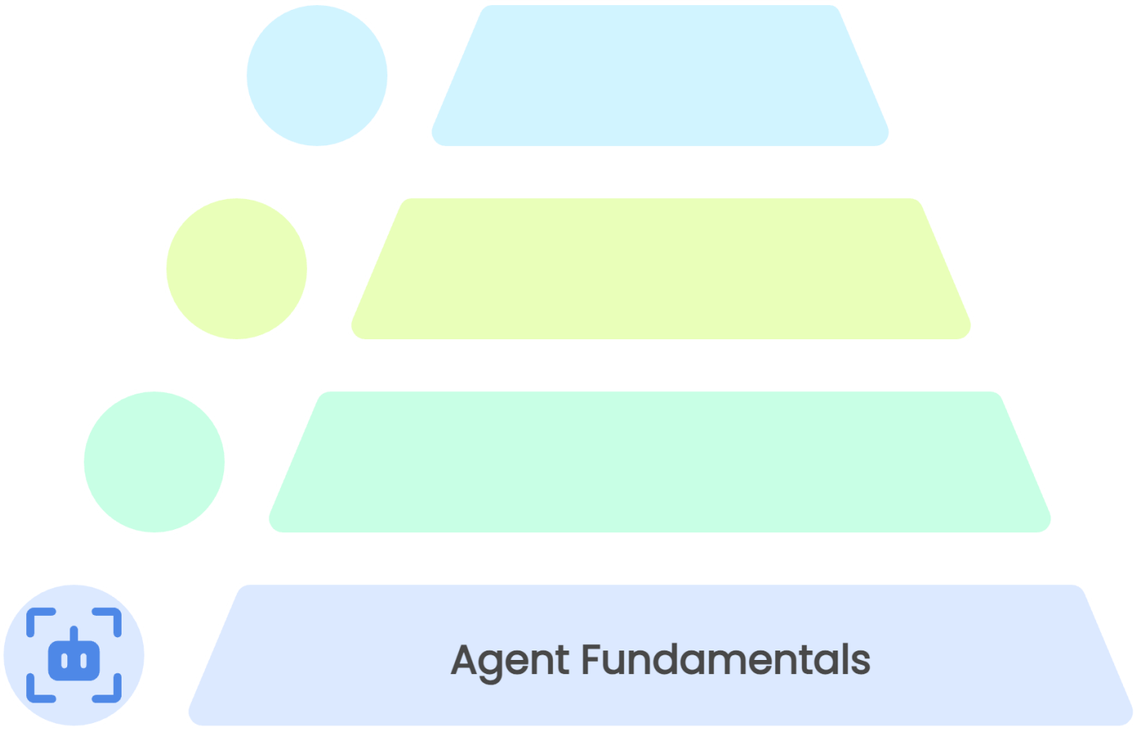
Introduction to Hugging Face smolagents

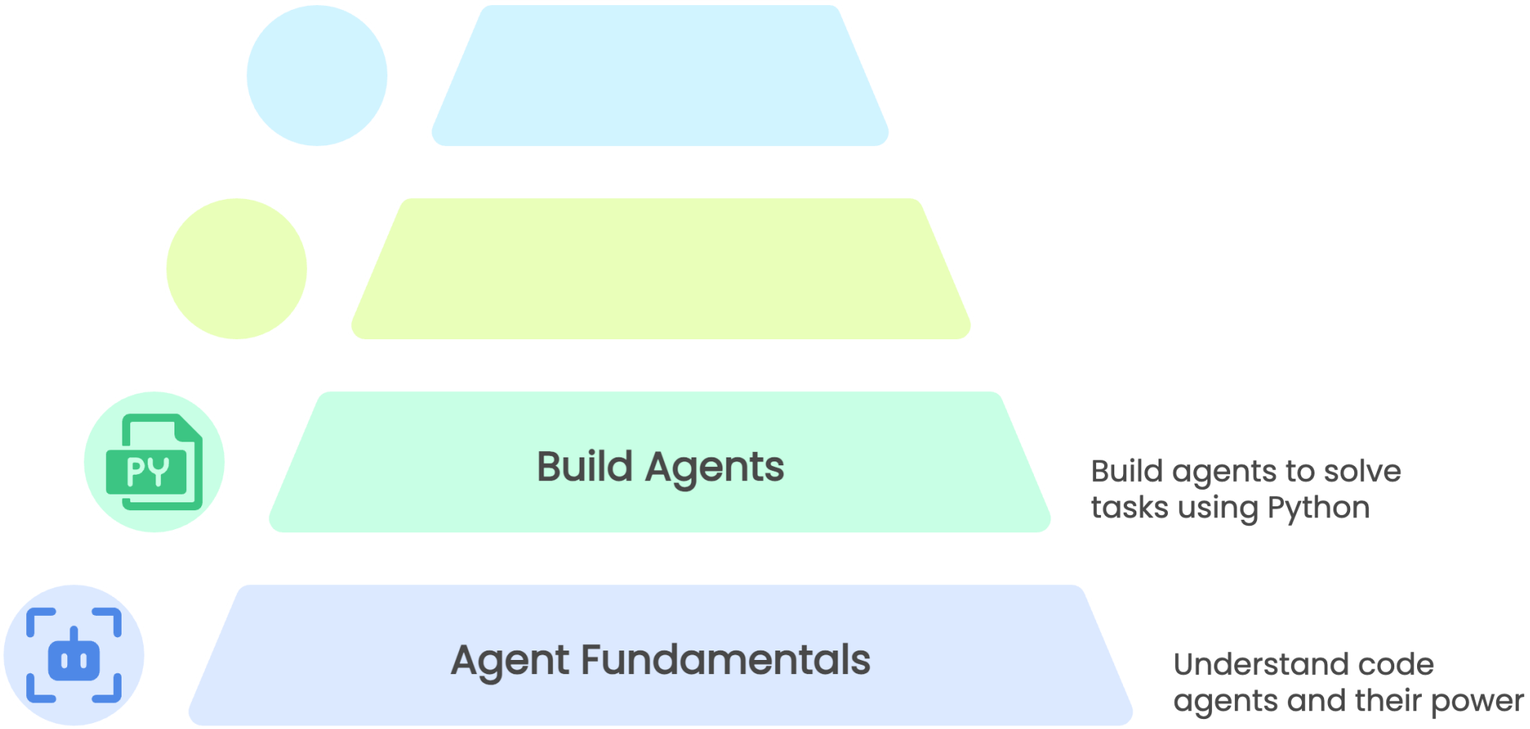
