

## 作业题目【黑体小三】

姓名：思亲赫

学号：61962051

班级：计 191

北京科技大学计通学院

---

**摘要：**利用 GUI-tkinter 库建立了 pdf 应用程序的界面，该界面可以提取文本、提取图像、顺时针旋转页面、将多个文件合并成一个大的 pdf 文件、将一个 pdf 文件按其包含的页数拆分、对文件进行加密以确保其安全以及标记为 python 的文件。

**关键字：**tkinter, pdf, 提取、拆分、合并旋转、加密、水印；设计；3-5 个分号分开。

**代码行数：**300 行

---

## 1. 项目背景和意义：

1.1. 为了这个项目，我决定做简化的 Acrobat。我决定这样做有很多原因。原因之一是 PDF 文档本身是非常灵活的文件系统。PDF 的主要目标是在不同的操作系统和设备之间共享文档。创建、转换和导出文件从来都不是一件容易的工作。几年前，跨多个计算机平台共享文档的问题非常严重。因此，迫切需要创建一种能够保持固定格式的文件格式。Acrobat 使文件共享更加容易。但 Acrobat 应用程序不是免费应用程序，所以要使用函数拆分文件我使用有限的免费站点。我创建应用程序的第一个原因是我可以自由使用自己。创建诸如提取文本和图像、将一个文件拆分为多个部分、旋转文件页面、将文件合并为一个大文件、加密文件、添加降价等免费功能对我来说非常重要。来构建这个简化的 Acrobat 的界面。我没有 GUI 方面的经验。但我试着挑战自己，尝试一些新的东西。我大部分时间都在使用 Tkinter 图书馆。我从未使用过这个模块，所以这对我来说是一次重要的学习经历。通过研究这个模块，我学到了很多東西。

从界面我可以看到我的代码可以做的一切，提取文本，合并，分割，提取图像，加密，旋转页面，添加水印。

## 2. 需求分析:

- 2.1 使用 GUI 库构建界面
- 2.2 用户可以理解界面
- 2.3 所有的按钮 widget 都在工作，做他们应该做的功能

## 3. 概要和详细设计:

### 3.1 概要:

首先，我将提到代码 `tkinter`、`pillow` 中使用的所有模块。然后使用根设置窗口的尺寸。然后设置应用说明标识。全部使用模块 `tkinter` 作为 `tk`。我写的第一个函数是提取图像。其次是从文件中提取文本。第三个功能是旋转文件中的页面。然后将 `pdf` 文件合并到一个文件中。之后是加密文件的函数。第六个功能是将文件分成不同的部分。最后一个用代码编写的函数是向文件中添加水印。

### 3.2 详细设计

#### 3.2.1 使用 `tkinter` 放置徽标:

`Tkinter` 是 `Tcl` 和 `Python` 标准 GUI 框架的组合，为您提供为您正在处理的任何应用程序创建丰富 UI 所需的所有小部件，但它特别适合开发桌面应用程序。

#### 3.2.3 背景

我设置背景不使用 `geometry class`，而只使用 `canvas class`。背景颜色为浅蓝色，因为它非常适合紫色按钮。宽度为 800，高度为 700，我根据应用程序中的功能数量设置它们。列跨度为 3，行也跨度为 3。当然，使用 `tkinter` 可以轻松设置背景。

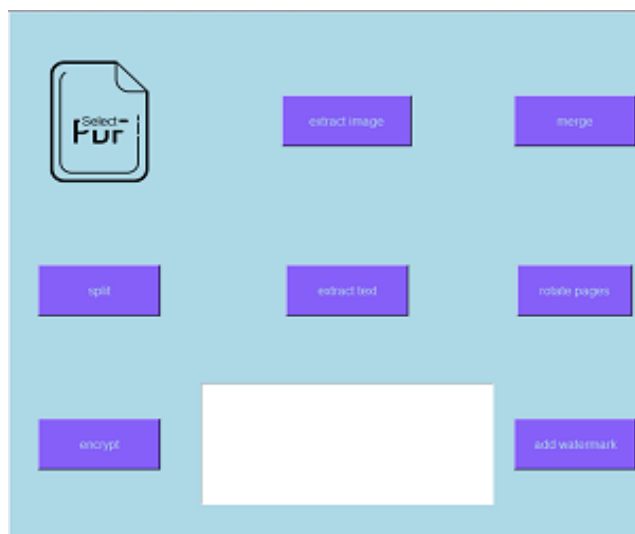


Figure 1

### 3.2.3 标志和说明:

- a. 我从 png 格式的 pdf 互联网标志下载。我选择这个简单的标志是因为我认为它更适合配色方案。这个简单的标志立即说明了它的含义。使用 tk 的 class label, 我将徽标设置为图像, 它是适合应用程序的背景。第一次添加时, 它有白色背景, 看起来不合适。它可以在下图中看到



