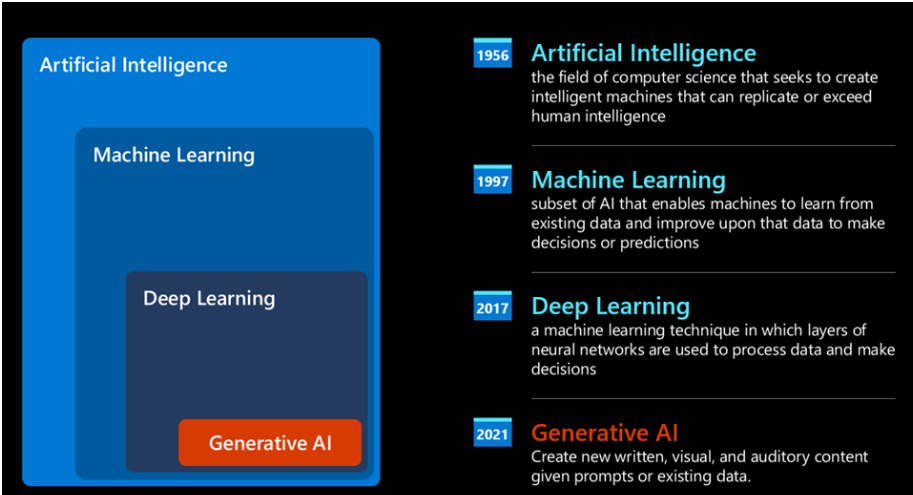
# Generative AI:

## LLM

Generative AI models – also known as **Large Language Models** (LLMs)

Those models are trained on a huge amount of unlabeled data from diverse sources like books, articles and websites and learn from the human feedbacks



## How do LLM Works?

Receive a text as input and generate a text as output.

Input to the model is processed by a **tokenizer.** Tokenizer breaks into **token** – chunks of word

Each token is mapped with a **token index**, which is the integer encoding of the original text chunk.

The input of a large language model is known as **prompt**

## Types of LLMs

**Audio** – whisper type model

**Image generation -** DALL-E and Midjourney

**Text generation** - GPT-3.5 to GPT-4 (Too expensive ), Azure OpenAI resources(cost friendly)

**Multi-modality**. - [gpt-4 turbo with vision or gpt-4o](https://learn.microsoft.com/azure/ai-services/openai/concepts/models#gpt-4-and-gpt-4-turbo-models?WT.mc_id=academic-105485-koreyst)

**Foundation Models** are broad, versatile models that support **multiple data** modalities (text, image, video, etc.) and serve as a base for various tasks.

**LLMs** are a specific subset of foundation models focused exclusively on **natural language (Mostly Text)** understanding and generation tasks.

While LLMs like GPT are a type of foundation model, not all foundation models are LLMs. Some foundation models work across different data types and tasks (like CLIP, which handles both text and images).

**Open Source versus Proprietary Models**

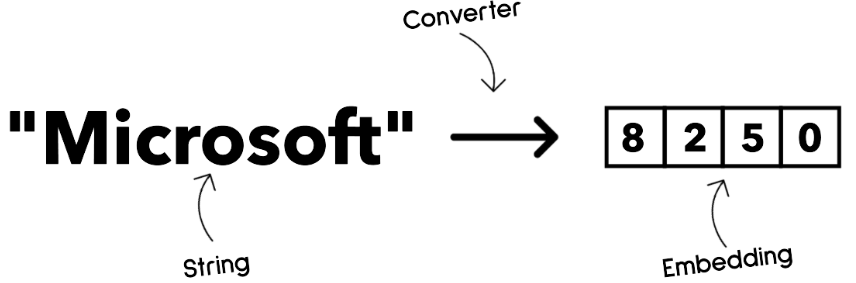
Open source - models that are made available to the **public** and can be used by anyone. Some of them are [Alpaca](https://crfm.stanford.edu/2023/03/13/alpaca.html?WT.mc_id=academic-105485-koreyst), [Bloom](https://huggingface.co/bigscience/bloom) and [LLaMA](https://llama.meta.com/).

Proprietary Models - models that are owned by a company and are not made available to the public. These models are often optimized for production use. Some of them are [OpenAI models](https://platform.openai.com/docs/models/overview?WT.mc_id=academic-105485-koreyst), [Google Bard](https://sapling.ai/llm/bard?WT.mc_id=academic-105485-koreyst) or [Claude 2](https://www.anthropic.com/index/claude-2?WT.mc_id=academic-105485-koreyst).

