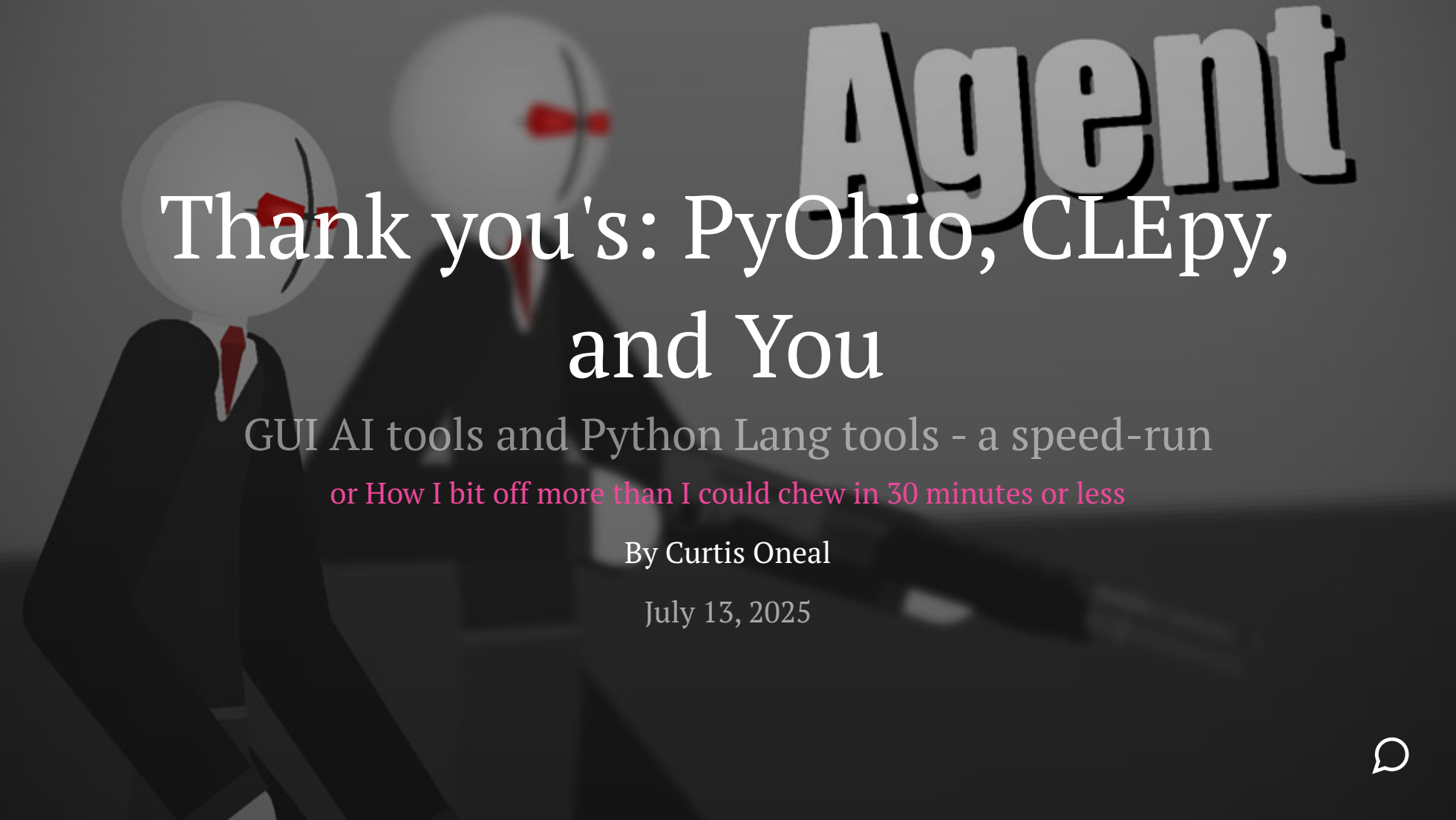

A Very Brief Overview of: Python’s Lang AI Tools and Two Low-Code, GUI, AI Tools - Flowise and N8N
July 27, 20205
Sunday at 2:45 pm in Ballroom C



Thank you's: PyOhio, CLEpy, and You

GUI AI tools and Python Lang tools - a speed-run

or How I bit off more than I could chew in 30 minutes or less

By Curtis Oneal

July 13, 2025

Introduction

1. This talk is intended for intermediate Pythonista and the AI curious person who has wondered about AI and also deployment. After all this is a Python Conference and you are here.

