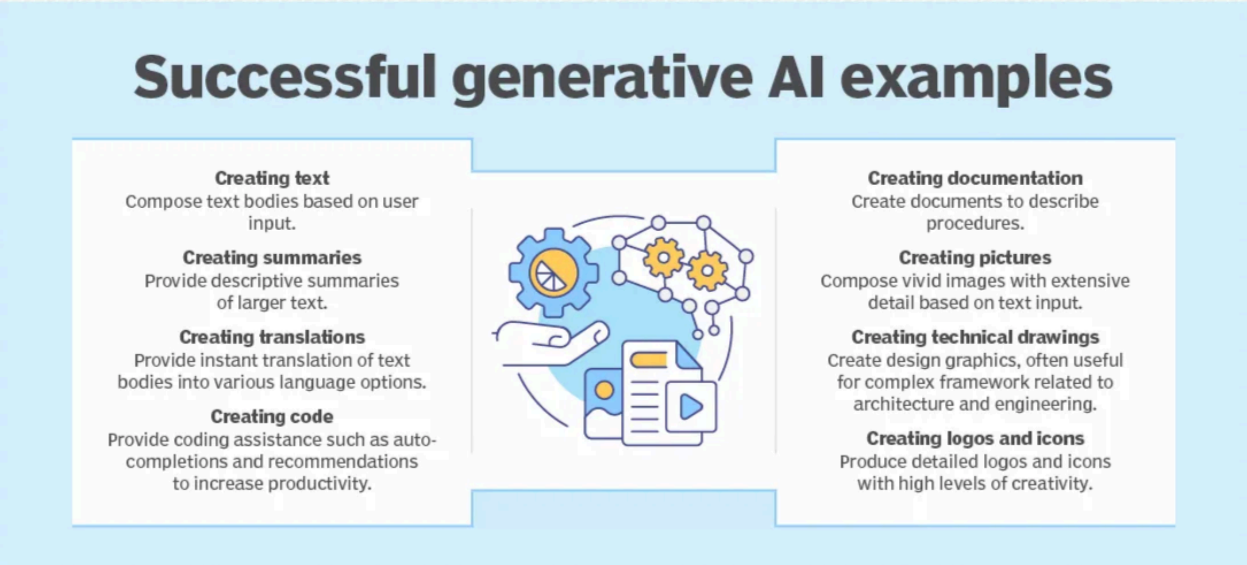
Lesson 1: Intro to GenAI & LLMs (Concepts: GenAI, LLM Models, Prompt Engineering)

Introduction to Generative AI

What is GenAI?

Generative artificial intelligence, or GenAI, uses sophisticated algorithms to organise large, complex data sets into meaningful clusters of information in order to create new content, including text, images and audio, in response to a query or prompt. GenAI typically does two things: First, it encodes a collection of existing information into a form (vector space) that maps data points based on the strength of their correlations (dependencies). Second, when prompted, it then generates (decodes) new content by finding the correct context within the existing dependencies in the vector space.

Familiar to users through popular interfaces such as Openai's ChatGPT and Google's Gemini, generative AI can answer complex questions, summarise vast amounts of information, and automate many tasks previously done by humans. For example, businesses use generative AI to help draft reports, personalise marketing campaigns, make commercial films and improve code. Software vendors are integrating generative AI into core business applications, such as CRM and ERP, to boost efficiency and improve decision-making. GenAI is also being added to existing automation software, such as robotic process automation (RPA) and customer service chatbots, to make them more proactive. Under the hood, generative AI is used to create synthetic data for training other AI and machine learning models.



Successful generative AI examples

Creating text
Compose text bodies based on user input.

Creating summaries
Provide descriptive summaries of larger text.

Creating translations
Provide instant translation of text bodies into various language options.

Creating code
Provide coding assistance such as auto-completions and recommendations to increase productivity.

Creating documentation
Create documents to describe procedures.

Creating pictures
Compose vivid images with extensive detail based on text input.

Creating technical drawings
Create design graphics, often useful for complex framework related to architecture and engineering.

Creating logos and icons
Produce detailed logos and icons with high levels of creativity.