AI AGENTS WITH HUGGING FACE SMOLAGENTS

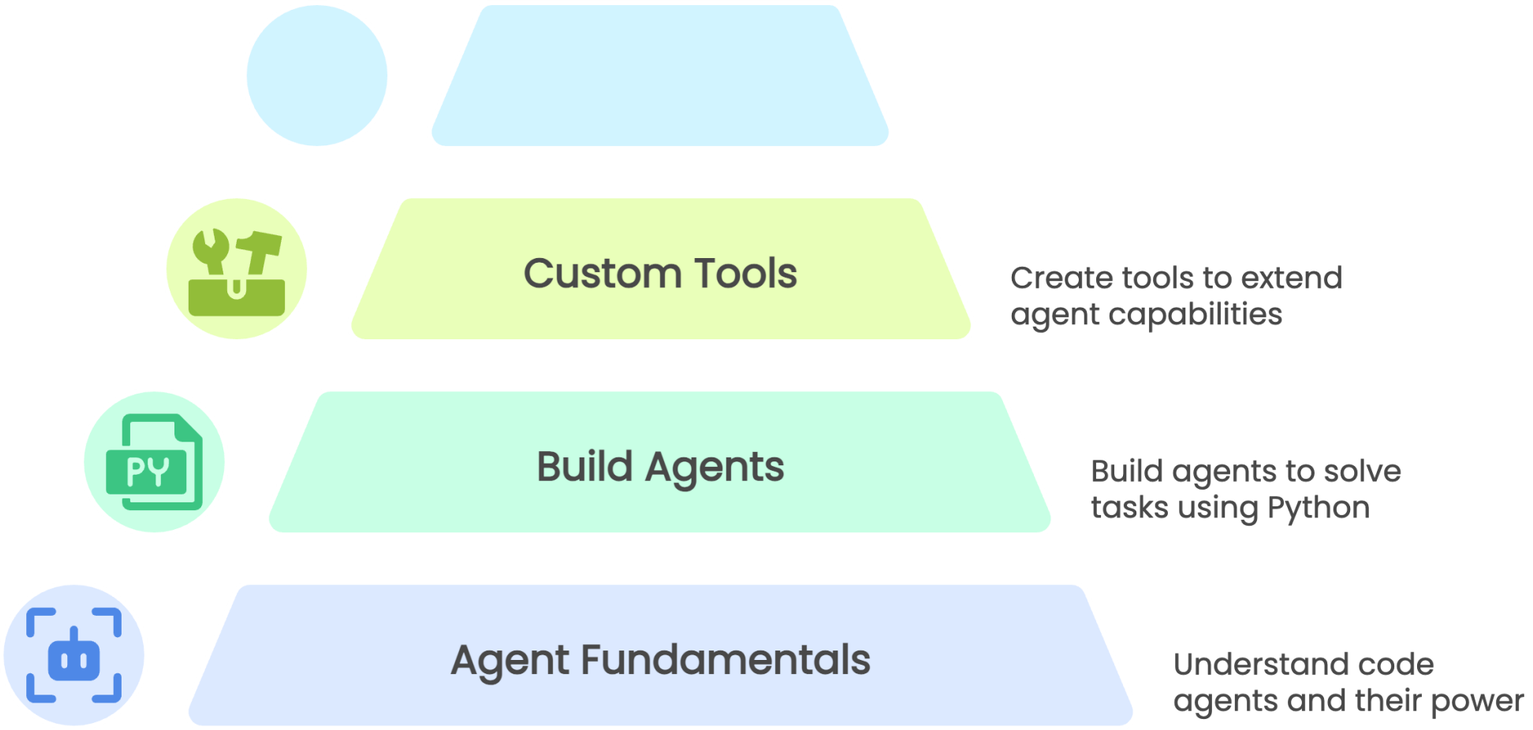
Adel Nehme
VP of AI Curriculum, DataCamp

