Week 7: Unstructured data; exploration

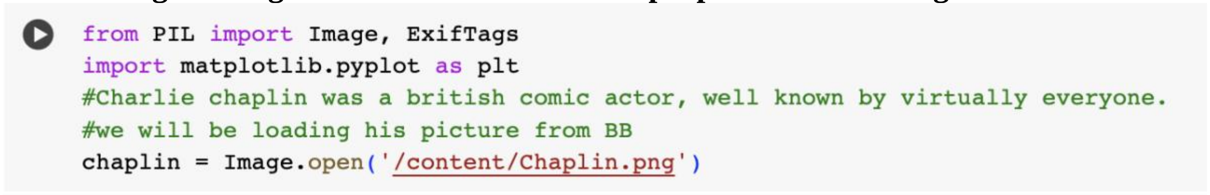
Objective: In this lab session, you will learn basic data exploration techniques for image and text data. We will use Google Colab - Python IDE for most of our tasks.

Go to https://colab.research.google.com/ to start a new Python notebook. You will need to log in to use the Google Colab. You can use your university email or your personal.

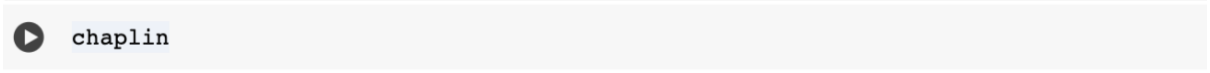
Image Data Exploration
This section investigates the exploration of image data.

- We will explore basic info about image data, including format, size, mode, EXIF and histogram of an image.
- To start, please download the Image folder, unzip the folder and load images from the folder.

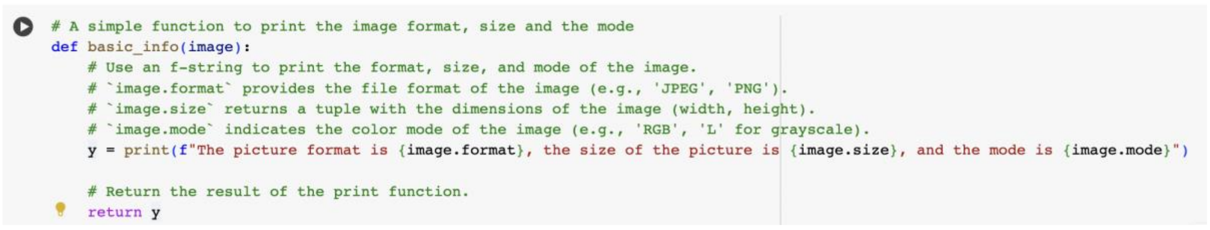
Task1a: Ingest Image data and check the basic properties of the image



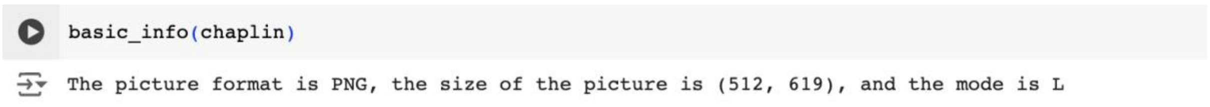
```python
from PIL import Image, ExifTags
import matplotlib.pyplot as plt
#Charlie chaplin was a british comic actor, well known by virtually everyone.
#we will be loading his picture from BB
chaplin = Image.open('/content/Chaplin.png')
```



```python
chaplin
```



```python
# A simple function to print the image format, size and the mode
def basic_info(image):
    # Use an f-string to print the format, size, and mode of the image.
    # `image.format` provides the file format of the image (e.g., 'JPEG', 'PNG').
    # `image.size` returns a tuple with the dimensions of the image (width, height).
    # `image.mode` indicates the color mode of the image (e.g., 'RGB', 'L' for grayscale).
    y = print(f"The picture format is {image.format}, the size of the picture is {image.size}, and the mode is {image.mode}")

    # Return the result of the print function.
    return y
```

