

# Let's Learn LangChain!



550 Vandalia St #231

Saint Paul, MN 55114

[lab651.com](https://lab651.com)

[recursiveawesome.com](https://recursiveawesome.com)

[justin@lab651.com](mailto:justin@lab651.com)

# About Me

- Founder, **Recursive Awesome & Lab651** – AI Consulting and Training & Software Development
- Co-founder, **Captovation** – AI powered online presentation coaching platform
- Owner, **Applied AI Weekly** – Publication covering AI
- Host of the “**Conversations on Applied AI Podcast**”
- Adjunct Professor, **University of Saint Thomas** – Teaching graduate courses on IoT & AI/ML
- Co-founder, **AppliedAI.MN** – 501(c)(3) non-profit: Monthly meetups, conferences, videos and podcasts on AI/ML



# Agenda

- What is an LLM
  - What are they good at?
  - What are they not so good at?
- What is LangChain
  - How does it work
  - Components
- LangChain in action
- Streamlit for easy deployment!

# LLMs and What Are They Good At?

A large language model is a type of artificial neural network that uses deep learning techniques to process, understand, and generate human-like text.

- **Content Generation:** Used for creating articles, reports, stories, and even poetry
- **Language Translation:** High-quality translations between numerous languages
- **Learning and Research:** Assist in educational and research activities by summarizing information
- **Conversation:** Human-like dialogue makes them useful for chatbots and virtual assistant

# What Are LLMs NOT Good At?



**You**

Who won the World Series in 2023?

< 2 / 2 >

**ChatGPT 3.5**



**ChatGPT**

I'm sorry, but I don't have access to information beyond my knowledge cutoff date of January 2022. Therefore, I cannot provide information on the winner of the 2023 World Series. I recommend checking a reliable sports news source or conducting an online search to find the most up-to-date information on the 2023 World Series winner.



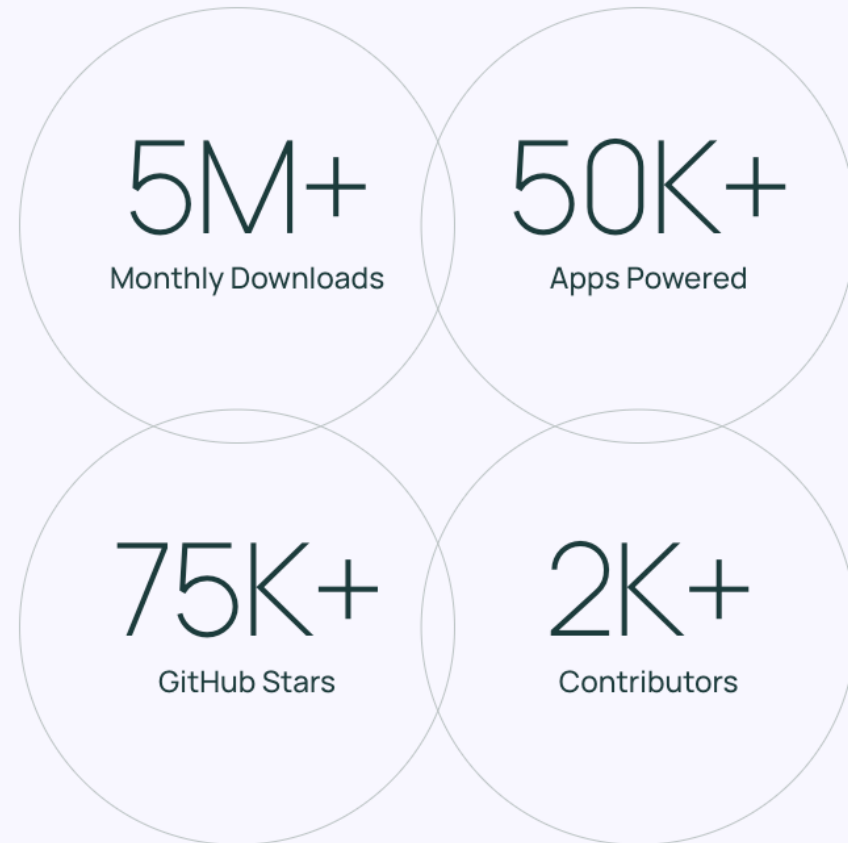
# What Are LLMs NOT Good At?

- **Fixed Knowledge Base:** Knowledge is static and limited to what was available when it was trained.
- **Understanding Context Depth:** Lack common sense reasoning, so they struggle with domain specific context. Prone to hallucinations!
- **Moral & Ethical Bias:** Can inadvertently perpetuate biases present in training data and lacks human emotions.
- **Conversational Continuity:** Generally don't maintain continuity over long conversations and lose track of previous conversations.
- **Dependency on User Input:** Relies heavily on the quality and clarity of user queries ( prompts ).