Figure 2



Figure 3

- b. 为了指导, 我决定让它成为迷你弹力。所以它直接位于标志上方, 只说 pdf。它只是意味着选择您的 pdf。我不想长句说“请选择您的 pdf”。我想要一个简短而中肯的说明。



Figure 4

### 3.2.4 小部件按钮的设计:

使用 tk 的类 stringvar 设置变量作为已经设置的小部件。使用 tk 的 class button 设计, 它连接到根、文本变量和命令。按钮设置字体、背景颜色、按钮上所写内容的颜色及其高度和宽度。按钮的高度为 3, 宽度为 16。宽度为 16, 因为 15 有点奇怪。我喜欢宽度是 16, 它看起来更成比例。变量 grid 设置小部件的位置。字体为 arial, 按钮颜色为紫色, 因为它适合应用程序背景。

- a. 建立按钮说“extract text”使用 tkinter 模块和文本框的功能也使用 tkinter 模块。点击按钮后提取文本并显示在文本框中。

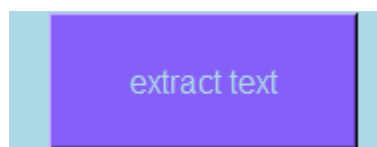


Figure 5

- b. 直接链接到提取文本有文本小部件。此窗口将显示提取的文本。它是由 `tk` 的 `class Text` 而制作的，并且位于应用程序的中心。标签配置放置在中心，它只停留在中心并在那里结束。两者都在中心，所以它会更容易理解。

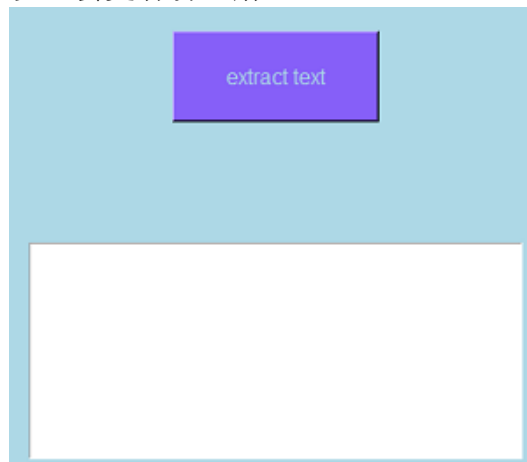


Figure 6

- c. 显示“extract image”的按钮就在“extract text”的正上方，因为我认为这两个提取功能应该很接近。此按钮打开文件夹，您可以在其中选择 `pdf` 并提取图像将提取的图像保存在文件夹中。按钮设计相似，但宽度稍长。

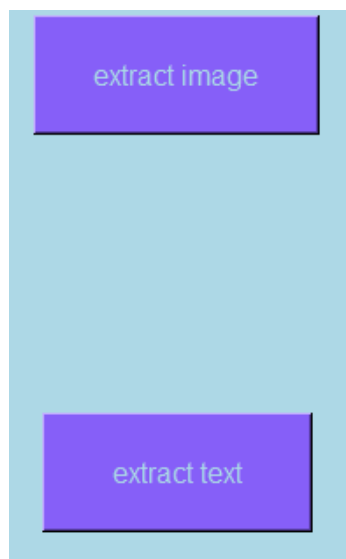


Figure 7

- d. 按钮说“merge”，这个功能合并文件。按钮的设计与提取文本相同。



Figure 8

- e. 按钮说“旋转页面”，这个功能旋转文件的页面。按钮小部件的设计与提取文本相同。



Figure 9

- f. 按钮说“添加水印”，此功能为文件添加水印。按钮小部件的设计与提取文本相同。

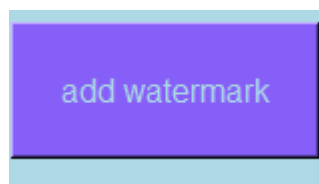


Figure 10

- g. 按钮说“拆分”，此功能将文件分成几部分。按钮小部件的设计与提取文本相同。

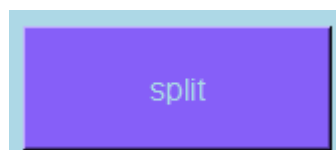


Figure 11

- h. 按钮说“加密”，这将加密文件。按钮小部件的设计与提取文本相同。

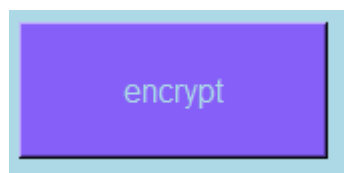


Figure 12

### 3.2.5 框图

使用 tk make 背景->使用 tk make logo->构建功能提取文本->制作功能的小部件按钮->构建文本框和它的小部件->提取图像的构建功能和小部件按钮->构建功能和小部件 合并  
-> 构建旋转页面的功能和小部件 -> 构建添加水印的功能和小部件 -> 构建拆分的功能和小部件 -> 构建加密功能和小部件

