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 learn more, check out the Gemini cookbook or visit the Gemini API documentation.

Colab now has AI features powered by Gemini. The video below provides information on how to use these features, whether you're new to Python, or a seasoned veteran.
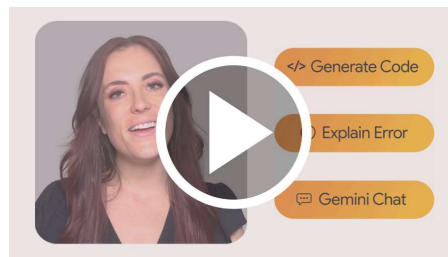

```
Start coding or generate with AI.
```

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 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.

```
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)
```

⇥
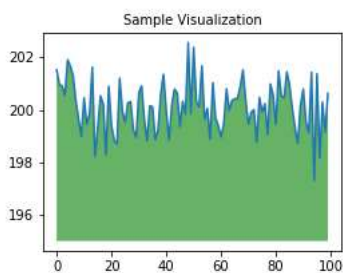
Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

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](#)
- [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](#)
- [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.

Start coding or [generate](#) with AI.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import filters, measure, morphology
from skimage.color import rgb2gray
from skimage.segmentation import clear_border
from scipy import ndimage as ndi

# 1. Load and Display the Image
def load_image(image_path):
    image = cv2.imread(image_path)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.imshow(image_rgb)
    plt.title('Original Image')
```

```python
        plt.axis('off')
        plt.show()
        return image_rgb


    # 2. Preprocess the Image
    def preprocess_image(image_rgb):
        gray_image = rgb2gray(image_rgb)
        blurred_image = filters.gaussian(gray_image, sigma=1.0)
        plt.imshow(blurred_image, cmap='gray')
        plt.title('Grayscale and Blurred Image')
        plt.axis('off')
        plt.show()
        return blurred_image


    # 3. Edge Detection
    def detect_edges(blurred_image):
        edges = filters.sobel(blurred_image)
        plt.imshow(edges, cmap='gray')
        plt.title('Edge Detection')
        plt.axis('off')
        plt.show()
        return edges


    # 4. Thresholding and Morphological Operations
    def apply_threshold(edges):
        threshold_value = filters.threshold_otsu(edges)
        binary_image = edges > threshold_value

        # Morphological operations
        binary_image = morphology.binary_opening(binary_image, morphology.disk(2))
        binary_image = morphology.binary_closing(binary_image, morphology.disk(2))
        binary_image = clear_border(binary_image)  # Remove artifacts connected to the image border

        plt.imshow(binary_image, cmap='gray')
        plt.title('Binary Image after Morphological Operations')
        plt.axis('off')
        plt.show()
        return binary_image


    # 5. Label and Measure Regions
    def label_and_measure(binary_image):
        labeled_image, num_labels = ndi.label(binary_image)
        props = measure.regionprops(labeled_image)

        plt.imshow(labeled_image, cmap='nipy_spectral')
        plt.title(f'Labeled Regions ({num_labels} regions)')
        plt.axis('off')
        plt.show()

        return labeled_image, props


    # 6. Contaminant Detection and Highlighting
    def highlight_contaminants(image_rgb, labeled_image, props):
        output_image = image_rgb.copy()

        for prop in props:
            if prop.area > 50:  # Example criterion for contamination
                minr, minc, maxr, maxc = prop.bbox
                cv2.rectangle(output_image, (minc, minr), (maxc, maxr), (255, 0, 0), 2)

        plt.imshow(output_image)
        plt.title('Contaminants Highlighted')
        plt.axis('off')
        plt.show()


    # 7. Main function to run the project
    def main(image_path):
        image_rgb = load_image(image_path)
        blurred_image = preprocess_image(image_rgb)
        edges = detect_edges(blurred_image)
        binary_image = apply_threshold(edges)
        labeled_image, props = label_and_measure(binary_image)
        highlight_contaminants(image_rgb, labeled_image, props)

