📗 Week 2 – Data Collection & Extraction

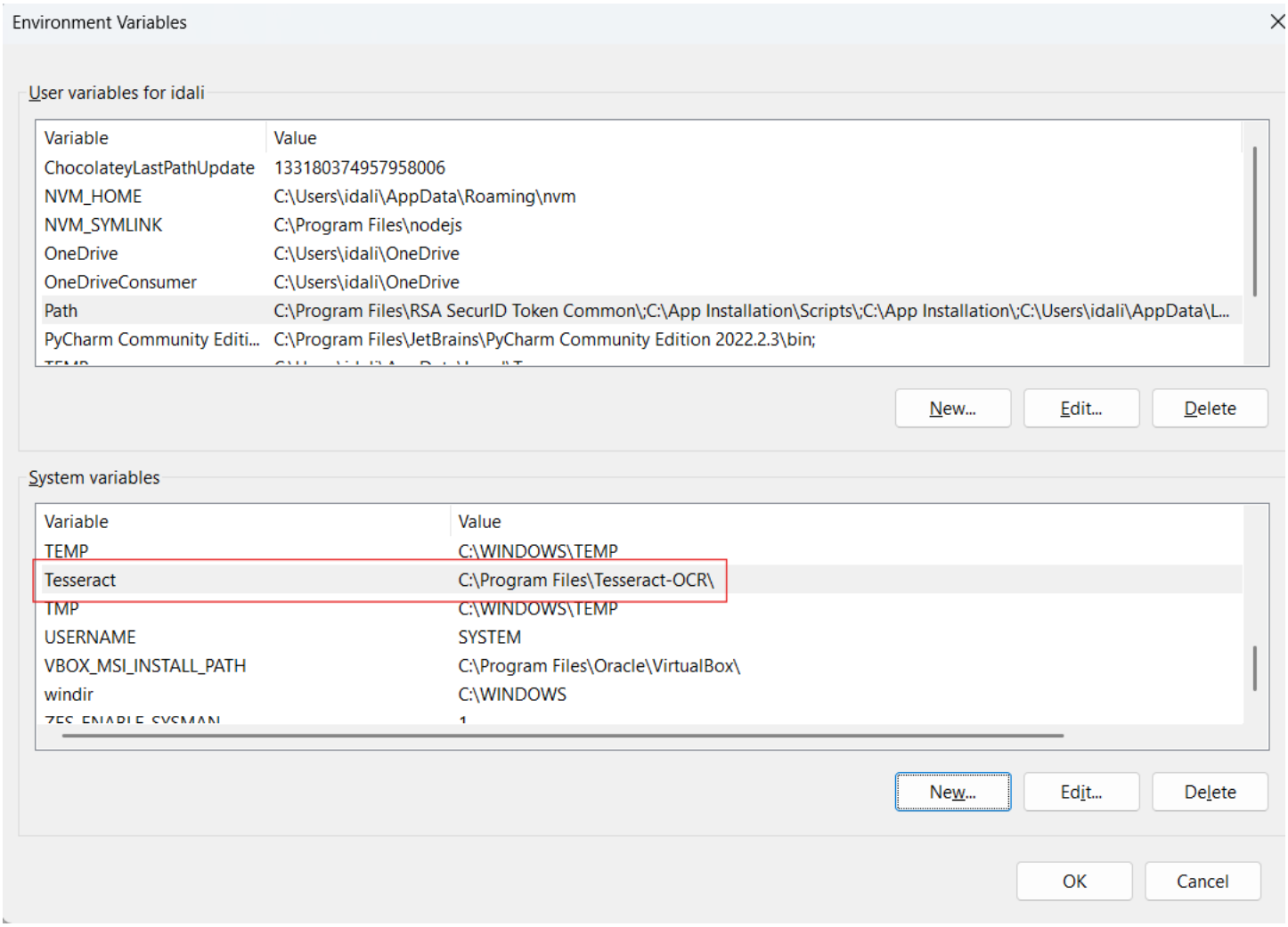
📝 Tesseract OCR Guide

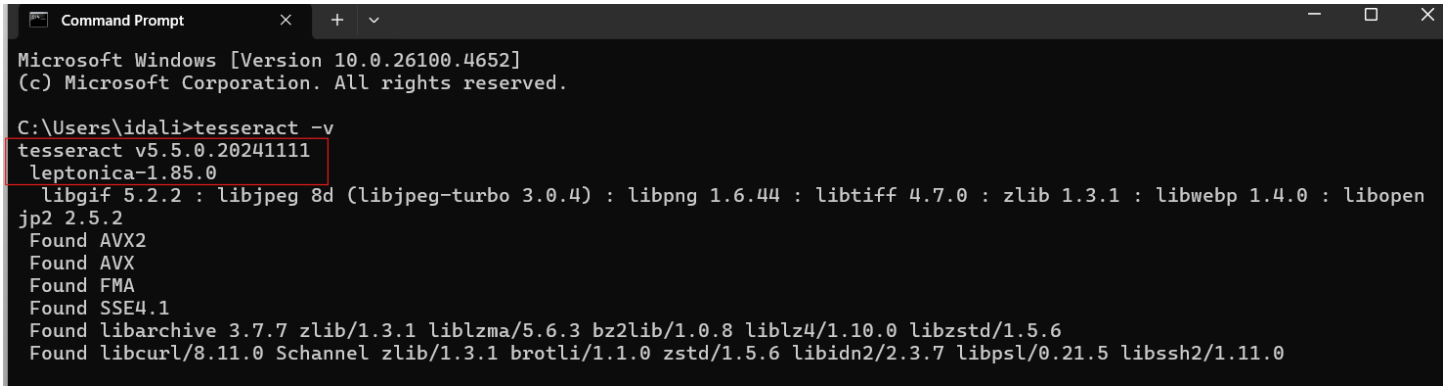
Tesseract is an open-source Optical Character Recognition (OCR) engine developed by Google. It is used to convert scanned images, PDFs, and images with text into machine-readable text. Tesseract supports more than 100 languages, including complex scripts like Arabic, Chinese, and many others.

🔧 Installation

Installing Tesseract on Different Platforms

1. **On macOS** (using Homebrew):
2. brew install tesseract
3. **On Ubuntu/Debian**:
4. sudo apt update
5. sudo apt install tesseract-ocr
6. **On Windows**:
    - Download the Tesseract installer from the Tesseract GitHub releases.
    - Follow the installation instructions, and ensure to add Tesseract to your system's PATH.





Basic Usage

Once Tesseract is installed, you can use it either from the command line or by using the Python wrapper pytesseract.

Command Line Usage

To perform OCR on an image:

tesseract image.png output.txt

Use Tesseract with python

```
from PIL import Image
import pytesseract #pip install pytesseract first

# Load an image using Pillow (PIL)
image = Image.open('image.png')

# Perform OCR on the image
text = pytesseract.image_to_string(image)

print(text)
```

Tesseract supports over 100 languages, and you can even train it for custom languages or fonts. To use a different language, you can download the corresponding trained data files and specify the language in the -l flag.

For example, to use Spanish (spa):

```
tesseract image.png output -l spa
```

👷 Tesseract Best Practices

Preprocess Images: Always preprocess images by converting to grayscale, adjusting brightness/contrast, and removing noise to improve Tesseract's accuracy.

Use Correct --psm: The Page Segmentation Mode (--psm) plays a crucial role in how Tesseract segments the image and interprets the text. Experiment with different modes for complex documents.

Choose the Right Language: Always specify the correct language (-l lang_code) for better accuracy. Tesseract performs poorly when the language is incorrect.

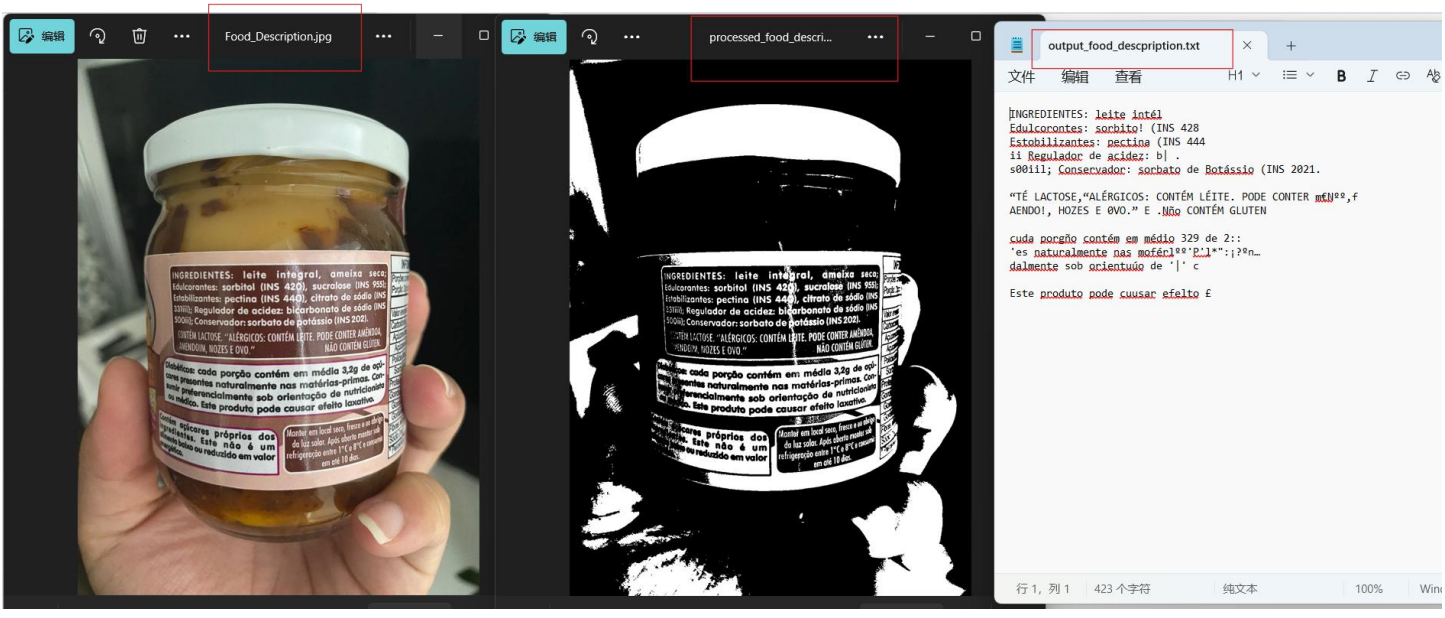
Example1-Preprocess Images：

ocr_with_preprocessing.py

```python
print(f"Image preprocessing completed. Saved as {processed_image}")

# ===== Step 2: OCR Recognition =====
# Open the preprocessed image
image = Image.open(processed_image)

# Perform OCR using Spain
text = pytesseract.image_to_string(image, lang='spa')

# Print the recognized text
print("OCR Result:\n")
print(text)

# ===== Optional: Save the result to a text file =====
with open("output_food_descpription.txt", "w", encoding="utf-8") as f:
    f.write(text)
print("\nText recognition completed. Result saved to output.txt")
```

Problems Output Debug Console Terminal Ports

```
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preprocessing.py
Image preprocessing completed. Saved as processed_food_description.png
OCR Result:

INGREDIENTES: leite intél
Edulcorontes: sorbito! (INS 428
Estobilizantes: pectina (INS 444
ii Regulador de acidez: b| .
s00iil; Conservador: sorbato de Botássio (INS 2021.

“TÉ LACTOSE,“ALÉRGICOS: CONTÉM LÉITE. PODE CONTER m€N⁰⁰,f
AENDO!, HOZES E 0VO.” E .Nño CONTÉM GLUTEN

cuda porgño contém em médio 329 de 2::
'es naturalmente nas moférl⁰⁰'P'l*":¡?ºn…
dalmente sob orientuúo de '|' c

Este produto pode cuusar efelto £


Text recognition completed. Result saved to output.txt
```



Example 2 - Use Correct --psm:

ocr_with_preprocessing.py ×

ocr_with_preprocessing.py > ...