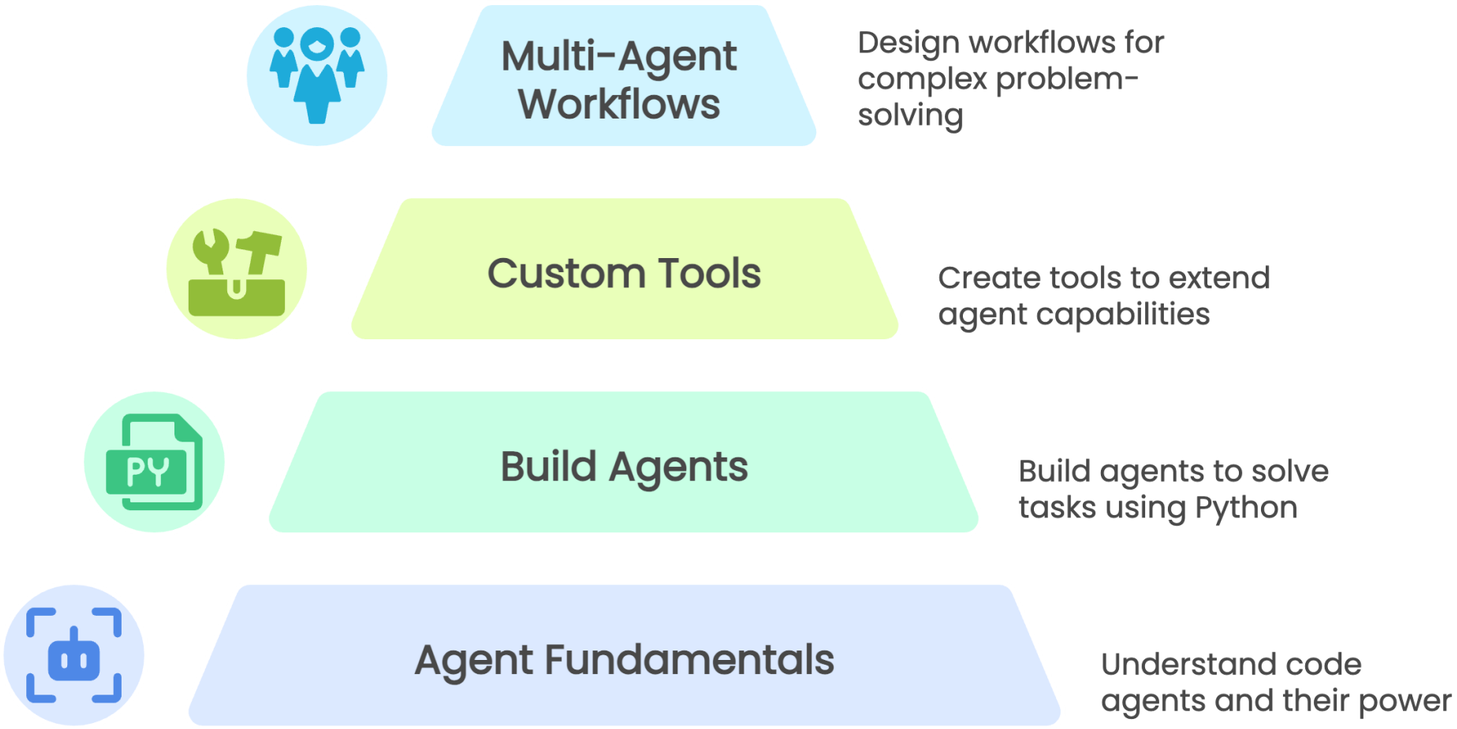
datacamp



Understand code
agents and their power


Build Agents
Build agents to solve tasks using Python
Agent Fundamentals
Understand code agents and their power


Custom Tools
Create tools to extend agent capabilities
Build Agents
Build agents to solve tasks using Python
Agent Fundamentals
Understand code agents and their power


Multi-Agent Workflows
Design workflows for complex problem-solving
Custom Tools
Create tools to extend agent capabilities
PY
Build Agents
Build agents to solve tasks using Python
Agent Fundamentals
Understand code agents and their power

What is an AI agent?

An AI agent is a system that uses an LLM to interact with its environment to achieve a user-defined objective.

