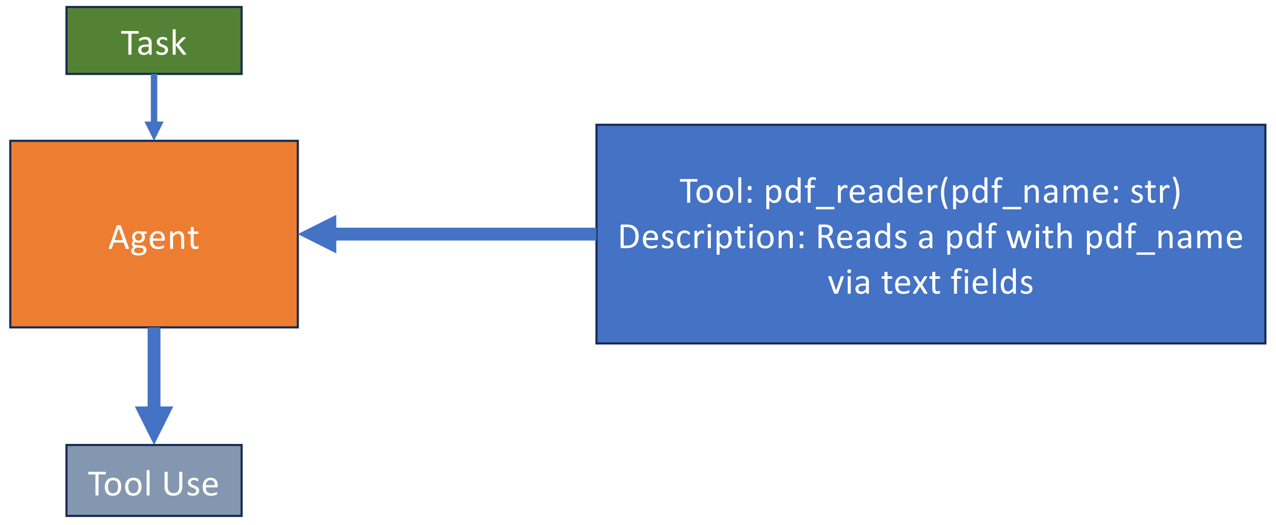
Agent Skills

Inspired by Claude

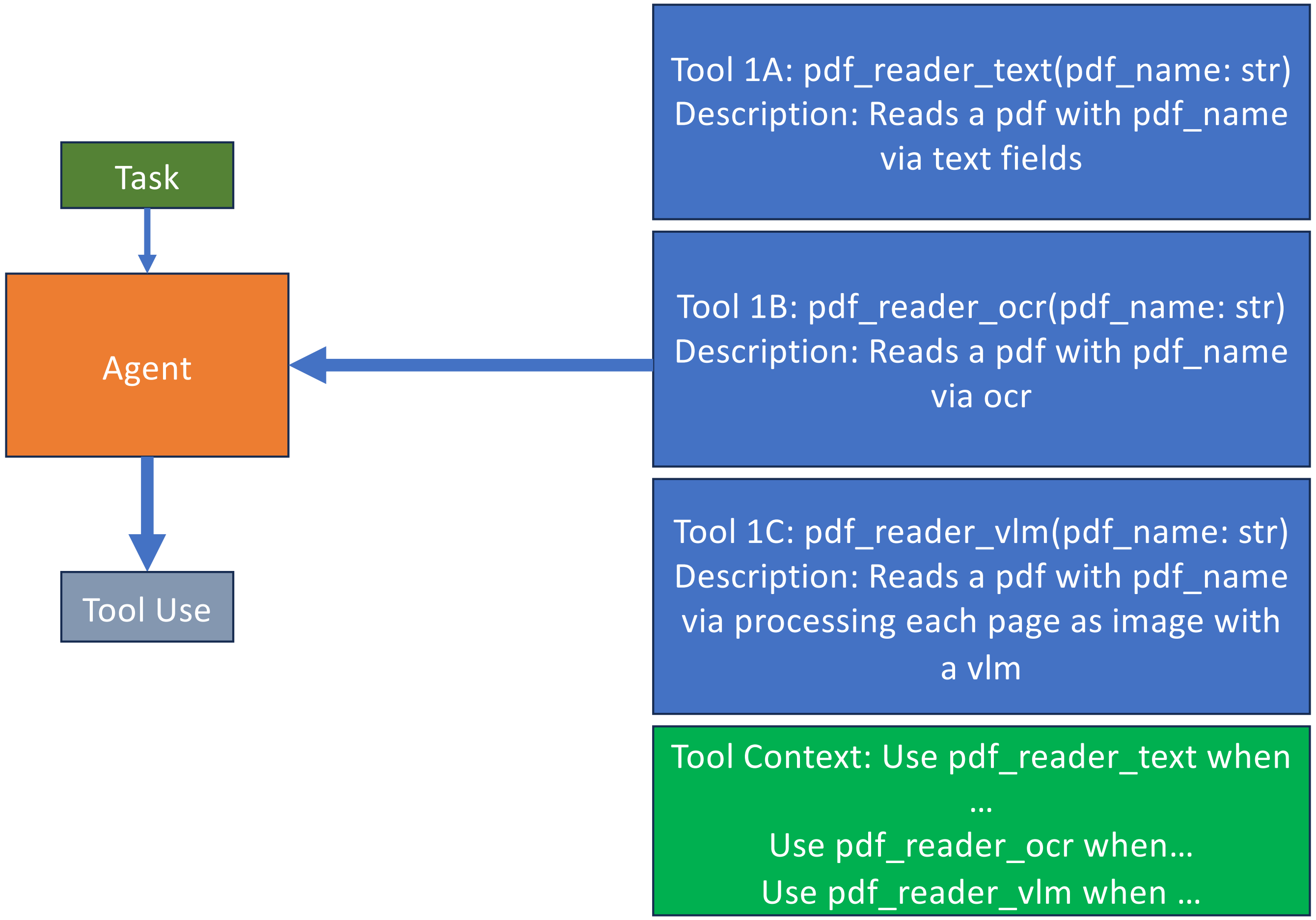
Presented by:

John Tan Chong Min

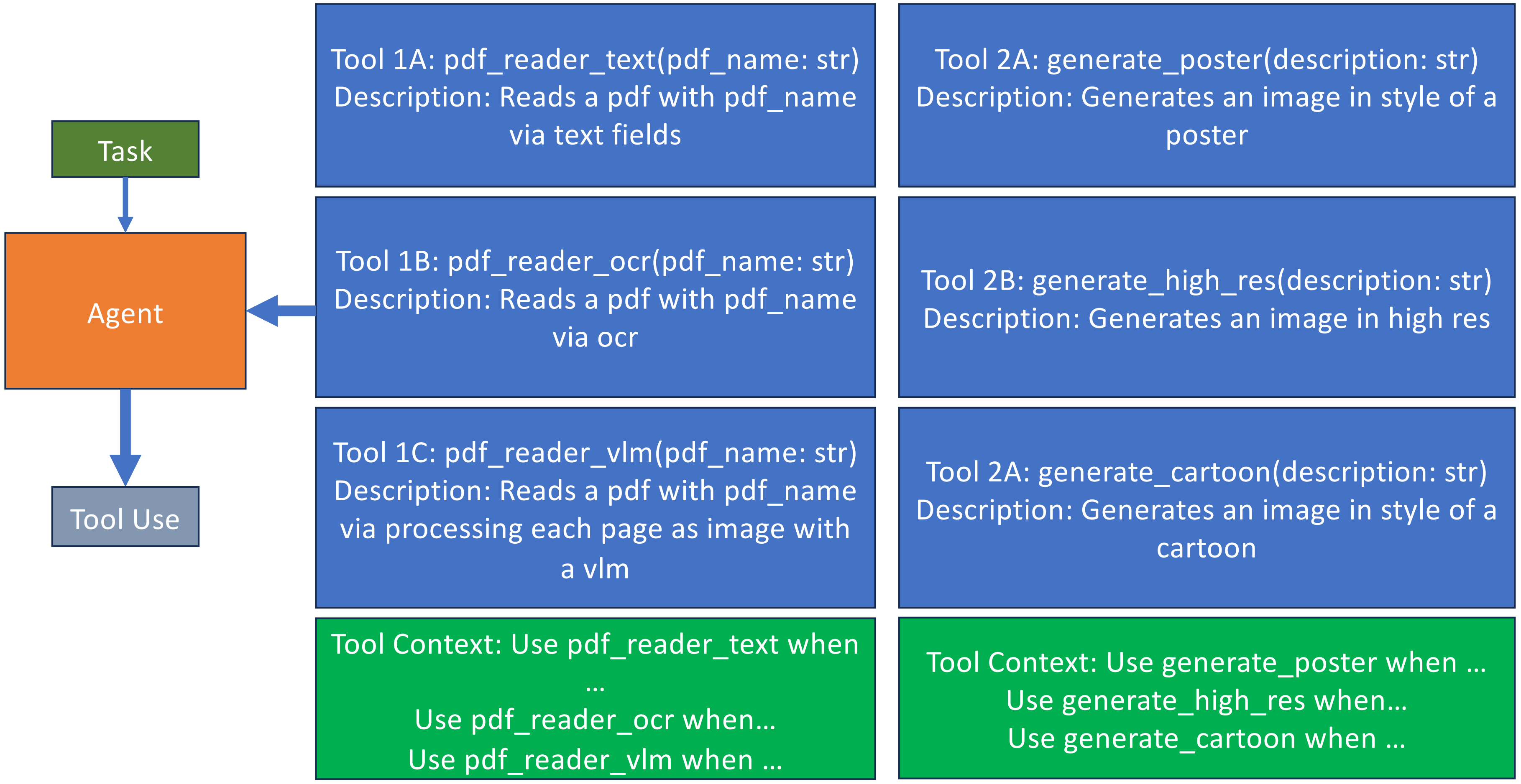
Traditional Agent + Tools (1/4)


