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.


```
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 find out 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.

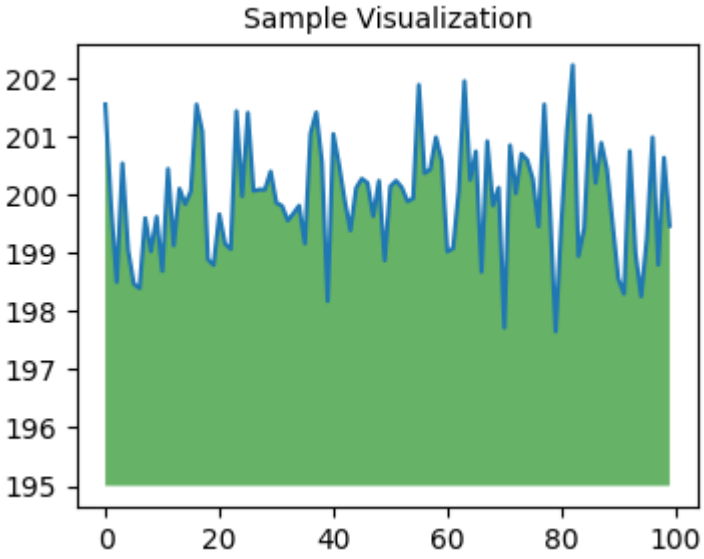
```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

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)
```



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.

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: visualising 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 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 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.

```
import pandas as pd
import numpy as np

df=pd.read_excel("Demo Data DSBDA.xlsx")
print(df)
```

```
          Sr. no.                    Roll No        Name of the Student   DM  \
0               1                     SCOA01              ALISHA SHEIKH   24
1               2                     SCOA02  SOHAM JAYYANNTH DESHMUKKH   16
2               3                     SCOA03         SHELAR NEHA NILESH   20
3               4                     SCOA04                RIYA JAWALE   18
4               5                     SCOA05      NARHE VARSHA RAOSAHEB   20
..            ...                        ...                         ...  ...
77  Maximum marks                        NaN                         NaN   24
78            NaN                        NaN                         NaN  NaN
79            NaN                        NaN                         NaN  NaN
80            NaN                        NaN                         NaN  NaN
81            NaN  Internal Exam Coordinator                         NaN  NaN

    FDS  OOP   CG DELD GENDER  Unnamed: 9  ...  Unnamed: 14  Unnamed: 15  \
0    27   20   15   16      F         NaN  ...          NaN          NaN
1     9    9   AB   AB      M         NaN  ...          NaN          NaN
2    23   16   15   15      F         NaN  ...          NaN          NaN
3    14   11    8   15      F         NaN  ...          NaN          NaN
4    24   16   24   20      M         NaN  ...          NaN          NaN
..  ...  ...  ...  ...    ...         ...  ...          ...          ...
77   27   25   29   27    NaN         NaN  ...          NaN          NaN
78  NaN  NaN  NaN  NaN    NaN         NaN  ...          NaN          NaN
79  NaN  NaN  NaN  NaN    NaN         NaN  ...          NaN          NaN
80  NaN  NaN  NaN  NaN    NaN         NaN  ...          NaN          NaN
81  NaN  NaN  HOD  NaN    NaN         NaN  ...          NaN          NaN

    Unnamed: 16  Unnamed: 17  Unnamed: 18  Unnamed: 19  Unnamed: 20  \
0           NaN          NaN          NaN          NaN          NaN
1           NaN          NaN          NaN          NaN          NaN
2           NaN          NaN          NaN          NaN          NaN
3           NaN          NaN          NaN          NaN          NaN
4           NaN          NaN          NaN          NaN          NaN
..          ...          ...          ...          ...          ...
77          NaN          NaN          NaN          NaN          NaN
78          NaN          NaN          NaN          NaN          NaN
79          NaN          NaN          NaN          NaN          NaN
80          NaN          NaN          NaN          NaN          NaN
81          NaN          NaN          NaN          NaN          NaN

    Unnamed: 21  Unnamed: 22  Unnamed: 23
0           NaN          NaN          NaN
1           NaN          NaN          NaN
2           NaN          NaN          NaN
3           NaN          NaN          NaN
4           NaN          NaN          NaN
..          ...          ...          ...
77          NaN          NaN          NaN
78          NaN          NaN         321`
79          NaN          NaN            `
80          NaN          NaN          NaN
```

```
81          NaN          NaN          NaN

[82 rows x 24 columns]
```

```
import pandas as pd

# Load the dataset (replace the path with the correct location if downloading)
url = 'https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv'  # Ex
df = pd.read_csv(url)

# Display the first few rows of the dataset
print(df.head())

print(df.isnull().sum())

print(df.describe())

print(df.shape)

print(df.dtypes)
```

```
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
(150, 5)
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
species          object
```

```
      dtype: object
```

```
import pandas as pd

url = 'https://raw.githubusercontent.com/uiuc-cse/data-fa14/gh-pages/data/iris.csv'
df = pd.read_csv(url)

print(df.columns)

df['species'] = df['species'].astype('category')

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']] = scaler.fit_transform
    df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']])


from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['species'] = label_encoder.fit_transform(df['species'])

df = pd.get_dummies(df, columns=['species'], drop_first=True)
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
       'species'],
      dtype='object')
```

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, LabelEncoder

# CSV URL for the Iris dataset
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Column names for the dataset
column_names = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Specie

# Read the CSV file into a pandas DataFrame
iris = pd.read_csv(csv_url, header=None, names=column_names)

# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(iris.head())

# Check for missing values in each column
print("\nMissing Values per Column:")
print(iris.isnull().sum())

# Get summary statistics (like mean, std, min, max, etc.)
print("\nSummary Statistics:")
```

```
print(iris.describe())

# Check the dimensions of the data (rows, columns)
print("\nShape of the dataset (rows, columns):", iris.shape)

# Check the data types of each column
print("\nData Types of Columns:")
print(iris.dtypes)

# Normalize the numeric columns (Sepal and Petal measurements)
scaler = MinMaxScaler()
iris[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']] = scaler.fit_tra
    iris[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']])

# Display the first few rows of the normalized data
print("\nNormalized Data:")
print(iris.head())

# Encode the 'Species' column using LabelEncoder
label_encoder = LabelEncoder()
iris['Species'] = label_encoder.fit_transform(iris['Species'])

# Display the first few rows after encoding
print("\nData After Label Encoding 'Species':")
print(iris.head())
```

```
First few rows of the dataset:
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0            5.1           3.5            1.4           0.2  Iris-setosa
1            4.9           3.0            1.4           0.2  Iris-setosa
2            4.7           3.2            1.3           0.2  Iris-setosa
3            4.6           3.1            1.5           0.2  Iris-setosa
4            5.0           3.6            1.4           0.2  Iris-setosa

Missing Values per Column:
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64

Summary Statistics:
       SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
count     150.000000    150.000000     150.000000    150.000000
mean        5.843333      3.054000       3.758667      1.198667
std         0.828066      0.433594       1.764420      0.763161
min         4.300000      2.000000       1.000000      0.100000
25%         5.100000      2.800000       1.600000      0.300000
50%         5.800000      3.000000       4.350000      1.300000
75%         6.400000      3.300000       5.100000      1.800000
max         7.900000      4.400000       6.900000      2.500000

Shape of the dataset (rows, columns): (150, 5)

Data Types of Columns:
```

```
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object

Normalized Data:
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0       0.222222      0.625000       0.067797      0.041667  Iris-setosa
1       0.166667      0.416667       0.067797      0.041667  Iris-setosa
2       0.111111      0.500000       0.050847      0.041667  Iris-setosa
3       0.083333      0.458333       0.084746      0.041667  Iris-setosa