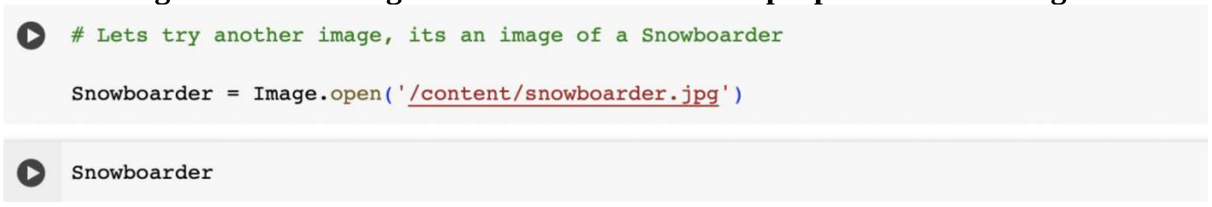


```python
basic_info(chaplin)
```

```
The picture format is PNG, the size of the picture is (512, 619), and the mode is L
```

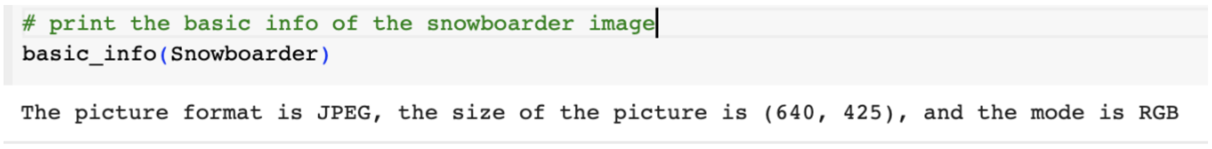
Note:
- The format attribute identifies the source of an image. If the image was not read from a file, it is set to None.
- The size attribute is a 2-tuple containing width and height (in pixels).
- The mode attribute defines the number and names of the bands in the image and the pixel type and depth. Typical modes are “L” (luminance) for grayscale images, “RGB” for actual colour images, and “CMYK” for pre-press images.

Task1b: Ingest another Image data and check the basic properties of the image



```
# Lets try another image, its an image of a Snowboarder

Snowboarder = Image.open('/content/snowboarder.jpg')
```

```
Snowboarder
```



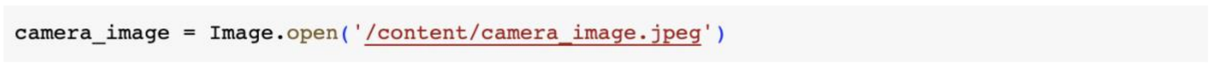
```
# print the basic info of the snowboarder image
basic_info(Snowboarder)
```

```
The picture format is JPEG, the size of the picture is (640, 425), and the mode is RGB
```

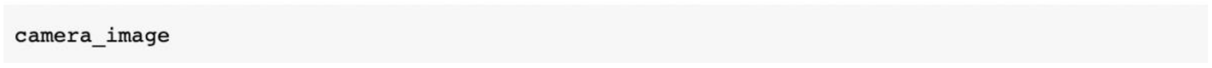
Task2: EXIF Data for Image data

As part of image exploration.

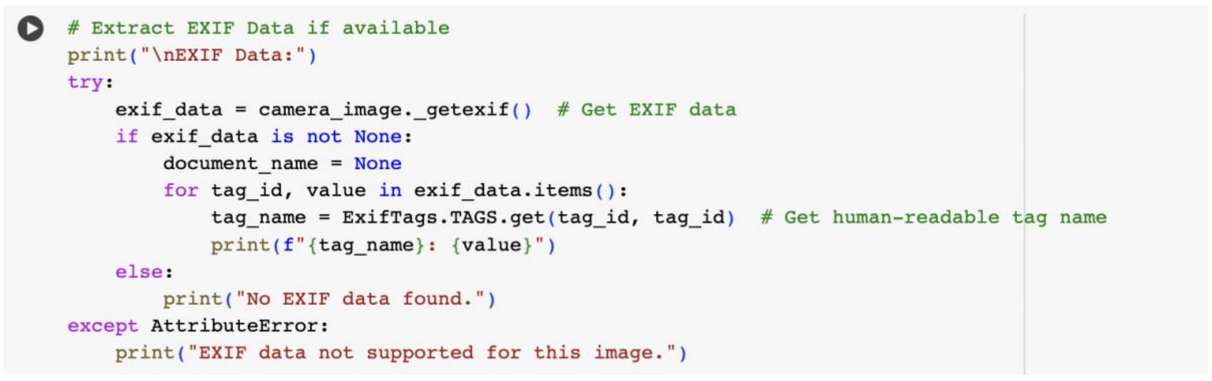
- We can also get the EXIF of an image, particularly images from digital devices.
- EXIF (Exchangeable Image File Format) files store essential data about photographs. Almost all digital cameras create these data files when you snap a new picture. An EXIF file holds all the information about the image itself — such as the exposure level, where you took the photo, and any settings you used



```
camera_image = Image.open('/content/camera_image.jpeg')
```



```
camera_image
```



```
# Extract EXIF Data if available
print("\nEXIF Data:")
try:
    exif_data = camera_image._getexif()  # Get EXIF data
    if exif_data is not None:
        document_name = None
        for tag_id, value in exif_data.items():
            tag_name = ExifTags.TAGS.get(tag_id, tag_id)  # Get human-readable tag name
            print(f"{tag_name}: {value}")
    else:
        print("No EXIF data found.")
except AttributeError:
    print("EXIF data not supported for this image.")
```