2. I make a lot of assumptions about your interest, your llm exposure and willingness to dig into documentation or tutorials.

3. This talk is not a code walk-through, but more of a tourist map of some of the tools you could use.

4. It bridges the conversation between low-code AI tools and some of Python's Lang-named ecosystem.

You'll see how to prototype quickly with:

- Flowise
- n8n
- We'll transition to scalable Python implementations using: *LangChain, *Langflow,*LangGraph, *LangSmith. We'll wave as we pass by at a few other Lang' tools.
- We'll explore some practical challenges along the way.

My Background

- My AI, Data Engineering, and coding background is largely based in Medical Metrics at Enterprise scale
  - Watson Health
  - Arbormetrics (Ann Arbor MI)
  - Arkos Health (AZ)
  - BlueSkyAI (NC)
- I have past experience with some low-ish code tooling that hides the logic
  - Nifi
  - Mulesoft Dataweave (Salesforce)
  - Mirth Connect
  - Talend Open Studio
- And huge integration systems where each touch-point is its own specific adaptation.

My Initial Approach's Bias

- Don't hide the logic
- Transparent diffs
- Git commits and change tracking
- Cloud scaling/High Availability(AWS|GCP|Azure)
- Infrastructure as code (e.g.,Terraform, Ansible)
- Automated deployments

Danger of Confirmation Bias

- While approaching the topic I'm open to other evidence that these GUI tools have capabilities that can satisfy the above needs.
- Limited research on the topic shows N8N and Flowise in production at other companies.

Most Commonly talked about LLM Architectural Patterns

1. **ReAct (Reasoning and Acting)**

LLM reasons, plans actions, executes tools, observes results, iteratively refines

2. **Tool-Calling Agent**

LLM equipped with external tools (APIs, databases, calculators) for enhanced capabilities