LLMs can also be categorized by the output they generate.

**Embedding**

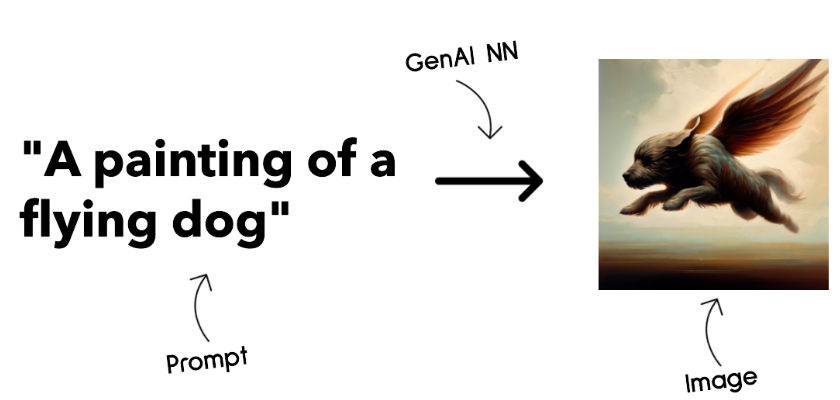
set of models that can convert text into a numerical form, called embedding .Eg: [OpenAI embeddings](https://platform.openai.com/docs/models/embeddings?WT.mc_id=academic-105485-koreyst).

**Image generation**

Models that generate images. They create,edit image

Image generation models are often trained on large datasets of images, such as [LAION-5B](https://laion.ai/blog/laion-5b/?WT.mc_id=academic-105485-koreyst),

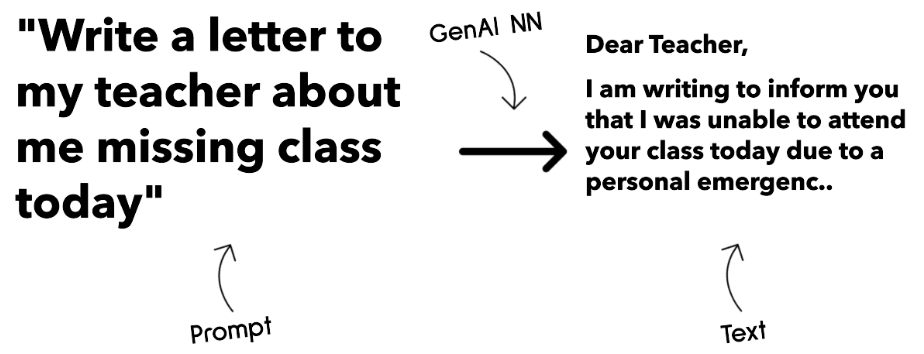
Eg: [DALL-E-3](https://openai.com/dall-e-3?WT.mc_id=academic-105485-koreyst) and [Stable Diffusion models](https://github.com/Stability-AI/StableDiffusion?WT.mc_id=academic-105485-koreyst).



**Text and code generation**

Models are models that generate **text or code**.

These models are often used for text summarization, translation, and question answering.

Text generation models are often trained on large datasets of text, such as [BookCorpus](https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Zhu_Aligning_Books_and_ICCV_2015_paper.html?WT.mc_id=academic-105485-koreyst), Code generation models, like [CodeParrot](https://huggingface.co/codeparrot?WT.mc_id=academic-105485-koreyst), are often trained on large datasets of code, such as GitHub

**Endcoder-only,Decoder-only Vs Encoder-Decoder**

**Decoder-only**

They are very good at **writing engaging and informative content**, but they are not very good at understanding the topic and the learning objectives. Some examples of Decoder models are GPT family models, such as **GPT-3.**

**Encoder-only**

They are **good at understanding context**, but they are not good at generating content. An example of an Encoder only model would be **BERT**.

**Encoder-Decoder:**

Model that can understand the context & generating the content

examples would be **BART and T5**

**Service Vs Model**

**Service**

A service is a product that is offered by a **Cloud Service Provider**, and is often a combination of models, data, and other components require a subscription or payment to use

Example: [Azure OpenAI Service](https://learn.microsoft.com/azure/ai-services/openai/overview?WT.mc_id=academic-105485-koreyst)

**Model**

Models are just the Neural Network, with the parameters, weights, and others. Allowing companies to run locally

They can be both open source & licensed

Example: LLaMA

**Improving LLM results**

Prompt engineering with context.

Retrieval Augmented Generation, RAG

Fine-tuned model

Prompt

Prompt engineering is the process by which we **guide the model towards more relevant** responses by providing more useful instructions or context.

