import glob

import os

import PySimpleGUI as sg

import fitz

import pygame

import pytesseract

from PIL import Image

from gtts import gTTS

from pygame import mixer

from googletrans import Translator

# First and last page

def get\_text(value):

    string = value

    string = string.strip()

    if "-" in string:

        first\_page\_number = int(string.split("-")[0])

        last\_page\_number = int(string.split("-")[1])

    else:

        first\_page\_number = int(string)

        last\_page\_number = 0

    return first\_page\_number,last\_page\_number

def main():

    global e,first\_page\_number,last\_page\_number

    #Directory

    current\_directory = os.getcwd()

    final\_directory = os.path.join(current\_directory,r'Text\_to\_speech\_software')

    if not os.path.exists(final\_directory):

        os.makedirs(final\_directory)

    print(current\_directory)

    print(final\_directory)

    # GUI Part #

    layout = [  [sg.Text('Choose the desired PDF File'),sg.Input(),sg.FileBrowse()],

                [sg.Text('Enter PDF Page number or range separated by - '), sg.InputText()],

                [sg.Button('Ok'), sg.Button('Cancel')]

            ]

    window = sg.Window('Input', layout)

    valid = False

    while True:

        event, values = window.read()

        #Path of the pdf file

        pdf\_to\_read = values[0]

        if event in (None, 'Cancel'):

            print("Exitting")

            window.close()

            exit()

        if event == "Ok":

            if values[0] == "":

                sg.Popup("Enter value", "Enter PDF file to be transcribed ")

            if values[1] == "":

                sg.Popup("Enter value", "Enter page number(s) to be transcribed")

            if values[0]!="" and values[1]!="":

                for char in values[1]:

                    if char.isdigit()==False:

                        sg.Popup("Invalid value","Enter valid number(s) separated by -")

                        break

                    else:

                        valid=True

                        break

        if valid==True:

            print('You entered ', values[1])

            break

    window.close()

    first\_page\_number,last\_page\_number = get\_text(values[1])

    image\_directory = glob.glob(final\_directory)

    for file in os.listdir(final\_directory):

        filepath = os.path.join(final\_directory,file)

        print(filepath)

        os.remove(filepath)

    # Store PDF pages as images in a folder

    doc = fitz.open(pdf\_to\_read)

    k=1

    # Single page

    if last\_page\_number == 0:

        page = doc.loadPage(first\_page\_number-1)

        zoom\_x = 2.0

        zoom\_y = 2.0

        mat = fitz.Matrix(zoom\_x,zoom\_y)

        pix = page.getPixmap(matrix=mat)

        output = os.path.join(final\_directory, r"image\_to\_read.png")

        pix.writePNG(output)

    # Range of pages

    else:

        for i in range(first\_page\_number-1,last\_page\_number):

            page = doc.loadPage(i)

            zoom\_x = 2.0

            zoom\_y = 2.0

            mat = fitz.Matrix(zoom\_x,zoom\_y)

            pix = page.getPixmap(matrix=mat)

            output = os.path.join(final\_directory, r"image\_"+str(k)+"\_to\_read.png")

            pix.writePNG(output)

            k+=1

    print("Done")

    # Initialize the Pytesseract OCR software

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

    mytext = []

    # Read the text in image via pytesseract Optical Character Recognition (OCR) software

    for file in os.listdir(final\_directory):

        data = pytesseract.image\_to\_string(Image.open(os.path.join(final\_directory,file)),lang="eng")

        data = data.replace("|","I")

        data = data.split('\n')

        mytext.append(data)

    language = 'en'

    print(mytext)

    newtext= ""

    for text in mytext:

        for line in text:

            line = line.strip()

            if len(line.split(" ")) < 10 and len(line.split(" "))>0:

                newtext= newtext + " " + str(line) + "\n"

            elif len(line.split(" "))<2:

                pass

            else:

                if line[-1]!=".":

                    newtext = newtext + " " + str(line)

                else:

                    newtext = newtext + " " + line + "\n"

    print(newtext)

    translator = Translator()

    languages = {'af': 'afrikaans', 'sq': 'albanian', 'am': 'amharic', 'ar': 'arabic', 'hy': 'armenian',