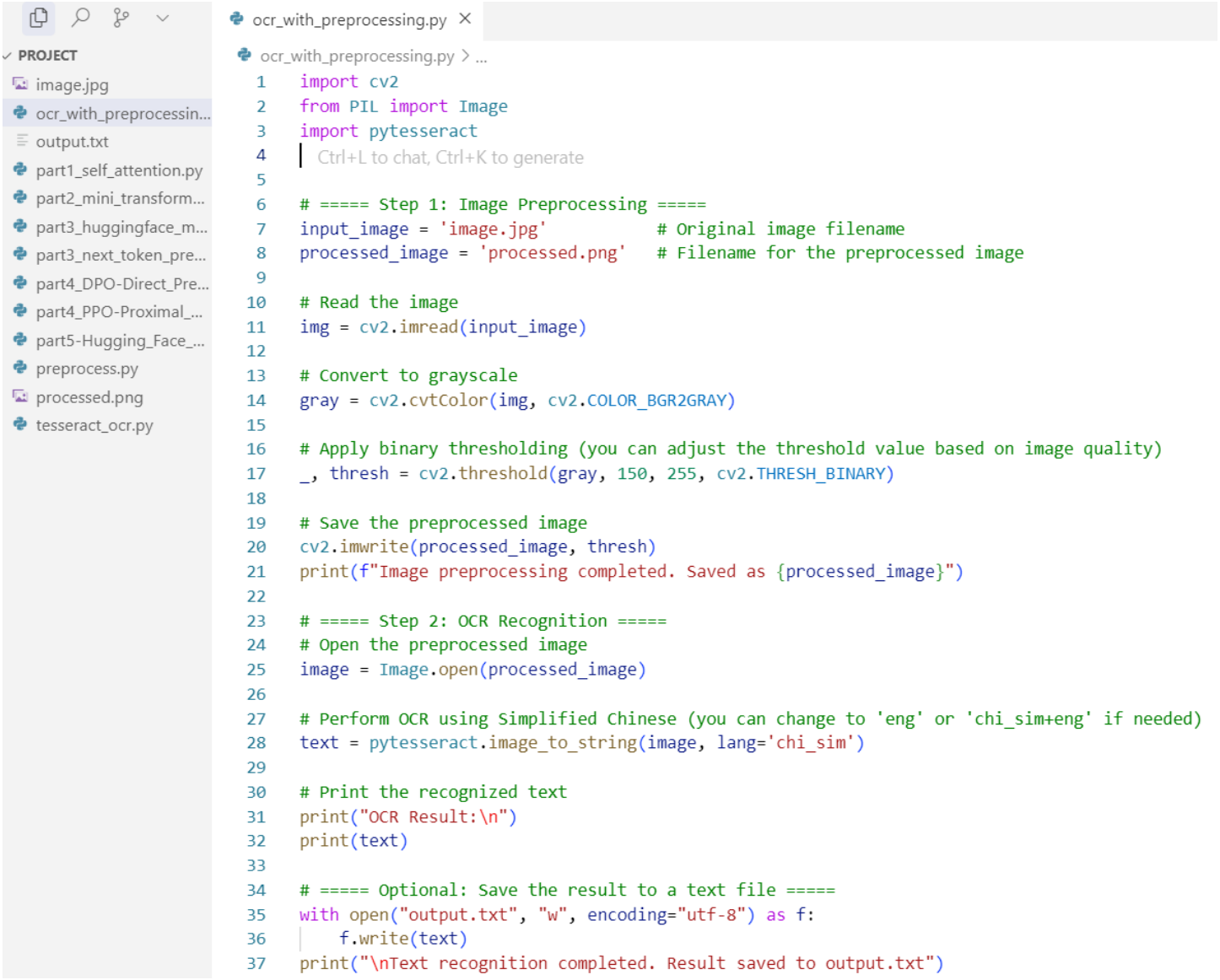
```python
# Apply binary thresholding (you can adjust the threshold value based on image quality)
_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)

# Save the preprocessed image
cv2.imwrite(processed_image, thresh)
print(f"Image preprocessing completed. Saved as {processed_image}")

# ===== Step 2: OCR Recognition =====
# Open the preprocessed image
image = Image.open(processed_image)

# Test several PSM modes
for psm in [3, 6, 7, 11]:
    print(f"\n--- OCR Result with PSM {psm} ---")
    text = pytesseract.image_to_string(image, lang='spa', config=f'--psm {psm}')
    print(text)

    # ===== Optional: Save the result to a text file =====
    with open(f"output_food_descpription_psm{psm}.txt", "w", encoding="utf-8") as f:
        f.write(text)

print("\nText recognition completed. Result saved to output.txt")
```

Problems Output Debug Console Terminal Ports

```
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preproc
essing.py
Image preprocessing completed. Saved as processed_food_description_multi.png

--- OCR Result with PSM 3 ---
INGREDIENTES: leite intél
Edulcorontes: sorbito! (INS 428
Estobilizantes: pectina (INS 444
```

Food_Description.jpg	8/2/2025 9:34 PM
image.jpg	8/2/2025 8:55 PM
ocr_with_preprocessing.py	8/2/2025 11:10 PM
output.txt	8/2/2025 9:22 PM
output_food_descpription.txt	8/2/2025 9:41 PM
output_food_descpription_psm3.txt	8/2/2025 11:11 PM
output_food_descpription_psm6.txt	8/2/2025 11:11 PM
output_food_descpription_psm7.txt	8/2/2025 11:11 PM
output_food_descpription_psm11.txt	8/2/2025 11:11 PM
part1_self_attention.py	7/29/2025 10:07 PM
part2_mini_transformer_block_in_pytorch.py	7/31/2025 7:27 PM
part3_huggingface_mistral_test.py	8/1/2025 8:47 PM
part3_next_token_prediction_using_huggingface.py	8/1/2025 10:27 PM
part4_DPO-Direct_Preference_Optimization.py	8/1/2025 9:27 PM
part4_PPO-Proximal_Policy_Optimization.py	8/1/2025 10:38 PM
part5-Hugging_Face_Transformers.py	8/1/2025 10:38 PM
preprocess.py	7/29/2025 9:10 PM
processed.png	8/2/2025 9:22 PM
processed_food_description.png	8/2/2025 9:41 PM
processed_food_description_multi.png	8/2/2025 11:11 PM
tesseract_ocr.py	7/29/2025 9:11 PM

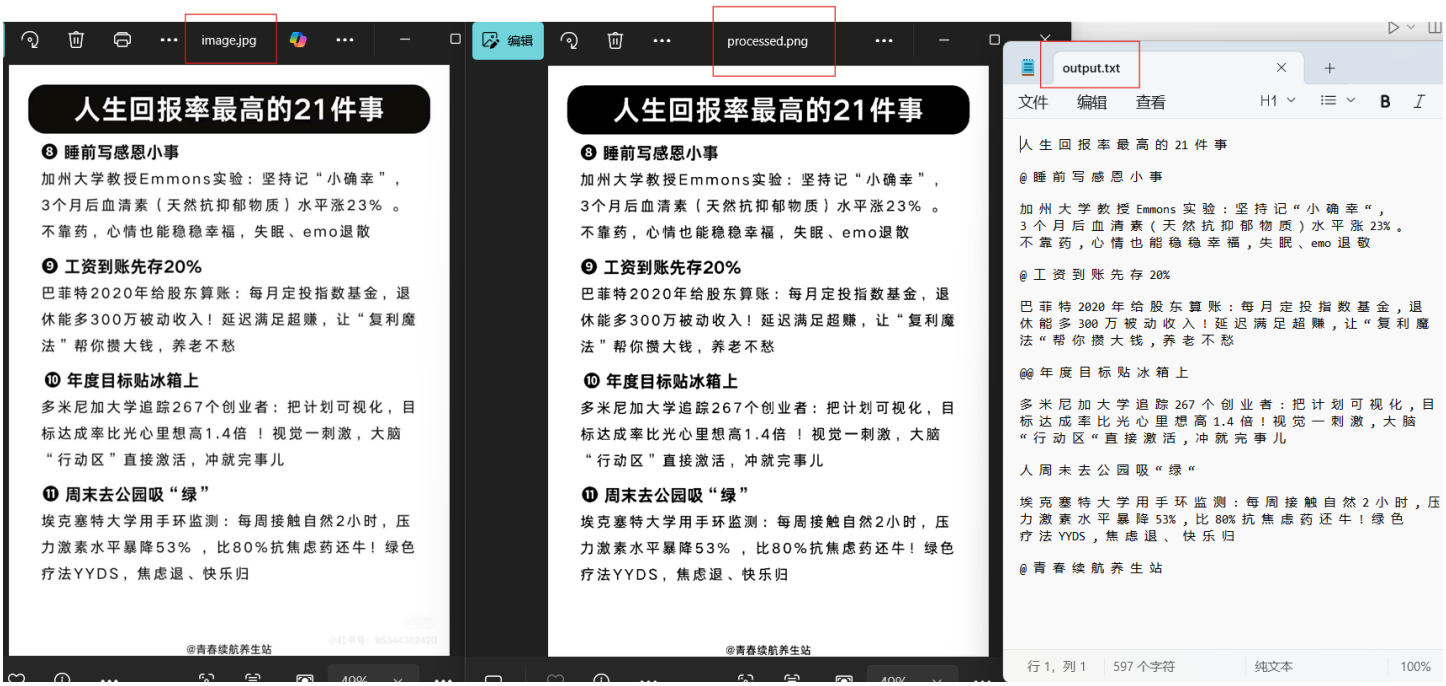
Example 3 - Choose the Right Language: (-l lang_code)




```python
import cv2
from PIL import Image
import pytesseract

# ===== Step 1: Image Preprocessing =====
input_image = 'image.jpg'           # Original image filename
processed_image = 'processed.png'   # Filename for the preprocessed image

# Read the image
img = cv2.imread(input_image)

# Convert to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Apply binary thresholding (you can adjust the threshold value based on image quality)
_, thresh = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY)

# Save the preprocessed image
cv2.imwrite(processed_image, thresh)
print(f"Image preprocessing completed. Saved as {processed_image}")

# ===== Step 2: OCR Recognition =====
# Open the preprocessed image
image = Image.open(processed_image)

# Perform OCR using Simplified Chinese (you can change to 'eng' or 'chi_sim+eng' if needed)
text = pytesseract.image_to_string(image, lang='chi_sim')

# Print the recognized text
print("OCR Result:\n")
print(text)

# ===== Optional: Save the result to a text file =====
with open("output.txt", "w", encoding="utf-8") as f:
    f.write(text)
print("\nText recognition completed. Result saved to output.txt")
```