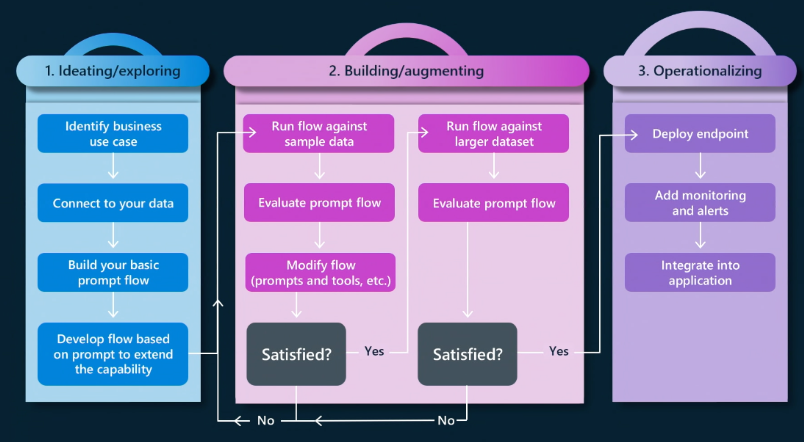
Techniques Of prompting

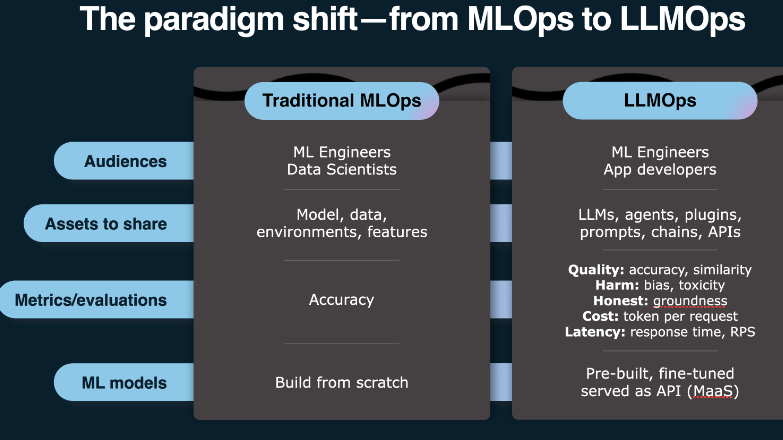
* **Zero-shot prompting**, this is the most basic form of prompting. It's a single prompt requesting a response from the LLM based solely on its training data.
* **Few-shot prompting**, this type of prompting guides the LLM by providing 1 or more examples it can rely on to generate its response.
* **Chain-of-thought**, this type of prompting tells the LLM how to break down a problem into steps.
* **Generated knowledge**, to improve the response of a prompt, you can provide generated facts or knowledge additionally to your prompt.
* **Least to most**, like chain-of-thought, this technique is about breaking down a problem into a series of steps and then ask these steps to be performed in order.
* **Self-refine**, this technique is about critiquing the LLM's output and then asking it to improve.
* **Maieutic prompting**. What you want here is to ensure the LLM answer is correct and you ask it to explain various parts of the answer. This is a form of self-refine.

## Gen-AI Lifecycle

The generative AI lifecycle is a framework that guides you through the stages of developing, deploying, and maintaining a generative AI application.

We can categorize older AI apps as "ML Apps" and newer AI Apps as "GenAI Apps" or just "AI Apps"