# LangChain – What Is it?

- Open source framework for developing applications powered by language models. Been around since October 2022.
- Enables applications to be:
  - Context Aware – Current events / documents, vocabulary, etc.
  - Reason – Custom functions, agents, memory / history, etc.
  - Consistency – Prompt templates
- Examples
  - Chatbots
  - Retrieval-Augmented Generation ( RAG )
  - Summarization
- Model agnostic

# Gaining in Popularity!



**LangChain lands \$25M round,  
launches platform to support  
entire LLM application lifecycle**



## The biggest developer community in GenAI

Learn alongside the 100K+ practitioners who are pushing the industry forward.

[Explore LangChain ↗](#)



# Making LLMs Context Aware

ChatGPT 4



**You**

Who won the world series in 2023?

< 4 / 4 >



**ChatGPT**

I did a [quick search](#) for more information and here's what I discovered.

The Texas Rangers won the 2023 World Series, securing their first championship in franchise history. They achieved this victory by defeating the Arizona Diamondbacks with a score of 5-0 in Game 5 of the series.



OBSERVABILITY



LangSmith

DEPLOYMENT



LangServe

Chains as Rest APIs



Templates

Reference Applications

APPLICATION



LangChain

Python

JavaScript

Chains, Agents, Agent Executors

Common Application Logic

INTEGRATIONS  
COMPONENTS

Models I/O

Model  
Prompt  
Example Selector  
Output Parser

Retrieval

Retriever  
Document Loader  
Vector Store  
Text Splitter  
Embedding Model

Agent Tooling

PROTOCOL

LCEL

Parallelization · Fallbacks · Tracing · Batching · Streaming · Async · Composition