                 'az': 'azerbaijani', 'eu': 'basque', 'be': 'belarusian', 'bn': 'bengali', 'bs': 'bosnian',

                 'bg': 'bulgarian', 'ca': 'catalan', 'ceb': 'cebuano', 'ny': 'chichewa',

                 'zh-cn': 'chinese (simplified)', 'zh-tw': 'chinese (traditional)', 'co': 'corsican', 'hr': 'croatian',

                 'cs': 'czech', 'da': 'danish', 'nl': 'dutch', 'en': 'english', 'eo': 'esperanto', 'et': 'estonian',

                 'tl': 'filipino', 'fi': 'finnish', 'fr': 'french', 'fy': 'frisian', 'gl': 'galician', 'ka': 'georgian',

                 'de': 'german', 'el': 'greek', 'gu': 'gujarati', 'ht': 'haitian creole', 'ha': 'hausa',

                 'haw': 'hawaiian', 'iw': 'hebrew', 'hi': 'hindi', 'hmn': 'hmong', 'hu': 'hungarian', 'is': 'icelandic',

                 'ig': 'igbo', 'id': 'indonesian', 'ga': 'irish', 'it': 'italian', 'ja': 'japanese', 'jw': 'javanese',

                 'kn': 'kannada', 'kk': 'kazakh', 'km': 'khmer', 'ko': 'korean', 'ku': 'kurdish (kurmanji)',

                 'ky': 'kyrgyz', 'lo': 'lao', 'la': 'latin', 'lv': 'latvian', 'lt': 'lithuanian', 'lb': 'luxembourgish',

                 'mk': 'macedonian', 'mg': 'malagasy', 'ms': 'malay', 'ml': 'malayalam', 'mt': 'maltese', 'mi': 'maori',

                 'mr': 'marathi', 'mn': 'mongolian', 'my': 'myanmar (burmese)', 'ne': 'nepali', 'no': 'norwegian',

                 'ps': 'pashto', 'fa': 'persian', 'pl': 'polish', 'pt': 'portuguese', 'pa': 'punjabi', 'ro': 'romanian',

                 'ru': 'russian', 'sm': 'samoan', 'gd': 'scots gaelic', 'sr': 'serbian', 'st': 'sesotho', 'sn': 'shona',

                 'sd': 'sindhi', 'si': 'sinhala', 'sk': 'slovak', 'sl': 'slovenian', 'so': 'somali', 'es': 'spanish',

                 'su': 'sundanese', 'sw': 'swahili', 'sv': 'swedish', 'tg': 'tajik', 'ta': 'tamil', 'te': 'telugu',

                 'th': 'thai', 'tr': 'turkish', 'uk': 'ukrainian', 'ur': 'urdu', 'uz': 'uzbek', 'vi': 'vietnamese',

                 'cy': 'welsh', 'xh': 'xhosa', 'yi': 'yiddish', 'yo': 'yoruba', 'zu': 'zulu', 'fil': 'Filipino',

                 'he': 'Hebrew'}

    destination\_lang = input("Destination Language :")

    print(destination\_lang, '---')

    result = translator.translate(newtext, src='english', dest=destination\_lang)

    with open('translated\_doc\_{}.txt'.format(languages[destination\_lang]), 'w', encoding="utf-8") as f:

        f.write(result.text)

    print(result.text)

    myobj = gTTS(text=newtext, lang=language, slow=False)

    # Audio in a mp3 file

    myobj.save(os.path.join(final\_directory,"pdf\_audio.mp3"))

    # Play the audio file

    pygame.init()

    mixer.init()

    mixer.music.load(os.path.join(final\_directory,"pdf\_audio.mp3"))

    mixer.music.play()

    pygame.event.wait()

   # GUI END #

if \_\_name\_\_ == '\_\_main\_\_':

    main()