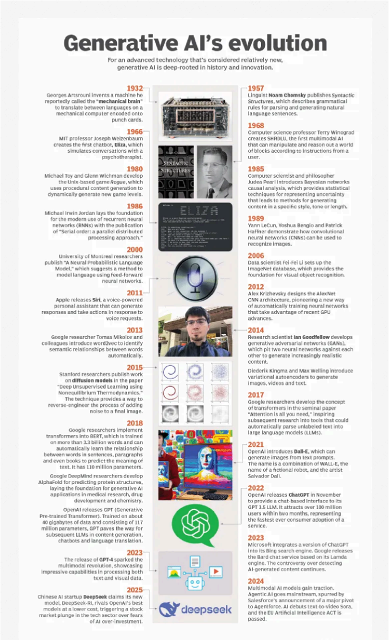
Generative AI history

The Eliza chatbot, created by Joseph Weizenbaum in the 1960s, represented a major breakthrough in natural language processing (NLP) and was one of the earliest examples of generative AI. These early implementations used a rules-based approach that broke easily due to a limited vocabulary, lack of context and overreliance on patterns, among other shortcomings. Early chatbots were also difficult to customise and extend.

The field saw a resurgence in the wake of advances in neural networks and deep learning in 2010, which enabled the technology to automatically learn to parse existing text, classify image elements, and transcribe audio.

Ian Goodfellow introduced GANS in 2014. This deep learning technique provided a novel approach for organising competing neural networks to generate and then rate content variations. These could generate realistic people, voices, music and text. This inspired interest in -- and fear of -- how generative AI could be used to create realistic deepfakes that impersonate voices and people in videos.

Since then, progress in other neural network techniques and architectures has helped expand the capabilities of generative AI. Techniques include VAES, long short-term memory, transformers, diffusion models, Gaussian splatting and neural radiance fields. Researchers and vendors are also starting to combine techniques in novel ways to improve accuracy, precision and performance while reducing hallucinations.



Generative AI's evolution

For an advanced technology that's considered relatively new, generative AI is deep-rooted in history and innovation.

1932
Georges Artsrouni invents a machine he reportedly called the "**mechanical brain**" to translate between languages on a mechanical computer encoded onto punch cards.

1966
MIT professor Joseph Weizenbaum creates the first chatbot, **Eliza**, which simulates conversations with a psychotherapist.

1980
Michael Toy and Glenn Wichman develop the Unix-based game *Rogue*, which uses procedural content generation to dynamically generate new game levels.

1986
Michael Irwin Jordan lays the foundation for the modern use of recurrent neural networks (RNNs) with the publication of "Serial order: a parallel distributed processing approach."

2000
University of Montreal researchers publish "A Neural Probabilistic Language Model," which suggests a method to model language using feed-forward neural networks.

2011
Apple releases **Siri**, a voice-powered personal assistant that can generate responses and take actions in response to voice requests.

2013
Google researcher Tomas Mikolov and colleagues introduce word2vec to identify semantic relationships between words automatically.

2015
Stanford researchers publish work on **diffusion models** in the paper "Deep Unsupervised Learning using Nonequilibrium Thermodynamics." The technique provides a way to reverse-engineer the process of adding noise to a final image.

2018
Google researchers implement transformers into BERT, which is trained on more than 3.3 billion words and can automatically learn the relationship between words in sentences, paragraphs and even books to predict the meaning of text. It has 110 million parameters.

Google DeepMind researchers develop AlphaFold for predicting protein structures, laying the foundation for generative AI applications in medical research, drug development and chemistry.

OpenAI releases GPT (Generative Pre-trained Transformer). Trained on about 40 gigabytes of data and consisting of 117 million parameters, GPT paves the way for subsequent LLMs in content generation, chatbots and language translation.

2023
The release of **GPT-4** sparked the multimodal revolution, showcasing impressive capabilities in processing both text and visual data.

2025
Chinese AI startup **DeepSeek** claims its new model, DeepSeek-RI, rivals OpenAI's best models at a lower cost, triggering a stock market plunge in the tech sector over fears of AI over-investment.

1957
Linguist **Noam Chomsky** publishes *Syntactic Structures*, which describes grammatical rules for parsing and generating natural language sentences.

1968
Computer science professor Terry Winograd creates SHRDLU, the first multimodal AI that can manipulate and reason out a world of blocks according to instructions from a user.

1985
Computer scientist and philosopher Judea Pearl introduces Bayesian networks causal analysis, which provides statistical techniques for representing uncertainty that leads to methods for generating content in a specific style, tone or length.

1989
Yann LeCun, Yoshua Bengio and Patrick Haffner demonstrate how convolutional neural networks (CNNs) can be used to recognize images.