Monitoring



Evaluation



Annotation



Feedback



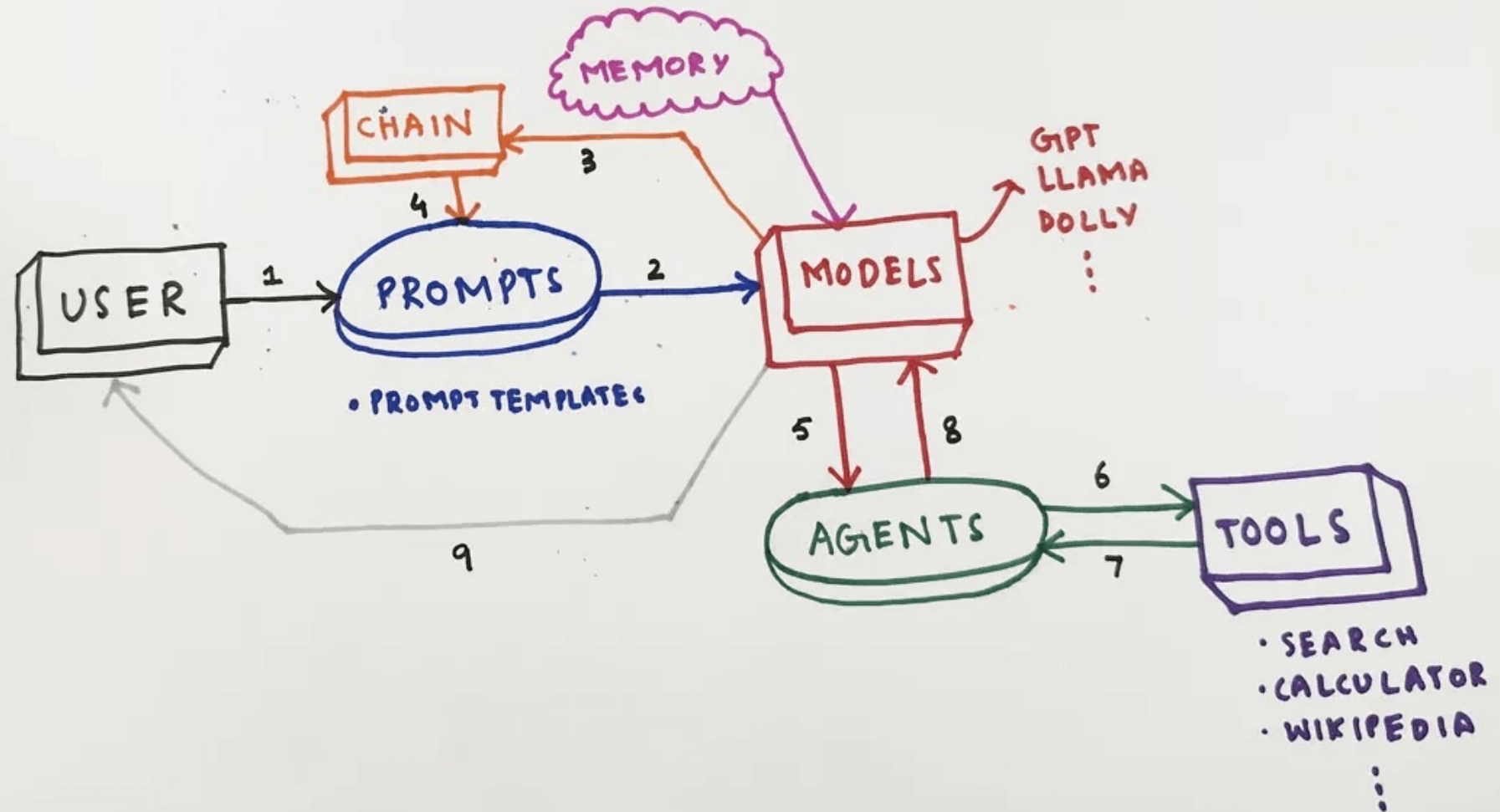
Testing



Debugging

# LANGCHAIN

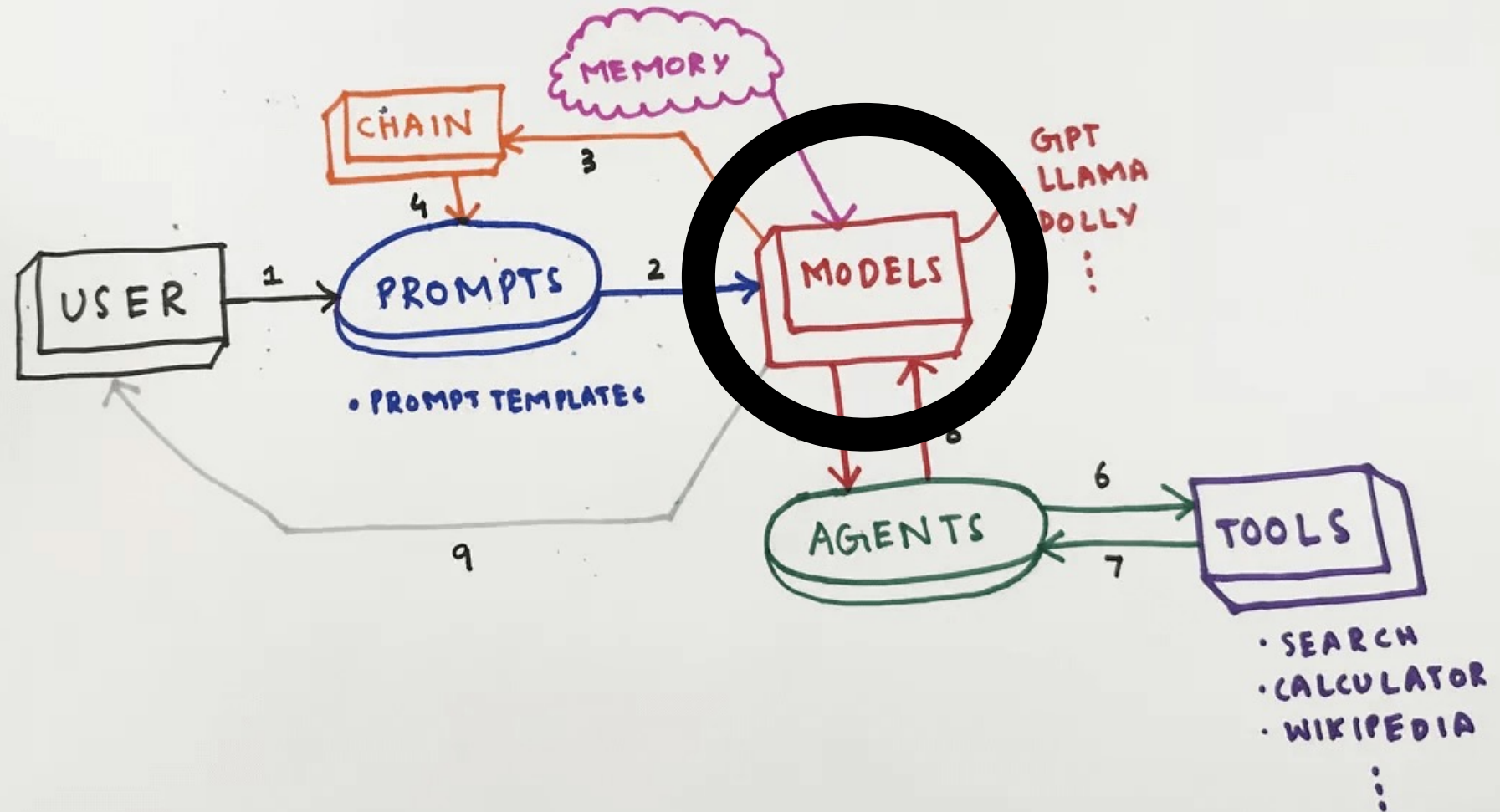
by BRILLIANT BYTES



Credit: <https://www.reddit.com/user/BrilliantBytes/>

# LANGCHAIN

by BRILLIANT BYTES



Let's start with the models!

# Models

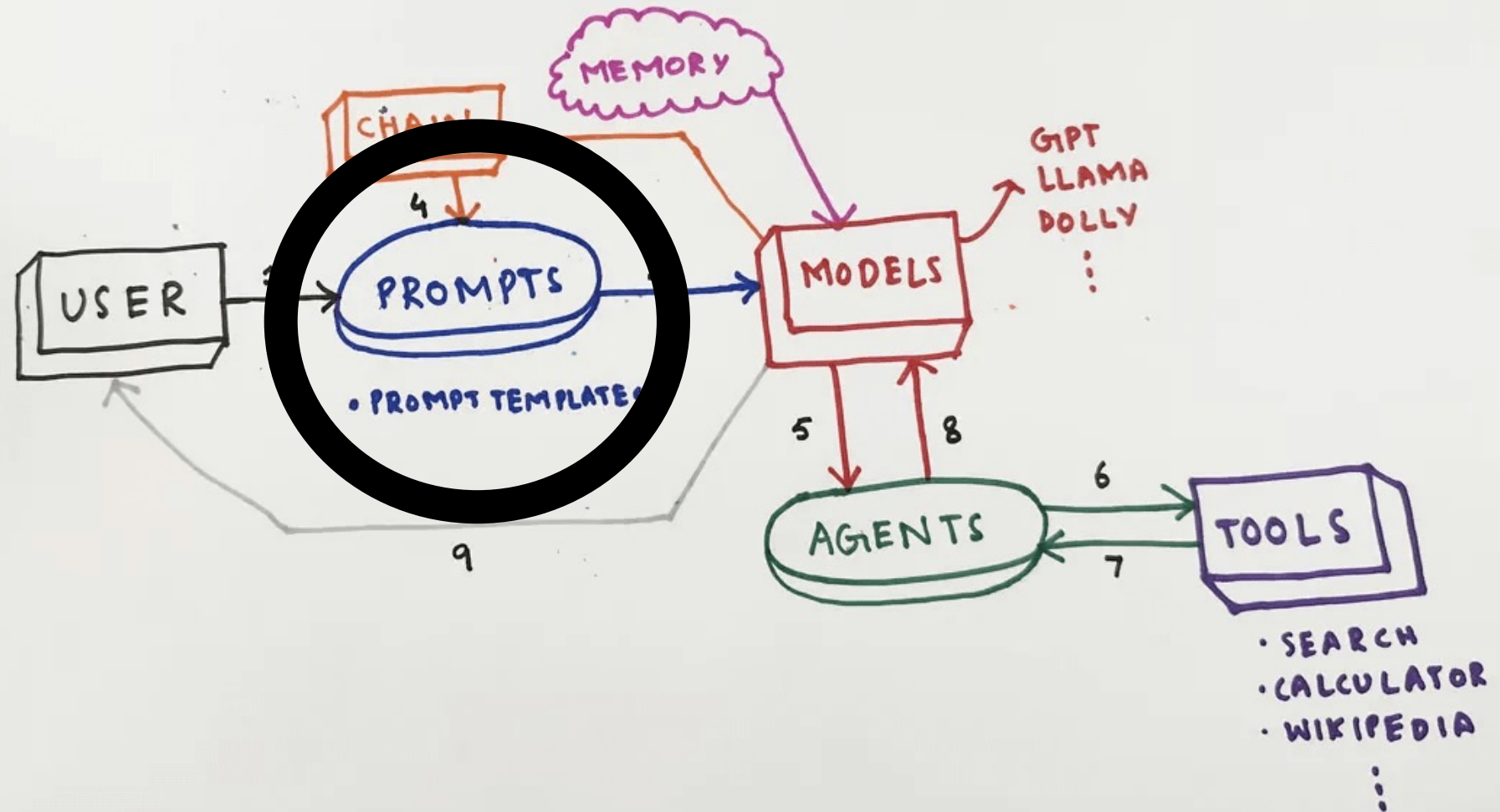
Model agnostic – OpenAI, Anthropic, Google and many others!

```
from langchain.chat_models.openai import ChatOpenAI
from langchain_community.chatmodels import ChatAnthropic
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatOpenAI(model_name='gpt-3.5-turbo', temperature=1)
llm = ChatAnthropic()
llm = ChatGoogleGenerativeAI(model_name='gemini-pro-vision')
```

# LANGCHAIN

by BRILLIANT BYTES



Next, Let's move to the prompts & prompt templates

# Prompts – Zero Shot

Zero-shot learning enables a model to make predictions about data it has never seen during training.

```
from langchain.chat_models.openai import ChatOpenAI

llm = ChatOpenAI(model_name='gpt-3.5-turbo', temperature=1,
max_tokens = 256)
prompt = "Generate for me a haiku about a dog"
print(llm.invoke(prompt))
```

```
Output: 'Loyal pup so sweet, Tail wagging, love complete,
Boundless joy we meet.'
```

# Prompts – Few Shot

Allows a model to learn from a very limited set of examples to apply this knowledge to similar tasks.

```
from langchain.chat_models.openai import ChatOpenAI

llm = ChatOpenAI(model_name='gpt-3.5-turbo', temperature=1,
max_tokens = 256)
prompt = "Complete given the equipment used for each sport
Tennis: Racquet
Baseball: Bat
Golf:"
print(llm.invoke(prompt))
```

Output: `Club`



# Prompts – Templates

Allows defining of prompts with variables to be passed in. Note: We defined our first chain in this example!

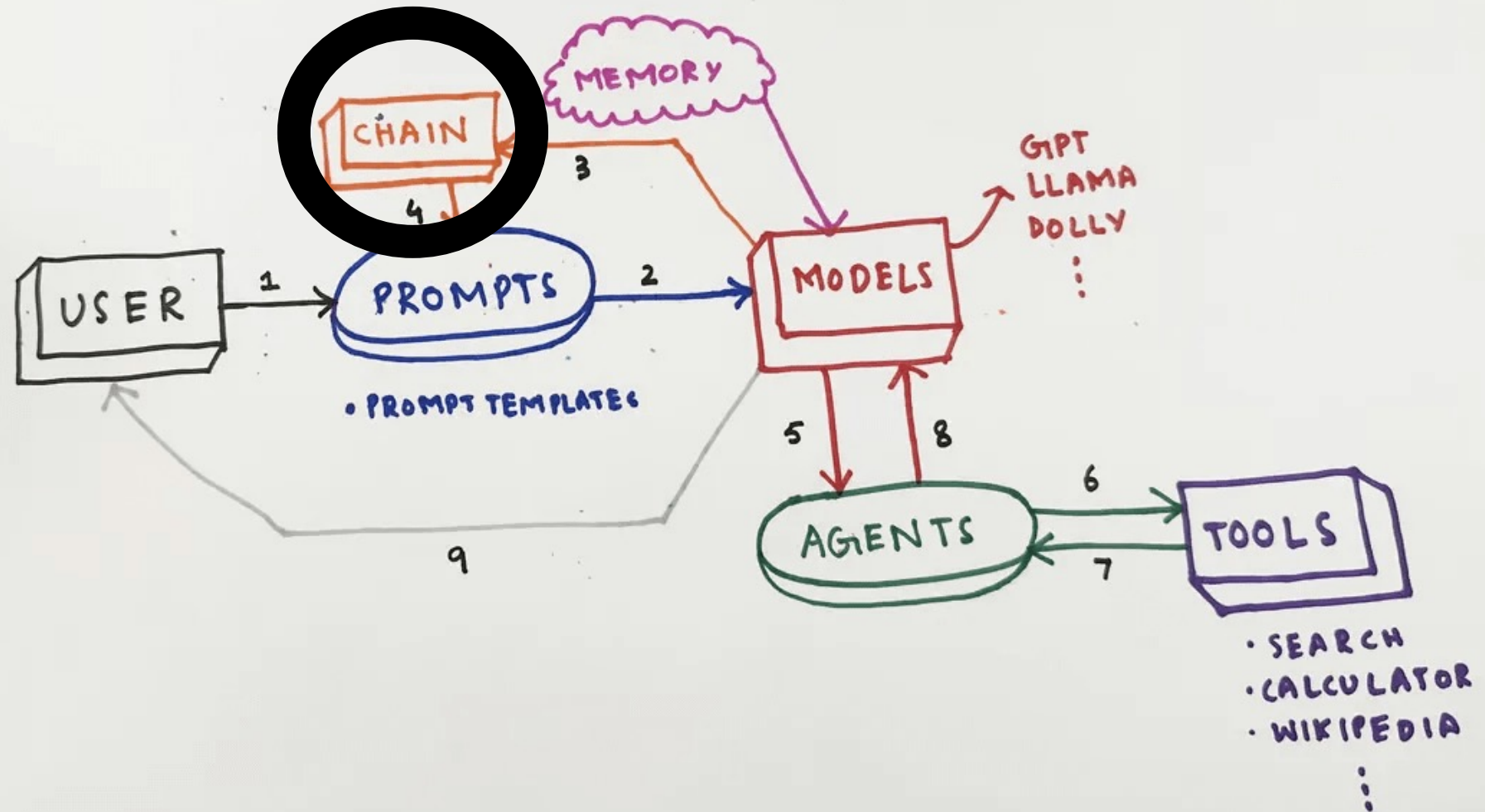
```
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain

prompt = PromptTemplate(
    input_variables=["city"],
    template="Describe a perfect day in {city}?",
)
llmchain = LLMChain(llm=llm, prompt=prompt)
llmchain.run("Paris")
```

