21MIA1097 Amith Reddy

Link: https://drive.google.com/file/d/14aeU6ugz6pTbX-Mj2DmtPtEUAoUiQT-u/view?usp=sharing

```
pip install ffmpeg-python av
```

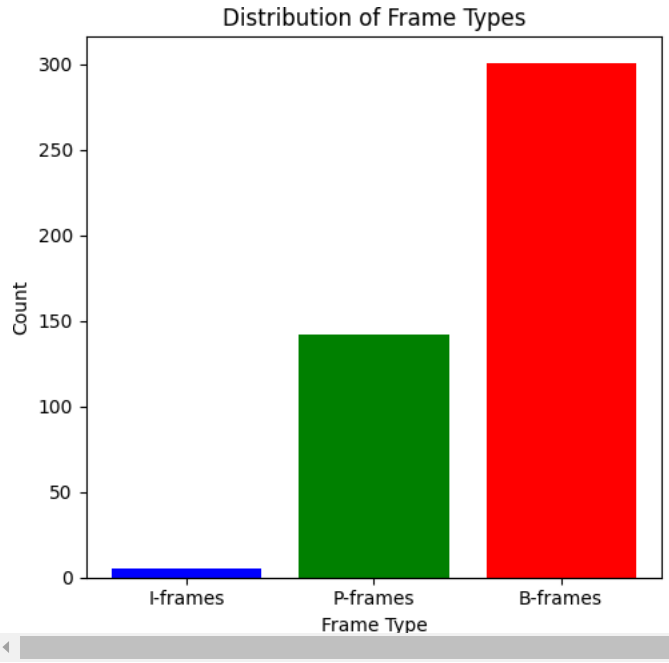
```
Requirement already satisfied: ffmpeg-python in /usr/local/lib/python3.10/dist-packages (0.2.0)
Requirement already satisfied: av in /usr/local/lib/python3.10/dist-packages (12.3.0)
Requirement already satisfied: future in /usr/local/lib/python3.10/dist-packages (from ffmpeg-python) (1.0.0)
```

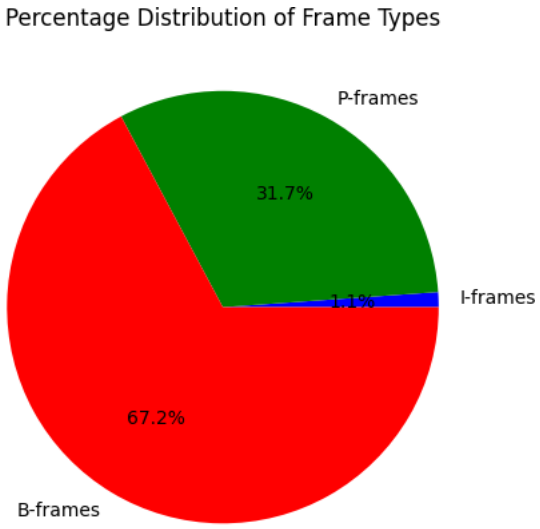
```
import av

#video file
video_path = '/content/12336802-hd_1280_720_30fps.mp4'

container = av.open(video_path)

frame_count = 0

for frame in container.decode(video=0):
    frame_count += 1
    print(f"Frame: {frame_count}, PTS: {frame.pts}, Time: {frame.time}, Width: {frame.width}, Height: {frame.height}")
```

```
Frame: 391, PTS: 390390, Time: 13.013, Width: 1280, Height: 720
Frame: 392, PTS: 391391, Time: 13.046366666666666, Width: 1280, Height: 720
Frame: 393, PTS: 392392, Time: 13.079733333333333, Width: 1280, Height: 720
Frame: 394, PTS: 393393, Time: 13.1131, Width: 1280, Height: 720
Frame: 395, PTS: 394394, Time: 13.146466666666667, Width: 1280, Height: 720
Frame: 396, PTS: 395395, Time: 13.179833333333333, Width: 1280, Height: 720
Frame: 397, PTS: 396396, Time: 13.2132, Width: 1280, Height: 720
Frame: 398, PTS: 397397, Time: 13.246566666666666, Width: 1280, Height: 720
Frame: 399, PTS: 398398, Time: 13.279933333333334, Width: 1280, Height: 720
Frame: 400, PTS: 399399, Time: 13.3133, Width: 1280, Height: 720
Frame: 401, PTS: 400400, Time: 13.346666666666666, Width: 1280, Height: 720
Frame: 402, PTS: 401401, Time: 13.380033333333333, Width: 1280, Height: 720
Frame: 403, PTS: 402402, Time: 13.4134, Width: 1280, Height: 720
Frame: 404, PTS: 403403, Time: 13.446766666666667, Width: 1280, Height: 720
Frame: 405, PTS: 404404, Time: 13.480133333333333, Width: 1280, Height: 720
Frame: 406, PTS: 405405, Time: 13.5135, Width: 1280, Height: 720
Frame: 407, PTS: 406406, Time: 13.546866666666666, Width: 1280, Height: 720
Frame: 408, PTS: 407407, Time: 13.580233333333334, Width: 1280, Height: 720
Frame: 409, PTS: 408408, Time: 13.6136, Width: 1280, Height: 720
Frame: 410, PTS: 409409, Time: 13.646966666666666, Width: 1280, Height: 720
Frame: 411, PTS: 410410, Time: 13.680333333333333, Width: 1280, Height: 720
Frame: 412, PTS: 411411, Time: 13.7137, Width: 1280, Height: 720
Frame: 413, PTS: 412412, Time: 13.747066666666667, Width: 1280, Height: 720
Frame: 414, PTS: 413413, Time: 13.780433333333333, Width: 1280, Height: 720
