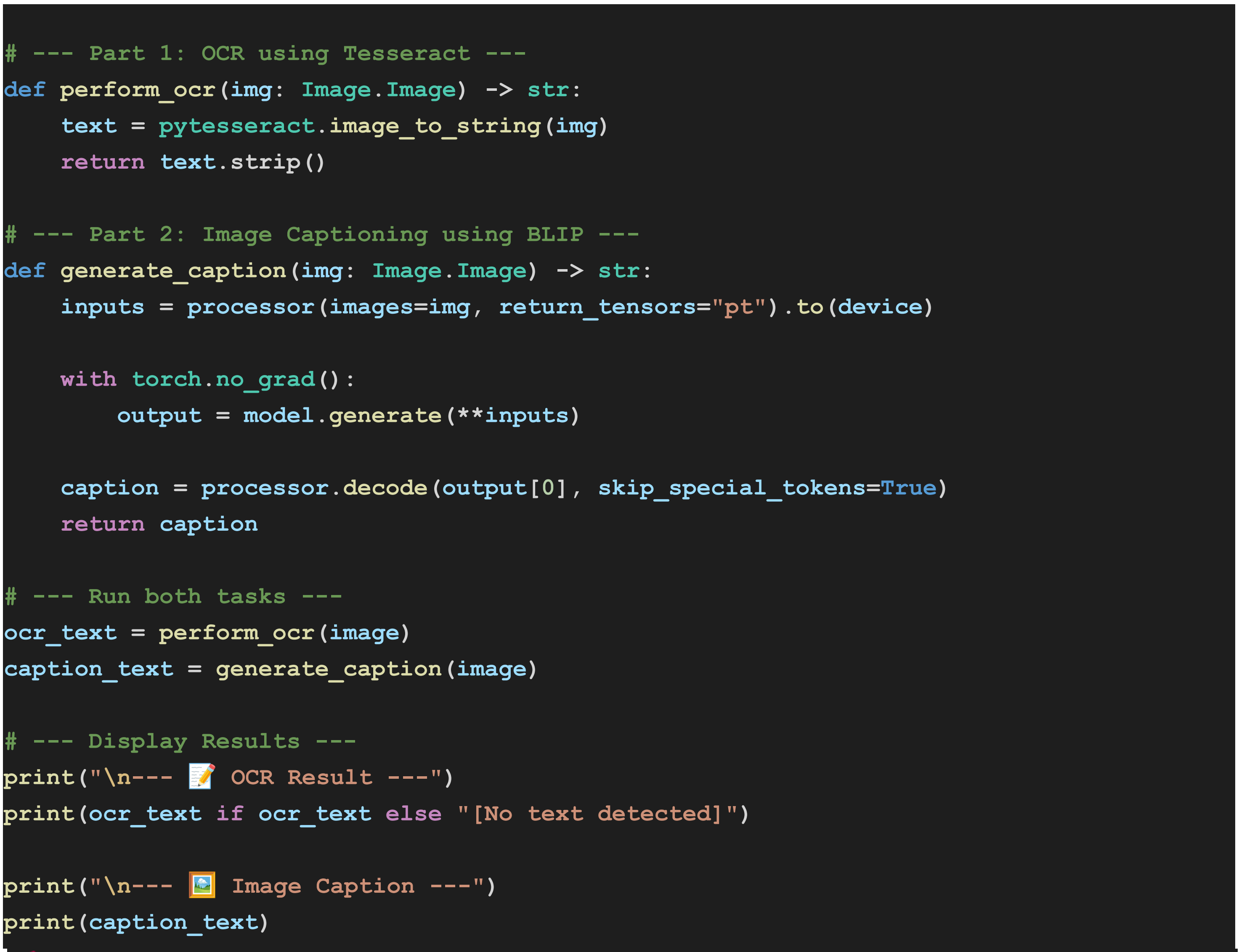
EX.NO.: 05
DATE: 30.06.2025

CV WITH NLP TO PERFORM OCR

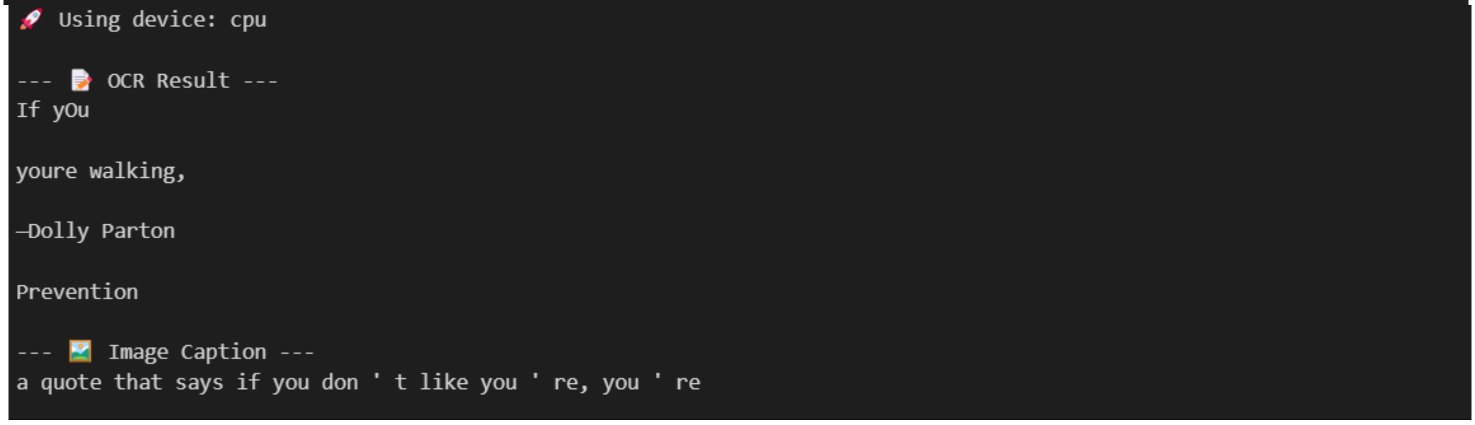
AIM
To develop a Python-based system that integrates **Computer Vision (CV)** and **Natural Language Processing (NLP)** to:
1. Extract textual content from an image using **Optical Character Recognition (OCR)**.
2. Generate a meaningful natural language **caption** describing the image content.

ALGORITHM
1. **Import necessary libraries** (`PIL`, `pytesseract`, `transformers`, `torch`).
2. **Set the path to Tesseract OCR executable** to ensure `pytesseract` can find and use it.
3. **Load the input image** using `PIL.Image`.
4. **Perform OCR**:
   - Use `pytesseract.image_to_string(image)` to extract text from the image.
5. **Perform Image Captioning**:
   - Load the BLIP processor and model from HuggingFace (`Salesforce/blip-image-captioning-base`).
   - Use the processor to preprocess the image and generate inputs for the model.
   - Pass the image through the model to generate a caption.
6. **Display results**:
   - Show the image using `matplotlib`.
   - Print both OCR text and the generated image caption.

CODE AND OUTPUT
```
# 📦 Import Libraries
from PIL import Image
import pytesseract
from transformers import BlipProcessor, BlipForConditionalGeneration
import torch
import os

# ✅ Load Image
image_path = "sample1.jpg"
assert os.path.exists(image_path), f"❌ Image not found: {image_path}"
image = Image.open(image_path).convert("RGB")

# ✅ Device setup
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"🚀 Using device: {device}")

# ✅ Load processor and model ONCE globally
processor = BlipProcessor.from_pretrained(
    "Salesforce/blip-image-captioning-base",
    use_fast=True
)
model =
BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")
model.to(device)
```



```python
# --- Part 1: OCR using Tesseract ---
def perform_ocr(img: Image.Image) -> str:
    text = pytesseract.image_to_string(img)
    return text.strip()

# --- Part 2: Image Captioning using BLIP ---
def generate_caption(img: Image.Image) -> str:
    inputs = processor(images=img, return_tensors="pt").to(device)

    with torch.no_grad():
        output = model.generate(**inputs)

    caption = processor.decode(output[0], skip_special_tokens=True)
    return caption

# --- Run both tasks ---
ocr_text = perform_ocr(image)
caption_text = generate_caption(image)

# --- Display Results ---
print("\n--- 📝 OCR Result ---")
print(ocr_text if ocr_text else "[No text detected]")

print("\n--- 🖼️ Image Caption ---")
print(caption_text)
```



```
🚀 Using device: cpu

--- 📝 OCR Result ---
If yOu

youre walking,

—Dolly Parton

Prevention

--- 🖼️ Image Caption ---
a quote that says if you don ' t like you ' re, you ' re
```

INFERENCE

The system successfully integrates two tasks: **text detection** from images and **semantic image understanding**. This hybrid approach can be applied to intelligent document readers, accessibility tools, and AI-powered content analyzers.