LangChain Expression Language (LCEL)

- Runnables support:
  - Async, Batch and Streaming Support
  - Fallbacks
  - Parallelism
    - LLM calls can be time consuming!
    - Any components that can be run in parallel are!
  - Logging is built in

Function Calling

- OpenAI has fine-tuned the `gpt-3.5-turbo-0613` and `gpt-4-0613` models to:

1. Accept additional arguments through which users can pass in descriptions of functions.
2. If it is relevant, return the name of the function to use, along with a JSON object with the appropriate input parameters.

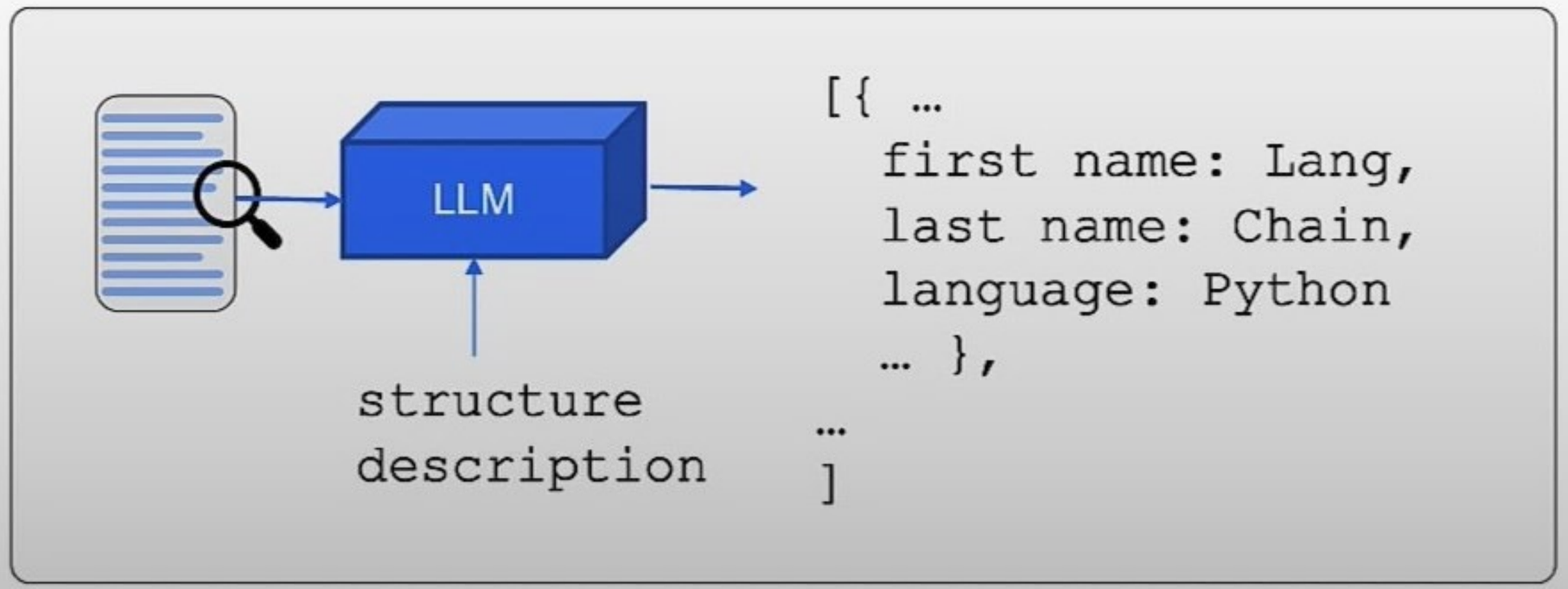
Interface

- Components implement "**Runnable**" protocol
- Common methods include:
  - `invoke [ainvoke]`
  - `stream [astream]`
  - `batch [abatch]`
- Common properties
  - `input_schema, output_schema`
- Common I/O

Component	Input Type	Output Type
Prompt	Dictionary	Prompt Value
Retriever	Single String	List of Documents
LLM	String, list of messages or Prompt Value	String
ChatModel	String, list of messages or Prompt Value	ChatMessage
Tool	String/Dictionary	Tool dependent
Output Parser	Output of LLM or ChatModel	Parser dependent

Extraction

- When given an input JSON schema, the LLM has been fine tuned to find and fill in the parameters of that schema.
- The capability is not limited to function schema.
- This can be used for general purpose extraction.



Tools and routing

- Functions and services an LLM can utilize to extend its capabilities are named “tools” in LangChain
- LangChain has many tools available
  - Search tools
  - Math tools
  - SQL tools
  - …

In this lab you will:

- Create your own tools
- Build a tool based on an OpenAPI spec
  - Predating LLMs, the OpenAPI specification is routinely used by service providers to describe their APIs
- Select from multiple possible tools – called ‘routing’

Agent Basics

- Agents
  - are a combination of LLMs and code
  - LLMs reason about what steps to take and call for actions.
- Agent loop
  - Choose a tool to use
  - Observe the output of the tool
  - Repeat until a stopping condition is met
- Stopping conditions can be:
  - LLM determined
  - Hardcoded rules

In this lab you will
- Build some tools
- Write your own agent loop using LCEL
- Utilize `agent_executor` which:
  - Implements the agent loop
  - Adds error handling, early stopping, tracing, etc