2006
Data scientist Fei-Fei Li sets up the ImageNet database, which provides the foundation for visual object recognition.

2012
Alex Krizhevsky designs the AlexNet CNN architecture, pioneering a new way of automatically training neural networks that take advantage of recent GPU advances.

2014
Research scientist **Ian Goodfellow** develops generative adversarial networks (GANs), which pit two neural networks against each other to generate increasingly realistic content.

Diederik Kingma and Max Welling introduce variational autoencoders to generate images, videos and text.

2017
Google researchers develop the concept of transformers in the seminal paper "Attention is all you need," inspiring subsequent research into tools that could automatically parse unlabeled text into large language models (LLMs).

2021
OpenAI introduces **Dall-E**, which can generate images from text prompts. The name is a combination of WALL-E, the name of a fictional robot, and the artist Salvador Dali.

2022
OpenAI releases **ChatGPT** in November to provide a chat-based interface to its GPT 3.5 LLM. It attracts over 100 million users within two months, representing the fastest ever consumer adoption of a service.

2023
Microsoft integrates a version of ChatGPT into its Bing search engine. Google releases the Bard chat service based on its Lamda engine. The controversy over detecting AI-generated content continues.

2024
Multimodal AI models gain traction. Agentic AI goes mainstream, spurred by Salesforce's announcement of a major pivot to Agentforce. AI debuts text-to-video Sora, and the EU Artificial Intelligence ACT is passed.

How does generative AI work?

The generative AI process starts with foundation models, such as the GPT series, Palm and Gemini. These are large neural networks trained on massive datasets that provide a broad assimilation of known information and knowledge. They generally include text, which provides a way to distill human concepts. Other data sources include images, video, IoT, robot instructions and enterprise data.

As noted, basic generative AI models consist of an encoder and a decoder. The encoder transforms text, code, images and other prompts into a format AI can process. This *intermediate representation* could be a vector embedding or a probabilistic latent space. The decoder generates content by transforming the intermediate representation into new content, such as a chatbot response, a document summary, a translation, or another data type.

Best practices for using generative AI

The best practices for using generative AI vary depending on the modalities, workflow and desired goals. That said, it is always important to consider factors such as accuracy, transparency and ease of use when working with generative AI. The following practices serve as a guide:

- Clearly label all generative AI content for users and consumers.

- Assess the cost/benefit tradeoffs compared with other tools.
- Vet the accuracy of generated content using primary sources, where applicable.
- Consider how bias might get woven into generated AI results.
- Double-check the quality of AI-generated code and content using other tools.
- Learn the strengths and limitations of each generative AI tool.
- Familiarise yourself with common failure modes in results and work around them.
- Vet new applications with subject matter experts to identify problems.
- Implement guardrails to mitigate issues with trust and security.

Specialised GenAI tools

A variety of best-of-breed commercial and open-source tools also excel at generating specific kinds of content or for various use cases. The following examples are a case in point:

🔤Text-to-Text (Text Generation & Completion)

These tools generate or transform text based on input prompts.

1. **OpenAI ChatGPT**
    - Capabilities: Text generation, summarization, Q&A, translation, etc.
    - Link: https://chat.openai.com
2. **Anthropic Claude**
    - Capabilities: Long-form reasoning, summarization, creative writing.
    - Link: https://claude.ai
3. **Google Gemini (Bard)**
    - Capabilities: Multimodal reasoning, code generation, creative writing.
    - Link: https://gemini.google.com
4. **Cohere**

    - Capabilities: Custom NLP model training, embeddings, and text generation.
    - Link: https://cohere.com

5. **Writer.com**
    - Capabilities: Enterprise-focused text generation with brand voice control.
    - Link: https://writer.com

🎬Text-to-Video

These tools generate short videos from text prompts.

1. **Runway Gen-2**
    - Capabilities: Text-to-video, video editing, style transfer.
    - Link: https://runwayml.com
2. **Pika Labs**
    - Capabilities: AI-generated video from prompts, with cinematic effects.
    - Link: https://pika.art
3. **Synthesia**
    - Capabilities: Text-to-video with AI avatars for corporate or educational content.
    - Link: https://www.synthesia.io
4. **DeepBrain AI Studios**
    - Capabilities: AI avatar videos from scripts or text inputs.
    - Link: https://www.deepbrain.io

🎧 Audio-to-Text (Speech Recognition)

These tools convert spoken audio into written text.

