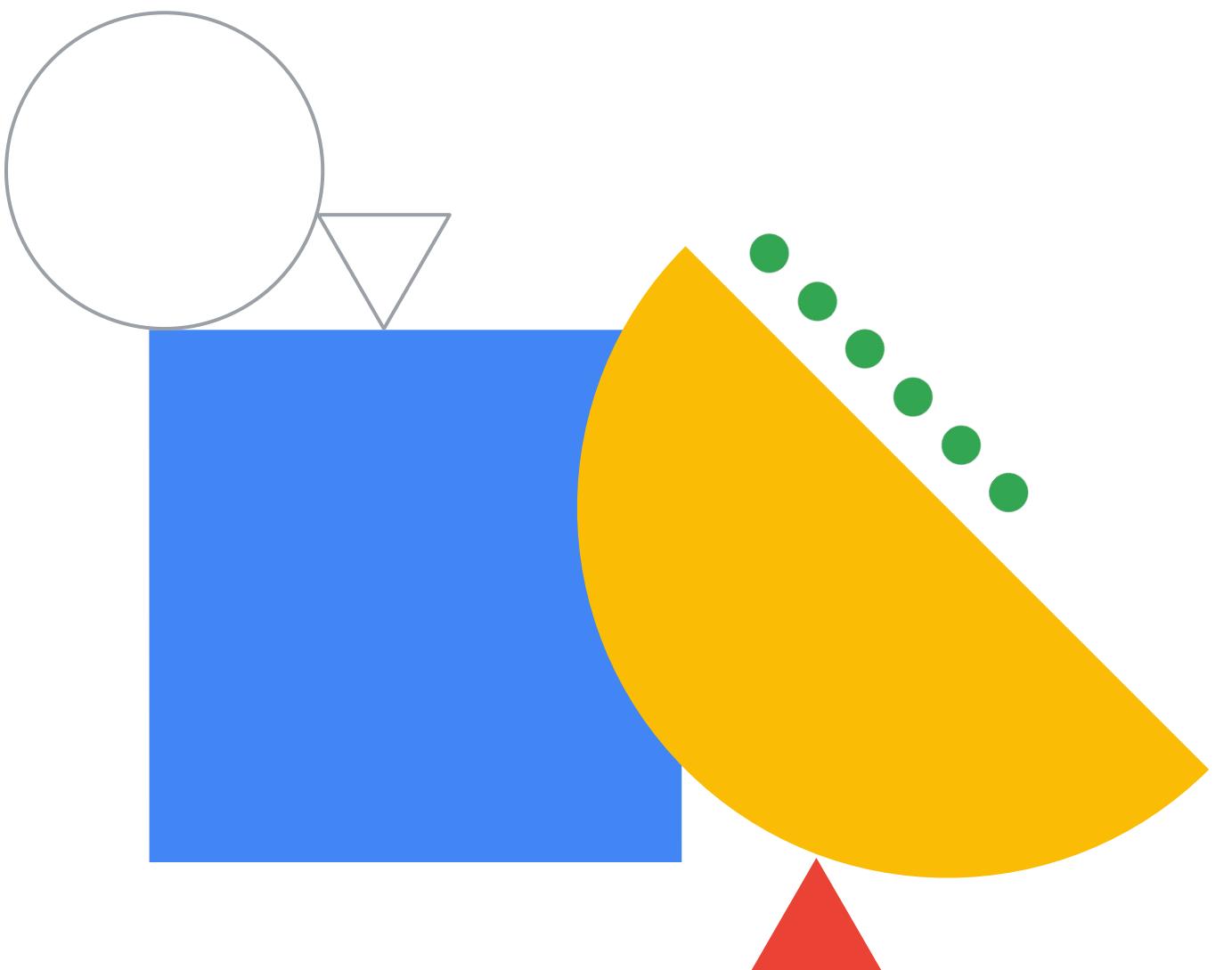
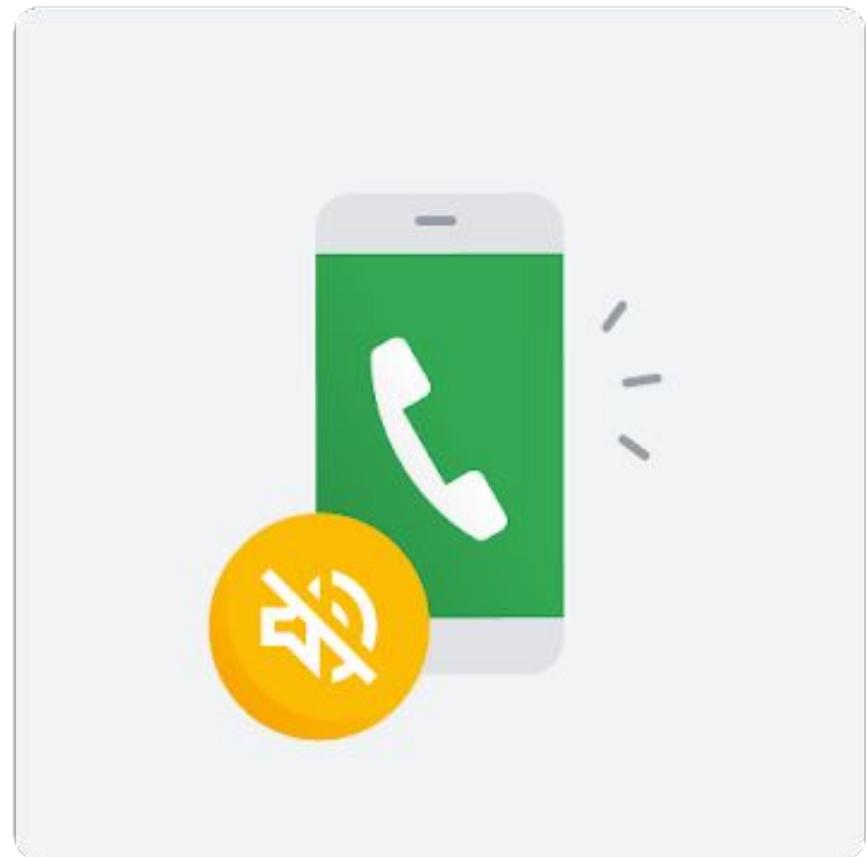


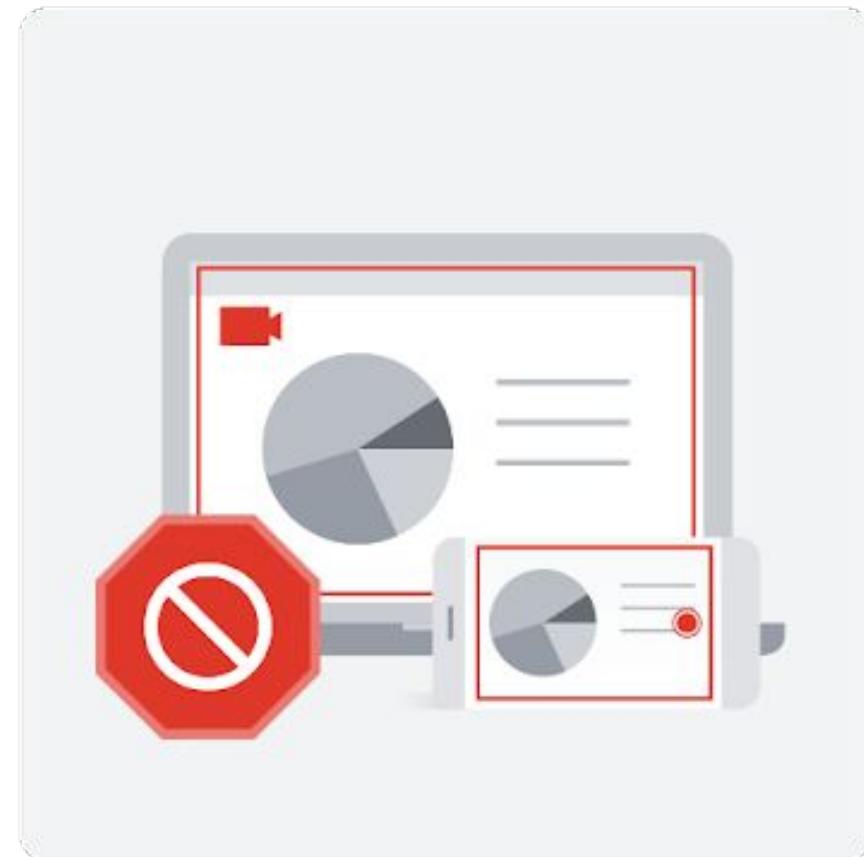
Text Generation for Applications using Gen AI Studio



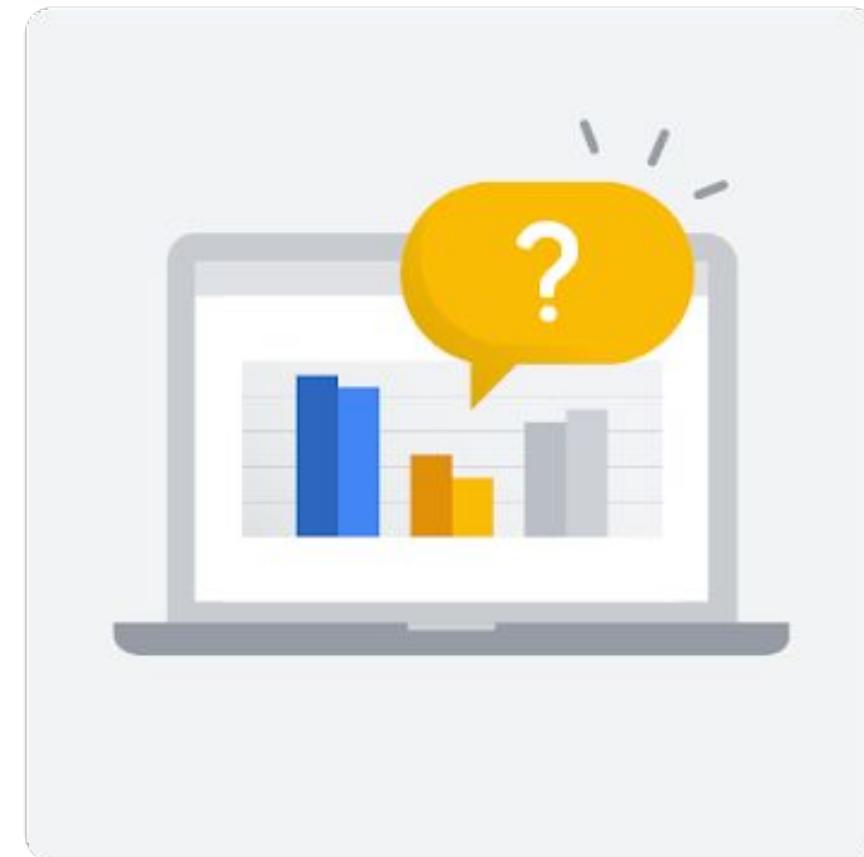
Etiquette



No calls

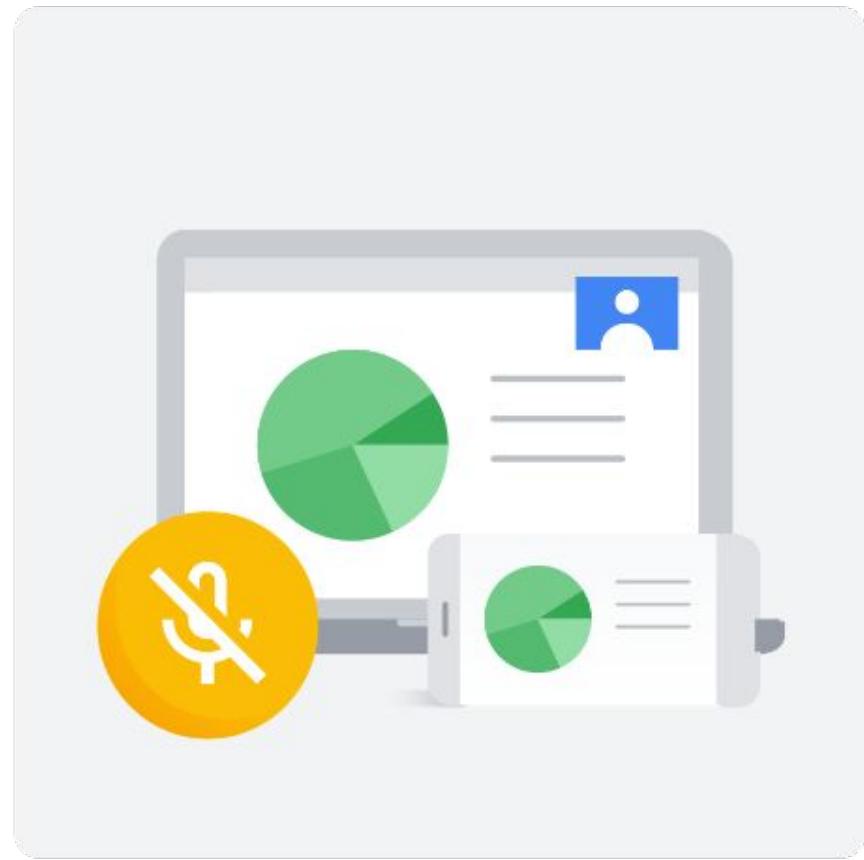


No recording

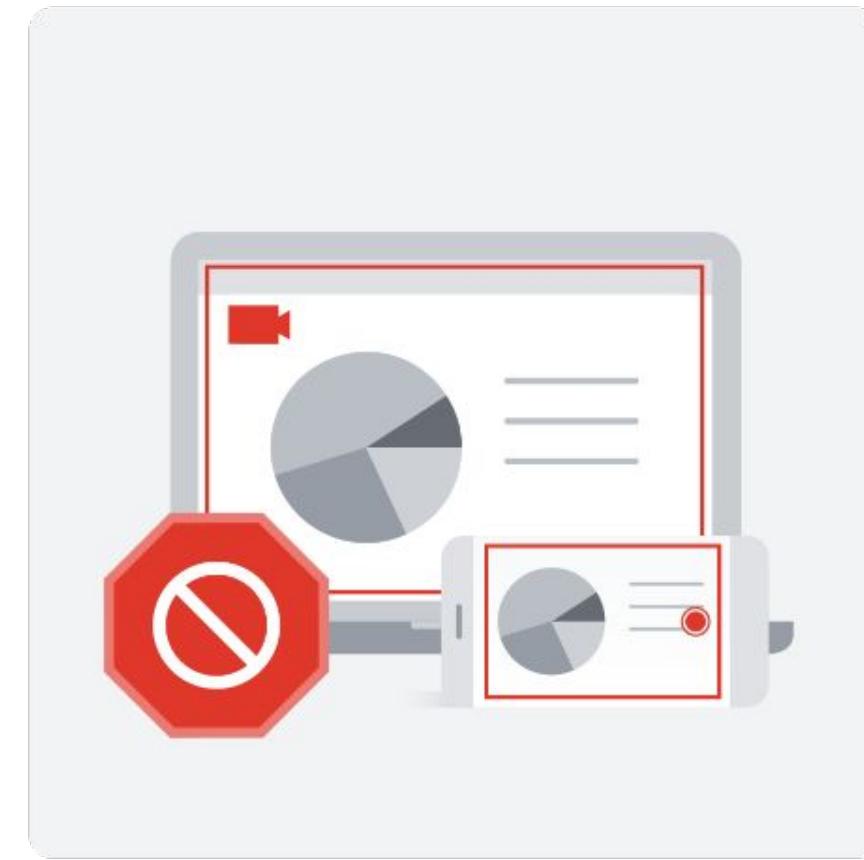


Ask questions

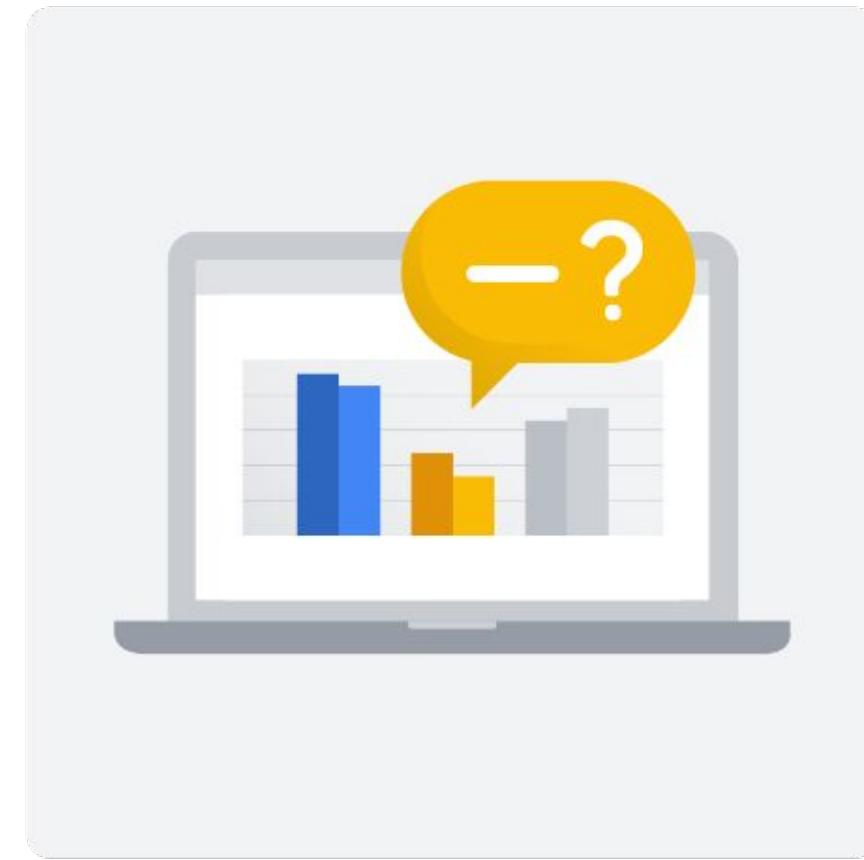
Etiquette



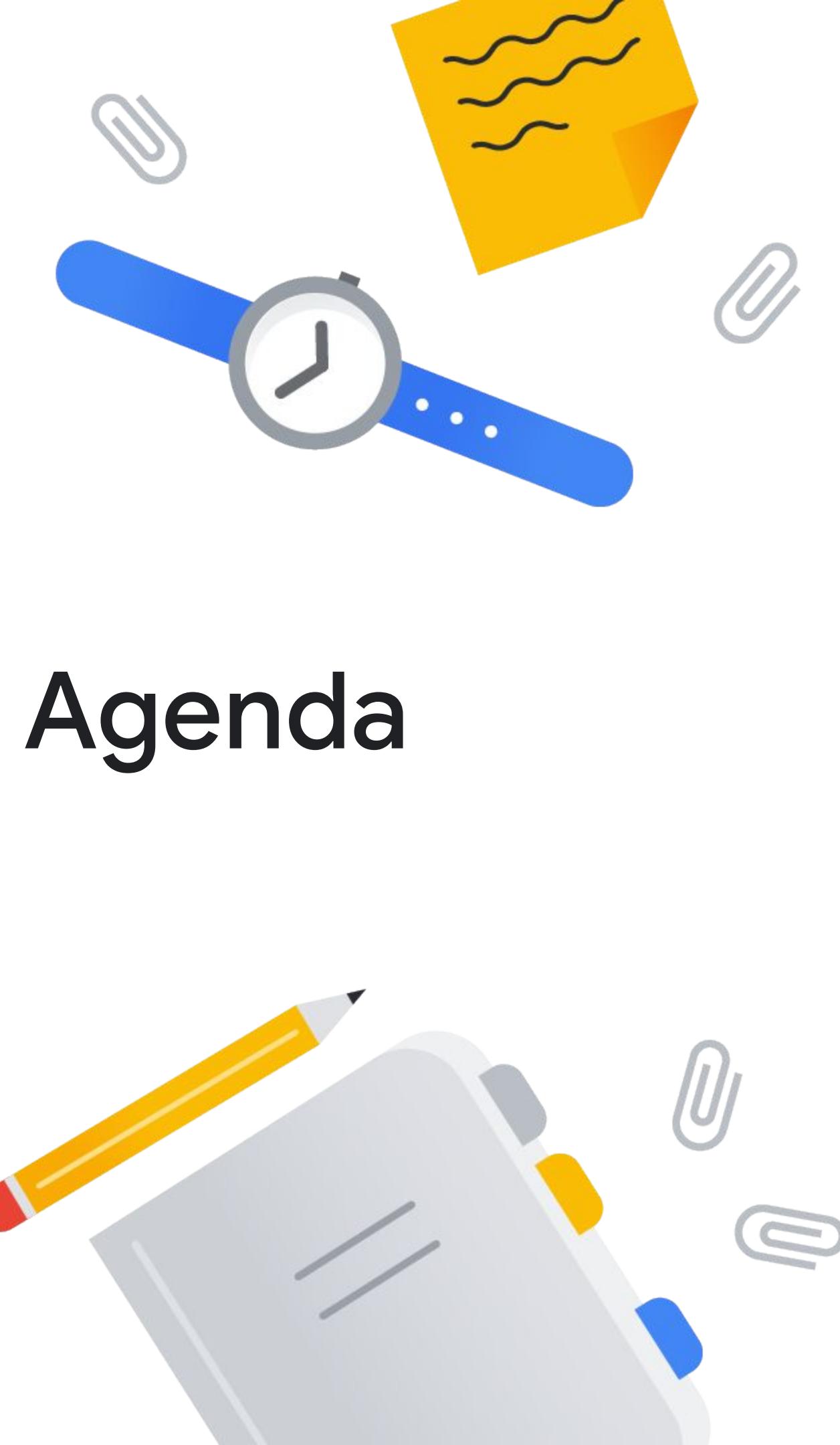
Mute microphone



No recording



Ask questions



Agenda

- 01 Generative AI on Vertex AI
- 02 Gen AI Studio
- 03 Prompt Design
- 04 Implementing the PaLM API
- 05 Fine-tuning Models

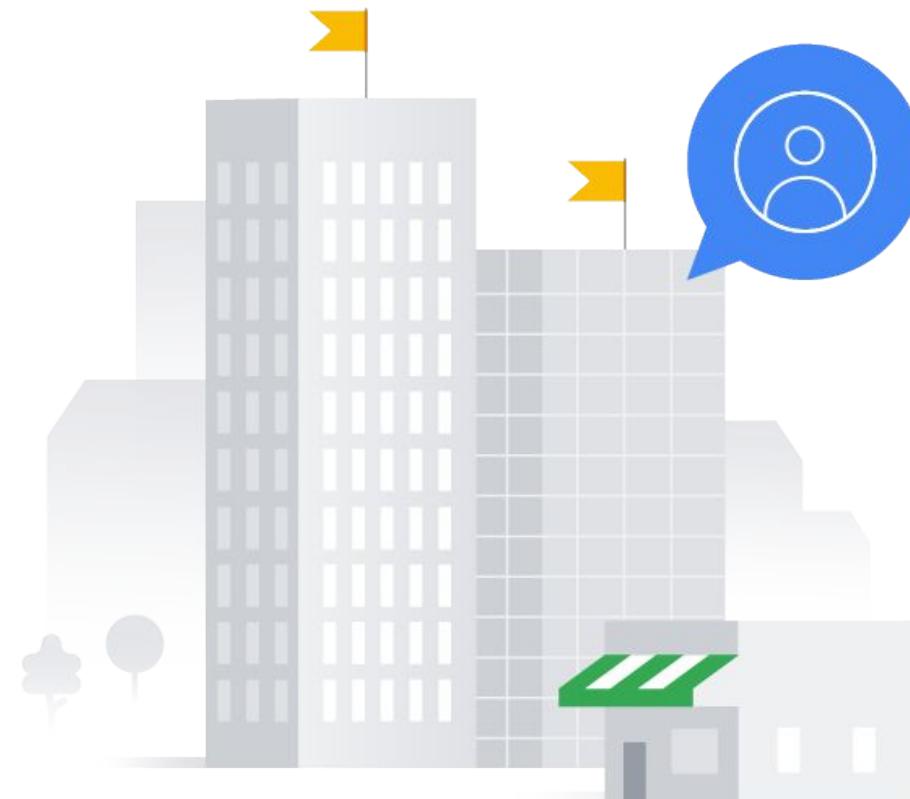
Objectives

- 01 Understand Vertex AI generative AI options for your applications
- 02 Explore Gen AI Studio to interact with foundation models
- 03 Design and tune prompts for your Generative AI use cases
- 04 Implement the PaLM API into your applications using the Python SDK
- 05 Fine-tune foundation model weights to improve model output quality

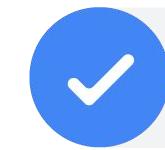


Target audience

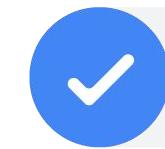
- Those developing using Generative AI features on Google Cloud's Vertex AI platform



ML researchers



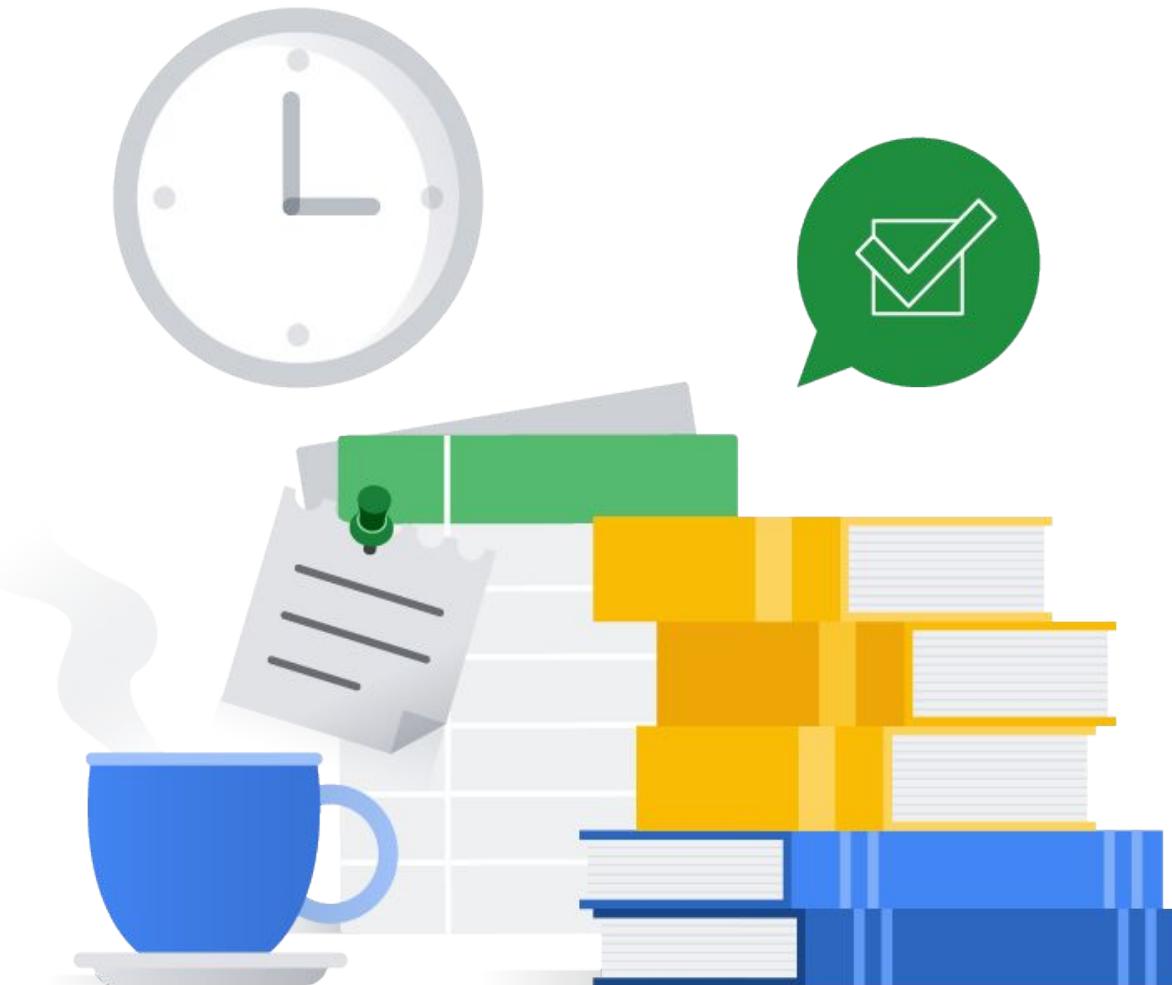
Programmers



App developers

Google Cloud

Helpful knowledge



Google Cloud basics



Helpful to be familiar with



Python programming



Machine learning basics



Leveraging APIs in applications

Lab environment

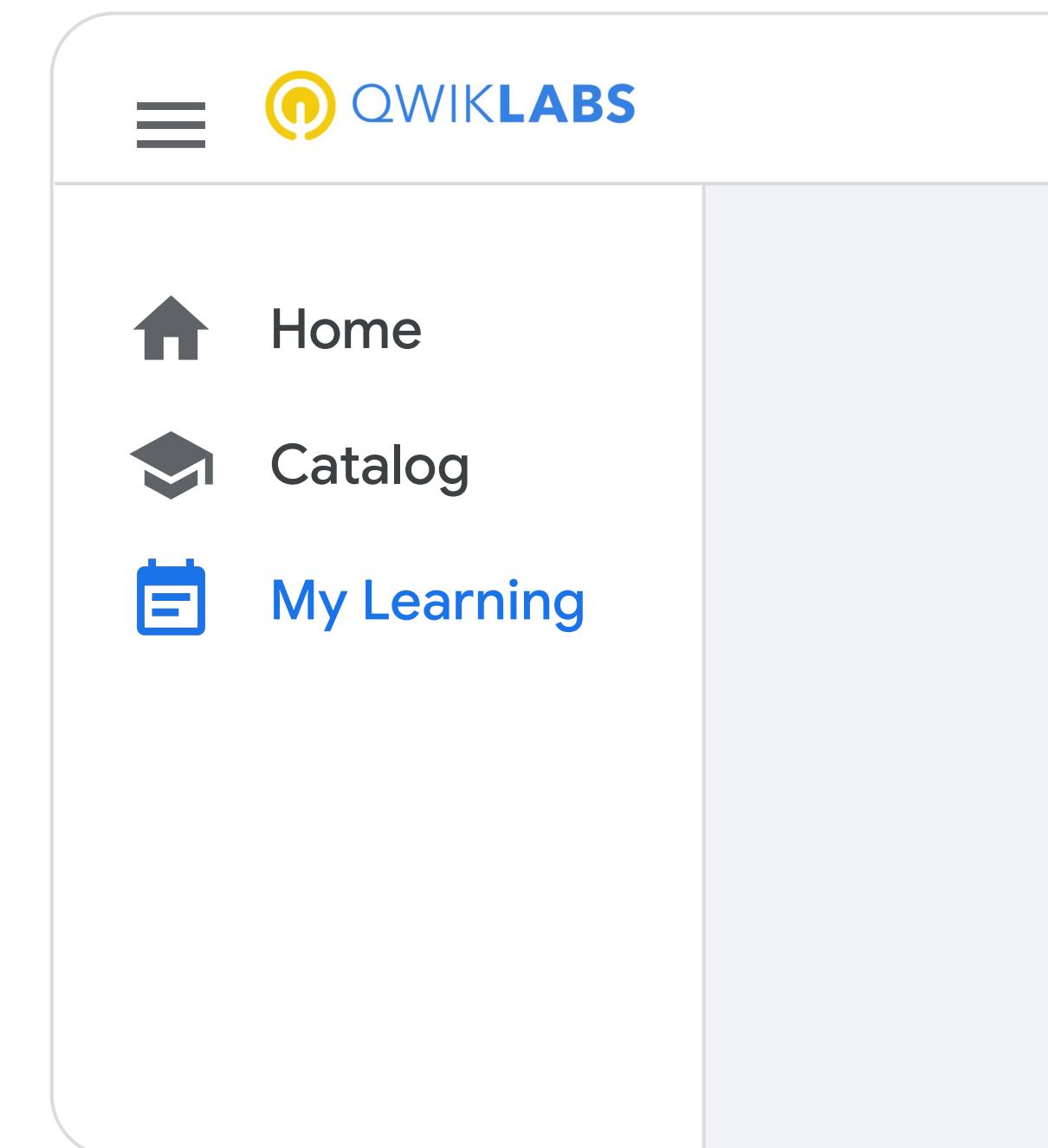
For each lab, Qwiklabs offers:

- A free set of resources for a fixed amount of time
- A clean environment with permissions



Open Qwiklabs

- 1 Open an incognito window (or private/anonymous window).
- 2 Go to the Qwiklabs URL your instructor provides.
- 3 Sign In with existing account or Join with new account (with email you used to register for the course).
- 4 Launch the course from My Learning.