3. **Agent Supervising Agent (Hierarchical)**

Supervisor agent delegates tasks to specialized sub-agents, coordinates results

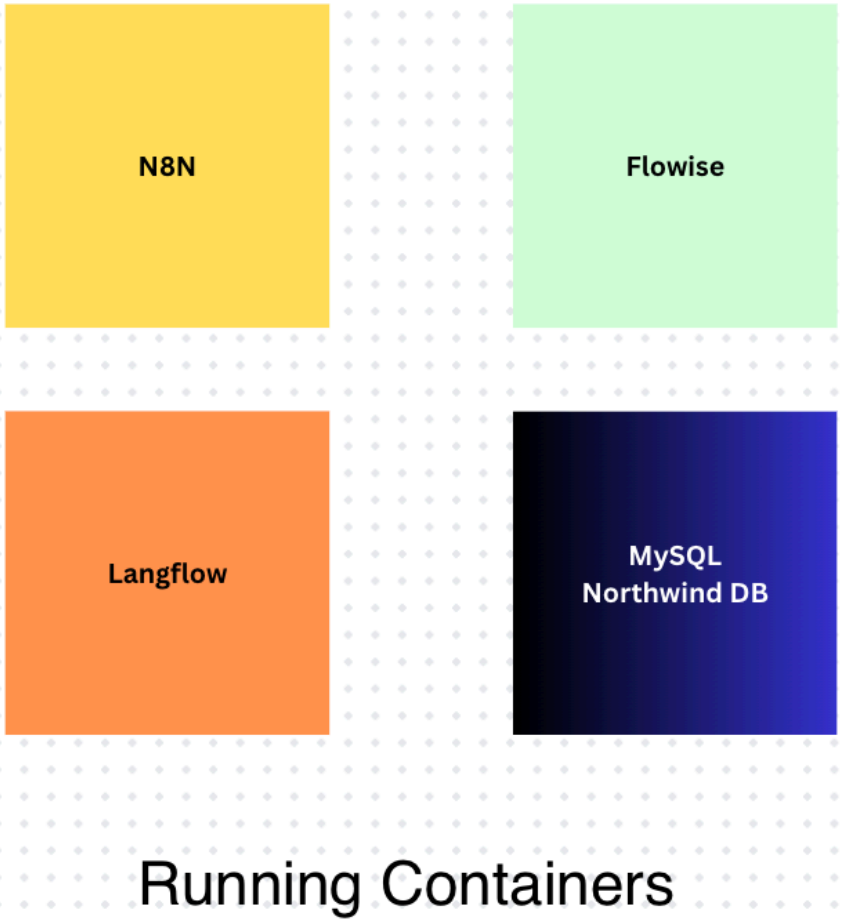
4. **Multi-Agent Collaboration**

Peer agents communicate and work together toward shared goals

5. **Retrieval-Augmented Generation (RAG)**

"A Few" AI SDKs

- LangChain - open-source framework for LLM applications with modular components, RAG capabilities, and memory for building chatbots and AI agents
- **Google AI SDK/Vertex AI** - Google's platform with Gemini models, document summarization, unified API access, and fully-managed ML deployment
- **LlamaIndex** - Specialized framework for connecting LLMs to diverse data sources with powerful indexing, querying, and retrieval-augmented generation
- **OpenAI SDK** - Direct API access to GPT models providing text generation, embeddings, and fine-tuning capabilities for developers
- Low-Code/No-Code Platforms - n8n (workflow automation), Flowise (visual LLM orchestration) democratizing AI development for non-technical users
- **Multi-Agent Frameworks - CrewAI** (role-playing agents), AutoGen (cooperative agents) enabling sophisticated AI team coordination and task distribution
- **Specialized Provider SDKs** - Cohere (NLP focus), AI21 Labs (creative writing), Anthropic Claude (ethical AI), each offering unique strengths and model characteristics
- **etc** ...(word salad)



GUI Tool-sets Advantages

Key Takeaways:

- Rapid prototyping advantages
- Visualization of complex workflows
- Learning deeper concepts without getting lost in code
- Applying complex concepts staying at a high-level of strategy

Note:
N8N is Typescript under the hood.
Flowise is Langchain Typescript under the hood.

Platform Comparison

Feature	N8N	Flowise	LangFlow	LangChain & Graph
Visual Interface	✅ Node-based	✅ AgentFlow V2	✅ Component-based	❌ Code-only
Learning Curve	Low	Medium	Medium	High
Customization	Medium	High	High	Highest
Error Handling	Good	Good	Good	Excellent
Scalability	Good	Good	Good	Excellent
Deployment	Easy	Easy	Easy	Flexible

This was a Claude.ai comparison. I find Flowise UI easier to navigate, but N8N easier to troubleshoot.

Flowise

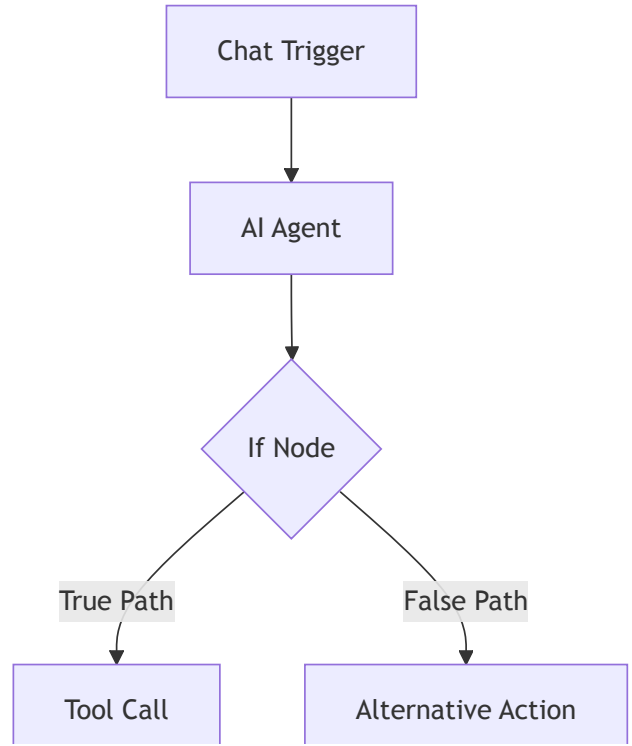
- GUI demonstration: Creating a basic chatbot
- Key components: nodes, connections, interfaces
- Live 'demo': Building a simple assistant
- If you have Node and NPM or PNPM:

Shortcut to running local

Getting Started

Official Tutorials

Leon Van Zyl




```
# Install flowise with NodeJS
npm install -g flowise
# Specific version
npm install -g flowise@x.x.x
# Start Flowise:
npx flowise start
```

My Interactive Slide

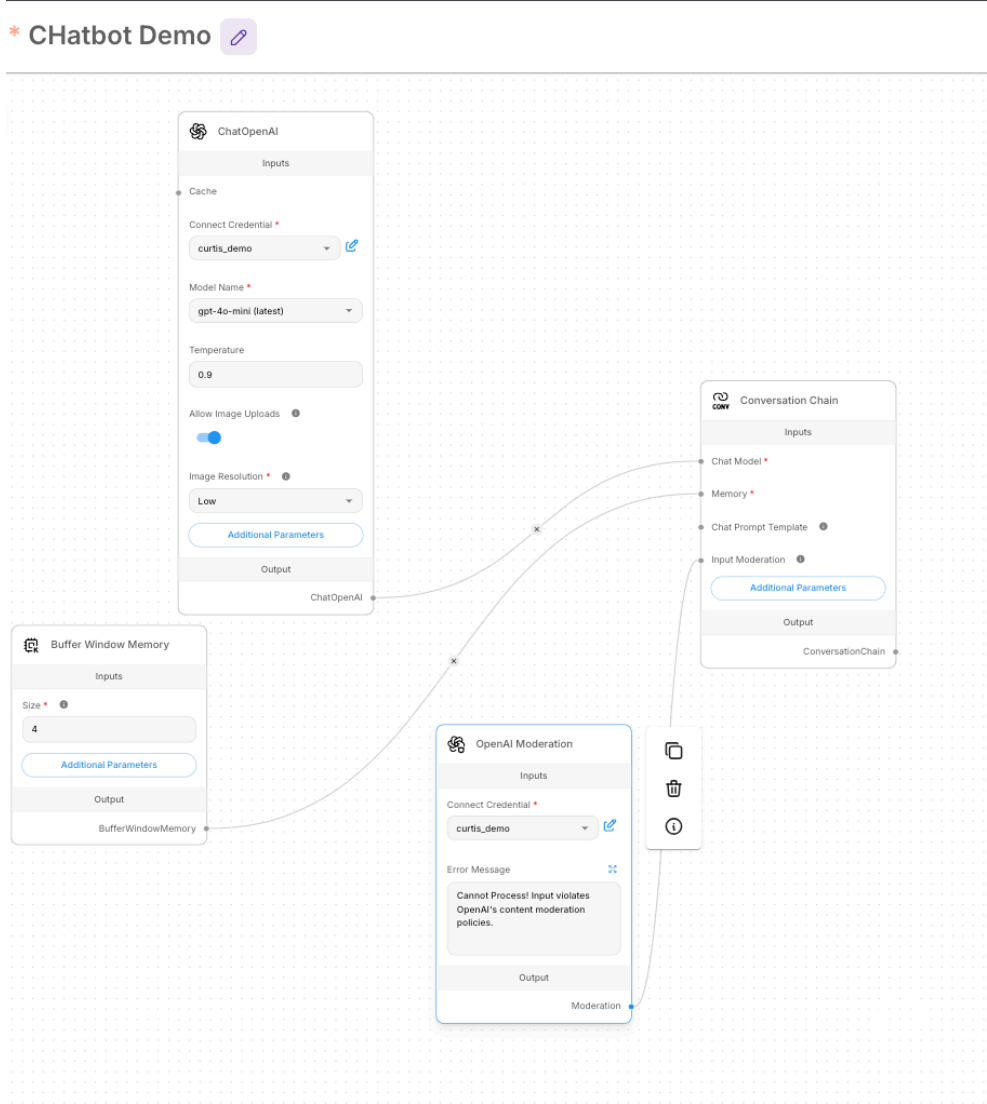
(Click on the blue bubble below.)