Task
Agent
Tool Use
Tool: pdf_reader(pdf_name: str)
Description: Reads a pdf with pdf_name via text fields

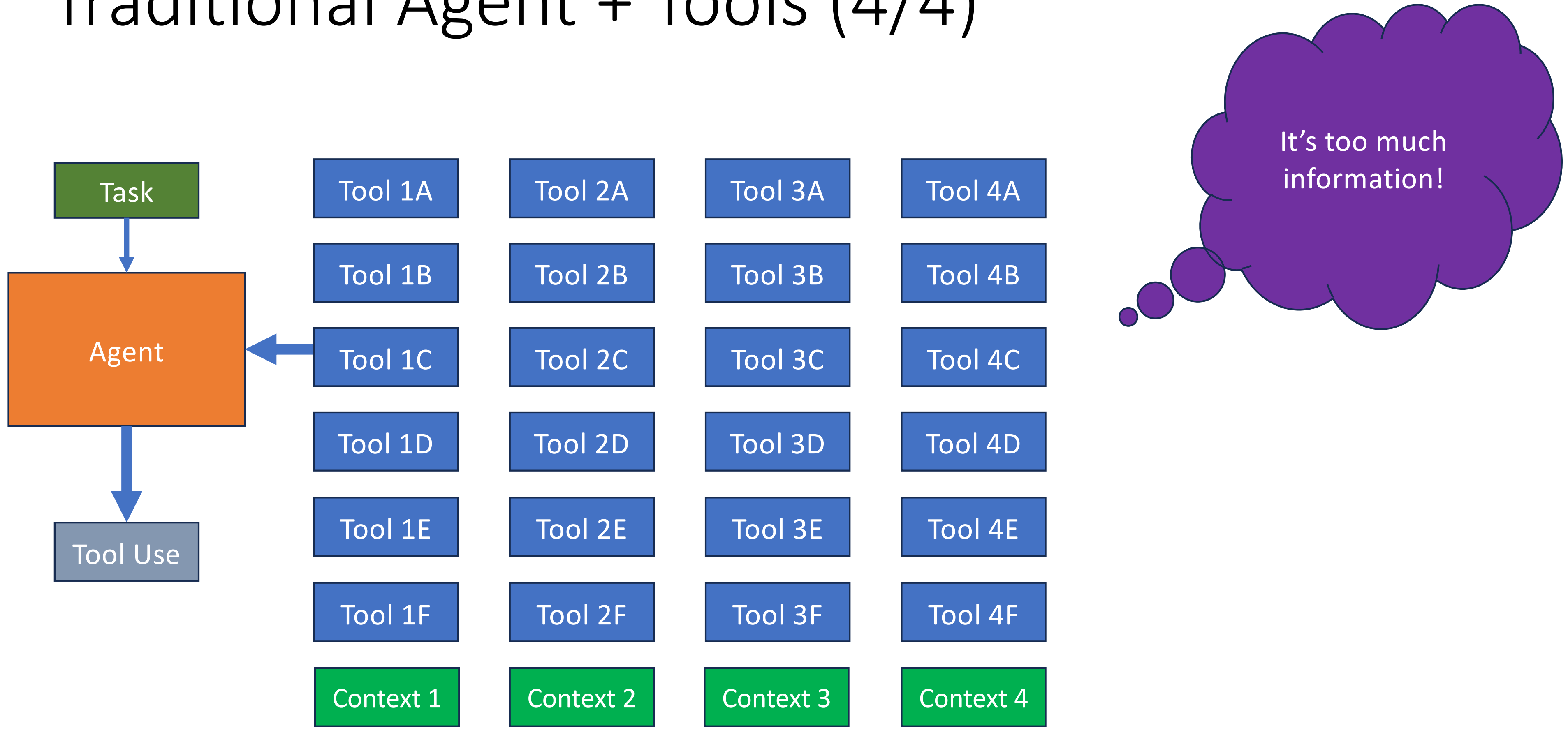
Traditional Agent + Tools (2/4)


Task
Agent
Tool Use
Tool 1A: pdf_reader_text(pdf_name: str)
Description: Reads a pdf with pdf_name via text fields
Tool 1B: pdf_reader_ocr(pdf_name: str)
Description: Reads a pdf with pdf_name via ocr
Tool 1C: pdf_reader_vlm(pdf_name: str)
Description: Reads a pdf with pdf_name via processing each page as image with a vlm
Tool Context: Use pdf_reader_text when …
Use pdf_reader_ocr when…
Use pdf_reader_vlm when …

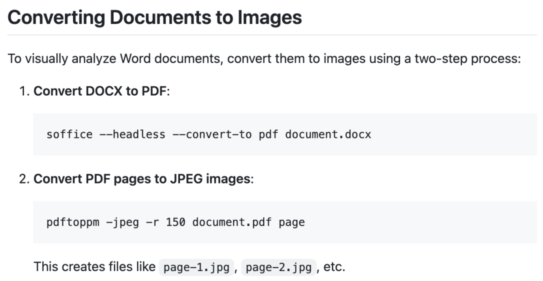
Traditional Agent + Tools (3/4)


Task
Agent
Tool Use
Tool 1A: pdf_reader_text(pdf_name: str)
Description: Reads a pdf with pdf_name via text fields
Tool 1B: pdf_reader_ocr(pdf_name: str)
Description: Reads a pdf with pdf_name via ocr
Tool 1C: pdf_reader_vlm(pdf_name: str)
Description: Reads a pdf with pdf_name via processing each page as image with a vlm
Tool Context: Use pdf_reader_text when …
Use pdf_reader_ocr when…
Use pdf_reader_vlm when …
Tool 2A: generate_poster(description: str)
Description: Generates an image in style of a poster
Tool 2B: generate_high_res(description: str)
Description: Generates an image in high res
Tool 2A: generate_cartoon(description: str)
Description: Generates an image in style of a cartoon
Tool Context: Use generate_poster when …
Use generate_high_res when…
Use generate_cartoon when …

Traditional Agent + Tools (4/4)



Code Agent + Context



Converting Documents to Images

To visually analyze Word documents, convert them to images using a two-step process:

1. **Convert DOCX to PDF:**

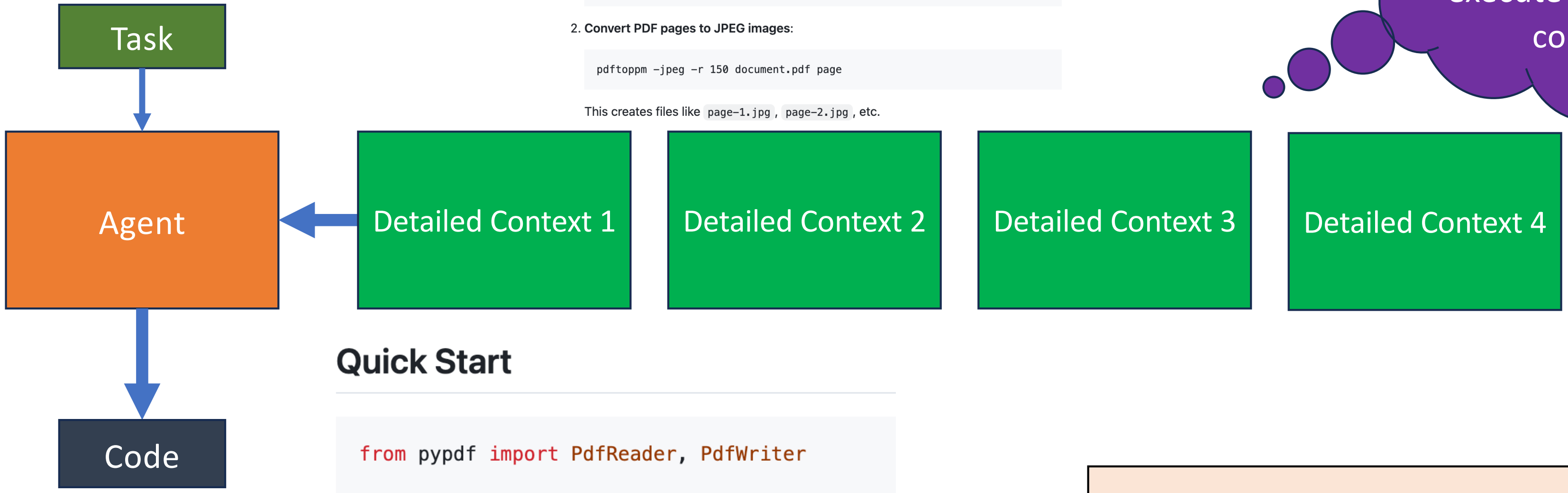
```
soffice --headless --convert-to pdf document.docx
```