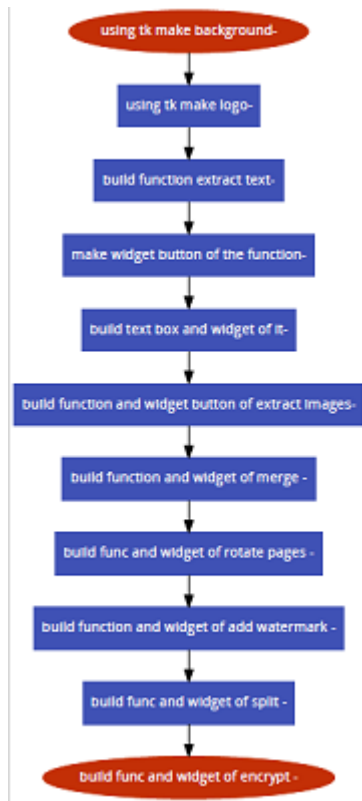


Figure 13

## 4. 代码实现:

### 4.1. Python 版本: Python3.8

### 4.2. 使用的库:

- Tkinter 是 Python 附带的 Tk GUI 工具包的 Python 接口。我们将在本章中查看此选项。并导入 `tkinter.filedialog` 以用作 `fd` 和 `askopenfile`。
- PyPdf2 包是一个纯 python pdf 库，可用于拆分、合并、裁剪和转换 pdf 中的页面。从 `pydpf2` 我使用了导入的 `pdffilemerger`、`pdffilereader` 和 `pdffilewriter`
- Pillow 是一个 Python 图像库 (PIL)，它增加了对打开、操作和保存图像的支持。从 PIL 我使用或导入 `image` 来打开图像。
- PyMuPDF (aka-fitz)，一种支持 MuPDF 1.9a 的 Python 绑定——“轻量级 PDF 和 XPS 查看器”。MuPDF 可以访问 PDF、XPS、OpenXPS 和 EPUB（电子书）格式的文件，以顶级性能和高渲染质量著称。
- Python io 模块允许我们管理与文件相关的输入和输出操作。使用 IO 模块的优点是可用的类和函数允许我们扩展功能以启用写入 Unicode 数据的功能。

### 4.3 代码:

整个项目的代码深受互联网教师的启发。我看到了使用 `tkinter` 的 pdf 提取应用程序的视频，它决定使用 `tkinter` 框架来完成这项任务。

- a) root 是 tkinter gui 框架首先编写的。和这段代码无需 class geometry 即可设置背景。

```
root = tk.Tk()
```

- b) 然后我设置背景类画布，设置它的高度和宽度。然后我设置它的位置。由于 root tk 不需要类几何测量。

```
background = tk.Canvas(root, width=800, height=700, bg="light blue")
background.grid(columnspan=3, rowspan=3)
```

- c) 要设置徽标，我使用模块 Pillow 中的 Image class 打开图像，png 导致其重要的是它是透明的。然后我使用 tk class Label 设置图像和背景颜色。并将 grid variable 用于显示徽标的位置。

```
logo = Image.open('logo.png')
logo = ImageTk.PhotoImage(logo)
logo_label = tk.Label(image=logo, bg="light blue")
logo_label.image = logo
logo_label.grid(column=0, row=0,)
```

- d) 我设置指令只是说“选择”，使用 class label 选择它的文本、字体、背景。之后使用 grid 将其放置在徽标正上方。选择它的列，行，列跨度。如上所示，整个应用程序由 3 列和 3 行组成

```
instructions = tk.Label(root, text="Select", font="Arial", bg="light blue")
instructions.grid(columnspan=1, column=0, row=0)
```

- e) 功能是从该文件夹中的 pdf 中提取图像。使用 filedialog 方法打开文件。迭代 PDF 页面。获取页面本身。打印在此页面中找到的图像数量。获取图像的外部参照。提取图像字节。在图像提取中打开图像字节。将输出文件保存为 png。关闭输出文件。

```
def image():
    # use filedialog method to open files
    file = fd.askopenfile(parent=root, mode='rb', title="Choose a file", filetypes=[("Pdf file", "*.pdf")])
    file = fitz.open(file)
    # iterate over PDF pages
    for page_index in range(len(file)):
        # get the page itself
        page = file[page_index]
        image_list = page.getImageList()
        # printing number of images found in this page
```

```

    if image_list:
        print(f"[+] Found a total of {len(image_list)} images in page {page_index}")
    else:
        print("[!] No images found on page", page_index)
    for image_index, img in enumerate(page.getImageList(), start=1):
        # get the XREF of the image
        xref = img[0]
        # extract the image bytes
        base_image = file.extractImage(xref)
        image_bytes = base_image["image"]

        image_ext = base_image["ext"]
        # open image bytes in image extract
        image_ext = Image.open(io.BytesIO(image_bytes))

        # save output file as png
        image_ext.save(open(f"image{page_index+1}_{image_index}.png", "wb"))

        # close outfile
        image_ext.close()
        image_ext.set("extract image")

```

- f) 将文本变量设置为 stringvar。使用 tkbutton 创建图像提取按钮。设置按钮小部件。设置按钮的下落。

```

# text variable as stringvar
image_ext = tk.StringVar()
# create image extract button
image_btn = tk.Button(root, textvar=image_ext, command=lambda:image_ext.set("extract image"), font="Arial", bg="#865ff8", fg="light blue", height=3, width=17)
# set the button
image_ext.set("extract image")
# button's whereabouts
image_btn.grid(column=1, row=0)

```

- g) 为提取的文本创建文本框。将文本框置于界面中央。设置文本框的位置。

```

# text box for extracted text
text_box = tk.Text(root, height=10, width=48, padx=15, pady=15)
# place the text box in the center of interface

```



```

text_box.tag_configure("center", justify="center")
text_box.tag_add("center", 1.0, "end")
# text box's location
text_box.grid(column=1, row=2)

```

- h) 功能是从 pdf 文件中提取文本。当功能正在加载时打印这个。要打开文件夹而不仅仅是随机 pdf 文件以从其他文件中选择文件，请使用此变量，在设置模式下，在这种情况下为 pdf 文件的文件类型。

```

def extract_file():
    # while loading
    extract_text.set("loading...")
    # open file
    file = askopenfile(parent=root, mode='rb', title="Choose a file",
    , filetypes=[("Pdf file", "*.pdf")])

```

- i) 循环以 ig 开始，在从 pages 提取文本之后，首先必须读取文件，下一行 page 变量 getpage 。最后一行将文本插入文本框。

```

if file:
    # at first has to read files
    read_pdf = PdfFileReader(file)
    # page getpage
    page = read_pdf.getPage(0)
    # extract text from pages
    page_content = page.extractText()

    text_box.insert(1.0, page_content)

```

- j) 设置提取文本按钮小部件。然后创建同名的 tk stringvar。下一行创建提取文本按钮的设计。这与前一个相似。之后使用 tkgrid 设置按钮位置。

```

extract_text.set("extract text")
# module tk stringvar
extract_text = tk.StringVar()
# create extract text button
extract_btn = tk.Button(root, textvar=extract_text, command=lambda:extract_file(), font="Arial", bg="#865ff8", fg="light blue", height=3, width=16)
extract_text.set("extract text")
# button's location
extract_btn.grid(column=1, row=1)

```

- k) 功能是旋转文件中的页面。第一部分与前面的函数相同，它们的代码相似。无需解释。当开始编写循环代码时，事情变得有点棘手。首先循环有 2 个变量，pdf\_reader 和 pdf\_writer。

```
if file:
    pdf_writer = PdfFileWriter()
    pdf_reader = PdfFileReader(file)
```

在那之后，我有一些不起作用的代码。

```
# Rotate page 90 degrees to the right
page_1 = pdf_reader.getPage(0).rotateClockwise(90)
pdf_writer.addPage(page_1)
# Rotate page 90 degrees to the left
page_2 = pdf_reader.getPage(1).rotateCounterClockwise(90)
pdf_writer.addPage(page_2)
# Add a page in normal orientation
pdf_writer.addPage(pdf_reader.getPage(2))
rotate_page.set("rotate pages")'''
```

