Guiding unstructured responses

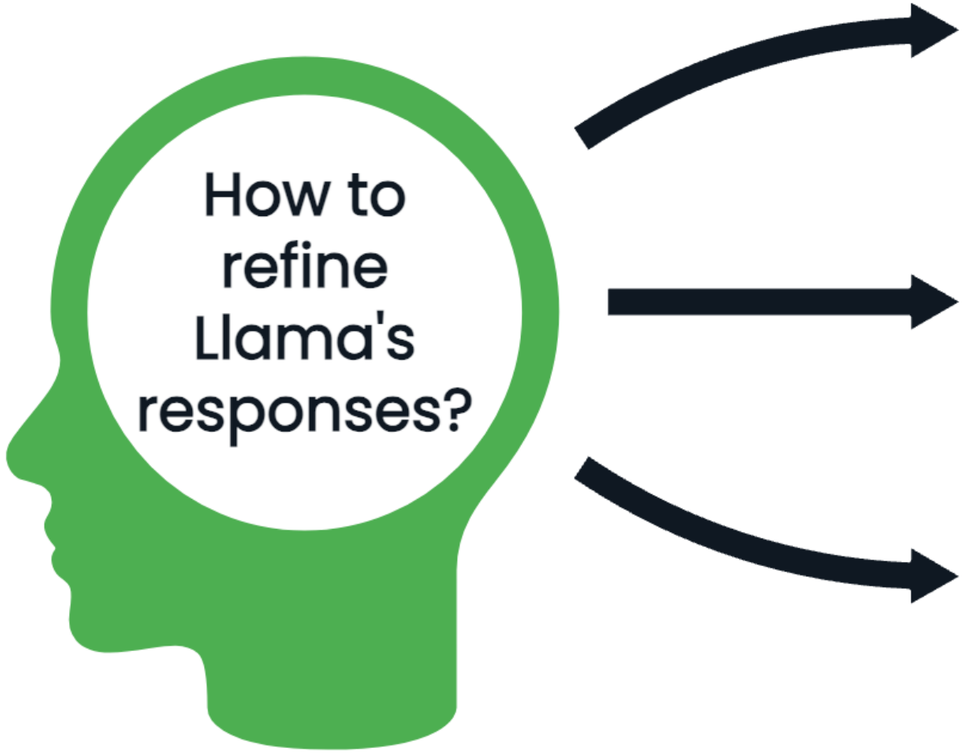
WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

