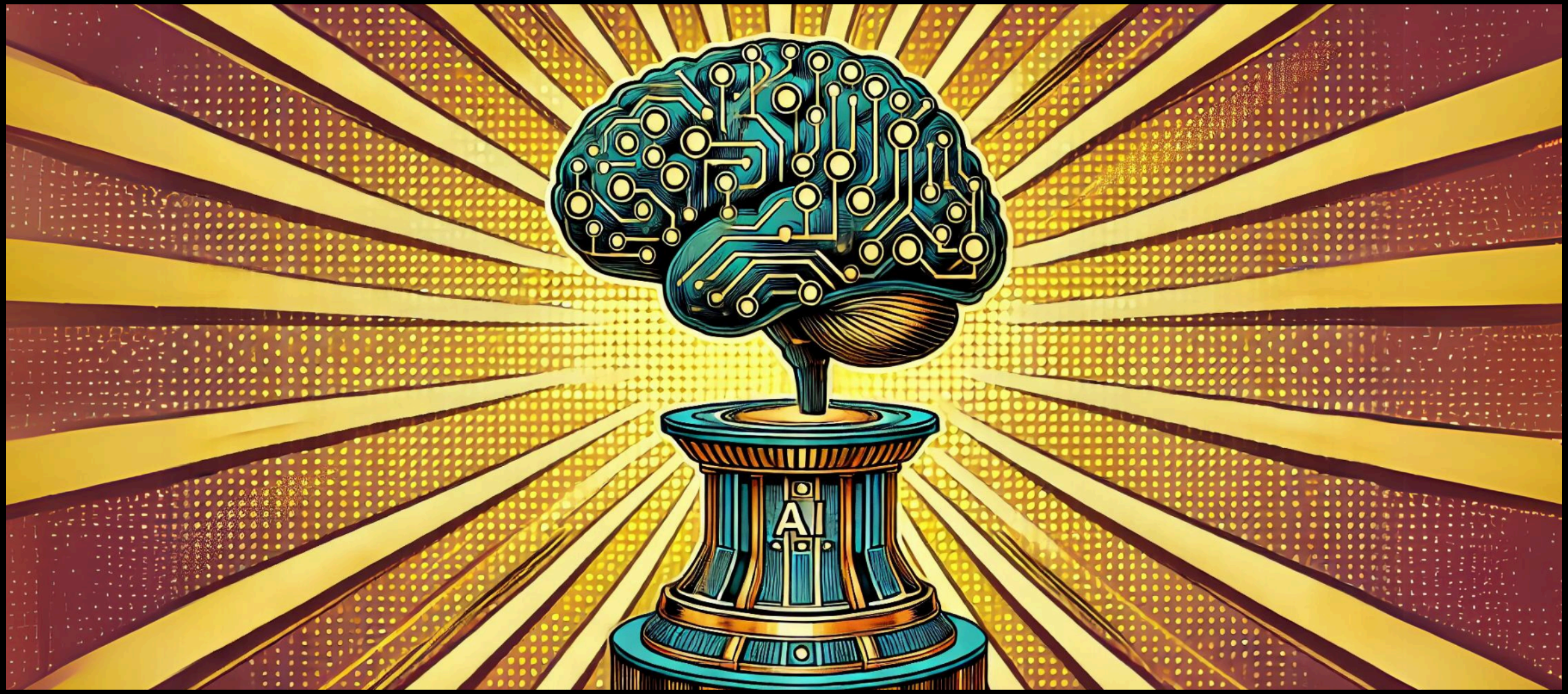
# LLM Engineering

## MASTER AI & LARGE LANGUAGE MODELS



WELCOME TO WEEK 5

# RAG Week

## What you can now do

- Code with Frontier Models including AI Assistants with Tools, and with open-source models with HuggingFace transformers
- Confidently choose the right LLM for your project, backed by metrics
- Build solutions to generate code with Frontier and open-source LLMs

---

## By end of today you'll be able to

- Explain the big idea behind Retrieval Augmented Generation (RAG)
- Walk through the high level flow for adding expertise to queries
- Implement a toy version of RAG without vector databases.. yet