通过互联网冲浪并咨询我的朋友后，我们找到了没有问题的代码。它以变量 pdf 阅读器 numPages 中的 iritate pagenum 开始。下一个变量页面选择页面以使用变量 getPage 旋转。相同的变量旋转页面。然后创建旋转页面。然后旋转的文件获得名称并保存在文件夹中。然后关闭输出文件变量。

```
# iritate pagenum in variable pdf reader numPages
for pagenum in range(pdf_reader.numPages):
    # select page to rotate with variable getPage
    page = pdf_reader.getPage(pagenum)
    # rotate the page
    page.rotateClockwise(90)
    # create rotated page
    pdf_writer.addPage(page)
# output file
pdf_out = open('rotated.pdf', 'wb')
pdf_writer.write(pdf_out)
# close output file
pdf_out.close()
```

- l) 旋转按钮小部件代码类似于其他两个按钮小部件。  
m) 下一个功能是 merge 将文件合并到一个大的 pdf 文件中。代码的开头与其他类似。循环代码起初很糟糕，然后由于互联网，它变得更简单了。

```
'''pdf_writer=PdfFileWriter
pdf_reader = PdfFileReader(files)
for page in range(pdf_reader.getNumPages()):
```

```

        pdf_writer.addPage(pdf_reader.getPage(page))
    pdf_out = open('merged.pdf','wb')
    pdf_writer.write(pdf_out)
    pdf_out.close()'''

```

变量是 merge 这是 pdfmerger pypdf2 模块的基类。找到 pdf 文件，然后附加它们。结果文件合并为一个不同的 pdf 文件。

```

merge = PdfFileMerger()
for pdf in files:
    merge.append(pdf)
merge.write('result.pdf')
merge.close()
merge_text.set("merge")

```

“合并”按钮的代码与其他按钮相同。

- n) 在函数合并之后，我不知道还有什么函数可以做。在网上冲浪后，我看到了看起来很有趣的 pdf 加密。然后在稍微改变它的变量后实现了代码。输入 pdf 读取文件，加密 pdf 文件写入并将页面添加到文件中。加密文件设置密码为 'password'。并打开标题为“受保护”的输出文件。

```

inpupdf = PdfFileReader(file)

pages_no = inpupdf.numPages
for i in range(pages_no):

    inputpdf = PdfFileReader(file)
    # pdfwriter part of pypdf2 module
    encrypt_file = PdfFileWriter()
    # encrypting pdf file
    # add page to inputfile
    encrypt_file.addPage(inputpdf.getPage(i))
    encrypt_file.encrypt("password")
    # open outputfile
    with open("protected.pdf", "wb") as outputenc:
        encrypt_file.write(outputenc)

```

加密按钮小部件代码与其他按钮相同。

- o) 至于功能拆分 pdf，起初我发现并编写了失败的尝试。

```

'''
    for pdf in range(pdf.getNumPages()):
        pdf_writer = PdfFileWriter()
        pdf_writer.addPage(pdf.getPage(pdf))
        output_filename = '{}_page_{}.pdf'.format(file+1)

```

```

        with open(output_filename, 'wb') as out:
            pdf_writer.write(out)
            print('Created: {}'.format(output_filename))
    if __name__ == '__main__':
        split_text(file)'''

```

通过互联网搜索后，我发现可以在我的代码中实现的更简单的代码。

```

# split text in pdf file
for i in range(inputpdf.numPages):
    split_text = PdfFileWriter()
    split_text.addPage(inputpdf.getPage(i))
    # split page by page
    with open("doc-page%s.pdf"%(i+1), "wb") as outputStream:
        split_text.write(outputStream)
# close splitted file
file.close()

```

拆分按钮小部件代码与其他代码类似。

- p) 对于功能添加水印，我遇到了类似的问题。 我实现了无效的复杂代码。 在寻找好的答案后，我终于找到了在稍作更改并实现后可以工作的代码。

```

pdf_file = file()
watermark = "watermark.pdf"
merged_file = "addedwatermark.pdf"
inputpdf = open(pdf_file,'rb')
input_pdf = PdfFileReader(inputpdf,'rb')
watermark_file = open(watermark)
watermark_pdf = PdfFileReader(watermark_file,'rb')
pdf_pg = input_pdf.getPage(0)
watermark_page = watermark_pdf.getPage(0)
pdf_pg.mergePage(watermark_page)
wmoutput = PdfFileWriter()
wmoutput.addPage(pdf_pg)
merged_file = open(merged_file,'wb')
wmoutput.write(merged_file)
merged_file.close()
watermark_file.close()
file.close()'''