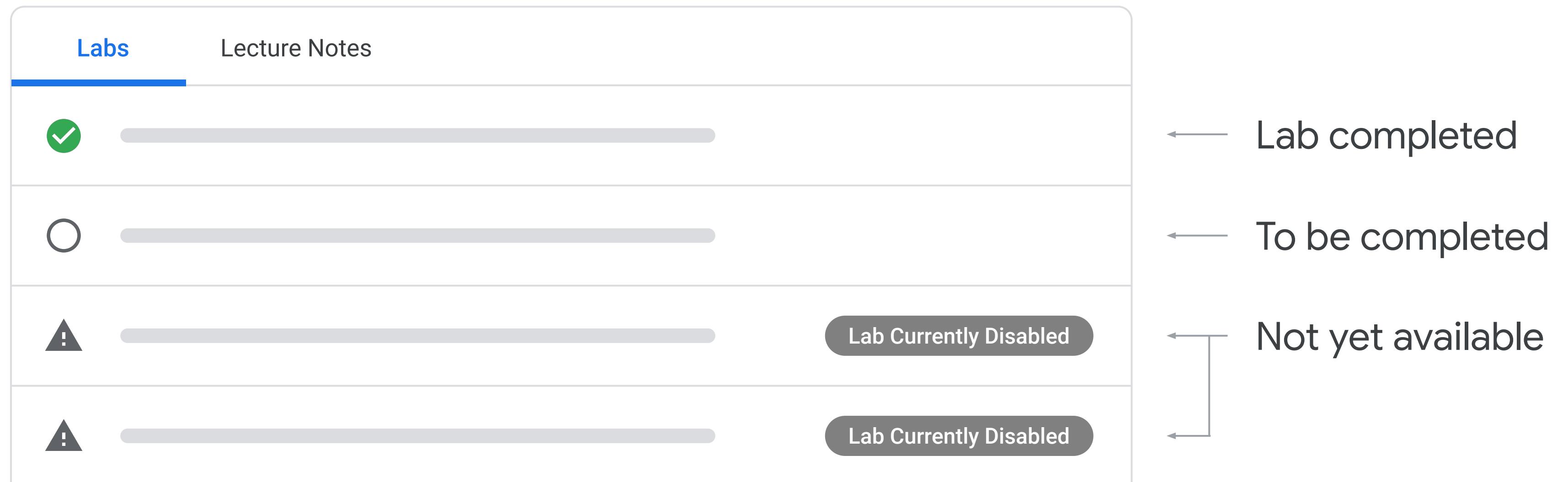


Access issues

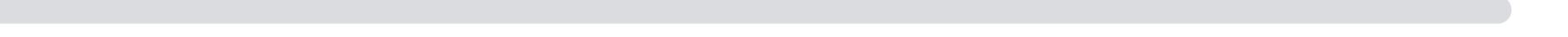
The process to open Qwiklabs can differ based on credentials used. Please reach out to your trainer if you have any access issues.

View your labs

Do **NOT** launch a lab until instructed to do so!



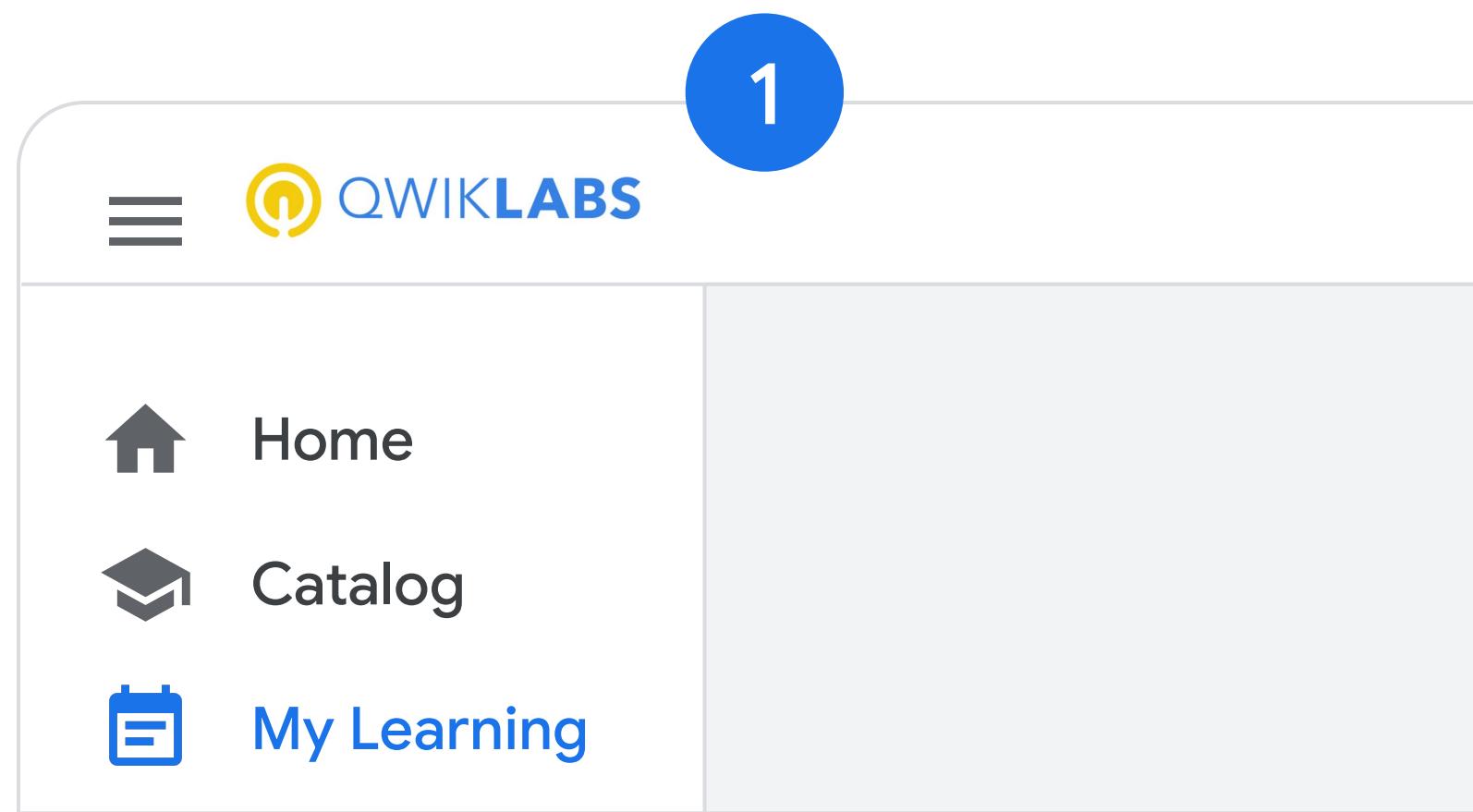
View lecture notes

Labs	Lecture Notes
01	 
02	 
03	 
04	 

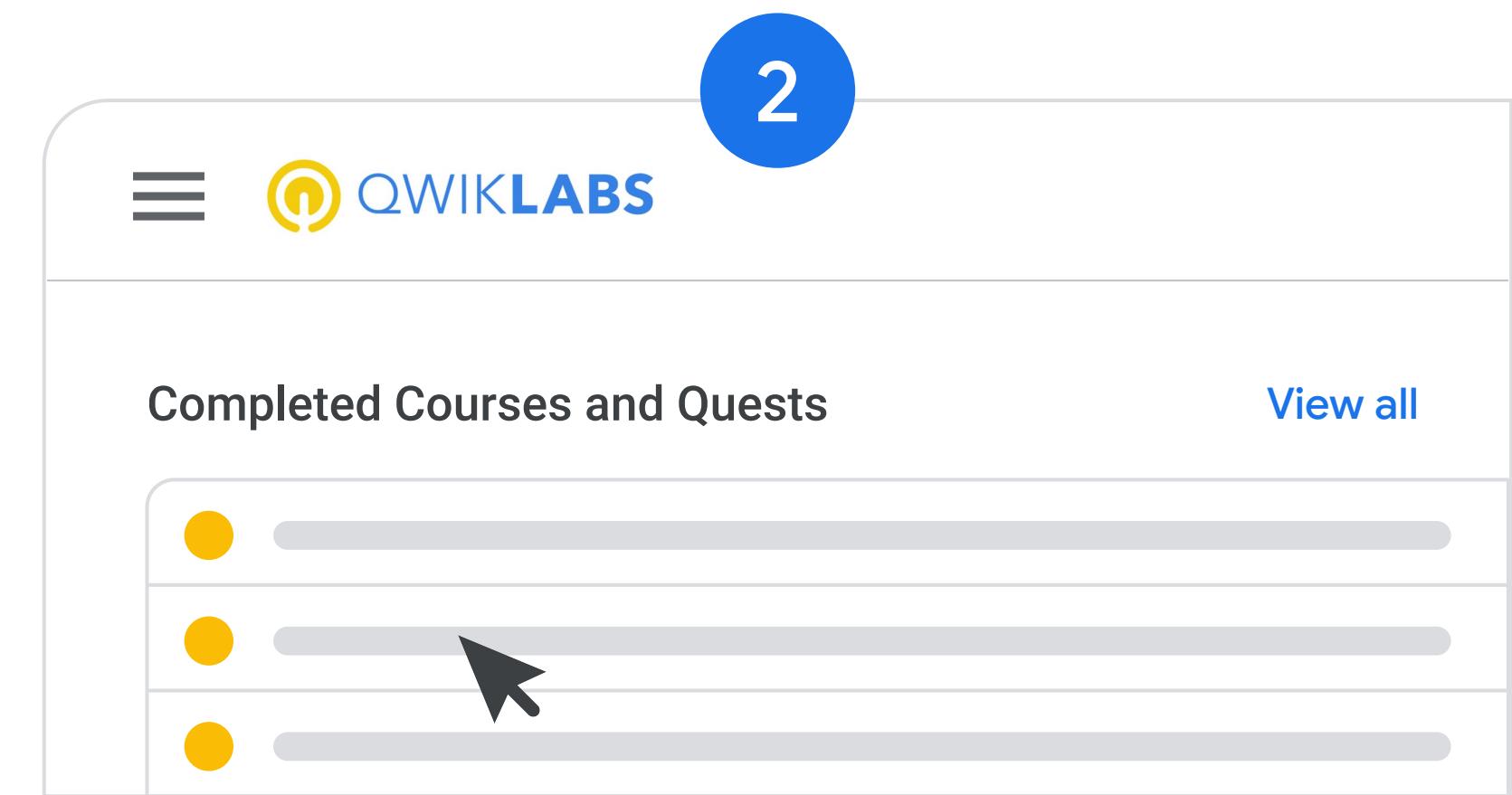
You can download
these as PDF files

End of class - Materials

Materials are available for 2 years



Click on My Learning in the left-hand navigation bar



Select the class from the **Completed Courses** list



Generative AI on Vertex AI

In this module, you learn to ...

- 01 Differentiate between machine learning in general and generative AI
- 02 Automate ML tasks using Vertex AI on Google Cloud
- 03 Choose from the available Generative AI options on Google Cloud
- 04 Explore the text generation AI use case

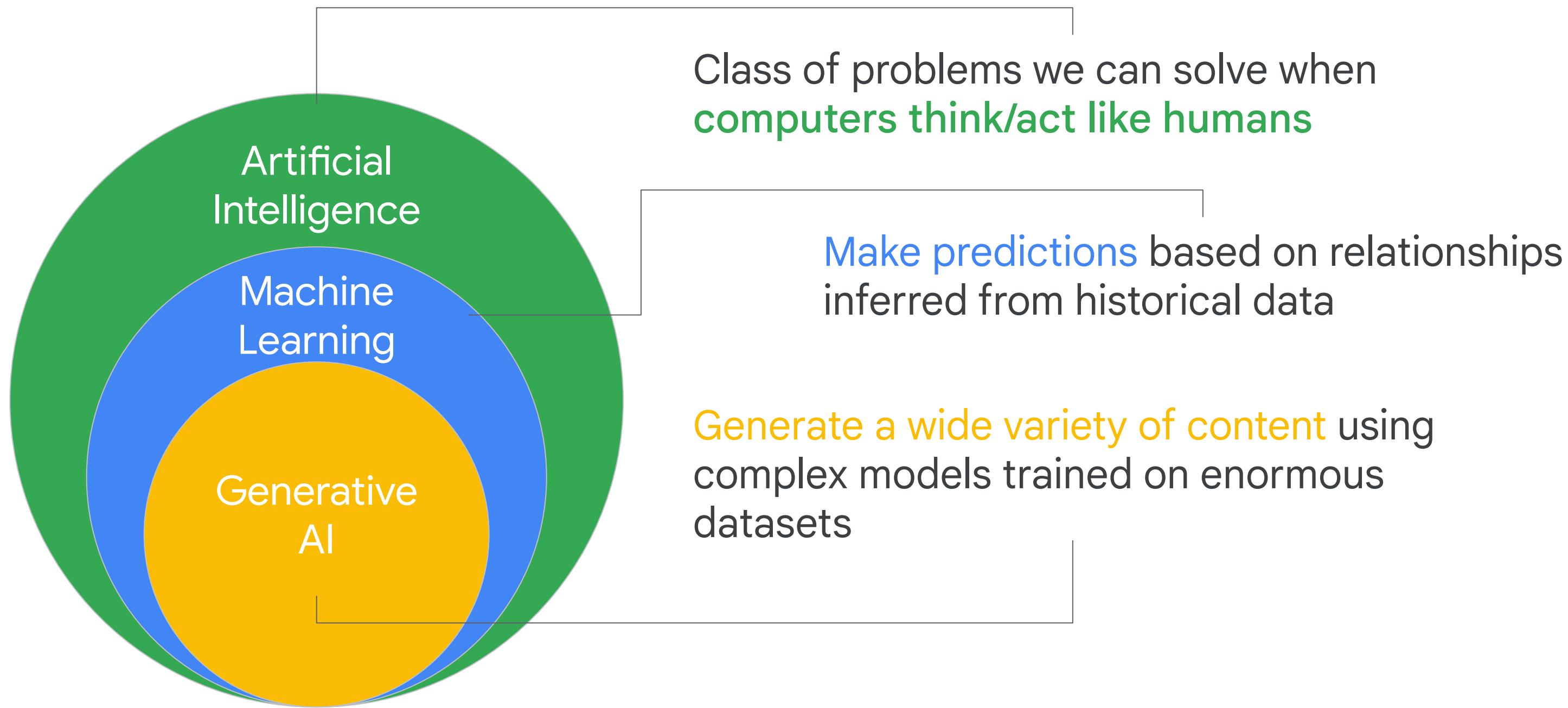


Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case



Machine Learning is a type of AI, and Generative AI (GenAI) is a type of machine learning



Machine learning allows computers to learn without explicit programming

- In traditional programming, the programmer writes the code to perform a task
- In machine learning, algorithms are trained to make predictions using historical data
 - Computers iterate over the algorithm making adjustments to find the best solution

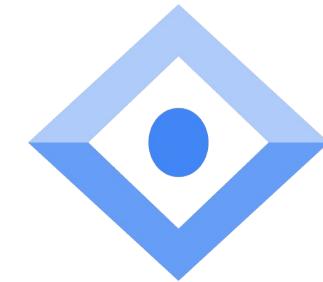


Machine learning use cases include:

- Image recognition
- Sentiment analysis
- Speech recognition
- Fraud detection
- Customer segmentation
- Recommendation systems



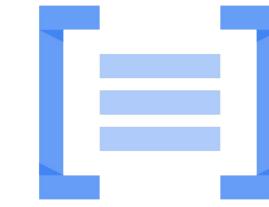
Google has been a pioneer in machine learning for many years



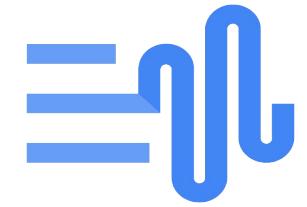
Cloud
Vision API



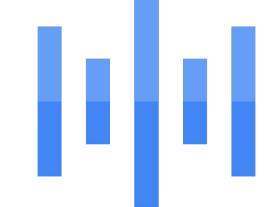
Cloud
Translation API



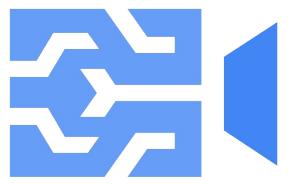
Natural
Language API



Text to
Speech



Speech
to Text



Video
Intelligence API

Google Vision API tells you what's in a picture

- Detects objects and landmarks
- Labels photos
- Recognizes text
- Detects inappropriate content

Landmarks Labels Text Properties Safe Search

Stonehenge 85%

Stonehenge Cursus

A303

Google Map data ©2023

Location: E1 170000 N 1 026215

IMG_20180218_131834.jpg

<https://cloud.google.com/vision/docs/drag-and-drop>

Google Cloud

You can use Google ML to translate between different languages

The screenshot shows the Google Translate web interface. At the top, there's a navigation bar with the Google Translate logo, a menu icon, and a user profile icon. Below the navigation bar are four tabs: Text, Images, Documents, and Websites. The Text tab is selected.

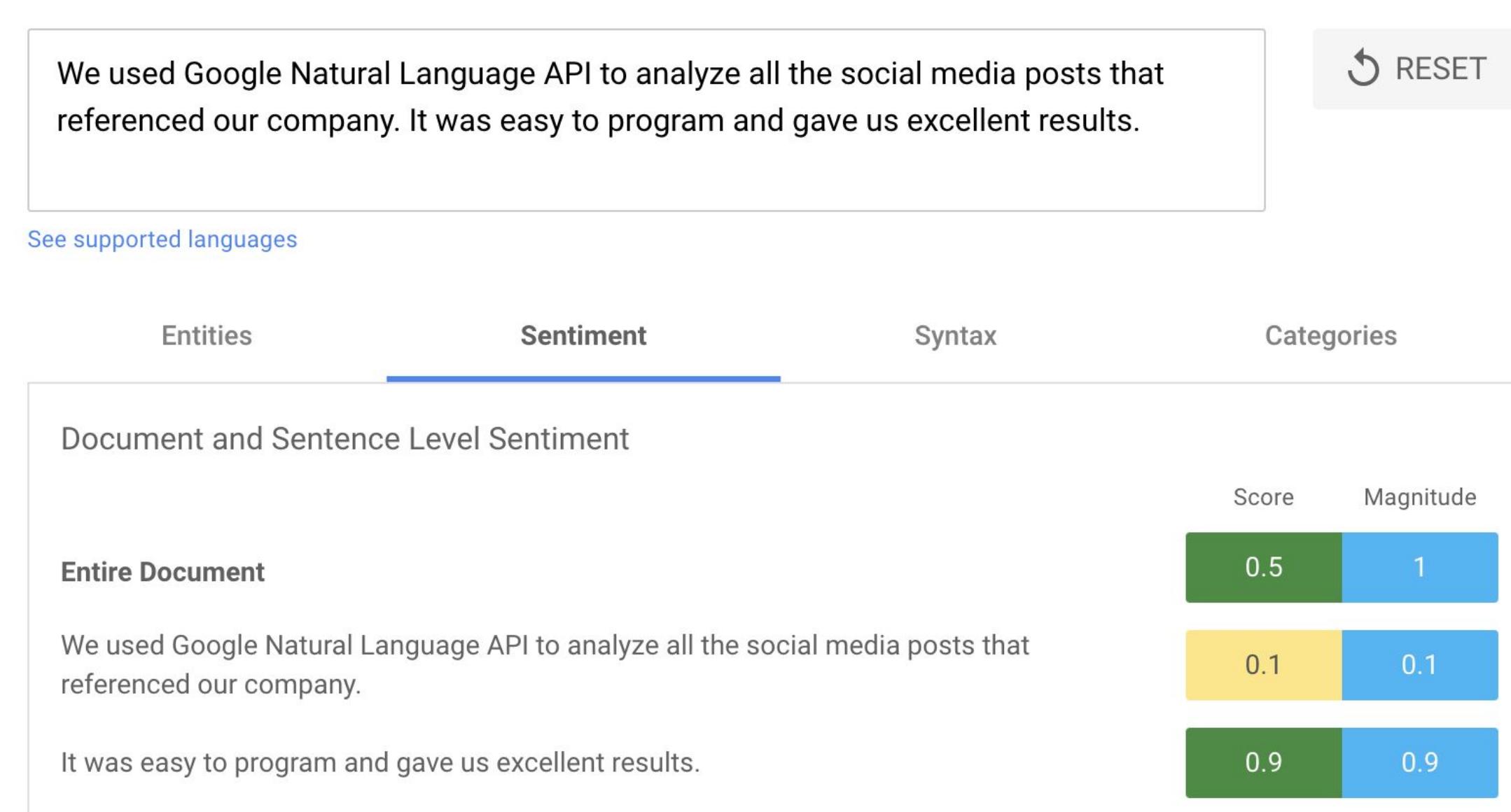
Below the tabs, there are two language detection sections. The left section shows "Detect language" followed by a dropdown menu with "English" selected, "Spanish", and "French". The right section shows a double-headed arrow icon and a dropdown menu with "Italian", "English" (selected), and "Spanish".

The main content area displays a text input and its translation. On the left, the English text is: "Google Translate is a multilingual neural machine translation service developed by Google to translate text, documents and websites from one language into another." On the right, the Spanish translation is: "Google Translate es un servicio de traducción automática neuronal multilingüe desarrollado por Google para traducir texto, documentos y sitios web de un idioma a otro." There are also icons for microphone, speaker, and sharing.

<https://translate.google.com/>

Google Cloud

Natural Language API can analyze text, extract entities, and perform sentiment analysis



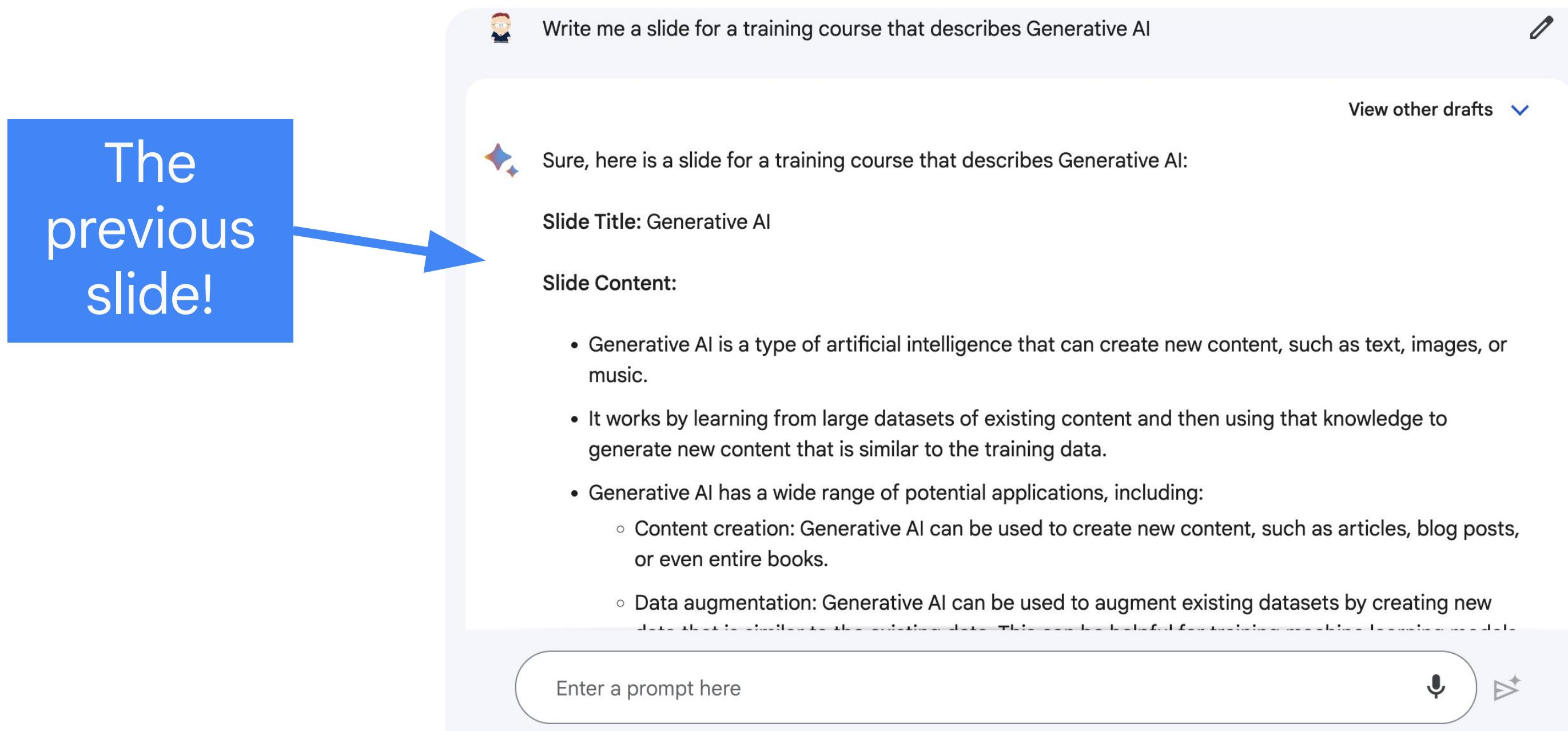
<https://cloud.google.com/natural-language/>

Google Cloud

Generative AI

- Generative AI is a type of artificial intelligence that can create new content, such as text, images, or music
- It works by learning from large datasets of existing content and then using that knowledge to generate new content that is similar to the training data
- Generative AI has a wide range of potential applications, including:
 - Content creation: Generative AI can be used to create new content, such as articles, blog posts, or even entire books.
 - Data augmentation: Generative AI can be used to augment existing datasets by creating new data that is similar to the existing data. This can be helpful for training machine learning models.
 - Creative applications: Generative AI can be used to create new and innovative art, music, and other creative content.

Bard is a Google consumer-oriented generative AI tool for creating custom text-based content



The previous slide!