Output: 'A perfect day in Paris would start with a leisurely breakfast at a charming café, sipping on a freshly brewed coffee and indulging in a buttery croissant...'

# LANGCHAIN

by BRILLIANT BYTES



Now the chains

# Chains – Simple Chain

Generating text based on topic and language inputs

```
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain

title_template = PromptTemplate(
    input_variables=["topic", "language"],
    template="Create the title of blog post on {topic} in {language}")

title_chain = LLMChain(llm=llm, prompt=title_template)
title_chain.run({'topic': 'the best way to learn a language',
                'language': 'french'})

"Le Meilleur Moyen d\'Apprendre une Langue : Guide Complet en Français"
```

# Chains – Sequential Chains

```
title_template = PromptTemplate(
    input_variables=["topic"],
    template="Create the title for a blog post on {topic}")

article_template = PromptTemplate(
    input_variables=["title"],
    template="Create a blog post on the topic of {title}")

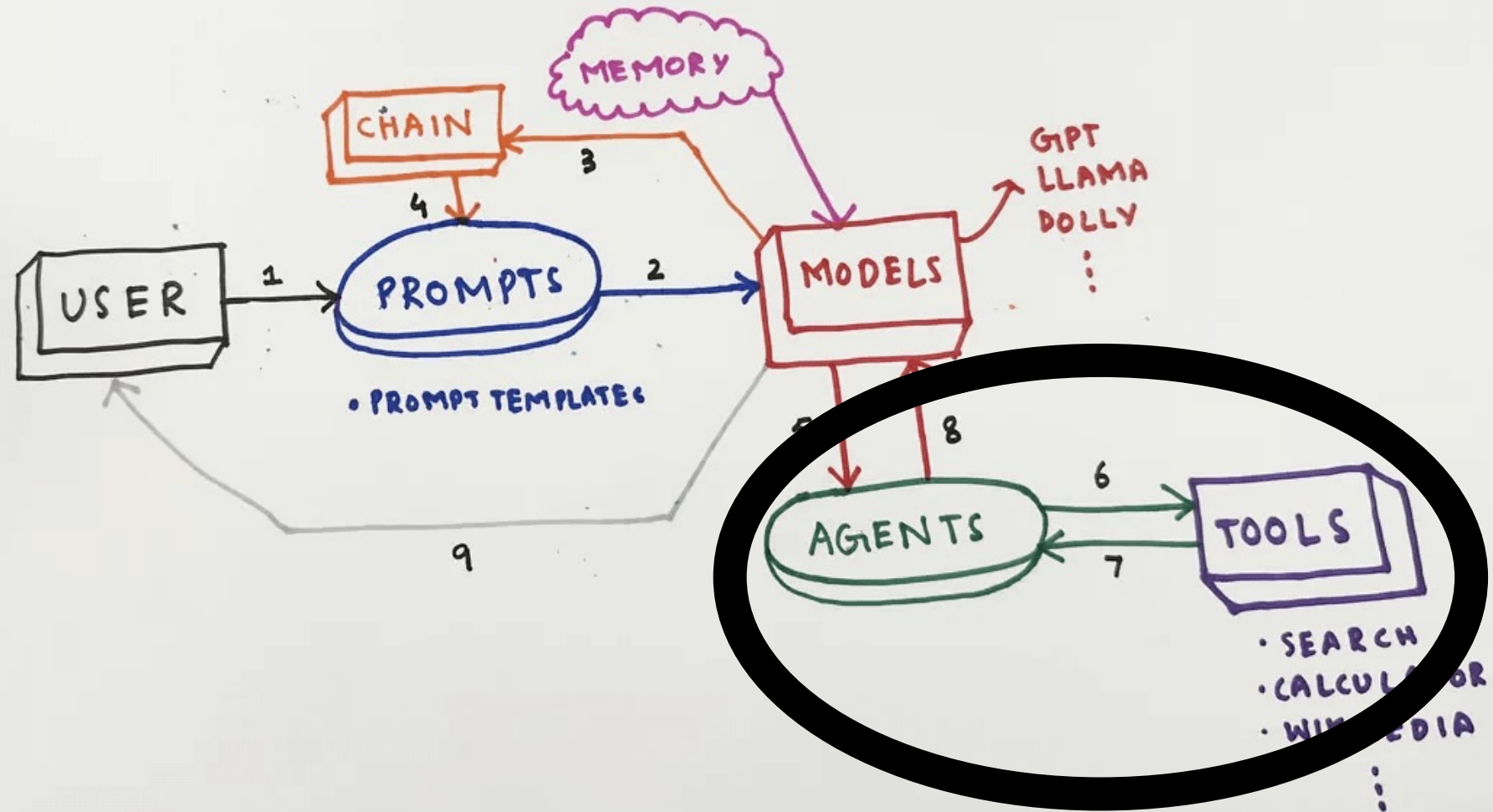
title_chain = LLMChain(llm=llm, prompt=title_template)
article_chain = LLMChain(llm=llm2, prompt=article_template)

overall_chain = SimpleSequentialChain(chains=[title_chain,
    article_chain], verbose=True)
overall_chain.run('How to run a marathon')
```