```

首先打开 watermark.pdf。下一行代码使用 pdfreader 读取文件并写入新变量 eith pdfwriter。int Pagenum 在变量 pdfreader 中,这是我们的文件。变量 pageobj 将文件 watermark.pdf 与所选文件合并。 pdfwriter 变量添加页面以制作带水印的全新文件。

```
# watermark file
wmfile = open("watermark.pdf", 'rb')
pdfwmreader = PdfFileReader(wmfile)
# read file with pdfreader and write in new variable eith pdfwriter
file = askopenfile(parent=root, mode='rb', title="Choose a file",
, filetypes=[("Pdf file", "*.pdf")])
pdfreader=PdfFileReader(file)
pdfwriter = PdfFileWriter()

for pageNum in range(pdfreader.numPages):
    pageObj = pdfreader.getPage(pageNum)
    pageObj.mergePage(pdfwmreader.getPage(0))
    pdfwriter.addPage(pageObj)

wmoutput = open("watermarked.pdf", 'wb')
pdfwriter.write(wmoutput)

wmfile.close()
file.close()
wmoutput.close()
```

添加水印按钮小部件代码与其他按钮类似。

## 5. 程序展示和测试:

5.1 功能提取文本。 首先,单击按钮提取文本,然后打开文件夹并选择带有文本的 pdf 文件。 他们是 Fami.pdf 和 Through.pdf,我选择了 Fami.pdf。 选择文本后出现在文本框中。

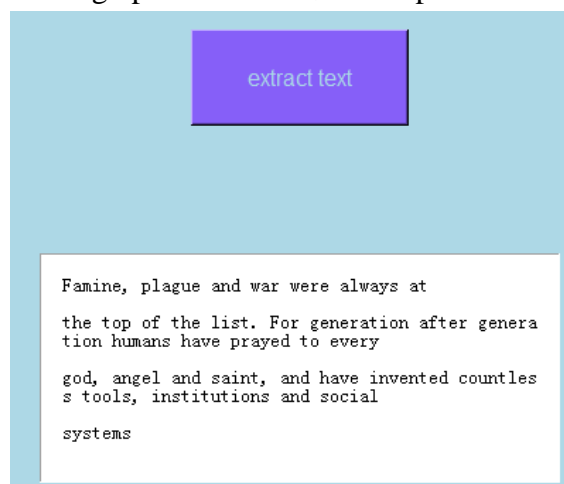
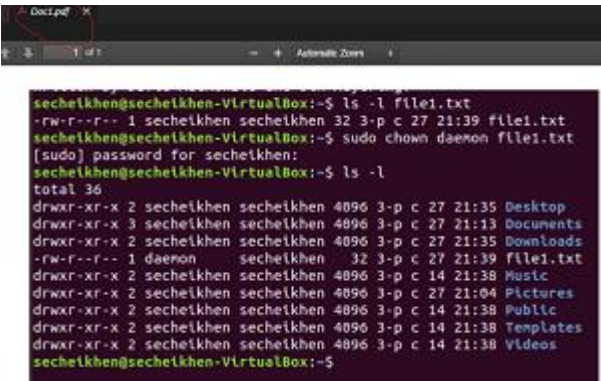


Figure 14

5.2 提取图像的功能。我点击按钮提取图像，它打开文件夹，我选择 Doc1.pdf 因为这个 pdf 包含照片。然后函数将这张新提取的照片保存为 image1\_1.png。它仅适用于包含图像的 pdf。



```
secheikhen@secheikhen-VirtualBox:~$ ls -l file1.txt
-rw-r--r-- 1 secheikhen secheikhen 32 3-p c 27 21:39 file1.txt
secheikhen@secheikhen-VirtualBox:~$ sudo chown daemon file1.txt
[sudo] password for secheikhen:
secheikhen@secheikhen-VirtualBox:~$ ls -l
total 36
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 27 21:35 Desktop
drwxr-xr-x 3 secheikhen secheikhen 4096 3-p c 27 21:13 Documents
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 27 21:35 Downloads
-rw-r--r-- 1 daemon secheikhen 32 3-p c 27 21:39 file1.txt
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 14 21:38 Music
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 27 21:04 Pictures
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 14 21:38 Public
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 14 21:38 Templates
drwxr-xr-x 2 secheikhen secheikhen 4096 3-p c 14 21:38 Videos
secheikhen@secheikhen-VirtualBox:~$
```