1. **Whisper by OpenAI**
    - Capabilities: High-quality speech-to-text in multiple languages.
    - Link: https://github.com/openai/whisper

2. **AssemblyAI**
    - Capabilities: Transcription, summarization, speaker detection.
    - Link: https://www.assemblyai.com
3. **Descript**
    - Capabilities: Audio/video transcription, editing, and podcasting.
    - Link: https://www.descript.com
4. **Google Speech-to-Text API**
    - Capabilities: Real-time transcription, word-level timestamps.
    - Link: https://cloud.google.com/speech-to-text
5. **Rev AI**
    - Capabilities: High-accuracy speech recognition and transcription.
    - Link: https://www.rev.ai

🖼️ Image Generation (Text-to-Image AI)

1. **Midjourney**
    - Capabilities: High-quality, artistic, detailed image generation from prompts.
    - Link: https://www.midjourney.com
2. **DALL·E 3 (by OpenAI)**
    - Capabilities: Realistic and creative image generation from text prompts.
    - Link: https://openai.com/dall-e
3. **Stable Diffusion (by Stability AI)**
    - Capabilities: Open-source text-to-image generation, customizable models.
    - Link: https://stability.ai/stable-diffusion
4. **Adobe Firefly**
    - Capabilities: Image generation, vector generation, text effects using AI.
    - Link: https://firefly.adobe.com

💻 Code Generation (Text-to-Code AI)

1. **GitHub Copilot**
    - Capabilities: Code autocompletion, code snippets generation, documentation help.
    - Link: https://github.com/features/copilot
2. **Tabnine**
    - Capabilities: AI code completion for multiple languages and IDEs.
    - Link: https://www.tabnine.com
3. **Replit Ghostwriter**
    - Capabilities: Code suggestion, debugging help, entire project creation.
    - Link: https://replit.com/ghostwriter
4. **Codeium**
    - Capabilities: Free AI-powered code completion across languages.
    - Link: https://codeium.com

🎵 Music Generation (Text-to-Music AI)

1. **Suno AI**
    - Capabilities: Create full songs (music + vocals) from a text description.
    - Link: https://suno.ai
2. **Boomy**
    - Capabilities: Generate original songs quickly, even without music knowledge.
    - Link: https://www.boomy.com
3. **AIVA (Artificial Intelligence Virtual Artist)**
    - Capabilities: Compose original music (soundtracks, classical, jazz, pop).
    - Link: https://www.aiva.ai
4. **Soundraw**

- Capabilities: Create and customize music based on mood and theme.
- Link: https://soundraw.io

Industry-specific use cases for generative AI

Finance. Generative AI could add $200 billion to $340 billion in annual value to banking, chiefly through productivity gain, according to consulting firm McKinsey. GenAI's ability to identify patterns in vast amounts of customer and market data is enabling banks to hyperpersonalize customer service and improve fraud detection.

Legal services. Inundated with generative AI products, the legal industry is learning to effectively and safely use tools that are designed to do everything from legal research and summarising briefs to preparing tax returns, drafting contracts and suggesting legal arguments.

Manufacturing. GenAI models can integrate data from cameras, X-rays and other metrics to identify defective parts and the root causes, accelerating time to insight. Plant operators can query in natural language to get comprehensive reports on internal and external operations.

Education. Generative AI helps teachers and administrators through task automation, including grading assignments, generating quizzes and crafting individualized study programs. GenAI's ability to find answers instantly is challenging educators to rethink teaching methods and focus on higher-order skills, such as critical thinking and problem-solving, as well as address the ethical issues posed by AI use.

What is a large language model (LLM)?

A large language model (LLM) is a type of artificial intelligence (AI) program that can recognize and generate text, among other tasks. LLMs are trained on huge sets of data, hence the name "large." LLMs are built on machine learning, specifically, a type of neural network called a transformer model.

In simpler terms, an LLM is a computer program that has been fed enough examples to be able to recognize and interpret human language or other types of complex data. Many LLMs are trained on data that has been gathered from the

Internet — thousands or millions of gigabytes' worth of text. But the quality of the samples impacts how well LLMs will learn natural language, so an LLM's programmers may use a more curated data set.

LLMs use a type of machine learning called deep learning in order to understand how characters, words, and sentences function together. Deep learning involves the probabilistic analysis of unstructured data, which eventually enables the deep learning model to recognize distinctions between pieces of content without human intervention.