Flowise Additional Patterns

- API key/Secret Organization is Straightforward
- Naive RAG is fairly easy
- Tool Use can be easy
- Multi-agent interaction is simple
- Complex model understanding is easy

Flowise Areas of Concern

- Self hosting in cloud is challenging
- Documentation lags
- Requires a database for persistence of configurations
  - Leads to problematic scaling designs
  - This makes it brittle
- Path to reload it as code is difficult



N8N

- GUI demonstration: Creating a basic Workflow
- Key components: nodes, connections, Triggers, Actions
- Live 'demo': Building a simple Flow
- AI bots are doable, start small
- Using Docker

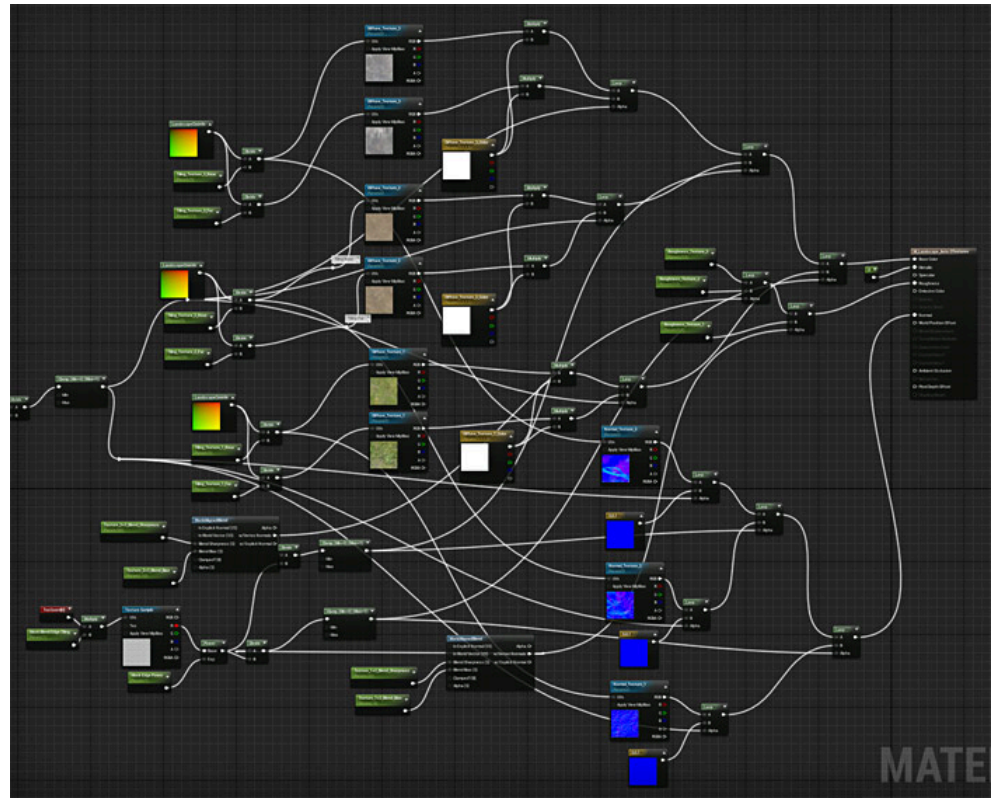
```
# Set up a dedicated space for N8N
docker volume create n8n_data
# This complex set of commands will pull a docker
# image set it up with and serve it locally, and
# mount a docker volume to persist your data locally
docker run \
  -it \
  --rm \
  --name n8n \
  -p 5678:5678 \
  -v n8n_data:/home/node/.n8n \
  docker.n8n.io/n8nio/n8n
```

Docs

- Shortcut to my running N8N
- N8N Installation guide Docker
- N8N Installation guide NPM NodeJS
- Very Quick start
- 15 Minute Tutorial video

Jump to:

https://community.n8n.io/t/re-route-nodes-connections-for-complex-workflows/62003



Shifting back toward python

Decorators

Decorators are first class objects in python, that can get deep quickly.