Figure 15

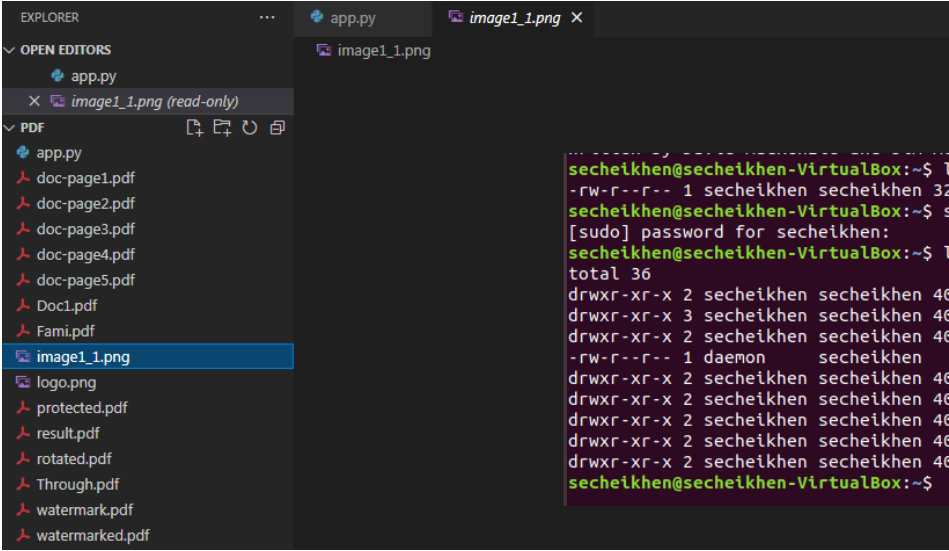


Figure 16

5.3 功能旋转页面。此功能是将页面顺时针旋转 90 度。我单击按钮旋转提取文本右侧的页面。它打开包含 pdf 的文件夹，然后选择 doc1.pdf。现在 rotated.pdf 已出现在文件夹中并已保存。

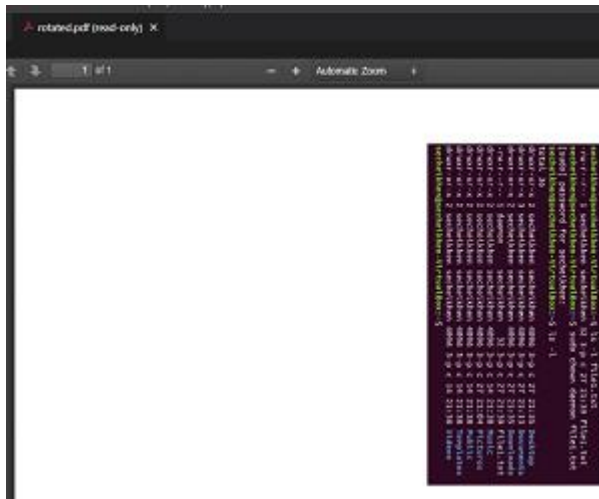


Figure 17

它确实将文件中的页面旋转了 90 度。

5.4 功能是将文件分成几部分。我点击按钮拆分，它打开文件夹。我选择文件 Fami.pdf。然后函数保存文件：doc\_page1、doc\_page2、doc\_page3、doc\_page4、doc\_page5.pdf。函数在其页面中拆分了 pdf 文件。

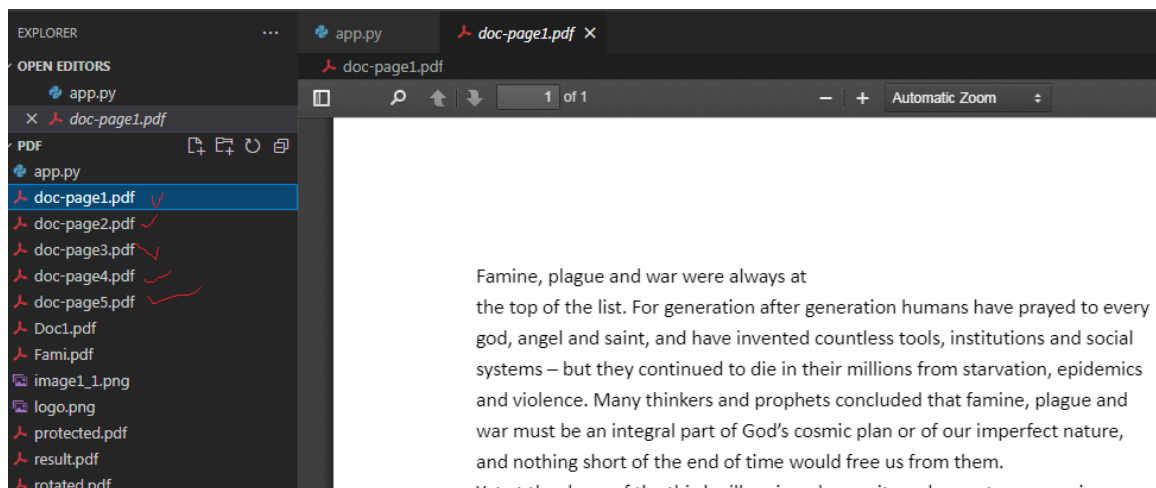


Figure 18

5.5 功能是合并文件。我点击按钮合并，它转移到文件夹。然后我选择 doc\_page1.pdf 和 doc\_page2.pdf。函数将它们合并在一起并保存在名为 "result.pdf" 的新 pdf 文件中。此 pdf 包含 2 页。