LLMs are then further trained via tuning: they are fine-tuned or prompt-tuned to the particular task that the programmer wants them to do, such as interpreting questions and generating responses, or translating text from one language to another.

Examples of real-world LLMs include ChatGPT (from OpenAI), Bard (Google), Llama (Meta), and Bing Chat (Microsoft). GitHub's Copilot is another example, but for coding instead of natural human language.

Key Features of LLMs

- **Massive Training Data:** LLMs are trained on enormous amounts of text data, which allows them to learn patterns and relationships in language.
- **Deep Learning:** They are typically based on deep learning architectures, such as the Transformer model, which enables them to process and understand complex language structures.
- **Natural Language Processing:**LLMs can perform a wide range of NLP tasks, including:
    - **Text generation:** Creating new text, such as articles, poems, or code. https://chatgpt.com/
    - **Translation:** Translating text from one language to another. https://translate.google.co.in/
    - **Question answering:** Answering questions based on the text they have been trained on. https://whylabs.ai/learning-center/llm-use-cases/question-answering-q-a-systems-with-llms

  - **Sentiment analysis:** Determining the sentiment or emotion expressed in text. https://aws.amazon.com/what-is/sentiment-analysis/#:~:text=Sentiment analysis is the process,social media comments%2C and reviews.
  - **Chatbots:** Engaging in conversational interactions.

How LLMs Work

1. **Input:** You provide the LLM with a text input (e.g., a question, a piece of text, a prompt).
2. **Processing:** The LLM uses its learned knowledge and the Transformer architecture to process the input.
3. **Output:** The LLM generates an output based on the input and its understanding of language.

Examples of LLMs: GPT-3, GPT-4, BERT, PaLM, FLAN

Applications of LLMs

- **Content creation:** Generating articles, blog posts, and other written content.
- **Customer service:** Powering chatbots and virtual assistants.
- **Translation:** Translating documents and websites.
- **Education:** Creating personalised learning experiences.
- **Research:** Analyzing large datasets of text.
- **Coding:** Assisting with code generation and debugging.

How do you build a generative AI model?

A generative AI model starts by efficiently encoding a representation of what you want to generate. For example, a generative AI model for text might begin by finding a way to represent the words as vectors that characterise the similarity between words often used in the same sentence or that mean similar things.

Recent progress in LLM research has helped the industry implement the same process to represent patterns found in images, sounds, proteins, DNA, drugs and

3D designs. This generative AI model provides an efficient way to represent the desired type of content and iterate on useful variations.

How do you train a generative AI model?

The generative AI model needs to be trained for a particular use case. The recent progress in LLMS provides an ideal starting point for customising applications for different use cases. For example, the popular GPT model developed by OpenAI has been used to write text, generate code and create imagery based on written descriptions.

Training involves tuning the model's parameters for different use cases and then fine-tuning the results on a given set of training data. For example, a call center might train a chatbot against the kinds of questions AI agents get from various customer types and the responses that service agents give in return. An image-generating app, in distinction to text, might start with labels that describe the content and style of images to train the model to generate new images.

What's the difference between large language models and generative AI?

Large language models are a type of generative AI designed for linguistic tasks, such as text generation, question answering, and summarisation. The broad category of generative AI includes a variety of model architectures and data types, such as video, images, and audio.

What are the limitations of generative AI?

- The app does not always identify the source of content and sometimes points to plausible but erroneous sources of information.
- It can be challenging to assess the biases of original sources.
- Realistic-sounding content makes it harder to identify inaccurate information.
- It can be difficult to understand how to tune for new circumstances.
- The app lacks the capability for original thought.

Introduction to Prompt Engineering

What is Prompt Engineering?

Prompt engineering is the process of crafting queries or instructions to guide an AI model to produce the desired output. A well-designed prompt ensures that the model understands the context, scope, and goal of your request, resulting in outputs that are both relevant and precise.

At its core, prompt engineering is about effective communication with an AI. It involves breaking down complex tasks into understandable instructions and optimising the phrasing to maximise clarity and relevance.

Why is Prompt Engineering Important?

1. **Maximise AI Potential**: AI models are only as good as the instructions they receive. Well-crafted prompts unlock the full capabilities of these models.

2. **Reduce Iterations**: Clear prompts minimise the need for repetitive queries and corrections.