    # Run the project with your image path
    if __name__ == "__main__":
        image_path = '/content/apple.jpg'        # Replace with the path to your image
```
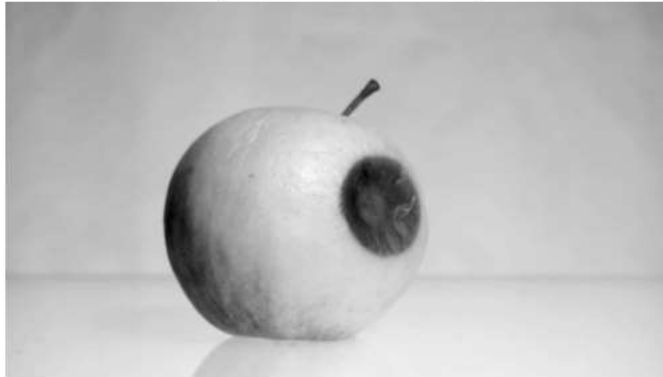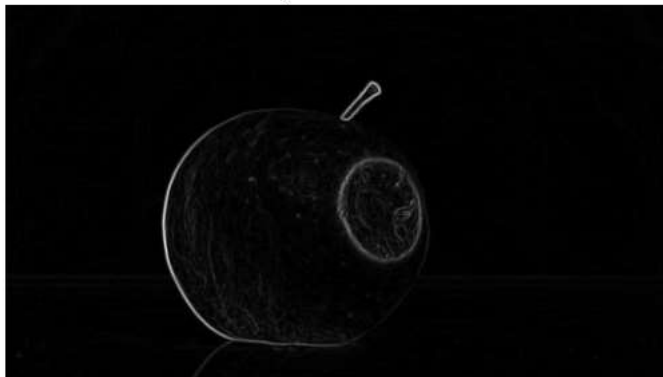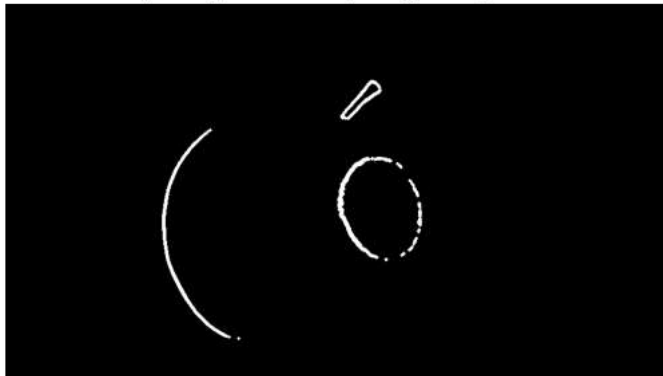
```
main(image_path)
```

Original Image

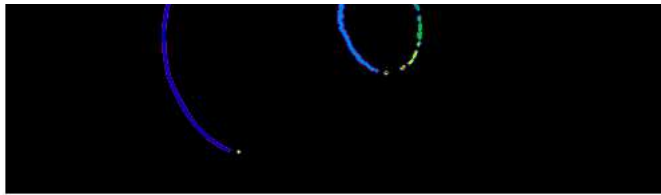Grayscale and Blurred Image

Edge Detection

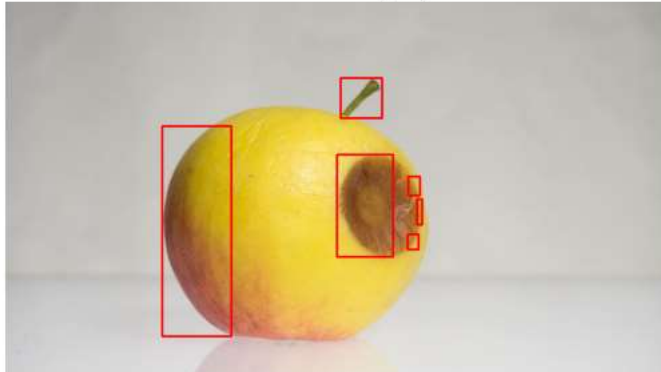Binary Image after Morphological Operations

Labeled Regions (11 regions)

Contaminants Highlighted


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from skimage import filters, measure, morphology
from skimage.color import rgb2gray
from skimage.segmentation import clear_border
from scipy import ndimage as ndi

# 1. Load and Display the Image
def load_image(image_path):
    image = cv2.imread(image_path)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.imshow(image_rgb)
    plt.title('Original Image')
    plt.axis('off')
    plt.show()
    return image_rgb


# 2. Preprocess the Image
def preprocess_image(image_rgb):
    gray_image = rgb2gray(image_rgb)
    blurred_image = filters.gaussian(gray_image, sigma=1.0)
    plt.imshow(blurred_image, cmap='gray')
    plt.title('Grayscale and Blurred Image')
    plt.axis('off')
    plt.show()
    return blurred_image


# 3. Edge Detection with Canny
def detect_edges(blurred_image):
    # Use Canny edge detection instead of Sobel to avoid detecting darkness as edges
    low_threshold = 0.1  # lower threshold for edge detection
    high_threshold = 0.3  # upper threshold for edge detection
    edges = cv2.Canny((blurred_image * 255).astype(np.uint8), int(low_threshold * 255), int(high_threshold * 255))

    plt.imshow(edges, cmap='gray')
    plt.title('Canny Edge Detection')
    plt.axis('off')
    plt.show()
    return edges


# 4. Adaptive Thresholding and Morphological Operations
def apply_threshold(edges):
    # Adaptive thresholding to account for uneven lighting/darkness
    binary_image = cv2.adaptiveThreshold(edges, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                         cv2.THRESH_BINARY, 11, 2)