From Chatbots to Agents

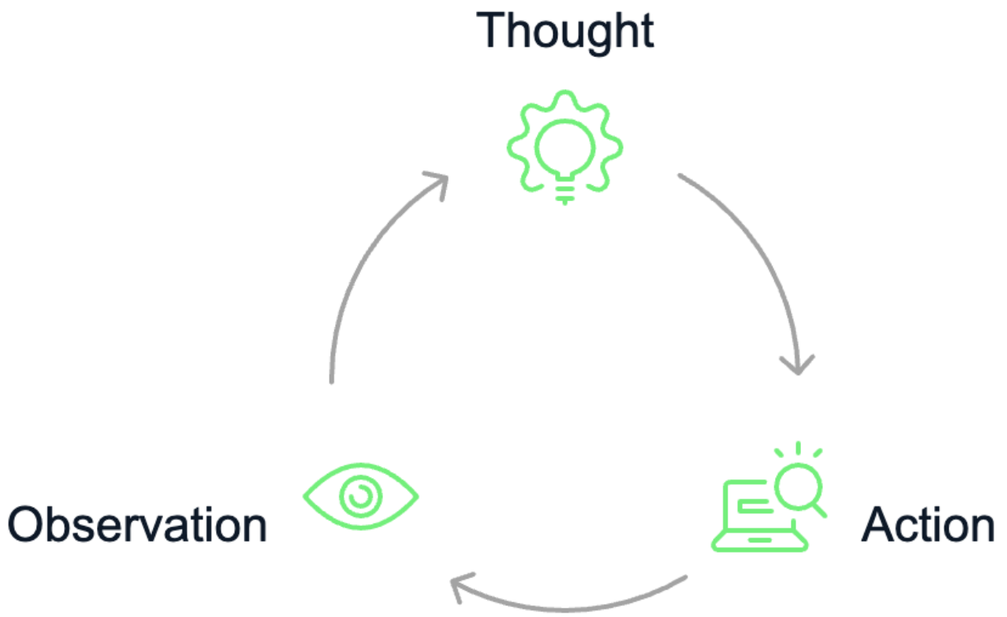
Chatbots:

- Respond with natural language
- Passive and reactive

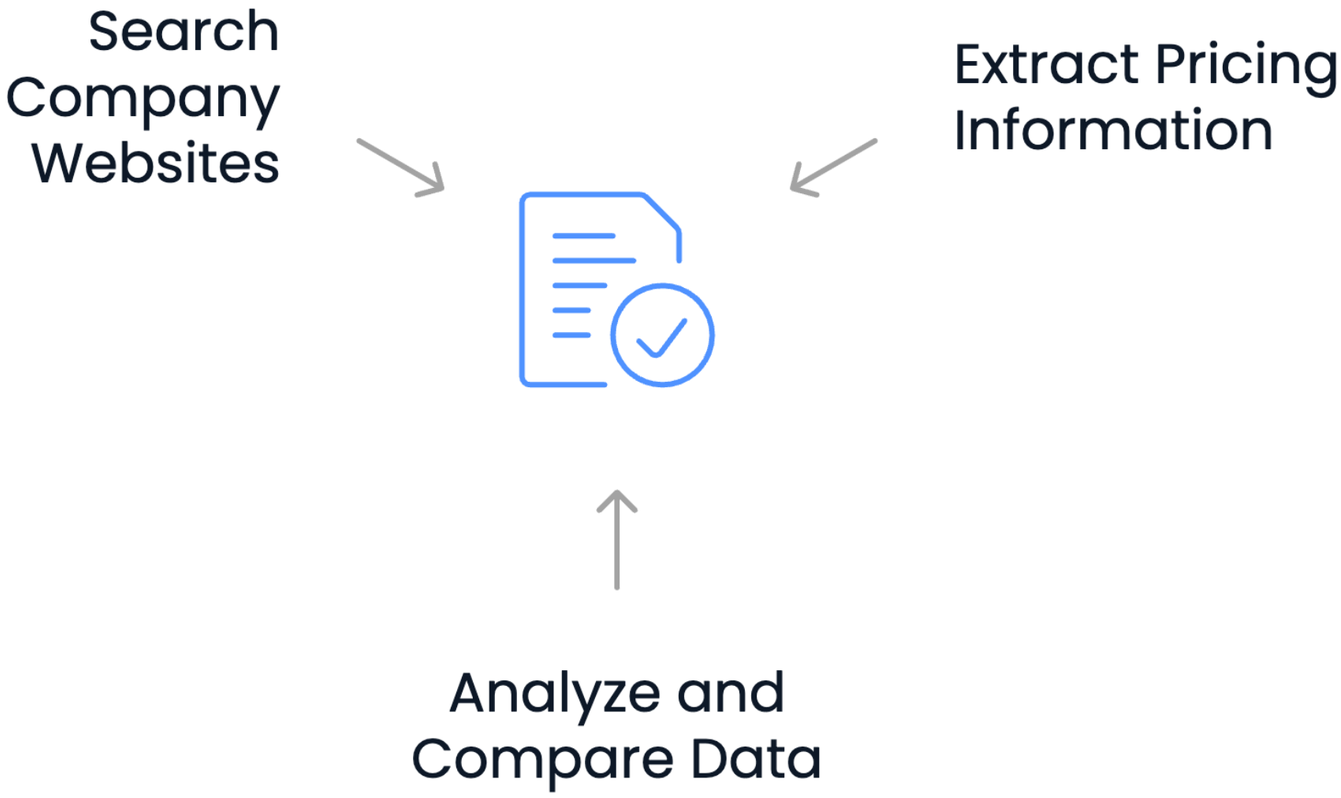
Agents:

- Can take actions (web search, read files, analyze data, etc.)
- Actively reason toward a goal

Agents follow a cycle:



Example: Competitor Pricing Research



All from a single prompt!