Write me a slide for a training course that describes Generative AI

Sure, here is a slide for a training course that describes Generative AI:

Slide Title: Generative AI

Slide Content:

- Generative AI is a type of artificial intelligence that can create new content, such as text, images, or music.
- It works by learning from large datasets of existing content and then using that knowledge to generate new content that is similar to the training data.
- Generative AI has a wide range of potential applications, including:
 - Content creation: Generative AI can be used to create new content, such as articles, blog posts, or even entire books.
 - Data augmentation: Generative AI can be used to augment existing datasets by creating new

Enter a prompt here

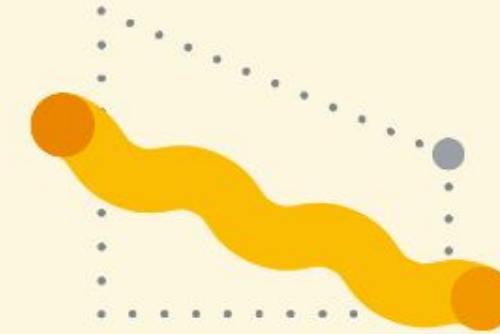
<https://bard.google.com/>

Google Cloud

There are huge benefits of generative AI



Increase in efficiency & productivity



Reduce costs for your organization



Automate monotonous tasks

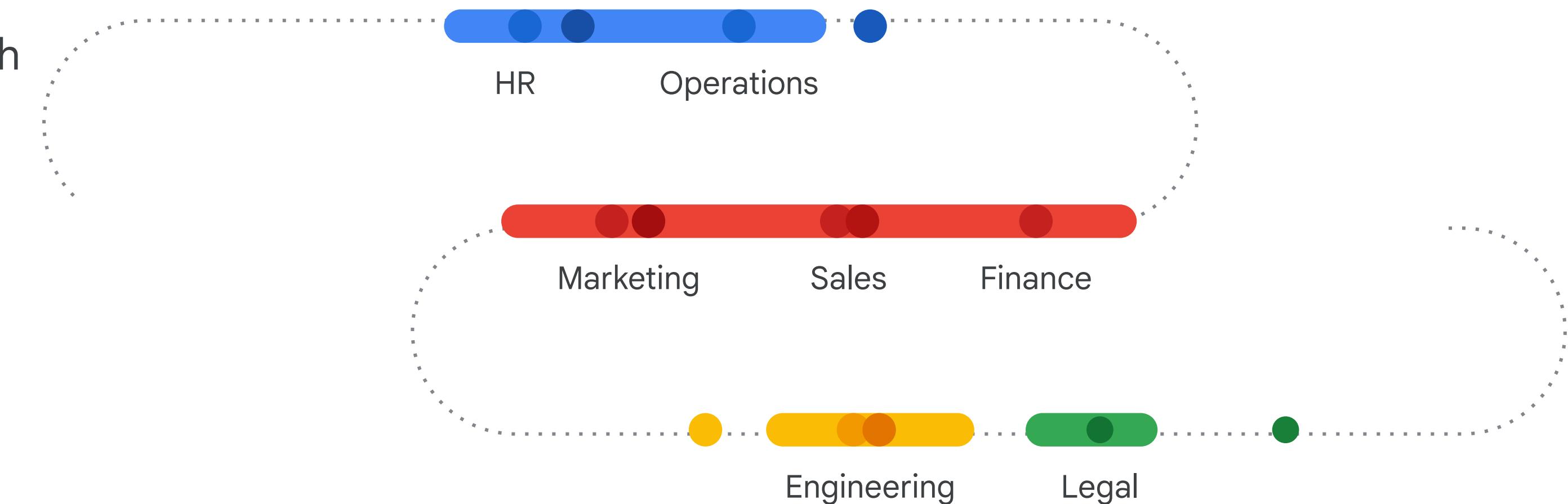
There are also challenges with generative AI

- Can be difficult to control the quality of generated content
- Can be difficult to ensure that generated content is accurate
 - Untrue statements can be presented in a confident manner
 - These are known as hallucinations in generative AI terms
- Can be difficult to ensure that generated content is not offensive or harmful



There are many potential use-cases for generative AI across all aspects of an organization

- Content creation
- Marketing and advertising
- Customer service
- Education and research
- Many more...



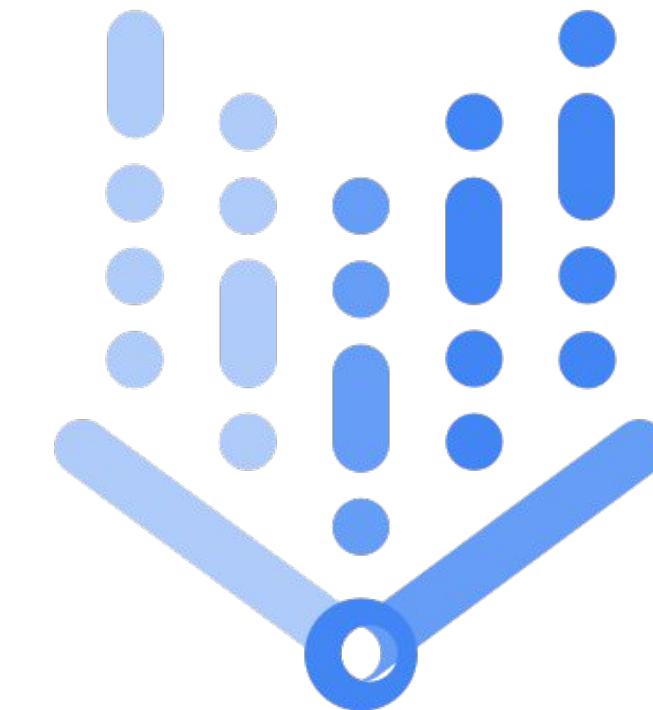
Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case



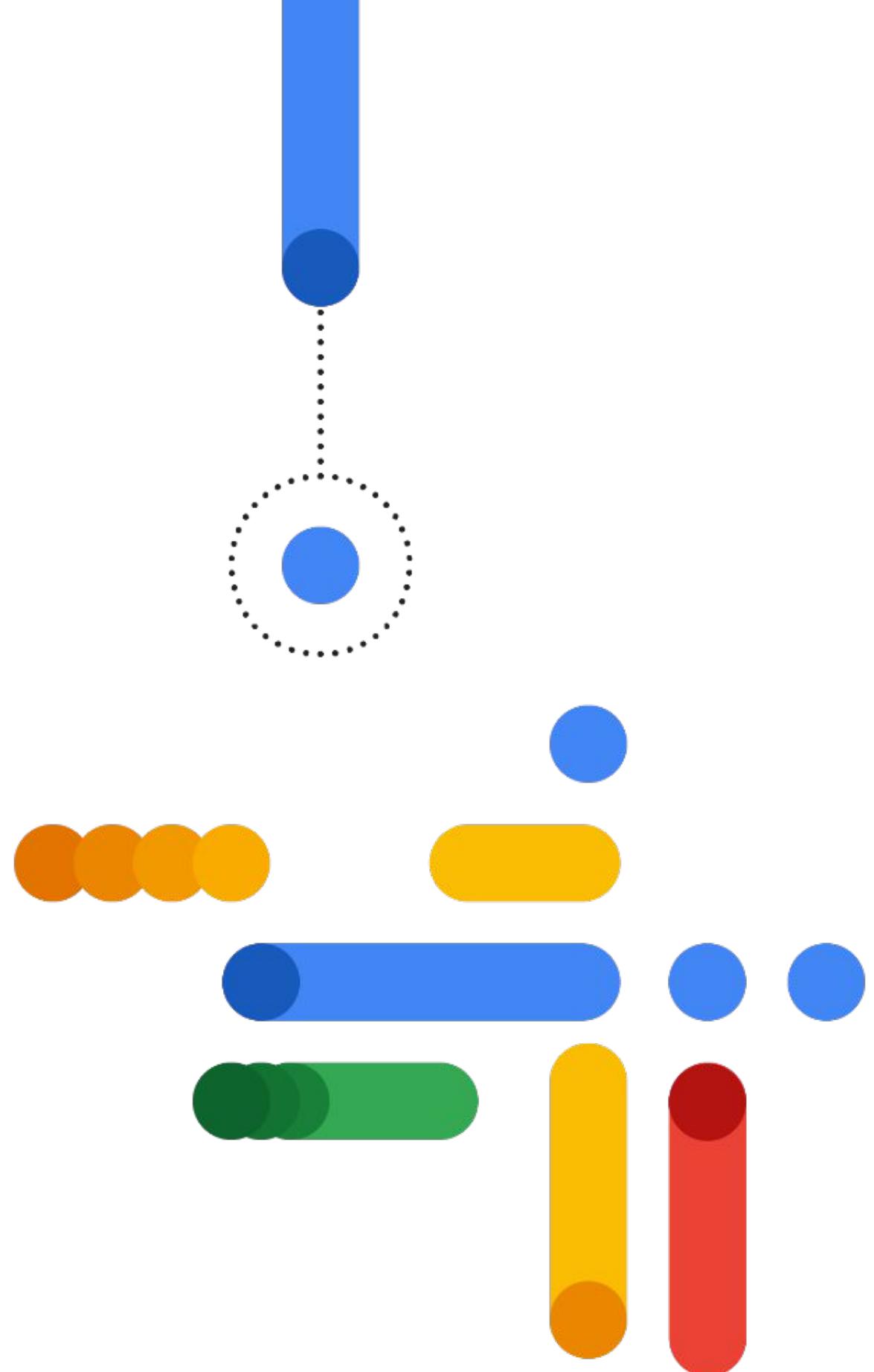
Vertex AI is a machine learning platform that helps you build, deploy, and manage ML models

- Combines data engineering, data science, and ML engineering workflows
- Provides several options for model training and deployment
 - AutoML
 - Custom Training
 - Model Garden
 - Generative AI Studio
- Uses serverless infrastructure that you can customize based on your performance and budget needs
- Supports Python, the Google Cloud Console, the gcloud command line tool, client libraries, and Terraform



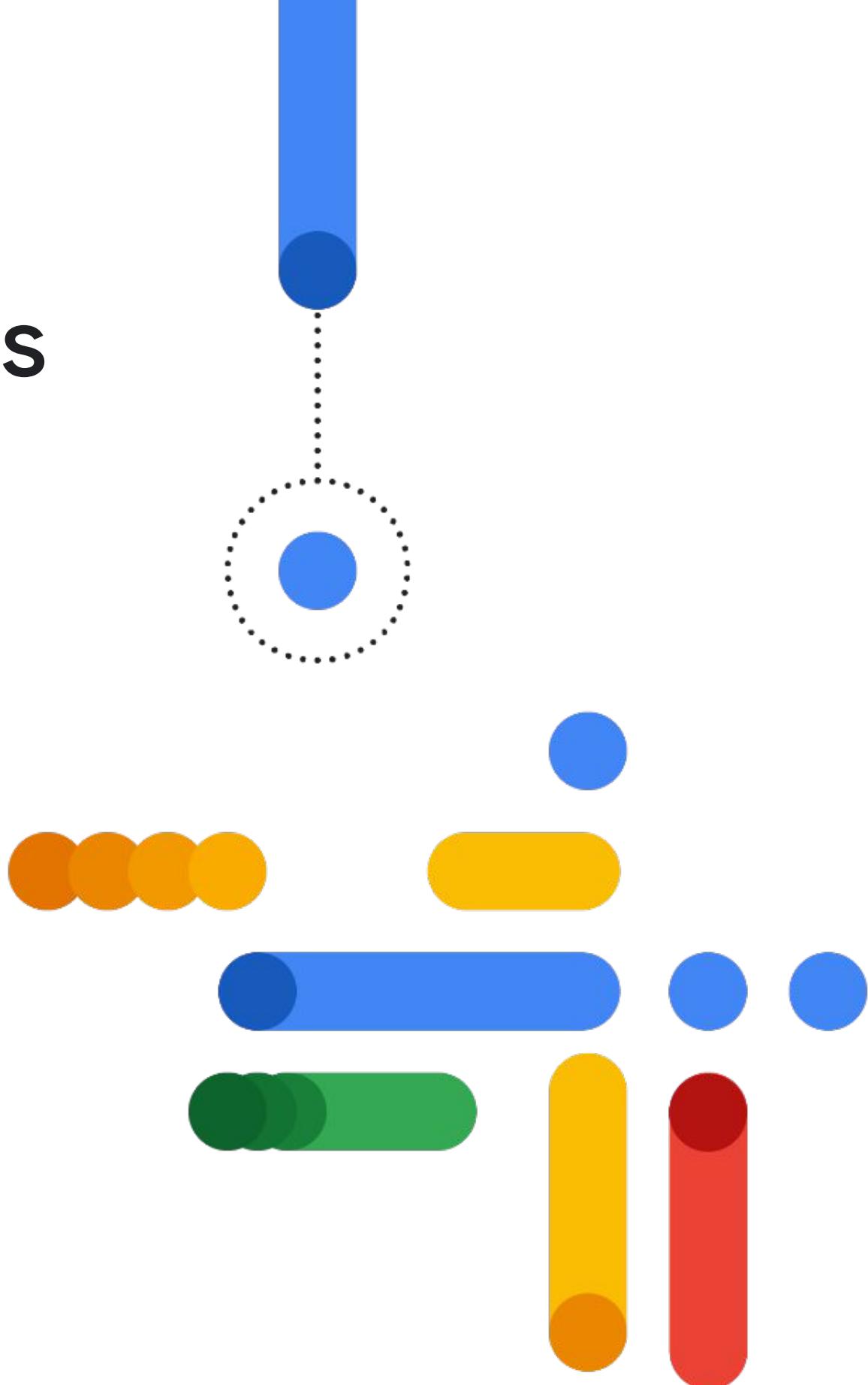
Vertex AI simplifies model training with AutoML

- Supports datasets created with image, text, tabular, and video data
- Zero-code, custom-model training for various use cases
 - Image detection and classification
 - Text classification, entity extraction, and sentiment analysis
 - Linear regression, classification, and forecasting from tabular datasets
 - Video action recognition and object tracking
- Automated deployment of models to service endpoints managed by Google



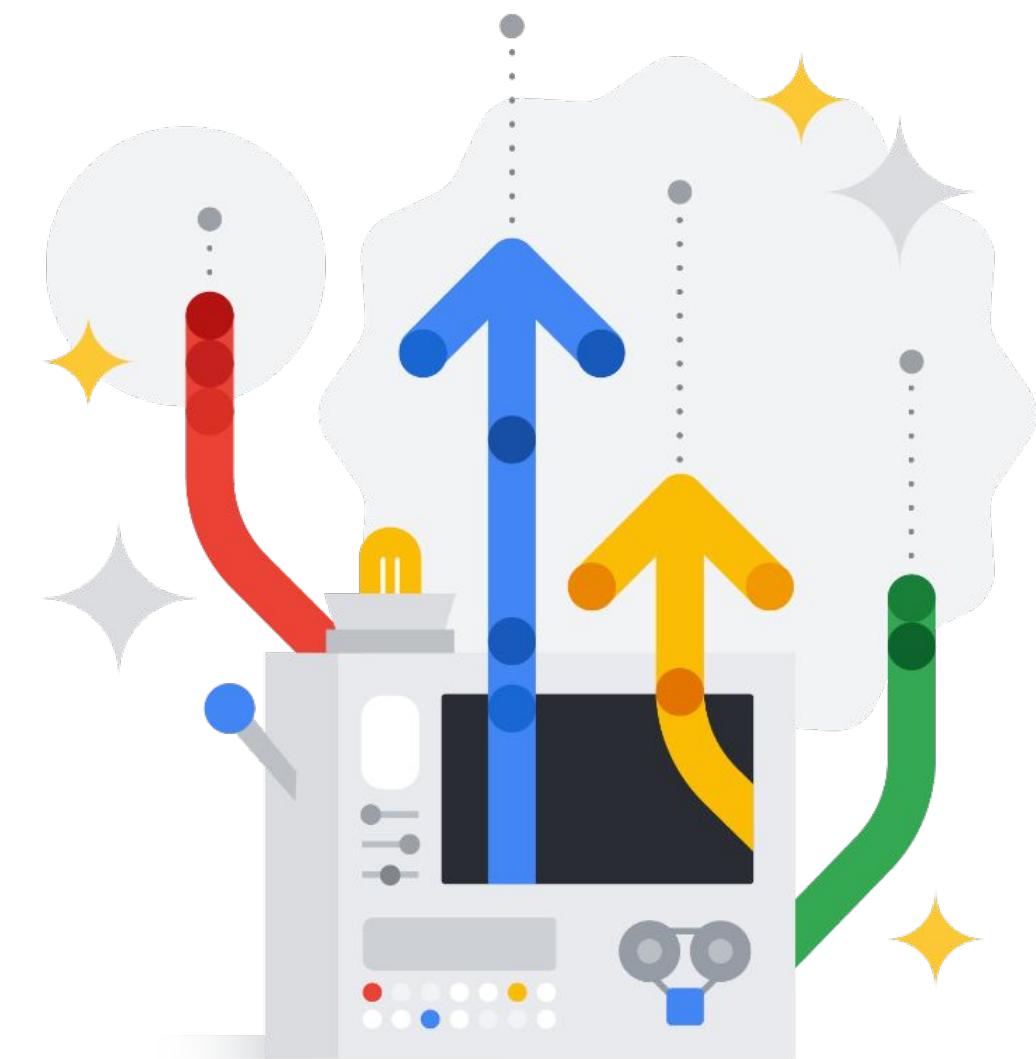
Model Garden provides a catalog of pre-trained models to build AI applications

- Foundation models are pre-trained multitask large models that can be tuned or customized for specific tasks
- Fine-tunable models are models that you can fine-tune using a custom notebook or pipeline
- Task-specific solutions are pre-built models that are ready to use and can be customized using your own data
- Model garden provides and easy to use interface for searching for models
 - Documentation and code samples are provided



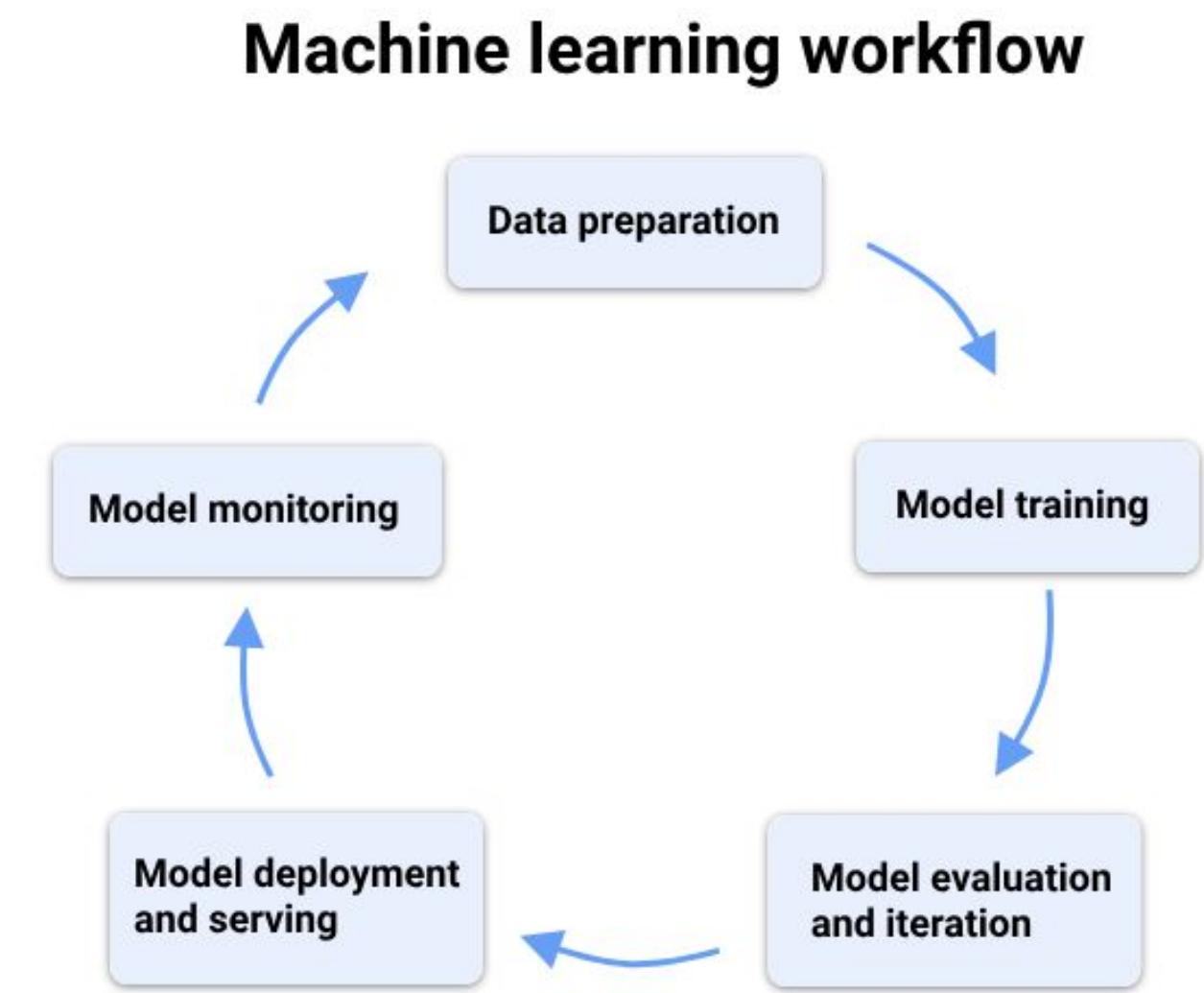
Generative AI Studio simplifies prompting, tuning, and deploying Google foundational models

- Low-code, customizable solution for building generative AI solutions
- Easy access to Google foundational models
 - PaLM 2 (Pathways Language Model) for text and chat
 - Codey for code generation, completion, and chat
 - Chirp for speech to text
 - Imagen for text to image generation



Vertex AI helps with all steps in an AI workflow

- Data preparation: clean and explore data, apply transformations and feature engineering
- Model training: choose a training method, train and tune the model
- Model evaluation and iteration: evaluate the model, make adjustments and iterate
- Model serving: deploy the model to production and get predictions
- Model monitoring: monitor the performance of the deployed model and retrain as needed



Vertex AI features by task

01

Data Preparation

Datasets
Labeling tasks
Feature store
AI Pipelines

02

Model Training

Auto ML
Model Garden
Gen AI Studio
Online Training
Workbench

03

Model Evaluation

Built-in metrics
Experiments

04

Model Deployment

AI Pipelines
Endpoints
Batch predictions

05

Model Monitoring

Operations Suite
Monitoring
Logging

Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case



Generative AI use cases with Vertex AI



Language

Writing
Summarization
Ideation
Classification
Sentiment analysis
Extraction
Customer chat

Code

Code generation
Code completion
Code chat
Code conversion

Speech

Speech to text
Text to speech

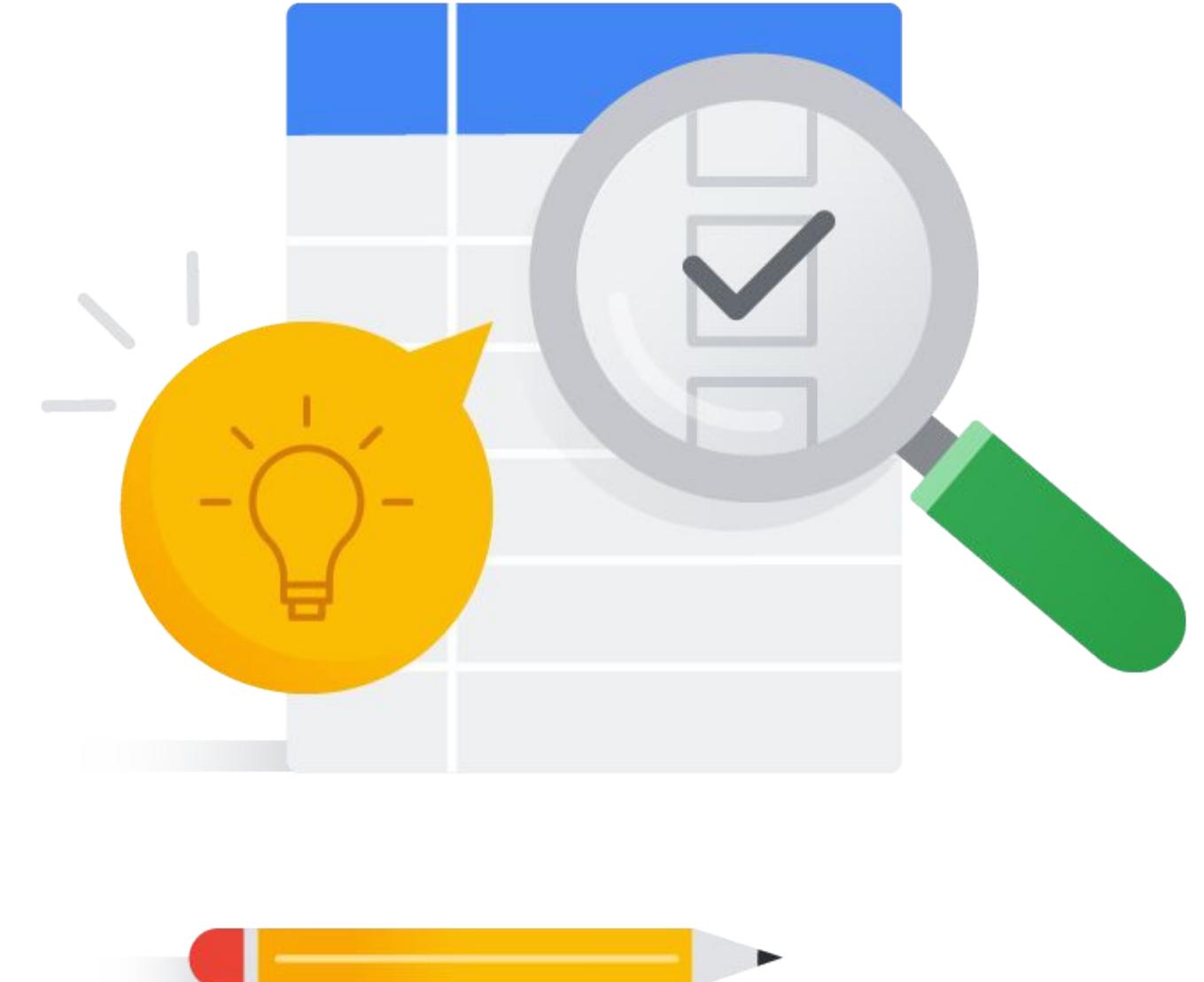
Group Discussion

10 min



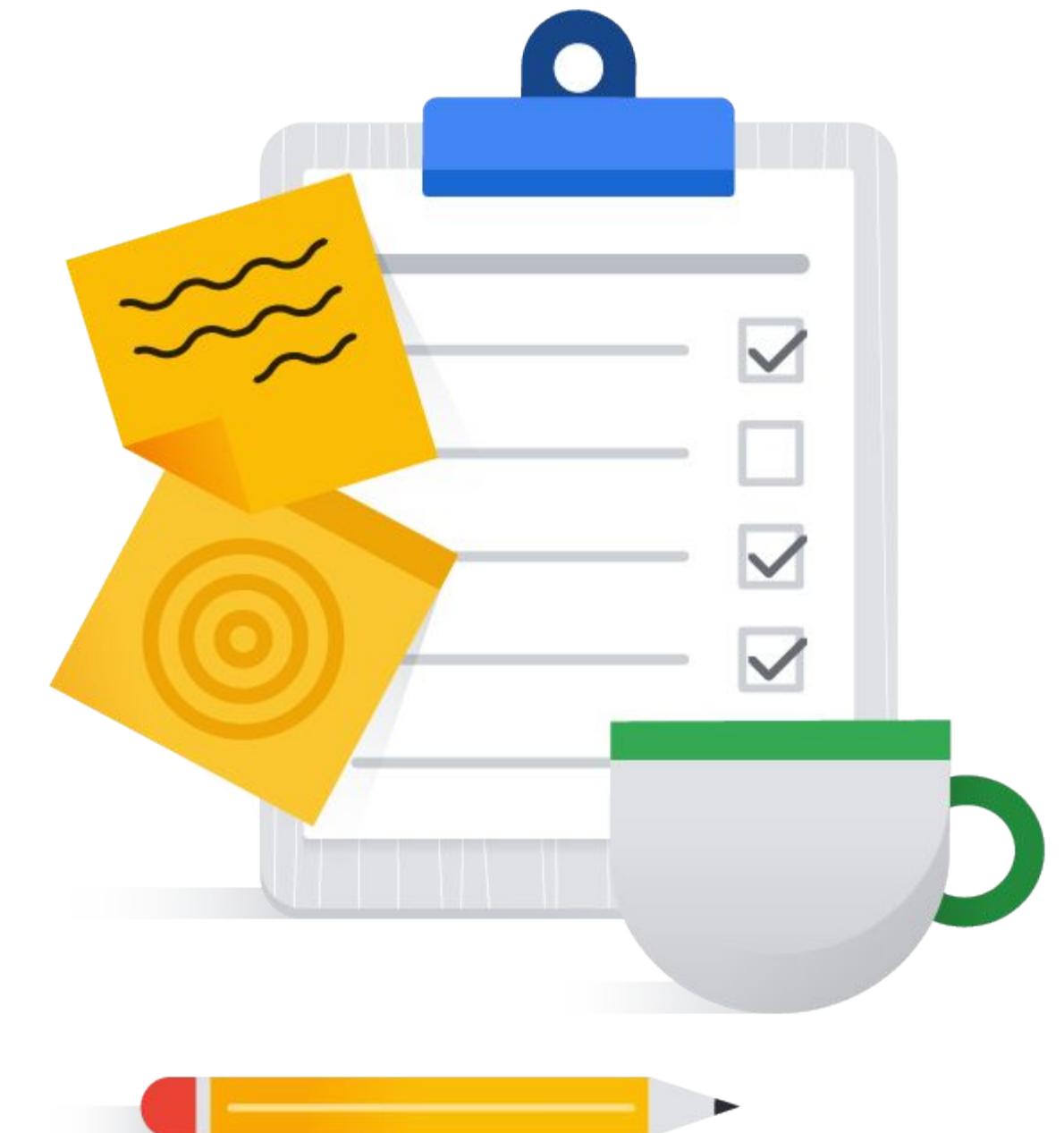
Take a few minutes to come up with some use cases for generative AI that would benefit you or your organization