Frame: 415, PTS: 414414, Time: 13.8138, Width: 1280, Height: 720
Frame: 416, PTS: 415415, Time: 13.847166666666666, Width: 1280, Height: 720
Frame: 417, PTS: 416416, Time: 13.880533333333334, Width: 1280, Height: 720
Frame: 418, PTS: 417417, Time: 13.9139, Width: 1280, Height: 720
Frame: 419, PTS: 418418, Time: 13.947266666666666, Width: 1280, Height: 720
Frame: 420, PTS: 419419, Time: 13.980633333333333, Width: 1280, Height: 720
Frame: 421, PTS: 420420, Time: 14.014, Width: 1280, Height: 720
Frame: 422, PTS: 421421, Time: 14.047366666666667, Width: 1280, Height: 720
Frame: 423, PTS: 422422, Time: 14.080733333333333, Width: 1280, Height: 720
Frame: 424, PTS: 423423, Time: 14.1141, Width: 1280, Height: 720
Frame: 425, PTS: 424424, Time: 14.147466666666666, Width: 1280, Height: 720
Frame: 426, PTS: 425425, Time: 14.180833333333334, Width: 1280, Height: 720
Frame: 427, PTS: 426426, Time: 14.2142, Width: 1280, Height: 720
Frame: 428, PTS: 427427, Time: 14.247566666666666, Width: 1280, Height: 720
Frame: 429, PTS: 428428, Time: 14.280933333333333, Width: 1280, Height: 720
Frame: 430, PTS: 429429, Time: 14.3143, Width: 1280, Height: 720
Frame: 431, PTS: 430430, Time: 14.347666666666667, Width: 1280, Height: 720
Frame: 432, PTS: 431431, Time: 14.381033333333333, Width: 1280, Height: 720
Frame: 433, PTS: 432432, Time: 14.4144, Width: 1280, Height: 720
Frame: 434, PTS: 433433, Time: 14.447766666666666, Width: 1280, Height: 720
Frame: 435, PTS: 434434, Time: 14.481133333333334, Width: 1280, Height: 720
Frame: 436, PTS: 435435, Time: 14.5145, Width: 1280, Height: 720
Frame: 437, PTS: 436436, Time: 14.547866666666666, Width: 1280, Height: 720
Frame: 438, PTS: 437437, Time: 14.581233333333333, Width: 1280, Height: 720
Frame: 439, PTS: 438438, Time: 14.6146, Width: 1280, Height: 720
Frame: 440, PTS: 439439, Time: 14.647966666666667, Width: 1280, Height: 720
Frame: 441, PTS: 440440, Time: 14.681333333333333, Width: 1280, Height: 720
Frame: 442, PTS: 441441, Time: 14.7147, Width: 1280, Height: 720
Frame: 443, PTS: 442442, Time: 14.748066666666666, Width: 1280, Height: 720
Frame: 444, PTS: 443443, Time: 14.781433333333334, Width: 1280, Height: 720
Frame: 445, PTS: 444444, Time: 14.8148, Width: 1280, Height: 720
Frame: 446, PTS: 445445, Time: 14.848166666666666, Width: 1280, Height: 720
Frame: 447, PTS: 446446, Time: 14.881533333333334, Width: 1280, Height: 720
Frame: 448, PTS: 447447, Time: 14.9149, Width: 1280, Height: 720
```

```
import matplotlib.pyplot as plt

# Path to the video file
video_path = '/content/12336802-hd_1280_720_30fps.mp4'

# Open the video file
container = av.open(video_path)

