What is Colaboratory?

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

- Zero configuration required
- Free access to GPUs
- 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:

In [ ]:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

Out[ ]:

```
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:

In [ ]:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

Out[ ]:

```
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.

In [ ]:

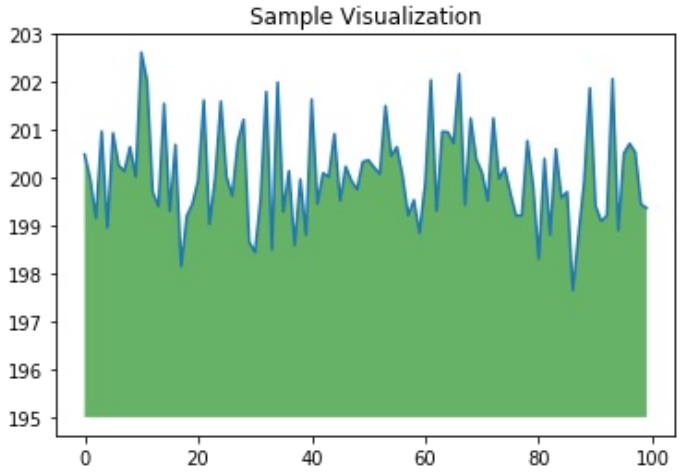
```
import numpy as np
```

```
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```



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
  - TensorFlow 2 in Colab

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
- Tensorflow concepts
- First steps with TensorFlow
- Intro to neural nets
- Intro to sparse data and embeddings

Using Accelerated Hardware

- TensorFlow with GPUs
- TensorFlow with TPUs

Machine Learning Examples

To see end-to-end examples of the interactive machine learning analyses that Colaboratory makes possible, check out these tutorials using models from TensorFlow Hub.

A few 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.**

In [4]:

```
import matplotlib.pyplot as plt
import time
import cv2
%matplotlib inline
```

In [8]:

```
# path = input("Enter first Image path")
firstImg = cv2.imread("cos1.jpg")
```

In [9]:

```
# path = input("Enter first Image path")
secondImg = cv2.imread("tree.jpg")
```

In [10]:

```
firstImg.shape
```

Out[10]:

```
(1534, 950, 3)
```

In [12]:

```
secondImg = cv2.resize(secondImg,(950,1534),interpolation = cv2.INTER_NEAREST)
```

```
secondImg.shape
```

Out[12]:

```
(1534, 950, 3)
```

In [11]:

```
(1080, 1920, 3)
```

Out[11]:

```
(1080, 1920, 3)
```

In [ ]:

```
secondImg.shape
```

In [ ]:

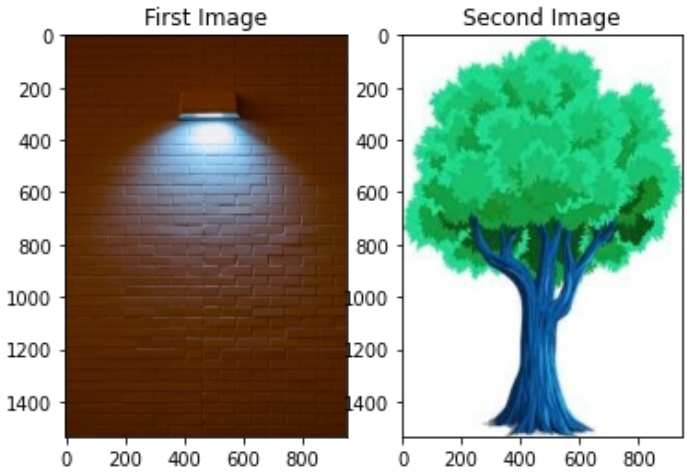
```
(1080, 1920, 3)
```

In [13]:

```
plt.subplot(1,2,1)
plt.title("First Image")
plt.imshow(firstImg,aspect ="auto")
plt.subplot(1,2,2)
plt.title("Second Image")
plt.imshow(secondImg,aspect= "auto")
```

Out[13]:

```
<matplotlib.image.AxesImage at 0x7fb874352b70>
```



In [ ]:

```
<matplotlib.image.AxesImage at 0x7f49c8a36f10>
```

In [14]:

```
plt.show()
addition = cv2.add(firstImg,secondImg);
# addition = addition%255
```

In [15]:

```
subtraction = cv2.subtract(secondImg,firstImg)
```

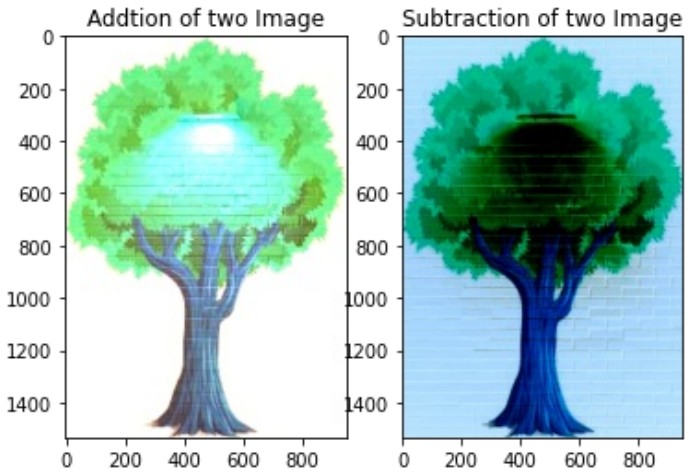
In [16]:

```
plt.subplot(1,2,1)
plt.title("Addtion of two Image")
plt.imshow(addition , aspect='auto')
plt.subplot(1,2,2)
```

```
plt.title("Subtraction of two Image")
plt.imshow(subtraction, aspect='auto')
```

Out[16]:

```
<matplotlib.image.AxesImage at 0x7fb8725f84a8>
```



In [ ]:

```
<matplotlib.image.AxesImage at 0x7f49c89d0a00>
```