What Is smolagents?

- Lightweight Python framework
- Developed by Hugging Face

Supports two types of agents:

- `ToolCallingAgent` : Uses structured function calls
- `CodeAgent` : Writes and runs Python code

How Function-Calling Works

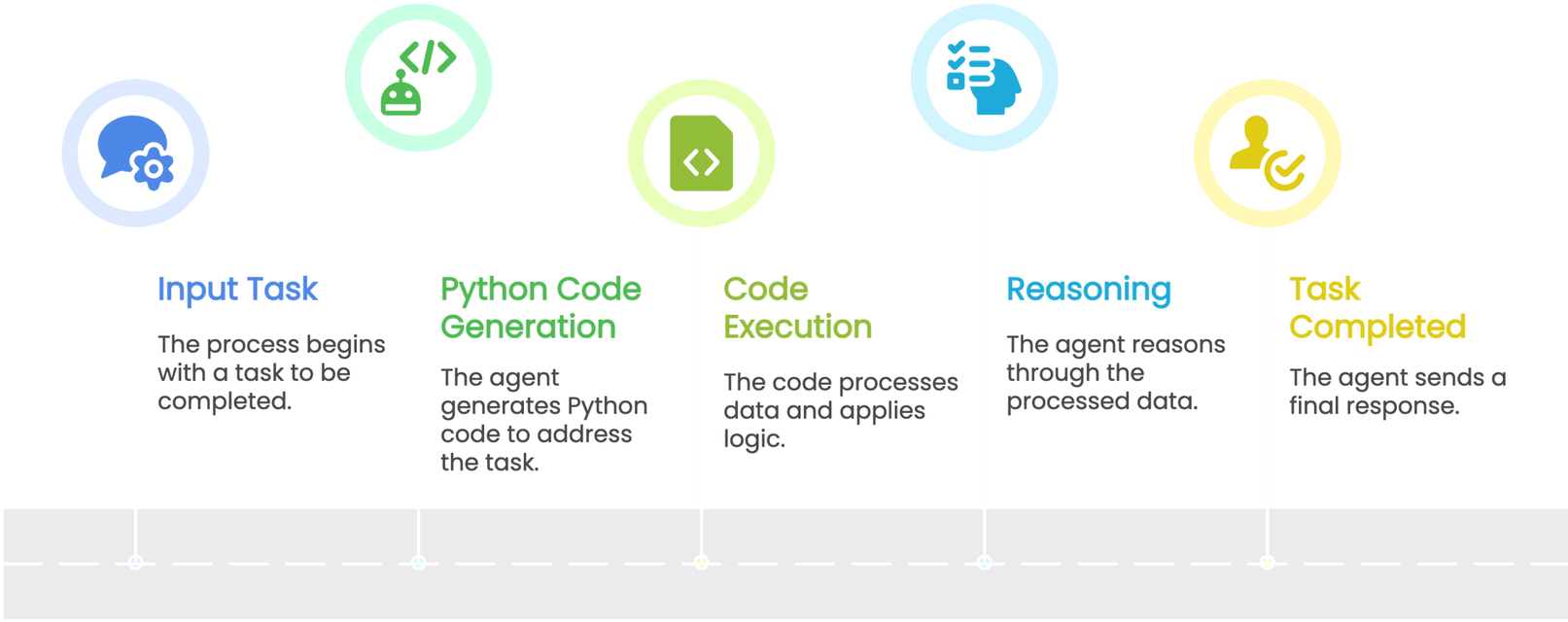

```
Action 1: {"tool": "search_company", "company": "Competitor A"}
Action 2: {"tool": "get_pricing", "company": "Competitor A", "plan": "Basic"}
Action 3: {"tool": "get_pricing", "company": "Competitor A", "plan": "Pro"}
Action 4: {"tool": "search_company", "company": "Competitor B"}
Action 5: {"tool": "get_pricing", "company": "Competitor B", "plan": "Basic"}
```

How Code Agents Work

```
competitors = ["Competitor A", "Competitor B", "Competitor C"]
pricing_data = {}

for company in competitors:
    company_info = search_company(company)
    plans = extract_pricing_plans(company_info)
    pricing_data[company] = plans

most_affordable_option = min(pricing_data,
                             key=lambda x: pricing_data[x]['basic_plan'])
```

The Code Agent Flow

Research shows ~20% higher success rate than function-calling methods.



