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

1. Go to [Google AI Studio](#) and log in with your Google account.
2. [Create an API key](#).
3. Use a quickstart for [Python](#), or call the REST API using [curl](#).

Explore use cases

- [Create a marketing campaign](#)
- [Analyze audio recordings](#)
- [Use System instructions in chat](#)

To learn more, check out the [Gemini cookbook](#) or visit the [Gemini API documentation](#).

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.


```
Start coding or generate with AI.
```

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](#) to learn more, or just get started below!

⌄ Getting started

The document 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:

```python
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:

```python
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 learn 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 learn more about the Jupyter project, see [jupyter.org](#).

⌄ Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize 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 learn more about importing data, and how Colab can be used for data science, see the links below under Working with Data.

```python
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"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"""![{alt}]({image})"""))
plt.close(fig)
```
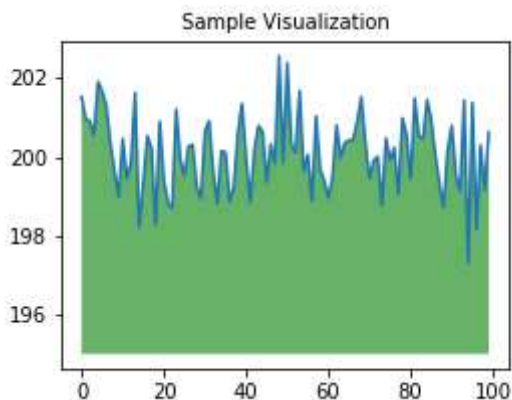

Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

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: visualizing data](#)
- [Getting started with BigQuery](#)

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the [full course website](#) for more.

- [Intro to Pandas DataFrame](#)
- [Intro to RAPIDS cuDF to accelerate pandas](#)
- [Linear regression with tf.keras using synthetic data](#)

Using Accelerated Hardware

- [TensorFlow with GPUs](#)
- [TensorFlow with TPUs](#)

⌄ 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 movie 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.

```python
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Convert to a DataFrame for easier manipulation
iris_df = pd.DataFrame(X, columns=iris.feature_names)
iris_df['species'] = y

# Convert the problem to binary classification: Setosa vs. Non-Setosa
iris_df['binary_species'] = iris_df['species'].apply(lambda x: 0 if x == 0 else 1)
X = iris_df.drop(['species', 'binary_species'], axis=1)
```

```python
y = iris_df['binary_species']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

# Standardize the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Perceptron algorithm implementation
class Perceptron:
    def __init__(self, learning_rate=0.01, n_iter=50):
        self.learning_rate = learning_rate
        self.n_iter = n_iter

    def fit(self, X, y):
        self.weights = np.zeros(X.shape[1] + 1)
        self.errors = []

        for _ in range(self.n_iter):
            error = 0
            for xi, target in zip(X, y):
                update = self.learning_rate * (target - self.predict(xi))
                self.weights[1:] += update * xi
                self.weights[0] += update
                error += int(update != 0.0)
            self.errors.append(error)
        return self

    def net_input(self, X):
        return np.dot(X, self.weights[1:]) + self.weights[0]

    def predict(self, X):
        return np.where(self.net_input(X) >= 0.0, 1, 0)

# Create a Perceptron instance
perceptron = Perceptron(learning_rate=0.01, n_iter=50)

# Train the perceptron
perceptron.fit(X_train, y_train)

# Print the learned weights
print("Learned weights:", perceptron.weights)

# Make predictions on the test set
y_pred = perceptron.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nConfusion Matrix:")
conf_matrix = confusion_matrix(y_test, y_pred)
print(conf_matrix)

# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=['Non-Setos
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

⮕ Learned weights: [0.01   0.01329692 -0.01926598 0.01990417 0.01624068]
   Accuracy: 1.0

   Classification Report:
                 precision   recall f1-score   support

              0   1.00   1.00   1.00   10
              1   1.00   1.00   1.00   20

       accuracy   1.00   30
      macro avg   1.00   1.00   1.00   30
   weighted avg   1.00   1.00   1.00   30

   Confusion Matrix:
   [[10 0]
    [0 20]]

Confusion Matrix