Welcome to Colab!

(New) Try the Gemini API

- Generate a Gemini API key
- Talk to Gemini with the Speech-to-Text API
- Gemini API: Quickstart with Python
- Gemini API code sample
- Compare Gemini with ChatGPT
- More notebooks

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.

What is Colab?

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

pdated_file.csv •••

1 to 5 of 5 entries Filter

ID	Name	Description
1	Kelly	Avid reader of English literature and History, enjoys Geography field trips
2	Olivia	Excels in Math competitions, passionate about Science experiments
3	Noah	Enjoys analyzing historical events, participates in Geography quizzes
4	Liam	Sports enthusiast, particularly interested in Sports Science
5	Maya	Balances Math problem-solving with English poetry appreciation

Show 10 per page

- 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:

```
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 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

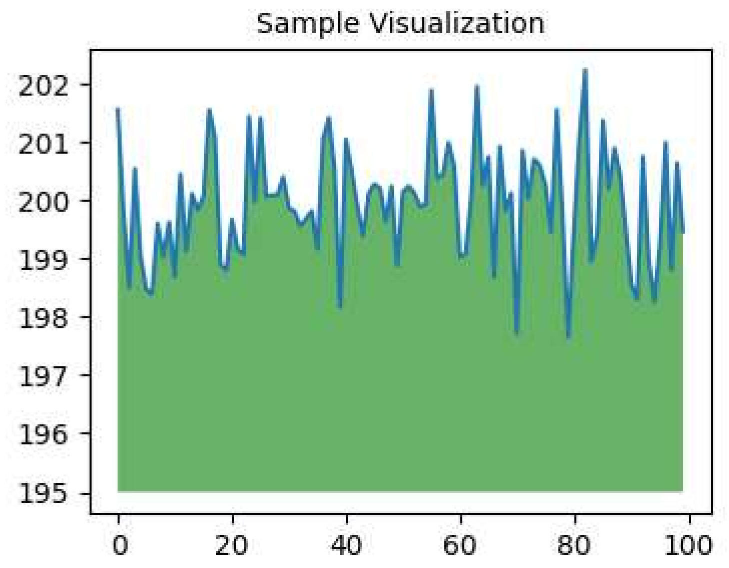
With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

```
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)
```



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.

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 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.

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 Colaboratory
- 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
- Linear regression with tf.keras using synthetic data

Using Accelerated Hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

Featured examples

- NeMo Voice Swap: Use Nvidia's NeMo conversational AI Toolkit to swap a voice in an audio fragment with a computer generated one.
- 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.

```
!pip install nltk
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
True
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

def generate_tags(text):
    # Tokenize the text
    tokens = word_tokenize(text)
    # Perform Part-of-Speech tagging
    tagged = pos_tag(tokens)

    # Extract nouns and proper nouns as tags
    tags = [word for word, pos in tagged if pos in ['NN', 'NNS', 'NNP', 'NNPS']]

    return tags

# Example text content
text_content = "I am curious if there is an algorithm/method to generate keywords/tags from a give

# Generate tags from the text content
tags = generate_tags(text_content)

print("Generated Tags:", tags)
```

```
Generated Tags: ['algorithm/method', 'keywords/tags', 'text']
```

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

def update_database(student_name, subjects):
    # Implement your database update logic here
    # Insert records into the Student_Subject junction table
    pass

# Sample student description
description = "She likes Math and Science"

# Tokenize the description
tokens = word_tokenize(description)

# Perform Part-of-Speech tagging
tagged = pos_tag(tokens)

# Extract subjects based on keywords
subjects = [word for word, pos in tagged if pos == 'NNP' and word in ['Math', 'Science']]
```

```
    # Update the database with the extracted subjects for the student
    if subjects:
        # Assuming you have a function to update the database with subjects
        update_database(student_name='Gauri', subjects=subjects)
        print(f"Subjects {subjects} tagged for student Gauri successfully.")
    else:
        print("No relevant subjects found in the description.")
```

```
Subjects ['Math', 'Science'] tagged for student Gauri successfully.
```

```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

# Function to extract tags from description
def extract_tags(description):
    tokens = word_tokenize(description)
    tagged = pos_tag(tokens)
    tags = [word for word, pos in tagged if pos == 'NNP' and word in ['Math',
                                                                       'English',
                                                                       'Science',
                                                                       'History',
                                                                       'Geography',
                                                                       'Sports']]
    return tags

# Read CSV file
data = pd.read_csv('/content/sample_data/student.csv')

# Update data with extracted tags
data['Tags'] = data['Description'].apply(extract_tags)

# Save updated data back to CSV
data.to_csv('updated_file.csv', index=False)
```

```
import pandas as pd
import nltk
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag

# Function to extract important words based on criteria
def extract_important_words(description):
    tokens = word_tokenize(description)
    tagged = pos_tag(tokens)

    # Define your criteria for selecting important words here
    important_words = [word for word, pos in tagged if pos.startswith('NN')
    or pos.startswith('VB')]  # Example: Nouns and Verbs

    return important_words

# Read CSV file
data = pd.read_csv('/content/sample_data/student.csv')

# Update data with extracted important words
data['Important_Words'] = data['Description'].apply(extract_important_words)

# Save updated data back to CSV
data.to_csv('updated_file-2.csv', index=False)
```