Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code and audio.

How to get started

- Go to Google AI Studio and log in with your Google Account.
- Create an API key.
- Use a quickstart for Python or call the REST API using curl.

Discover Gemini's advanced capabilities

- Play with Gemini multimodal outputs, mixing text and images in an iterative way.
- Discover the multimodal Live API (demo here).
- Learn how to analyse images and detect items in your pictures using Gemini (bonus, there's a 3D version as well!).
- Unlock the power of the Gemini thinking model, capable of solving complex tasks with its inner thoughts.

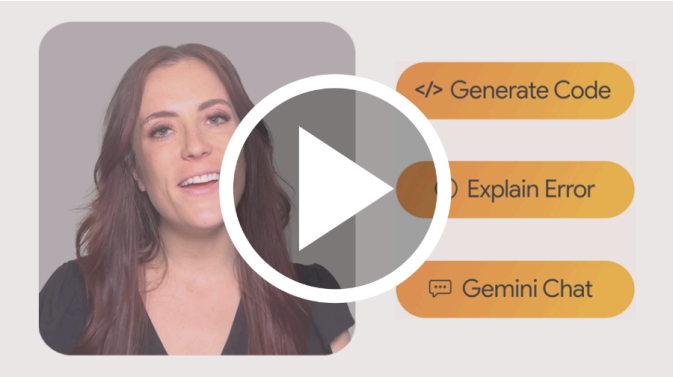
Explore complex use cases

- Use Gemini grounding capabilities to create a report on a company based on what the model can find on the Internet.
- Extract invoices and form data from PDFs in a structured way.
- Create illustrations based on a whole book using Gemini large context window and Imagen.

To learn more, take a look at the Gemini cookbook or visit the Gemini API documentation.

Double-click (or enter) to edit

Colab now has AI features powered by Gemini. The video below provides information on how to use these features, whether you're new to Python or a seasoned veteran.



What is Colab?

Colab, or 'Colaboratory', allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab or Colab features you may have missed to learn more or just get started below!

Getting started

The document that you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut 'Command/Ctrl+Enter'. To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

604800

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To find out more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: Create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To find out more about the Jupyter project, see jupyter.org.

Data science

With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualise it. To edit the code, just click the cell and start editing.

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from GitHub and many other sources. To find out more about importing data, and how Colab can be used for data science, see the links below under Working with data.

```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```

Colab notebooks execute code on Google's cloud servers, meaning that you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

For example, if you find yourself waiting for **pandas** code to finish running and want to go faster, you can switch to a GPU runtime and use libraries like RAPIDS cuDF that provide zero-code-change acceleration.

To learn more about accelerating pandas on Colab, see the 10-minute guide or US stock market data analysis demo.

Machine learning

With Colab you can import an image dataset, train an image classifier on it and evaluate the model, all in just a few lines of code.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

More resources

Working with notebooks in Colab

- Overview of Colab
- Guide to markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets

Working with data

- Loading data: Drive, Sheets and Google Cloud Storage
- Charts: visualising data
- Getting started with BigQuery

Machine learning

These are a few of the notebooks related to machine learning, including Google's online machine learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Intro to RAPIDS cuDF to accelerate pandas
- Getting started with cuML's accelerator mode
- Linear regression with tf.keras using synthetic data

Using accelerated hardware

- TensorFlow with GPUs
- TPUs in Colab

Featured examples

- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB film reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine-learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

Double-click (or enter) to edit

import gradio as gr import torch from transformers import AutoTokenizer, AutoModelForCausalLM

Load model and tokenizer

model_name = "ibm-granite/granite-3.2-2b-instruct" tokenizer = AutoTokenizer.from_pretrained(model_name) model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32, device_map="auto" if torch.cuda.is_available() else None)

if tokenizer.pad_token is None: tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512): inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

