import os  
from PIL import Image, ImageEnhance, ImageFilter  
from io import BytesIO  
import base64  
import re  
import pytesseract  
import fitz *# PyMuPDF for PDF processing*import csv  
  
*# 手动指定 Tesseract 的路径*pytesseract.pytesseract.tesseract\_cmd = os.path.join(os.path.dirname(\_\_file\_\_), "Tesseract-OCR", "tesseract.exe")  
  
  
*# 图片转base64*def get\_base64(image\_path):  
 with open(image\_path, "rb") as f:  
 base64\_data = base64.b64encode(f.read())  
 return base64\_data  
  
*# 调整图片大小*def resize\_image(image, max\_size=960):  
 width, height = image.size  
 if width > height:  
 new\_width = max\_size  
 new\_height = int((max\_size / width) \* height)  
 else:  
 new\_height = max\_size  
 new\_width = int((max\_size / height) \* width)  
 return image.resize((new\_width, new\_height), Image.Resampling.LANCZOS)  
  
*# 增强对比度并锐化图像*def enhance\_image(image):  
 *# 提高对比度* enhancer = ImageEnhance.Contrast(image)  
 enhanced\_img = enhancer.enhance(2)  
  
 *# 去噪声处理* cleaned\_img = enhanced\_img.filter(ImageFilter.MedianFilter(size=3))  
  
 *# 锐化处理* sharpened\_img = cleaned\_img.filter(ImageFilter.UnsharpMask())  
  
 *# 转换为灰度并进行二值化* grayscale\_img = sharpened\_img.convert('L')  
 binary\_img = grayscale\_img.point(lambda x: 0 if x < 128 else 255, '1')  
  
 return binary\_img  
  
*# 清理文本，提取英文字母和浮点数*def clean\_text(text):  
 *# 提取英文字符* english\_parts = re.findall(r'[A-Za-z]+', text)  
  
 *# 提取符号* numeric\_parts = re.findall(r'[^\s\w]', text)  
  
 *# 提取浮点数和整数，支持千分位逗号* numeric\_parts += re.findall(r'-?\d+(?:,\d{3})\*(?:\.\d+)?', text.replace(',', ''))  
  
 *# 仅保留合理的数字范围（根据需要修改）* def filter\_invalid\_numbers(numbers):  
 valid\_numbers = []  
 for num in numbers:  
 try:  
 *# 去掉千分位逗号* cleaned\_num = num.replace(',', '')  
 *# 尝试转换为浮点数* float\_num = float(cleaned\_num)  
 *# 仅保留在指定范围内的数字* if -100000 < float\_num < 100000:  
 valid\_numbers.append(num)  
 except ValueError:  
 *# 如果转换失败，跳过该条数字* continue  
 return valid\_numbers  
  
 numeric\_parts = filter\_invalid\_numbers(numeric\_parts)  
  
 return english\_parts, numeric\_parts  
  
*# 裁剪图片*def crop\_image(image, crop\_box):  
 return image.crop(crop\_box)  
  
*# PDF 转图片*def pdf\_to\_img(pdf\_dir, output\_img\_dir):  
 print("正在将PDF文件转换成图片......")  
 cons = []  
 pdf\_files = [f for f in os.listdir(pdf\_dir) if f.endswith('.pdf')]  
 for pdf in pdf\_files:  
 pdf\_path = os.path.join(pdf\_dir, pdf)  
 pdf\_doc = fitz.open(pdf\_path)  
 page = pdf\_doc[0]  
 zoom\_x = 3  
 zoom\_y = 3  
 mat = fitz.Matrix(zoom\_x, zoom\_y)  
 pix = page.get\_pixmap(matrix=mat, alpha=False)  
 Name = pdf[:pdf.rfind(".")]  
 image\_ori = os.path.join(output\_img\_dir, Name + ".png")  
 pix.save(image\_ori)  
 cons.append(image\_ori)  
 return cons  
  
*# 调整曝光度、饱和度和对比度*def adjust\_image\_properties(image):  
 *# 降低曝光度（亮度）* brightness\_enhancer = ImageEnhance.Brightness(image)  
 image = brightness\_enhancer.enhance(0.8) *# 降低曝光度  
  