You will share your ideas with the class



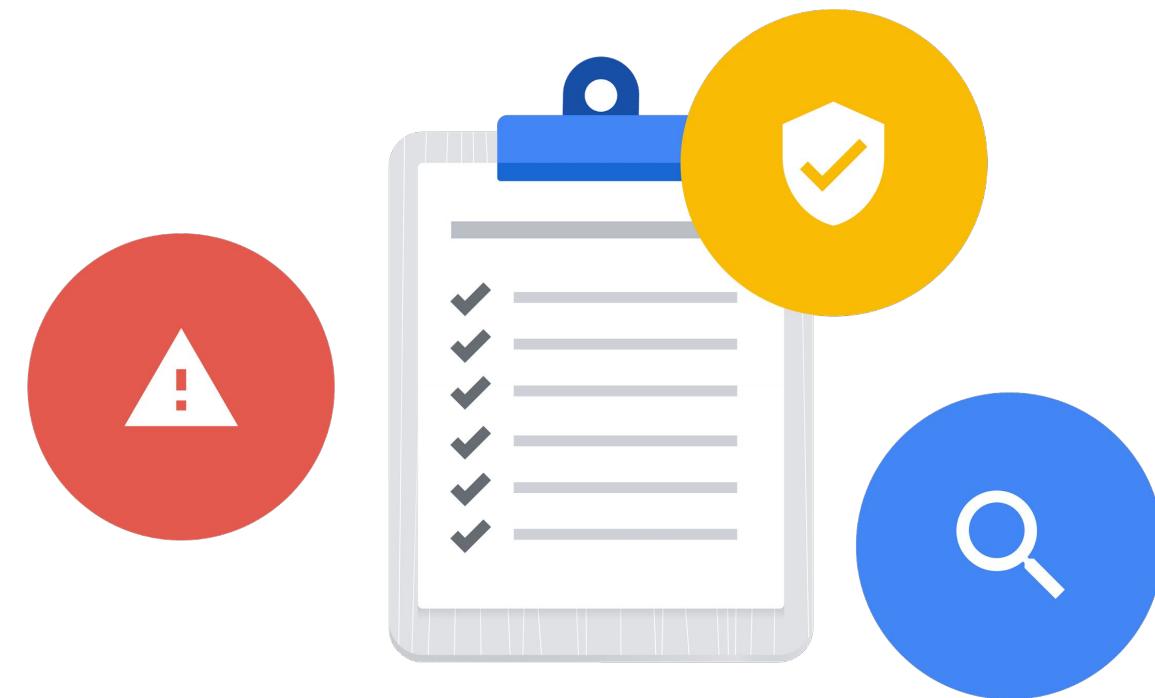
Topics

- 01 What is Generative AI
- 02 Vertex AI on Google Cloud
- 03 Generative AI Options on Google Cloud
- 04 Introduction to the Course Use Case

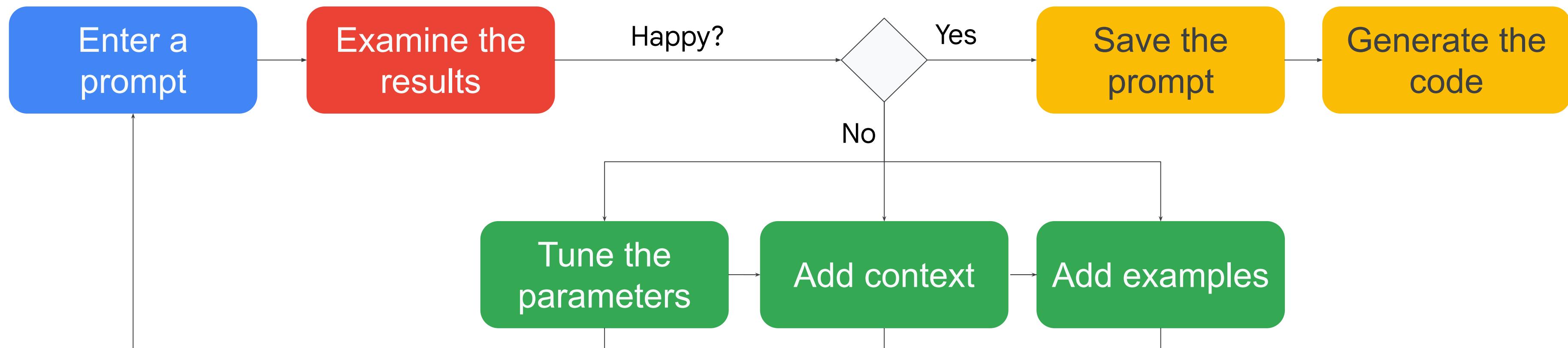


This course focuses on using generative AI for language-based content creation use cases

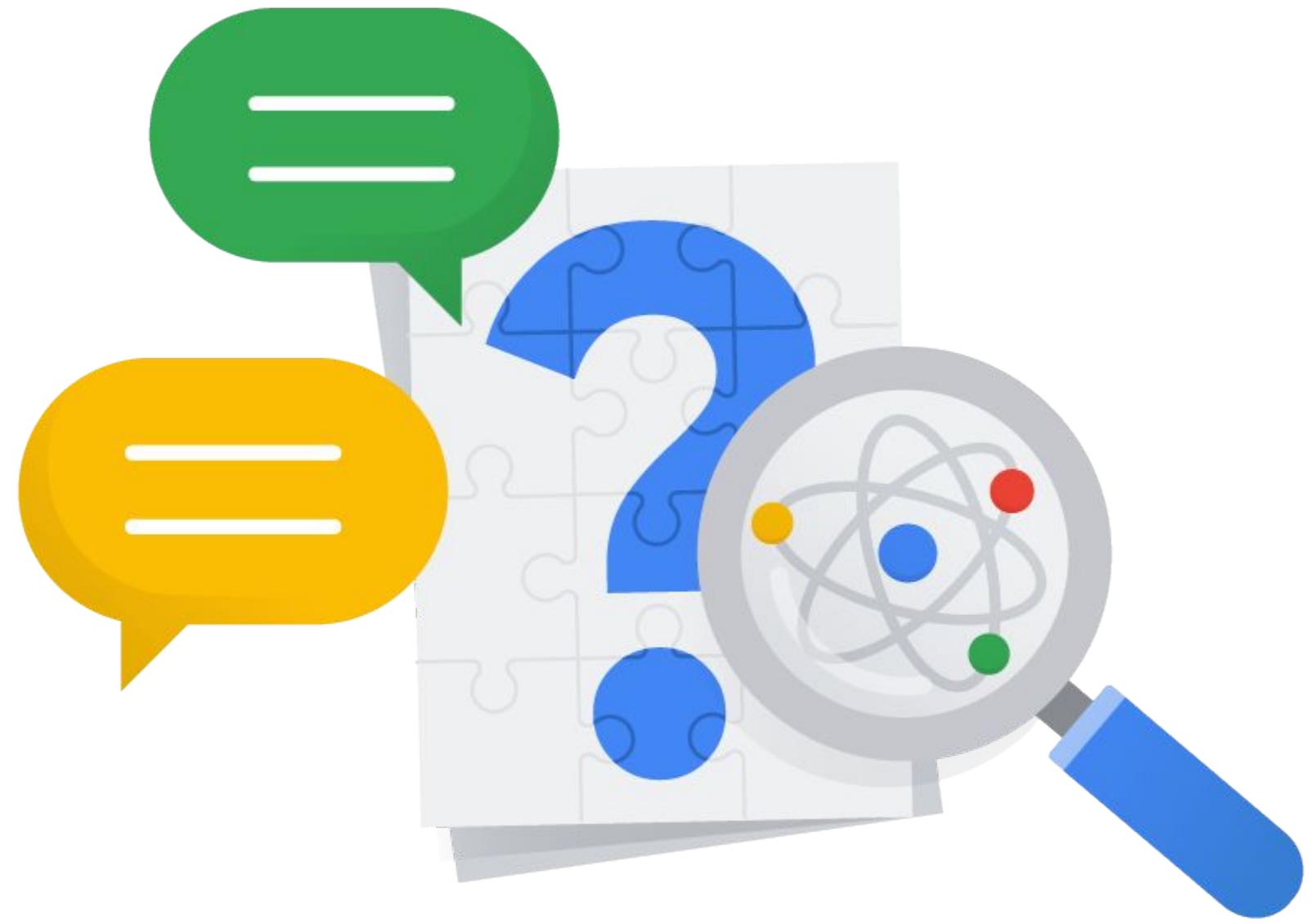
- Writing
- Summarization
- Ideation
- Classification
- Sentiment analysis
- Extraction



You will learn the process of designing, tuning, and deploying prompts to generate content for each use case



Questions and answers



In this module, you learned to ...

- 01 Differentiate between machine learning in general and generative AI
- 02 Automate ML tasks using Vertex AI on Google Cloud
- 03 Choose from the available Generative AI options on Google Cloud
- 04 Explore the text generation AI use case



Quiz question

What is a type of artificial intelligence that allows computers to learn based on data without having to be explicitly programmed?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

What is a type of artificial intelligence that allows computers to learn based on data without having to be explicitly programmed?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

What is a type of artificial intelligence that can create new content, such as text, images, or music?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

What is a type of artificial intelligence that can create new content, such as text, images, or music?

- A: Generative AI
- B: Machine learning
- C: Linear regression
- D: Linear classification

Quiz question

- When generative AI systems just make something up that isn't true, it is called what?
- A: A lie
 - B: A hallucination
 - C: A bug
 - D: It is not possible for that to happen

Quiz question

- When generative AI systems just make something up that isn't true, it is called what?
- A: A lie
 - B: A hallucination
 - C: A bug
 - D: It is not possible for that to happen

Quiz question

List some use cases for generative AI:

Quiz question

List some use cases for generative AI:

Customer Service Chat

Writing content

Ideation

Summarization

Classification

Sentiment analysis

....



Gen AI Studio

In this module, you learn to ...

- 01 Use Gen AI Studio to implement generative AI use cases
- 02 Explore the available models and sample use cases in Gen AI Studio
- 03 Write and test prompts using the Google Cloud Console
- 04 Protect your data while using Gen AI Studio



Topics

01

Gen AI Studio

02

Designing and testing prompts

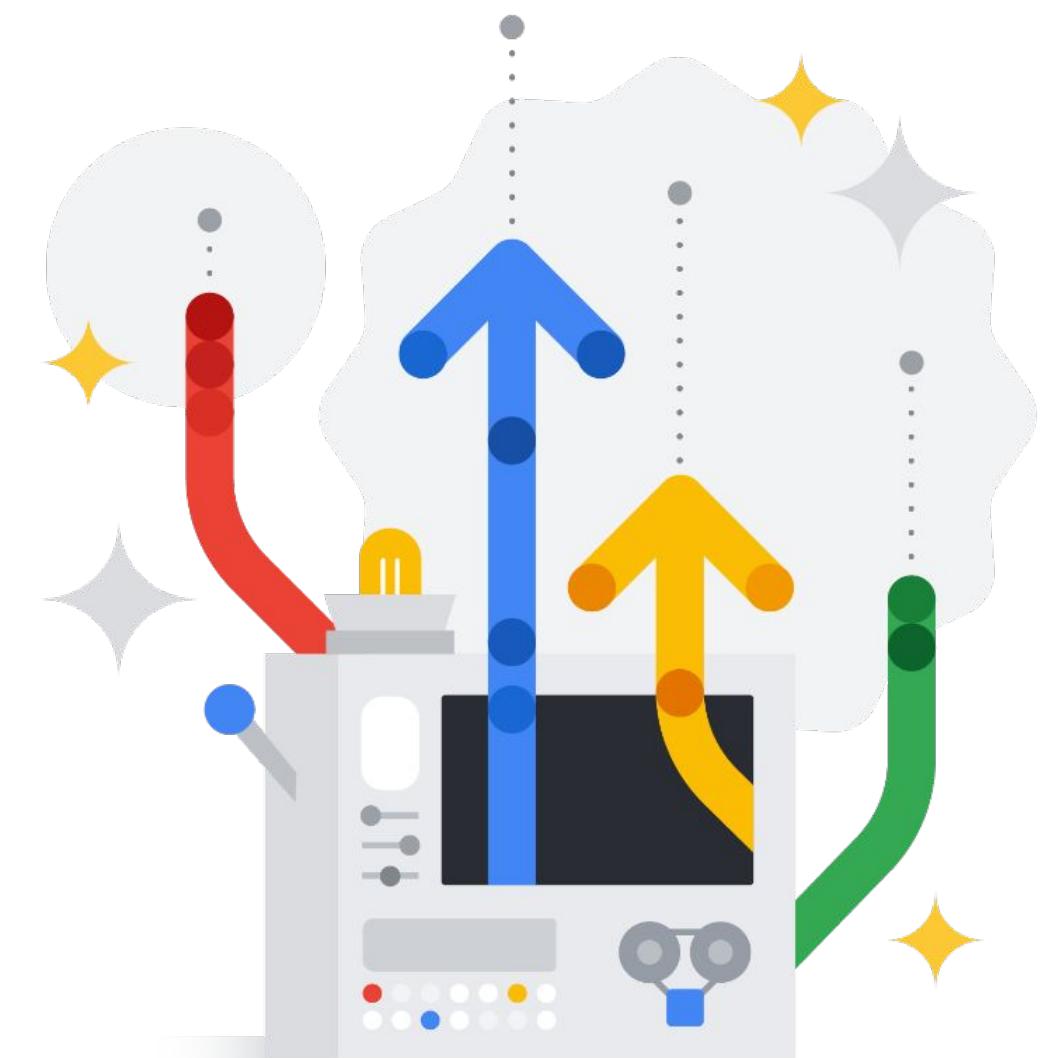
03

Data governance in GenAI Studio



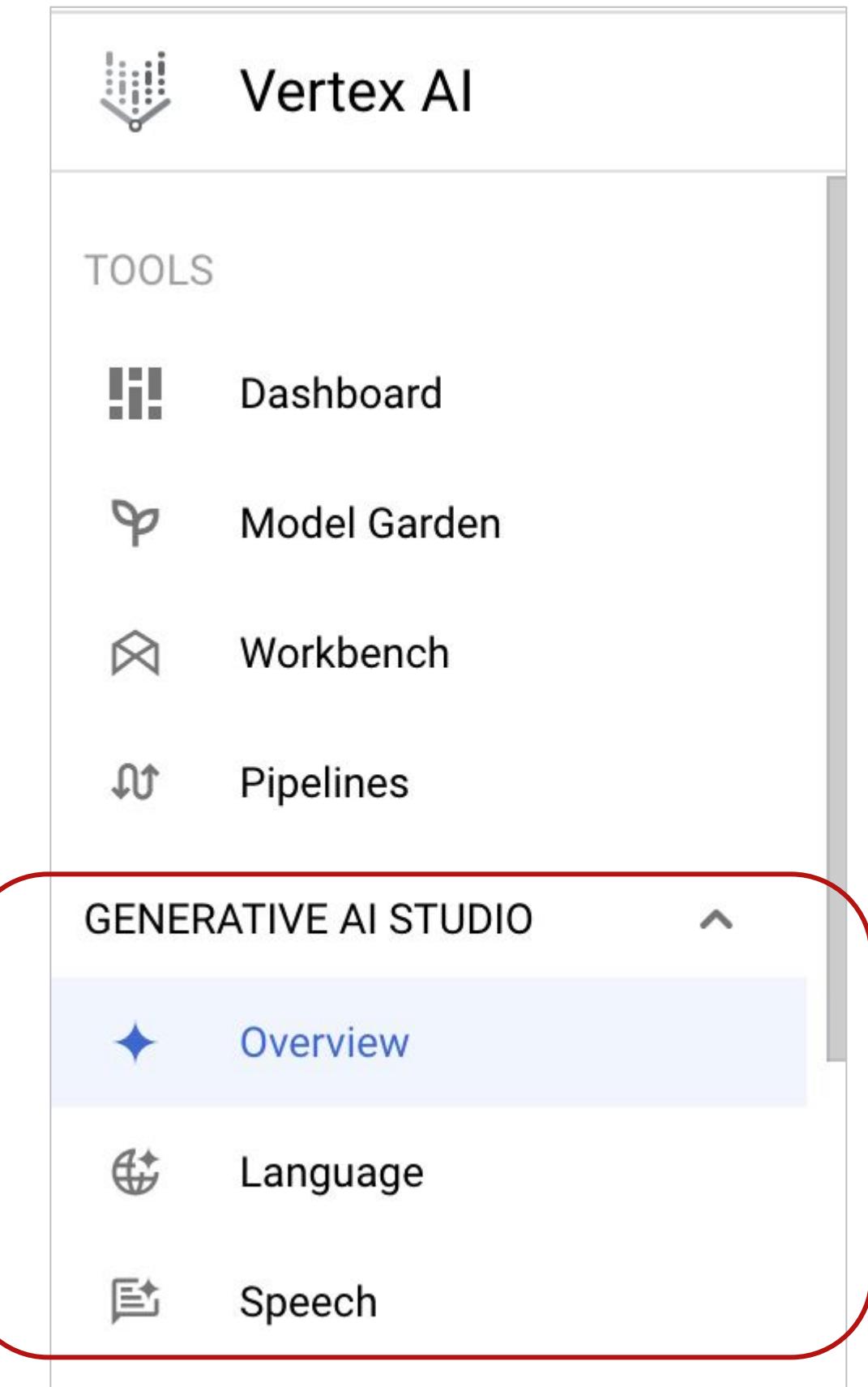
Generative AI Studio provides an intuitive interface to prompt, tune, and deploy Google foundation models

- Supports Google foundational models
 - PaLM 2 for text and chat
 - Codey for code generation and chat
 - Chirp for speech to text
 - Imagen for text to image generation
- Allows users to easily experiment with prompts
 - Simple, intuitive design
 - Easily experiment with parameters
 - Add context and examples

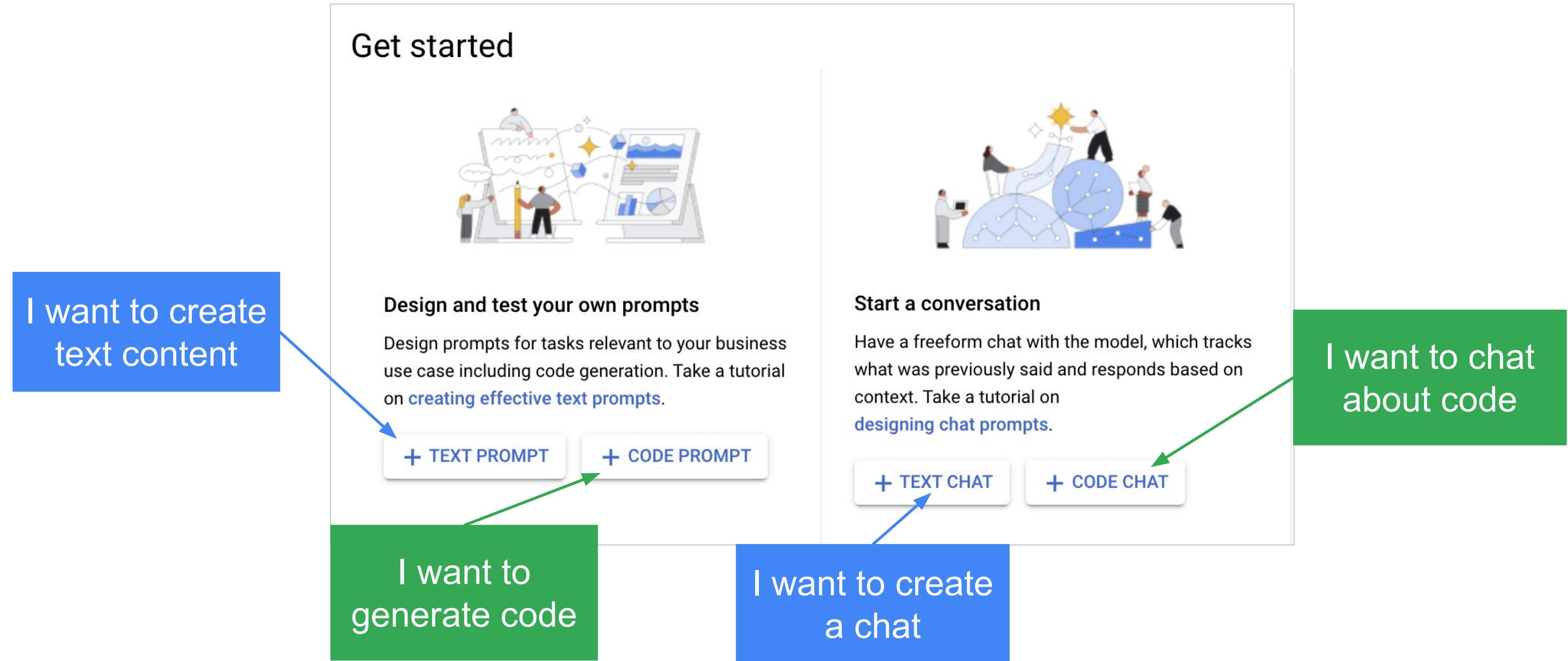


Gen AI Studio is available as a feature of Google Cloud Vertex AI

- Choose from Language or Speech models



To get started, choose a task based on your goal



Gen AI Studio UI

The screenshot illustrates the Gen AI Studio UI workflow:

- Enter a Prompt**: A blue callout points to the "Prompt" section where the user can type a question like "Who was Grace Hopper?".
- Select the foundational model**: A blue callout points to the "Model" dropdown menu set to "text-bison@001".
- Adjust the parameters**: A blue callout points to the parameter settings:
 - Temperature: 0.2
 - Token limit: 256
 - Top-K: 40
 - Top-P: 0.8
 - Safety filter threshold: Block few
- Submit the prompt**: A blue callout points to the "SUBMIT" button.
- The response**: A blue callout points to the "Response" section, which displays the generated text: "Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the pioneers of computer programming and is credited with coining the term "debugging". Hopper was born in New York City in 1906. She attended Vassar College and earned a master's degree in mathematics from Yale University. In 1943, Hopper joined the United States Navy Reserve. She was assigned to the Bureau of Ordnance Computation Project where she worked on the Mark I computer. In 1945, Hopper was one of the first programmers for the Harvard Mark II computer."
- Save the model**: A blue callout points to the "SAVE" button at the top right of the interface.

You can also use Gen AI Studio to fine-tune foundational models

- You supply your own training data in JSON format
- The foundational model is customized for your particular use case
 - A new model is created based on PaLM 2
 - Uses a process called “Transfer Learning”
- The training job runs on Google infrastructure
- Your model and data are stored separately in your own project



Tune a model

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on [creating a tuned model](#).

[+ CREATE TUNED MODEL](#)

Topics

01

Gen AI Studio

02

Designing and testing prompts

03

Data governance in GenAI Studio



Start with a simple text prompt and click the Submit button to see the results

▼ **Prompt**

Tell me about Grace Hopper



▼ **Response**

Markdown

Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the pioneers of computer programming, and is credited with coining the term "debugging". She also helped develop the first compiler, which translates human-readable code into machine code that computers can understand.

Hopper was born in New York City in 1906. She attended Vassar College, where she studied mathematics and physics. After graduating, she taught mathematics at Vassar for a year before joining the United States Navy in 1943.

During World War II, Hopper worked on the Mark I computer, one of the first electronic computers. She was responsible for developing the first compiler, which translated human-readable code into machine code that computers could understand. This work made it much easier for programmers to write code for computers, and helped to make computers more accessible to a wider range of people.

After the war, Hopper continued her work in computer science. She helped develop the COBOL programming language, which is still widely used today. She also served as the director of the Navy's computer science research center.

In 1986, Hopper was awarded the National Medal of Technology for her contributions to computer science

How about a code prompt?

Prompt

Write the Terraform code to create a Google Cloud VPC with Subnets in us-central1 and us-east4. Create variables for the IP address ranges.



Response

```
provider "google" {
  project = var.project_id
  region = var.region
}

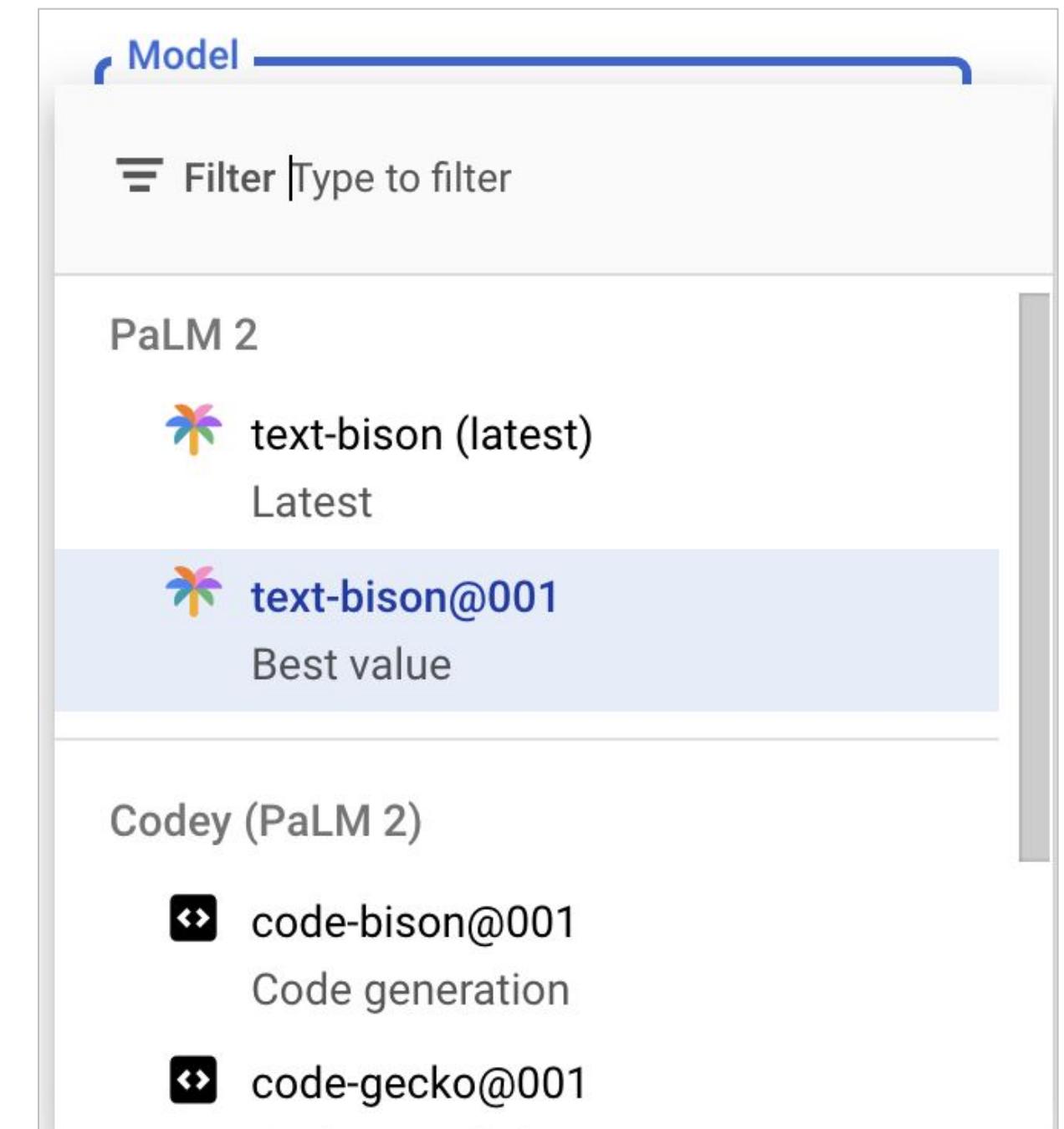
resource "google_compute_network" "default" {
  name = var.network_name
}

resource "google_compute_subnetwork" "us-central1" {
  name = var.subnetwork_name
  network = google_compute_network.default.name
  region = var.region
  ip_range = var.ip_range_us_central1
}

resource "google_compute_subnetwork" "us-east4" {
  name = var.subnetwork_name
  network = google_compute_network.default.name
  region = var.region
  ip_range = var.ip_range_us_east4
}
```

In Gen AI Studio, choose the right model for text or code

- Use different model versions depending on the use case
 - The latest versions are more capable, but also more expensive
- PaLM 2 is for text
 - `text-bison (latest)` is the best, but most expensive
 - `text-bison@###` is not the latest, but cheaper
- Codey is for code
 - `code-bison@###` is the best
 - `code-gecko@###` is less expensive
- There are smaller models that will run entirely on mobile devices when building apps



Select a chat to have a conversation, use Codey to talk about code

USER Write me the code to create a Google Cloud Storage bucket

AI To create a Google Cloud Storage bucket, you can use the following code:

```
import google.cloud.storage as gcs

# Create a client
client = gcs.Client()

# Create a bucket
bucket = client.create_bucket('my_bucket')
```

This code will create a bucket named my_bucket in your Google Cloud Storage account. You can then use this bucket to store data.

USER How about using a shell script

AI Sure, here is the code for creating a Google Cloud Storage bucket using a shell script:

```
#!/bin/bash

# Create the bucket
gsutil mb gs://my_bucket
```

This code will create a bucket named my_bucket in your Google Cloud Storage account. You can then use this bucket to store data.

PaLM 2 is good at summarizing articles and documents

Prompt

Summarize the following Wikipedia article about Grace Hopper:

Grace Brewster Hopper (née Murray; December 9, 1906 – American computer scientist, mathematician, and United

[1] One of the first programmers of the Harvard Mark I computer, she was one of the first to coin the term "bug" of computer programming who invented one of the first high-level programming languages, and she was the first to devise the theory of machine-independent programming languages. She developed the FLOW-MATIC programming language she created using the concepts of flowcharts, which extended to create COBOL, an early high-level programming language used in business, science, and engineering today.

Prior to joining the Navy, Hopper earned a Ph.D. in both mathematics and mathematical physics from Yale University and was a professor at Vassar College. Hopper attempted to enlist in the Navy during World War II but was rejected due to her age.

Response

Markdown

Grace Hopper was an American computer scientist and United States Navy rear admiral. She was one of the first programmers of the Harvard Mark I computer, and she was a pioneer in the development of computer programming languages. She is credited with coining the term "debugging".

