AI Agents 101: Quick Reference

AI Agents 101: Quick Reference

The views expressed are the author's own and do not necessarily represent the views of the U.S. Census Bureau or the U.S. Department of Commerce.

Course materials: github.com/brockwebb/ai-demos/ai_agents_101

Core Vocabulary

Term	What It Is	Key Question
Workflow	Structure, container, sequence of steps	What's the process?
Agent	Entity that does work	What's doing the work?
Agency	Granted decision-making authority	What decisions can it make?
Agentic	Behavior where agency is exercised	How much can it adapt?
Tool	Single discrete operation	What can it do?

The Loop — the heartbeat of agent behavior:

```
Observe → Decide → Act → Check → (repeat)
```

- **Observe:** Take in information about current state
- **Decide:** Determine what to do based on goal and state
- **Act:** Execute the chosen action (often using a tool)
- **Check:** Evaluate the result. Done? Continue? Stuck?

Six Design Principles

#	Principle	One-Liner
1	Good judgment upfront	Design quality bounds output quality
2	Autonomy is governance	Less autonomy is often better

#	Principle	One-Liner
3	Most problems don't need agents	Simple solutions beat complex ones
4	Specification is the skill	Clarity beats capability
5	Design for uncertainty	Plan for failure, not just success
6	Digestible chunks	Focused beats sprawling

The meta-principle: AI amplifies your process. Good process + AI = faster good outcomes. Bad process + AI = faster bad outcomes.

The stakes: Microsoft's AI Red Team documented failure modes in agentic systems—misalignment, cascading failures, unauthorized actions, denial of service. Nearly every failure traces back to ignoring these principles. They're not caution; they're how you build systems that work.

Template: Describing an Agent

When specifying what an agent should do, fill in these blanks:

> As a [**role**], I want the agent to [**narrow outcome**] so that [**business value**].
>
> The agent is allowed to see: [**systems/data types**]
>
> The agent is allowed to do: [**actions, e.g., propose flags, draft responses**]
>
> The agent must never: [**forbidden actions**]
>
> If the agent is unsure or the decision is high stakes, it must: [**who to ask / how to escalate**]

If you can't fill in every blank, you don't have a clear enough design yet.

Checklist: Reviewing Agent Behavior

When evaluating whether an agent is working correctly, ask:

- ☐ Did it stay within its job description and data boundaries?
- ☐ Is the action correct in context?
- ☐ Is the rationale understandable?
- ☐ Would I sign my name under this action?

If any answer is "no," mark it as a bad example. Collect these. They become your test cases.

Try It Yourself: Recipe Workflow Prompt

Copy this into Claude or another capable model. Watch it work. Then try breaking it.

```
You are a recipe assistant that helps users plan meals and create grocery lists.

When the user requests a dish:

1. SEARCH for highly-rated recipes (4+ stars, reputable sources like Serious Eats, NYT Cooking, Bon App
reviewed AllRecipes submissions). Prefer recipes with clear ingredient lists and reasonable prep times.

2. SELECT one recipe. Briefly explain why you chose it (rating, source credibility, matches the dish re

3. EXTRACT the full ingredient list. Standardize quantities where reasonable (e.g., "1 28-
oz can crushed tomatoes" not "one large can tomatoes").

4. CHECK against user's stated dietary restrictions or allergies. If there's a conflict:
   - STOP
   - Explain the conflict clearly
   - Ask if they want a substitution or a different recipe
   - Do not proceed until they respond

5. GENERATE a grocery list organized by store section (produce, meat, dairy, pantry, etc.). Include qua

6. ASK: "Anything on this list you already have? Give me the numbers and I'll remove them."

If the user provides items they have, regenerate the list without those items.

---

User dietary restrictions: [none stated yet - ask if unclear]

User request: I want to make chicken cacciatore
```

What to notice: - Decision points with criteria (step 2) - Hard stop with human checkpoint (step 4) - Iteration loop (step 6) - Transparency requirement ("explain why you chose it")

Try breaking it: - Request a dish with an ingredient you're allergic to - Ask for something obscure - Give a vague request like "something Italian"

Try different models: This prompt was tested across model tiers with dramatically different results. Sonnet-class models (Claude Sonnet 4, GPT-4o, Gemini Pro) followed the workflow correctly — waiting for input, using web search, following steps in order. Smaller models (Claude Haiku 3.5, GPT-4o-mini) broke spectacularly: ignoring the workflow entirely, searching academic databases for meatball recipes, or building a React app instead of following instructions.

The lesson: modern frontier models have built-in reasoning capabilities that make them natural "reasoning agents." They can infer intent, route to the right tools, and follow conditional logic. Smaller/older models lack this and need much more explicit instruction — or simply can't handle the inference load. **The same prompt, the same framework, wildly different results.** The agent lives in the interaction between prompt design and model capability.

The Bottom Line

You don't need expensive tools or deep technical skills to work with AI agents effectively. You need:

- Clear thinking
- Well-structured prompts
- Appropriate skepticism about what automation can and should do

Start simple. Add complexity only when justified. Design for uncertainty. Keep humans in the loop where it matters.

Further Reading

- Microsoft AI Red Team, "Taxonomy of Failure Modes in Agentic AI Systems" (2025)
- OpenAI, "A Practical Guide to Building Agents" (2025)
- Anthropic, "Building Effective Agents" (2024)

Course: AI Agents 101 — Bounded Agency Over Autonomous Agents