INTRODUCING RAG

# Motivating RAG

**We've already used techniques to improve prompts**

- Multi-shot prompting
- Use of tools
- Additional context

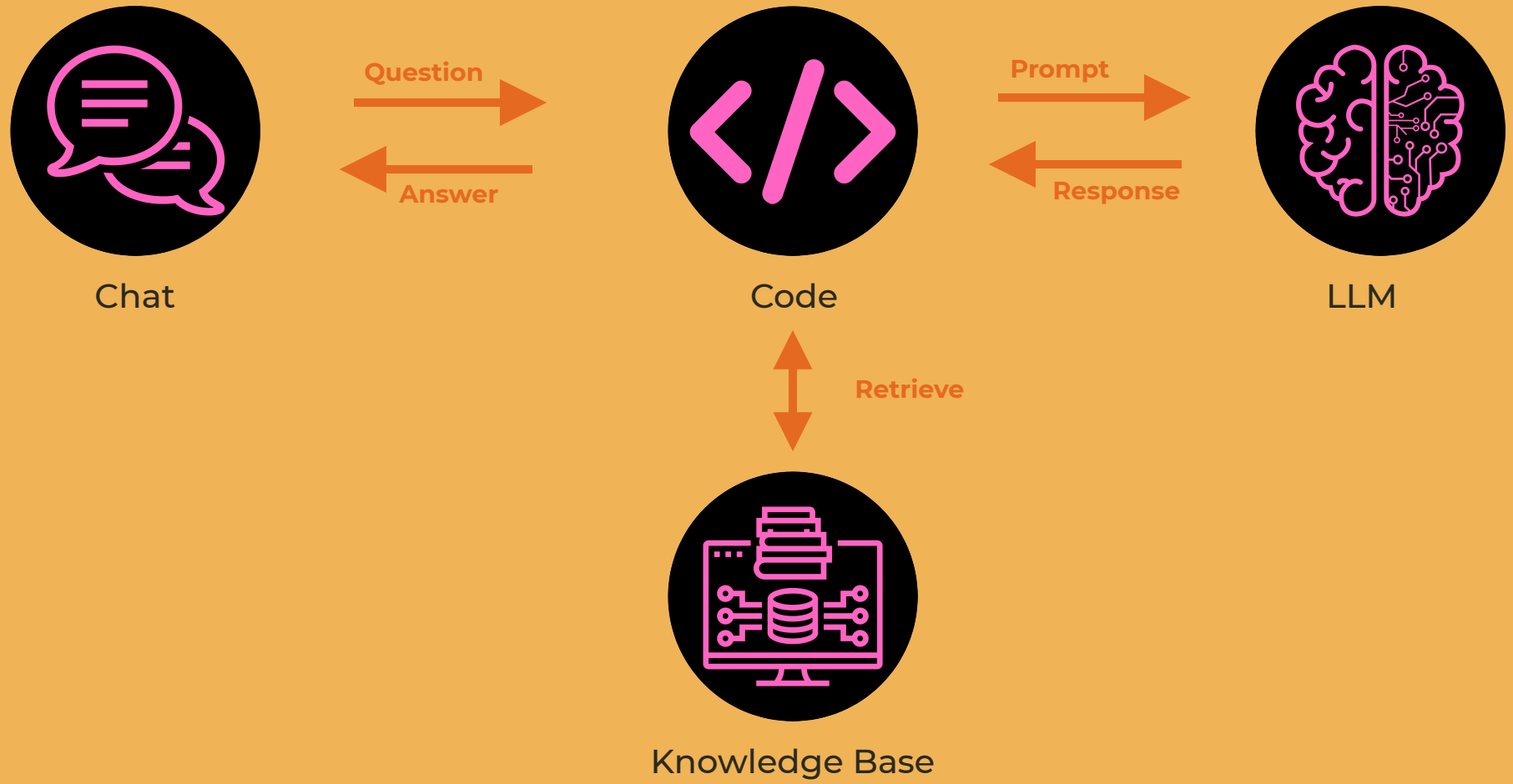
---

**We can take this to the next level**

- Build a database of expert information, called a Knowledge Base
- Every time the user asks a question, search for anything relevant in the Knowledge Base
- Add relevant details to the prompt