Hopper was born in New York City in 1906. She earned a Ph.D. in mathematics from Yale University in 1934. She began her career as a professor of mathematics at Vassar College, but she left in 1943 to join the United States Navy Reserve.

Hopper served on the Mark I computer programming staff headed by Howard H. Aiken. She worked on the development of the UNIVAC I computer, and she was a key figure in the development of the COBOL programming language.

Hopper retired from the Navy in 1986, but she continued to work as a consultant for Digital Equipment Corporation. She died in 1992 at the age of 85.

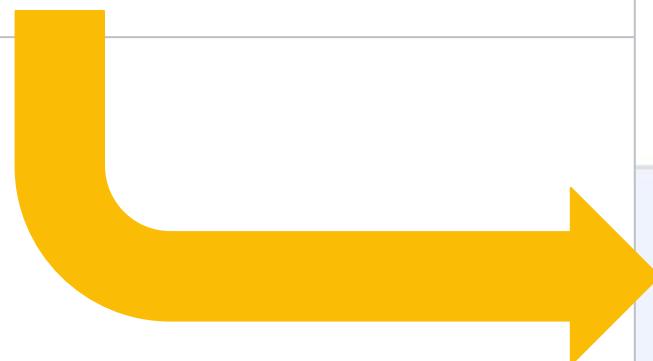
Hopper was a pioneer in the field of computer science, and she made significant contributions to the

How about sentiment analysis...

▼ Prompt

Is the following comment positive or negative:

Your summary of Grace Hopper was interesting and informative. I had no idea how important she was in the the field of computer science!



▼ Response

positive

Ideation...

▼ Prompt

I wrote an article about Grace Hopper. What are some good titles?



▼ Response

- Grace Hopper: A Pioneer in Computer Science
- Grace Hopper: The Woman Who Invented COBOL
- Grace Hopper: A Life of Ingenuity and Inspiration
- Grace Hopper: A True Renaissance Woman
- Grace Hopper: A Legacy of Leadership and Innovation

Classification...

▼ Prompt

Is the following Spam or Not Spam:

Sign up today for the Grace Hopper fan club. For the bargain price of \$9.99 per month you will get new articles every day. Plus, you will be entered into our drawing for a one-of-a-kind Grace Hopper Tee Shirt.



▼ Response

Spam

Extraction...

Prompt

Extract the important dates from the following Wikipedia article about Grace Hopper

Grace Brewster Hopper (née Murray; December 9, 1906 – American computer scientist, mathematician, and United States Navy rear admiral. [1] One of the first programmers of the Harvard Mark I computer, she was one of the first computer programmers who invented one of the first compilers. She also helped develop the first compiler for the FLOW-MATIC programming language she created using the concepts of flowcharts, which later extended to create COBOL, an early high-level programming language used today.

Prior to joining the Navy, Hopper earned a Ph.D. in both mathematics and mathematical physics from Yale University and was a pro-

Response

Important dates in Grace Hopper's life:

- * December 9, 1906: Grace Brewster Murray is born in New York City.
- * 1928: Graduates from Vassar College with a bachelor's degree in mathematics and physics.
- * 1930: Earns a master's degree from Yale University.
- * 1934: Earns a Ph.D. in mathematics from Yale University.
- * 1943: Joins the United States Navy Reserve.
- * 1944: Serves on the Mark I computer programming staff headed by Howard H. Aiken.
- * 1949: Joins the Eckert–Mauchly Computer Corporation as a senior mathematician.
- * 1952: Develops the first compiler, which is written for the A-0 System.
- * 1954: Named the company's first director of automatic programming.
- * 1966: Retires from the Naval Reserve with the rank of commander.
- * 1967: Recalled to active duty for a six-month period that turns into an indefinite assignment



Ok, but how do I use this in my app?

- In Gen AI Studio, click the **View Code** button
 - Returns the code in Python, iPython, and cURL



Python code example

View code

PYTHON

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    """Is the following Spam or Not Spam:

    Hi Doug, I read this article from Wikipedia on Grace Hopper. Thought you might
    like it.

    -- Grace Hopper"""
)
print(f"Response from Model: {response.text}")
```

cURL code example

View code

PYTHON PYTH

Use the command-line interface (CLI) to request a model response

1. Install Google Cloud SDK [Google Cloud SDK](#) if you haven't already or open Cloud Shell and skip to step three.
2. Run the following command to authenticate using your Google account.

```
$ gcloud auth application-default login
```

3. Enter the following to request a model response

```
API_ENDPOINT="us-central1-aiplatform.googleapis.com"
PROJECT_ID="vertext-ai-dar"
MODEL_ID="text-bison@001"

curl \
-X POST \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
-H "Content-Type: application/json" \
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/us-central1/publishers/text-bison/models/text-bison@001:predict"
$'{
  "instances": [
    {
      "content": "Is the following Spam or Not Spam:

Hi Doug, I read this article from Wikipedia on Grace Hopper. Thought you might be interested.

"
    }
  ],
  "parameters": {
    "temperature": 0.2,
    "maxOutputTokens": 256,
    "topP": 0.8,
    "topK": 40
  }
}'
```

But I want a different language... Ask Codey!

▼ **Prompt**

Can you convert the following Python code into Go?

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8
```



▼ **Response**

Markdown

```
package main

import (
    "fmt"
    "google.golang.org/api/vertexai/v1alpha1"
)

func main() {
    // Initialize the client.
    client, err := vertexai.NewClient(vertexai.ClientConfig{
        ProjectID: "vertext-ai-dar",
        Location:  "us-central1",
    })
    if err != nil {
        panic(err)
    }
```

You can explore sample use cases and prompts from Vertex AI Gen AI Studio

The screenshot displays the Vertex AI Gen AI Studio interface, specifically the 'Prompt examples' section. At the top, there are navigation tabs: 'GET STARTED' (blue), 'MY PROMPTS' (grey), 'TUNING' (grey), and 'PREVIEW' (black). Below the tabs, the title 'Prompt examples' is followed by the sub-instruction 'Try a sample prompt to test a foundation model with a practical use case'.

Chat

This section contains three examples:

- Summarization:** Kindergarten Science Teacher. Your name is Miles. You are an astronomer who wants to teach children about the solar system. You write short sentences. **OPEN**
- Summarization:** Online Return Customer Support. You work in a customer support center. You help customers return products. You are friendly and patient. **OPEN**
- Summarization:** Gluten Free Advisor. You are a nutritionist who helps people with gluten allergies. You provide advice on healthy eating. **OPEN**

Classification

This section contains two examples:

- Freeform:** Support rep chat summarization. You are a customer support manager and would like to see what your team is talking about. **OPEN**
- Structured:** Sentiment analysis about people. You would like to see how reporters write about certain people. You have articles and would like to see if a certain person is written about... **OPEN**

Extraction

This section contains one example:

- Structured:** Pixel Technical Specification. Generate technical specifications from text of a Pixel phone into JSON, one-shot. **OPEN**

Writing

This section contains three examples:

- Freeform:** Marketing generation Pixel. You work in Google's device marketing team and you need to create marketing pitch for the new Pixel 7 Pro. You have writers block... **OPEN**
- Freeform:** Ad copy generation. You are a marketer and want to create different versions of the same ad to target different audiences. You would like some suggestions. **OPEN**
- Freeform:** Essay outline. Generate an outline for an essay on a particular topic. **OPEN**

Model Garden allows you to explore both Google and open-source models for various use cases

The screenshot shows the Model Garden interface. On the left, there's a sidebar with two sections: 'Modalities' and 'Tasks'. The 'Modalities' section lists Language (23), Vision (46), Tabular (2), Documents (2), and Speech (1). The 'Tasks' section lists Generation (22), Classification (26), Detection (17), Extraction (7), Recognition (6), Translation (2), and Embedding (2). At the top right, there are buttons for 'EXPLORE GENERATIVE AI' and 'VIEW MY MODELS'. Below these are a search bar ('Search models') and a 'Suggestions' section with links to 'text embedding', 'essay outline', and 'BERT'. The main content area is titled 'Foundation models' and describes pre-trained multi-task models. It features three cards: 'PaLM 2 for Text', 'PaLM 2 for Chat', and 'Embeddings for text'. Each card has a 'Foundation' and 'Language' tag, a brief description, and a 'chat-bison@001' or 'textembedding-gecko@001' identifier, followed by a 'VIEW DETAILS' button. At the bottom, there's a link to 'SHOW ALL (25)'.

Model Garden

EXPLORE GENERATIVE AI

VIEW MY MODELS

Modalities

Modality	Count
Language	23
Vision	46
Tabular	2
Documents	2
Speech	1

Tasks

Task	Count
Generation	22
Classification	26
Detection	17
Extraction	7
Recognition	6
Translation	2
Embedding	2

Search models

Suggestions [text embedding](#) [essay outline](#) [BERT](#)

Foundation models

Pre-trained multi-task models that can be further tuned or customized for specific tasks.

PaLM 2 for Text
Fine-tuned to follow natural language instructions and is suitable for a variety of language tasks, such as: classification, extraction, summarization and content...
text-bison@001
[VIEW DETAILS](#)

PaLM 2 for Chat
Fine-tuned to conduct natural conversation. Use this model to build and customize your own chatbot application.
chat-bison@001
[VIEW DETAILS](#)

Embeddings for text
Text embedding is an important NLP technique that converts textual data into numerical vectors that can be processed by machine learning algorithms, especially large models...
textembedding-gecko@001
[VIEW DETAILS](#)

SHOW ALL (25)

Vertex Model Garden

Choose a model that fits your needs

Model Type	Model Description	Model Details
First Party	<p>Foundation models Leverage Google's multimodal models across vision, dialog, code generation/completion</p> <p>Pre-trained APIs Build and deploy AI applications faster with our pre-trained APIs powered by the best Google AI research and technology</p>	<ul style="list-style-type: none">• PaLM for Text PaLM for Chat• Imagen for text-to-image• Codey for code completion• Chirp for speech-to-text <ul style="list-style-type: none">• Speech-to-Text• Natural Language Processing• Translation• Vision
Open Source	<p>Open Source Access a wide variety of enterprise-ready open source models</p>	<ul style="list-style-type: none">• StableDiffusion• ViT• EfficientNet
3rd Party	<p>3rd Party Over time Model Garden will support 3rd-party models from partners with foundation models</p>	Coming soon

Topics

01

Gen AI Studio

02

Designing and testing prompts

03

Data governance in GenAI Studio



AI Data Governance

Google Cloud's approach to governance of customer data for Cloud AI Large Models and Generative AI



We have no access to customer data, in accordance with [GCP Terms](#) and [Cloud Data Processing Addendum](#).



We're committed to transparency, compliance with regulations like the EU General Data Protection Regulation (GDPR), and privacy best practices.



We will not use customer data to train our models



Security Gen AI Risk Pillars

01

Privacy & Safety

Transparency on how Google LLMs operate is core to the Google's mission for Responsible AI

02

Prompt Hacking

Large Language Models can be manipulated to output responses that are not aligned with its objectives

03

Sec Operations

Gen AI can be leveraged to monitor for malicious activities

04

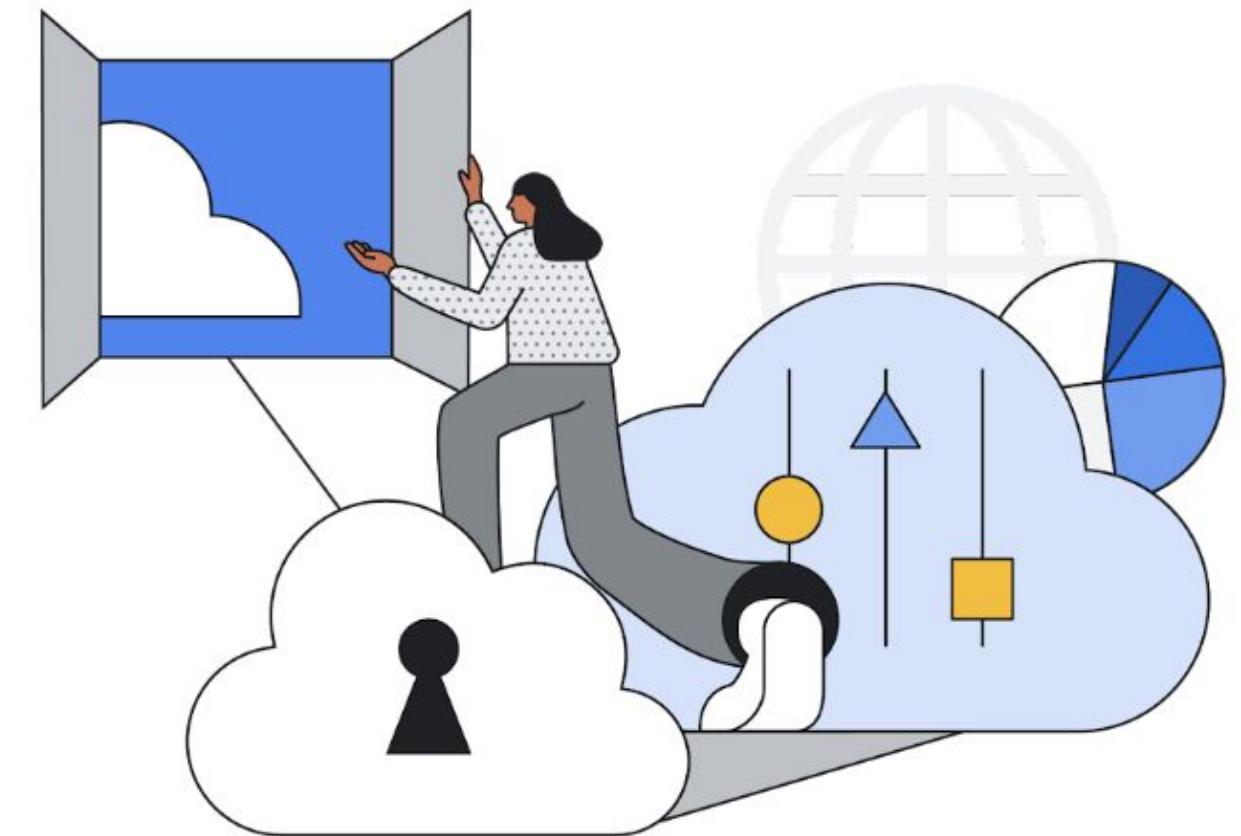
Cloud Resources

Gen AI workloads require the same security controls as traditional workloads



Customer data is kept private during prompt design

- Customer Data is encrypted in-transit when a request is submitted to a foundational model
 - Uses industry-standard cryptographic protocols to protect the confidentiality of customer data
- The foundation model processes customer data to generate a response
 - Does not retain the data for any other purpose
- Google does not use customer data to train its models without the express consent of its customers



Model tuning data is not retained by Google

Fine tuning does not rebuild foundational models

- Creates additional parameters, called “adapter weights”
- Passed to foundational model during inference

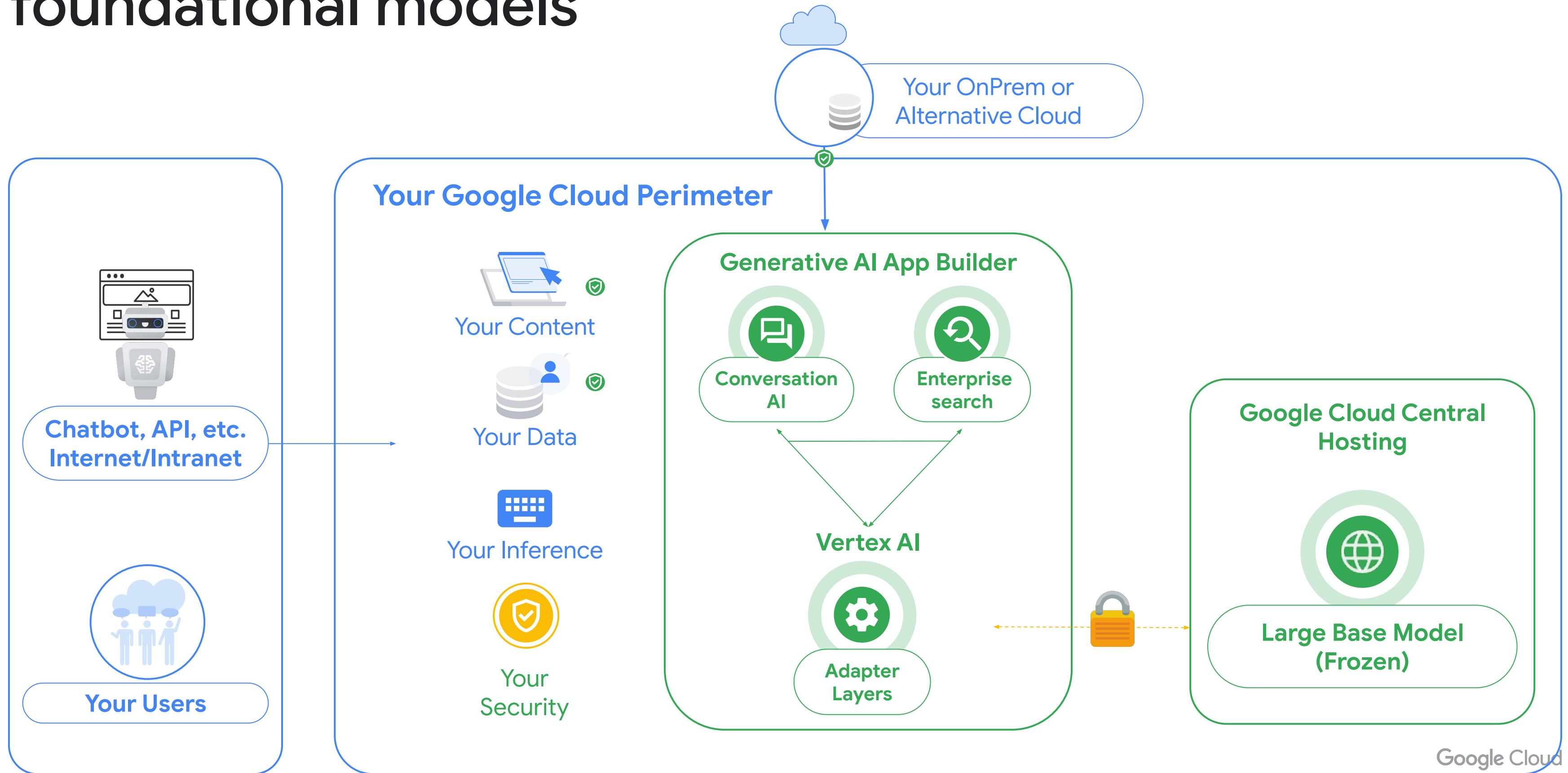
Customers have sole access to tuned models

- Can control encryption keys
- Can delete the model

Customer data is secure

- All data is encrypted at rest and in-transit
- Prompts and training data aren't logged

Your fine-tuned models are separated from the foundational models



Content processing in the PaLM API is assessed against a list of safety attributes

- Scores are values from 0.0 to 1.0
 - Rounded to one decimal
- The scores are ML predictions
 - Thus, cannot be relied on for 100% accuracy
- If a response exceeds the safety threshold it is blocked
- If content is blocked, the model will return a canned response
 - e.g. "I'm not able to help with that, as I'm only a language model"

```
"predictions": [
  {
    "safetyAttributes": {
      "categories": [
        "Derogatory",
        "Toxic",
        "Violent",
        "Sexual",
        "Insult",
        "Obscene",
        "Death, Harm & Trage",
        "Firearms & Weapons",
        "Public Safety",
        "Health",
        "Religion & Belief",
        "Drugs",
        "War & Conflict",
        "Politics",
        "Finance",
        "Legal"
      ],
      "scores": [
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1,
        0.1
      ]
    }
  }
]
```

To adjust the likelihood of content being blocked, set the Safety filter threshold attribute in Gen AI Studio



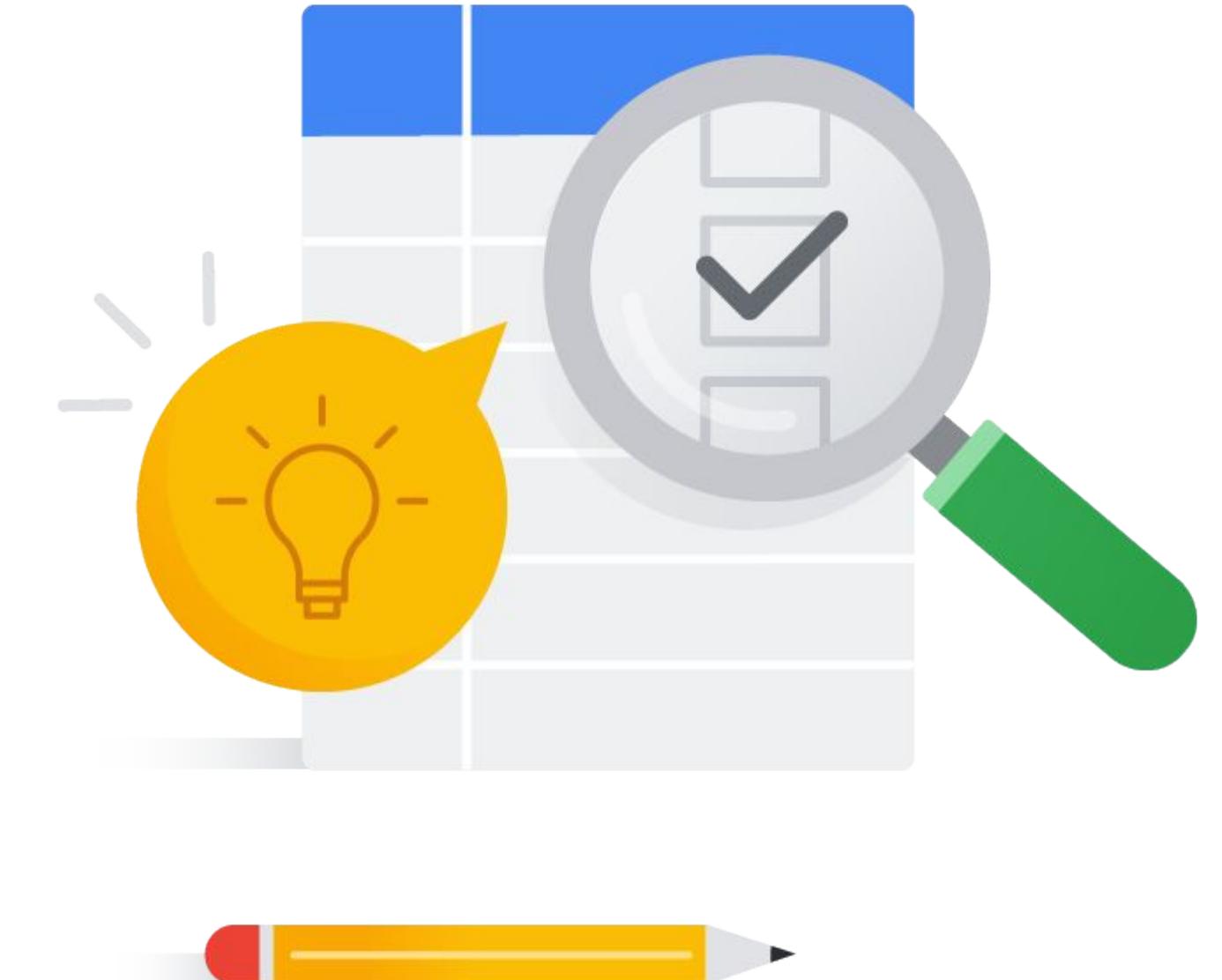
To use Gen AI responsibly follow these best practices

- Assess your application's security risks
- Consider adjustments to mitigate safety risks
- Perform safety testing appropriate to your use case
- Solicit user feedback and monitor content

Lab

⌚ 30 min ⚙️

Getting started with Vertex AI Gen AI Studio's User Interface

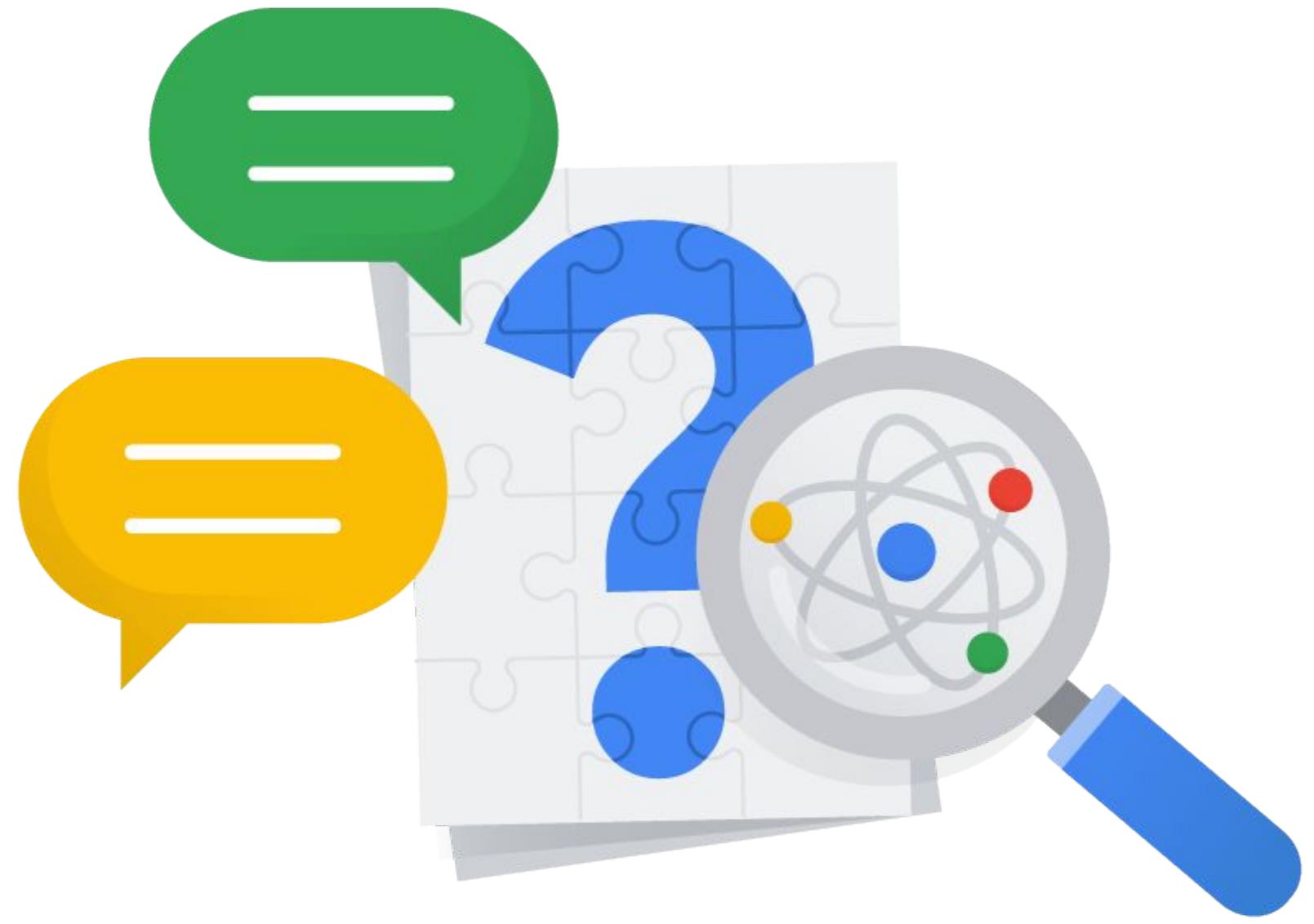


In this module, you learned to ...

- 01 Use Gen AI Studio to implement generative AI use cases
- 02 Explore the available models and sample use cases in Gen AI Studio
- 03 Write and test prompts using the Google Cloud Console
- 04 Protect your data while using Gen AI Studio



Questions and answers



Quiz question

Match the model to the task

PaLM 2

Generate text from speech

Codey

Generate code

Chirp

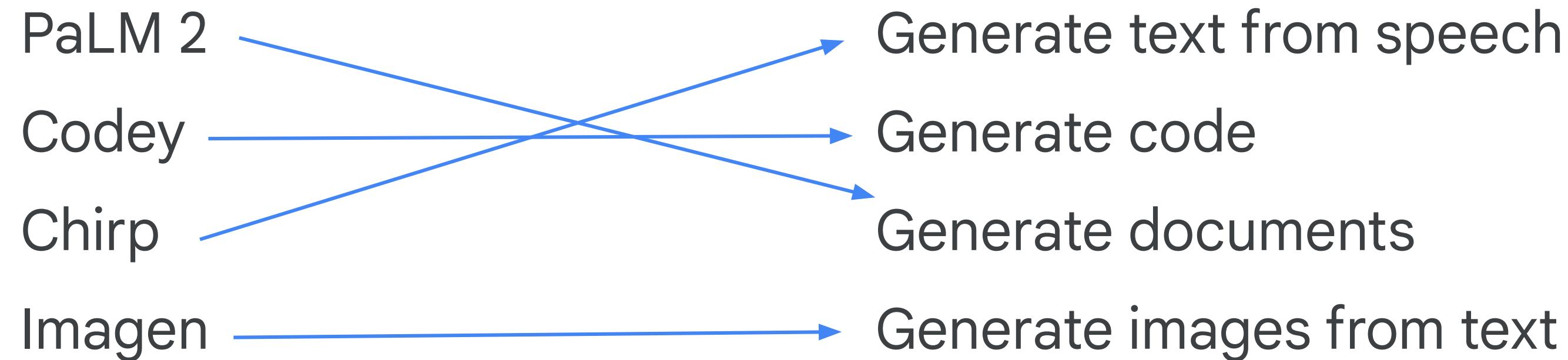
Generate documents

Imagen

Generate images from text

Quiz question

Match the model to the task



Quiz question

Gen AI Studio will generate code in which languages? (choose all that apply)

- A: Python
- B: IPython (Jupyter)
- C: Go
- D: C#
- E: cURL
- F: Java

Quiz question

Gen AI Studio will generate code in which languages? (choose all that apply)

- A: Python
- B: IPython (Jupyter)
- C: Go
- D: C#
- E: cURL
- F: Java

Quiz question

List some use cases for PaLM 2 language models:

Quiz question

List some use cases for PaLM 2 language models:

Writing

Sentiment analysis

Classification

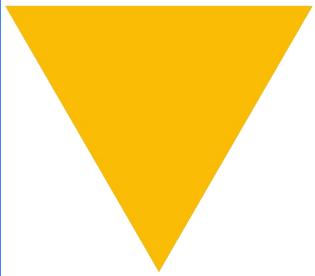
Extraction

Summarization

Ideation

Chat

03



Prompt Design

In this module, you learn to ...