3. **Cost Efficiency**: Many AI services charge based on usage. Precise prompts reduce the computational load by avoiding unnecessary back-and-forth queries.

4. **Versatility**: From generating creative content to solving technical problems, prompt engineering makes AI adaptable to a wide range of use cases.

Key Principles of Effective Prompt Engineering

1. Be Specific and Clear

Ambiguous prompts lead to ambiguous responses. Clearly define the task, context, and format you expect. For example:

- **Unclear Prompt**: "Tell me about technology."
- **Clear Prompt**: "Write a 200-word article about the impact of AI on modern healthcare."

2. Use Context

Providing context ensures the AI understands the background of your request. Context can include roles, scenarios, or objectives.

- **Without Context**: "List some challenges."
- **With Context**: "List the challenges faced by AI in ethical decision-making."

3. Set Constraints

If your task requires specific conditions (e.g., word limits, tone, or format), include them in the prompt.

- **Without Constraints**: "Summarize this article."
- **With Constraints**: "Summarize this article in 100 words, focusing on the key findings.

4. Iterate and Refine

Sometimes, the first attempt might not yield the desired result. Analyze the AI's output and refine your prompt accordingly.

- **Initial Prompt**: "Explain neural networks."
- **Refined Prompt**: "Explain neural networks in simple terms for a high school student."

5. Leverage Examples

For complex tasks, examples act as templates that guide the AI's response.

- **Without Example**: "Generate a code snippet to sort a list."
- **With Example**: "Generate a Python code snippet to sort a list. Example: Input = [4, 2, 9]; Output = [2, 4, 9]."

Advanced Techniques in Prompt Engineering

1. **Chain-of-Thought Prompting:** Encourage the AI to provide reasoning steps before delivering the final answer.

- **Basic Prompt**: "What is 23 multiplied by 12?"
- **Chain-of-Thought Prompt**: "Explain the steps to calculate 23 multiplied by 12, then provide the final answer."

2. **Role-Based Prompts:** Assign the AI a specific role to guide its perspective and tone.

- **Basic Prompt**: "Explain the benefits of exercise."
- **Role-Based Prompt**: "You are a fitness trainer. Explain the benefits of exercise to a beginner."

3. **Zero-Shot, One-Shot, and Few-Shot Learning**

- **Zero-Shot**: The AI performs a task without prior examples.
- **One-Shot**: Provide one example before the task.
- **Few-Shot**: Provide multiple examples to guide the AI.
- **Few-Shot Example**:

"Translate the following phrases into Spanish:

- 'Good morning' → 'Buenos días'

- 'How are you?' → '¿Cómo estás?'

Now translate: 'I am fine.'"

4. **Multi-Turn Conversations:** Break a complex task into smaller steps by interacting with the AI iteratively.

- **Initial Prompt**: "Write a blog post about AI."
- **Iterative Approach**:

- "List the key sections for a blog post about AI."

- "Expand on the section 'Applications of AI.'"

Prompting Techniques

1. Zero-Shot Prompting

The model is given a task **without any prior examples**. It relies solely on the prompt to understand and complete the task.

Example:

Prompt: *"Translate this sentence to French: I love programming."*

Output: *"J'adore la programmation."*

2. **One-Shot Prompting**

The model is provided **one example** to guide its response. This helps it infer the expected format and logic.

Example:

Prompt:

"Translate to French: I am happy → Je suis heureux.
*Translate to French: She is smart →"***Output:** *"Elle est intelligente."*

3. **Few-Shot Prompting**

The prompt contains **multiple examples** to help the model learn the pattern or task before solving a new one.

Example:

Prompt:

"Translate to Spanish:
Apple → Manzana
Car → Coche
*Book →"***Output:** *"Libro"*

4. **Chain-of-Thought Prompting**

Encourages the model to **explain its reasoning step-by-step** before giving the final answer, improving logical accuracy.

Example:

Prompt: *"If Sam has 2 apples and buys 3 more, how many apples does he have now? Think step by step."*

Output: *"Sam starts with 2 apples. He buys 3 more, so now he has 2 + 3 = 5 apples."*

Examples of Prompts

Text Summarization

One of the standard tasks in natural language generation is text summarization. Text summarisation can include many different flavours and domains. In fact, one of the most promising applications of language models is the ability to summarize articles and concepts into quick and easy-to-read summaries. Let's try a basic summarization task using prompts.