# The small idea behind RAG

Improve the prompt with context from a Knowledge Base



INTRODUCING RAG

# A small example of the small idea

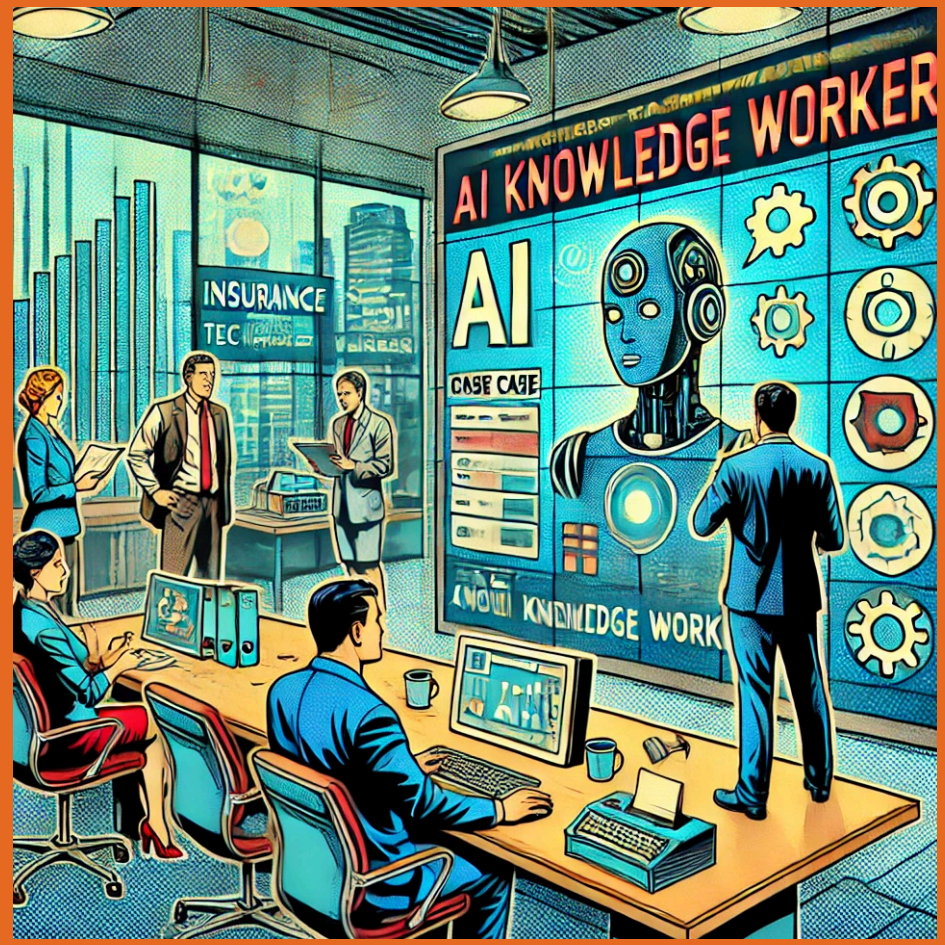
**Business setup**

- We work for an Insurance Tech startup
- We have a Knowledge Base of the company shared drive
- Task is to build an AI Knowledge Worker

---

**The Blunt Instrument implementation**

- Read names of products and employees
- See if questions refer to employee or products by name
- Add relevant details to the prompt