The can be used on both functions and classes, and allow you to modify the behavior of the code without actually modifying it.

In Python they have an @ symbol and name and go in front of a function definition.

They change the behavior of the function, and glossing over detail level are used in essentially pass functions to and through one another keeping their state, and keeping track of why you are doing it.

The one you've most likely seen is the `@asyc` decorator.

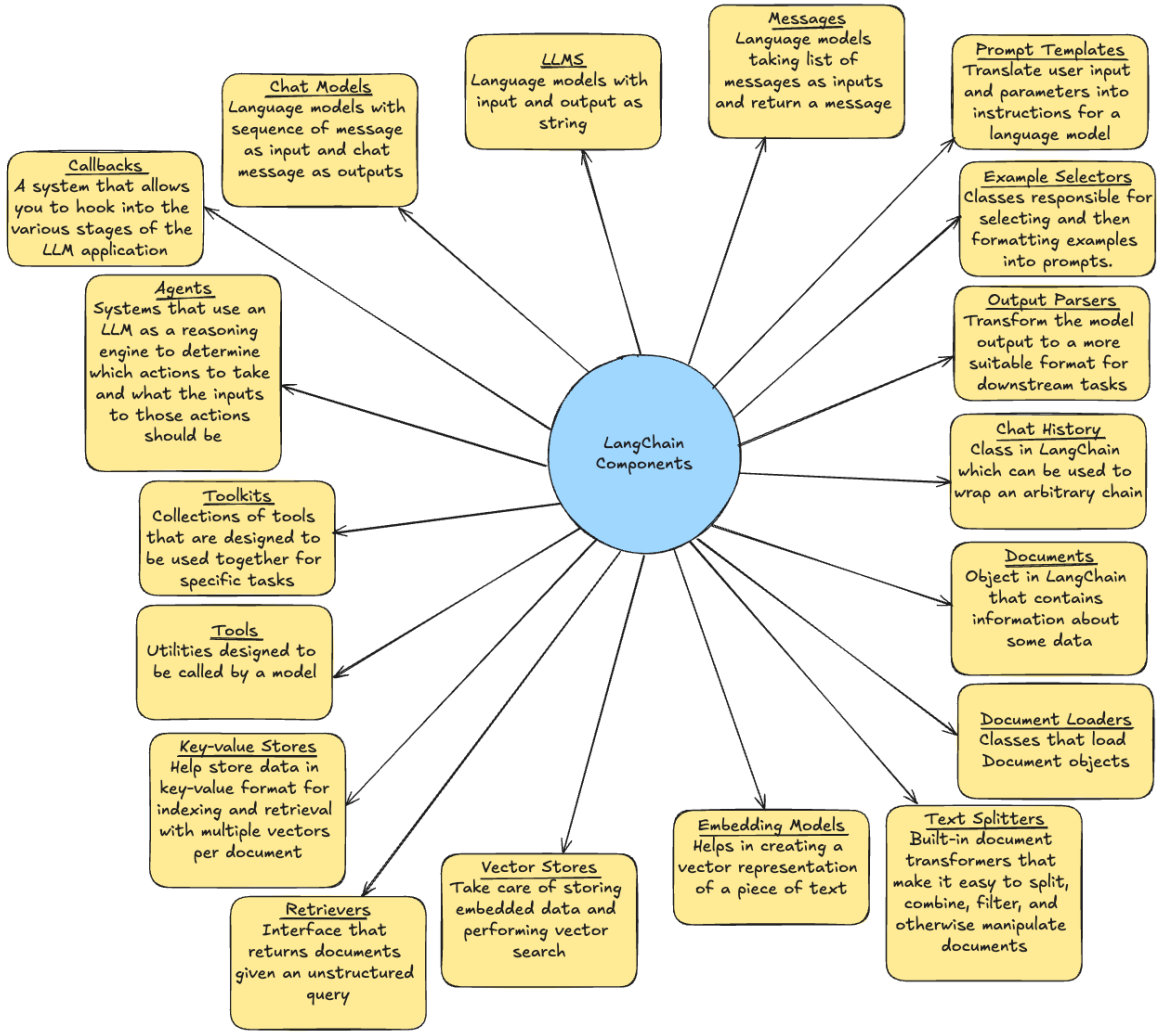
Langchain et al. use these frequently.

