)

Mastercard uses agentic AI to monitor your transactions in real time. If something seems suspicious (like a sudden international purchase), the agent doesn't wait—it takes action: flagging or even stopping the transaction automatically. It doesn't need to ask a human—this agent thinks and acts on its own.

## 3. eBay's Code-Writing and Marketing Agent (Helping developers and marketers work faster)

eBay uses agentic AI to write code and generate marketing content automatically. Say a team wants to launch a campaign—an AI agent can write the email, adjust the tone, and even suggest a headline—without a human copywriter. Developers also get agents that auto-generate code snippets based on prompts.

## 4. Deutsche Telekom's Internal Help Agent (Helping employees with HR tasks)

Employees at Deutsche Telekom can talk to an internal agent for HR help—like checking leave balances, filing time-off requests, or updating information. The agent doesn't just answer; it fills forms, triggers internal workflows, and sends confirmations.

# Best Practices for Designing AI Agents

*1. Manage Tool Integration:* It's important to define each tool clearly: what inputs it needs, what it's allowed to do, and what kind of output it returns. Start small with just a few well-tested tools, and build in error handling in case a tool doesn't work as expected. Keep an eye on which tools your agent uses the most and how they're performing so you can improve them over time.

*2. Manage Model Reasoning & Decision-Making:* Unlike traditional programs that follow strict rules, AI agents reason probabilistically. That means their decisions can sometimes be unpredictable. To help them reason better, use structured prompting methods like ReAct, which guide the agent to think step-by-step. Also, set up guardrails and checkpoints to validate the agent's decisions before it takes action. Adjust the "temperature" setting of the model (a number between 0 and 1) to control how creative or cautious the agent is—lower values like 0.2 make it more focused and predictable.

*3. Handle Multi-Step Processes & Context:* Real-world tasks are rarely one-and-done. Agents need to remember what they've done so far, keep track of the goal, and manage different steps along the way. For example, an agent helping with a loan application might need to pull a credit report, validate the result, and then update a CRM system. To do this well, agents need proper memory and "state management" to track progress. Also, always plan for the unexpected—build in error handling for every step, and design backup paths in case something fails. Logging each step helps you debug and improve your workflows.

*4. Control Hallucinations & Improve Accuracy:* Validate responses using structured formats like JSON. Use grounding, citations, and confidence scoring. Escalate uncertain outputs for human review. Continuously test for inaccurate outputs using test suites.

*5. Ensure Performance at Scale:* To keep performance smooth, use caching to avoid repeating the same expensive tasks. Set up queue systems to manage how many requests go through at once. Use monitoring tools (like LLM Ops dashboards) to track things like tool failures, slowdowns, or weird outputs. This helps you find problems before they impact users.