For those who haven't encountered Vectors

# Encoding LLMs and Vector Embeddings

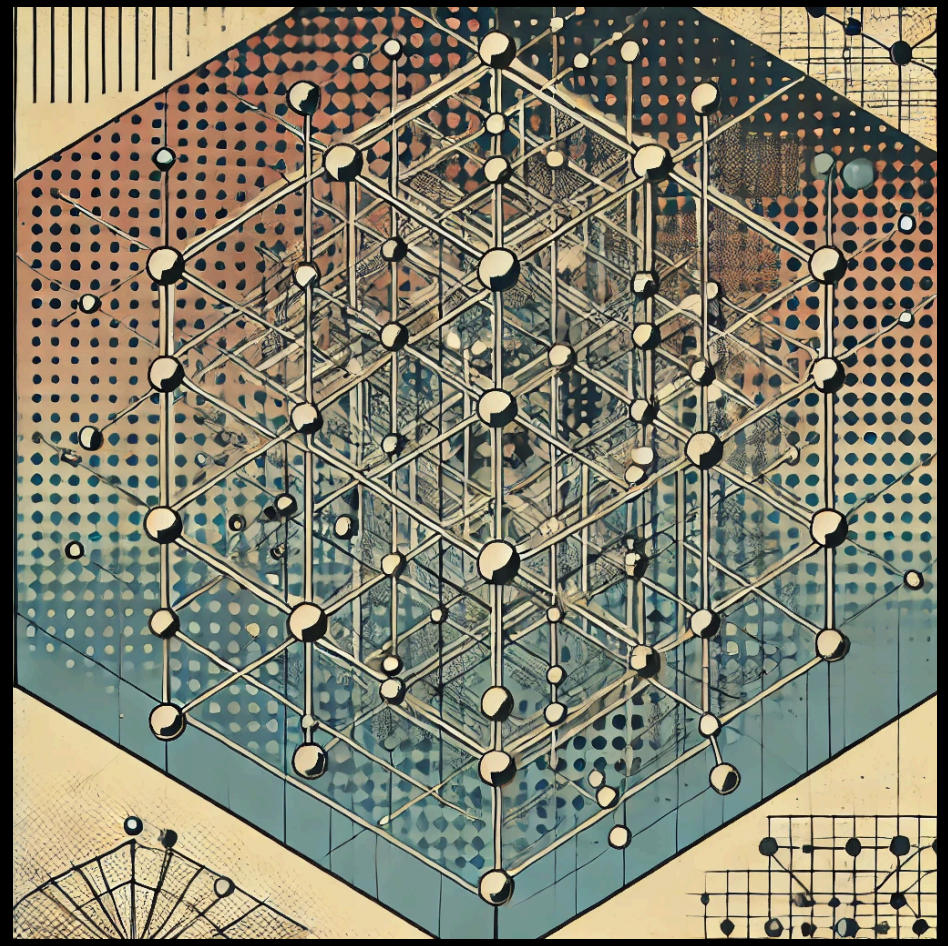
### Auto-Encoding vs Auto-Regressive LLMs

- Auto-regressive LLMs predict a future token from the past
- Auto-encoding LLMs produce output based on the full input

---

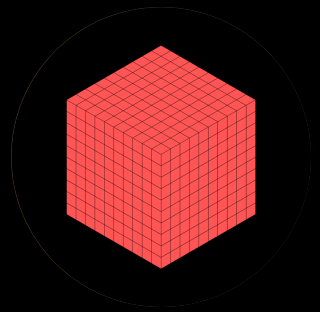
### Auto-encoding LLMs

- Applications include Sentiment Analysis and classification
- Also used to calculate "Vector Embeddings", representing an input as a list of numbers - i.e. a vector
- Examples include BERT from Google and OpenAIEmbeddings from OpenAI

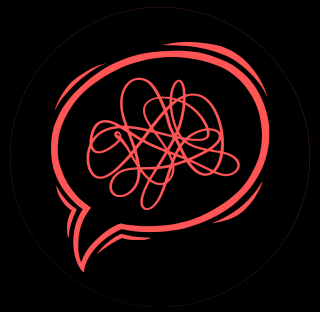


# These vectors mathematically represent the 'meaning' of an input

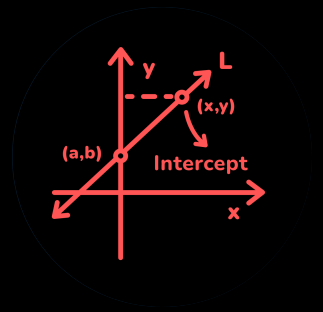
Can represent a character, a token, a word, an entire document, or something abstract