Realpython Decorator overview

Langchain

High-level Concepts

- When you use a desktop or web-based LLM Application it manages many details for you that as an API caller you manage
  - Current conversational context, chat History, past chats
  - Which part of the conversation is you vs the LLM
  - Custom settings and prompts
  - Access to custom knowledge
  - Access to specific integrations that you may have allowed
- Langchain abstracts away the specifics of the API implementation of each LLM.
- reference for LLMs APIs(https://python.langchain.com/api_reference/langchain/index.html
  - Look down the left side menu
  - Examine the following image on the next slide Attribution is below.) ^1
    - The abstraction makes the data transportation uniform across the various implementations


LangChain Components
LLMS
Language models with input and output as string
Messages
Language models taking list of messages as inputs and return a message
Chat Models
Language models with sequence of message as input and chat message as outputs
Prompt Templates
Translate user input and parameters into instructions for a language model
Callbacks
A system that allows you to hook into the various stages of the LLM application
Example Selectors
Classes responsible for selecting and then formatting examples into prompts.
Agents
Systems that use an LLM as a reasoning engine to determine which actions to take and what the inputs to those actions should be
Output Parsers
Transform the model output to a more suitable format for downstream tasks
Chat History
Class in LangChain which can be used to wrap an arbitrary chain
Toolkits
Collections of tools that are designed to be used together for specific tasks
Documents
Object in LangChain that contains information about some data
Tools
Utilities designed to be called by a model
Document Loaders
Classes that load Document objects
Key-value Stores
Help store data in key-value format for indexing and retrieval with multiple vectors per document
Text Splitters
Built-in document transformers that make it easy to split, combine, filter, and otherwise manipulate documents
Embedding Models
Helps in creating a vector representation of a piece of text
Vector Stores
Take care of storing embedded data and performing vector search
Retrievers
Interface that returns documents given an unstructured query

Langchain - simplest case

- prompt chaining
- while this is the simplest case other uses make runnable objects of LLMs, prompts and states that "pipe" into each other like the bash | operator or R's %>% (from the magrittr package)

```python
from langchain_core.prompts import ChatPromptTemplate
from langchain_openai import ChatOpenAI

prompt = ChatPromptTemplate.from_messages([
    ("system", "You are a helpful assistant"),
    ("human", "{input}")
])
model = ChatOpenAI()
chain = prompt | model
result = chain.invoke({"input": "Hello world"})
```

LangGraph

- Graph-based architecture for complex workflows
- State management and check-pointing
- Human-in-the-loop capabilities
- Multi-agent coordination
- Langgraph is how syntactic python does the nodes and connections we saw in the GUI's

Reference: Langgraph Documentation

```python
from langgraph.graph import StateGraph
from typing import TypedDict

class State(TypedDict):
    messages: list

def agent_node(state):
    # Agent logic here
    return {"messages": state["messages"] + ["response"]}

graph = StateGraph(State)
graph.add_node("agent", agent_node)
graph.set_entry_point("agent")
graph.set_finish_point("agent")
```

LangSmith And Web Platform

- Real-time tracing and debugging
- Performance metrics (latency, cost, quality)
- A/B testing capabilities
- Framework-agnostic (works beyond LangChain)
- My Login Link

Reference: LangSmith Documentation

```python
from langsmith import traceable

@traceable
def my_llm_function(input_text):
    # Your LLM logic here
    return response

# Automatic tracing to LangSmith dashboard
result = my_llm_function("Hello")
```

You might ask yourself Why?