Problems Output Debug Console Terminal Ports

```
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/ocr_with_preprocessing.py
Image preprocessing completed. Saved as processed.png
OCR Result:

人 生 回 报 率 最 高 的 21 件 事

@睡 前 写 感 恩 小 事

加 州 大 学 教 授 Emmons 实 验 ：坚 持 记 “ 小 确 幸 “ ，
3 个 月 后 血 清 素 （ 天 然 抗 抑 郁 物 质 ） 水 平 涨 23% 。
不 靠 药 ，心 情 也 能 稳 稳 幸 福 ，失 眠 、emo 退 敬

@工 资 到 账 先 存 20%

巴 菲 特 2020 年 给 股 东 算 账 ：每 月 定 投 指 数 基 金 ，退
休 能 多 300 万 被 动 收 入 ！延 迟 满 足 超 赚 ，让 “ 复 利 魔
法 “ 帮 你 攒 大 钱 ，养 老 不 愁

@@ 年 度 目 标 贴 冰 箱 上

多 米 尼 加 大 学 追 踪 267 个 创 业 者 ：把 计 划 可 视 化 ，目
标 达 成 率 比 光 心 里 想 高 1.4 倍 ！视 觉 一 刺 激 ，大 脑
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿
@@ 年 度 目 标 贴 冰 箱 上

多 米 尼 加 大 学 追 踪 267 个 创 业 者 ：把 计 划 可 视 化 ，目
标 达 成 率 比 光 心 里 想 高 1.4 倍 ！视 觉 一 刺 激 ，大 脑
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿

多 米 尼 加 大 学 追 踪 267 个 创 业 者 ：把 计 划 可 视 化 ，目
标 达 成 率 比 光 心 里 想 高 1.4 倍 ！视 觉 一 刺 激 ，大 脑
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿
多 米 尼 加 大 学 追 踪 267 个 创 业 者 ：把 计 划 可 视 化 ，目
标 达 成 率 比 光 心 里 想 高 1.4 倍 ！视 觉 一 刺 激 ，大 脑
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿
标 达 成 率 比 光 心 里 想 高 1.4 倍 ！视 觉 一 刺 激 ，大 脑
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿
“ 行 动 区 “ 直 接 激 活 ，冲 就 完 事 儿


人 周 未 去 公 园 吸 “ 绿 “


埃 克 塞 特 大 学 用 手 环 监 测 ：每 周 接 触 自 然 2 小 时 ，压
力 激 素 水 平 暴 降 53% ，比 80% 抗 焦 虑 药 还 牛 ！绿 色

疗 法 YYDS ，焦 虑 退 、 快 乐 归


@青 春 续 航 养 生 站
@青 春 续 航 养 生 站


Text recognition completed. Result saved to output.txt
PS C:\AI_Coop\Homework\Week2\Project>
```



Custom Training: For specialized fonts or languages, you can train Tesseract to recognize custom fonts or languages. This is especially useful for handwriting or unusual fonts.

```
root@530399ee769c: /train
Microsoft Windows [Version 10.0.26100.4652]
(c) Microsoft Corporation. All rights reserved.

C:\Users\idali>git clone https://github.com/guiem/train-tesseract.git
fatal: destination path 'train-tesseract' already exists and is not an empty directory.

C:\Users\idali>cd train-tesseract

C:\Users\idali\train-tesseract>docker build -t my-tesseract-trainer .
[+] Building 0.8s (14/14) FINISHED        docker:desktop-linux
 => [internal] load build definition from Dockerfile      0.0s
 => => transferring dockerfile: 1.59kB                    0.0s
 => [internal] load metadata for docker.io/library/ubunt  0.5s
 => [auth] library/ubuntu:pull token for registry-1.dock  0.0s
 => [internal] load .dockerignore                         0.0s
 => => transferring context: 2B                           0.0s
 => [1/8] FROM docker.io/library/ubuntu:18.04@sha256:152  0.0s
 => => resolve docker.io/library/ubuntu:18.04@sha256:152  0.0s
 => [internal] load build context                         0.0s
 => => transferring context: 38B                          0.0s
 => CACHED [2/8] RUN apt-get update &&     apt-get insta  0.0s
 => CACHED [3/8] RUN apt-get install -y --reinstall make  0.0s
 => CACHED [4/8] WORKDIR /app                             0.0s
 => CACHED [5/8] COPY requirements.txt ./                 0.0s
 => CACHED [6/8] RUN mkdir src && cd /app/src &&     wge  0.0s
 => CACHED [7/8] RUN pip3 install -r requirements.txt     0.0s
 => CACHED [8/8] RUN apt-get install -y locales && local  0.0s
 => exporting to image                                    0.1s
 => => exporting layers                                   0.0s
 => => exporting manifest sha256:c81050894bc86d1794f9321  0.0s
 => => exporting config sha256:41a0f309cebc730f40c2df799  0.0s
 => => exporting attestation manifest sha256:22d33f70976  0.0s
 => => exporting manifest list sha256:06d550f01bd4fb891c  0.0s
 => => naming to docker.io/library/my-tesseract-trainer:  0.0s
 => => unpacking to docker.io/library/my-tesseract-train  0.0s

 1 warning found (use docker --debug to expand):
 - LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 5)
```

```
C:\Users\idali\train-tesseract>docker run -it -v C:\AI_Coop\Homework\Week2\Project\my_handwriting_train:/train my-tesseract-trainer bash
root@530399ee769c:/app# cd /train
root@530399ee769c:/train# ls
my_handwriting_sample.jpg  training_text.txt
root@530399ee769c:/train#
```

```
root@530399ee769c:/train# convert my_handwriting_sample.jpg my_handwriting_sample.tif
bash: convert: command not found
root@530399ee769c:/train# apt update && apt install -y imagemagick
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-droid-fallback fonts-noto-mono ghostscript gsfonts
  hicolor-icon-theme imagemagick-6-common imagemagick-6.q16
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2
  libcupsfilters1 libcupsimage2 libdjvulibre-text libdjvulibre21
  libfftw3-double3 libgs9 libgs9-common libidn11 libijs-0.35
  libilmbase12 libjbig2dec0 liblcms2-2 liblqr-1-0
  libmagickcore-6.q16-3 libmagickcore-6.q16-3-extra
  libmagickwand-6.q16-3 libnetpbm10 libopenexr22 libpaper-utils
  libpaper1 libwmf0.2-7 netpbm poppler-data
Suggested packages:
  fonts-noto ghostscript-x imagemagick-doc autotrace cups-bsd
  | lpr | lprng curl enscript ffmpeg gimp gnuplot grads graphviz
  hp2xx html2ps libwmf-bin mplayer povray radiance sane-utils
  texlive-base-bin transfig ufraw-batch xdg-utils cups-common
  libfftw3-bin libfftw3-dev liblcms2-utils inkscape libjxr-tools
  libwmf0.2-7-gtk poppler-utils fonts-japanese-mincho
  | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming
```

```
root@530399ee769c:/train# convert my_handwriting_sample.jpg my_handwriting_sample.tif
```

```
root@530399ee769c:/train# apt update && apt install -y imagemagick
Hit:1 http://security.ubuntu.com/ubuntu bionic-security InRelease
Hit:2 http://archive.ubuntu.com/ubuntu bionic InRelease
Hit:3 http://archive.ubuntu.com/ubuntu bionic-updates InRelease
Hit:4 http://archive.ubuntu.com/ubuntu bionic-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
1 package can be upgraded. Run 'apt list --upgradable' to see it.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fonts-droid-fallback fonts-noto-mono ghostscript gsfonts
  hicolor-icon-theme imagemagick-6-common imagemagick-6.q16
  libavahi-client3 libavahi-common-data libavahi-common3 libcups2
  libcupsfilters1 libcupsimage2 libdjvulibre-text libdjvulibre21
  libfftw3-double3 libgs9 libgs9-common libidn11 libijs-0.35
  libilmbase12 libjbig2dec0 liblcms2-2 liblqr-1-0
  libmagickcore-6.q16-3 libmagickcore-6.q16-3-extra
  libmagickwand-6.q16-3 libnetpbm10 libopenexr22 libpaper-utils
  libpaper1 libwmf0.2-7 netpbm poppler-data
Suggested packages:
  fonts-noto ghostscript-x imagemagick-doc autotrace cups-bsd
  | lpr | lprng curl enscript ffmpeg gimp gnuplot grads graphviz
  hp2xx html2ps libwmf-bin mplayer povray radiance sane-utils
  texlive-base-bin transfig ufraw-batch xdg-utils cups-common
  libfftw3-bin libfftw3-dev liblcms2-utils inkscape libjxr-tools
  libwmf0.2-7-gtk poppler-utils fonts-japanese-mincho
  | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming
  fonts-nanum
The following NEW packages will be installed:
  fonts-droid-fallback fonts-noto-mono ghostscript gsfonts
  hicolor-icon-theme imagemagick imagemagick-6-common
  imagemagick-6.q16 libavahi-client3 libavahi-common-data
  libavahi-common3 libcups2 libcupsfilters1 libcupsimage2
  libdjvulibre-text libdjvulibre21 libfftw3-double3 libgs9
```

cd train-tesseract

docker build -t my-tesseract-trainer .

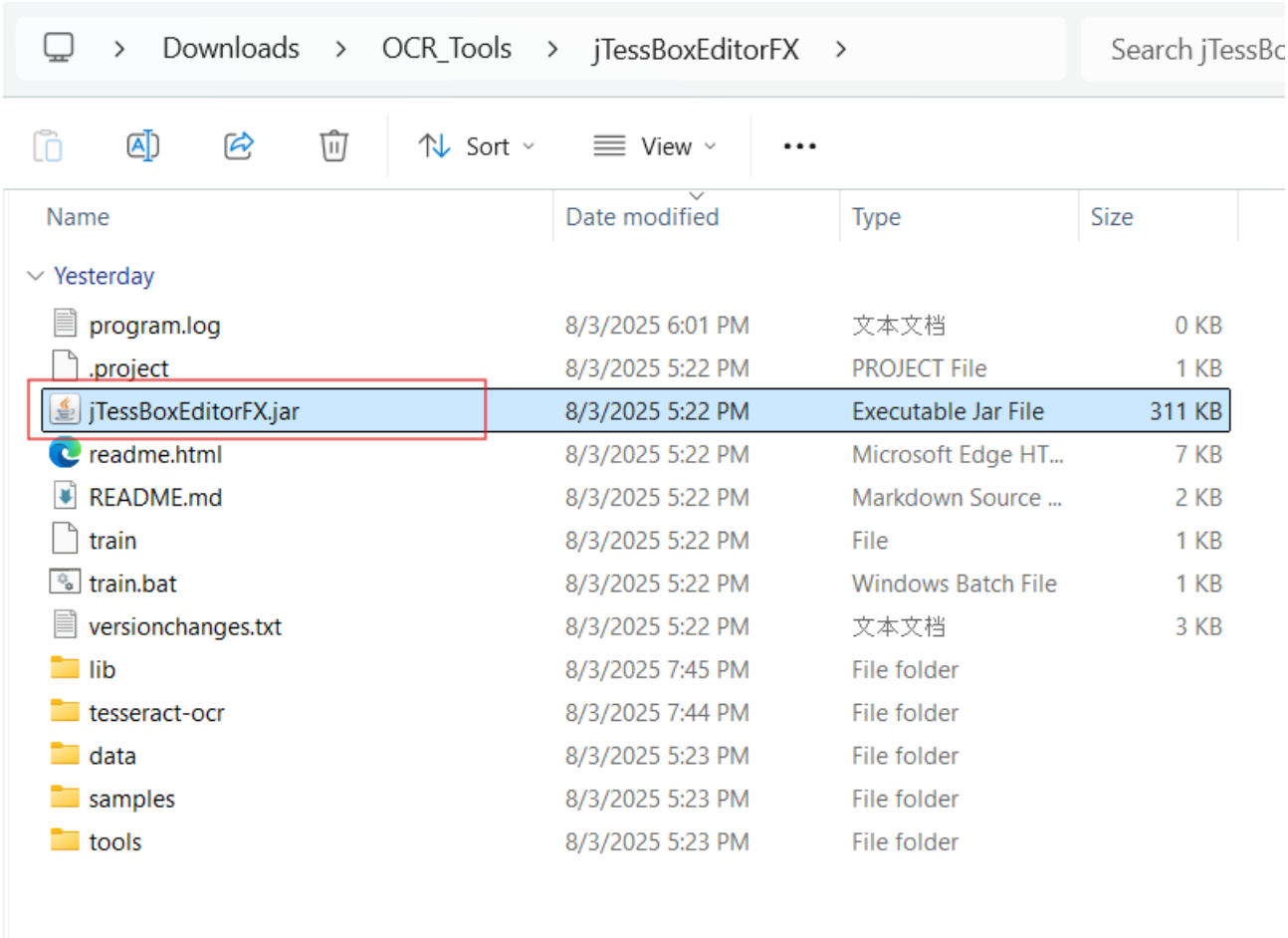
docker run -it -v C:\AI_Coop\Homework\Week2\Project\my_handwriting_train:/train my-tesseract-trainer bash