01

Design prompts for various Gen AI use cases

02

Add examples and context to customize generative AI output

03

Experiment with parameters for model fine-tuning



Topics

01

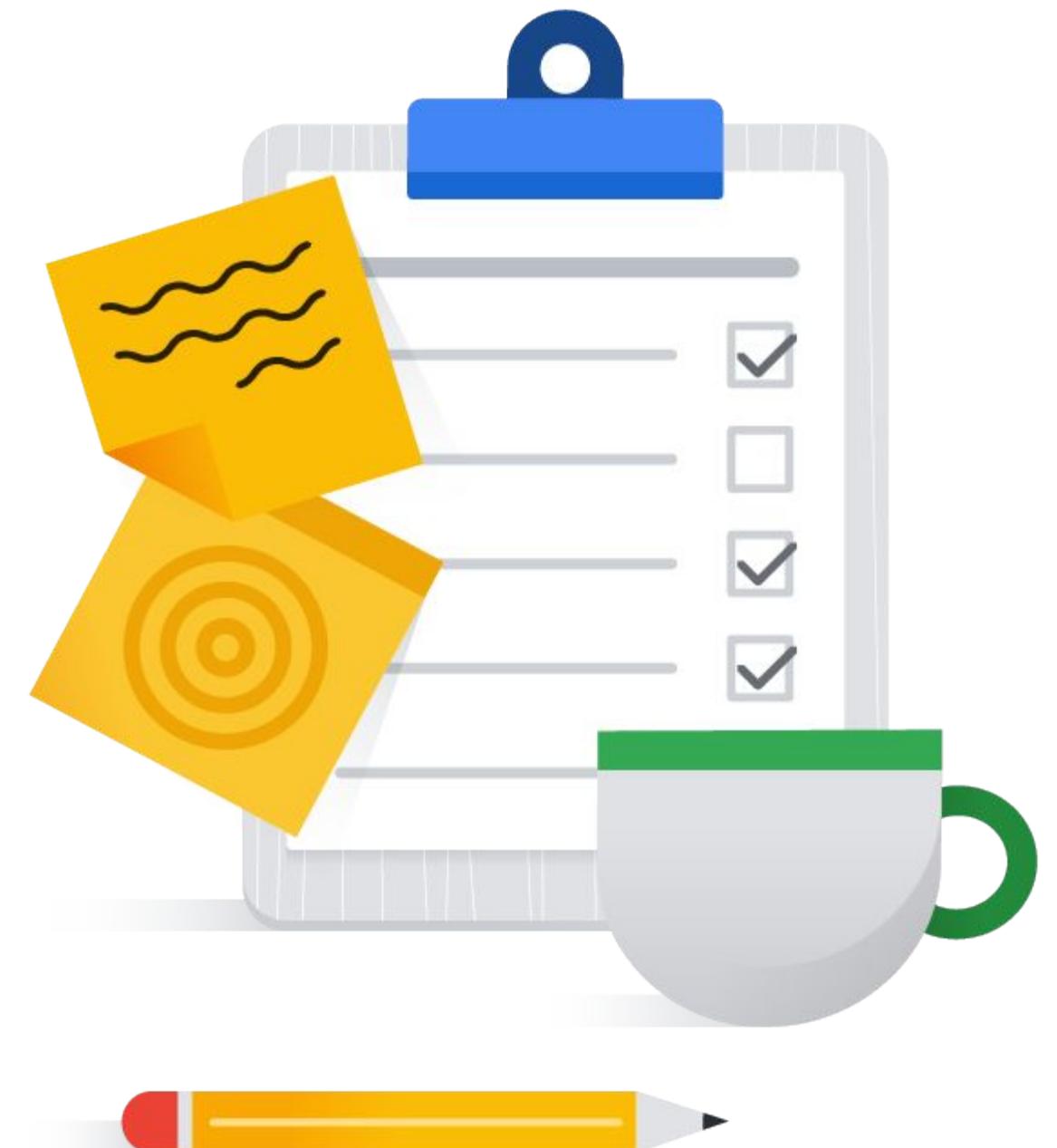
Understanding Prompt Design

02

Context, Examples, and Fine-Tuning Parameters

03

Prompt Design Best Practices



Prompt design is the process of writing instructions to generative AI models

- Well-designed prompts can help to improve the accuracy, creativity, and safety of generative AI output
- Factors to consider when designing prompts
 - Desired output
 - Target audience
 - Limitations of the generative AI model

Some tips for writing effective prompts include...

- Be clear and concise
- Use specific language
- Provide examples
- Use negative prompts to avoid unwanted output
 - E.g. “Who are the greatest NFL quarterbacks of all time? Don't include Tom Brady.”

With careful attention to prompt design, generative AI models can be used to create powerful and useful applications

Those last couple of slides were written by Bard again

A screenshot of a Google Docs draft interface. At the top, there's a header bar with a small profile icon of a person with glasses, followed by the text "Write for me a slide that summarizes the importance of prompt design when using generative AI" and a pencil icon for editing. To the right of the text is a "View other drafts" button with a dropdown arrow. The main content area starts with a yellow starburst icon and the text "Sure, here is a slide that summarizes the importance of prompt design when using generative AI:". Below this, the slide structure is defined: "Title: The Importance of Prompt Design in Generative AI", "Subtitle: How to write effective prompts that guide generative AI models to produce the desired output.", and "Body:" followed by a bulleted list of five points about prompt design. At the bottom of the slide, there's a footer bar with a "Enter a prompt here" input field, a microphone icon, and a large blue "Send" button with a white starburst icon.

Write for me a slide that summarizes the importance of prompt design when using generative AI

View other drafts ▾

Sure, here is a slide that summarizes the importance of prompt design when using generative AI:

Title: The Importance of Prompt Design in Generative AI

Subtitle: How to write effective prompts that guide generative AI models to produce the desired output.

Body:

- Prompt design is the process of writing instructions to generative AI models.
- Well-designed prompts can help to improve the accuracy, creativity, and safety of generative AI output.
- There are a number of factors to consider when designing prompts, including the desired output, the target audience, and the limitations of the generative AI model.
- Some tips for writing effective prompts include:

Enter a prompt here

How prompts effect output

Prompt

Who is Linus Torvalds?

Summarize some important dates in the life of Linus Torvalds

Write me a tweet to celebrate Linus Torvald's birthday

Results

Linus Torvalds is a Finnish software engineer who is best known as the creator and lead developer of the Linux kernel, the operating system kernel at the core of most Linux distributions. He is also the creator of the Git distributed version control system.

1969: Linus Torvalds is born in Helsinki, Finland.
1988: Torvalds begins studying computer science at the University of Helsinki.
1991: Torvalds creates the Linux kernel.
1992: The first version of Linux is released to the public.
1996: Torvalds becomes the lead developer of Linux.
2000: Linux is adopted by IBM as the operating system for its servers.
2007: Torvalds is named one of Time magazine's "100 Most Influential People in the World."
2011: Torvalds receives the ACM Turing Award for his contributions to the development of Linux.
2015: Torvalds steps down as the lead developer of Linux, but remains an active contributor to the project.

Happy birthday to Linus Torvalds, the creator of Linux! Thank you for your contributions to the open source community. #Linux #OpenSource

Be specific when writing your prompts

Prompt	Response
Write for me a short 2-level outline in bullets for a 1-page paper about Linus Torvalds	<p>Linus Torvalds</p> <ul style="list-style-type: none">● Biography<ul style="list-style-type: none">○ Born in Helsinki, Finland in 1969○ Studied computer science at the University of Helsinki○ Started working on Linux in 1991○ Linus Torvalds is the creator of the Linux kernel, the core of the Linux operating system.● Accomplishments<ul style="list-style-type: none">○ Linux is the most popular operating system for servers and supercomputers○ Linux is also used on a wide variety of devices, including smartphones, tablets, and personal computers○ Linus Torvalds has received numerous awards for his work on Linux, including the ACM Turing Award in 2015● Influence<ul style="list-style-type: none">○ Linus Torvalds has had a profound influence on the open source software movement○ Linux has inspired the development of many other open source operating systems, such as FreeBSD and Solaris○ Linus Torvalds is a role model for many programmers and engineers● ...

Topics

01

Understanding Prompt Design

02

Context, Examples, and Fine-Tuning Parameters

03

Prompt Design Best Practices



Freeform versus Structured prompts in Gen AI Studio

- So far in class, all the examples shown have been freeform prompts
 - Consist only of the prompt
- Structured prompts in Gen AI Studio allow you to add context and examples
 - Context tells the model more about your use case characteristics
 - Examples show the model how to structure the response

Gen AI Studio UI for structured prompts

The screenshot shows the Gen AI Studio interface for creating structured prompts. The top navigation bar includes a back arrow, the project name "bbq-examples", and tabs for "FREEFORM" and "STRUCTURED". The "STRUCTURED" tab is selected. The main area is divided into sections: "Context", "Test", and "Examples".

- Context:** A descriptive paragraph about a chef specializing in Southern Barbecue. A callout box labeled "Context adds information about the use case" points to this section.
- Test:** A section for testing prompts. It contains two rows:
 - INPUT:** "Add an input prefix" followed by the prompt "Write a post about our latest Smoked Brisket video."
 - OUTPUT:** "Add an output prefix" followed by the response "Hey y'all, get ready for a flavor-packed ride down South with my latest YouTube video: Smoked Brisket Low and Slow!"A callout box labeled "Examples demonstrate desired output" points to the input row, and another callout box labeled "Prompt and response" points to the output row.
- Examples:** A section for adding examples. It has fields for "INPUT" and "OUTPUT". The "INPUT" field contains "Write a post about our new video about smoking Prime Rib for the holidays." The "OUTPUT" field contains "Y'all ready for some mouth-watering Prime Rib? It's the perfect centerpiece for your holiday feast, and it's so easy". A blue button labeled "+ ADD TO EXAMPLES" is located next to the output field.
- Parameters:** On the right, a sidebar titled "Select Structured" contains the following settings:
 - Model:** text-bison@001
 - Temperature:** 0.5
 - Token limit:** 256
 - Top-K:** 40
 - Top-P:** (unspecified)Below these are buttons for "SUBMIT!" and "RESET PARAMETERS".
- Report Inappropriate Responses:** A section explaining that content is assessed against safety attributes, with a note that confidence scores are returned in API responses but are not visible in the UI.

Quality of the output can be improved by providing context to the prompt

- Context can be a few words or a paragraph
- You can specify information about you or your organization
- Specify a genre or style for the results

Here are the results of a prompt with no context

INPUT	Add an input prefix	OUTPUT	Add an output prefix
			<input checked="" type="checkbox"/> Markdown
	Write for me an ad for our new orange vest.	<p>Introducing the new orange vest, the perfect way to stay safe and visible on the job.</p> <p>Our orange vest is made from durable, water-resistant material that will keep you protected from the elements. It features reflective strips for added visibility, and a comfortable fit that won't restrict your movement.</p> <p>Whether you're working on a construction site, a farm, or anywhere else where safety is important, our orange vest is the perfect choice for you. Order yours today!</p> <p>Benefits of our orange vest:</p> <ul style="list-style-type: none">• Durable, water-resistant material• Reflective strips for added visibility• Comfortable fit• Perfect for a variety of jobs <p>Order your orange vest today and stay safe on the job!</p>	

Let's see how adding context changes the output

▼ Context ?

You work writing ads in the marketing department for a hunting and fishing supply company. The name of the company is Deep Creek Outfitters. The web site is: www.deepinthecreek.com.

The phone number is: (123) 123-4567

Deep Creek Outfitters

New Orange Vest

Stay Safe and Visible in the Field

Our new orange vest is the perfect way to stay safe and visible in the field. Made from durable, water-resistant material, this vest is perfect for hunting, fishing, or any other outdoor activity. It features a front zipper closure, two front pockets, and a back pocket with a flap. The vest is also adjustable, so you can find the perfect fit.

Order your new orange vest today from Deep Creek Outfitters!

www.deepinthecreek.com

(123) 123-4567

Let's try a different context

▼ Context

You write ads for the marketing department of a company called StyleUp that sells clothing targeted at young people in their teens and 20s.

You sell everything online. The website is: www.styleup.cool

Introducing the new orange vest from StyleUp!

This vest is perfect for anyone looking for a stylish and functional piece of outerwear. It's made from a lightweight, water-resistant fabric that will keep you warm and dry on even the coldest days. The orange color is eye-catching and sure to turn heads, and the relaxed fit means you can move freely without feeling restricted.

The vest features two front pockets and a zippered chest pocket, so you have plenty of space to store your belongings. It also has a removable hood that can be zipped on or off depending on the weather.

Whether you're running errands, going to class, or just hanging out with friends, the new orange vest from StyleUp is the perfect way to stay warm and stylish. Order yours today!

Shop now at www.styleup.cool

Let's try requesting a style of output

▼ Context

You write ads for the marketing department of a company called StyleUp that sells clothing targeted at young people in their teens and 20s.

You sell everything online. The website is: www.styleup.cool

Your ads are always written in the style of Dr. Seuss.

Oh, the places you'll go In your orange vest!
You'll climb mountains high And swim in the sea.

You'll wear it to school And to the mall. You'll wear it to parties And to the ball.

Your orange vest is so stylish And so cool. It's the perfect way to show Your personal style.

So what are you waiting for? Order your orange vest today! You'll be glad you did.

Visit www.styleup.cool to order yours now!

Examples demonstrate what the output should look like

- Use a history of what you have written in the past to get the model to output content in your style
 - Zero-shot prompting means there are no examples
 - One-shot prompting means there is one example
 - Few-shot prompting means there are multiple examples

Let's add some context and some examples and see how it affects output

Context ?

You work for a chef specializing in Southern Barbecue with a large audience on YouTube and social media. You post on Social Media about the latest videos. You like to post with a Southern Style.

Examples ? III ADD INPUT COLUMN

INPUT	OUTPUT
Add an input prefix	Add an output prefix
Write a post about our latest Smoked Brisket video.	Get ready for a flavor-packed ride down South with my latest YouTube video: Smoked Brisket Low and Slow! Slow-smoked to juicy perfection, seasoned with secret spices. Mouthwatering sides included! Join me on the grill, y'all! #SouthernBBQ #SmokedBrisket #BBQPassion

INPUT Add an input prefix

Write a post about our new video about smoking Prime Rib for the holidays. MICROPHONE

OUTPUT Add an output prefix - Markdown

It's the most wonderful time of the year, and what better way to celebrate than with a big ol' slab of smoked prime rib? Check out my new YouTube video for all the details on how to smoke the perfect prime rib for your holiday feast. #SmokedPrimeRib #HolidayFeast #BBQPassion

Use the other model parameters to further customize the results

- Temperature
- Token limit
- Top-K
- Top-P

The screenshot shows a configuration interface for model parameters. It includes five sliders with their current values displayed in boxes:

- Temperature: 0.2
- Token limit: 256
- Top-K: 40
- Top-P: 0.8
- Safety filter threshold: Block few

Below the sliders are two buttons: SUBMIT and RESET PARAMETERS. At the bottom, there is a link to REPORT INAPPROPRIATE RESPONSES and a note about content processing.

Content processed through Vertex AI is

Temperature controls the degree of randomness in token selection

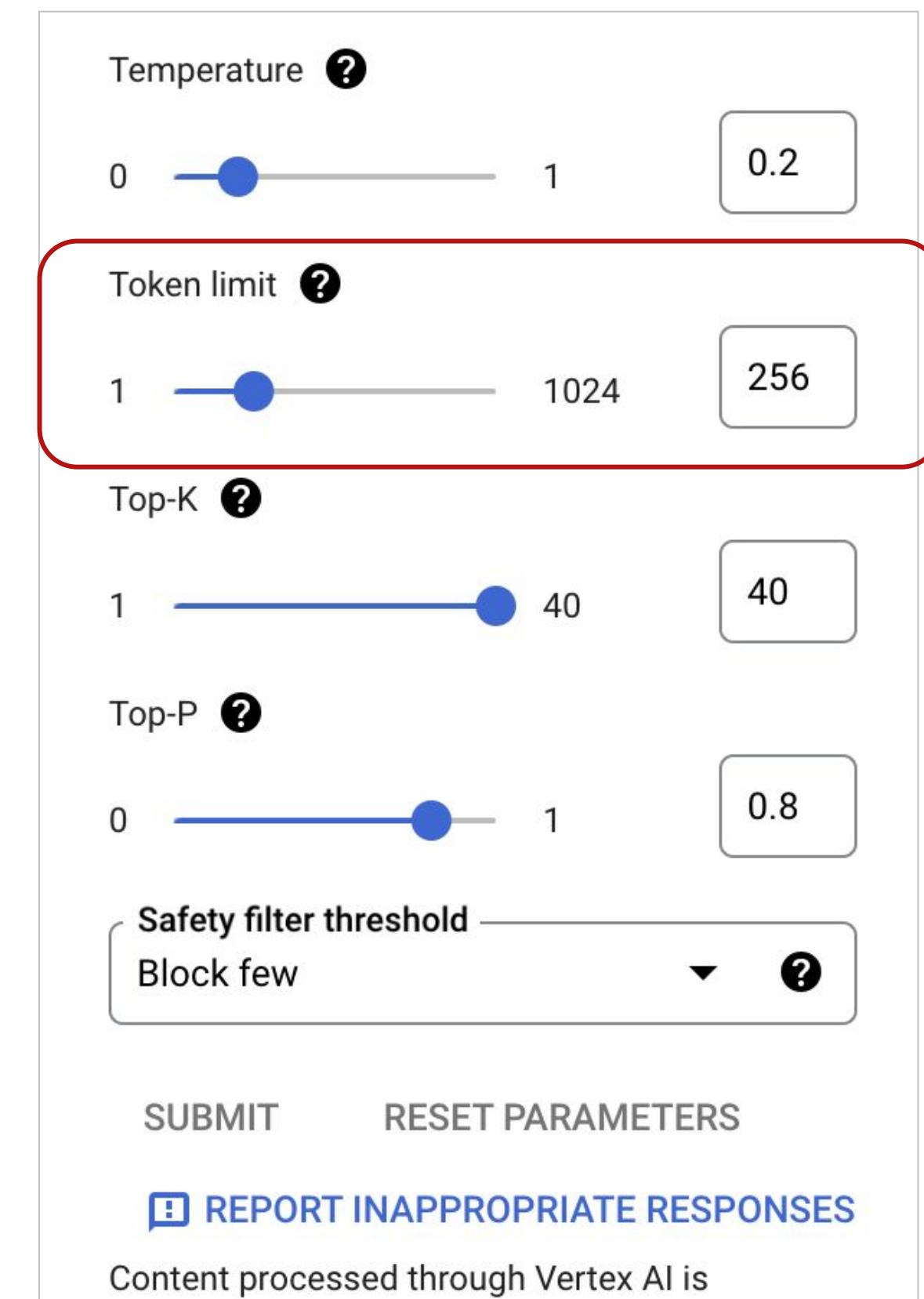
- Range from 0 to 1
- Lower temperatures are good for prompts that require correctness
 - More likely to select the most expected next token
- Higher temperatures can lead to more diverse or unexpected results
 - More creative
- A temperature of 0 is deterministic
 - The highest probability token is always selected
 - Will always return the same result for a given prompt
- Default is a temperature of .2
 - Start there and experiment with your results

The screenshot shows the Vertex AI Text Generation interface with several configuration options:

- Temperature**: A slider set to 0.2.
- Token limit**: A slider set to 256.
- Top-K**: A slider set to 40.
- Top-P**: A slider set to 0.8.
- Safety filter threshold**: Set to "Block few".
- SUBMIT** and **RESET PARAMETERS** buttons.
- REPORT INAPPROPRIATE RESPONSES** button.
- A footer message: "Content processed through Vertex AI is".

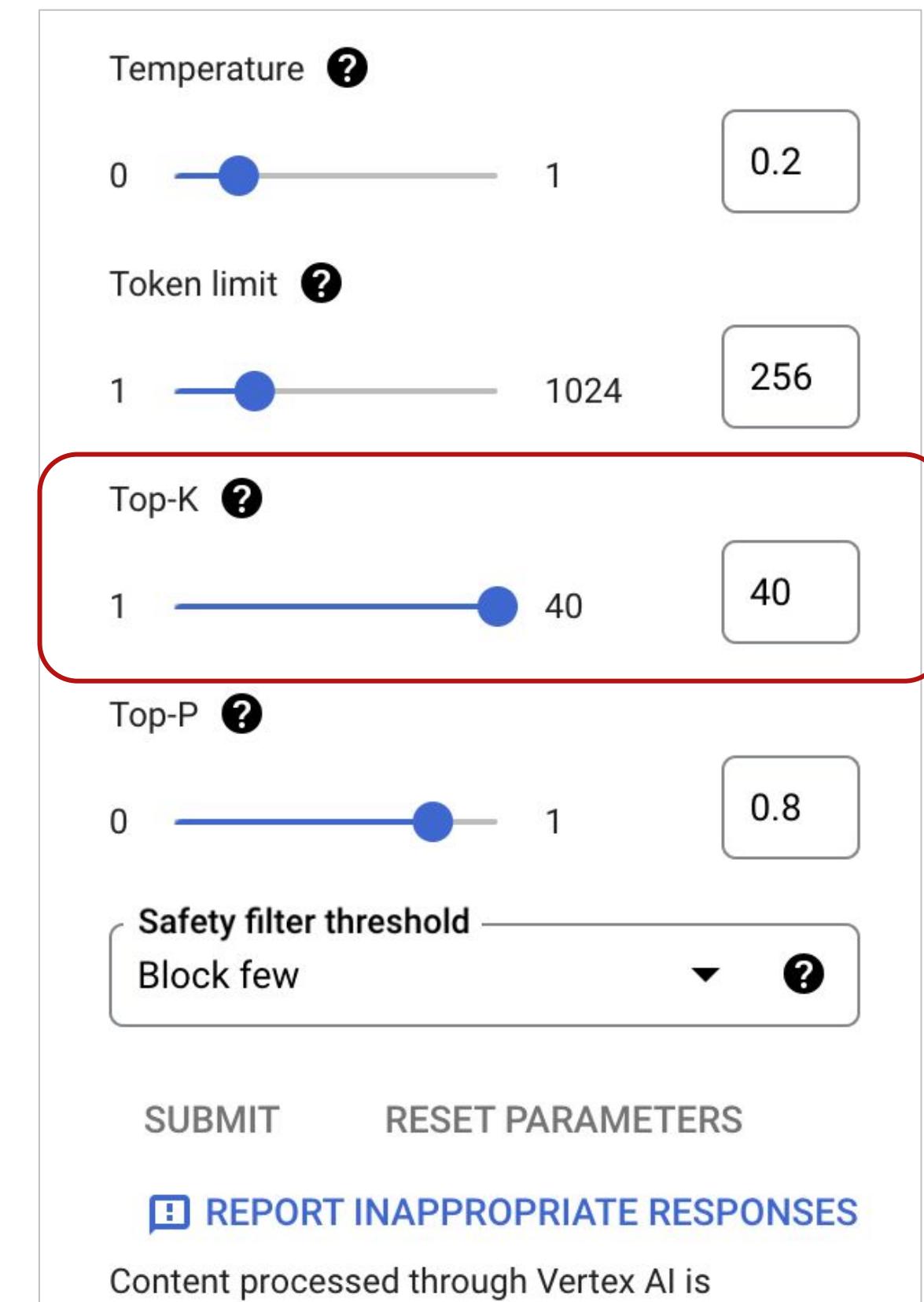
Token limit determines the maximum amount of output for each prompt

- The range is from 1 to 1024
 - The default value is 256
- A token is a basic unit of text that the LLM understands
 - When using the PaLM API, a token is approximately four characters
- A token is a string of characters that represents a concept or object in GenAI
 - For example, the token "dog" represents the concept of a dog



Top-K changes how the model selects tokens for output

- Range is from 1 to 40
- A top-K of 1 means the selected token is the most probable among all tokens in the model's vocabulary
 - Called greedy decoding
 - See: https://www.tensorflow.org/text/guide/decoding_api
- A top-K of 3 means that the next token is selected from among the 3 most probable tokens
- The default top-K value is 40
- The higher the value the more tokens are possible when selecting the next token



Top-P also changes how the model selects tokens for output

- Range is from 0 to 1
- Tokens are selected from most probable to least until the sum of their probabilities equals the top-P value
 - For example, if tokens A, B, and C have a probability of .3, .2, and .1 and the top-P value is .5, then the model will select either A or B as the next token
- The default top-P value is .8
- The higher the value the more likely the model would be to select a token that is not the most probable one

The screenshot shows the 'Model Parameters' section of the Vertex AI interface. It includes the following settings:

- Temperature: 0.2
- Token limit: 256
- Top-K: 40
- Top-P: 0.8 (highlighted with a red border)
- Safety filter threshold: Block few

At the bottom are 'SUBMIT' and 'RESET PARAMETERS' buttons, and a link to 'REPORT INAPPROPRIATE RESPONSES'.