datacamp

Controlling model output

- Parameters
- Roles

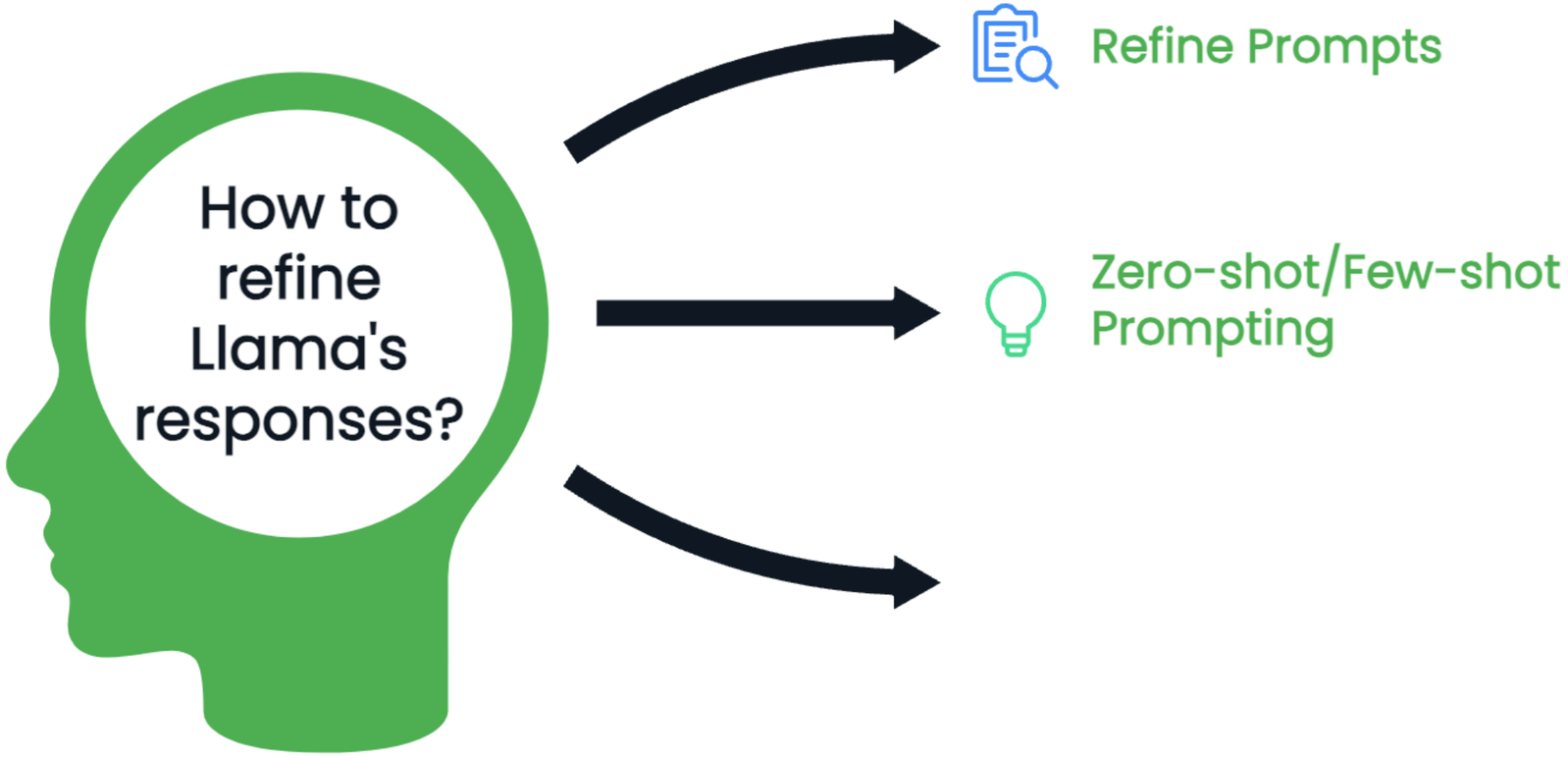


Controlling model output

- Parameters
- Roles

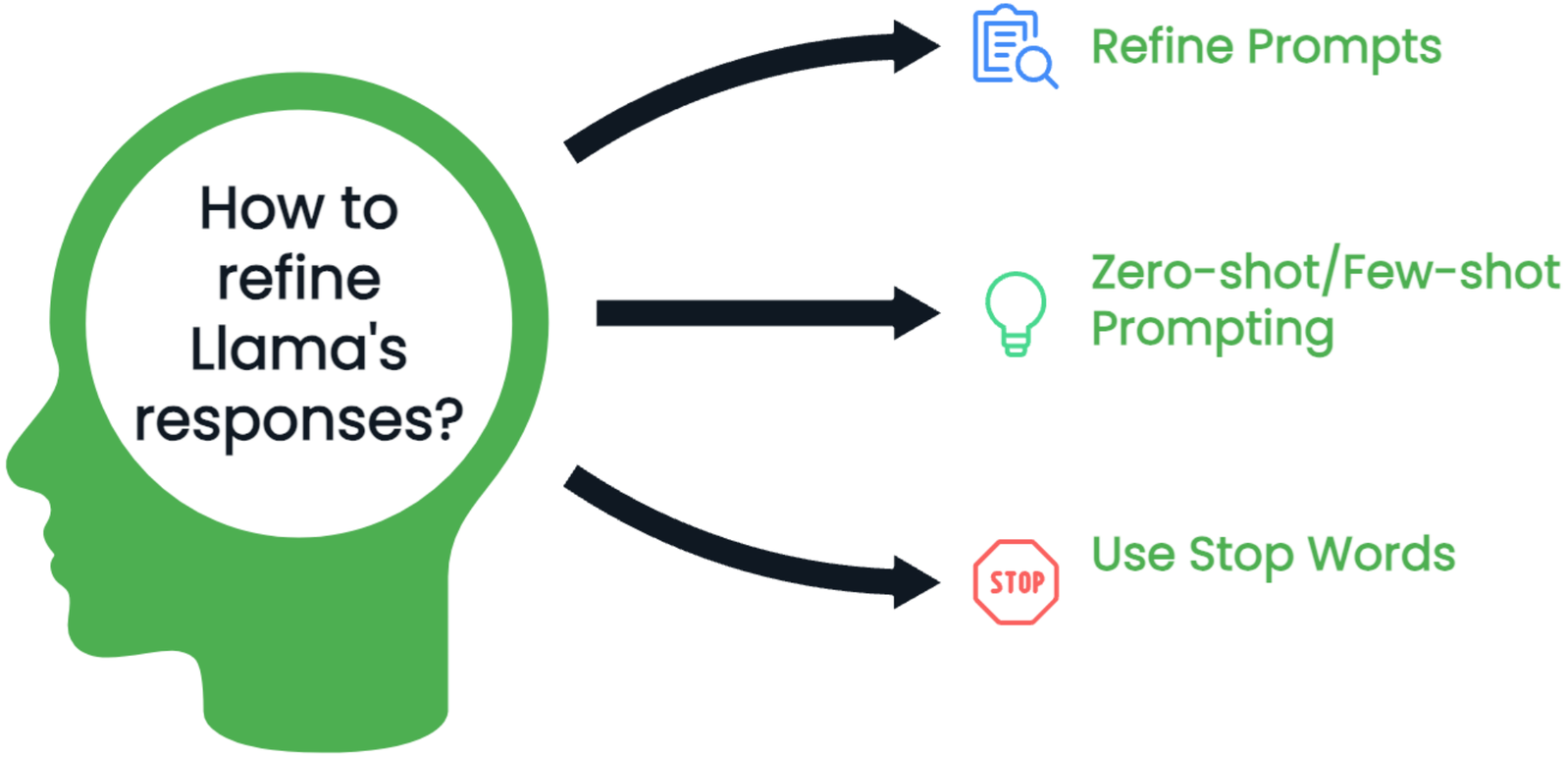
Controlling model output

- Parameters
- Roles



Controlling model output

- Parameters
- Roles



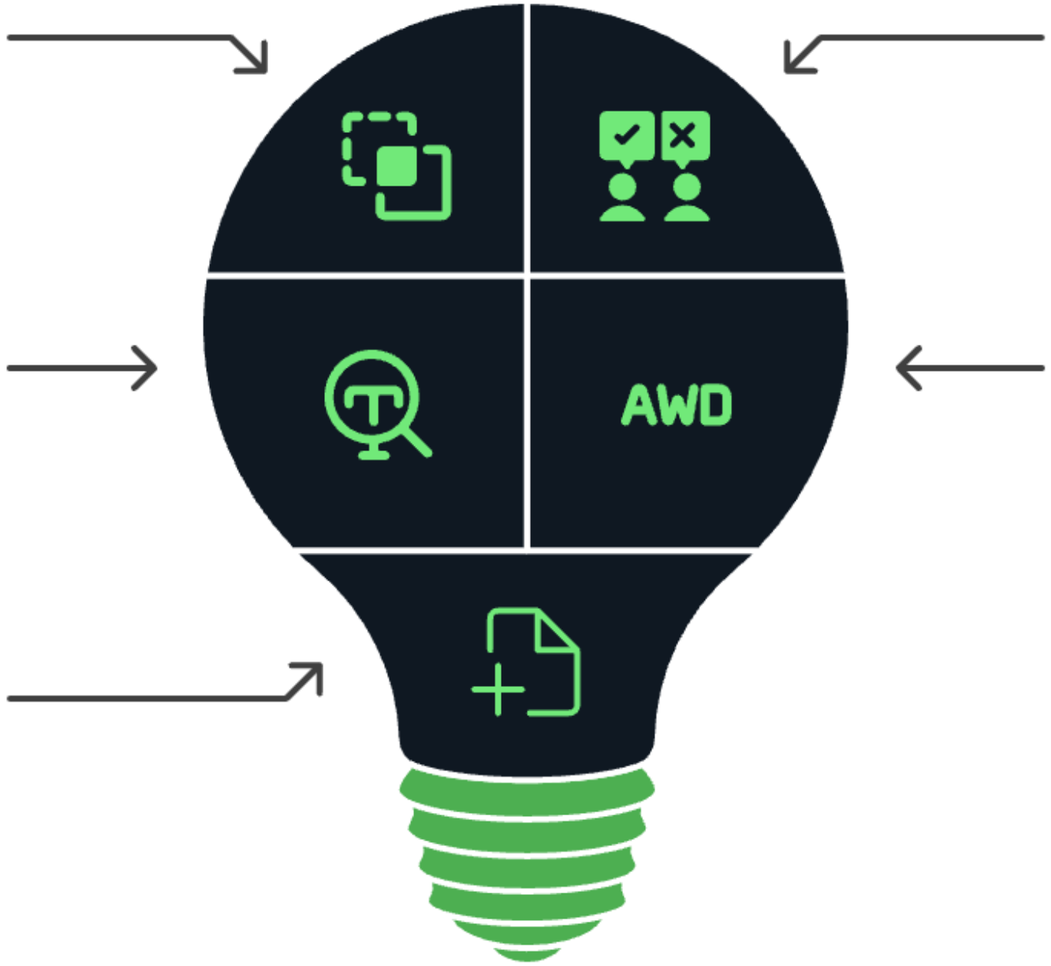
Refining prompts

- Example: summarization

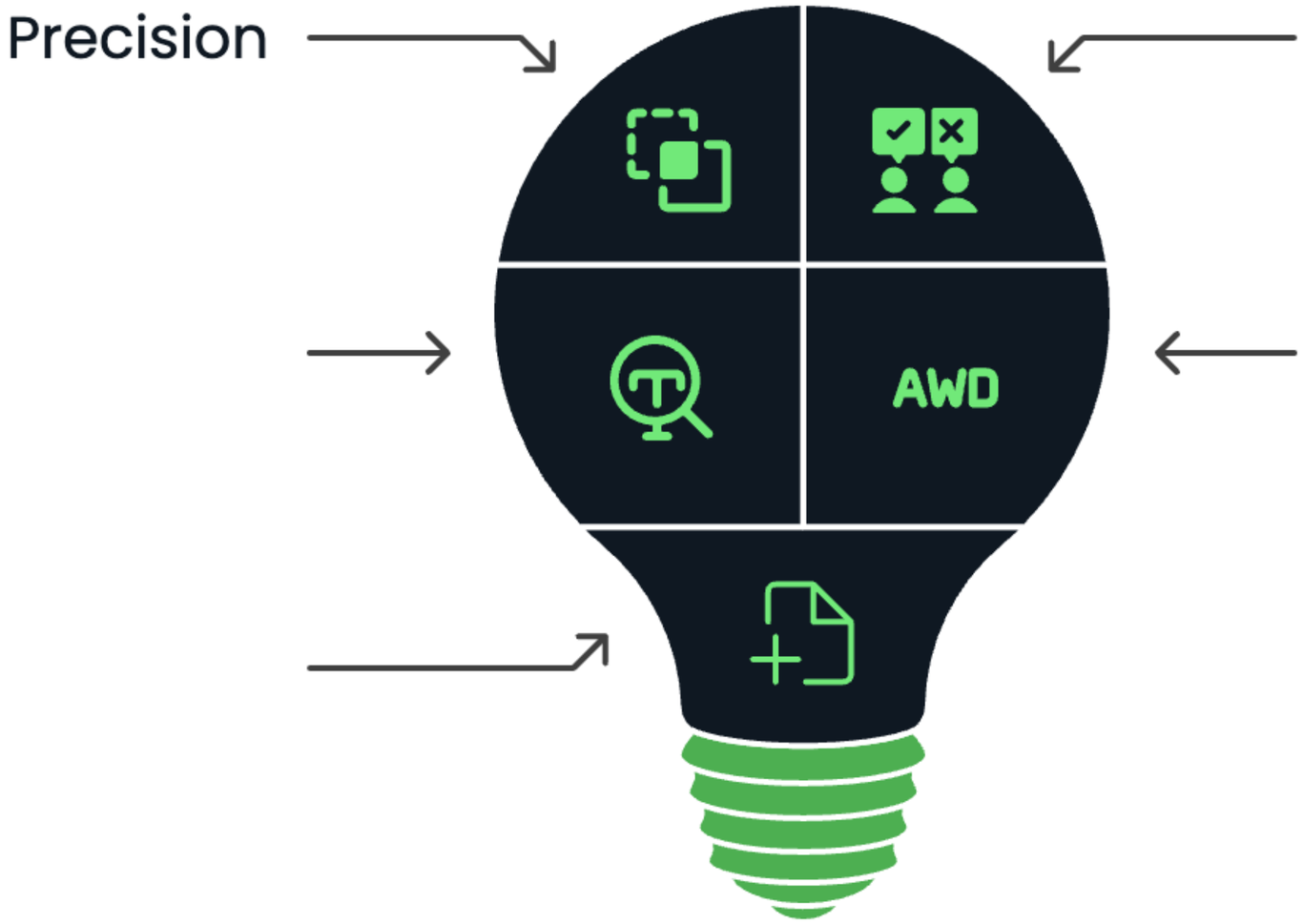
```
text_choice1 = "Summarize key trends in the aviation industry from the last year,
                focusing on fuel efficiency innovations."

text_choice2 = "Tell me about the aviation industry."

output = llm(text_choice1) # More specific prompt is more effective

print(output['choices'][0]['text'])
```

```
The aviation industry has made significant strides in fuel efficiency innovations
over the last year, driven by the need to reduce greenhouse gas emissions and
operating costs. Sustainable Aviation Fuels (SAFs) have emerged as a key trend...
```