cd /train

ls

convert my_handwriting_sample.jpg my_handwriting_sample.tif

apt update && apt install -y imagemagick



I cannot open this jTessBoxEditorFX.jar to adjust the sample handwriting document and .box document and I got stuck.

Bonus Hands-on Assignment Sheet

here is bonus part of our homework,which focuses on various aspects of data collection, extraction, and cleaning using OCR (Optical Character Recognition) technology like Tesseract, as well as other techniques such as Web Scraping and Automatic Speech Recognition (ASR). The goal is to apply different tools and methods to extract useful information from web pages, PDFs, audio files, and clean the data for further analysis.

Task Overview

#	💡 Module / Skill	🎯 Task Goal	🛠️ Core Tools	📌 Deliverables
1	🌐 *Web Scraping &*	**arXiv Paper Abstract Scraper** • Query any subcategory	trafilatura, pytesseract,	arxiv_clean.json (≤1MB) + scraper script

#	💡 Module / Skill	🎯 Task Goal	🛠️ Core Tools	📌 Deliverables
	HTML Cleaning	(e.g., *cs.CL*) to fetch the latest 200 papers. • Scrape the /abs/ page and use **Trafilatura** to clean the content. • Use **Tesseract OCR** to extract abstract text from screenshots of the downloaded pages. • Save the data as JSON: {url, title, abstract, authors, date}		

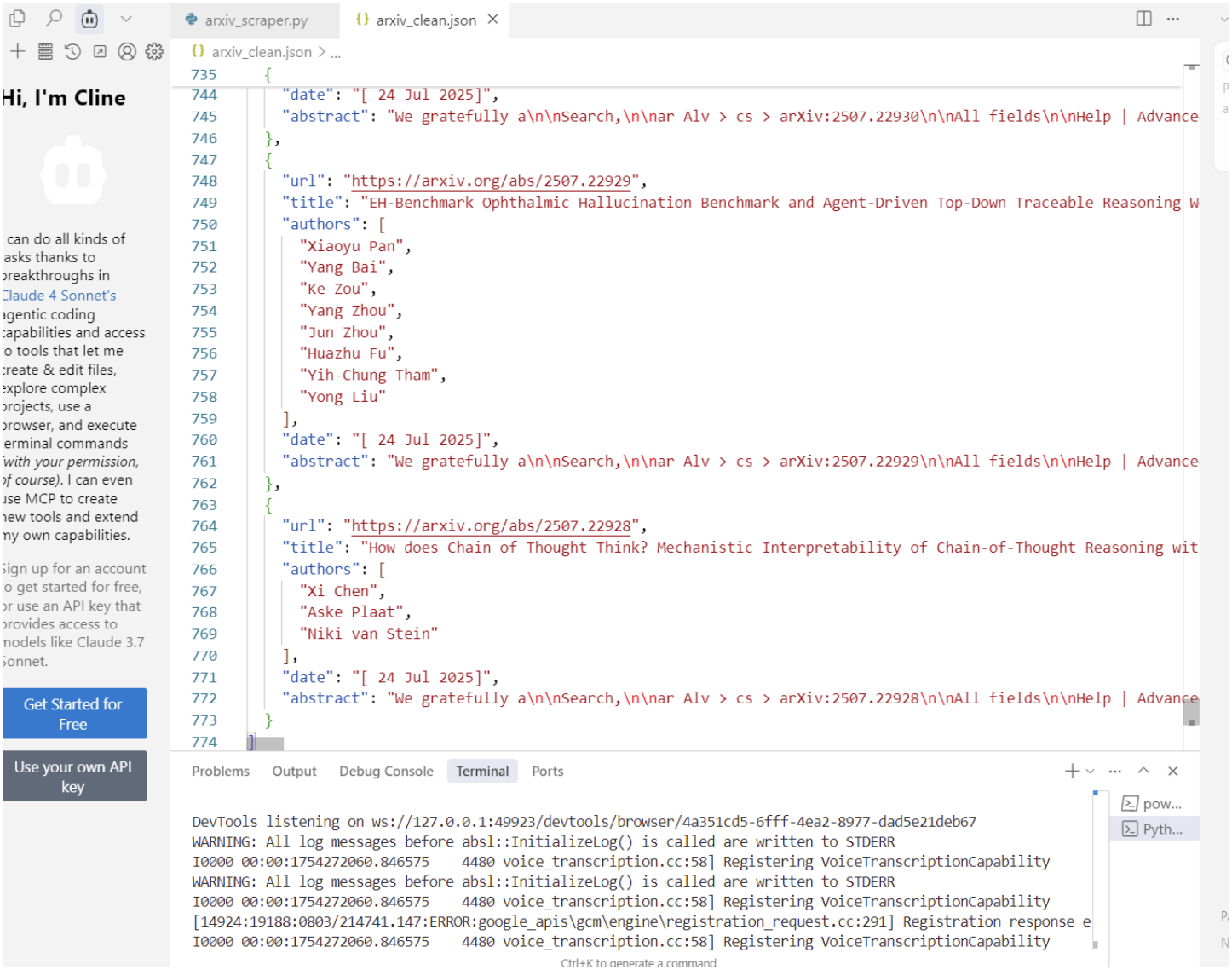
arxiv_scraper.py

```python
# arxiv_scraper.py

from bs4 import BeautifulSoup
import requests
import trafilatura
from PIL import Image
import pytesseract
import time
import json
from selenium import webdriver

# Step 1: 获取 /abs/ 页面链接
category = "cs.CL"
url = f"https://arxiv.org/list/{category}/recent"
resp = requests.get(url)
soup = BeautifulSoup(resp.text, "html.parser")

abs_links = ["https://arxiv.org" + a['href'] for a in soup.select("dt a[href^='/abs/']")]
abs_links = list(dict.fromkeys(abs_links))  # 去重
abs_links = abs_links[:200]  # 限制最多200个

# Step 2: 提取结构化信息 + 用 Trafilatura 清洗摘要（备用）
all_data = []
```

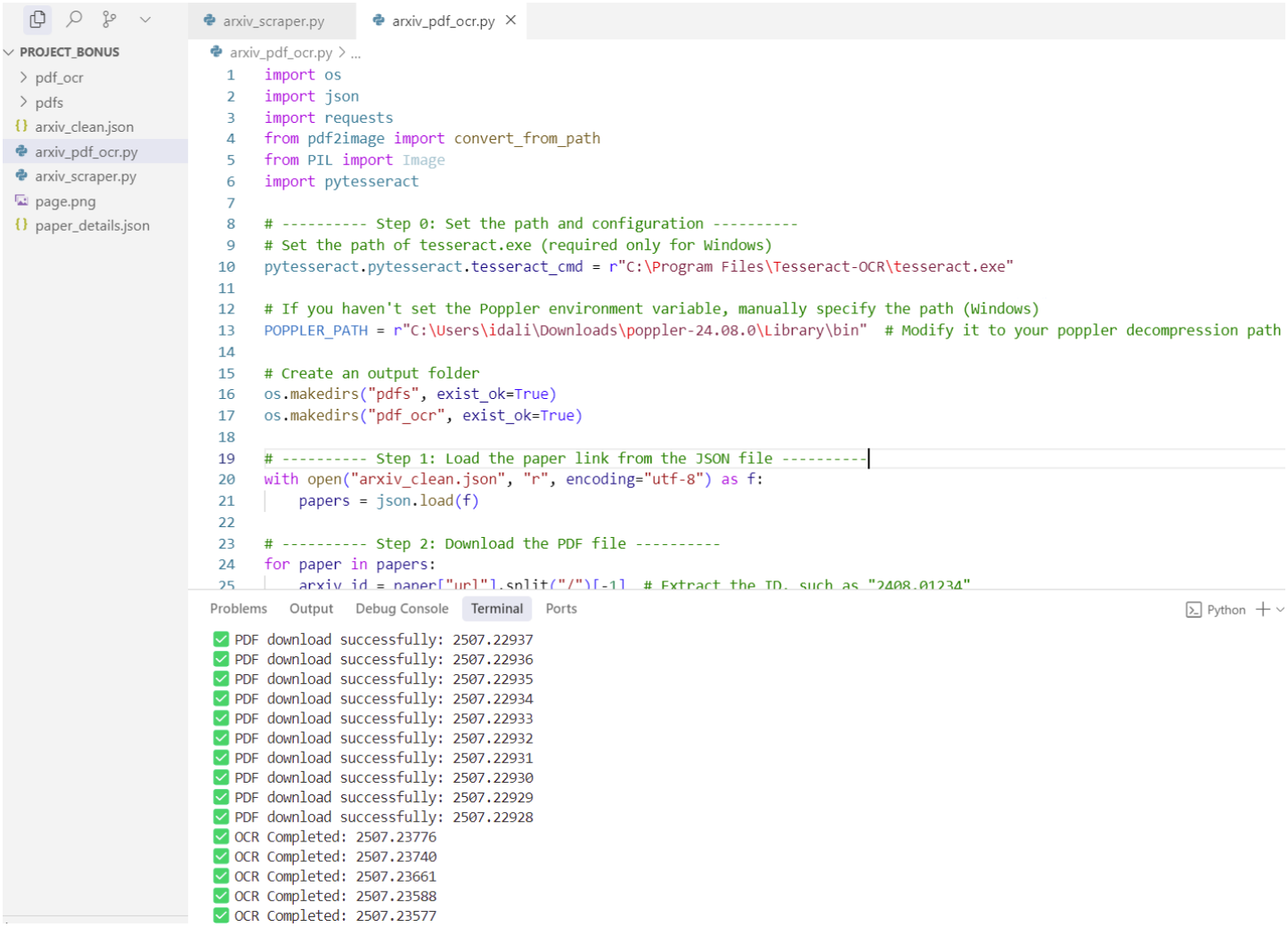
Problems Output Debug Console Terminal Ports