***"Title: Mastering the Marathon: A Comprehensive Guide to Training and Finishing Strong – Introduction: Running a marathon is a remarkable achievement that requires dedication, perseverance..."***

# LANGCHAIN

by BRILLIANT BYTES



Finally, the agents and the tools


# Agents & Tools


- Think of an agent as the “conductors” of an orchestra
  - Agents leverages the LLM for the next action
  - Prompt: “What the current age of the president divided by 2?”


```
> Entering new AgentExecutor chain...  
  I should use a search engine to find the current age of the United States President.  
Action: Search  
Action Input: "United States President current age"  
Observation: 81 years  
Thought: I should use a calculator to divide the current age by 2.  
Action: Calculator  
Action Input: 81 / 2  
Observation: Answer: 40.5  
Thought: I now know the final answer.  
Final Answer: The United States President's current age divided by 2 is 40.5 years.  
  
> Finished chain.
```


## Tools


# Need something done? There's a **tool** for that!


**Alpha Vantage**  
Alpha Vantage Alpha Vantage provides


**Apify**  
This notebook shows how to use the [Apify


**Requests**  
The web contains a lot of information that LLMs do not have acce...


**SceneXplain**  
SceneXplain is an ImageCaptioning service


**ArXiv**  
This notebook goes over how to u


**AWS Lambda**  
This notebook goes over how to use the AWS Lambda component to ...


**Search Tools**  
This notebook shows off usage of various search tools.


**SearchApi**  
This notebook shows examples of how to use SearchApi to searc...


**Shell (bash)**  
Giving agents access to the shell


**Eleven Labs Text2Speech**  
This notebook shows how to interact with the ElevenLabs API to ...


**File System**  
LangChain provides tools for interacting with .


**SearxNG Search**  
This notebook goes over how to use a self hosted SearxNG searc...


**Semantic Scholar API Tool**  
This notebook demos how to use the semantic scholar tool with a...


**Bing Search**  
Microsoft Bing, commonly referre


**Golden Query**  
Golden provides a set of natural language APIs


**Google Cloud Text-to-Speech**  
Google Cloud Text-to-Speech


**SerpAPI**  
This notebook goes over how to use the SerpAPI component to s...


**StackExchange**  
This notebook goes over how to use the stack exchange compon...


**ChatGPT Plugins**  
This example shows how to use C


**Google Drive**  
This notebook walks through connecting a LangChain to the


**Google Finance**  
This notebook goes over how to use the Goog


**Tavily Search**  
Tavily's Search API is a search engine built


**Twilio**  
This notebook goes over how to use the Twilio


**Google Jobs**  
This notebook goes over how to use the Google Jobs Tool to fetc...


**Google Lens**  
This notebook goes over how to use the Goog


**Wikipedia**  
Wikipedia is a multilingual free online

**Wolfram Alpha**  
This notebook goes over how to use the wolfram alpha component.

**Google Places**  
This notebook goes through how to use Google Places API

**Google Scholar**  
This notebook goes through how to use Google Scholar tool

**Yahoo Finance News**  
This notebook goes over how to use the yahoofinancenews tool ...

**YouTube**  
YouTube Search package

Let's See Some  
Examples!