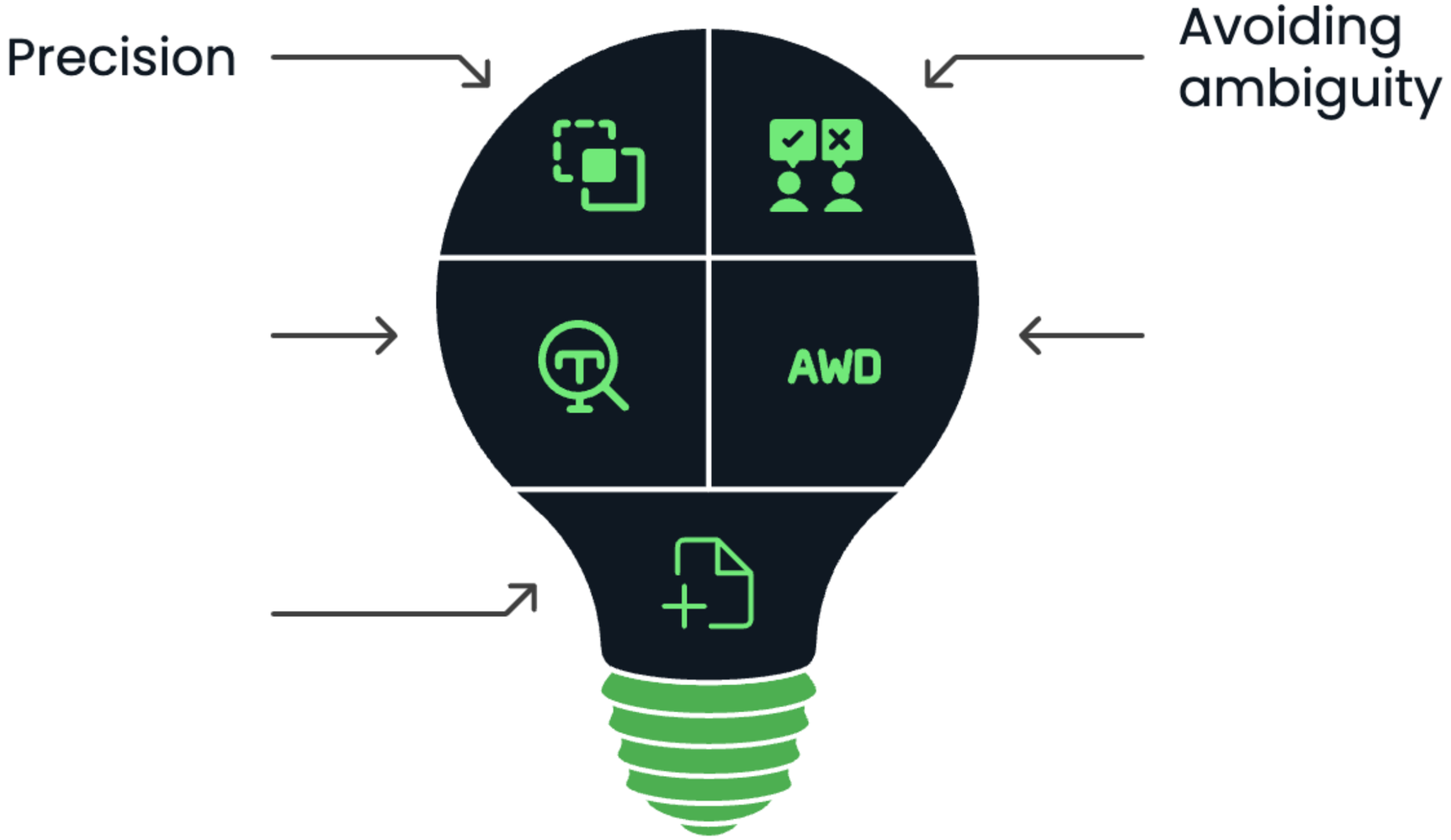
Components of effective prompting



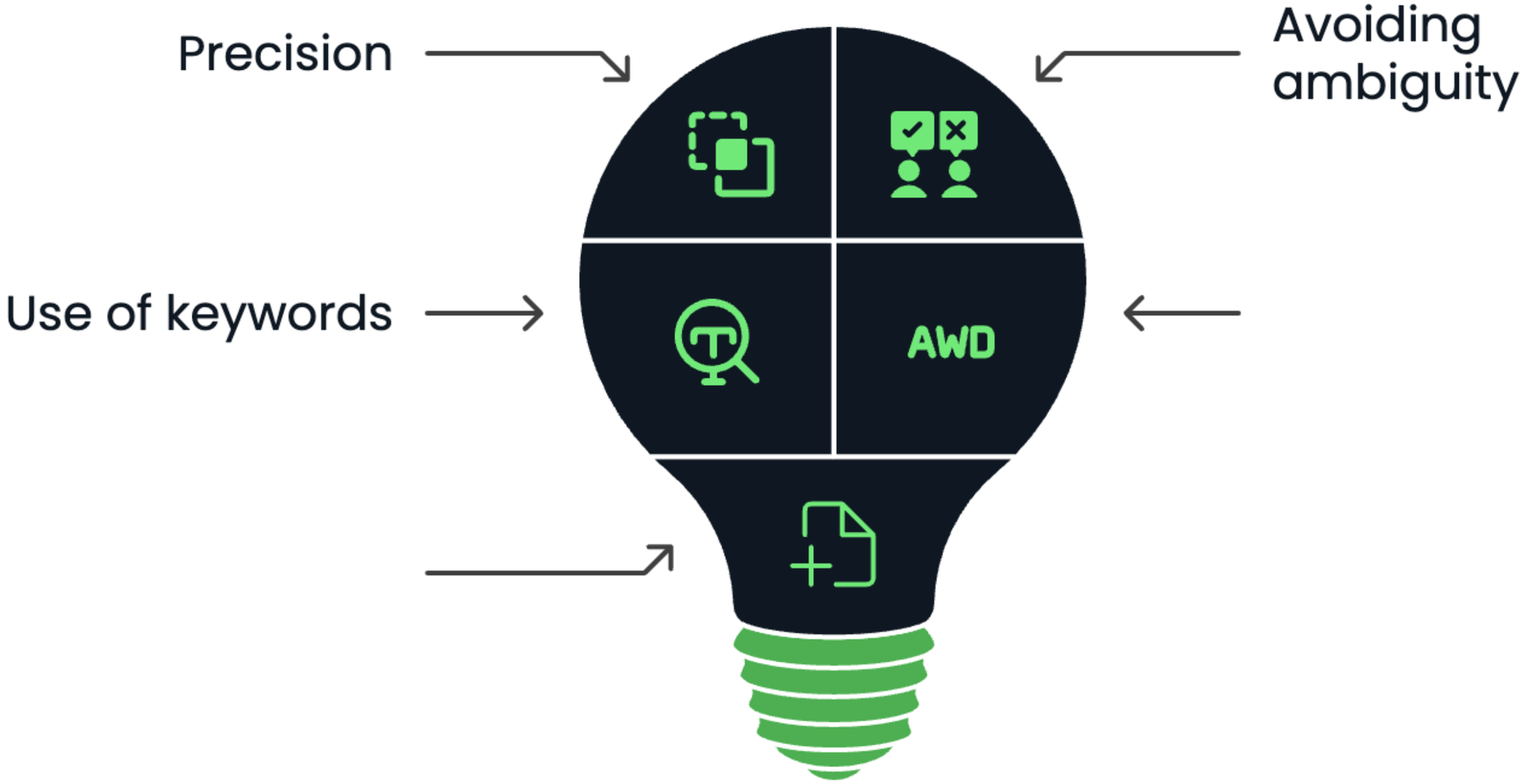
Components of effective prompting



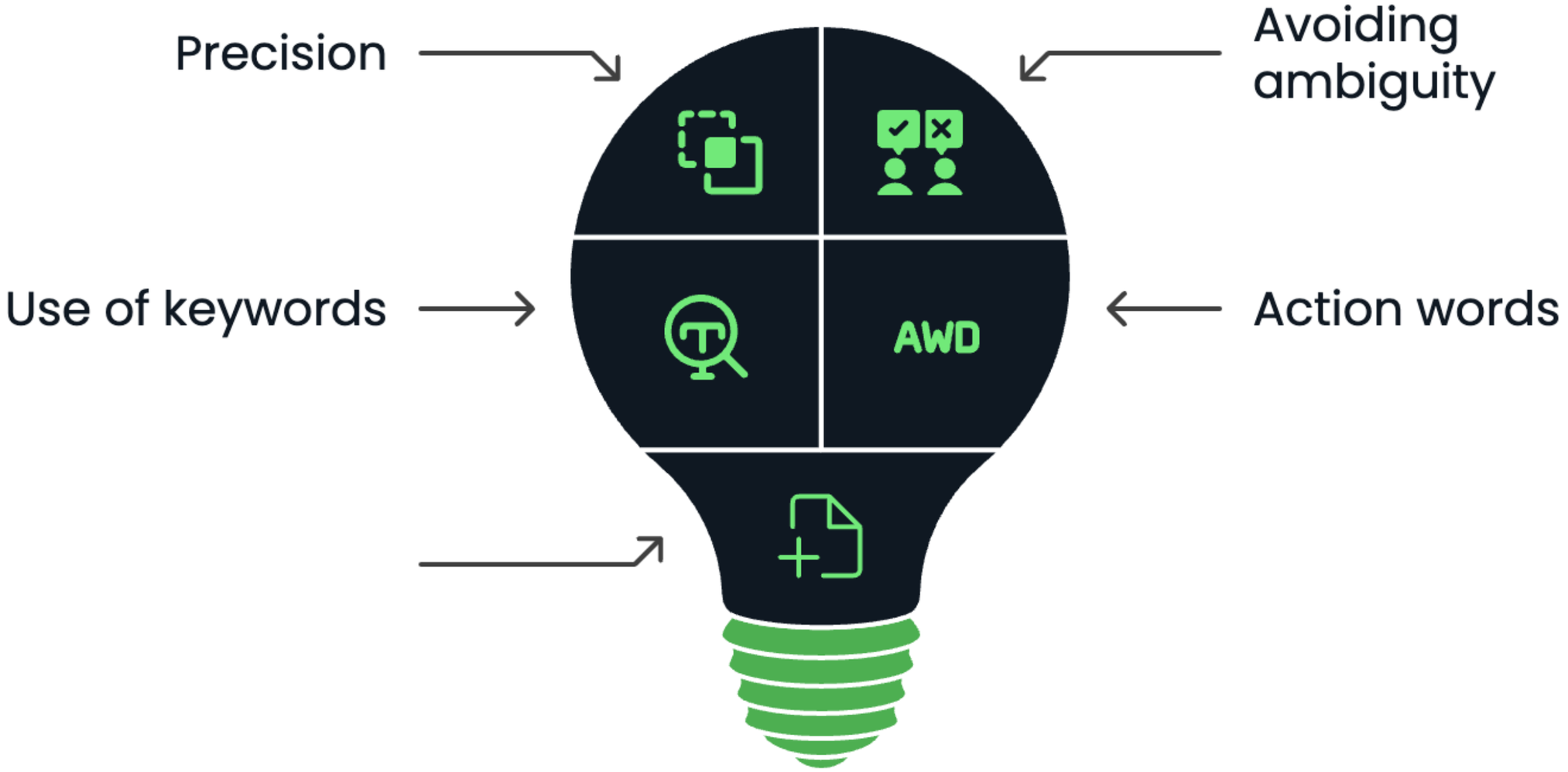
Components of effective prompting



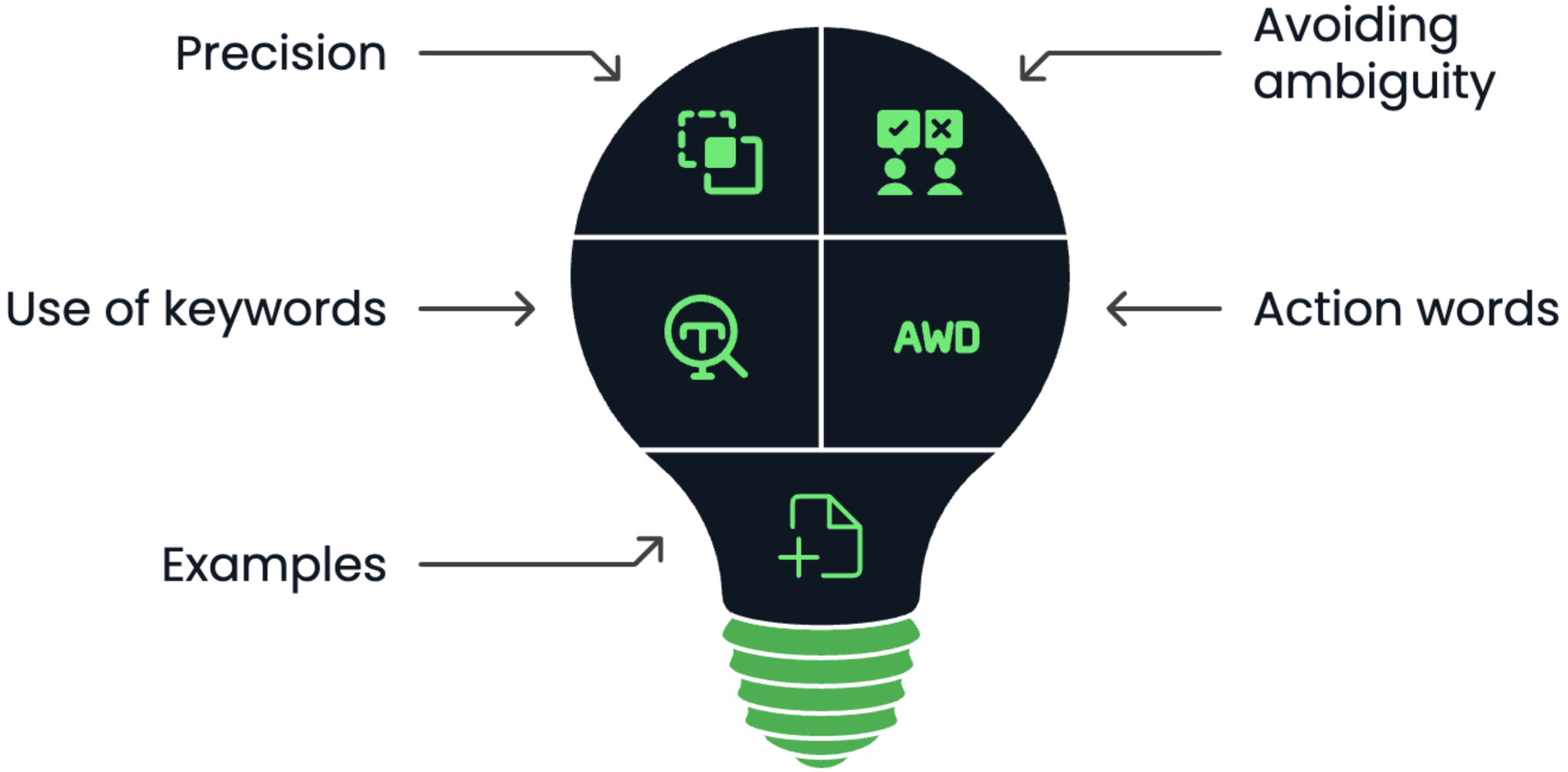
Components of effective prompting



Components of effective prompting



Components of effective prompting



Zero-shot prompting

- Zero-shot prompting: a single instruction

```
text = "Summarize recent mergers in the airline industry."

output = llm(text)

print(output['choices'][0]['text'])
```

```
Recent mergers in the airline industry include Alaska Air Group's acquisition of
Hawaiian Airlines in 2024, with both airlines continuing to operate as separate
brands. In 2022, Delta Air Lines purchased 20% of LATAM Airlines Group...
```