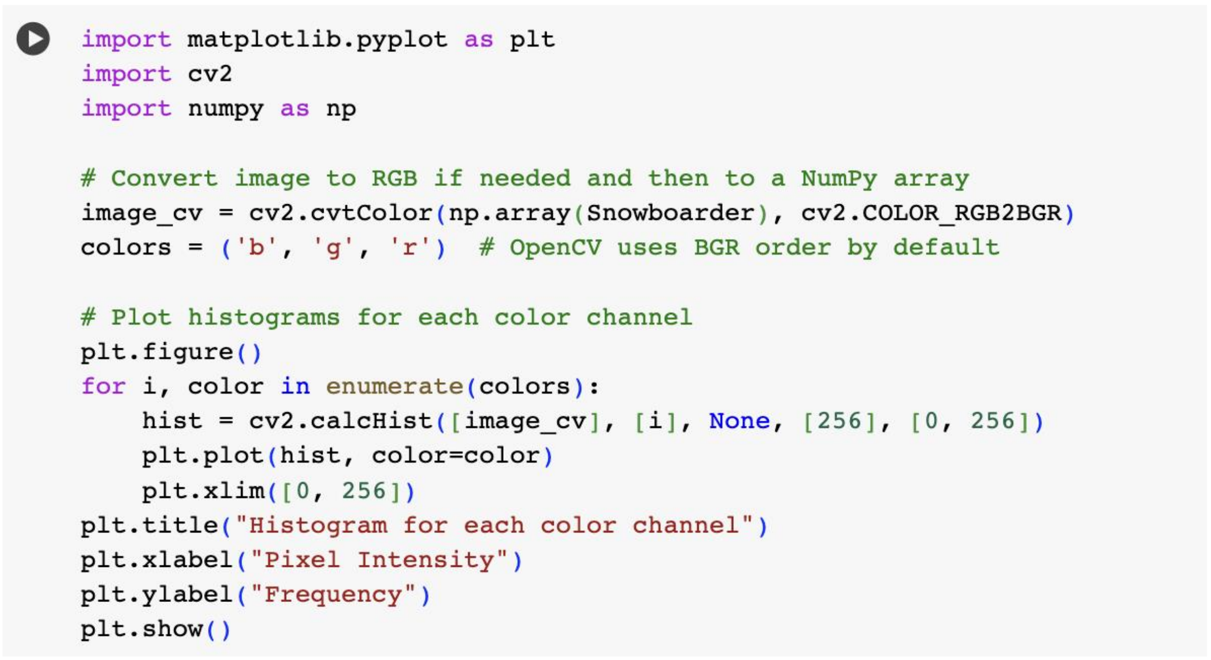
Extra Exercise on Exif

Try to snap a picture on your phone, save it and load it here. Try getting the Exif of the image you have just snapped and loaded here.