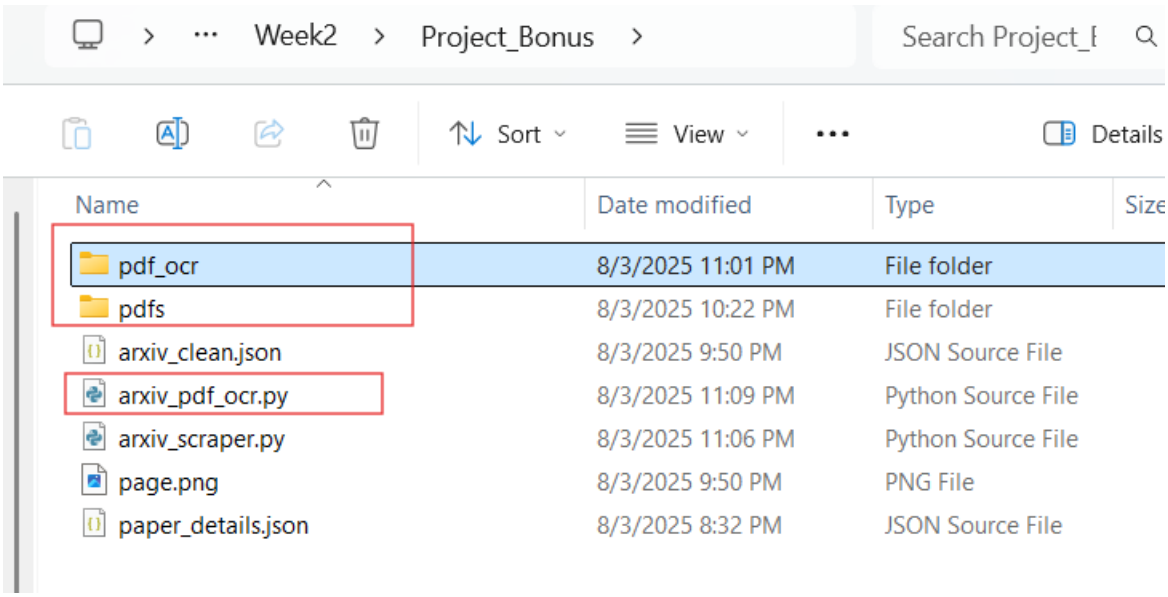
```
I0000 00:00:1754268252.152370   30428 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
[16992:30632:0803/204412.471:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response e
rror message: DEPRECATED_ENDPOINT

DevTools listening on ws://127.0.0.1:52668/devtools/browser/7c6946a5-e8f1-45c5-a8eb-62495dc9c80a
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1754268281.109236   31348 voice_transcription.cc:58] Registering VoiceTranscriptionCapability
[31252:31360:0803/204441.406:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response e
rror message: DEPRECATED_ENDPOINT
[31252:31360:0803/204441.476:ERROR:google_apis\gcm\engine\mcs_client.cc:700]   Error code: 401  Error message:
 Authentication Failed: wrong_secret
[31252:31360:0803/204441.476:ERROR:google_apis\gcm\engine\mcs_client.cc:702] Failed to log in to GCM, resettin
g connection.
[31252:31360:0803/204502.798:ERROR:google_apis\gcm\engine\registration_request.cc:291] Registration response e
rror message: DEPRECATED_ENDPOINT
Created TensorFlow Lite XNNPACK delegate for CPU.
Attempting to use a delegate that only supports static-sized tensors with a graph that has dynamic-sized tenso
```

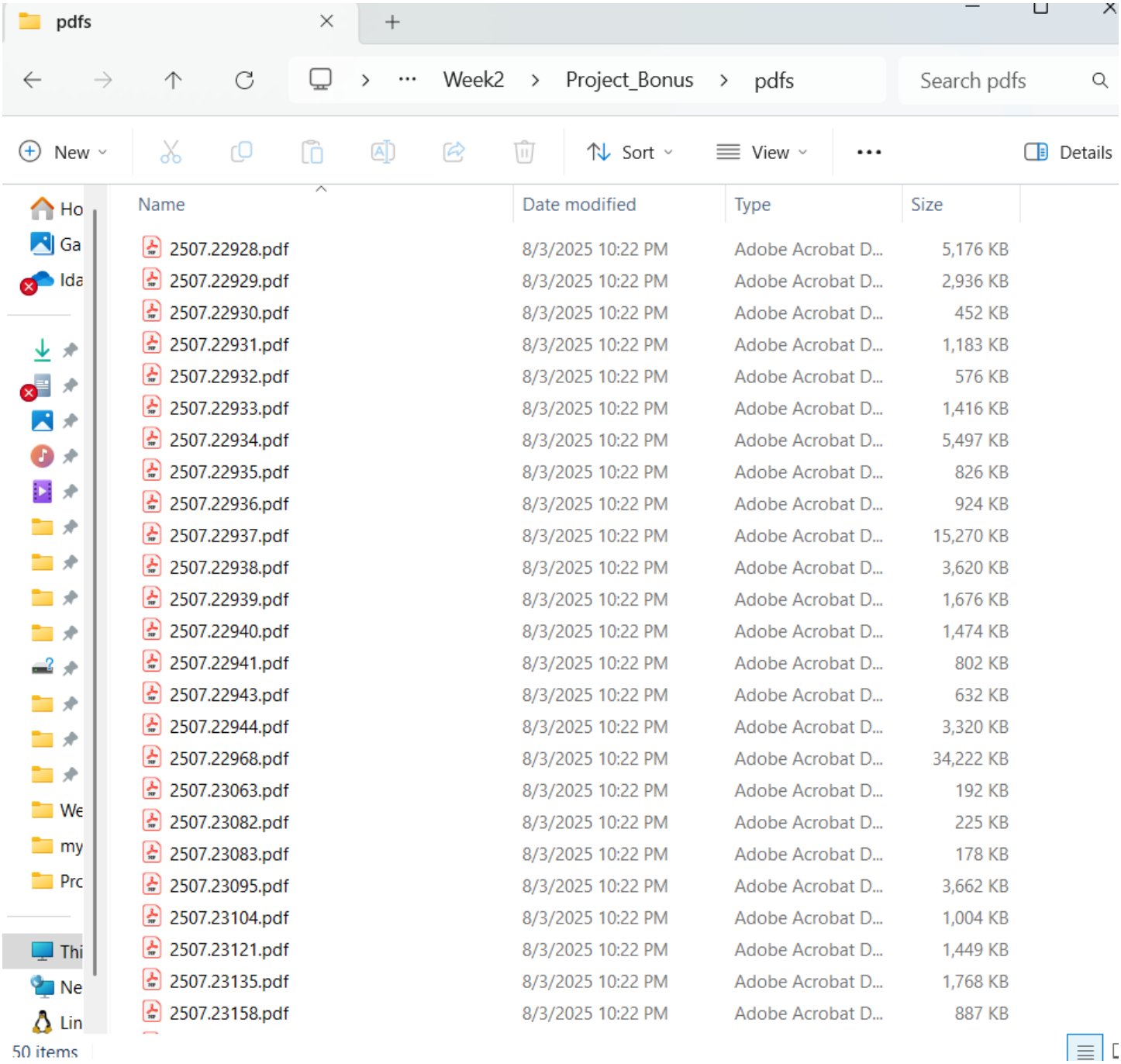
pow...
Pyth...

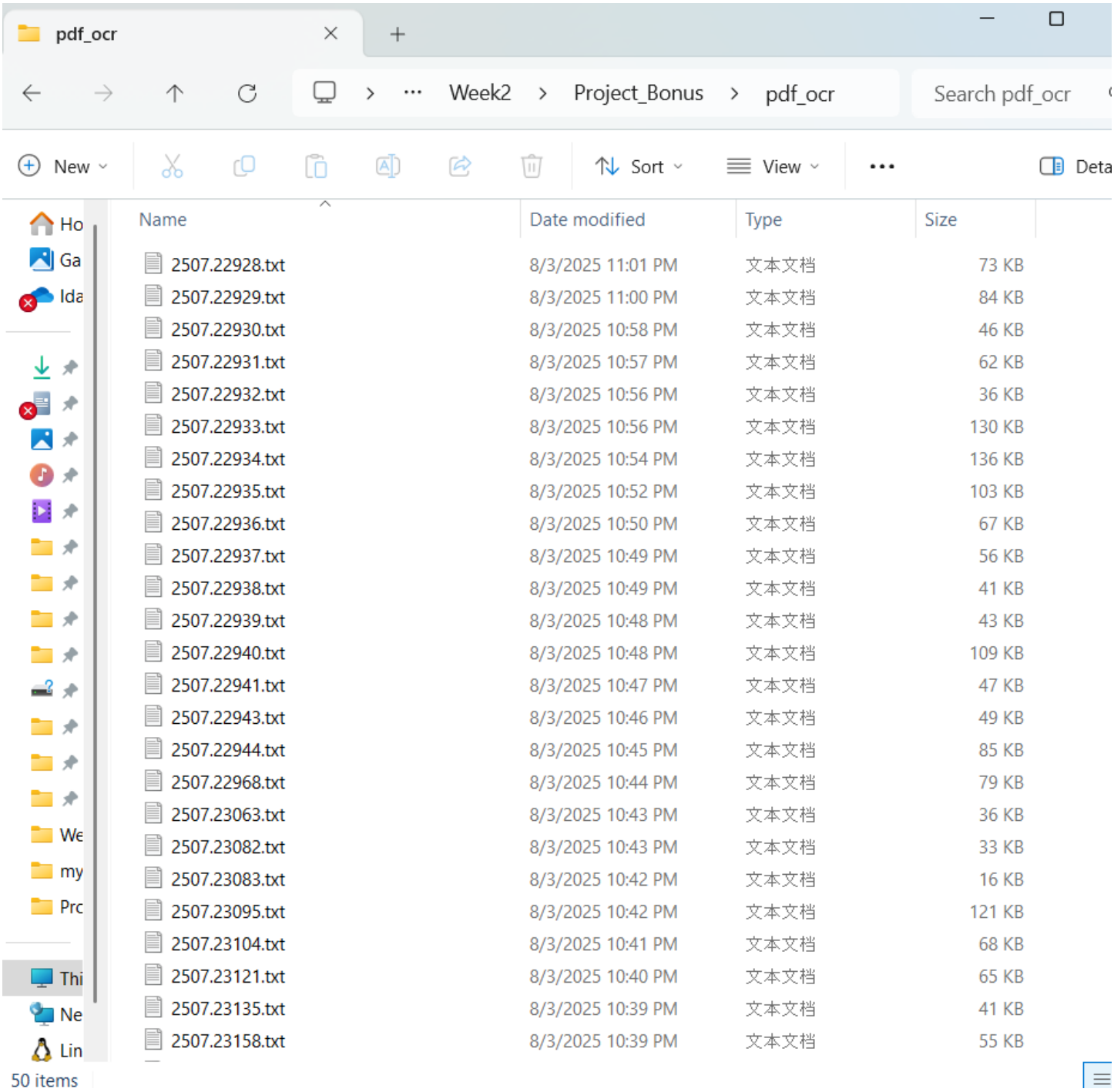


2	🖼️ *PDF to Text OCR*	**Batch OCR for arXiv PDFs** (same paper set as Task 1). • Use **Tesseract** to convert PDFs to text. • Retain OCR layout (e.g., titles, sections) if needed.	pytesseract, pdf2image	pdf_ocr/ folder with TXT files + code notebook


PROJECT_BONUS
pdf_ocr
pdfs
arxiv_clean.json
arxiv_pdf_ocr.py
arxiv_scraper.py
page.png
paper_details.json
arxiv_scraper.py
arxiv_pdf_ocr.py
arxiv_pdf_ocr.py > ...
1 import os
2 import json
3 import requests
4 from pdf2image import convert_from_path
5 from PIL import Image
6 import pytesseract
7
8 # ---------- Step 0: Set the path and configuration ----------
9 # Set the path of tesseract.exe (required only for Windows)
10 pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
11
12 # If you haven't set the Poppler environment variable, manually specify the path (Windows)
13 POPPLER_PATH = r"C:\Users\idali\Downloads\poppler-24.08.0\Library\bin" # Modify it to your poppler decompression path
14
15 # Create an output folder
16 os.makedirs("pdfs", exist_ok=True)
17 os.makedirs("pdf_ocr", exist_ok=True)
18
19 # ---------- Step 1: Load the paper link from the JSON file ----------
20 with open("arxiv_clean.json", "r", encoding="utf-8") as f:
21   papers = json.load(f)
22
23 # ---------- Step 2: Download the PDF file ----------
24 for paper in papers:
25   arxiv_id = paper["url"].split("/")[-1] # Extract the ID, such as "2408.01234"
Problems
Output
Debug Console
Terminal
Ports
Python
PDF download successfully: 2507.22937
PDF download successfully: 2507.22936
PDF download successfully: 2507.22935
PDF download successfully: 2507.22934
PDF download successfully: 2507.22933
PDF download successfully: 2507.22932
PDF download successfully: 2507.22931
PDF download successfully: 2507.22930
PDF download successfully: 2507.22929
PDF download successfully: 2507.22928
OCR Completed: 2507.23776
OCR Completed: 2507.23740
OCR Completed: 2507.23661
OCR Completed: 2507.23588
OCR Completed: 2507.23577

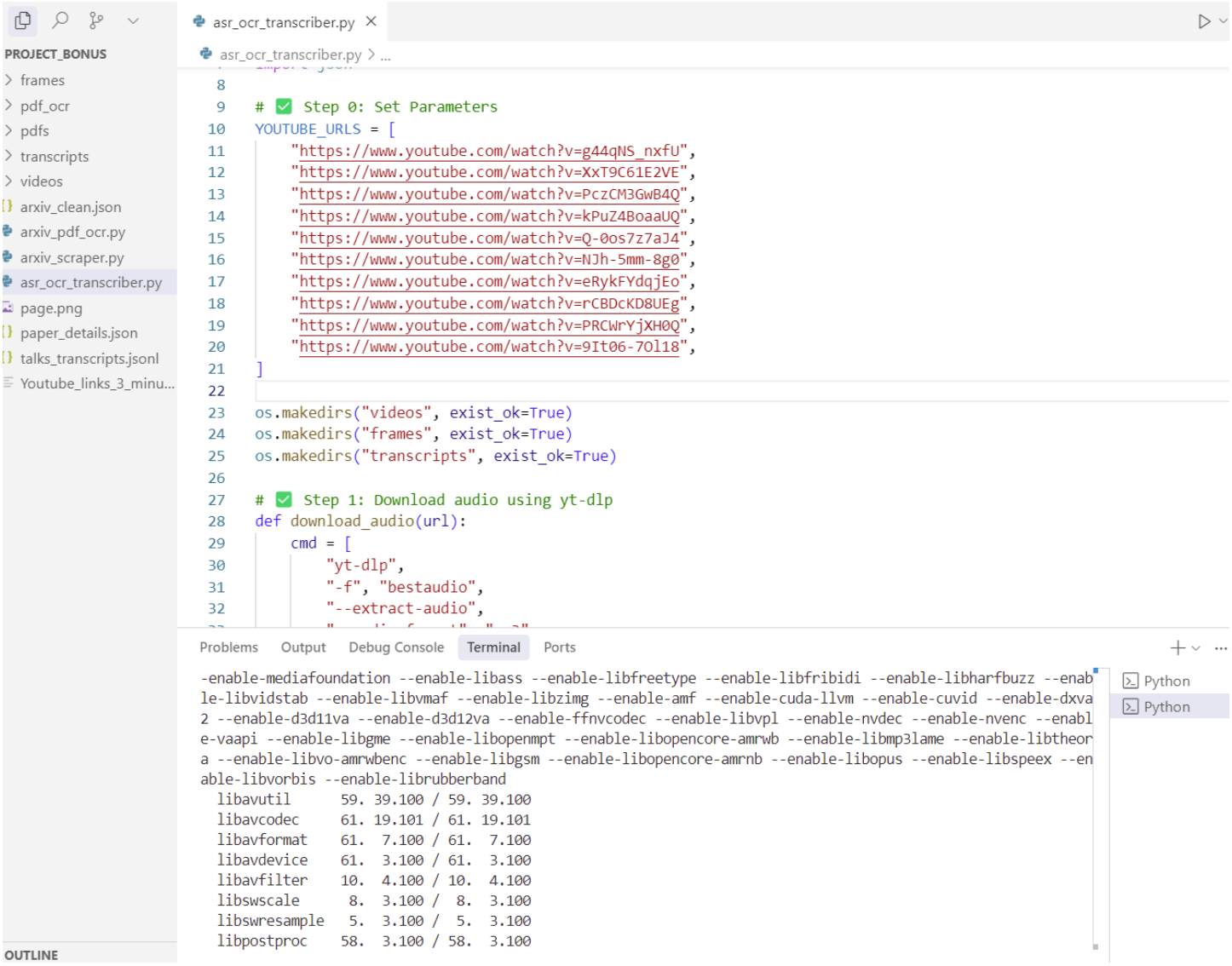

Week2
Project_Bonus
Search Project_I
Sort
View
Details
Name
Date modified
Type
Size
pdf_ocr
8/3/2025 11:01 PM
File folder
pdfs
8/3/2025 10:22 PM
File folder
arxiv_clean.json
8/3/2025 9:50 PM
JSON Source File
arxiv_pdf_ocr.py
8/3/2025 11:09 PM
Python Source File
arxiv_scraper.py
8/3/2025 11:06 PM
Python Source File
page.png
8/3/2025 9:50 PM
PNG File
paper_details.json
8/3/2025 8:32 PM
JSON Source File

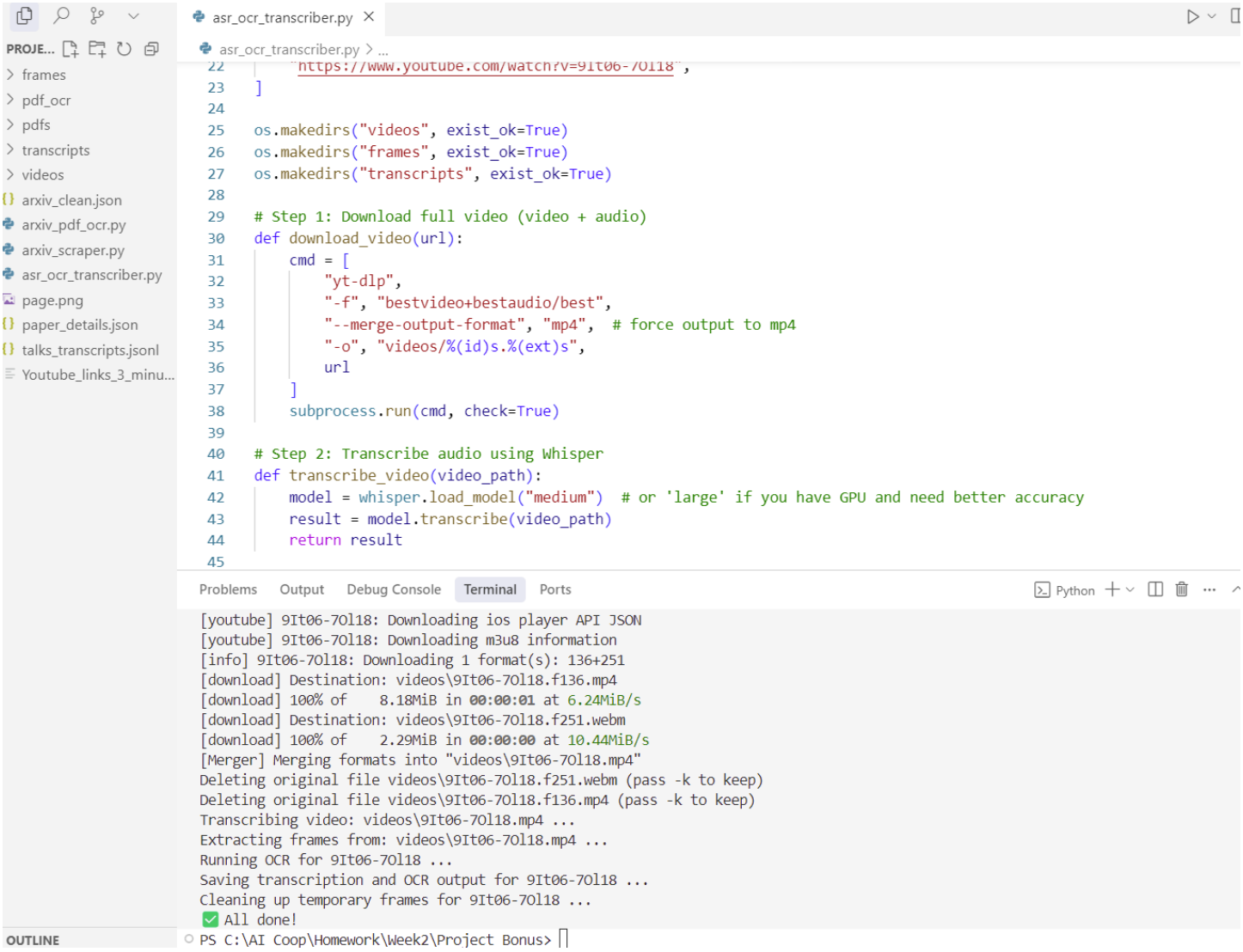

pdfs
Week2
Project_Bonus
pdfs
Search pdfs
New
Sort
View
Details
Name
Date modified
Type
Size
2507.22928.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 5,176 KB
2507.22929.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 2,936 KB
2507.22930.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 452 KB
2507.22931.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,183 KB
2507.22932.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 576 KB
2507.22933.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,416 KB
2507.22934.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 5,497 KB
2507.22935.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 826 KB
2507.22936.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 924 KB
2507.22937.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 15,270 KB
2507.22938.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 3,620 KB
2507.22939.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,676 KB
2507.22940.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,474 KB
2507.22941.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 802 KB
2507.22943.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 632 KB
2507.22944.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 3,320 KB
2507.22968.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 34,222 KB
2507.23063.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 192 KB
2507.23082.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 225 KB
2507.23083.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 178 KB
2507.23095.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 3,662 KB
2507.23104.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,004 KB
2507.23121.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,449 KB
2507.23135.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 1,768 KB
2507.23158.pdf 8/3/2025 10:22 PM Adobe Acrobat D... 887 KB
50 items



3	*Automatic Speech Recognition (ASR)*	**Whisper Transcription Bot** for 10 short NLP conference talks (~3 minutes each). • Use **yt-dl** to fetch YouTube audio. • Transcribe with **Tesseract** for any OCR-based text in the transcript	yt-dlp, pytesseract	talks_transcripts.jsonl + transcription script

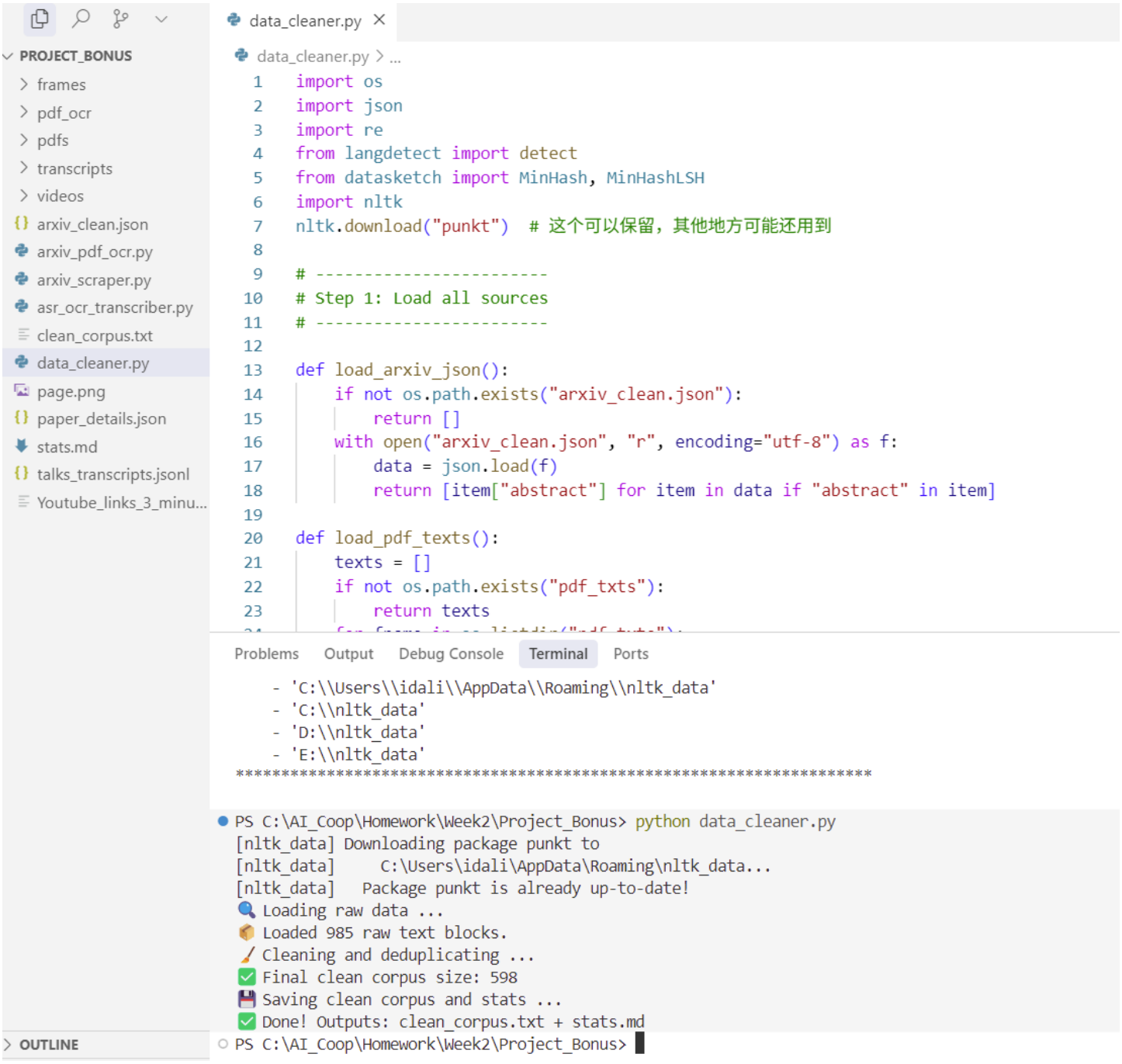
images.
• Save .jsonl with timestamps.





4	*Data Cleaning & Deduplication*	**End-to-End Cleaner**: • Merge the outputs from Tasks 1-3 into one dataset. • Steps: language detection → strip HTML noise → use MinHash for deduplication (similarity ≥ 0.7) → remove PII (emails, credit card numbers, phone numbers) →	langdetect, datasketch	clean_corpus.txt + stats.md (token count, removal percentage)

remove repetitive
n-grams.



class_2_lecture

Part 1: Attention Mechanism (Self-Attention)

part1_attention_mechanism.py

C: > AI_Coop > Homework > Week2 > Project > part1_attention_mechanism.py > self_attention

```python
import numpy as np

# Random Q, K, V matrices
def generate_random_qkv(seq_len=4, d_model=8):
    return [np.random.rand(seq_len, d_model) for _ in range(3)]

# Scaled dot-product attention
def self_attention(Q, K, V):
    d_k = Q.shape[-1]
    scores = np.dot(Q, K.T) / np.sqrt(d_k)
    weights = softmax(scores)
    output = np.dot(weights, V)
    return output, weights

def softmax(x):
    exp_x = np.exp(x - np.max(x, axis=-1, keepdims=True))
    return exp_x / np.sum(exp_x, axis=-1, keepdims=True)

Q, K, V = generate_random_qkv()
out, attn_weights = self_attention(Q, K, V)
print("Attention Output:\n", out)
print("Attention Weights:\n", attn_weights)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\idali> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/part1_attention_mechanism.py
Attention Output:
 [[0.51834168 0.36705926 0.42507334 0.37301351 0.54690054 0.42639484
  0.6993746  0.80091463]
 [0.51944028 0.36302342 0.41606231 0.36579824 0.54249989 0.42096458
  0.70112831 0.79805703]
 [0.51580608 0.35286444 0.40977176 0.3749141  0.54439827 0.42541188
  0.69883605 0.8045721 ]
 [0.51727463 0.36269226 0.41930202 0.37298043 0.54056164 0.42292799
  0.69904642 0.80130066]]

Attention Weights:
 [[0.27602408 0.25294813 0.20237117 0.26865662]
 [0.2902997  0.24662881 0.19095182 0.27211967]
 [0.29429571 0.25946855 0.19451864 0.2517171 ]
 [0.28984034 0.24785633 0.20092363 0.26137971]]
PS C:\Users\idali>
```

Part 2: Mini Transformer Block in PyTorch

part2_mini_transformer_block_in_pytorch.py

part2_mini_transformer_block_in_pytorch.py > ...

```python
import torch
import torch.nn as nn

class MiniTransformerBlock(nn.Module):
    def __init__(self, embed_dim):
        super().__init__()
        self.attn = nn.MultiheadAttention(embed_dim, num_heads=2, batch_first=True)
        self.ffn = nn.Sequential(
            nn.Linear(embed_dim, embed_dim * 4),
            nn.ReLU(),
            nn.Linear(embed_dim * 4, embed_dim)
        )
        self.norm1 = nn.LayerNorm(embed_dim)
        self.norm2 = nn.LayerNorm(embed_dim)

    def forward(self, x):
        attn_output, _ = self.attn(x, x, x)
        x = self.norm1(x + attn_output)
        ffn_output = self.ffn(x)
        x = self.norm2(x + ffn_output)
        return x

x = torch.randn(1, 5, 16)  # batch_size=1, seq_len=5, embed_dim=16
```

Problems Output Debug Console Terminal Ports

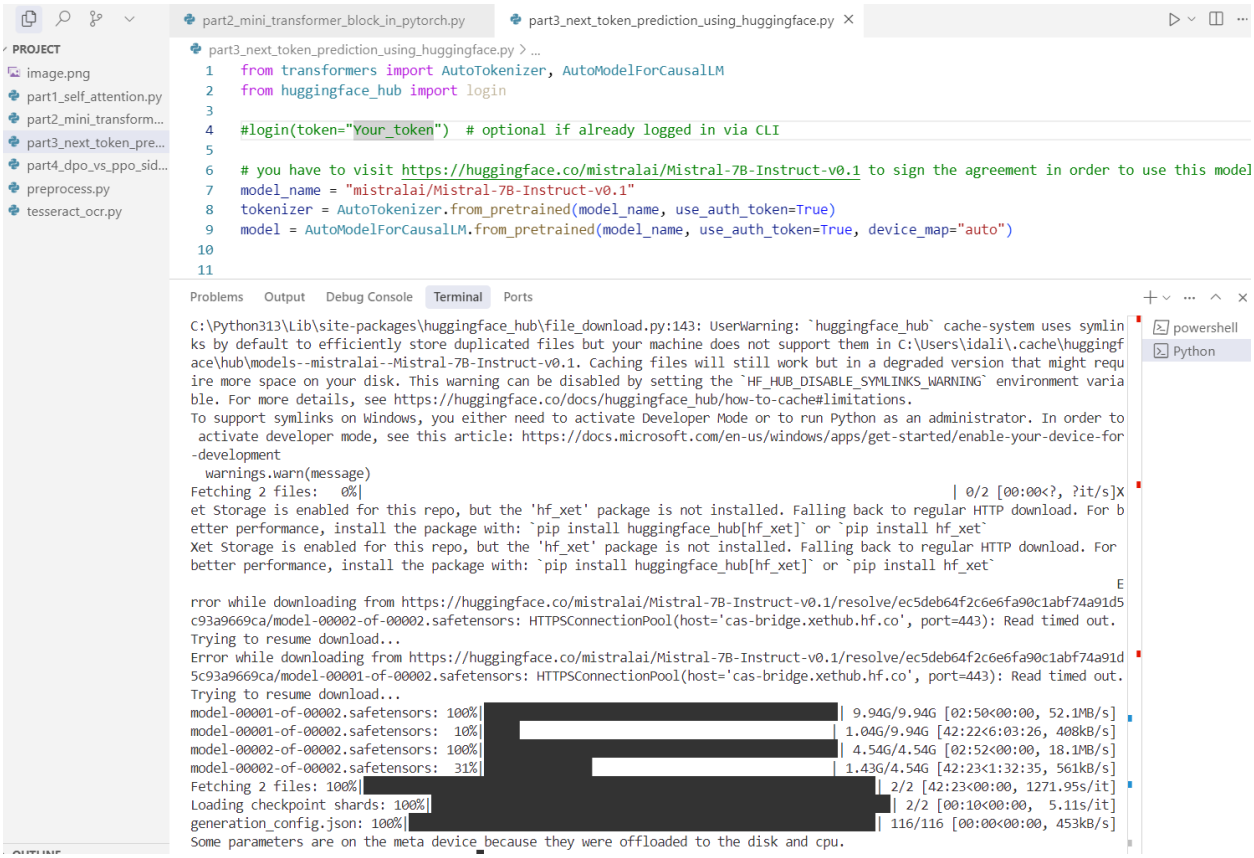
```
PS C:\AI_Coop\Homework\Week2\Project> & C:/Python313/python.exe c:/AI_Coop/Homework/Week2/Project/part2_mini_transformer_block_in_pytorch.py
torch.Size([1, 5, 16])
```