4       0.194444      0.666667       0.067797      0.041667  Iris-setosa

Data After Label Encoding 'Species':
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
0       0.222222      0.625000       0.067797      0.041667        0
1       0.166667      0.416667       0.067797      0.041667        0
2       0.111111      0.500000       0.050847      0.041667        0
3       0.083333      0.458333       0.084746      0.041667        0
4       0.194444      0.666667       0.067797      0.041667        0
```

```
import pandas as pd

# Define column names if they are not present in the CSV
col_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

# Load the dataset from a URL
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris = pd.read_csv(csv_url, names=col_names)

# 1. Display the first 5 rows
print("First 5 rows:")
print(iris.head())

# 2. Display the last 5 rows
print("\nLast 5 rows:")
print(iris.tail())

# 3. Get the index (row labels) of the dataset
print("\nDataset index:")
print(iris.index)

# 4. Get the column labels of the dataset
print("\nDataset columns:")
print(iris.columns)

# 5. Get the shape of the dataset (rows, columns)
print("\nDataset shape (rows, columns):")
print(iris.shape)

# 6. Get the data types of each column
print("\nData types of each column:")
print(iris.dtypes)
```

```
# 7. Get the column names as an array
print("\nColumn names as array:")
print(iris.columns.values)

# 8. Generate descriptive statistics for the dataset
print("\nDescriptive statistics:")
print(iris.describe(include='all'))

# 9. Access a specific column by name
print("\nAccess 'sepal_length' column:")
print(iris['sepal_length'].head())

# 10. Sort columns by label in descending order
print("\nSort columns by label (descending):")
print(iris.sort_index(axis=1, ascending=False).head())

# 11. Sort rows by a specific column (e.g., 'sepal_length')
print("\nSort rows by 'sepal_length' column:")
print(iris.sort_values(by="sepal_length").head())

# 12. Use iloc to select the 5th row
print("\n5th row using iloc:")
print(iris.iloc[5])

# 13. Slice the first 3 rows
print("\nFirst 3 rows:")
print(iris[0:3])

# 14. Select specific columns using loc
print("\nSelect 'sepal_length' and 'petal_length' columns using loc:")
print(iris.loc[:, ["sepal_length", "petal_length"]].head())

# 15. Select the first 5 rows using iloc
print("\nFirst 5 rows using iloc:")
print(iris.iloc[:5, :])

# 16. Select the first 3 columns using iloc
print("\nFirst 3 columns using iloc:")
print(iris.iloc[:, :3].head())

# 17. Select a subset of the first 3 rows and 2 columns using iloc
print("\nFirst 3 rows and 2 columns using iloc:")
print(iris.iloc[:3, :2])
```

```
First 5 rows:
   sepal_length  sepal_width  petal_length  petal_width        class
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Last 5 rows:
     sepal_length  sepal_width  petal_length  petal_width           class
145           6.7          3.0           5.2          2.3  Iris-virginica
```

```
146           6.3          2.5           5.0          1.9  Iris-virginica
147           6.5          3.0           5.2          2.0  Iris-virginica
148           6.2          3.4           5.4          2.3  Iris-virginica
149           5.9          3.0           5.1          1.8  Iris-virginica

Dataset index:
RangeIndex(start=0, stop=150, step=1)

Dataset columns:
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class'], dty

Dataset shape (rows, columns):
(150, 5)

Data types of each column:
sepal_length    float64
sepal_width     float64
petal_length    float64
petal_width     float64
class            object
dtype: object

Column names as array:
['sepal_length' 'sepal_width' 'petal_length' 'petal_width' 'class']

Descriptive statistics:
        sepal_length  sepal_width  petal_length  petal_width        class
count     150.000000   150.000000    150.000000   150.000000          150
unique           NaN          NaN           NaN          NaN            3
top              NaN          NaN           NaN          NaN  Iris-setosa
freq             NaN          NaN           NaN          NaN           50
mean        5.843333     3.054000      3.758667     1.198667          NaN
std         0.828066     0.433594      1.764420     0.763161          NaN
min         4.300000     2.000000      1.000000     0.100000          NaN
25%         5.100000     2.800000      1.600000     0.300000          NaN
50%         5.800000     3.000000      4.350000     1.300000          NaN
75%         6.400000     3.300000      5.100000     1.800000          NaN
max         7.900000     4.400000      6.900000     2.500000          NaN

Access 'sepal_length' column:
0    5.1
1    4.9
2    4.7
3    4.6
4    5.0
Name: sepal_length, dtype: float64
```

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, LabelEncoder

# URL for the Iris dataset (without column names)
csv_url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

# Custom column names (as you mentioned)
col_names = ['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width', 'Species']

# Read the CSV file into a pandas DataFrame, assigning column names manually
iris = pd.read_csv(csv_url, header=None, names=col_names)
```

```
# Display the first few rows of the dataset
print("First few rows of the dataset:")
print(iris.head())

# Check for missing values in each column
print("\nMissing Values per Column:")
print(iris.isnull().sum())

# Get summary statistics (like mean, std, min, max, etc.)
print("\nSummary Statistics:")
print(iris.describe())

# Check the dimensions of the data (rows, columns)
print("\nShape of the dataset (rows, columns):", iris.shape)

# Check the data types of each column
print("\nData Types of Columns:")
print(iris.dtypes)

# Normalize the numeric columns (Sepal and Petal measurements)
scaler = MinMaxScaler()
iris[['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']] = scaler.fit_transfo
    iris[['Sepal_Length', 'Sepal_Width', 'Petal_Length', 'Petal_Width']])

# Display the first few rows of the normalized data
print("\nNormalized Data:")
print(iris.head())

# Encode the 'Species' column using LabelEncoder
label_encoder = LabelEncoder()
iris['Species'] = label_encoder.fit_transform(iris['Species'])

# Display the first few rows after encoding
print("\nData After Label Encoding 'Species':")
print(iris.head())
```

```
First few rows of the dataset:
   Sepal_Length  Sepal_Width  Petal_Length  Petal_Width      Species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Missing Values per Column:
Sepal_Length    0
Sepal_Width     0
Petal_Length    0
Petal_Width     0
Species         0
dtype: int64

Summary Statistics:
       Sepal_Length  Sepal_Width  Petal_Length  Petal_Width
```

```
       count    150.000000   150.000000    150.000000   150.000000
       mean       5.843333     3.054000      3.758667     1.198667
       std        0.828066     0.433594      1.764420     0.763161
       min        4.300000     2.000000      1.000000     0.100000
       25%        5.100000     2.800000      1.600000     0.300000
       50%        5.800000     3.000000      4.350000     1.300000
       75%        6.400000     3.300000      5.100000     1.800000
       max        7.900000     4.400000      6.900000     2.500000

       Shape of the dataset (rows, columns): (150, 5)

       Data Types of Columns:
       Sepal_Length    float64
       Sepal_Width     float64
       Petal_Length    float64
       Petal_Width     float64
       Species          object
       dtype: object

       Normalized Data:
          Sepal_Length  Sepal_Width  Petal_Length  Petal_Width      Species
       0      0.222222     0.625000      0.067797     0.041667  Iris-setosa
       1      0.166667     0.416667      0.067797     0.041667  Iris-setosa
       2      0.111111     0.500000      0.050847     0.041667  Iris-setosa
       3      0.083333     0.458333      0.084746     0.041667  Iris-setosa
       4      0.194444     0.666667      0.067797     0.041667  Iris-setosa

       Data After Label Encoding 'Species':
          Sepal_Length  Sepal_Width  Petal_Length  Petal_Width  Species
       0      0.222222     0.625000      0.067797     0.041667        0
       1      0.166667     0.416667      0.067797     0.041667        0
       2      0.111111     0.500000      0.050847     0.041667        0
       3      0.083333     0.458333      0.084746     0.041667        0
       4      0.194444     0.666667      0.067797     0.041667        0
```