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

```
from bs4 import BeautifulSoup
```

```
import pandas as pd
```

```
import requests
```

```
density_response = requests.get("https://visaguide.world/asia/")
```

```
density_response.status_code
```

```
200
```

```
density_response.text
```

```
'<!DOCTYPE html>\n<html lang="en-US">\n<head>\n<meta charset="UTF-8">\n<meta name="viewport" content="width=device-width, initial-scale
=1">\n\t<link rel="profile" href="https://gmpg.org/xfn/11"> \n\t<meta name=\'robots\' content=\'index, follow, max-image-preview:large,
max-snippet:-1, max-video-preview:-1\' />\n\t<style>img:is([sizes="auto" i], [sizes^="auto," i]) { contain-intrinsic-size: 3000px 1500p
x }</style>\n\t\n\t<!-- This site is optimized with the Yoast SEO plugin v24.2 - https://yoast.com/wordpress/plugins/seo/ -->\n\t<title
>List of Countries in Asia - VisaGuide.World</title><link rel="preload" data-rocket-preload as="image" href="https://visaguide.world/wp
-content/uploads/2023/01/Visa-Guide-World.svg" fetchpriority="high"><link rel="preload" data-rocket-preload as="style" href="https://fo
```

```
type(density_response.text)
```

```
str
```

```
densitysoup = BeautifulSoup(density_response.text, "html.parser")
```

```
type(densitysoup)
```

```
bs4.BeautifulSoup
def __call__(name: Optional[_StrainableElement]=None, attrs: _StrainableAttributes={}, recursive:
bool=True, string: Optional[_StrainableString]=None, limit: Optional[int]=None, _stacklevel: int=2,
**kwargs: _StrainableAttribute) -> _QueryResults

A data structure representing a parsed HTML or XML document.

Most of the methods you'll call on a BeautifulSoup object are inherited from
PageElement or Tag.

Internally, this class defines the basic interface called by the
```

```
densitytables = densitysoup.find_all("table")
```

Double-click (or enter) to edit

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.


```
len(densitytables)
```

```
3
```

```
densityheaders_tag=densitytables[1].find_all("th")
print(densityheaders_tag)
```

```
[<th>Country</th>, <th>Capital</th>, <th>Area km2</th>, <th>Population (2021)</th>]
```

```
densityheaders = [i.text for i in densityheaders_tag]
print(densityheaders)
```

```
['Country', 'Capital', 'Area km2', 'Population (2021)']
```

```
densityrows_tag = densitytables[1].find_all("td")
densityrows_tag
```

```
[<td>Afghanistan</td>,
 <td>Kabul</td>,
 <td>652,864</td>,
 <td>39,835,428</td>,
 <td>Armenia</td>,
 <td>Yerevan</td>,
 <td>29,743</td>,
 <td>2,968,127</td>,
 <td>Azerbaijan</td>,
 <td>Baku</td>,
 <td>86,600</td>,
 <td>10,223,342</td>,
 <td>Bahrain</td>,
 <td>Manama</td>,
 <td>760</td>,
 <td>1,748,296</td>,
 <td>Bangladesh</td>,
 <td>Dhaka</td>,
 <td>147,570</td>,
 <td>166,303,498</td>,
 <td>Bhutan</td>,
 <td>Thimphu</td>,
 <td>38,394</td>,
 <td>779,898</td>,
 <td>Brunei</td>,
 <td>Bandar Seri Begawan</td>,
 <td>5,765</td>,
 <td>441,532</td>,
 <td>Cambodia</td>,
 <td>Phnom Penh</td>,
 <td>181,035</td>,
 <td>16,946,438</td>,
 <td>China (PRC)</td>,
 <td>Beijing</td>,
 <td>9,596,961</td>,
 <td>1,444,216,107</td>,
 <td>East Timor</td>,
 <td>Dili</td>,
 <td>14,874</td>,
 <td>1,343,873</td>,
 <td>Georgia</td>,
 <td>Tbilisi</td>,
 <td>69,700</td>,
 <td>3,979,765</td>,
 <td>Hong Kong</td>,
 <td>City of Victoria</td>,
 <td>2,755</td>,
 <td>7,552,810</td>,
 <td>India</td>,
 <td>New Delhi</td>,
 <td>3,287,263</td>,
```

```
        <td>1,393,409,038</td>,
        <td>Indonesia</td>,
        <td>Jakarta</td>,
        <td>1,904,569</td>,
        <td>276,361,783</td>,
        <td>Iran</td>,
```

```
densityrows = [i.text for i in densityrows_tag]
densityrows
```

```
['Afghanistan',
 'Kabul',
 '652,864',
 '39,835,428',
 'Armenia',
 'Yerevan',
 '29,743',
 '2,968,127',
 'Azerbaijan',
 'Baku',
 '86,600',
 '10,223,342',
 'Bahrain',
 'Manama',
 '760',
 '1,748,296',
 'Bangladesh',
 'Dhaka',
 '147,570',
 '166,303,498',
 'Bhutan',
 'Thimphu',
 '38,394',
 '779,898',
 'Brunei',
 'Bandar Seri Begawan',
 '5,765',
 '441,532',
 'Cambodia',
 'Phnom Penh',
 '181,035',
 '16,946,438',
 'China (PRC)',
 'Beijing',
 '9,596,961',
 '1,444,216,107',
 'East Timor',
 'Dili',
 '14,874',
 '1,343,873',
 'Georgia',
 'Tbilisi',
 '69,700',
 '3,979,765',
 'Hong Kong',
 'City of Victoria',
 '2,755',
 '7,552,810',
 'India',
 'New Delhi',
 '3,287,263',
 '1,393,409,038',
 'Indonesia',
 'Jakarta',
 '1,904,569',
 '276,361,783',
 'Iran',
 'Tehran',
```

```
density_dict= {}
n = 0
for i in densityheaders:
  density_dict[i] = [densityrows[j] for j in range(n,len(densityrows),len(densityheaders))]
  n += 1
density_dict
```

```
{'Country': ['Afghanistan',
  'Armenia',
  'Azerbaijan',
  'Bahrain',
  'Bangladesh',
  'Bhutan',
```

```
            'Brunei',
            'Cambodia',
            'China (PRC)',
            'East Timor',
            'Georgia',
            'Hong Kong',
            'India',
            'Indonesia',
            'Iran',
            'Iraq',
            'Israel',
            'Japan',
            'Jordan',
            'Kazakhstan',
            'Kuwait',
            'Kyrgyzstan',
            'Laos',
            'Lebanon',
            'Macau',
            'Malaysia',
            'Maldives',
            'Mongolia',
            'Myanmar',
            'Nepal',
            'North Korea',
            'Oman',
            'Pakistan',
            'Palestine',
            'Qatar',
            'Russia',
            'Saudi Arabia',
            'Singapore',
            'South Korea',
            'Sri Lanka',
            'Syria',
            'Taiwan',
            'Tajikistan',
            'Thailand',
            'The Philippines',
            'Turkey',
            'Turkmenistan',
            'United Arab Emirates',
            'Uzbekistan',
            'Vietnam',
            'Yemen'],
           'Capital': ['Kabul',
            'Yerevan',
            'Baku',
            'Manama',
            'Dhaka',
            'Thimphu',
```

```
density_df = pd.DataFrame(density_dict)


density_df.index = range(1,len(density_df) + 1)
density_df
```