Content processed through Vertex AI is

Topics

01

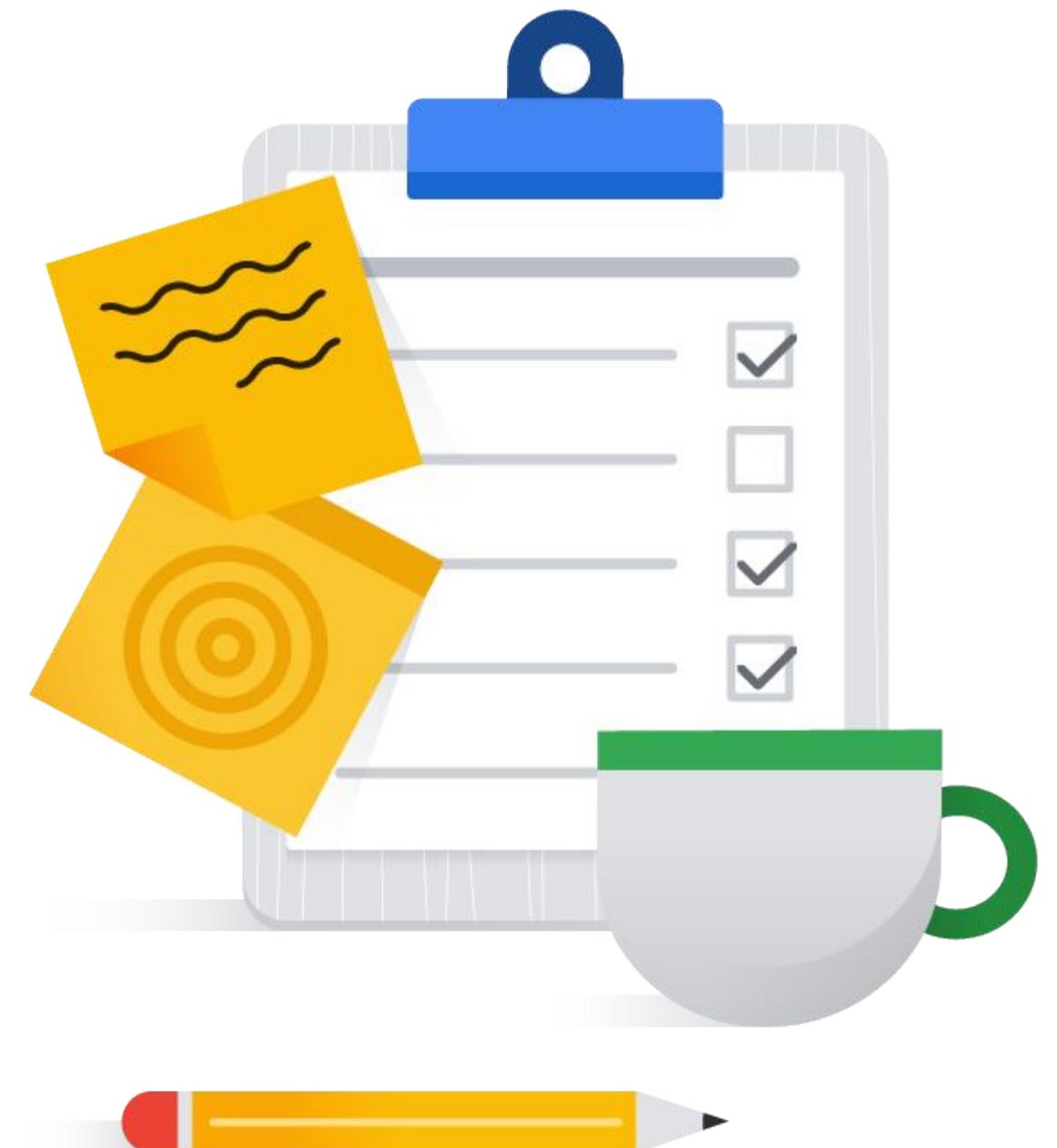
Understanding Prompt Design

02

Context, Examples, and Fine-Tuning Parameters

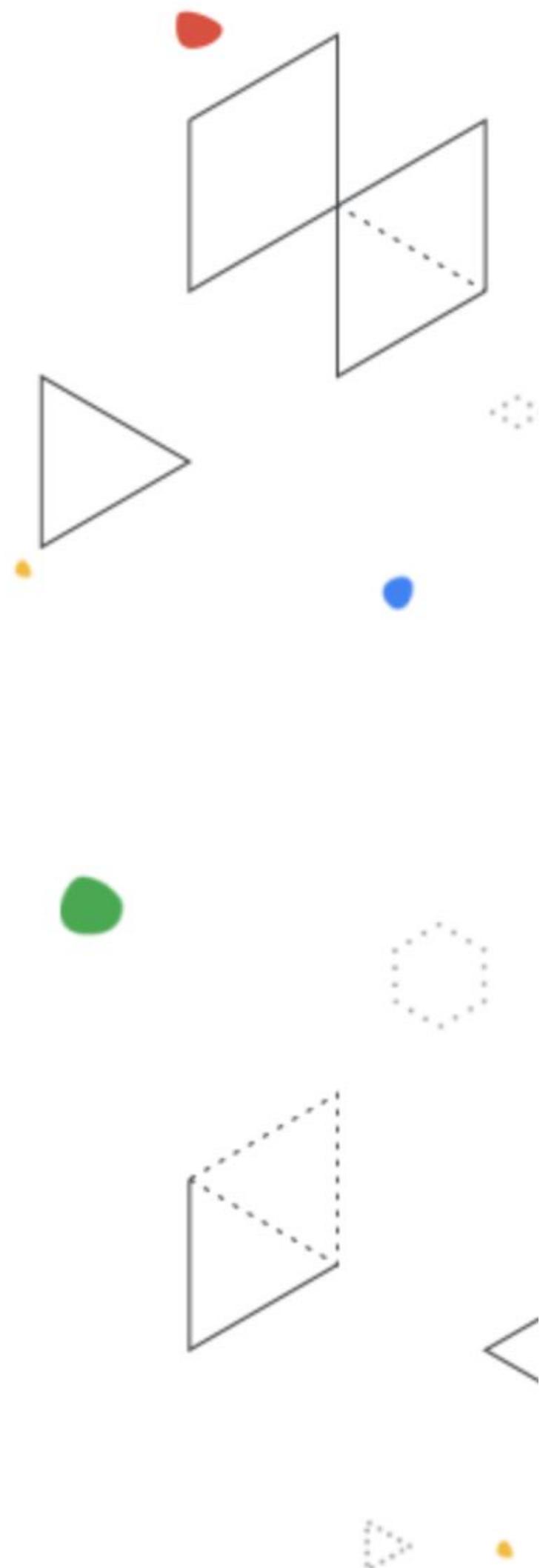
03

Prompt Design Best Practices



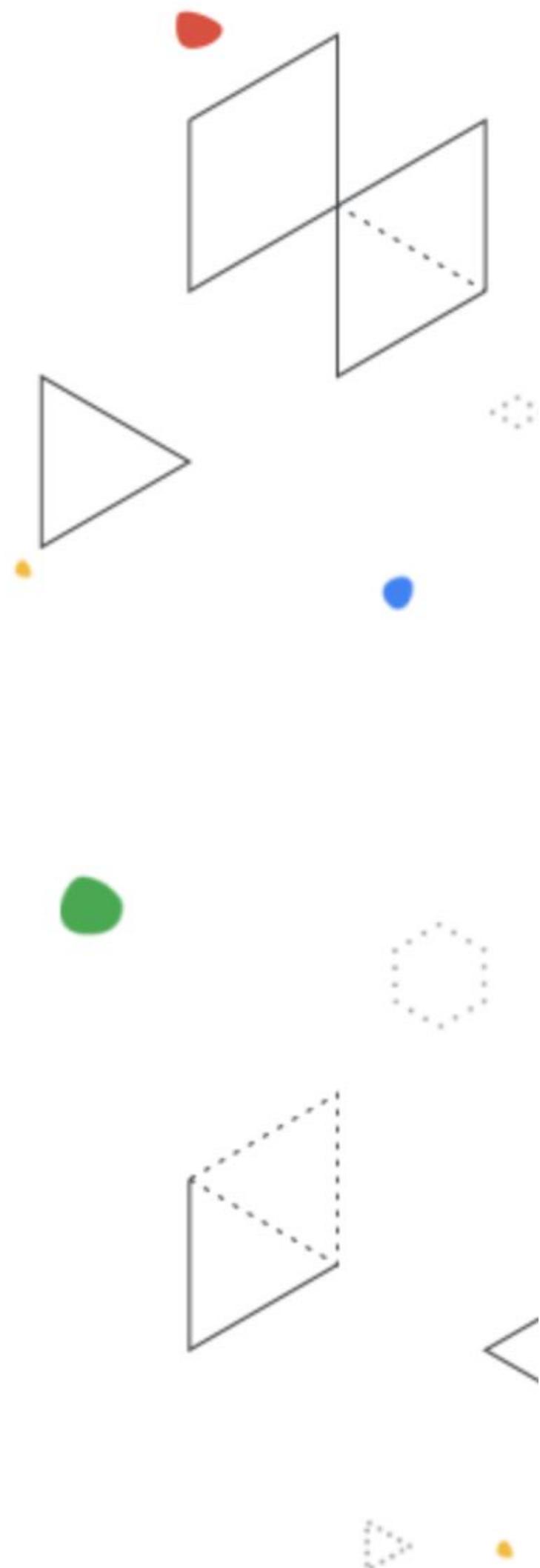
Prompt design best practices

- Be specific
 - Don't ask: "Write me an a poem"
 - Ask: "Write me a poem about COBOL programming in the style of Edgar Allan Poe"
- Add context
 - Add information about "who" you want the model to emulate
 - What is its job, where does it work, and any other information you would want it to convey
- Add examples
 - Use examples that you have written in the past to teach it to write in your style
 - Give examples of how it should answer questions



Be clear and concise

- Use positive language
 - Don't say: "Don't use long sentences"
 - Say: "Write short sentences as bullets"
- Use simple language
 - Avoid jargon
 - Avoid complex sentences
- Use correct grammar and spelling



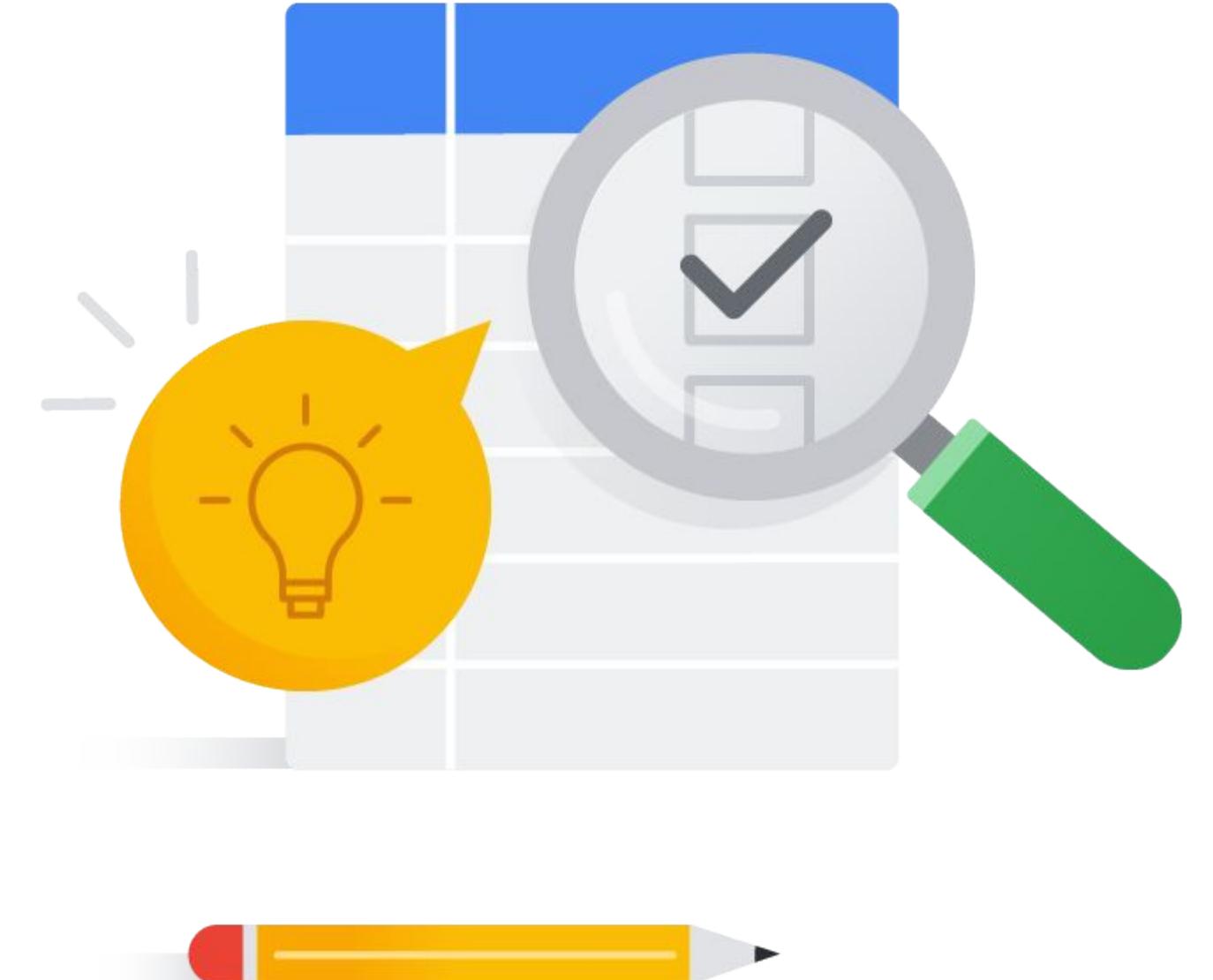
Experiment with parameter values

- To make the model more creative, set higher values for Temperature, Top-K, and Top-P
- Set lower values when you want the model to be consistent
- A Temperature of 0, Top-K to 1 , and Top-P to 0 would cause the model to always return the same result for a given prompt based on the most likely next token

Lab

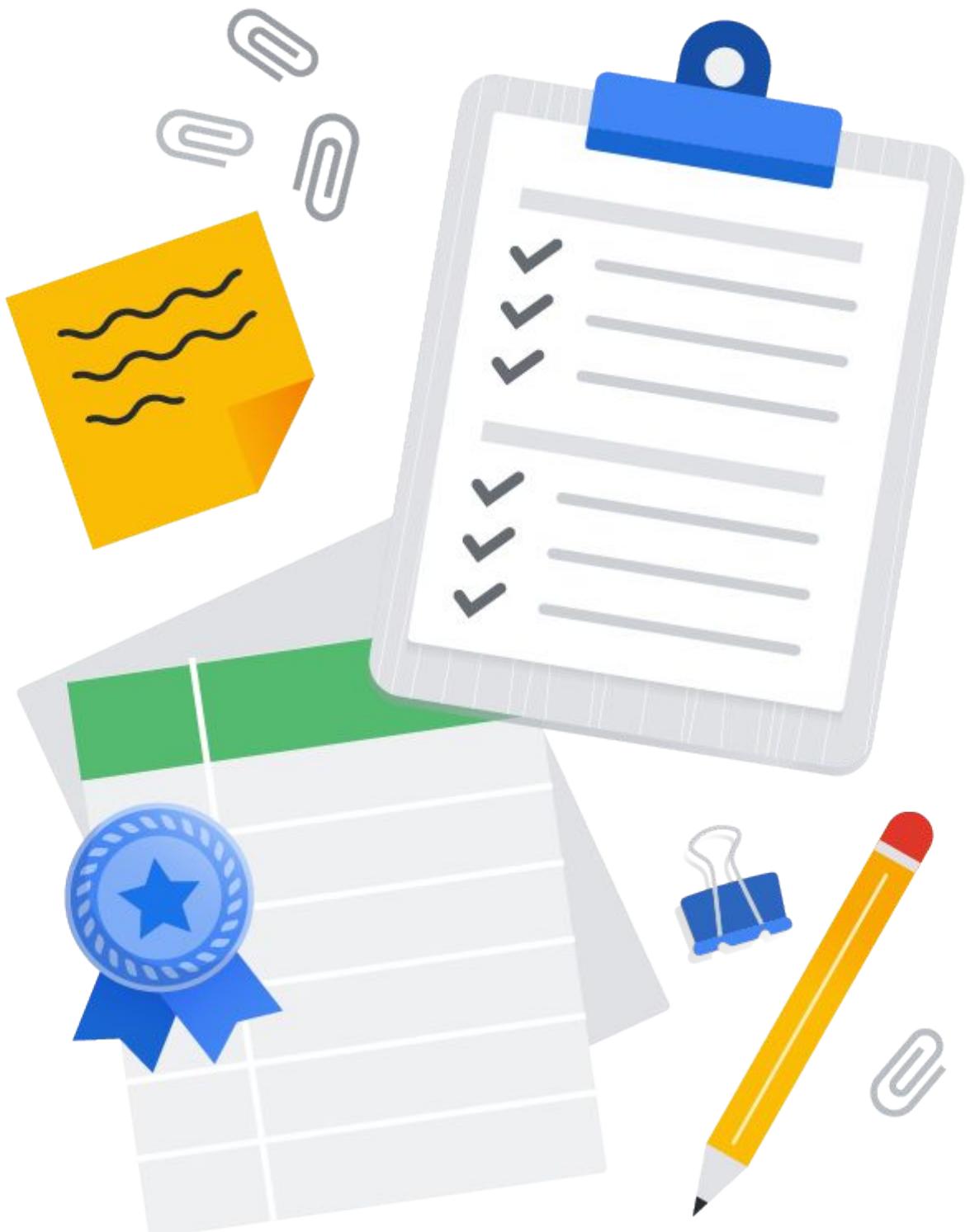
⌚ 30 min ⚙️

Prompt Design for Text Generation in Gen AI Studio

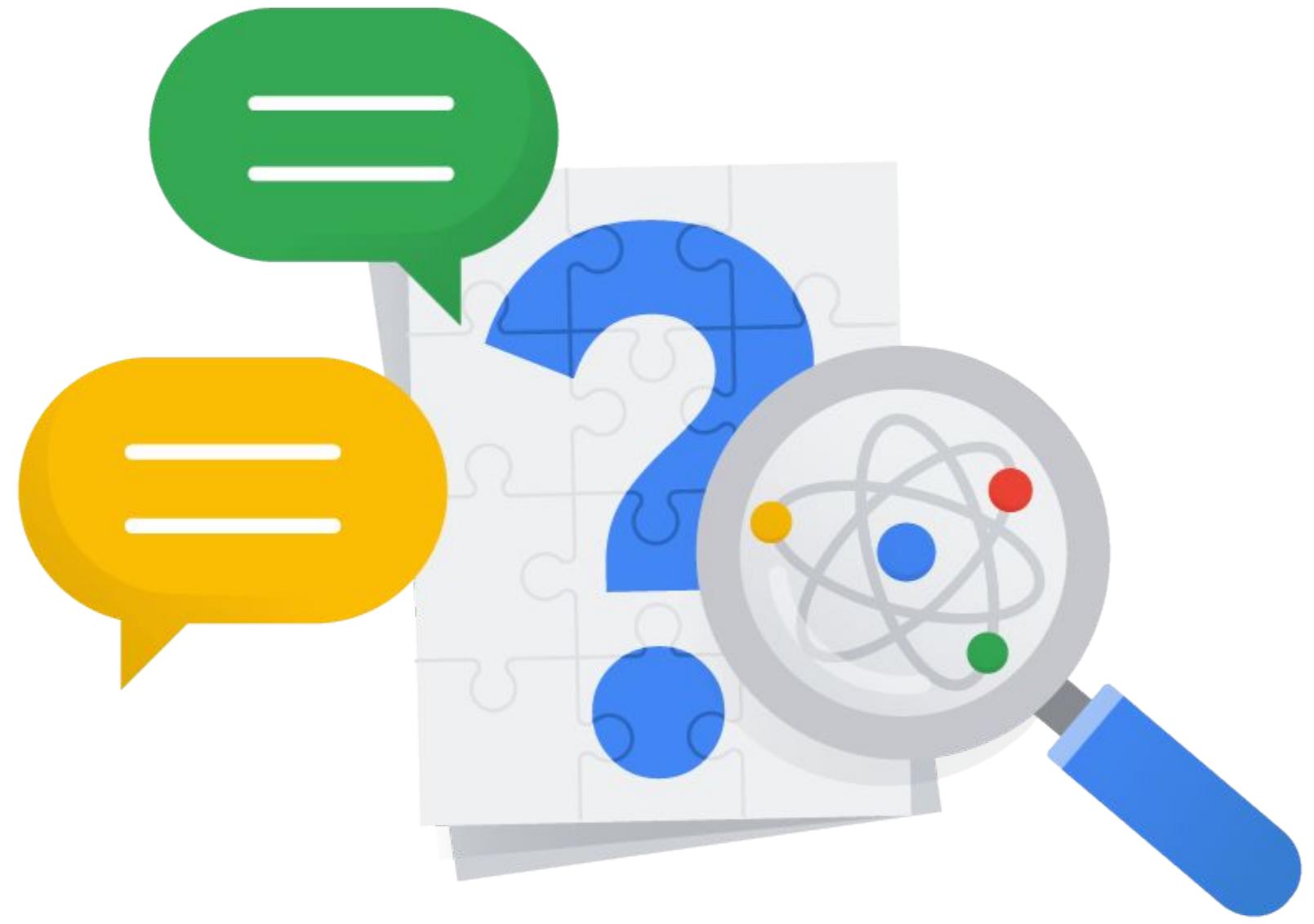


In this module, you learned to ...

- 01 Design prompts for various Gen AI use cases
- 02 Add examples and context to customize generative AI output
- 03 Experiment with parameters for model fine-tuning



Questions and answers



Quiz question

What's the difference between
freeform and structured prompts?

Quiz question

What's the difference between freeform and structured prompts?

With structured prompts you can add context and examples.

Quiz question

- Which setting for temperature would make the model more creative when answering prompts?
- A: -1
 - B: 0
 - C: 0.1
 - D: 0.9

Quiz question

- Which setting for temperature would make the model more creative when answering prompts?
- A: -1
 - B: 0
 - C: 0.1
 - D: 0.9

Quiz question

If you want the model to write in your style what should you do?

- A: Set the temperature to about 0.5
- B: Raise the Top-K and Top-P values
- C: Add context
- D: Add examples

Quiz question

If you want the model to write in your style what should you do?

- A: Set the temperature to about 0.5
- B: Raise the Top-K and Top-P values
- C: Add context
- D: Add examples



Implementing the PaLM API

In this module, you learn to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

Program Python applications that use PaLM to generate content

03

Integrate PaLM and GenAI into your applications



Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



PaLM is a Large Language Model (LLM)

- LLMs are very sophisticated autocomplete applications
 - They learn patterns from large amounts of text
 - Use those patterns to generate text
- When generating text they calculate the next most likely tokens (words)
 - They aren't smart; it's math and statistics
- PaLM can generate text with two basic services
 - Text service for single request interactions
 - Chat service is for interactive, multi-turn interactions

The PaLM API can be used via a REST service call

- cURL is a program that makes web requests
 - -X parameter is the HTTP verb (GET, POST, PUT, DELETE, etc.)
 - -H parameters adds header variables
 - -d parameter is the body of the request, in this case the prompt

```
curl \  
-X POST \  
-H "Authorization: Bearer $(gcloud auth print-access-token)" \  
-H "Content-Type: application/json" \  
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/us-central1/publishers/google  
/models/${MODEL_ID}:predict" -d \  
$'{  
    "instances": [{"content": "Write me a poem about FORTRAN programming"}],  
    "parameters": {  
        "temperature": 0.2,  
        "maxOutputTokens": 256,  
        "topP": 0.8,  
        "topK": 40}'}  
  
{{<<code omitted>>}}
```

You can use the API from various programming languages including Python, Node.js, Java, and Swift

Python:

```
completion = palm.generate_text(model=model, prompt=x)
```

Node.js:

```
client.generateText({
  model: MODEL_NAME,
  prompt: {
    text: prompt,
  },
})
.then((result) => {
  console.log(JSON.stringify(result));
});
)
```

To use the PaLM API, your application will need to be authenticated

- Ways to authenticate your application
 - Obtain an authorization token
 - Run the application using a service account

An authorization token identifies the caller of an API

- Created using the Google Cloud CLI
 - The gcloud CLI must be initialized with either a user or service account
- Set the Authorization header variable with the token generated using gcloud

```
curl -H 'Content-Type: application/json' \
      -H "Authorization: Bearer $(gcloud auth print-access-token)" \
      -d '{"prompt": {"text": "Give me five subcategories of jazz"} }' \
      "https://generativelanguage.googleapis.com/v1beta2/models/text-bison-001:generateText"
<<code omitted>>
```

If running an application in Google Cloud, assign a service account to the runtime

- Create a service account using **IAM**
 - Assign the **Vertex AI Service Agent** role
 - Use the service account to identify the runtime
- If using Cloud Run, App Engine, or Cloud Functions, the runtime will use the Compute Engine Default Service Account by default
 - This will work as it uses the Editor role
 - Violates principle of least privilege
- You can also download Service Account keys to authenticate programs that use the language client libraries

The screenshot shows the process of creating a new service account in Google Cloud's IAM. It consists of two main sections:

- 1 Service account details**:
 - Service account name: vertext-ai-sa
 - Display name for this service account: (empty)
 - Service account ID *: vertext-ai-sa
 - Email address: vertext-ai-sa@vertext-ai-dar.iam.gserviceaccount.com
- 2 Grant this service account access to project (optional)**:
 - Role: Vertex AI Service Agent
 - IAM condition (optional): + ADD IAM CONDITION
 - Description: Gives Vertex AI the permissions it needs to function.

Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



You can use Gen AI Studio to generate Python code for a Python app or Jupyter

- Click the **View Code** button and select Python or Python Colab
 - Colab is a free Google Jupyter service online

View code

PYTHON

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    "",
    **parameters
)
print(f"Response from Model: {response.text}")
```

View code

PYTHON PYTHON COLAB

Use this script to request a model response with a Colaboratory notebook.

1. Install the [Vertex AI SDK](#) for Python. After installing you'll be prompted to restart your Colab notebook runtime.
`!pip install "shapely<2.0.0"`
`!pip install google-cloud-aiplatform >= 1.25.0`
2. Use the following code in your notebook to request a model response

```
from google.colab import auth as google_auth
google_auth.authenticate_user()

import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="vertext-ai-dar", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    "",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Vertex AI requirements for Python

- Use pip to install Google Cloud AI Platform
 - Or add to your requirements.txt file

```
pip install google-cloud-aiplatform >= 1.25.0
```

- Import Vertex AI

```
import vertexai
from vertexai.language_models import TextGenerationModel
```

Using Vertex AI from Python

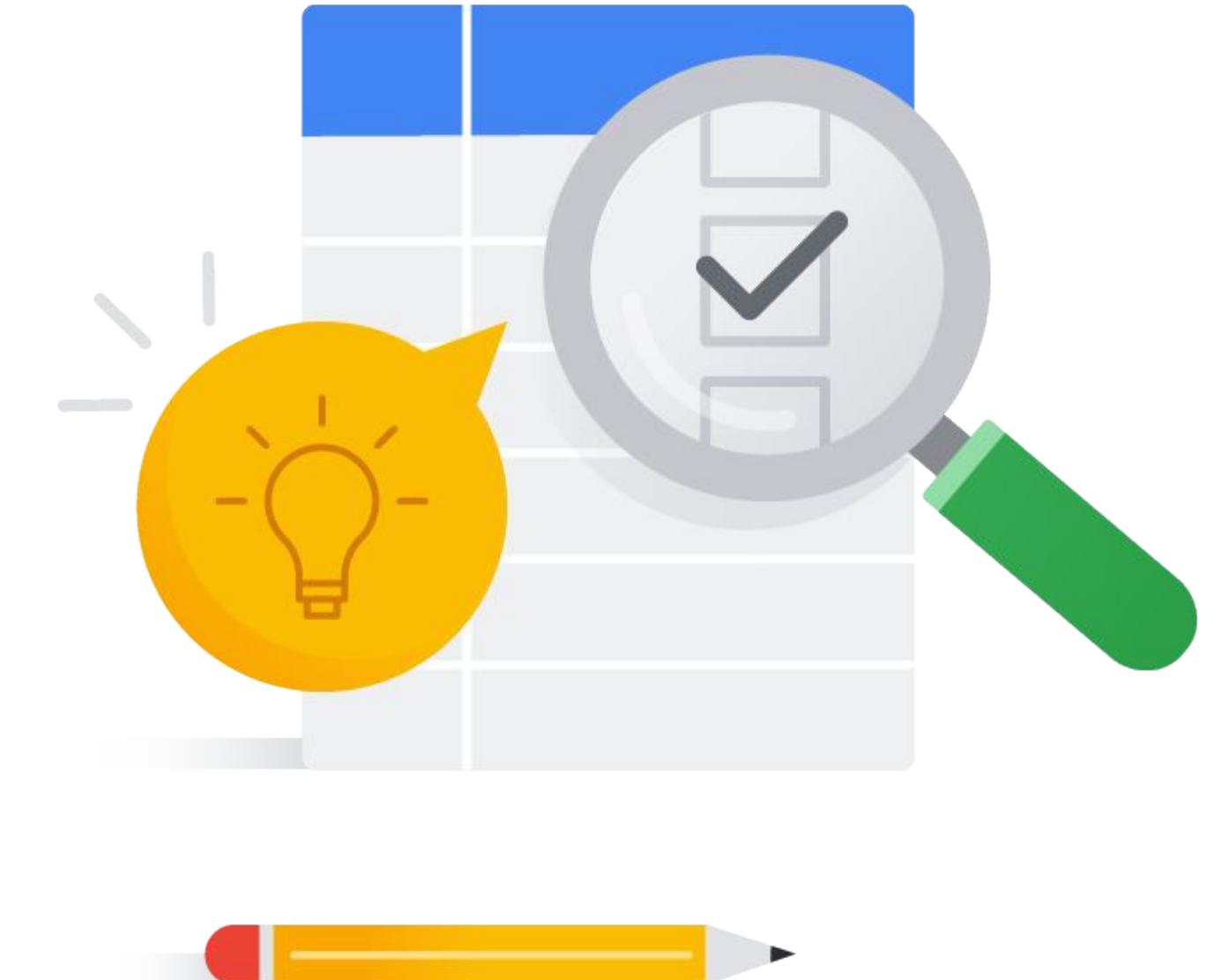
```
vertexai.init(project="vertext-ai-dar", location="us-central1")

parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
model = TextGenerationModel.from_pretrained("text-bison@001")
response = model.predict(
    """Write for me a limerick about dogs and Python programming.""",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Lab

⌚ 30 min ⚙️

Getting Started with the PaLM API



Topics

01

Introduction to the PaLM API

02

Generative AI Powered Applications with Python

03

Using the PaLM API in Applications



Python Flask Website example

- This is an example of using the text service with the PaLM API
 - Even though you may ask many questions, each one is independent
- Context must be added to tell the PaLM API to emulate a barista
- The coding is simple as you are just submitting an HTML form and making a request to the PaLM API for a response
 - The response is displayed on the screen

The image displays two screenshots of a Python Flask website for "CoffeeBot".

Screenshot 1: The top part shows the homepage of CoffeeBot. It features a header with the text "CoffeeBot" and "Your friendly online BaristAI". Below this is a green callout box containing the text: "I am CoffeeBot, a barista and expert on all things related to coffee and tea. I can help you find the perfect coffee or tea for your taste, and I can also teach you how to make your own coffee and tea drinks at home." The bottom part shows a form with the text "Ask CoffeeBot:" followed by an input field and a "Submit" button.

Screenshot 2: The bottom part shows the result of a query. It features a header with the text "CoffeeBot" and "Your friendly online BaristAI". Below this is a green callout box containing the text: "To make a latte, you will need: * 2 shots of espresso * 6 ounces of steamed milk * 1 tablespoon of foamed milk 1. Brew the espresso shots. 2. Steam the milk until it is hot and frothy. 3. Pour the espresso into a latte glass. 4. Add the steamed milk to the espresso. 5. Top with the foamed milk. 6. Enjoy!" The bottom part shows a form with the text "Ask CoffeeBot:" followed by an input field containing the question "How do you make a Latte?" and a "Submit" button.

Add the Python requirements

- Add Google Cloud AI Platform to the requirements.txt file
- Add the required imports at the top of the code file

```
requirements.txt ×  
coffeebot > requirements.txt  
1 Flask  
2 Jinja2  
3 pytest  
4 pyyaml  
5 google-cloud-aiplatform
```

```
from flask import Flask, render_template, request  
import os  
import vertexai  
from vertexai.language_models import TextGenerationModel
```

Handling web requests in Flask

- The default route will handle HTTP posts and gets
 - Post means a question was submitted from the HTML form
 - Get means there is no question (have CoffeeBot introduce itself)
- The code for using the PaLM API is in the `get_response()` function

```
@app.route("/", methods = ['POST', 'GET'])
def main():
    if request.method == 'POST':
        input = request.form['input']
        response = get_response(input)
    else:
        input = ""
        response = get_response("Who are you and what can you do?")

    model = {"title": "CoffeeBot", "message": response, "input": input}
    return render_template('index.html', model=model)
```

Making a request to the PaLM API

```
def get_response(input):
    vertexai.init(project="vertext-ai-dar", location="us-central1")
    parameters = {
        "temperature": 0.8,
        "max_output_tokens": 256,
        "top_p": 0.8,
        "top_k": 40
    }
    model = TextGenerationModel.from_pretrained("text-bison@001")
```

Initialize the API
and set up the
parameters

Create the model
using the correct
version of the
PaLM API

<< CODE ON NEXT SLIDE OMITTED >>

Making a request to the PaLM API (continued)

```
def get_response(input):
    << CODE FROM PREVIOUS SLIDE OMITTED >>

    model = TextGenerationModel.from_pretrained("text-bison@001")
    request = """Your name is CoffeeBot. You are a barista and expert on
    all things related to coffee and tea..

