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

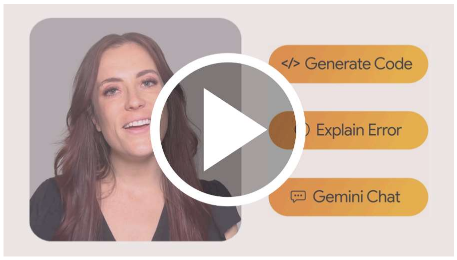
- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To lear book or visit the Gemini API documentation.

Run this cell to mount your Google Drive.
Learn more

Dismiss

Colab nini. The video below provides information on how to use these features, whether you're new to Pythor



```
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 or Colab Features You May Have Missed 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.

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many mporting data, and how Colab can be used for data science, see the links below under [Working with Data]().

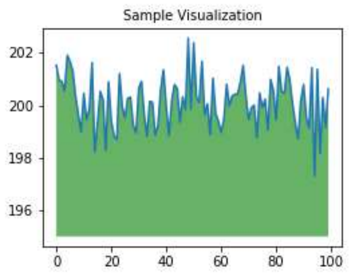
Run this cell to mount your Google Drive.
Learn more

```
import
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
- I
- I

Run this cell to mount your Google Drive.
Learn more

Work

- 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.

Importing the dependencies

```
import numpy as np
import pandas as pd
import difflib
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

+ Code + Text

Data Collection and Pre-Processing

Generate print hello world using rot13 Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
#loading the data from the csv file to apandas dataframe
movies_data = pd.read_csv('/content/movies.csv')
```

```
<ipython-input-2-398bc27a5780>:2: DtypeWarning: Columns (0,1,4,9,13,14,19,20,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43
  movies_data = pd.read_csv('/content/movies.csv')
```

```
#printing the first 5 rows of dataframe
movies_data.head()
```

	index	budget	genres	homepage	id	keywords
0	0	237000000	Action Adventure Fantasy Science Fiction	http://www.avatarmovie.com/	19995	culture clash future space war space colony so...
				ttp://disney.go.com/disneypictures/pirates/	285	ocean drug abuse exotic island east india trad...
2	2	245000000	Action Adventure Crime	http://www.sonypictures.com/movies/spectre/	206647	spy based on novel secret agent sequel mi6
3	3	250000000	Action Crime Drama Thriller	http://www.thedarkknightrises.com/	49026	dc comics crime fighter terrorist secret ident...
4	4	260000000	Action Adventure Science Fiction	http://movies.disney.com/john-carter	49529	based on novel mars medallion space travel pri...

Run this cell to mount your Google Drive.
Learn more

5 rows × 1265 columns

```
#number of rows and colums in the data frame
movies_data.shape
```

```
(4809, 1265)
```

```
#selecting the relevant features for recommendation
selected_features=['genres','keywords','tagline','cast','director']
print(selected_features)
```

```
['genres', 'keywords', 'tagline', 'cast', 'director']
```

Generate a slider using jupyter widgets Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
#replacing the null values with null string
for feature in selected_features:
  movies_data[feature] = movies_data[feature].fillna('')
```

```
#combined all the 5 selected features
combined_features = movies_data['genres']+' '+movies_data['keywords']+' '+movies_data['tagline']+' '+movies_data['cast']+' '+movies_data['di
```

```
print(combined_features)
```

```
0       Action Adventure Fantasy Science Fiction cultu...
1       Adventure Fantasy Action ocean drug abuse exot...
2       Action Adventure Crime spy based on novel secr...
3       Action Crime Drama Thriller dc comics crime fi...
4       Action Adventure Science Fiction based on nove...
                              ...
4804    Action Crime Thriller united states\u2013mexic...
4805    Comedy Romance  A newlywed couple's honeymoon ...
4806    Comedy Drama Romance TV Movie date love at fir...
4807      A New Yorker in Shanghai Daniel Henney Eliza...
4808    Documentary obsession camcorder crush dream gi...
Length: 4809, dtype: object
```

```
#converting the data to feature vectors
vectorizer = TfidfVectorizer()
```

Generate randomly select 5 items from a list Close

Run this cell to mount your Google Drive.
Learn more

```
featur                                         rm(combined_features)
print(
```