# Counters for frame types
i_frame_count = 0
p_frame_count = 0
b_frame_count = 0

# Total frames counter
total_frames = 0

# Iterate over frames in the video
for frame in container.decode(video=0):
    total_frames += 1
    if frame.pict_type == 'I':
        i_frame_count += 1
    elif frame.pict_type == 'P':
        p_frame_count += 1
    elif frame.pict_type == 'B':
        b_frame_count += 1

# Calculate percentages
i_frame_percentage = (i_frame_count / total_frames) * 100
p_frame_percentage = (p_frame_count / total_frames) * 100
b_frame_percentage = (b_frame_count / total_frames) * 100

# Print results
print(f"Total frames: {total_frames}")
print(f"I-frames: {i_frame_count} ({i_frame_percentage:.2f}%)")
print(f"P-frames: {p_frame_count} ({p_frame_percentage:.2f}%)")
print(f"B-frames: {b_frame_count} ({b_frame_percentage:.2f}%)")

# Plotting the distribution
frame_types = ['I-frames', 'P-frames', 'B-frames']
counts = [i_frame_count, p_frame_count, b_frame_count]

plt.figure(figsize=(10, 5))

# Bar chart
plt.subplot(1, 2, 1)
plt.bar(frame_types, counts, color=['blue', 'green', 'red'])
plt.title('Distribution of Frame Types')
plt.xlabel('Frame Type')
plt.ylabel('Count')

# Pie chart
plt.subplot(1, 2, 2)
plt.pie(counts, labels=frame_types, autopct='%1.1f%%', colors=['blue', 'green', 'red'])
plt.title('Percentage Distribution of Frame Types')

plt.tight_layout()
plt.show()
```

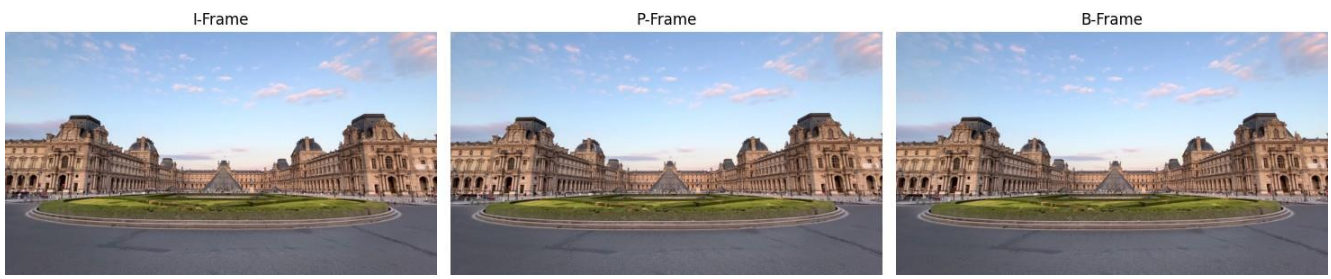
```
Total frames: 448
I-frames: 5 (1.12%)
P-frames: 142 (31.70%)
B-frames: 301 (67.19%)
```






```
import ffmpeg
import av
import os

# Path to the video file
video_path = '/content/12336802-hd_1280_720_30fps.mp4'
output_dir = 'extracted_frames'

# Create output directory if it doesn't exist
os.makedirs(output_dir, exist_ok=True)

# Open the video file
container = av.open(video_path)

# Iterate over frames in the video
frame_counter = 0
for frame in container.decode(video=0):
    # Define the frame type
    frame_type = frame.pict_type
    frame_counter += 1
    # Save frames as image files
    output_filename = os.path.join(output_dir, f"{frame_counter}_{frame_type}.png")
    frame.to_image().save(output_filename)
```

```
from PIL import Image
import matplotlib.pyplot as plt

# Load and display the first few frames of each type (I, P, B)
frame_filenames = sorted(os.listdir(output_dir))

plt.figure(figsize=(15, 5))

# Display the first I-frame
i_frame_path = next((os.path.join(output_dir, f) for f in frame_filenames if 'I' in f), None)
if i_frame_path:
    img = Image.open(i_frame_path)
    plt.subplot(1, 3, 1)
    plt.imshow(img)