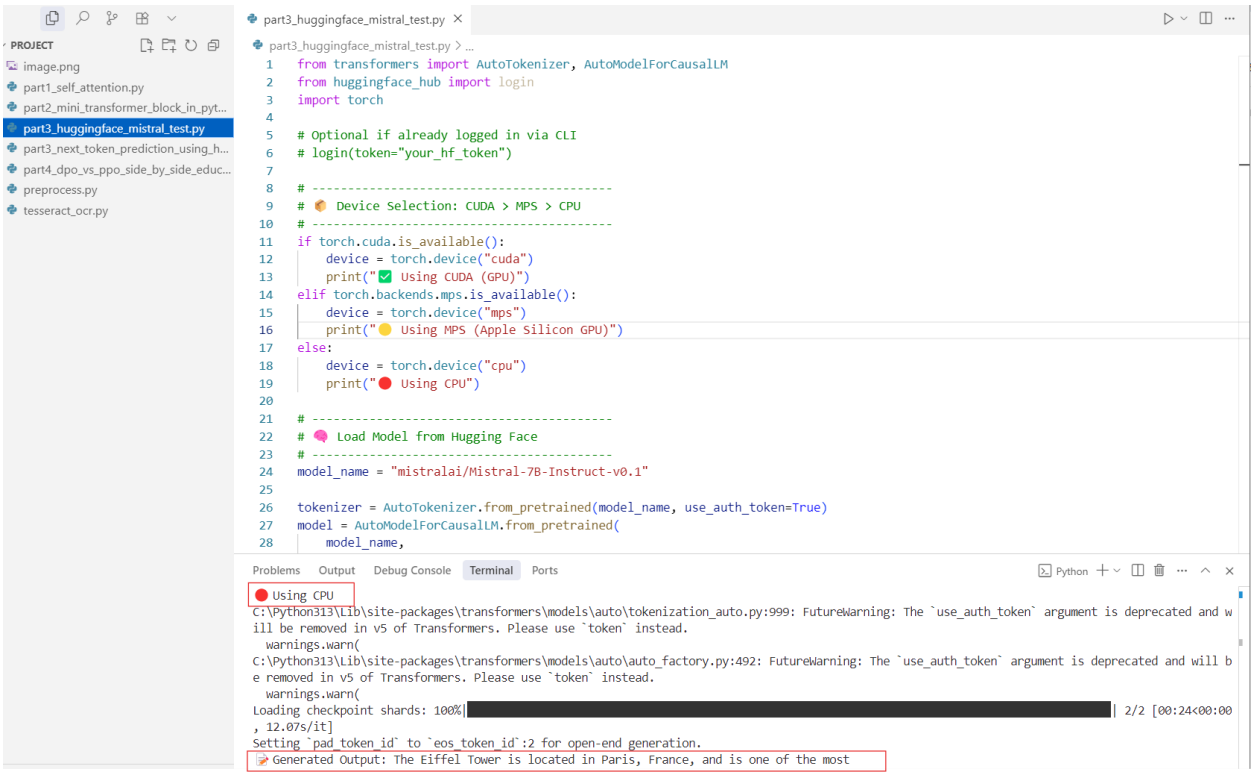
Part 3: Next Token Prediction using HuggingFace

Option 1: Use a Publicly Available Model

Problems 1 Output Debug Console Terminal Ports

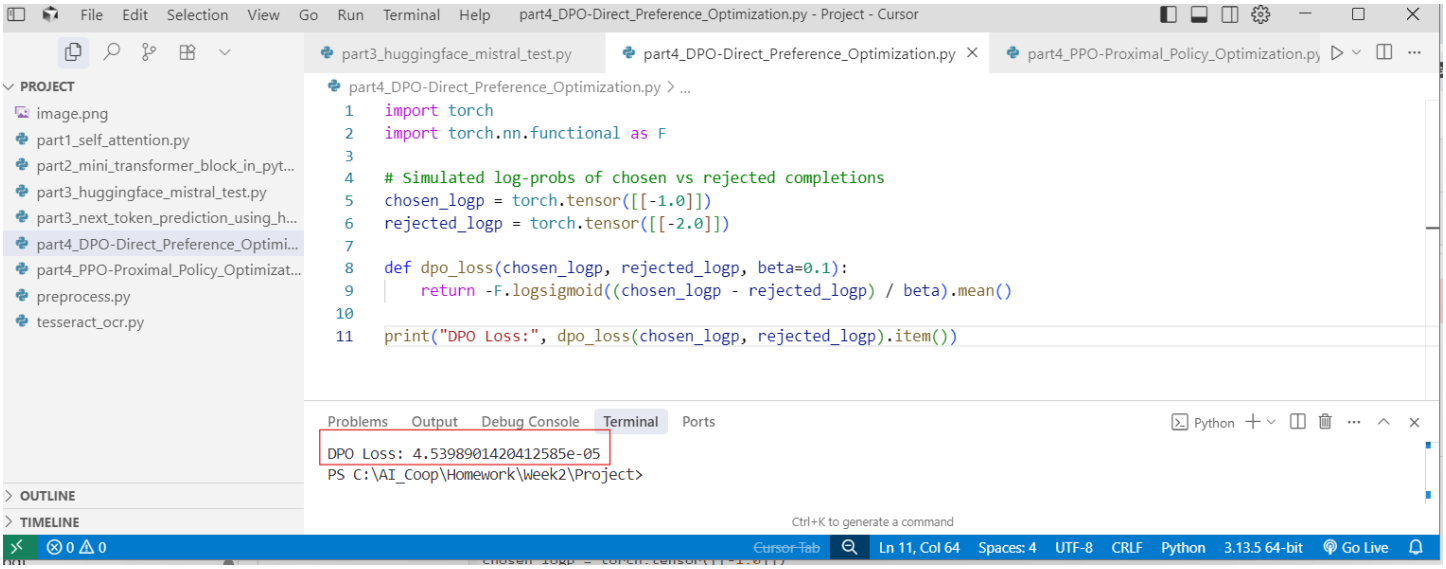
```
P
P
⚠ Warning: 'huggingface-cli login' is deprecated. Use 'hf auth login' instead.
The token has not been saved to the git credentials helper. Pass `add_to_git_credential=True` in this function directly or `--add-to-
git-credential` if using via `hf` CLI if you want to set the git credential as well.
Token is valid (permission: fineGrained).
The token `test` has been saved to C:\Users\idali\.cache\huggingface\stored_tokens
Your token has been saved to C:\Users\idali\.cache\huggingface\token
Login successful.
The current active token is: `test`
PS C:\AI_Coop\Homework\Week2\Project>
```



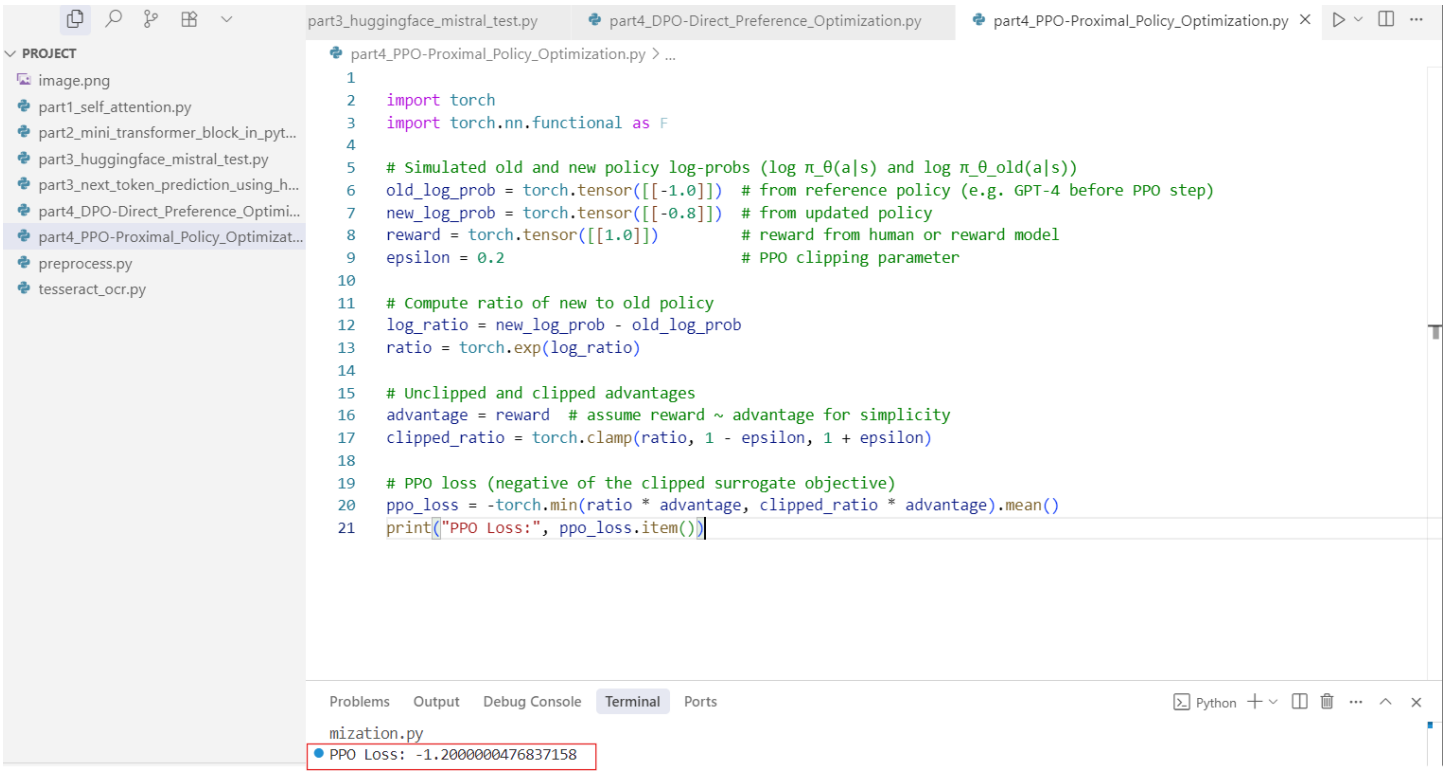


Part 4: DPO vs PPO – Side-by-Side Educational Example

DPO: Direct Preference Optimization



PPO: Proximal Policy Optimization (simplified for in-class demo)



Bonus: Inference with Quantization (O1 & O3)

 # Run model with torch_dtype=torch.float16 for O1

model = AutoModelForCausalLM.from_pretrained(model_name, torch_dtype=torch.float16, device_map="auto")

Loading checkpoint shards: 100%|██████████| 2/2 [00:10<00:00, 5.32s/it]

Concept: Explain how FP16/O1 optimizes memory and speed at inference.

要在 gpu 上跑一下

。。