Refining zero-shot prompts

- Need to distinguish task, expected output, additional context
- Zero-shot prompting with labels

```
text = """
       INSTRUCTION: Write concisely and in 2-3 sentences that cover only key points.
       QUESTION: Summarize recent mergers in the airline industry.
       ANSWER:
       """
```

Few-shot prompting

- Few-shot prompting: use of multiple examples

```
text = """
Aircraft Model: Boeing 787-9
Passenger Capacity: 296
Fuel Consumption: 2.5 liters per seat per 100 km

Aircraft Model: Airbus A321XLR
Passenger Capacity: 244
Fuel Consumption: 2.9 liters per seat per 100 km

Aircraft Model:
"""
```

Few-shot prompting

- Few-shot prompting: use of multiple examples

```
text = """
Aircraft Model: Boeing 787-9
Passenger Capacity: 296
Fuel Consumption: 2.5 liters per seat per 100 km

Aircraft Model: Airbus A321XLR
Passenger Capacity: 244
Fuel Consumption: 2.9 liters per seat per 100 km

Aircraft Model:
"""
```

Few-shot prompting

- Few-shot prompting: use of multiple examples

```
text = """
Aircraft Model: Boeing 787-9
Passenger Capacity: 296
Fuel Consumption: 2.5 liters per seat per 100 km

Aircraft Model: Airbus A321XLR
Passenger Capacity: 244
Fuel Consumption: 2.9 liters per seat per 100 km

Aircraft Model:
"""
```

Few-shot prompting

- Few-shot prompting: use of multiple examples

```
text = """
Aircraft Model: Boeing 787-9
Passenger Capacity: 296
Fuel Consumption: 2.5 liters per seat per 100 km

Aircraft Model: Airbus A321XLR
Passenger Capacity: 244
Fuel Consumption: 2.9 liters per seat per 100 km

Aircraft Model:
"""
```

Few-shot prompting

```
output = llm(f"Continue the entries: {text}")


print(output['choices'][0]['text'])
```

```
Aircraft Model: Airbus A350-900
Passenger Capacity: 350
Fuel Consumption: 2.39 liters per seat per 100 km
```

Using stop words

- Need concise insights
- Use `stop` words to end the response at a specific point
- Example: question-answering application

```
text = "Which airlines operate direct flights from London to Singapore?"

output = llm(text, stop=["Q:"]) # Stop responses at "Q:"
print(output['choices'][0]['text'])
```

```
You can fly direct from London to Singapore with Singapore Airlines and
British Airways.
```

Let's practice!

WORKING WITH LLAMA 3

Generating structured output

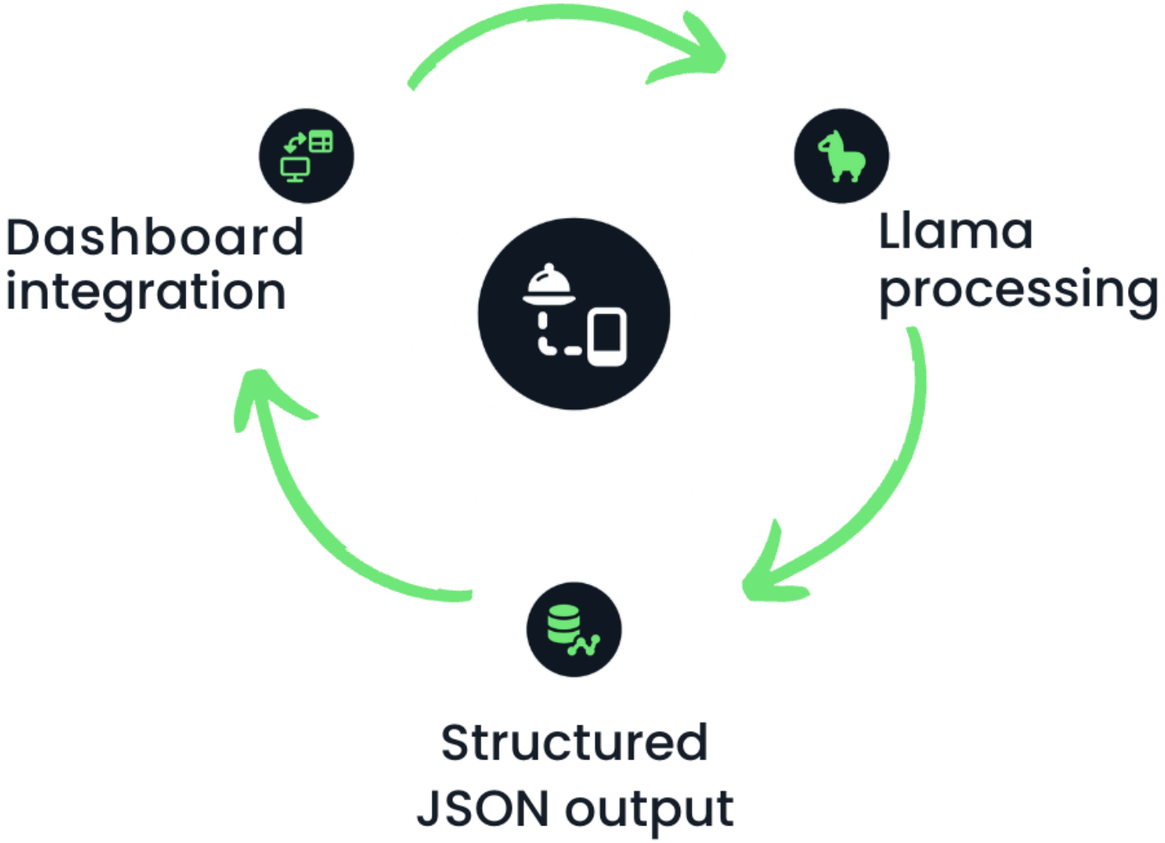
WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

datacamp

Structured output in JSON

- Example: Llama responses might be input to a dashboard
- Plain text responses won't work
- We need structured outputs



JSON responses with chat completion

```
response_format = {"type": "json_object"}


message_list = [
       {"role": "system", # System role defined as market analyst
        "content": "You are a food industry market analyst. You
         analyze sales data and generate structured JSON reports
         of top-selling beverages."},
       {"role": "user", # User role to pass the request
        "content": "Provide a structured report on the top-selling beverages
         this year."}
        ]
```

JSON responses with chat completion

```
output = llm.create_chat_completion(
          messages = message_list,
          response_format = "json_object"
)
```

- Response **format** specified as **JSON**
- Llama generates structured response, no free-flowing text

Extracting the JSON response

```
print(output['choices'][0]['message']['content'])
```

```
{
  "report_name": "Top-Selling Beverages 2024",
  "top_beverages": [
    {
      "rank": 1,
      "beverage_name": "Coca-Cola Classic",
      "sales_volume": 2.1,
      "growth_rate": 1.9
    },... ]
}
```

Defining a schema

```
response_format = {
"type": "json_object",
"schema": {
    "type": "object",
    "properties": {
            "Product Name": {"type": "string"},
            "Category": {"type": "string"},
            "Sales Growth": {"type": "float"}}
            }
}
```

- Can specify a **schema**: rules to define how the data should be formatted

Defining a schema

```
output = llm.create_chat_completion(
          messages = message_list,
          response_format = response_format)

print(output['choices'][0]['message']['content'])
```

```
{
  "Product Name": "Coca-Cola",
  "Category": "Soft Drink",
  "Sales Growth": 12.5
}
```

Let's practice!