# Streamlit

# A faster way to build and share data apps

Streamlit turns data scripts into shareable web apps in minutes.

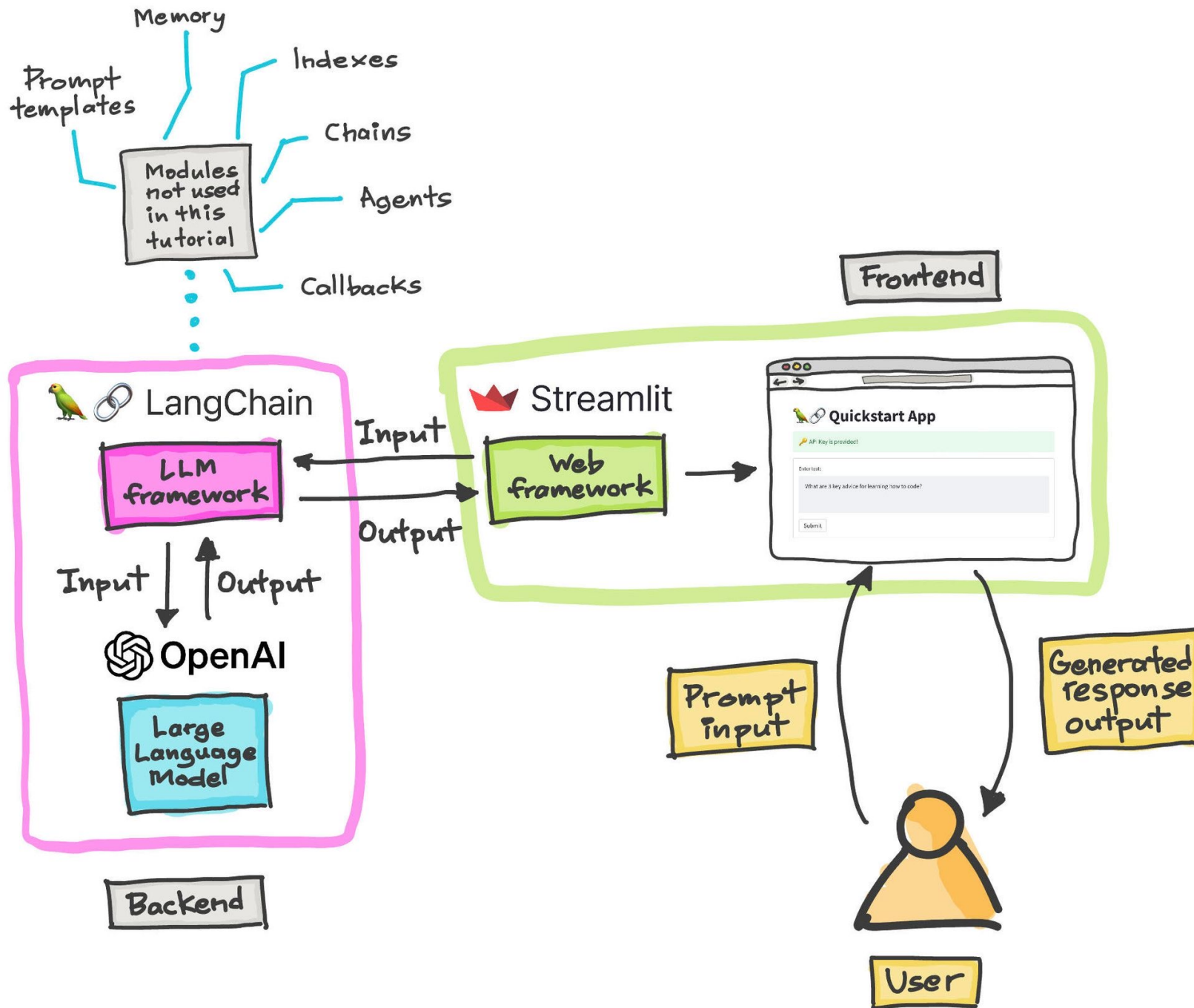
All in pure Python. No front-end experience required.

[Try Streamlit now](#)

[Deploy on Community Cloud \(it's free!\)](#)



**My Favorite  
Text Editor**



# Thank You

Justin Grammens  
Founder + CEO



**SOURCE CODE**

[justin@recursiveawesome.com](mailto:justin@recursiveawesome.com)



Please reach out. I would love to connect and continue the conversation on LangChain!