Task3: Histogram of images

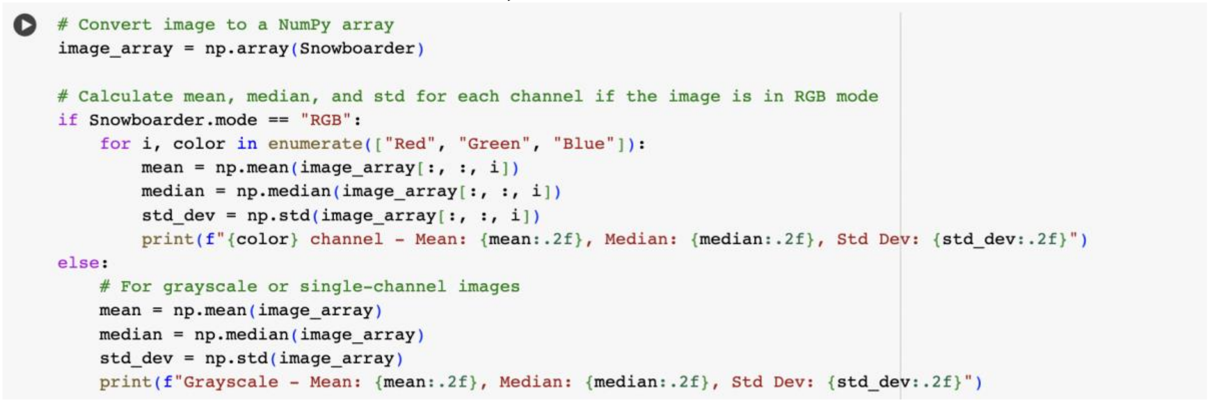
The histogram plots the number of pixels in the image (vertical axis) with a particular brightness or tonal value (horizontal axis).

- Let's try to plot the histogram for the snowboarder image.



```python
import matplotlib.pyplot as plt
import cv2
import numpy as np

# Convert image to RGB if needed and then to a NumPy array
image_cv = cv2.cvtColor(np.array(Snowboarder), cv2.COLOR_RGB2BGR)
colors = ('b', 'g', 'r')  # OpenCV uses BGR order by default

# Plot histograms for each color channel
plt.figure()
for i, color in enumerate(colors):
    hist = cv2.calcHist([image_cv], [i], None, [256], [0, 256])
    plt.plot(hist, color=color)
    plt.xlim([0, 256])
plt.title("Histogram for each color channel")
plt.xlabel("Pixel Intensity")
plt.ylabel("Frequency")
plt.show()
```

Task4: Calculate the statistics: Mean, median and Mode of the channels.



```python
# Convert image to a NumPy array
image_array = np.array(Snowboarder)

# Calculate mean, median, and std for each channel if the image is in RGB mode
if Snowboarder.mode == "RGB":
    for i, color in enumerate(["Red", "Green", "Blue"]):
        mean = np.mean(image_array[:, :, i])
        median = np.median(image_array[:, :, i])
        std_dev = np.std(image_array[:, :, i])
        print(f"{color} channel - Mean: {mean:.2f}, Median: {median:.2f}, Std Dev: {std_dev:.2f}")
else:
    # For grayscale or single-channel images
    mean = np.mean(image_array)
    median = np.median(image_array)
    std_dev = np.std(image_array)
    print(f"Grayscale - Mean: {mean:.2f}, Median: {median:.2f}, Std Dev: {std_dev:.2f}")
```

Extra Exercise on Image Loading & Exploration.

- Try to load 5 images from the Folder.
- Check the image format.
- Check the image size.
- Check the Image mode.
- Check the Exif data to see if it has
- Compute and print the histogram

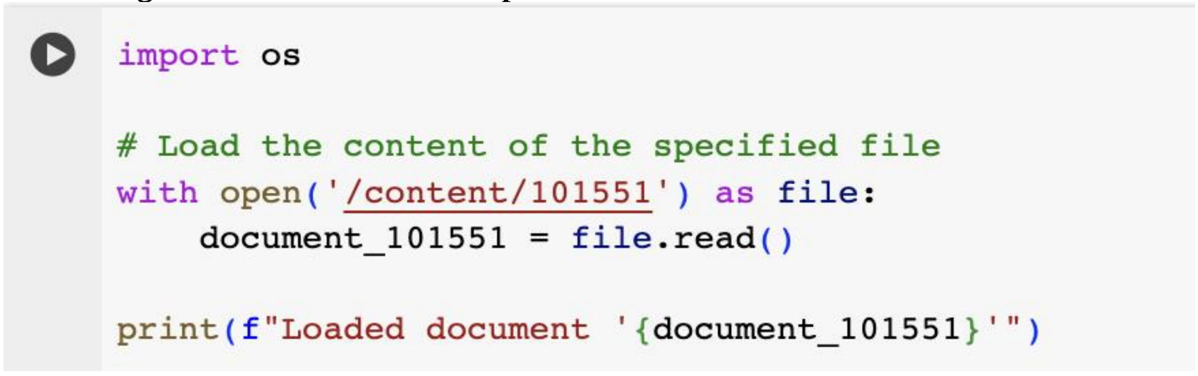
Ps. You can do that using our basic_info function. To invoke the function, for example, basic_info(image_variable_name)

Text Data

This is the second part of today's seminar, covering the exploration of textual data.
To start, please download the text data from BB and unzip the folder. There are 20 subfolders. Out of the 20, we will only work on the autos folder.