```
  (0, 13033)	0.19423432006987434
  (0, 10237)	0.16059231591328046
  (0, 8764)	0.22708109846346383
  (0, 14617)	0.15151416868595385
  (0, 16677)	0.19842983735418923
  (0, 14073)	0.205956457941733
  (0, 13328)	0.2177400340451317
  (0, 17299)	0.2019755488126334
  (0, 17016)	0.23642216288179602
  (0, 13358)	0.15022036822291404
  (0, 11511)	0.27209420901160897
  (0, 11200)	0.09051396620449952
  (0, 17007)	0.1282251636064218
  (0, 15270)	0.0709833692900418
  (0, 4952)	0.24024658931920426
  (0, 14280)	0.21391560760772346
  (0, 3231)	0.24958765373753639
  (0, 16596)	0.12550744845505796
  (0, 14387)	0.3396344162602317
  (0, 5843)	0.16467965954169428
  (0, 3071)	0.22207581114012026
  (0, 3685)	0.21391560760772346
  (0, 5444)	0.10365929555963475
  :	:
  (4807, 17275)	0.28866712901660546
  (4807, 4842)	0.2471929040013495
  (4807, 410)	0.17732764087392294
  (4807, 6942)	0.28866712901660546
  (4807, 11671)	0.21562869602539067
  (4807, 1678)	0.15653010034102868
  (4807, 10937)	0.13509136428761018
  (4807, 7482)	0.11312822232180882
  (4807, 3803)	0.33371131380910507
  (4808, 7003)	0.5699803352373706
  (4808, 5374)	0.22968913544507036
  (4808, 3661)	0.26250487802797257
  (4808, 2431)	0.24001958843725985
  (4808, 4615)	0.24001958843725985
  (4808, 6424)	0.21753429884654715
  (4808, 4378)	0.15383594649941473
  (4808, 12998)	0.16965675358995316
  (4808, 1322)	0.19607891848110867
  (4808, 4535)	0.19504900925583443
  (4808, 3442)	0.21753429884654715
  (4808, 6162)	0.1805717163739769
  (4808, 4987)	0.1607912774644998
  (4808, 2135)	0.3099892404640069
  (4808, 4525)	0.16785410008320537
  (4808, 11169)	0.1786815070359777
```

```
#getting the similarity scores using cosine similarity
similarity = cosine_similarity(feature_vectors)
print(similarity)
```

```
[[1.         0.0722257  0.03774545 ... 0.         0.         0.        ]
 [0.0722257  1.         0.03282986 ... 0.03576038 0.         0.        ]
 [0.03774545 0.03282986 1.         ... 0.         0.05370221 0.        ]
 ...
 [0.         0.03576038 0.         ... 1.         0.         0.02651822]
 [0.         0.         0.05370221 ... 0.         1.         0.        ]
 [0.         0.         0.         ... 0.02651822 0.         1.        ]]
```

Generate randomly select 5 items from a list Close

```
print(similarity.shape)
```

```
(4809, 4809)
```

```
#getting the movie name for the user
movie_name = input('Enter your favourite movie name : ')
```

```
Enter your favourite movie name : iron man
```

```
#creat                                      es given in the dataset
list_o                                      '].tolist()
print(
```

Run this cell to mount your Google Drive.
Learn more

```
[                                      an: At World's End", 'Spectre', 'The Dark Knight Rises', 'John Carter', 'Spider-Man 3', 'Tangled', 'Av
```

```
#finding the close match for the movie name given by the user
list_of_all_titles = [title for title in list_of_all_titles if isinstance(title, str)]
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)
```

```
['Iron Man', 'Iron Man 3', 'Iron Man 2']
```

```
close_match = find_close_match[0]
print(close_match)
```

```
Iron Man
```

```
#finding the index of the movie with title
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index_of_the_movie)
```

```
68
```

```
#getting a list of similar movies
if isinstance(index_of_the_movie, int) and 0 <= index_of_the_movie < len(similarity):
    similarity_score = list(enumerate(similarity[index_of_the_movie]))
    print(similarity_score)
```

```
len(similarity)
```

```
4809
```

```
#sorting the movies based on their similarity score
sorted_similar_movies = sorted(similarity, key = lambda x:x[1], reverse = True)
print(sorted_similar_movies)
```