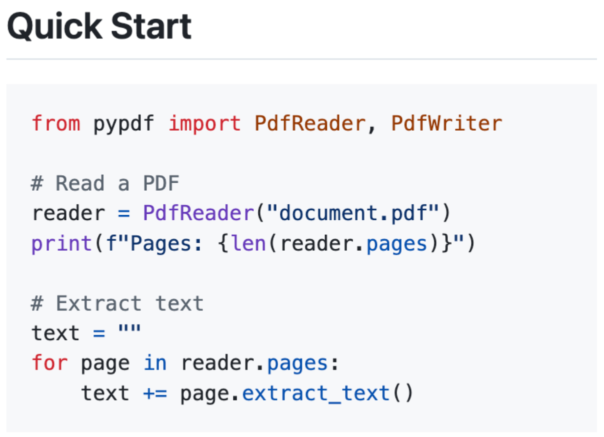
2. **Convert PDF pages to JPEG images:**

```
pdftoppm -jpeg -r 150 document.pdf page
```

This creates files like `page-1.jpg`, `page-2.jpg`, etc.

Maybe I can reduce the number of tools needed if I execute my own code





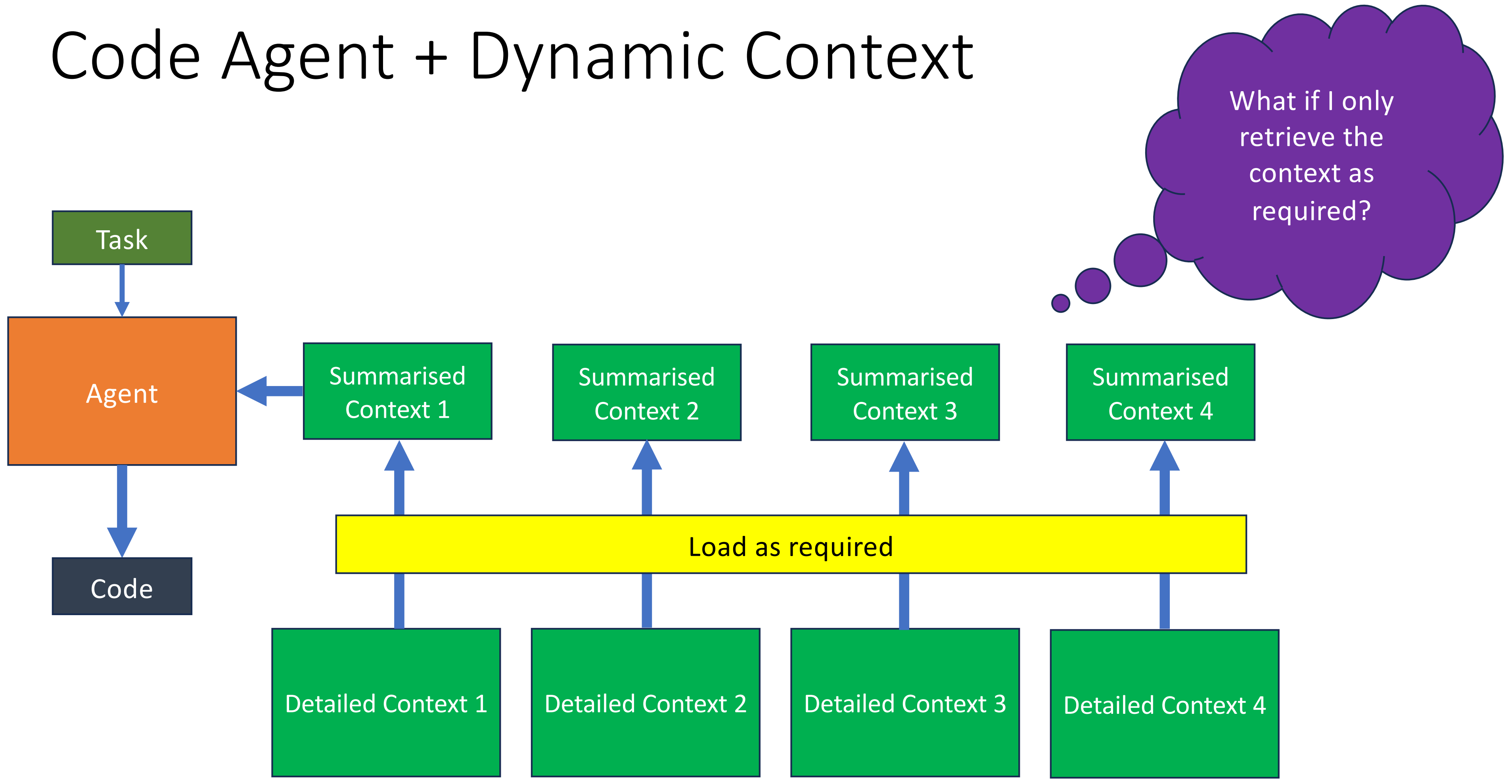
Quick Start

```
from pypdf import PdfReader, PdfWriter

# Read a PDF
reader = PdfReader("document.pdf")
print(f"Pages: {len(reader.pages)}")

# Extract text
text = ""
for page in reader.pages:
    text += page.extract_text()
```

Instead of having hard-coded tools, I can just get a context telling me the best practices, then I can generate my own code.

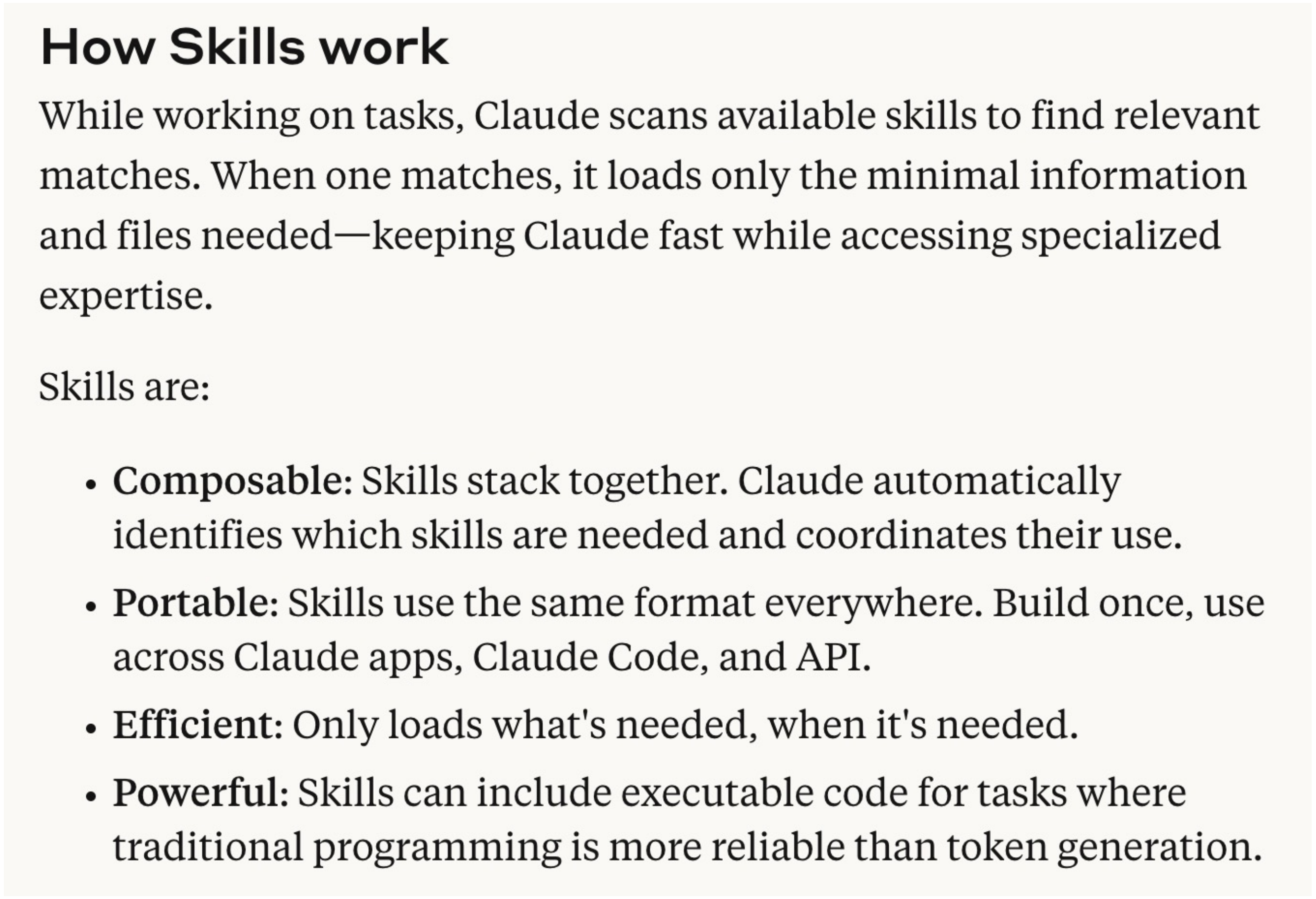
Very similar to **one-shot/few-shot** learning

Code Agent + Dynamic Context



How can we build LLM agents that can learn skills on the fly on a need-to-know basis?