[1] https://huggingface.co/papers/2402.01030

Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Creating an Agent With Tools

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Adel Nehme
VP of AI Curriculum, DataCamp

datacamp

Creating a Code Agent (No Tools)

```python
from smolagents import CodeAgent, InferenceClientModel

agent = CodeAgent(
    tools=[],
    model=InferenceClientModel()
)
agent.run("Calculate the average of the list [23, 45, 67, 89]")
```

```
Executing parsed code:
  numbers = [23, 45, 67, 89]
  average = sum(numbers) / len(numbers)
  final_answer(average)
Final answer: 56.0
[Step 1: Duration 4.14 seconds| Input tokens: 1,900 | Output tokens: 109]
56.0
```

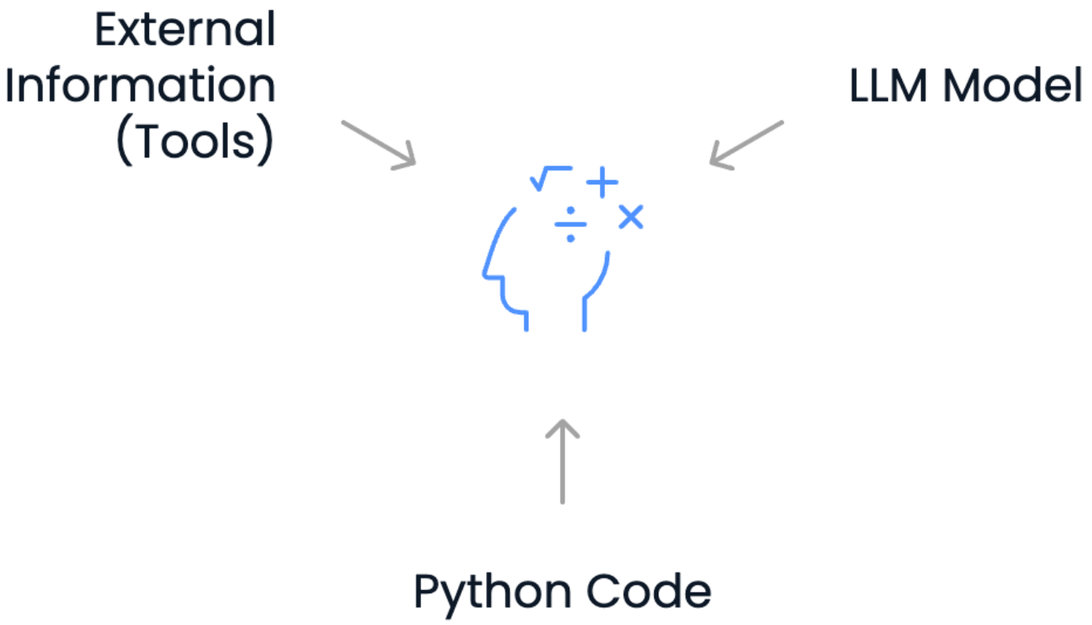
Why Use Tools with Code Agents?

The agent we defined can already solve many tasks using:

- An LLM model
- Python code

But it may also need access to external information:

- Example: live web data



That's where tools come in!

Adding a Web Search Tool

```
from smolagents import CodeAgent, InferenceClientModel, WebSearchTool


agent = CodeAgent(
    model=InferenceClientModel(),
    tools=[WebSearchTool()]
)
```

Code Agent With Web Search Tool Output

```
agent.run("What's the tallest building in the world right now?")
```

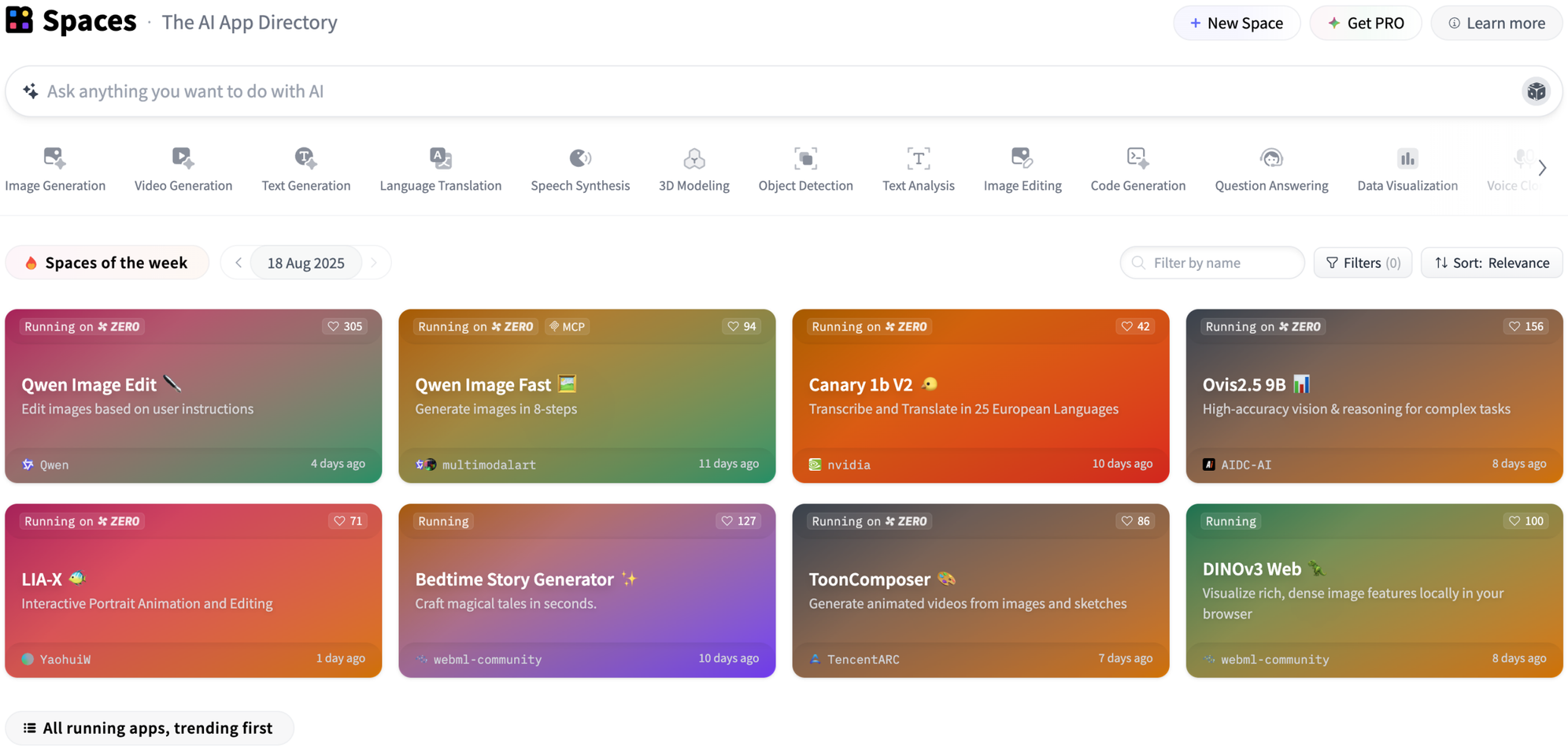
```
Executing parsed code:
  tallest_building_info = web_search("tallest building in the world 2023")
  print(tallest_building_info)

# Search results omitted for brevity...

Executing parsed code:
  final_answer("Burj Khalifa, Dubai, 828 meters")
Final answer: Burj Khalifa, Dubai, 828 meters
[Step 2: Duration 2.97 seconds| Input tokens: 5,078 | Output tokens: 153]
Burj Khalifa, Dubai, 828 meters
```