```
0.        ]), array([0.06324479, 0.0985745 , 0.01086339, ..., 0.        , 0.        ,
0.        ]), array([0.03900544, 0.09854862, 0.0065355 , ..., 0.        , 0.        ,
0.        ]), array([0.0240888 , 0.09833418, 0.00979612, ..., 0.        , 0.        ,
0.        ]), array([0.01564452, 0.09802993, 0.03043911, ..., 0.00313151, 0.05050143,
0.        ]), array([0.00848916, 0.09758606, 0.        , ..., 0.02276474, 0.        ,
0.        ]), array([0.07896154, 0.09741993, 0.08410794, ..., 0.01355122, 0.01215783,
0.        ]), array([0.00609112, 0.09631903, 0.04032133, ..., 0.00320682, 0.        ,
0.        ]), array([0.05921951, 0.0960857 , 0.0797289 , ..., 0.        , 0.01126444,
0.        ]), array([0.04896336, 0.09586289, 0.04051942, ..., 0.02217151, 0.        ,
0.        ]), array([0.04757257, 0.09572714, 0.02663568, ..., 0.02330793, 0.        ,
0.        ]), array([0.05427205, 0.09517056, 0.02724586, ..., 0.        , 0.        ,
0.        ]), array([0.07059715, 0.09459142, 0.01728617, ..., 0.00268494, 0.0199042 ,
0.        ]), array([0.10185813, 0.09426609, 0.02015778, ..., 0.        , 0.        ,
0.        ]), array([0.02542101, 0.09325186, 0.01910072, ..., 0.09628726, 0.00973477,
0.        ]), array([0.00872757, 0.09323706, 0.        , ..., 0.02327513, 0.        ,
0.        ]), array([0.02900354, 0.09259731, 0.03531083, ..., 0.02995529, 0.01360443,
0.        ]), array([0.04823744, 0.09242056, 0.01307849, ..., 0.03386428, 0.        ,
0.        ]), array([0.06382669, 0.09231794, 0.01428424, ..., 0.00317197, 0.        ,
0.        ]), array([0.03731574, 0.09224299, 0.03506401, ..., 0.        , 0.02207343,
0.        ]), array([0.01418253, 0.09223931, 0.12614152, ..., 0.        , 0.        ,
0.        ]), array([0.01252084, 0.09195268, 0.        , ..., 0.00330122, 0.01744889,
0.        ]), array([0.01795318, 0.09123099, 0.07474867, ..., 0.01428821, 0.04866523,
0.04391885]), array([0.13477979, 0.09096975, 0.01389593, ..., 0.01586446, 0.        ,
0.        ]), array([0.05040069, 0.0905249 , 0.        , ..., 0.        , 0.        ,
0.        ]), array([0.06429208, 0.090318  , 0.00695121, ..., 0.        , 0.        ,
0.0316628 ]), array([0.0516424 , 0.0902417 , 0.00609159, ..., 0.00426136, 0.        ,
                          2556, 0.09022396, 0.022951  , ..., 0.00240028, 0.        ,
                          7669, 0.09007937, 0.09470823, ..., 0.01063189, 0.        ,
                          4015, 0.08968682, 0.01936632, ..., 0.01134183, 0.02500183,
                          1806, 0.08952294, 0.00687501, ..., 0.0356868 , 0.        ,
                          9923, 0.08938545, 0.0138305 , ..., 0.        , 0.        ,
                          7187, 0.08909954, 0.01455673, ..., 0.        , 0.        ,
0.        ]), array([0.01900148, 0.08881416, 0.        , ..., 0.16827517, 0.        ,
0.        ]), array([0.03315676, 0.08876525, 0.00599761, ..., 0.        , 0.01143873,
0.        ]), array([0.00479702, 0.08860788, 0.04250729, ..., 0.00309843, 0.        ,
0.        ]), array([0.07837302, 0.08858134, 0.04785943, ..., 0.00349181, 0.        ,
0.        ]), array([0.05193114, 0.08799598, 0.00564187, ..., 0.        , 0.        ,
0.02569874]), array([0.0405003 , 0.08792024, 0.0129521 , ..., 0.        , 0.        ,
0.        ]), array([0.03973367, 0.0878371 , 0.01459525, ..., 0.01803278, 0.        ,
0.        ]), array([0.03037312, 0.08733637, 0.0490155 , ..., 0.03967718, 0.01116073,
0.        ]), array([0.02604783, 0.08695285, 0.01431544, ..., 0.        , 0.        ,
0.        ]), array([0.05750079, 0.08636587, 0.00582996, ..., 0.        , 0.        ,
0.        ]), array([0.04018469, 0.08578944, 0.02885184, ..., 0.00320648, 0.        ,
0.        ]), array([0.03728956, 0.08561927, 0.02624475, ..., 0.00317066, 0.00864433,
0.        ]), array([0.01565602, 0.08527648, 0.00715877, ..., 0.00659881, 0.
```

Run this cell to mount your Google Drive.
Learn more

```
from google.colab import drive
drive.mount('/content/drive')
```

```
#print the name of the similar movies based on the index
print('Movies suggested for you : \n')

i = 1

for movie in sorted_similar_movies:
    index = movie[0]
    title_from_index = movies_data[movies_data.index==index]['title'].values
    if (i<30):
        print(i,'.' ,title_from_index)
        i+=1
```

```
Movies suggested for you :

1 . []
2 . []
3 . []
4 . []
5 . []
6 . []
7 . []
8 . []
9 . []
10 . []
11 . []
12 . []
13 . []
14 . []
15 . []
16 . []
17 . []
18 . []
```

```
19 . []
20 . []
21 . []
22 . []
23 . []
24 . []
25 . []
26 . []
27 . []
28 . []
29 . []
```

Run this cell to mount your Google Drive.
Learn more