* Venv setup
  + Mac
    - python3 -m venv venv
    - source ./venv/bin/activate
  + Windows
    - python -m venv venv
    - venv\Scripts\activate
* LangChain Intro
  + <https://python.langchain.com/docs/introduction>
  + <https://python.langchain.com/docs/integrations/chat/>
* Setup Groq Key
  + <https://groq.com/>
  + Create key - <https://console.groq.com/keys>
  + pip install -r requirements.txt
  + https://console.groq.com/docs/models

import os

from dotenv import load\_dotenv

load\_dotenv()

* LangGraph basics
  + <https://langchain-ai.github.io/langgraph/concepts/low_level>
* LangGraph Chatbot
  + <https://langchain-ai.github.io/langgraph/tutorials/get-started/1-build-basic-chatbot/>

try:

img = graph.get\_graph().draw\_mermaid\_png()

with open("graph.png", "wb") as f:

f.write(img)

except Exception:

pass

* Prebuilt Agents with tools, memory and structured output
  + [https://langchain-ai.github.io/langgraph/agents/agents](https://langchain-ai.github.io/langgraph/agents/agents/)

import os

from dotenv import load\_dotenv

load\_dotenv()

from langgraph.prebuilt import create\_react\_agent

def get\_weather(city: str) -> str:

"""Get weather for a given city."""

return f"It's always sunny in {city}!"

agent = create\_react\_agent(

model="groq:llama-3.3-70b-versatile",

tools=[],

prompt="You are a helpful assistant"

)

# Run the agent

response = agent.invoke(

{"messages": [{"role": "user", "content": "what are large language models"}]}

)

print(response)

* Pydantic basics
  + <https://docs.pydantic.dev/latest/concepts/models/>
* MCP
  + <https://modelcontextprotocol.io/docs/getting-started/intro>
  + <https://modelcontextprotocol.io/docs/learn/architecture>
  + <https://langchain-ai.github.io/langgraph/agents/mcp/>
* LangChain MCP Adapters
  + <https://changelog.langchain.com/announcements/mcp-adapters-for-langchain-and-langgraph>
  + <https://github.com/langchain-ai/langchain-mcp-adapters>
* GitHub MCP Server - <https://github.com/modelcontextprotocol/servers-archived/tree/main/src/github>
  + GitHub Fine Grained Token - <https://github.com/settings/personal-access-tokens>
* Filesystem MCP Server - <https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem>
* Create Custom MCP Server - <https://langchain-ai.github.io/langgraph/agents/mcp/#custom-mcp-servers>

Day 2

* Streamlit
  + Playground - <https://streamlit.io/playground>
  + Chat elements - <https://docs.streamlit.io/develop/api-reference/chat>
  + Session state - <https://docs.streamlit.io/develop/concepts/architecture/session-state>
* MultiModel Models
  + Gemini API Key - <https://aistudio.google.com/apikey>
  + Gemini 2.0 Flash - <https://ai.google.dev/gemini-api/docs/models#gemini-2.0-flash>
  + Text to Image Generation - <https://ai.google.dev/gemini-api/docs/image-generation>
  + Image Caption Generation - <https://ai.google.dev/gemini-api/docs/image-understanding>
  + Video Caption Generation - <https://ai.google.dev/gemini-api/docs/video-understanding>
* RAG -
  + With prebuilt agent - <https://python.langchain.com/docs/tutorials/qa_chat_history/>
  + Embedding Models - <https://python.langchain.com/docs/concepts/embedding_models/>
  + Gemini API Key - <https://aistudio.google.com/apikey>
* Multi-Agent Architectures - <https://langchain-ai.github.io/langgraph/agents/multi-agent>
  + LangGraph Supervisor - <https://github.com/langchain-ai/langgraph-supervisor-py>
    - Demo Code GitHub - <https://github.com/keertipurswani/MachineLearningTutorials/blob/main/MCP/multi_agent_demo.py>
  + LangGraph Swarm - <https://github.com/langchain-ai/langgraph-swarm-py>
* Ollama -
  + <https://ollama.com>
  + LangChain ChatModel - <https://python.langchain.com/docs/integrations/chat/ollama/>
* HuggingFace -
  + <https://huggingface.co>
  + <https://python.langchain.com/docs/integrations/providers/huggingface>/