Let's say you are interested to learn about antibiotics, you could try a prompt like this:

Prompt:

```
Explain antibioticsA:
```

Output:

```
Antibiotics are a type of medication used to treat bacterial infections. They wo
rk by either killing the bacteria or preventing them from reproducing, allowing
the body’s immune system to fight off the infection. Antibiotics are usually tak
en orally in the form of pills, capsules, or liquid solutions, or sometimes admini
stered intravenously. They are not effective against viral infections, and using
them inappropriately can lead to antibiotic resistance.
```

The "A:" is an explicit prompt format that you use in question answering. You used it here to tell the model that an answer is expected further. In this example, it's not clear how this is useful compared to not using it, but we'll leave it for later examples. Let's just assume that this is too much information and you want to summarise it further. In fact, you can instruct the model to summarize into one sentence like so:

Prompt:

```
Antibiotics are a type of medication used to treat bacterial infections. They wo
rk by either killing the bacteria or preventing them from reproducing, allowing
the body’s immune system to fight off the infection. Antibiotics are usually tak
en orally in the form of pills, capsules, or liquid solutions, or sometimes admini
stered intravenously. They are not effective against viral infections, and using
```

```
them inappropriately can lead to antibiotic resistance.Explain the above in one sentence:
```

Output:

```
Antibiotics are medications used to treat bacterial infections by either killing the bacteria or stopping them from reproducing, but they are not effective against viruses and overuse can lead to antibiotic resistance
```

Information Extraction

While language models are trained to perform natural language generation and related tasks, it's also very capable of performing classification and a range of other natural language processing (NLP) tasks.

Here is an example of a prompt that extracts information from a given paragraph.

Prompt:

```
Author-contribution statements and acknowledgements in research papers should state clearly and specifically whether, and to what extent, the authors used AI technologies such as ChatGPT in the preparation of their manuscript and analysis. They should also indicate which LLMs were used. This will alert editors and reviewers to scrutinize manuscripts more carefully for potential biases, inaccuracies and improper source crediting. Likewise, scientific journals should be transparent about their use of LLMs, for example when selecting submitted manuscripts.Mention the large language model based product mentioned in the paragraph above:
```

Output:

```
The large language model based product mentioned in the paragraph above is ChatGPT.
```

There are many ways you can improve the results above, but this is already very useful.

By now it should be obvious that you can ask the model to perform different tasks by simply instructing it what to do. That's a powerful capability that AI product developers are already using to build powerful products and experiences.

Question Answering

One of the best ways to get the model to respond with specific answers is to improve the format of the prompt. As covered before, a prompt could combine instructions, context, input, and output indicators to get improved results. While these components are not required, it becomes a good practice as the more specific you are with instructions, the better results you will get. Below is an example of how this would look following a more structured prompt.

Prompt:

```
Answer the question based on the context below. Keep the answer short and concise. Respond "Unsure about answer" if not sure about the answer.Context: Teplizumab traces its roots to a New Jersey drug company called Ortho Pharmaceutical. There, scientists generated an early version of the antibody, dubbed OKT3. Originally sourced from mice, the molecule was able to bind to the surface of T cells and limit their cell-killing potential. In 1986, it was approved to help prevent organ rejection after kidney transplants, making it the first therapeutic antibody allowed for human use.Question: What was OKT3 originally sourced from?Answer:
```

Output:

```
Mice.
```

Code Generation

One application where LLMs are quite effective is code generation. Copilot is a great example of this. There are a vast number of code-generation tasks you can perform with clever prompts. Let's look at a few examples below.

First, let's try a simple program that greets the user.

Prompt:

```
/*Ask the user for their name and say "Hello"*/
```

Output:

```
let name = prompt("What is your name?");console.log(`Hello, ${name}!`);
```

Notice that you didn't even need to specify the language to use.

Let's switch levels a bit. The example below shows how powerful LLMs can be with a little more effort in designing the prompts.

Prompt:

```
"""Table departments, columns = [DepartmentId, DepartmentName]Table students, columns = [DepartmentId, StudentId, StudentName]Create a MySQL query for all students in the Computer Science Department"""
```

Output:

```
SELECT StudentId, StudentName FROM students WHERE DepartmentId IN (SELECT DepartmentId FROM departments WHERE DepartmentName = 'Computer Science');
```

This is very impressive. In this case, you provided data about the database schema and asked it to generate a valid MySQL query.