Built-in Tools (by Category)

Category	Tools
Information Retrieval	`ApiWebSearchTool`, `DuckDuckGoSearchTool`, `GoogleSearchTool`, `WebSearchTool`, `WikipediaSearchTool`
Web Interaction	`VisitWebpageTool`
Code Execution	`PythonInterpreterTool`
User Interaction	`UserInputTool`
Speech Processing	`SpeechToTextTool`
Workflow Control	`FinalAnswerTool`

[1] https://huggingface.co/docs/smolagents/main/en/reference/default_tools

Tools From the Hugging Face Hub



Using Community Tools from Hugging Face

```
from smolagents import load_tool

# Load remote tool from Hugging Face
model_downloads_tool = load_tool(
    repo_id="example-repo/hf-model-downloads",
    trust_remote_code=True
)

# Create agent with remote + built-in tools
agent = CodeAgent(
    tools=[model_downloads_tool, WebSearchTool()],
    model=InferenceClientModel()
)
agent.run("Find the most downloaded image classification model on Hugging Face")
```

```
google/vit-base-patch16-224-in21k
```

Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Creating an Agent With Custom Tools

AI AGENTS WITH HUGGING FACE SMOLAGENTS

Adel Nehme
VP of AI Curriculum, DataCamp

datacamp

If code agents can already write and run Python and use built-in tools, why would I need to create custom tools at all?

Because custom tools give you important advantages

Benefits of Custom Tools

- **Reliability:** Write and test logic explicitly instead of relying on the agent to guess
- **Reusability:** Use tools across projects and agents
- **Consistency:** Get predictable behavior across runs (great for debugging)
- **Controlled access:** Expose only what you want (files, APIs, databases, etc.)

Scenario: You Run a Retail Store

- Inventory data is stored in a CSV file (size, color, quantity, and price)
- Code agents can write code to read CSVs
- But they don't have access to files by default
- You need to wrap file access in a custom tool

Anatomy of a Custom Tool

```python
from smolagents import tool
import pandas as pd

@tool
def check_inventory(product_name: str) -> int:
    """
    Check the available quantity of a product in the inventory CSV.

    Args:
        product_name (str): The name of the product to look up.

    Returns:
        int: The quantity in stock. Returns 0 if the product is not found.
    """
    df = pd.read_csv("store_inventory.csv")
    match = df[df["product_name"] == product_name]
    stock_quantity = int(match.iloc[0]["quantity"]) if not match.empty else 0
    return stock_quantity
```