	Country	Capital	Area km2	Population (2021)
1	Afghanistan	Kabul	652,864	39,835,428
2	Armenia	Yerevan	29,743	2,968,127
3	Azerbaijan	Baku	86,600	10,223,342
4	Bahrain	Manama	760	1,748,296
5	Bangladesh	Dhaka	147,570	166,303,498
6	Bhutan	Thimphu	38,394	779,898
7	Brunei	Bandar Seri Begawan	5,765	441,532
8	Cambodia	Phnom Penh	181,035	16,946,438
9	China (PRC)	Beijing	9,596,961	1,444,216,107
10	East Timor	Dili	14,874	1,343,873
11	Georgia	Tbilisi	69,700	3,979,765
12	Hong Kong	City of Victoria	2,755	7,552,810
13	India	New Delhi	3,287,263	1,393,409,038
14	Indonesia	Jakarta	1,904,569	276,361,783
15	Iran	Tehran	1,648,195	85,028,759
16	Iraq	Baghdad	438,317	41,179,350
17	Israel	Jerusalem (disputed)	20,770	8,789,774
18	Japan	Tokyo	377,915	126,050,804
19	Jordan	Amman	89,342	10,269,021
20	Kazakhstan	Nur-Sultan	2,724,900	18,994,962
21	Kuwait	Kuwait City	17,818	4,328,550
22	Kyrgyzstan	Bishkek	199,951	6,628,356
23	Laos	Vientiane	236,800	7,379,358
24	Lebanon	Beirut	10,400	6,769,146
25	Macau	Macau	115	658,394
26	Malaysia	Kuala Lumpur	329,847	32,776,194
27	Maldives	Malé	298	543,617
28	Mongolia	Ulaanbaatar	1,564,116	3,329,289
29	Myanmar	Naypyidaw	676,578	54,806,012
30	Nepal	Kathmandu	147,181	29,674,920
31	North Korea	Pyongyang	120,538	25,887,041

```
density_df.to_csv("density_data.csv")

density_df_sorted = density_df.sort_value(by = "population_density",)
```

34	Palestine	Ramallah	6,220	5,222,748

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:

49 Uzbekistan Tashkent 447,400 33,935,763

```
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

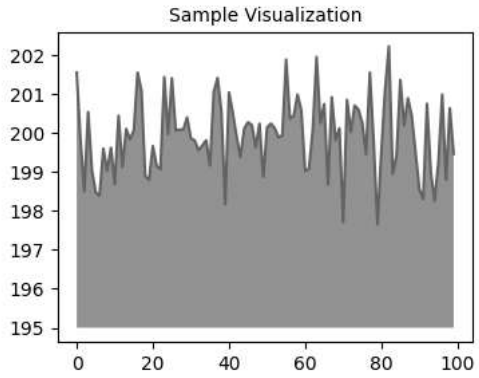
With Colab you can harness the full power of popular Python libraries to analyse and visualise data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualise it. To edit the code, just click the cell and start editing.

```
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.