Common Pitfalls to Avoid

1. **Vagueness**: Failing to specify the scope or format can lead to irrelevant outputs.
2. **Overloading the Prompt**: Too much information can confuse the AI.
3. **Ignoring Feedback**: Analyse the AI’s output and adjust your prompt as needed.
4. **Underestimating AI Limitations**: Understand the model’s strengths and weaknesses to set realistic expectations.

- **Content Creation**: Writing articles, poetry, or social media posts.

- **Programming Assistance**: Generating code snippets, debugging, or learning new concepts.
- **Data Analysis**: Summarising datasets, extracting insights, or creating reports.
- **Customer Support**: Automating responses to common queries.
- **Education**: Tutoring, explaining concepts, or creating study materials.

Becoming a Prompt Engineering Expert

1. **Practice Regularly**: Experiment with various prompt styles and tasks.

2. **Learn from Examples**: Study successful prompts and their structures.

3. **Stay Updated**: AI models evolve rapidly. Stay informed about new capabilities and best practices.

4. **Engage in Communities**: Participate in forums and workshops to share insights and learn from others.

Integrating GenAI to Applications

Integrating Generative AI (GenAI) into an app **involves several steps, starting with defining your needs and data requirements, choosing the right AI model and tools, and then creating a development environment, training the model, and finally, integrating the AI into your application**. This process can enhance user experience, automate tasks, and unlock new functionalities.

Strategies for integrating generative AI

1. Align AI implementation with business goals.
2. Prioritise data security and privacy.
3. Curate high-quality training samples.
4. Evaluate and test AI models for inaccuracies, inconsistencies, and biases.
5. Monitor and adapt AI-powered apps to evolving trends and user behaviour.

Prerequisites

Before we start, ensure you have the following:

1. **Node.js and npm**: Ensure you have Node.js installed, which comes with npm.
2. **React Application**: A basic React application setup.
3. **OpenAI API Key**: Sign up at **OpenAI** and get your API key.

Integrating ChatGpt to Web Applications

Step 1: Set Up Your React App

If you don't already have a React app, you can create one using create-react-app:

```
npx create-react-app openai-react-app
cd openai-react-app
```

Step 2: Install Axios

We'll use Axios for making HTTP requests to the OpenAI API. Install Axios using npm:

```
npm install axios
```

Step 3: Create OpenAI Service

Create a new file called openaiService.js in the src directory of your React app. This file will handle the API requests to OpenAI.

Replace YOUR_OPENAI_API_KEY with your actual OpenAI API key.

Step 4: Create a Component to Use OpenAI

Next, create a new component and use the getOpenAIResponse() function to fetch data from OpenAI.

Step 5: Integrate the Component into Your App

Finally, integrate the OpenAIComponent into your main application file (App.js).

Step 6: Run Your React App

Run your React application to see the integration in action:

```
npm start
```

Enhance Your Knowledge

Deepen your understanding with these trusted resources:

- **ChatGPT Explained:** OpenAI's ChatGPT Overview
- **Gemini AI Guide:** Google Gemini Explained
- **History of GenAI:** Timeline of GenAI Advancements
- **LLMs vs GenAI Comparison:** LLM vs GenAI Explained
- OpenAI API Documentation
- React Official Docs
- Explore how GenAI is used in products like **Notion AI**, **GrammarlyGO**, and **GitHub Copilot** for real-world inspiration.

Try It Yourself

Let's dive right into the questions and have a blast answering them together. Don't forget to join the AlmaBetter Community Platform to share your brilliant approaches and cheer each other on! 🚀 Happy learning! 💻😄

Task 1:

💡 *Question:* Explain in your own words how a GenAI model transforms a prompt into an output. Use an example from either text, image, or code generation.

Task 2:

🔍 *Question:* Identify one ethical challenge related to generative AI in your preferred industry (e.g., education, finance, or media). How would you mitigate it?

👉 *Bonus Discussion Prompt (Cohort Activity):*

Explore how GenAI could revolutionize **your field of interest** (e.g., healthcare, music, game design). Share your thoughts in the **AlmaBetter Community Platform** and provide constructive feedback on your peers' responses!

Task 3: Design a prompt to summarize a scientific article into 3 bullet points.

Task 4: Refine a prompt that generates a SQL query from a schema to handle edge cases.

Task 5: Share your best prompt for generating blog titles on the AlmaBetter Community Platform.