Learning a new skill is like downloading a new packet of information



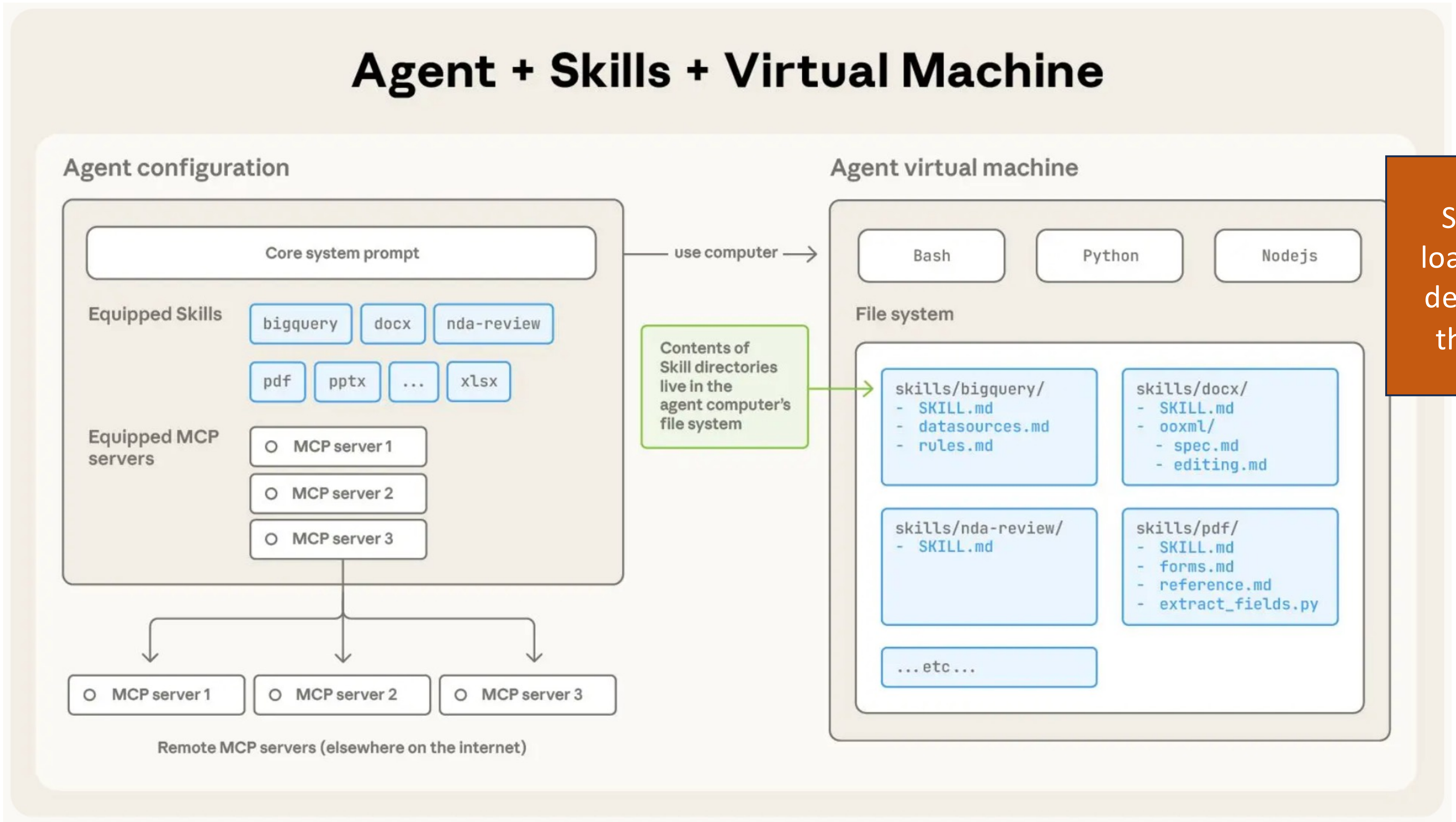
How Skills work

While working on tasks, Claude scans available skills to find relevant matches. When one matches, it loads only the minimal information and files needed—keeping Claude fast while accessing specialized expertise.

Skills are:

- **Composable:** Skills stack together. Claude automatically identifies which skills are needed and coordinates their use.
- **Portable:** Skills use the same format everywhere. Build once, use across Claude apps, Claude Code, and API.
- **Efficient:** Only loads what's needed, when it's needed.
- **Powerful:** Skills can include executable code for tasks where traditional programming is more reliable than token generation.

https://www.anthropic.com/news/skills



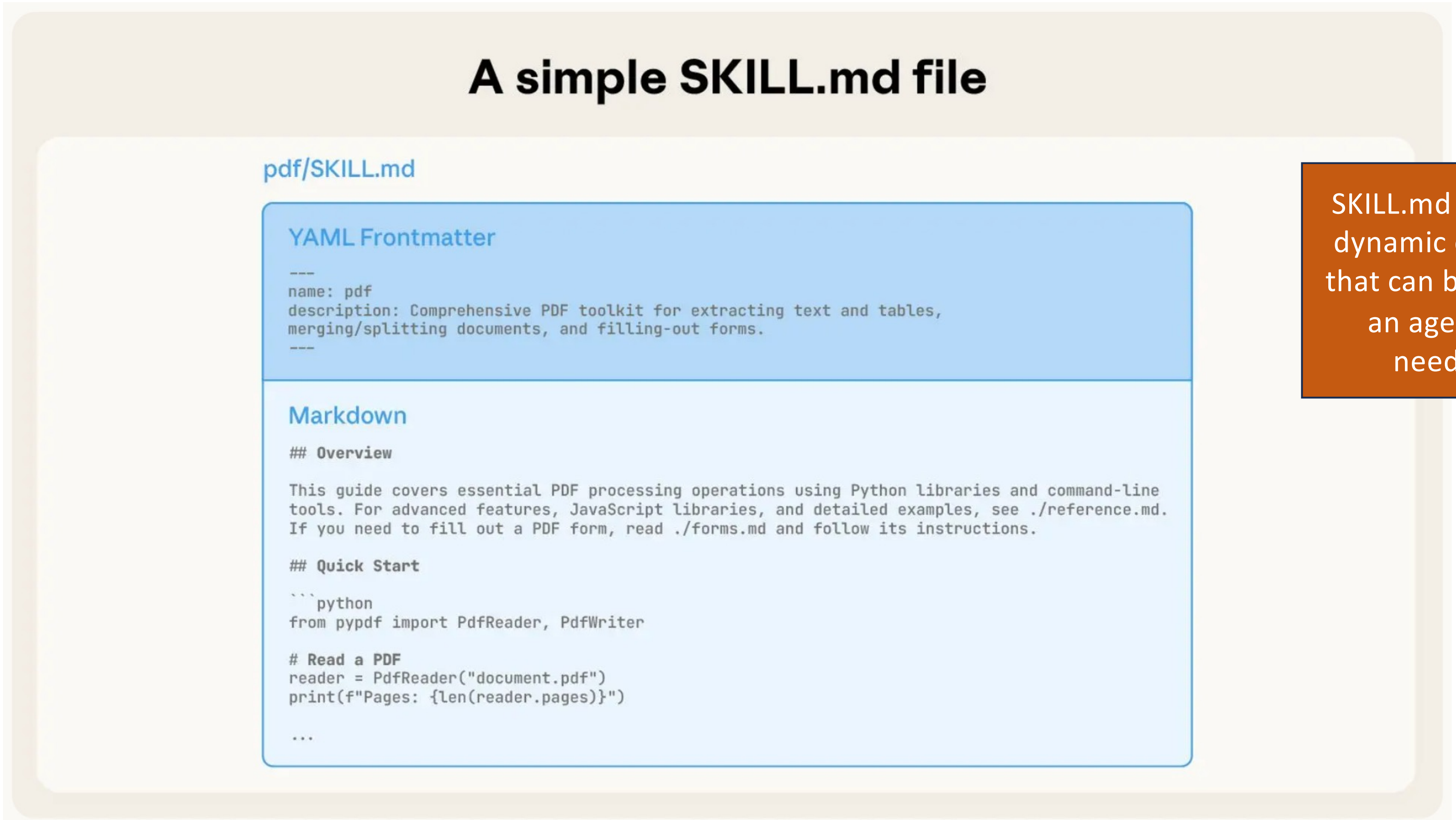
Skills are loadable on demand by the agent

A skill is a directory containing a SKILL.md file that contains organized folders of instructions, scripts, and resources that give agents additional capabilities.