WORKING WITH LLAMA 3

Building conversations

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

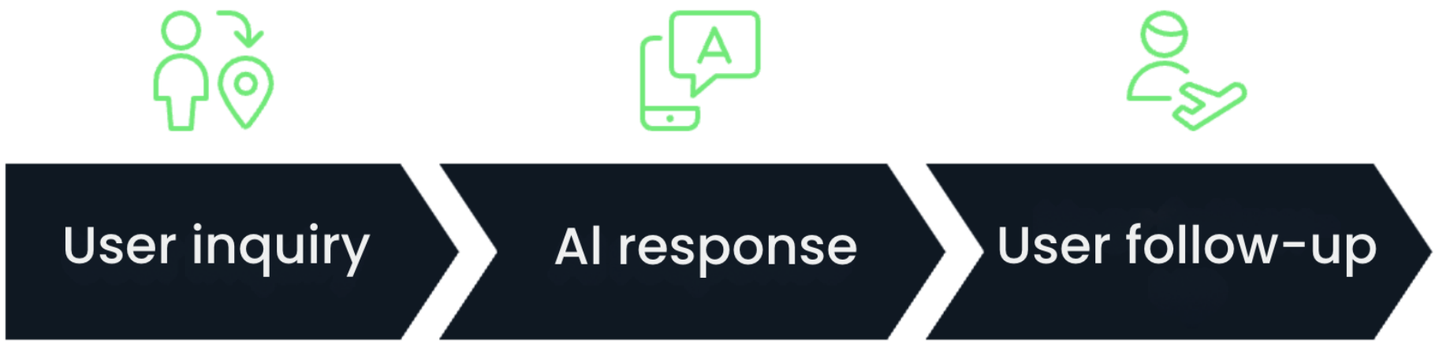
datacamp

Maintaining context

User inquiry

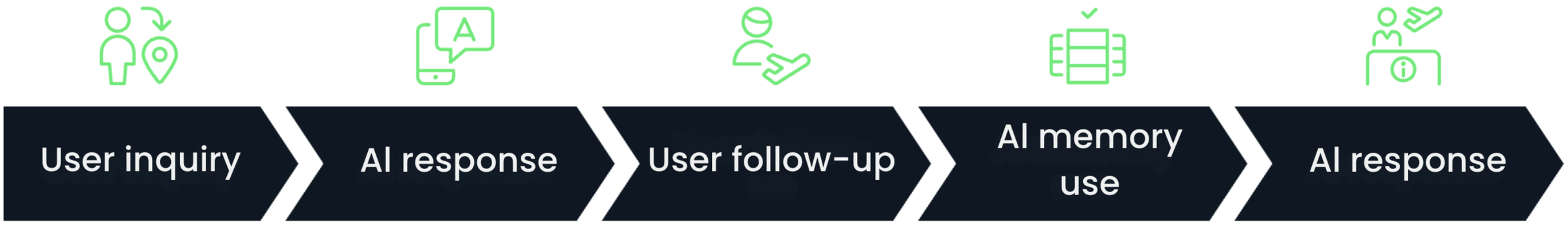
Maintaining context

Maintaining context



Maintaining context

Maintaining context



- Track a chat history with a `Conversation` class

Conversation class

- Can store a history of prior messages

```
class Conversation:
    def __init__(self, llm: Llama, system_prompt='', history=[]):
        self.llm = llm
        self.system_prompt = system_prompt
        self.history = [{"role": "system", "content": self.system_prompt}] + history
```

Conversation class

- Can store a history of prior messages

```
class Conversation:
    def __init__(self, llm: Llama, system_prompt='', history=[]):
        self.llm = llm
        self.system_prompt = system_prompt
        self.history = [{"role": "system", "content": self.system_prompt}] + history

    def create_completion(self, user_prompt=''):
        self.history.append({"role": "user", "content": user_prompt}) # Append input
        output = self.llm.create_chat_completion(messages=self.history)
        conversation_result = output['choices'][0]['message']
        self.history.append(conversation_result) # Append output
        return conversation_result['content'] # Return output
```

Running a multi-turn conversation

```
conversation = Conversation(llm, system_prompt="You are a virtual travel assistant
                                                helping with planning trips.")

response1 = conversation.create_completion("What are some destinations in France for a
                                           short weekend break?")
print(f"Response 1: {response1}")

response2 = conversation.create_completion("How about Spain?")

print(f"Response 2: {response2}")
```

Running a multi-turn conversation

```
print(f"Response 1: {response1}")


print(f"Response 2: {response2}")
```

```
Response 1: France is ideal for a short weekend break:
1.  **Paris**: The capital city is famous for its iconic landmarks like the ...
2.  **Provence**: Known for its beautiful landscapes, mild climate, and ...
Response 2:  Here are some destinations in Spain for a short weekend break:
1.  **Barcelona**: Visit the famous landmarks like the Sagrada Familia, ...
```

Running a multi-turn conversation

```
print(f"Response 1: {response1}")


print(f"Response 2: {response2}")
```

```
Response 1: France is ideal for a short weekend break:
1.  **Paris**: The capital city is famous for its iconic landmarks like the ...
2.  **Provence**: Known for its beautiful landscapes, mild climate, and ...
Response 2:  Here are some destinations in Spain for a short weekend break:
1.  **Barcelona**: Visit the famous landmarks like the Sagrada Familia, ...
```

Let's practice!

WORKING WITH LLAMA 3

Congratulations!

WORKING WITH LLAMA 3

Imtihan Ahmed
Machine Learning Engineer

datacamp

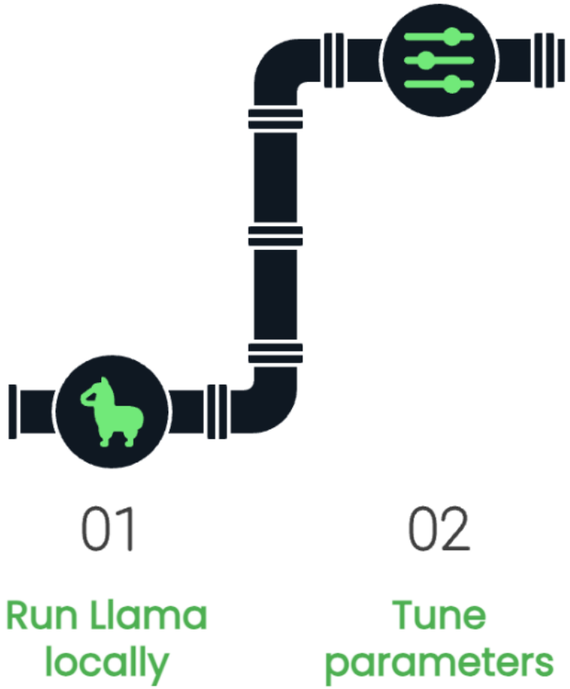
Let's recall

01

Run Llama locally

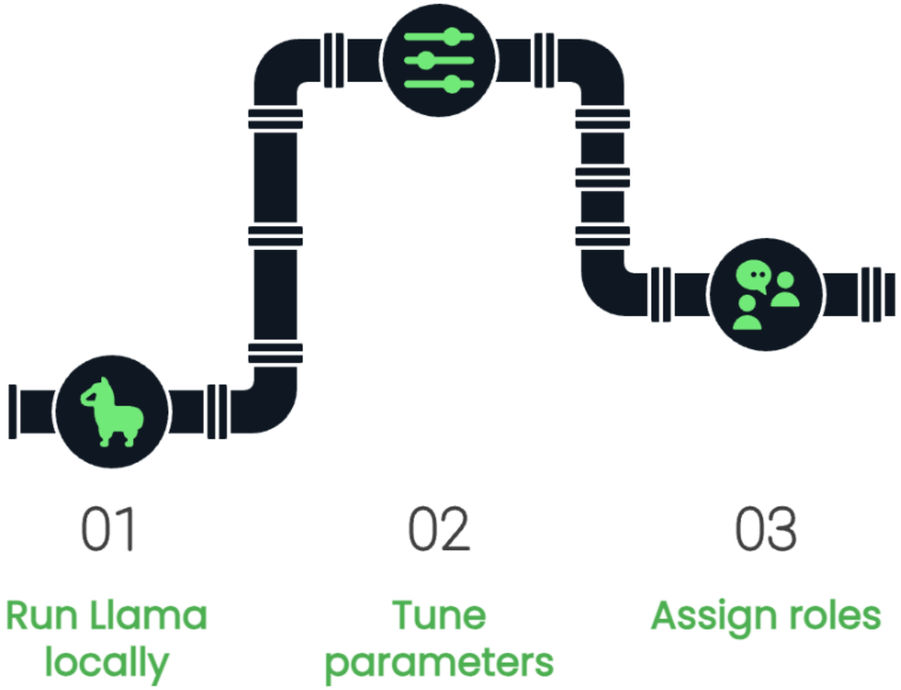
```
from llama_cpp import Llama


llm = Llama(model_path = "path/to/model.gguf")
```