Typically have hundreds, or thousands of dimensions

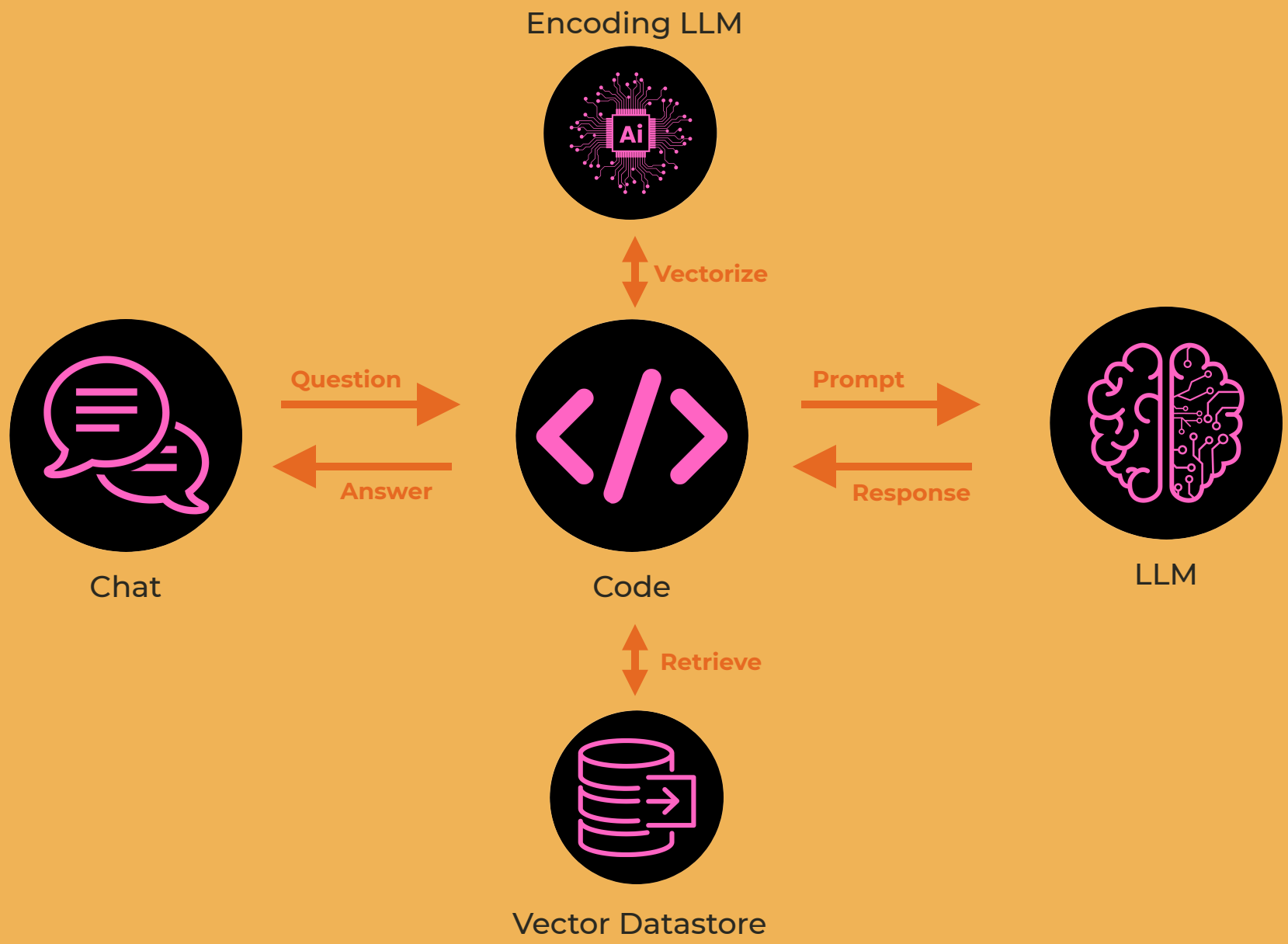


Represent an 'understanding' of the inputs; similar inputs are close to each other




Support 'vector math' like the famous example: "King - Man + Woman = Queen"

# The big idea behind RAG



PROGRESS REPORT

# RAG Week

## What you can now do

- Generate text and code with Frontier Models including AI Assistants with Tools, and with open-source models with HuggingFace transformers
- Confidently choose the right LLM for your project, backed by metrics
- Explain how RAG uses vector embeddings and vector datastores to add context to prompts

---

## By end of next time you'll be able to

- Describe the LangChain framework, with benefits and limitations
- Use LangChain to read in a Knowledge Base of documents
- Use LangChain to divide up documents into overlapping chunks