Best Practices for Custom Tools

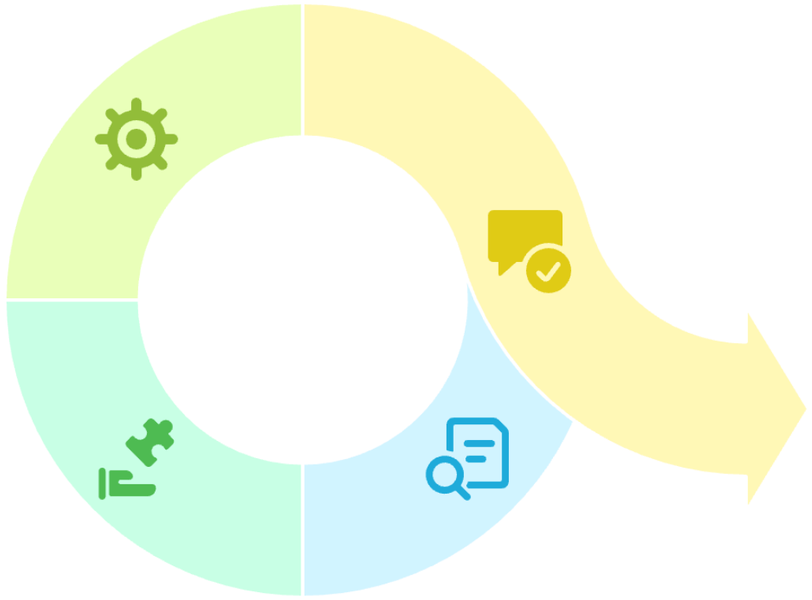
```python
@tool
def check_inventory(product_name: str) -> int:
    """
    Check the available quantity of a product in the inventory CSV.

    Args:
        product_name (str): The name of the product to look up.

    Returns:
        int: The quantity in stock. Returns 0 if the product is not found.
    """
    df = pd.read_csv("store_inventory.csv")
    match = df[df["product_name"] == product_name]
    stock_quantity = int(match.iloc[0]["quantity"]) if not match.empty else 0
    return stock_quantity
```

Parameter

Type hints

Docstring

Do we have any t-shirts in stock?



1

Analyze Question

The agent interprets the user's query.

2

Match to Tool

The agent identifies the appropriate tool.

3

Execute Tool

The agent runs the tool with necessary input.

4

Formulate Response

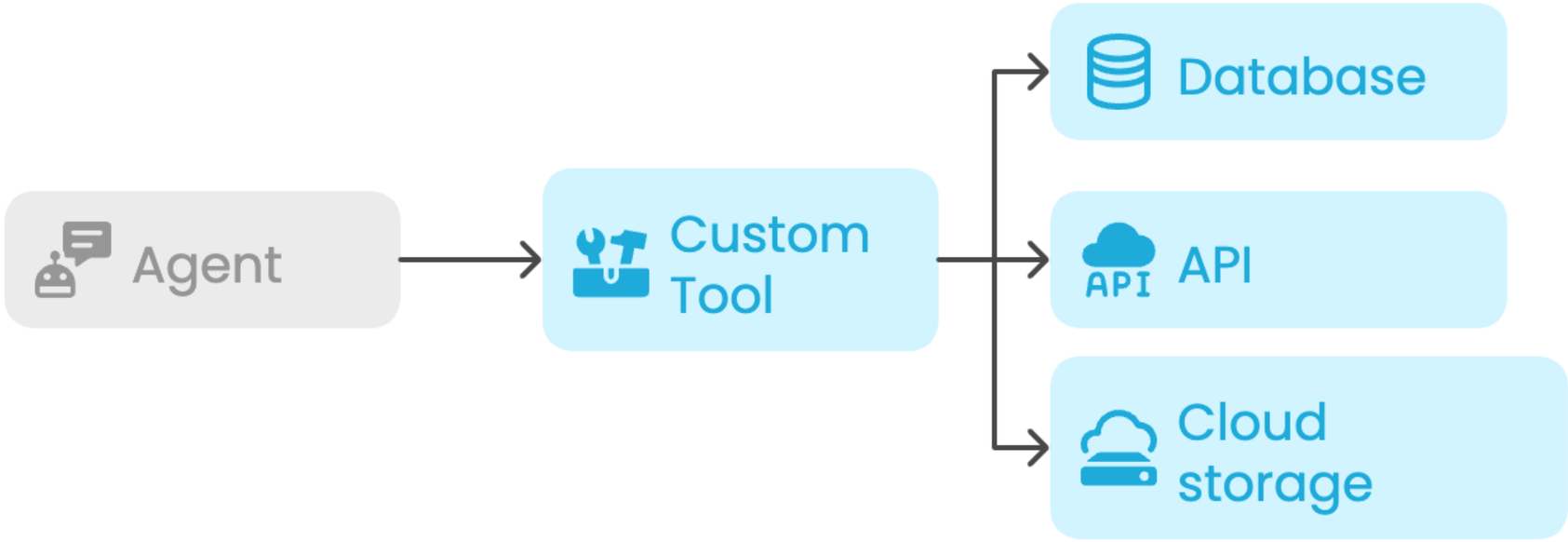
The agent crafts a helpful answer.

Registering a Custom Tool with Your Agent

```
from smolagents import CodeAgent

agent = CodeAgent(
    tools=[check_inventory], # Add custom tool
    model=InferenceClientModel(),
    additional_authorized_imports=["pandas"]  # Allow external package
)

agent.run("Do we have any large t-shirts in stock?")
```

```
Yes, we have 8 large t-shirts in stock.
```

Custom Tools in Production Projects



Let's practice!

AI AGENTS WITH HUGGING FACE SMOLAGENTS