Let's recall



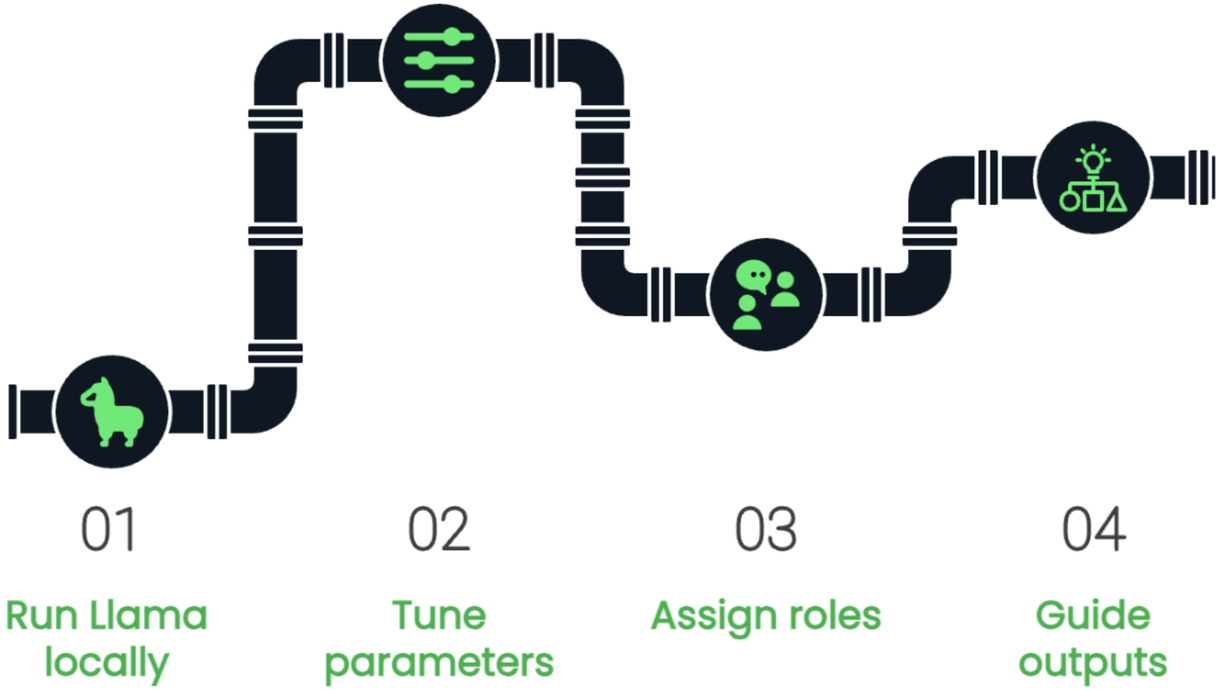
- `temperature`, `top_k`, `top_p` parameters

Let's recall



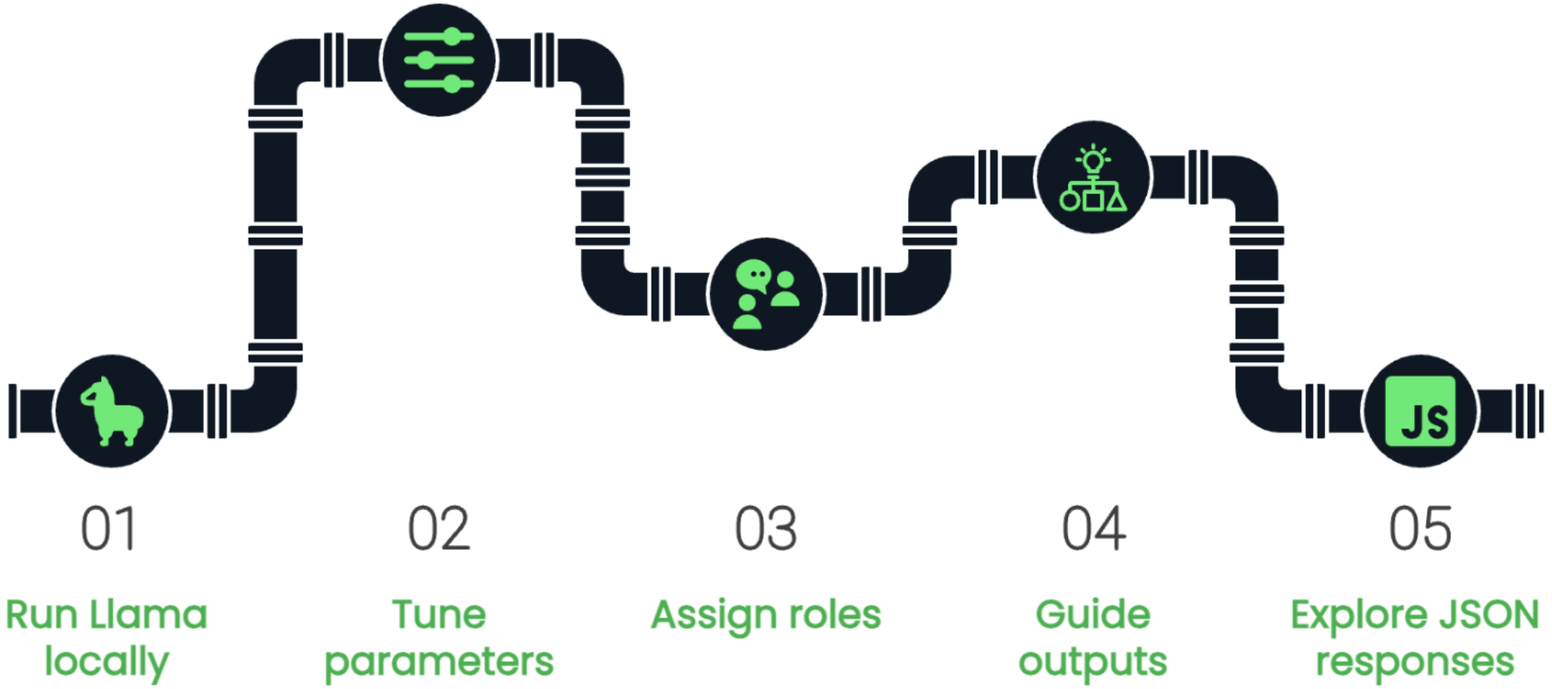

```
message_list = [{"role": "system", "content": system_message},
                {"role": "user", "content": user_message}]
```