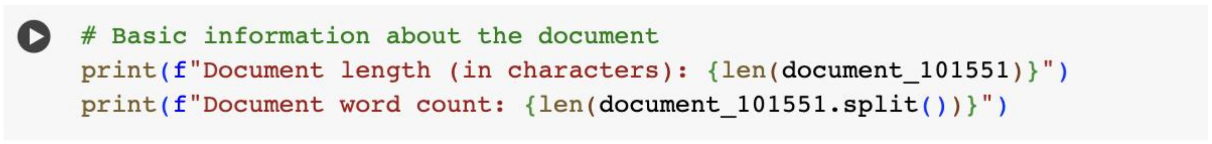
The text folder contains documents from the well-known text dataset downloadable from here, consisting of 20,000 messages collected from 20 internet newsgroups.

Task5: Ingest the document file and print the file

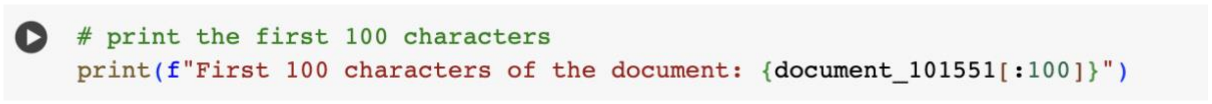


```python
import os

# Load the content of the specified file
with open('/content/101551') as file:
    document_101551 = file.read()

print(f"Loaded document '{document_101551}'")
```

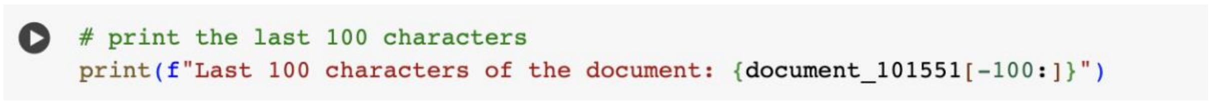
Task5b: Investigate basic properties like document length, word count, first 100 and last 100 characters



```python
# Basic information about the document
print(f"Document length (in characters): {len(document_101551)}")
print(f"Document word count: {len(document_101551.split())}")
```

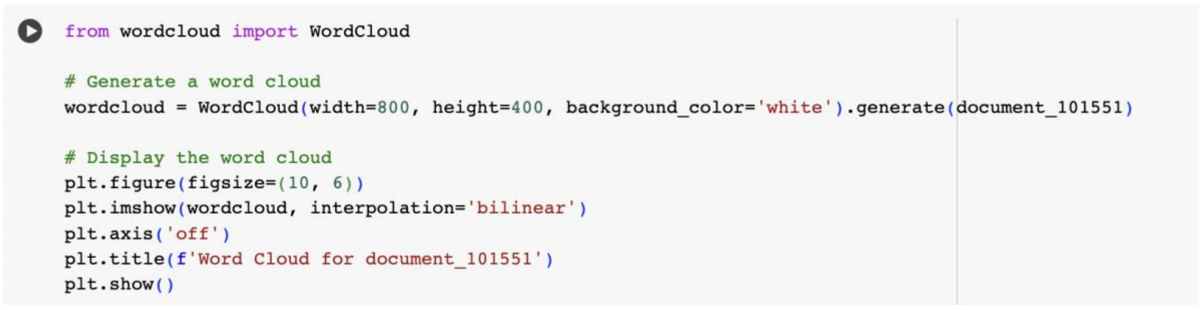


```python
# print the first 100 characters
print(f"First 100 characters of the document: {document_101551[:100]}")
```



```python
# print the last 100 characters
print(f"Last 100 characters of the document: {document_101551[-100:]}")
```

Task6: Plot a word count for the document file



```python
from wordcloud import WordCloud

# Generate a word cloud
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(document_101551)

# Display the word cloud
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title(f'Word Cloud for document_101551')
plt.show()
```

Extra Exercise on the Text Data Exploration

Ingest 2 more text documents from any folder of your choice.

- Check and print the length of the documents
- Check and print the word count of the documents
- Create a word cloud for each of the documents