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills

How does a skill work?

- **Name** + **description** of the skill known to agent at the start of runtime
- If the agent thinks a skill is useful, it will read the SKILL.md file (instructions on how to use the skill) to feed into context of the agent the details of the skill
- Agent can then use the skill’s instructions to write code and execute the skill
- **Summary: A skill is like a dynamic context given to an agent to know how to do specific tasks better**



A simple SKILL.md file

pdf/SKILL.md

YAML Frontmatter

```
---
name: pdf
description: Comprehensive PDF toolkit for extracting text and tables,
merging/splitting documents, and filling-out forms.
---
```

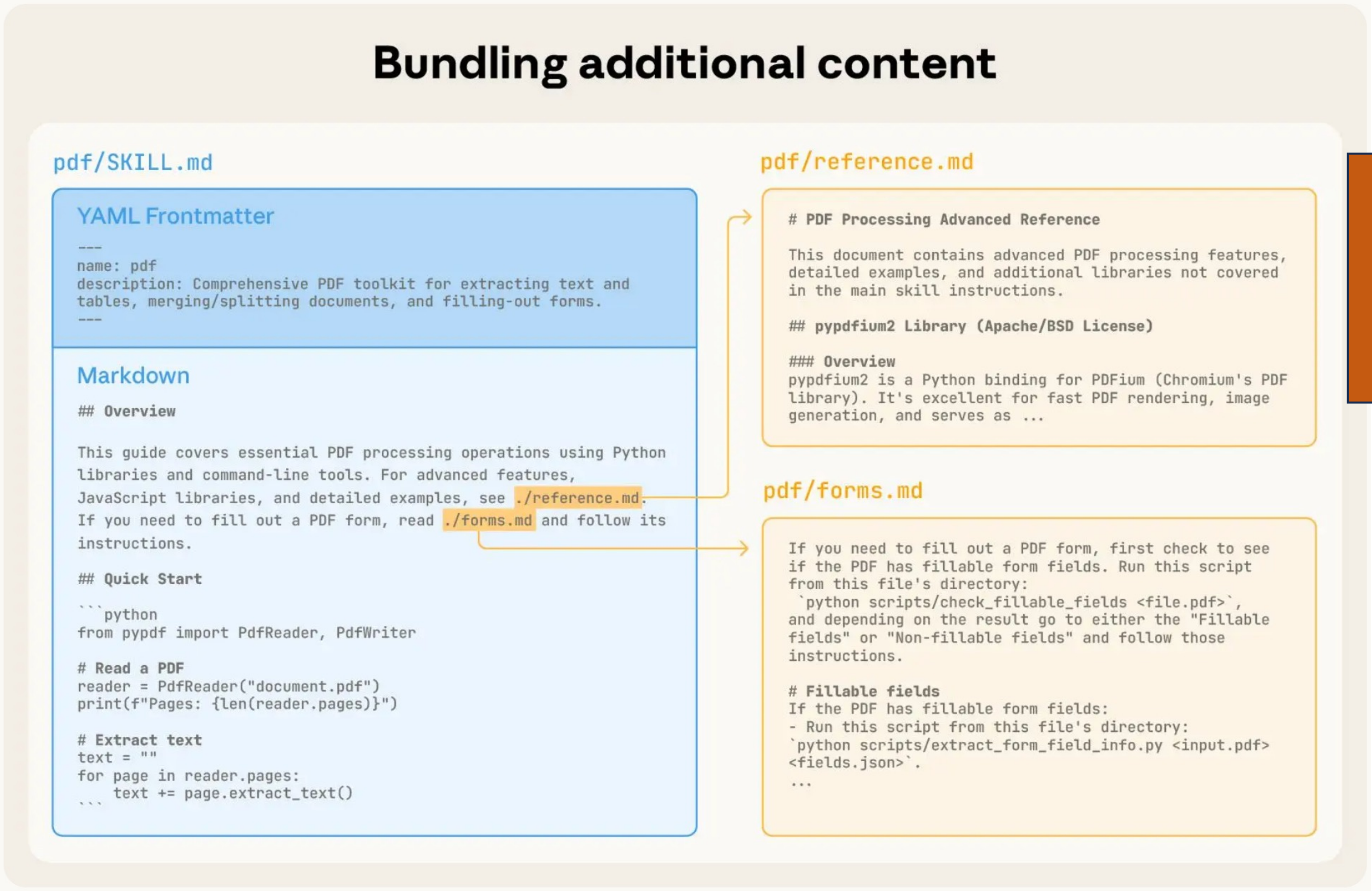
Markdown

~~~
## Overview

This guide covers essential PDF processing operations using Python libraries and command-line
tools. For advanced features, JavaScript libraries, and detailed examples, see ./reference.md.
If you need to fill out a PDF form, read ./forms.md and follow its instructions.

## Quick Start

```python
from pypdf import PdfReader, PdfWriter

# Read a PDF
reader = PdfReader("document.pdf")
print(f"Pages: {len(reader.pages)}")

...
~~~

A SKILL.md file must begin with YAML Frontmatter that contains a file name and description, which is loaded into its system prompt at startup.

SKILL.md is like a dynamic context that can be fed to an agent as needed

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills



Bundling additional content

pdf/SKILL.md

YAML Frontmatter

```
---
name: pdf
description: Comprehensive PDF toolkit for extracting text and
tables, merging/splitting documents, and filling-out forms.
---
```

Markdown

~~~
## Overview

This guide covers essential PDF processing operations using Python
libraries and command-line tools. For advanced features,
JavaScript libraries, and detailed examples, see ./reference.md.
If you need to fill out a PDF form, read ./forms.md and follow its
instructions.

## Quick Start

```python
from pypdf import PdfReader, PdfWriter

# Read a PDF
reader = PdfReader("document.pdf")
print(f"Pages: {len(reader.pages)}")

# Extract text
text = ""
for page in reader.pages:
    text += page.extract_text()
```
~~~

pdf/reference.md

```
# PDF Processing Advanced Reference

This document contains advanced PDF processing features,
detailed examples, and additional libraries not covered
in the main skill instructions.

## pypdfium2 Library (Apache/BSD License)

### Overview
pypdfium2 is a Python binding for PDFium (Chromium's PDF
library). It's excellent for fast PDF rendering, image
generation, and serves as ...
```

pdf/forms.md

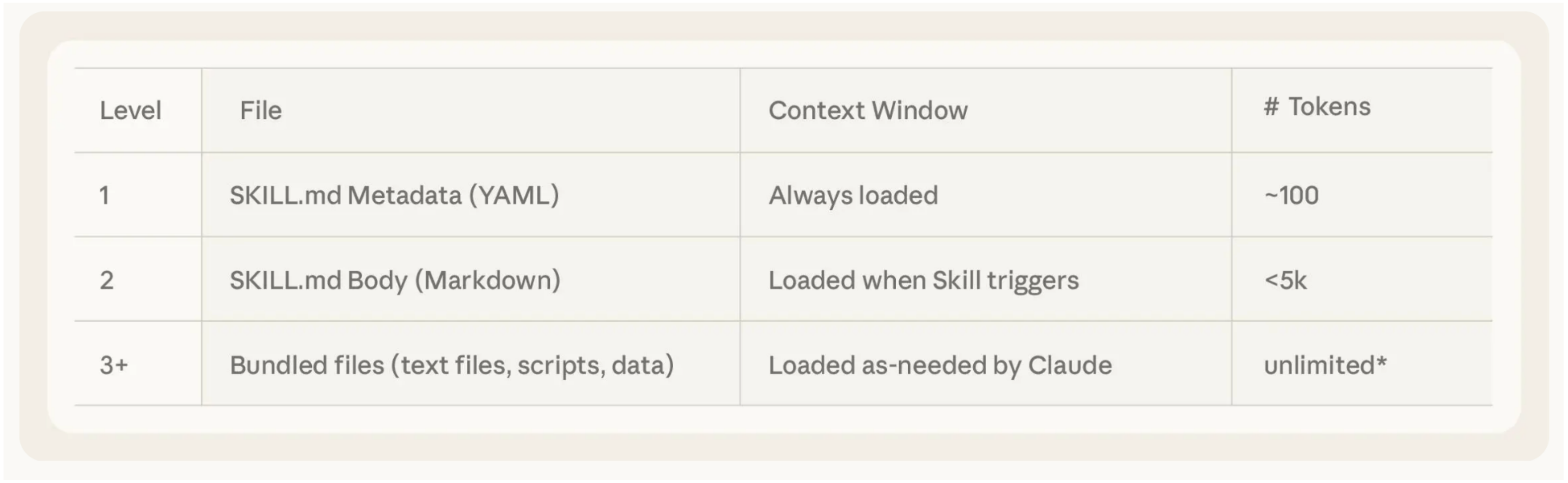
```
If you need to fill out a PDF form, first check to see
if the PDF has fillable form fields. Run this script
from this file's directory:
 `python scripts/check_fillable_fields <file.pdf>`,
and depending on the result go to either the "Fillable
fields" or "Non-fillable fields" and follow those
instructions.