Let's recall



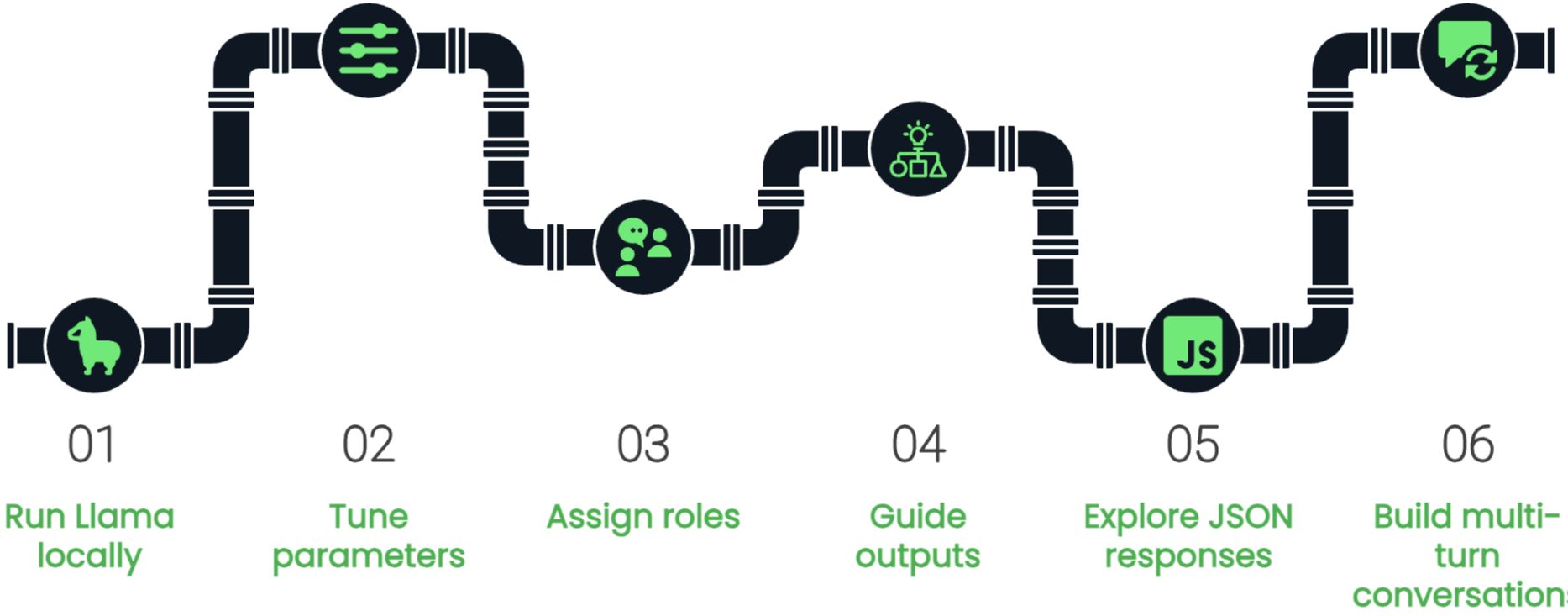
- Precise prompts
- `stop` words
- Zero-shot/Few-shot prompting

Let's recall



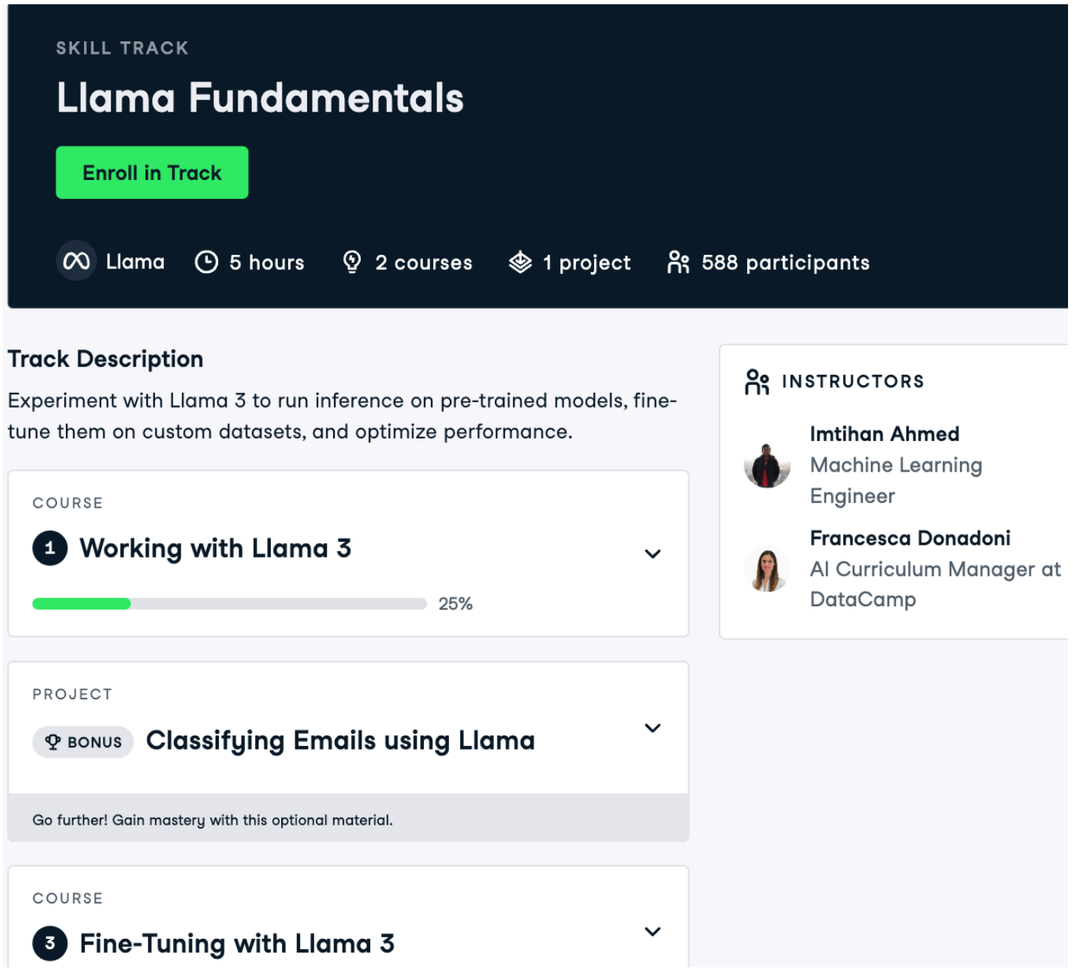

```
response_format = {"type": "json_object"}
```

Let's recall

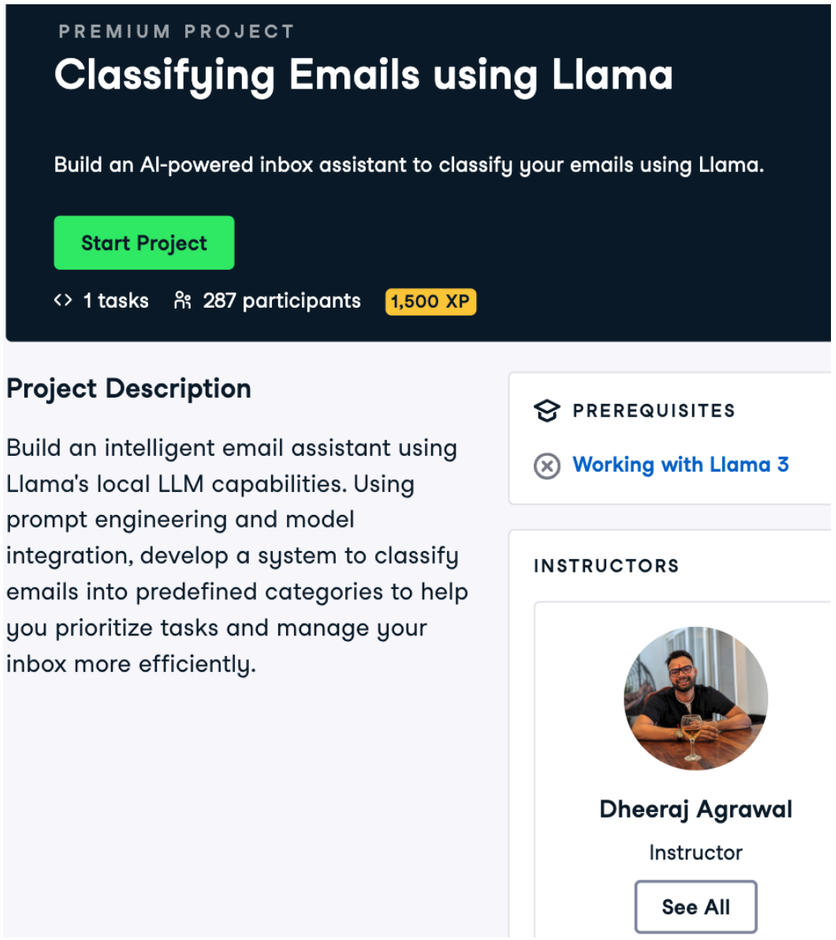


- `Conversation` class
- `.create_completion()` method

What's next?



Llama Fundamentals



Classifying Emails using Llama

Thank you!

WORKING WITH LLAMA 3