    plt.title("I-Frame")
    plt.axis('off')

# Display the first P-frame
p_frame_path = next((os.path.join(output_dir, f) for f in frame_filenames if 'P' in f), None)
if p_frame_path:
    img = Image.open(p_frame_path)
    plt.subplot(1, 3, 2)
    plt.imshow(img)
    plt.title("P-Frame")
    plt.axis('off')

# Display the first B-frame
b_frame_path = next((os.path.join(output_dir, f) for f in frame_filenames if 'B' in f), None)
if b_frame_path:
    img = Image.open(b_frame_path)
    plt.subplot(1, 3, 3)
    plt.imshow(img)
    plt.title("B-Frame")
    plt.axis('off')

plt.tight_layout()
plt.show()
```



Generate 10 random numbers using numpy Close

```
import os

# Path to the directory containing the extracted frames
output_dir = 'extracted_frames'

# Initialize counters and size accumulators
i_frame_sizes = []
p_frame_sizes = []
b_frame_sizes = []

# Calculate the file sizes for each frame type
for filename in os.listdir(output_dir):
    filepath = os.path.join(output_dir, filename)
    file_size = os.path.getsize(filepath)

    if 'I' in filename:
        i_frame_sizes.append(file_size)
    elif 'P' in filename:
        p_frame_sizes.append(file_size)
    elif 'B' in filename:
        b_frame_sizes.append(file_size)

# Calculate average file sizes
average_i_frame_size = sum(i_frame_sizes) / len(i_frame_sizes) if i_frame_sizes else 0
average_p_frame_size = sum(p_frame_sizes) / len(p_frame_sizes) if p_frame_sizes else 0
average_b_frame_size = sum(b_frame_sizes) / len(b_frame_sizes) if b_frame_sizes else 0

# Print results
print(f"Average I-Frame size: {average_i_frame_size:.2f} bytes")
print(f"Average P-Frame size: {average_p_frame_size:.2f} bytes")
print(f"Average B-Frame size: {average_b_frame_size:.2f} bytes")
```

```
Average I-Frame size: 885858.80 bytes
Average P-Frame size: 943993.34 bytes
Average B-Frame size: 932398.44 bytes
```

```
import av
import os

# Path to the video file and output directory
video_path = '/content/12336802-hd_1280_720_30fps.mp4'
output_dir = 'i_frames'

# Create the output directory if it doesn't exist
os.makedirs(output_dir, exist_ok=True)

# Open the video file
container = av.open(video_path)

# Iterate over frames and save only I-frames
frame_counter = 0
for frame in container.decode(video=0):
    if frame.pict_type == 'I':
        frame_counter += 1
        # Save the I-frame as an image file
        output_filename = os.path.join(output_dir, f"I_frame_{frame_counter}.png")
        frame.to_image().save(output_filename)

print(f"Extracted and saved {frame_counter} I-frames.")
```

```
Extracted and saved 5 I-frames.
```