```
if torch.cuda.is_available():
    inputs = {k: v.to(model.device) for k, v in inputs.items()}

with torch.no_grad():
    outputs = model.generate(
        **inputs,
        max_length=max_length,
        temperature=0.7,
        do_sample=True,
        pad_token_id=tokenizer.eos_token_id
    )

response = tokenizer.decode(outputs[0], skip_special_tokens=True)
response = response.replace(prompt, "").strip()
return response
```

def concept_explanation(concept): prompt = f"Explain the concept of {concept} in detail with examples:" return generate_response(prompt, max_length=800)

def quiz_generator(concept): prompt = f"Generate 5 quiz questions about {concept} with different question types (multiple choice, true/false, short answer). At the end, provide all the answers in a separate ANSWERS section:" return generate_response(prompt, max_length=1000)

Create Gradio interface

with gr.Blocks() as app: gr.Markdown("# Educational AI Assistant")

```
with gr.Tabs():
    with gr.TabItem("Concept Explanation"):
        concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machi
        explain_btn = gr.Button("Explain")
        explanation_output = gr.Textbox(label="Explanation", lines=10)

        explain_btn.click(concept_explanation, inputs=concept_input, outputs=explana

    with gr.TabItem("Quiz Generator"):
        quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
        quiz_btn = gr.Button("Generate Quiz")
        quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

        quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)
```

app.launch(share=True)

```
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
    device_map="auto" if torch.cuda.is_available() else None
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

def generate_response(prompt, max_length=512):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)

    if torch.cuda.is_available():
        inputs = {k: v.to(model.device) for k, v in inputs.items()}

    with torch.no_grad():
        outputs = model.generate(
            **inputs,
            max_length=max_length,
            temperature=0.7,
            do_sample=True,
            pad_token_id=tokenizer.eos_token_id
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    response = response.replace(prompt, "").strip()
    return response
```

```
def concept_explanation(concept):
    prompt = f"Explain the concept of {concept} in detail with examples:"
    return generate_response(prompt, max_length=800)

def quiz_generator(concept):
    prompt = f"Generate 5 quiz questions about {concept} with different question types (mult
    return generate_response(prompt, max_length=1000)

# Create Gradio interface
with gr.Blocks() as app:
    gr.Markdown("# Educational AI Assistant")

    with gr.Tabs():
        with gr.TabItem("Concept Explanation"):
            concept_input = gr.Textbox(label="Enter a concept", placeholder="e.g., machine l
            explain_btn = gr.Button("Explain")
            explanation_output = gr.Textbox(label="Explanation", lines=10)

            explain_btn.click(concept_explanation, inputs=concept_input, outputs=explanatior

        with gr.TabItem("Quiz Generator"):
            quiz_input = gr.Textbox(label="Enter a topic", placeholder="e.g., physics")
            quiz_btn = gr.Button("Generate Quiz")
            quiz_output = gr.Textbox(label="Quiz Questions", lines=15)

            quiz_btn.click(quiz_generator, inputs=quiz_input, outputs=quiz_output)

app.launch(share=True)
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserW
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access publ
  warnings.warn(
tokenizer_config.json:        8.88k/?  [00:00<00:00,  610kB/s]
vocab.json:          777k/?  [00:00<00:00,  12.9MB/s]
merges.txt:          442k/?  [00:00<00:00,  20.3MB/s]
tokenizer.json:         3.48M/?  [00:00<00:00,  63.7MB/s]
added_tokens.json:  100%                                                87.0/87.0  [00:00<00:00,  6.15kB/s]
special_tokens_map.json:  100%                                          701/701  [00:00<00:00,  49.1kB/s]
config.json:  100%                                                      786/786  [00:00<00:00,  49.8kB/s]
`torch_dtype` is deprecated! Use `dtype` instead!
model.safetensors.index.json:        29.8k/?  [00:00<00:00,  2.00MB/s]
Fetching  2  files:  100%                                               2/2  [01:28<00:00,  88.75s/it]
model-00002-of-                                                         67.1M/67.1M  [00:00<00:00,  30.6MB/s]
00002.safetensors:  100%
model-00001-of-                                                         5.00G/5.00G  [01:28<00:00,  104MB/s]
00002.safetensors:  100%
Loading  checkpoint  shards:  100%                                      2/2  [00:19<00:00,   8.01s/it]
generation_config.json:  100%                                           137/137  [00:00<00:00,  6.27kB/s]
Colab notebook detected. To show errors in colab notebook, set debug=True in lau
* Running on public URL: https://9d6ca0fb57e4bd9283.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
```