    input: {}
    output: """
    response = model.predict(
        request.format(input),
        **parameters
    )
    return response
```

The context tells
the API to
emulate a
bartender

The input is the
question

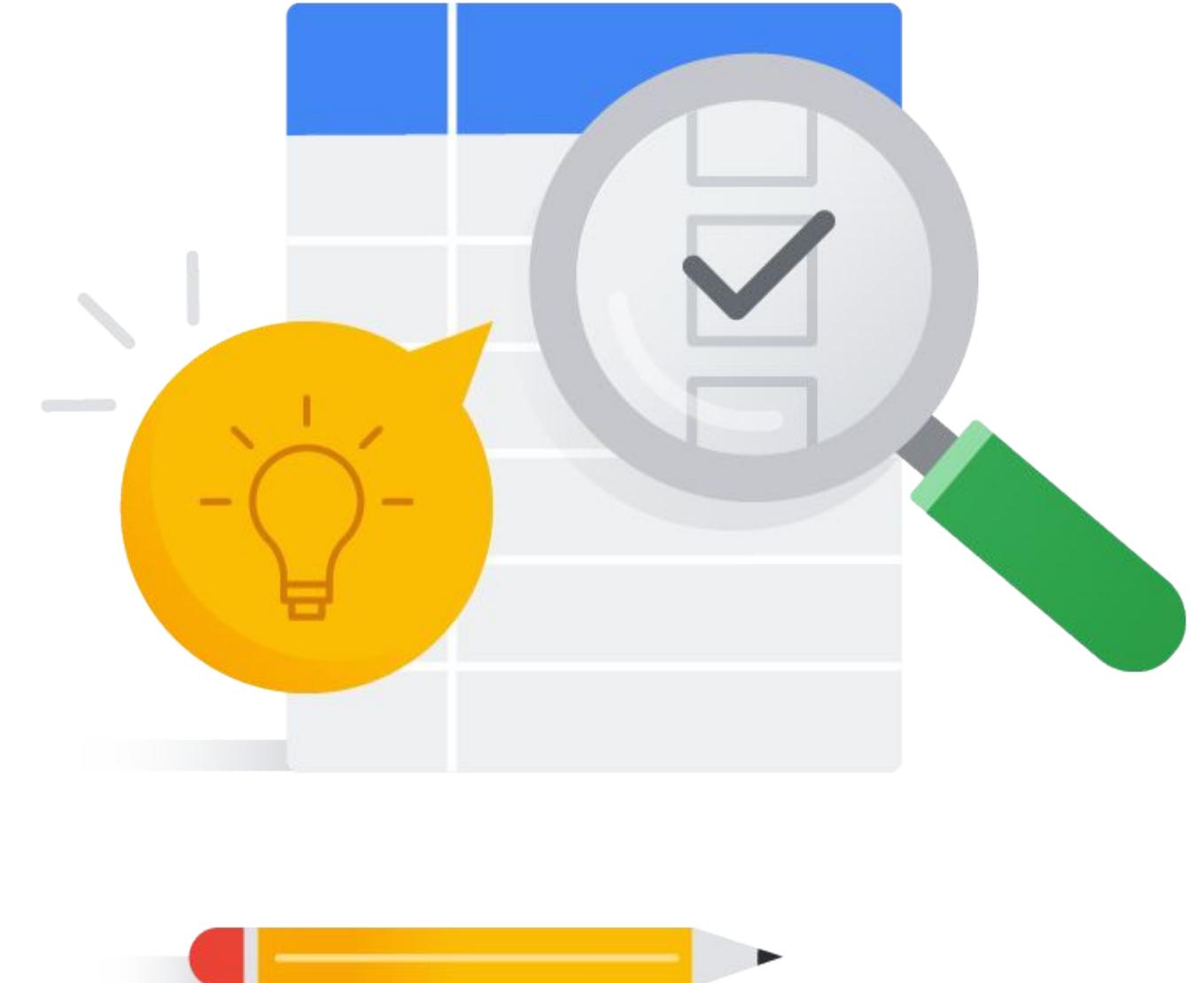
Call the model.predict()
function to send the request

The format function injects the
question into the prompt

Lab

⌚ 30 min ⚙️

Integrating the PaLM API into Applications



In this module, you learned to ...

01

Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02

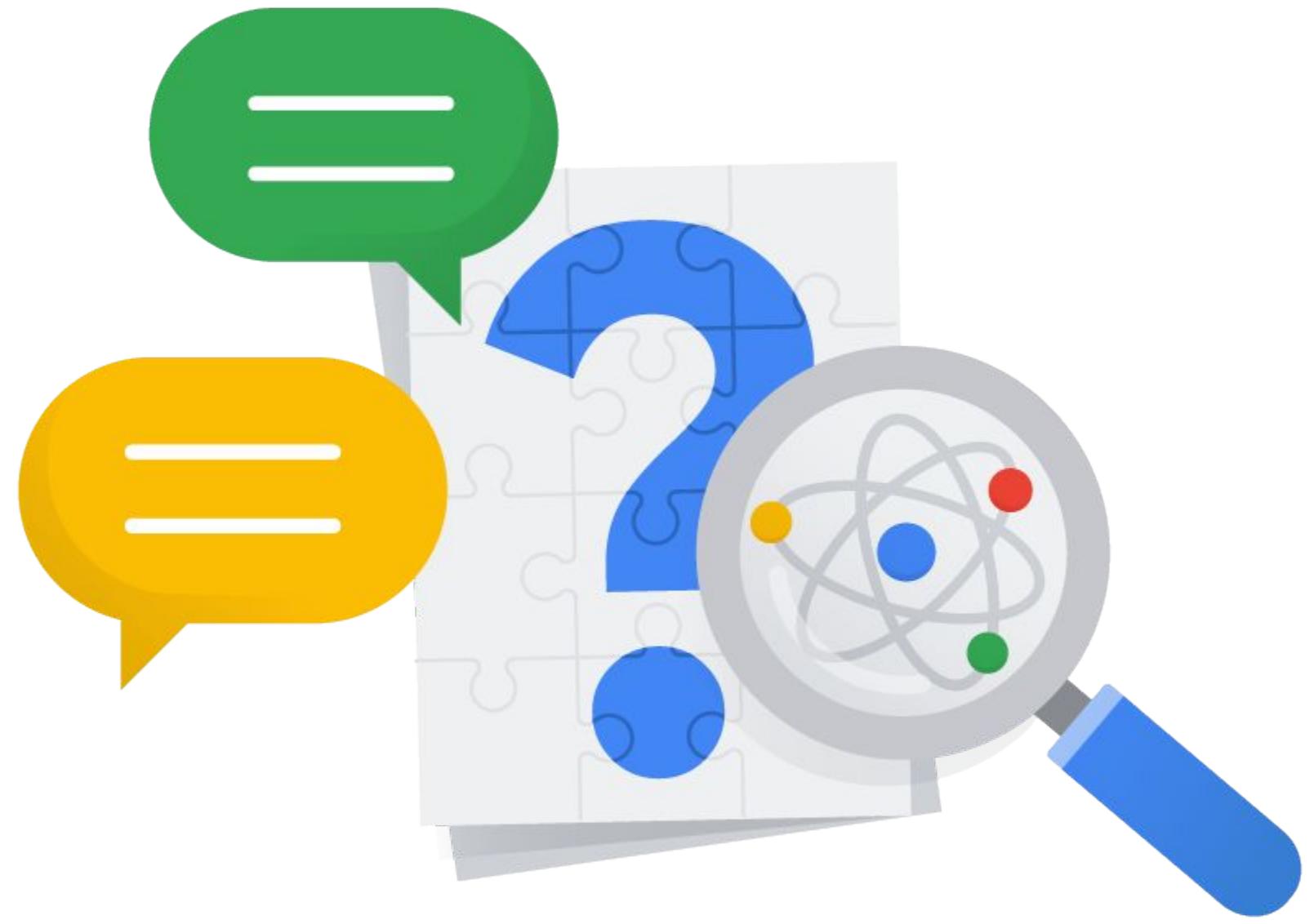
Program Python applications that use PaLM to generate content

03

Integrate PaLM and GenAI into your applications



Questions and answers



Quiz question

- How do you authenticate a request to the PaLM API ?
- A: Using a API Key
 - B: Using an authorization token
 - C: With a Service Account
 - D: All of the above would work

Quiz question

- How do you authenticate a request to the PaLM API ?
- A: Using a API Key
 - B: Using an authorization token
 - C: With a Service Account
 - D: All of the above would work

Quiz question

What programming languages are
Supported by the PaLM API ?

- A: Python
- B: Node.js
- C: Swift
- D: Java
- E: All of the above

Quiz question

What programming languages are
Supported by the PaLM API ?

- A: Python
- B: Node.js
- C: Swift
- D: Java
- E: All of the above



Fine-tuning Models

In this module, you learn to ...

- 01 Evaluate scenarios for creating tuned models
- 02 Build workflows for tuning and deploying models
- 03 Use tuned models in your applications



Topics

01

Tuned Models

02

Preparing a Model Tuning Dataset

03

Creating a Tuning Job



Use model tuning to improve model performance on specific tasks

- When few-shot prompting and adding context are not adequate to your use case
- Allows you to teach the model more about what your expected output should be
- You specify a custom dataset which includes prompts along with the expected output
 - Like adding examples, but more of them with custom training
- The custom training jobs learn the outputs (called weights)

Tuning is required when you want output that deviates from general language patterns

- Specific structures or formats for generating output
- Specific behaviors such as when to provide a terse or verbose output
- Specific customized outputs for specific types of inputs

When custom training may be required:

Classification

- Classification with custom classes (groups)
 - Give the model examples, with the correct answers

input_text:

Classify the following text into one of the following classes:
[HR, Sales, Marketing, Customer Service].

Text: Are you currently hiring?

output_text:

HR

When custom training may be required:

Summarization

- Summaries that require specific output
- In the example below, you want to remove personally identifiable information (PII) in a chat summary

```
input_text:
```

Summarize:

Jessica: That sounds great! See you in Times Square!

Alexander: See you at 10!

```
output_text:
```

#Person1 and #Person2 agree to meet at Times Square at 10:00 AM

When custom training may be required:

Extractive question answering

- The question is about a context and the answer is a substring of the context

input_text:

context: There is evidence that there have been significant changes in Amazon rainforest vegetation over the last 21,000 years through the Last Glacial Maximum (LGM) and subsequent deglaciation.

question: What does LGM stand for?

output_text:

Last Glacial Maximum

Including context in your training data

- In the example below, the **input_text** consists of both a **context** section and a **question** section

```
input_text:  
context: There is evidence that there have been significant changes in  
Amazon rainforest vegetation over the last 21,000 years through the Last  
Glacial Maximum (LGM) and subsequent deglaciation.  
question: What does LGM stand for?
```

- The context provides additional information for answering the question
- If your training data is formatted in this way, you must format prompts in the same way when using your model for inference
 - I.e. Whatever sections your training data has as input, must be included in prompts in the same order when using the model

Topics

01

Tuned Models

02

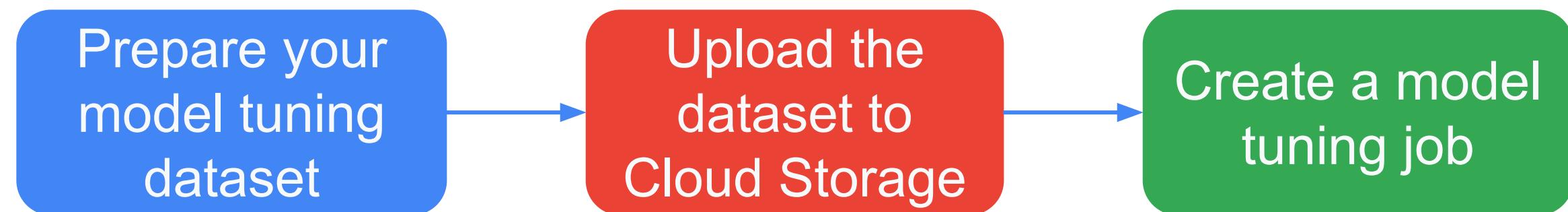
Preparing a Model Tuning Dataset

03

Creating a Tuning Job



Model tuning workflow on Vertex AI



- After tuning, the model is automatically deployed to a Vertex AI endpoint using the name you provide in the tuning job
- The model is also available in Gen AI Studio when creating prompts

Prepare your model tuning dataset

- The training data must be in JSONL format
 - The “L” is for “Line”
 - Each line in the JSONL file is one example
 - It is not an array of objects, it is one object per line
- Each object must have the properties `input_text` and `output_text`

```
{} custom-training-data.jsonl ×  
Users > doug > {} custom-training-data.jsonl  
1  {"input_text": "question: How many copies of Gears of War 3 were sold ? context: Like  
2  {"input_text": "question: How many parishes are there in Louisiana ? context: The U .
```

It is important to include instructions into the training data

- The following has no instructions, so it is not a good example

```
{"input_text": "5 stocks to buy now", "output_text": "business"}
```

- The following has instructions, so it is a better example

```
{"input_text": "Classify the following text into one of the following classes: [business, entertainment] Text: 5 stocks to buy now", "output_text": "business"}
```

Including context within the input text

- Notice that the following `input_text` has question and context sections
 - When using the model, remember that prompts need to be formatted the same way
 - Be consistent

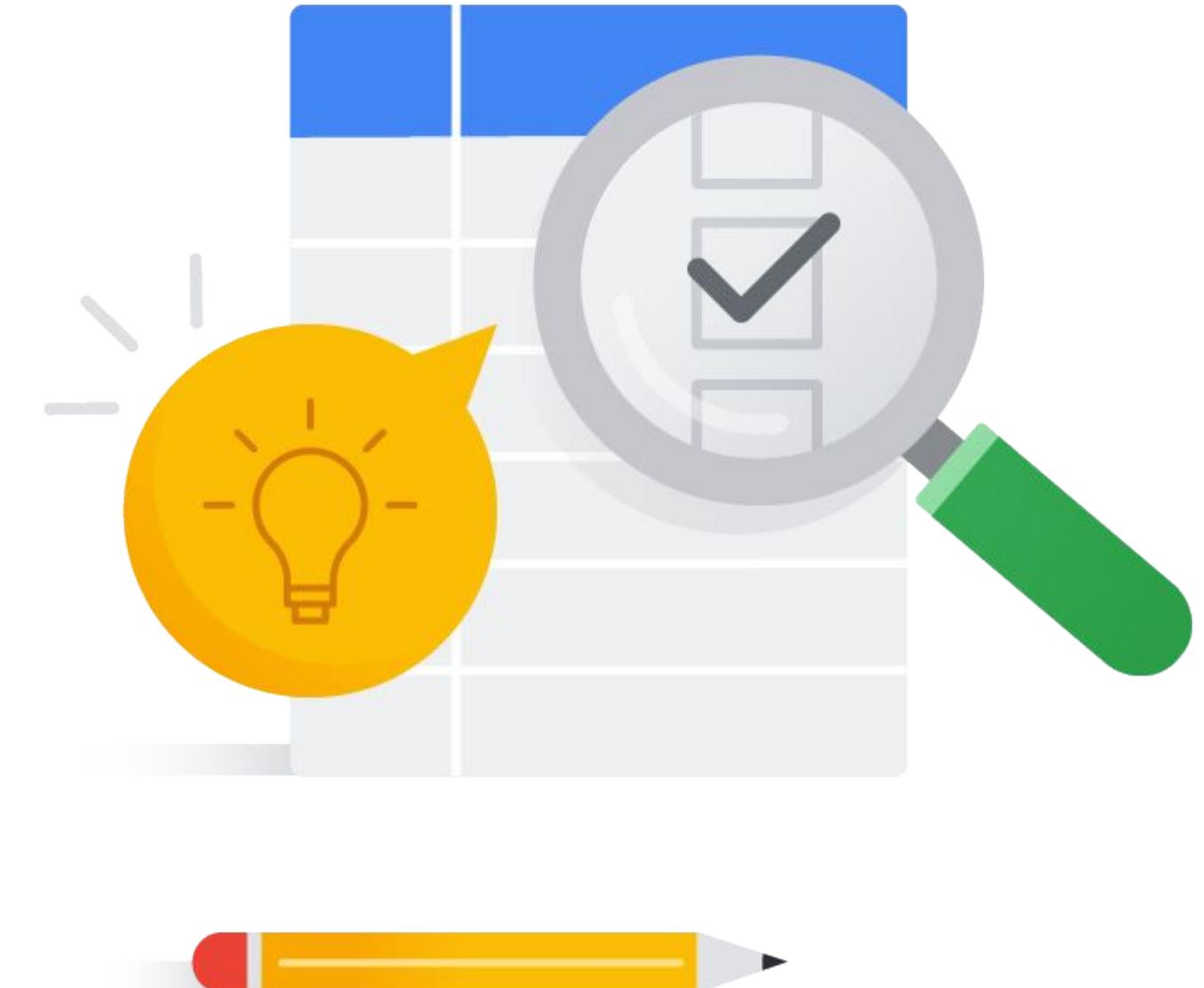
```
{"input_text": "question: How many parishes are there in Louisiana? context: The U.S. state of Louisiana is divided into 64 parishes (French: paroisses) in the same manner that 48 other states of the United States are divided into counties, and Alaska is divided into boroughs.", "output_text": "64"}
```

Do Now: Exploring Sample Training Data

5 min



1. Go to:
<https://github.com/roitraining/genai-model-tuning-examples>
2. You will find some example fine-tuning datasets
3. Click on a couple of them and explore the examples
 - a. Each file has 1 example per line
 - b. Each example has input_text and output_text attributes



Topics

- 01 Tuned Models
- 02 Preparing a Model Tuning Dataset
- 03 Creating a Tuning Job



Select the Tune a model task in Gen AI Studio



Tune a model

Tune a model so it's better equipped for your use case, then deploy to an endpoint to get predictions or test it in prompt design. Take a tutorial on [creating a tuned model](#).

+ CREATE TUNED MODEL

Specify the location of the data and the job parameters

The screenshot shows the Vertex AI Model Tuning interface. It consists of two main sections: 'Tuning dataset' (Step 1) and 'Model details' (Step 2).

Tuning dataset (Step 1):

- START TUNING:** A button at the top left.
- 1 Tuning dataset:** A step indicator.
- Description:** Model tuning creates and deploys a new model from an existing one that's better adapted to your use case. Currently, model tuning occurs in limited regions. If you have an organization policy restricting certain regions, model tuning may fail.
- Tuning dataset:** A section describing what a tuning dataset is and how to upload it.
- GCS file path ***: A text input field containing `gs://vertex-ai-dar-working-dir/custom-training-data`.
- CONTINUE**: A button at the bottom right of the first section.

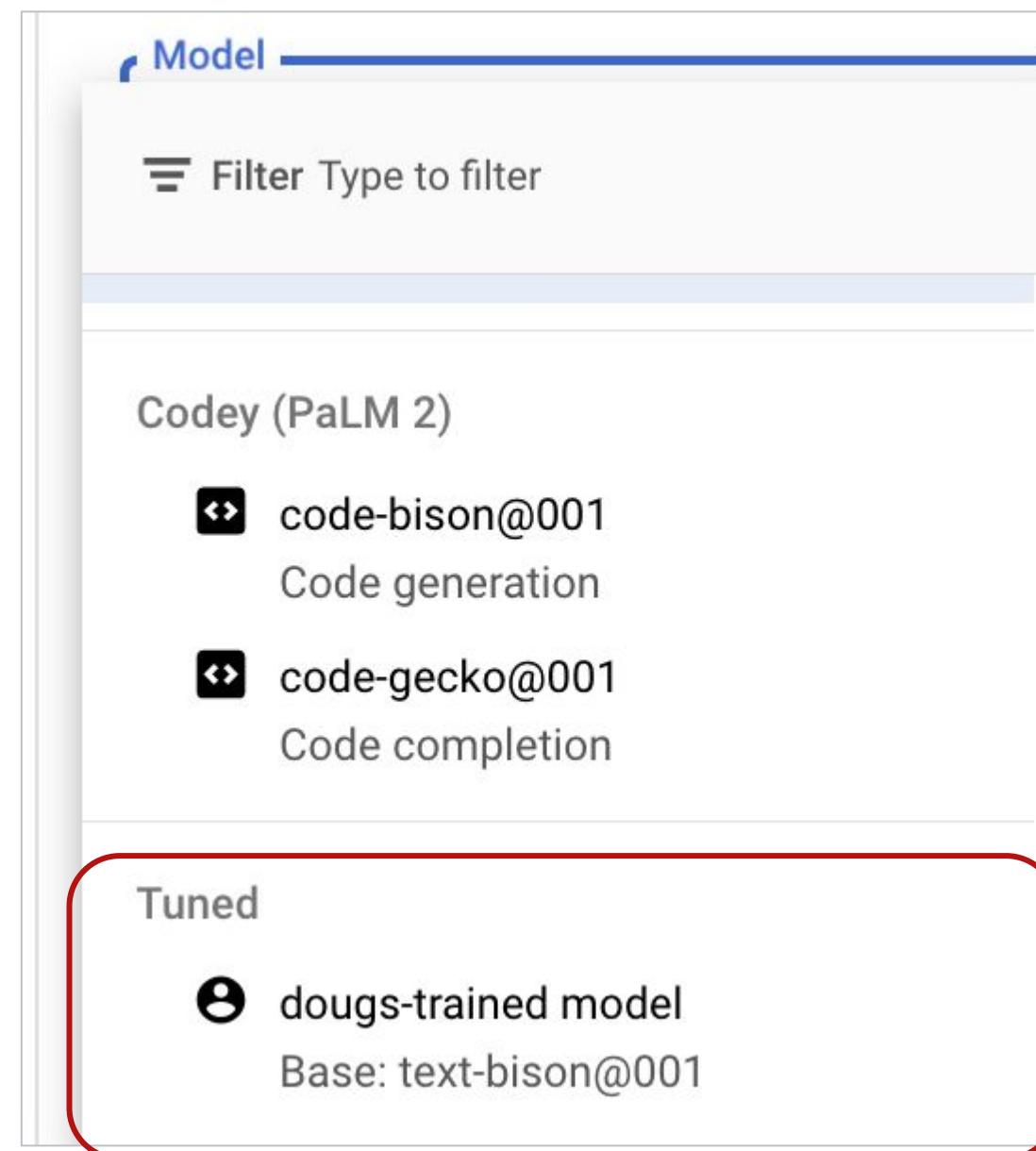
Model details (Step 2):

- 2 Model details:** A step indicator.
- START TUNING**: A large blue button at the top center.
- Tuning dataset:** A section indicating the tuning dataset is selected.
- Model name ***: A text input field containing `my-custom-model`.
Description: The name of the new model. Up to 128 characters.
- Settings:** A heading for configuration options.
- Base model**: A dropdown menu currently set to `text-bison@001`.
Description: The base model that will be used to create a new tuned model.
- Train steps ***: A text input field containing `100`.
Description: The number of training steps for the tuned model.
- Learning Rate ***: A text input field containing `3`.
Description: The learning rate for the tuned model.
- Working directory ***: A text input field containing `gs://vertex-ai-dar-working-dir`.
Description: The Cloud Storage location where the artifacts are stored during the pipeline tuning run.
- BROWSE**: A button next to the Working directory input field.

The number of training examples and training steps needed depends on the task

Task	Suggested # of examples	Training steps
Classification	100+	100-500
Summarization	100-500+	200-1000
Extractive QA	100+	100-500

Tuned models are available from Gen AI Studio



Gen AI Studio will generate the code for using tuned models

View code

PYTHON PYTHON COLAB CURL

Use this script to request a model response in your application.

1. Set up the [Vertex AI SDK for Python](#)
2. Use the following code in your application to request a model response

```
import vertexai
from vertexai.preview.language_models import TextGenerationModel

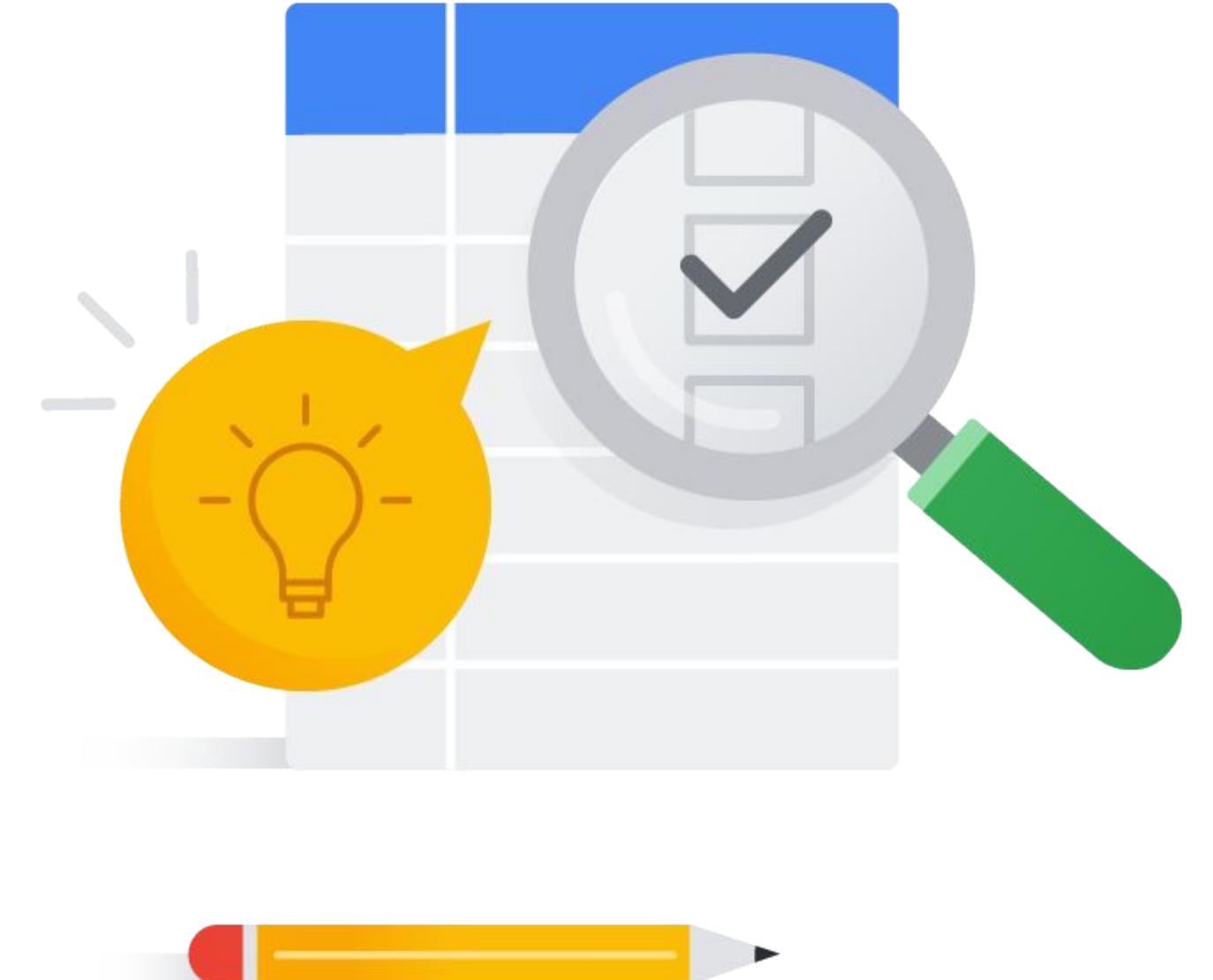
vertexai.init(project="982785856251", location="us-central1")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}

model = TextGenerationModel.from_pretrained("text-bison@001")
model = model.get_tuned_model("projects/982785856251/locations/us-central1/models/167990643268360"
response = model.predict(
    "What is the capital of France?",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Demo

⌚ 15 min ⚙️

Fine-Tuning Models for Specific Use Cases



In this module, you learned to ...

- 01 Evaluate scenarios for creating tuned models
- 02 Build workflows for tuning and deploying models
- 03 Use tuned models in your applications





Summary

In this course, you learned ...

- 01 The Vertex AI generative AI options for your applications
- 02 To use Gen AI Studio to interact with foundation models
- 03 How to design and tune prompts for your Generative AI use cases
- 04 How to Implement the PaLM API into your applications using the Python SDK
- 05 To fine-tune foundation model weights to improve model output quality



Take a structured approach to GenAI transformation



Ramp your skills

Continue your learning journey and complete our [Generative AI Learning Path](#), free of charge, on Google Cloud Skills Boost



Organizational readiness

Assess your organization's current status and business needs for generative & traditional AI capabilities



Identify use cases

Select from one of our [Jumpstart GenAI offers](#), and work with Google Cloud to develop a technical design doc and sample code to solve the use case



Test and scale

Purchase and implement generative AI solutions. Not all AI is built equal. POC often and fail fast to identify what works for your business.

[Contact your Google Cloud Representative to learn more](#)

4 GenAI Jumpstart offers - \$25k & 2 weeks per use case



CREATE

Bring your thoughts and visions to life

Use cases:

- Images from text
- Product descriptions from images
- Blog post from content*
- Email from content*
- Release notes from content*
- Report from content*
- Press releases from content*
- Personalized ads*



SUMMARIZE

Condense and summarize your knowledge base into a simple format

Use cases:

- Content/video summarization
- Intra-knowledge Q&A
- Explanations of code content*
- External chatbot using internal data*
- External chatbot using website data*



DISCOVER

Help your customers and employees find what they need at the right time

Use cases:

- Search for a document
- Machine-generated event monitoring
- File organization based on content*
- Exam questions from content*



AUTOMATE

Automate your customer service across multiple channels

Use cases:

- Contract information extraction
- Feedback classification and ticket creation
- Sentiment analysis*
- Content translation*
- Structured data extraction from file*
- Media tagging*
- Product tagging*
- Content moderation *

*may require Responsible AI Review