# Fillable fields
If the PDF has fillable form fields:
- Run this script from this file's directory:
`python scripts/extract_form_field_info.py <input.pdf>
<fields.json>`.
...
```

You can incorporate more context (via additional files) into your skill that can then be triggered by Claude based on the system prompt.

Detailed reference guides can be linked in main SKILL.md

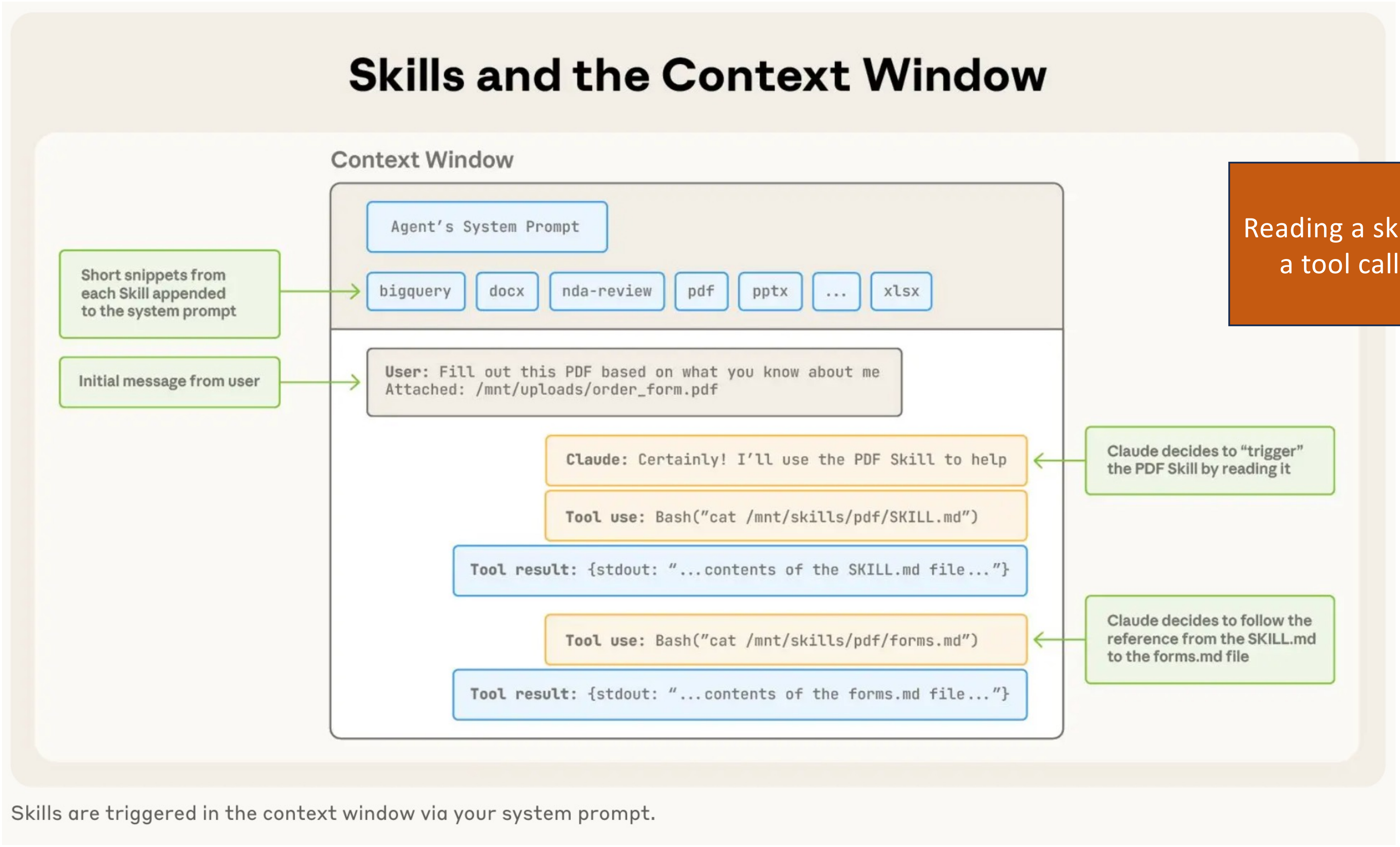
https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills

Different levels of context loading



Level	File	Context Window	# Tokens
1	SKILL.md Metadata (YAML)	Always loaded	~100
2	SKILL.md Body (Markdown)	Loaded when Skill triggers	<5k
3+	Bundled files (text files, scripts, data)	Loaded as-needed by Claude	unlimited*

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills



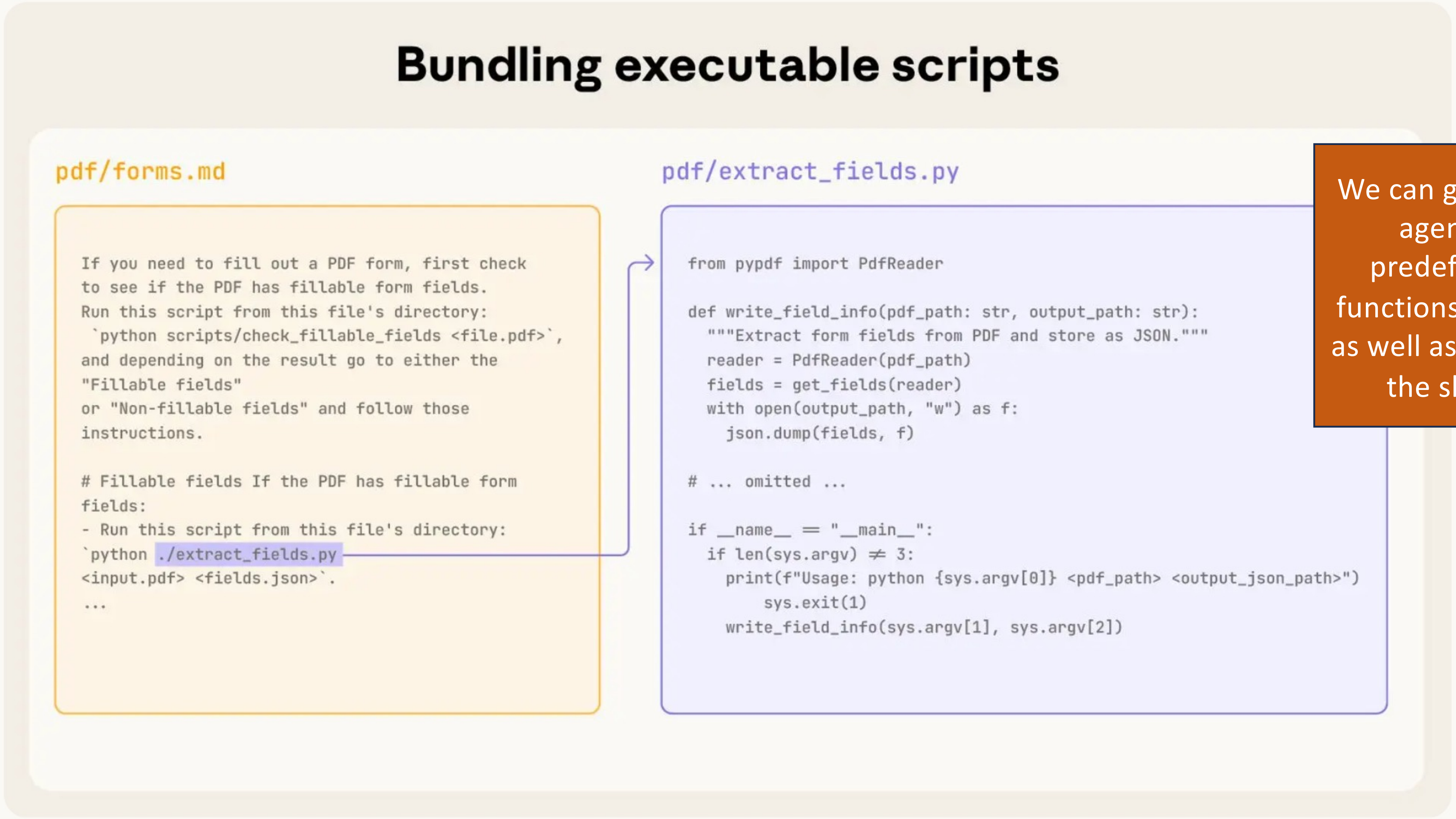
Skills and the Context Window
Context Window
Agent's System Prompt
bigquery
docx
nda-review
pdf
pptx
...
xlsx
Short snippets from each Skill appended to the system prompt
Initial message from user
Claude decides to "trigger" the PDF Skill by reading it
Claude decides to follow the reference from the SKILL.md to the forms.md file

```
User: Fill out this PDF based on what you know about me
Attached: /mnt/uploads/order_form.pdf
Claude: Certainly! I'll use the PDF Skill to help
Tool use: Bash("cat /mnt/skills/pdf/SKILL.md")
Tool result: {stdout: "...contents of the SKILL.md file..."}
Tool use: Bash("cat /mnt/skills/pdf/forms.md")
Tool result: {stdout: "...contents of the forms.md file..."}
```

Skills are triggered in the context window via your system prompt.

Reading a skill is a tool call!

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills



Bundling executable scripts

pdf/forms.md

```
If you need to fill out a PDF form, first check
to see if the PDF has fillable form fields.
Run this script from this file's directory:
 `python scripts/check_fillable_fields <file.pdf>`,
and depending on the result go to either the
"Fillable fields"
or "Non-fillable fields" and follow those
instructions.