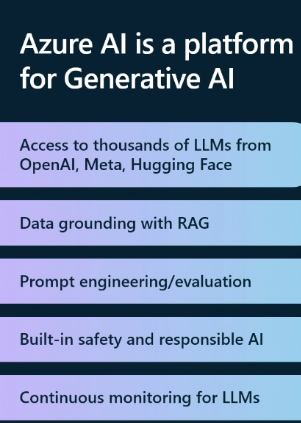




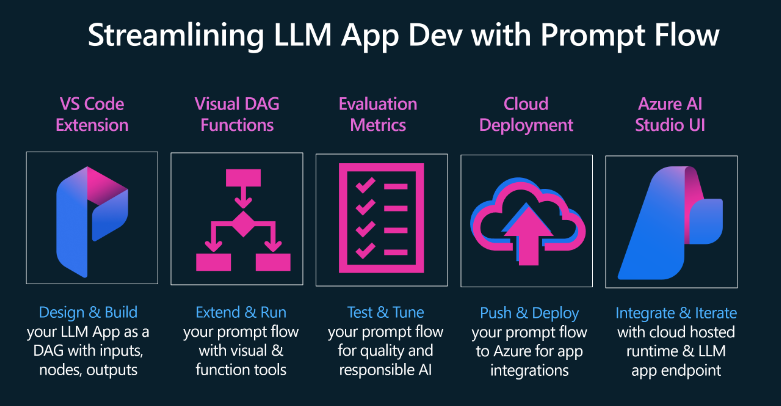
**Lifecycle Tooling**

Microsoft provides the [Azure AI Platform](https://azure.microsoft.com/solutions/ai/?WT.mc_id=academic-105485-koreys) and [PromptFlow](https://microsoft.github.io/promptflow/index.html?WT.mc_id=academic-105485-koreyst) facilitate and make your cycle easy to implement

The [Azure AI Platform](https://azure.microsoft.com/solutions/ai/?WT.mc_id=academic-105485-koreys), allows you to use [AI Studio](https://ai.azure.com/?WT.mc_id=academic-105485-koreys). AI Studio is a web portal allows you to Explore models, samples and tools. Managing your resources, UI development flows and SDK/CLI options for Code-First development.







**Open-Source Models:**

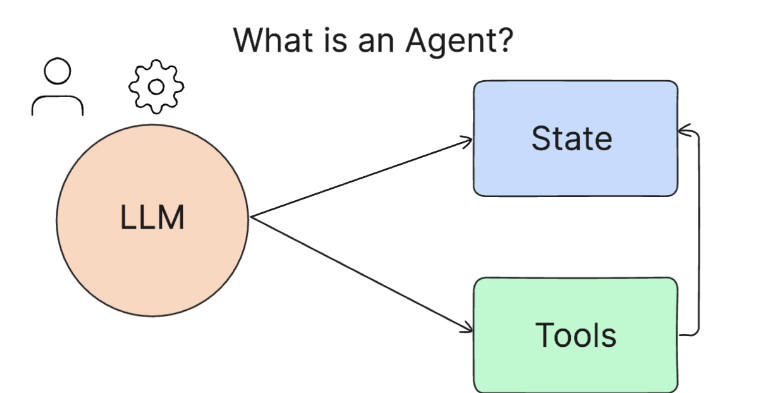
Lama

Mistral

Falcon

## AI-Agents

AI Agents allow Large Language Models (LLMs) to perform tasks by giving them access to a **state** and **tools**.



**State** - This refers to the context that the LLM is working in. The LLM uses the context of its **past actions** and **the current context,** guiding its decision-making for subsequent actions. AI Agent Frameworks allow developers to **maintain this context** easier.

**Tools** - To complete the task that the user has requested and that the LLM has planned out, the LLM needs access to tools. Some examples of tools can be a **database, an API,** an external application or even another LLM!

Some of the Ai-Agents

### LangChain Agents

To manage the **state,** it uses a built-in function called the **AgentExecutor**. This accepts the defined agent and the tools that are available to it.

The Agent Executor also stores the **chat history** to provide the **context** of the chat.

### AutoGen

The focus of AutoGen is conversations.

Agents are both **conversable** and **customizable**.

**Conversable -** LLMs can start and continue a conversation with another LLM in order to complete a task. This is done by creating **AssistantAgents** and giving them a specific system message.

**Customizable** - Agents can be defined not only as LLMs but be a **user or a tool.** As a developer, you can define a **UserProxyAgent** which is responsible for interacting with the user for **feedback** in completing a task. This feedback can either continue the execution of the task or stop it.

### **State and Tools**

To change and manage state, an assistant Agent generates Python code to complete the task.

### TaskWeaver

It is known as a "**code-first**" agent because instead of working strictly with strings, it can work with DataFrames in Python.

This becomes extremely useful for data analysis and generation tasks. This can be things like creating graphs and charts or generating random numbers.

**State and Tools**

To manage the state of the conversation, TaskWeaver uses the concept of a **Planner**. The Planner is a LLM that takes the request from the users and maps out the tasks that need to be completed to fulfill this request.

Planner is pluggins that can be a python class or general code interpreter

### Jarvis

It uses an LLM to manage the state of the conversation and the tools are other AI models.

Each of the AI models are specialized models that perform certain tasks such as **object detection, transcription** or **image captioning**.

**Freely available pre-trained models**

TensorFlow hub

FastText from facebook

GPT3

**Python Open-source Libraries to work with NLP**

NLTk

Spacy

Gensim

Scikit

TensorFlow

pyTorch

Hugging Face

Regular expression in python is necessary for pattern matching - <https://regex101.com/> this website would be helpful