    # Morphological operations to clean the binary image
    binary_image = morphology.binary_opening(binary_image, morphology.disk(2))
    binary_image = morphology.binary_closing(binary_image, morphology.disk(2))
```

```python
    binary_image = clear_border(binary_image)  # Remove artifacts connected to the image border

    plt.imshow(binary_image, cmap='gray')
    plt.title('Binary Image after Morphological Operations')
    plt.axis('off')
    plt.show()
    return binary_image


# 5. Label and Measure Regions
def label_and_measure(binary_image):
    labeled_image, num_labels = ndi.label(binary_image)
    props = measure.regionprops(labeled_image)

    plt.imshow(labeled_image, cmap='nipy_spectral')
    plt.title(f'Labeled Regions ({num_labels} regions)')
    plt.axis('off')
    plt.show()

    return labeled_image, props


# 6. Contaminant Detection and Highlighting
def highlight_contaminants(image_rgb, labeled_image, props):
    output_image = image_rgb.copy()

    for prop in props:
        if prop.area > 50:  # Example criterion for contamination
            minr, minc, maxr, maxc = prop.bbox
            cv2.rectangle(output_image, (minc, minr), (maxc, maxr), (255, 0, 0), 2)

    plt.imshow(output_image)
    plt.title('Contaminants Highlighted')
    plt.axis('off')
    plt.show()


# 7. Main function to run the project
def main(image_path):
    image_rgb = load_image(image_path)
    blurred_image = preprocess_image(image_rgb)
    edges = detect_edges(blurred_image)
    binary_image = apply_threshold(edges)
    labeled_image, props = label_and_measure(binary_image)
    highlight_contaminants(image_rgb, labeled_image, props)


# Run the project with your image path
if __name__ == "__main__":
    image_path = '/content/pic2.jpg'  # Replace with the path to your image
    main(image_path)
```
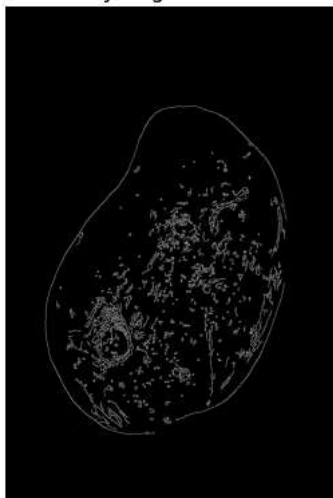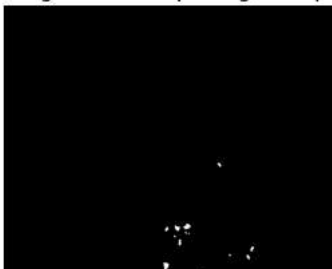
Original Image

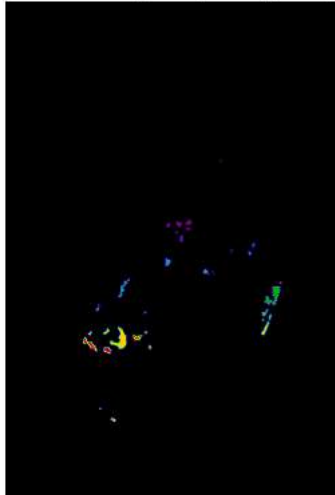Grayscale and Blurred Image

Canny Edge Detection

Binary Image after Morphological Operations

Labeled Regions (35 regions)

Contaminants Highlighted