Langsmith Business Impact:

- Cost Optimization: Track token usage and spending
- Quality Assurance: Monitor response quality
- Debugging: Trace failures in production
- Prompt testing

LangServe:

- Convert LangChain chains to REST endpoints
- Automatic API documentation
- Streaming support
- Production-ready deployment
- Uses Pydantic for Typing and FastAPI for API-ing

```python
from langserve import add_routes
from fastapi import FastAPI

app = FastAPI()
# Add your chain as an API endpoint
add_routes(app, chain, path="/my-chain")

# Now accessible at /my-chain/invoke
```

Use Cases:

- API Deployment: Expose LLM workflows as services
- Integration: Connect with existing systems
- Scalability: Production deployment

Reference: LangServe Documentation Sadly no longer adding features in preference to Paid platform

LangChain Hub (LangHub) - Prompt sharing

- Prompt version control
- Community sharing platform
- Model-specific optimizations
- Programmatic access
- Its a combination of github and hugging face for prompts

Why? / Value Proposition:

- Community: Share best practices
- Re-usability: Don’t reinvent prompts
- Versioning: Track prompt evolution
  - Then check the variation performance in LangSmith

```python
from langchain import hub

# Pull a public prompt
prompt = hub.pull("rlm/rag-prompt")

# Push your own prompt
hub.push("my-prompt", prompt_template)
```

LangFlow - Visual GUI Version

- Visual node-based editor
- Prebuilt components
- Integration with LangChain
- Can export individual flows as JSON … still can’t Diff it.
- Rapid prototyping
- Can self host, but holds state
  - while not transformable back to code, there is a feature request for this.
- DatStax, not LangChain, owns this and is in process of being acquired by IBM.

Langflow Setup Options

- **Repository**: https://github.com/langflow-ai/langflow
- **License**: MIT (completely free)
- **Installation**: Self-hosted

DataStax LangFlow (Paid Platform)

- **URL**: https://www.datastax.com/products/langflow
- Type: Managed cloud service
- Visual Interface**: Same visual interface + enterprise features

Link to running process

```
# Install with pip
pip install langflow

# Start the visual interface
python -m langflow run

# Or use the shortcut
langflow run --host 0.0.0.0 --port 7860
```

What We’ve Learned

- Flowise and n8n simplify AI workflows.
- Docker provides robust deployment.
- Persistence is critical for data integrity.
- Everyone is using these as freeware - with a managed hosting component, except for raw python
- Trying to serve your AI as a product or service, to an App, a business or an enterprise has different needs than your home lab does.
- Langchain family tools have deep methods for each, but has a much higher learning curve
- Thankfully we have AI to help digest all the documentation.

Succinctly:

The GUI tools allow you to learn the strategy, patterns, interactions, and to iterate quickly.

The code allows you to deploy seamlessly.

Open to Other Experiences

"When the Facts Change, I Change My Mind. What Do You Do, Sir?"

- John Maynard Keynes/ Paul Samuelson/ Winston Churchill … by quoteresearch July 22, 2011
- I’m open to being educated on the ways to run these GUI based methods in a more sustainable way. I’m likely ignorant of the way that is done.

N8N:

- Their website and community forums feature numerous case studies and user stories from companies of various sizes.
- Official case studies page: https://n8n.io/case-studies/

Flowise:

- The FlowiseAI GitHub repository and community discussions also point to production use cases.
- Examples in the "Show and Tell" section of their GitHub discussions: https://github.com/FlowiseAI/Flowise/discussions/categories/show-and-tell

Minimal Resources

N8N

- n8n Official Documentation
- n8n video courses
- n8n AI Automation Workflows
- n8n Workflow library and tutorials

Flowise

- Flowise Getting Started
- Flowise Official Tutorials
- Flowise Community Flows
- Leon Van Zyl

Slidev - Slidev

- The tutorial I wish I had read

Lang-various

- LangChain Python Documentation
- LangChain Templates
- RAG Advanced Patterns
- LangChain Gallery
- Agent Construction Patterns
- LangGraph Examples
- LangGraph Tutorials
- LangSmith Quick-start
- Advanced LangSmith Tracing
- LangSmith Evaluation Guides

Thank you

Contact me at: curtis@blueskyai.co

Curtis O on Cleveland Tech Slack, ClePy
Repo: https://github.com/CurtisONeal/pyohio_2025_talk

Please reach out or connect
http://www.linkedin.com/in/curtisoneal

Curtis O'Neal, MBA
Data Scientist at blueskyai.co