 # 降低饱和度* saturation\_enhancer = ImageEnhance.Color(image)  
 image = saturation\_enhancer.enhance(0.5) *# 降低饱和度  
  
 # 提高对比度* contrast\_enhancer = ImageEnhance.Contrast(image)  
 image = contrast\_enhancer.enhance(2.0) *# 提高对比度* return image  
  
*# 裁剪和处理图片并进行 OCR 识别*def process\_image(image\_path, crop\_box):  
 im = Image.open(image\_path)  
 cropped\_img = crop\_image(im, crop\_box)  
 cropped\_image\_path = os.path.join(cropped\_img\_dir, "cropped\_" + os.path.basename(image\_path))  
 cropped\_img.save(cropped\_image\_path)  
 print(f"裁剪后的图片已保存到: {cropped\_image\_path}")  
 img\_b64 = get\_base64(cropped\_image\_path)  
 raw\_image = base64.b64decode(img\_b64)  
 img = Image.open(BytesIO(raw\_image))  
  
 *# 调整图片的曝光度、饱和度和对比度* img = adjust\_image\_properties(img)  
  
 img = resize\_image(img, max\_size=960)  
 img = enhance\_image(img)  
  
 *# 使用 OCR 进行识别* custom\_config = r'--oem 3 --psm 6'  
 text = pytesseract.image\_to\_string(img, config=custom\_config, lang='eng')  
  
 *# 清理文本，提取英文和数字* english\_parts, numeric\_parts = clean\_text(text)  
  
 return english\_parts, numeric\_parts  
  
*# 主程序入口*if \_\_name\_\_ == '\_\_main\_\_':  
 pdf\_dir = "./input\_pdf"  
 output\_img\_dir = "./OCR\_img"  
 cropped\_img\_dir = "./cropped\_images"  
 output\_csv\_dir = "./output\_csv"  
  
 if not os.path.exists(output\_img\_dir):  
 os.makedirs(output\_img\_dir)  
  
 if not os.path.exists(cropped\_img\_dir):  
 os.makedirs(cropped\_img\_dir)  
  
 if not os.path.exists(output\_csv\_dir):  
 os.makedirs(output\_csv\_dir)  
  
 crop\_box = (655, 415, 1005, 1135) *# 根据需要调整裁剪区域* image\_paths = pdf\_to\_img(pdf\_dir, output\_img\_dir)  
  
 all\_english\_parts = []  
 all\_numeric\_parts = []  
  
 for image\_path in image\_paths:  
 english\_parts, numeric\_parts = process\_image(image\_path, crop\_box)  
 all\_english\_parts.extend(english\_parts)  
 all\_numeric\_parts.extend(numeric\_parts)  
  
 *# 确保A列和B列长度一致* max\_length = max(len(all\_english\_parts), len(all\_numeric\_parts))  
 all\_english\_parts.extend([''] \* (max\_length - len(all\_english\_parts)))  
 all\_numeric\_parts.extend([''] \* (max\_length - len(all\_numeric\_parts)))  
  
 *# 导出未转置的 CSV 文件* re\_csv\_file = os.path.join(output\_csv\_dir, "re\_finnal\_results.csv")  
 with open(re\_csv\_file, mode='w', newline='', encoding='utf-8') as file:  
 csv\_writer = csv.writer(file)  
 for english, numeric in zip(all\_english\_parts, all\_numeric\_parts):  
 csv\_writer.writerow([english, numeric])  
  
 *# 将数据转置后写入新的 CSV 文件* csv\_file = os.path.join(output\_csv\_dir, "finnal\_results1.csv")  
 with open(csv\_file, mode='w', newline='', encoding='utf-8') as file:  
 csv\_writer = csv.writer(file)  
 csv\_writer.writerow(all\_english\_parts) *# 第一行：英文* csv\_writer.writerow(all\_numeric\_parts) *# 第二行：数字和符号* print(f"结果已保存至 {csv\_file} 和 {re\_csv\_file}")