The following is a 10 minute zoo tour video . The Aim is to use pretrained YOLO to detect/track and identify animals in the video

In [ ]:
```
!pip install -U yt-dlp  # Download youtube video
!pip install opencv-python-headless  # OpenCV for video processing
!pip install ultralytics  # YOLOv8 library
```

...

In [ ]:
```
import cv2
import yt_dlp
import torch
from ultralytics import YOLO
```

```
Creating new Ultralytics Settings v0.0.6 file ✅
View Ultralytics Settings with 'yolo settings' or at '/root/.config/
Ultralytics/settings.json'
Update Settings with 'yolo settings key=value', i.e. 'yolo settings
runs_dir=path/to/dir'. For help see https://docs.ultralytics.com/qui
ckstart/#ultralytics-settings. (https://docs.ultralytics.com/quickst
art/#ultralytics-settings.)
```

In [ ]:
```
url = 'https://www.youtube.com/watch?v=bMp0yfuJcIg'
```

In [ ]:
```
ydl_opts = {
        'format': 'bestvideo+bestaudio',
        'outtmpl': '/content/video.mp4',
    }
```

In [ ]:
```
with yt_dlp.YoutubeDL(ydl_opts) as ydl:
        ydl.download([url])
print("Video Downloaded Successfully.")
```

```
[youtube] Extracting URL: https://www.youtube.com/watch?v=bMp0yfuJcI
g (https://www.youtube.com/watch?v=bMp0yfuJcIg)
[youtube] bMp0yfuJcIg: Downloading webpage
[youtube] bMp0yfuJcIg: Downloading ios player API JSON
[youtube] bMp0yfuJcIg: Downloading mweb player API JSON
[youtube] bMp0yfuJcIg: Downloading player 03dbdfab
[youtube] bMp0yfuJcIg: Downloading m3u8 information
[info] bMp0yfuJcIg: Downloading 1 format(s): 315+251
[download] Destination: /content/video.mp4.f315.webm
[download] 100% of    1.72GiB in 00:01:05 at 26.86MiB/s
[download] Destination: /content/video.mp4.f251.webm
[download] 100% of   10.83MiB in 00:00:00 at 27.04MiB/s
[Merger] Merging formats into "/content/video.mp4.webm"
Deleting original file /content/video.mp4.f251.webm (pass -k to kee
p)
Deleting original file /content/video.mp4.f315.webm (pass -k to kee
p)
Video Downloaded Successfully.
```

In [ ]:
```
model = YOLO('yolov8n.pt')
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt (https://github.com/ultralytics/assets/releases/download/v8.3.0/yolov8n.pt) to 'yolov8n.pt'...

100%|██████████| 6.25M/6.25M [00:00<00:00, 74.5MB/s]
```

In [ ]:
```
import os
video_path = '/content/video.mp4.webm'
if os.path.exists(video_path):
    print(f"File {video_path} exists.")
else:
    print(f"File {video_path} does not exist.")
```

```
File /content/video.mp4.webm exists.
```

In [ ]:
```
# Check if the video was loaded correctly
cap = cv2.VideoCapture('/content/video.mp4.webm')
if not cap.isOpened():
    print("Error: Couldn't open the video file!")
else:
    print("Video loaded successfully.")
```

```
Video loaded successfully.
```

In [ ]:
```
!pip uninstall -y opencv-python-headless
!pip install opencv-python --upgrade
```

```
Found existing installation: opencv-python-headless 4.10.0.84
Uninstalling opencv-python-headless-4.10.0.84:
  Successfully uninstalled opencv-python-headless-4.10.0.84
Requirement already satisfied: opencv-python in /usr/local/lib/pytho
n3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/pytho
n3.10/dist-packages (from opencv-python) (1.26.4)
```

In [ ]:
```
ret, frame = cap.read()
if ret:
    print("Frame read successfully.")
    cv2_imshow(frame)
    cap.release()
else:
    print("Error: Couldn't read the frame.")
```

```
Frame read successfully.
```

In [ ]:
```
# Function to detect animals in the video.
def detect_objects_in_video(video_path):
    cap = cv2.VideoCapture(video_path)
    detected_objects = []  # List to store all detected objects , YOL(

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        results = model(frame)
        for result in results:
            for box in result.boxes:
                class_id = int(box.cls[0])
                class_name = model.names[class_id]
                detected_objects.append(class_name)
        annotated_frame = results[0].plot()
        cv2_imshow(annotated_frame)

    cap.release()

    return detected_objects

detected_objects = detect_objects_in_video('/content/video.mp4.webm')
print("Detected Objects in Video:", detected_objects)
```


```
0: 384x640 4 persons, 56.5ms
```

In [ ]: