import cv2  
from PIL import Image  
import pytesseract  
  
# Update path if needed  
pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"  
  
# Path to your image  
image\_path = "ocr\_text.png"  
  
# Read the image  
image = cv2.imread(image\_path)  
  
if image is None:  
 print("Error: Could not load image.")  
else:  
 # Convert to grayscale  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
  
 # Convert to PIL for pytesseract  
 pil\_img = Image.fromarray(gray)  
  
 # Extract text  
 text = pytesseract.image\_to\_string(pil\_img)  
  
 print("Extracted Text:\n", text)  
  
 # Show the image (optional)  
 cv2.imshow("Image", image)  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()

Intallation: tesseract to windows

Go to pycharm create a project :

Follow the installation

pip install pytesseract

pip install opencv-python

pip install pillow

**Step 1: Importing libraries**

import cv2

from PIL import Image

import pytesseract

* cv2 → OpenCV library for computer vision. Used to read, process, and manipulate images.
* PIL.Image → Python Imaging Library (Pillow) to handle images in formats compatible with pytesseract.
* pytesseract → Python wrapper for **Tesseract OCR engine**, which converts images of text into actual text.

**Step 2: Specify Tesseract executable path**

pytesseract.pytesseract.tesseract\_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"

* Tesseract OCR is a separate software; pytesseract needs to know **where Tesseract is installed**.
* r"" → raw string notation, avoids issues with \ in Windows paths.

Without this, pytesseract would fail with "TesseractNotFoundError".

**Step 3: Specify image path**

image\_path = "ocr\_text.png"

* The variable image\_path contains the path to the image you want to process.
* This could be any image file (.png, .jpg, etc.) that contains text.

**Step 4: Read the image**

image = cv2.imread(image\_path)

* cv2.imread() reads the image from disk and stores it as a **NumPy array**.
* Each pixel is represented as [B, G, R] (Blue, Green, Red) in OpenCV.
* image now contains the **raw pixel data**.

**Step 5: Check if the image loaded correctly**

if image is None:

print("Error: Could not load image.")

* Sometimes the image path may be wrong or the file is missing.
* If cv2.imread() fails, it returns None. This check prevents further errors.

**Step 6: Convert to grayscale**

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

* OCR works better on grayscale images.
* cv2.cvtColor() converts the color space:
  + Input: image in BGR format
  + Output: gray → 2D array (pixel intensity 0–255)
* Grayscale reduces noise and simplifies text extraction.

**Step 7: Convert OpenCV image to PIL image**

pil\_img = Image.fromarray(gray)

* pytesseract works with **PIL Image objects**, not NumPy arrays.
* Image.fromarray(gray) converts the OpenCV grayscale array to a PIL image.

**Step 8: Extract text using pytesseract**

text = pytesseract.image\_to\_string(pil\_img)

* image\_to\_string() runs Tesseract OCR on the image.
* Returns a **string containing the text** found in the image.
* You can further pass parameters like lang='eng' or config='--psm 6' to tweak OCR.

**Step 9: Print the extracted text**

print("Extracted Text:\n", text)

* Displays the OCR result on the console.
* Example output:

Extracted Text:

Hello World

This is OCR test.

**Step 10: (Optional) Display the image**

cv2.imshow("Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

* cv2.imshow() → Shows the original image in a window.
* cv2.waitKey(0) → Waits indefinitely until a key is pressed.
* cv2.destroyAllWindows() → Closes the OpenCV window after key press.

This step is purely for visualization and **not necessary for OCR**.

✅ **Summary of the Flow:**

1. Load the image with OpenCV.
2. Convert it to grayscale (simplifies text detection).
3. Convert grayscale image to PIL format.
4. Use pytesseract to read text from the image.
5. Print the text.
6. Optionally, display the image.