# Fillable fields If the PDF has fillable form
fields:
- Run this script from this file's directory:
`python ./extract_fields.py
<input.pdf> <fields.json>`.
...
```

pdf/extract_fields.py

```python
from pypdf import PdfReader

def write_field_info(pdf_path: str, output_path: str):
  """Extract form fields from PDF and store as JSON."""
  reader = PdfReader(pdf_path)
  fields = get_fields(reader)
  with open(output_path, "w") as f:
    json.dump(fields, f)

# ... omitted ...

if __name__ == "__main__":
  if len(sys.argv) != 3:
    print(f"Usage: python {sys.argv[0]} <pdf_path> <output_json_path>")
        sys.exit(1)
    write_field_info(sys.argv[1], sys.argv[2])
```

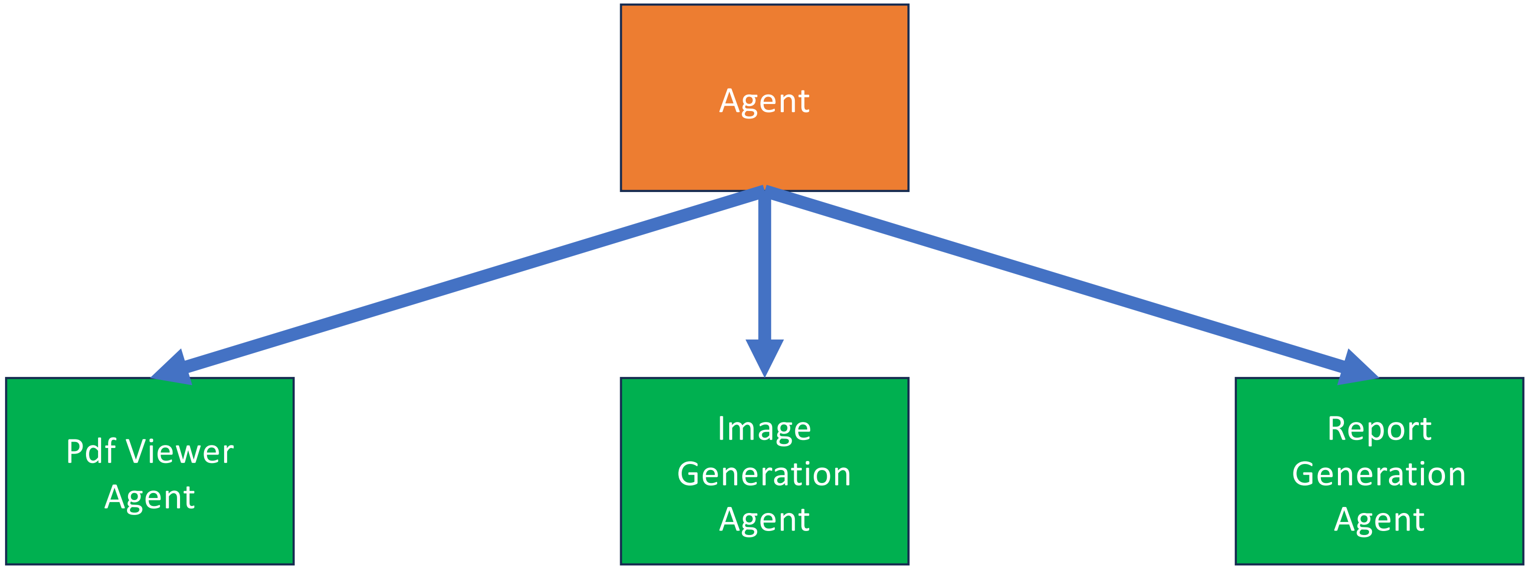
Skills can also include code for Claude to execute as tools at its discretion based on the nature of the task.

We can give the agents predefined functions to use as well as part of the skill!

https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills

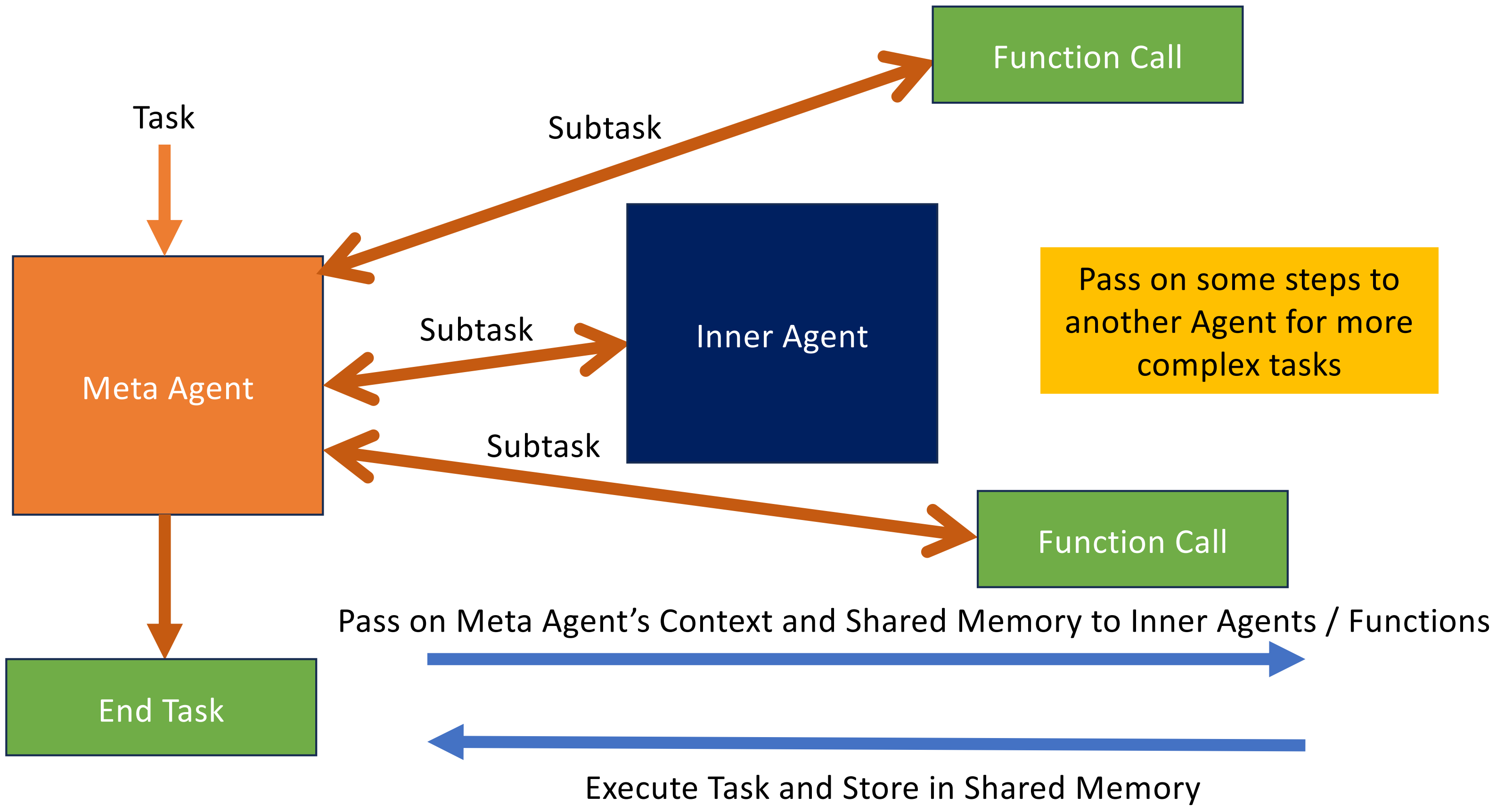
My thoughts

Why use only one agent?



Use minimal context for each agent. Only share information on need-to-know basis.

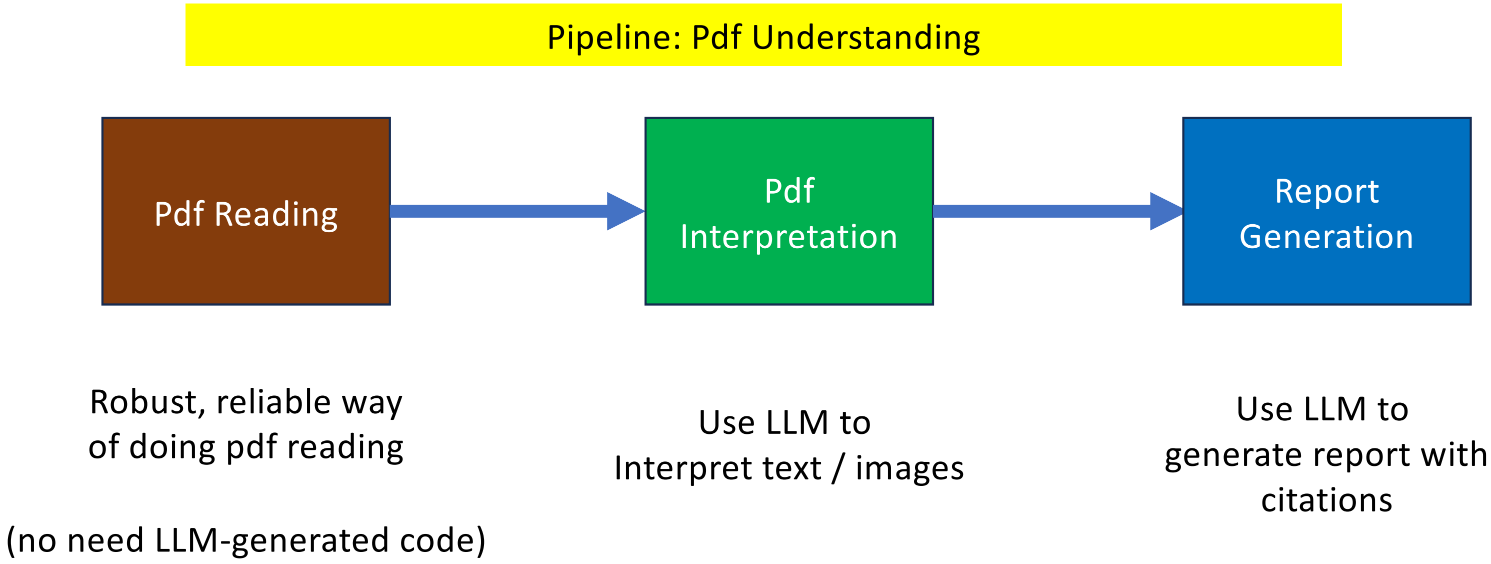
Brief Recap on Agentic Workflow in AgentJo



Task
Meta Agent
Subtask
Function Call
Subtask
Inner Agent
Subtask
Function Call
End Task
Pass on some steps to another Agent for more complex tasks
Pass on Meta Agent's Context and Shared Memory to Inner Agents / Functions
Execute Task and Store in Shared Memory

https://github.com/tanchongmin/agentjo

Why even use agents?



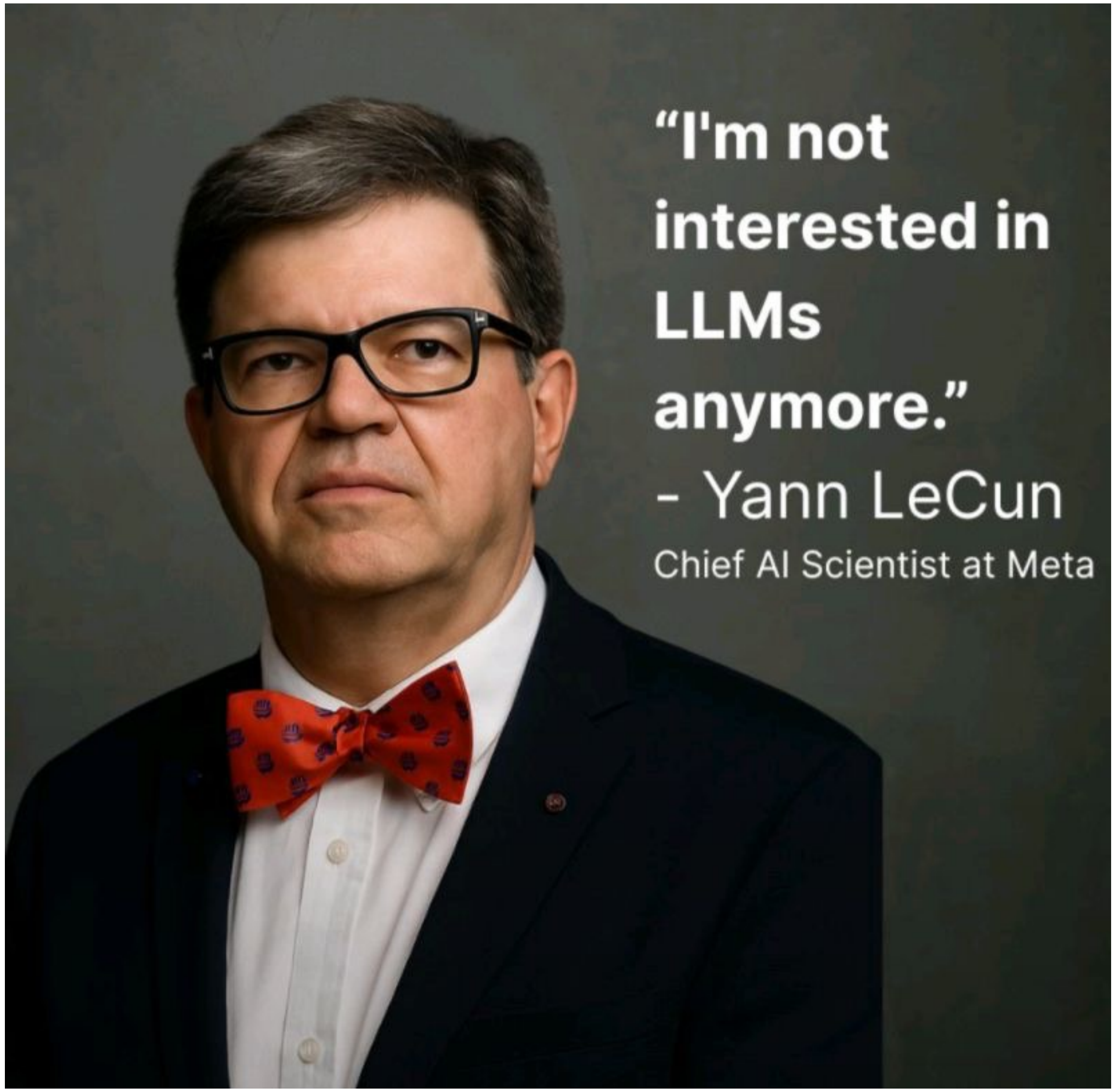
Robust, reliable way of doing pdf reading

(no need LLM-generated code)

Use LLM to Interpret text / images

Use LLM to generate report with citations

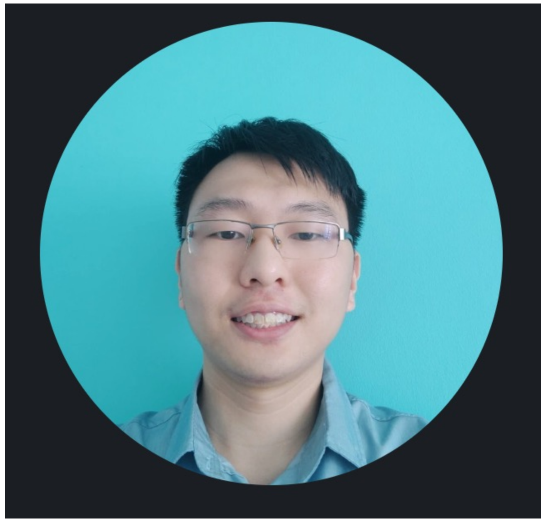
Each part of the pipeline might or might not use LLMs. Context is already pre-fed to each part


“I'm not interested in LLMs anymore.”
- Yann LeCun
Chief AI Scientist at Meta

https://www.linkedin.com/posts/matthewoldach_yann-lecun-recently-made-a-striking-statement-activity-7385024037418274816-1_kc/

"Use LLM agents only if you cannot accomplish a task using rule-based methods!"

- John Tan Chong Min



Question to Ponder

- How can skills be learned continually and how can the skill library be updated?
- How many additional skills (context) can an agent load at runtime?
- Should an agent be given many skills, or should we use few agents with specialised skills?
- Should we just run a fixed pipeline, rather than rely on an agent's dynamic code generation? (e.g. read pdfs)