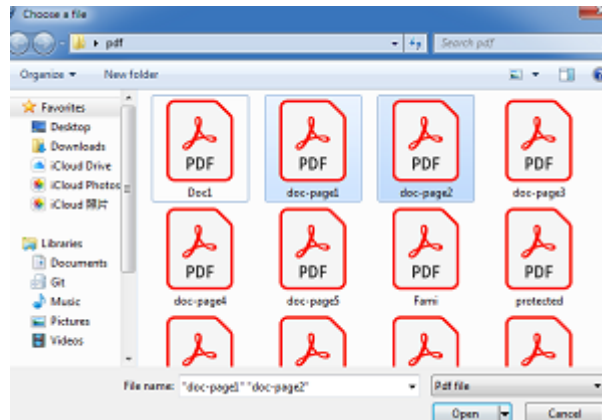
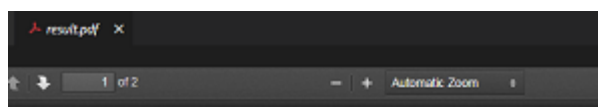


Figure 19



Famine, plague and war were always at the top of the list. For generation after generation humans have prayed to every god, angel and saint, and have invented countless tools, institutions and social systems – but they continued to die in their millions from starvation, epidemics and violence. Many thinkers and prophets concluded that famine, plague and war must be an integral part of God's cosmic plan or of our imperfect nature, and nothing short of the end of time would free us from them. Yet at the dawn of the third millennium, humanity wakes up to an amazing realisation. Most people rarely think about it, but in the last few decades we have managed to rein in famine, plague and war. Of course, these problems have not been completely solved, but they have been transformed from incomprehensible and uncontrollable forces of nature into manageable challenges. We don't need to pray to any god or saint to rescue us from them. We know quite well what needs to be done in order to prevent famine, plague and war – and we usually succeed in doing it. True, there are still notable failures; but when faced with such failures we no longer shrug our shoulders and say, 'Well, that's the way things work in our imperfect world' or 'God's will be done'. Rather, when famine, plague or war

Figure 20



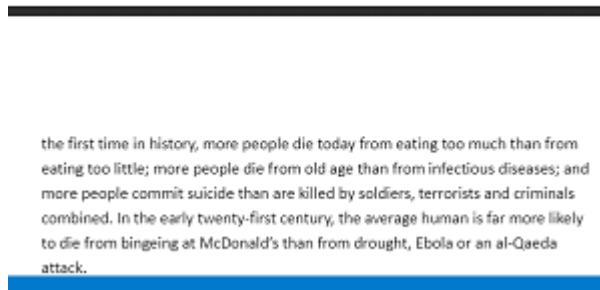
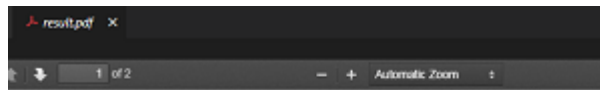


Figure 21

5.6 功能是加密文件。我点击按钮 `encryptd`，它打开 `pdf` 文件夹，然后我选择 `pdf` 文件。之后在文件夹 `pdf` 文件中出现“`protected.pdf`”，打开文件我必须输入密码。我输入密码，即“密码”。它打开文件。

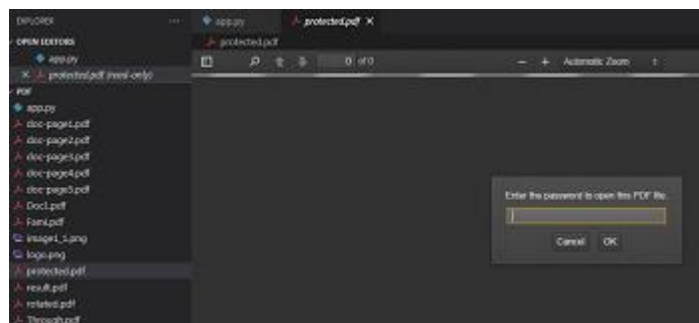


Figure 22

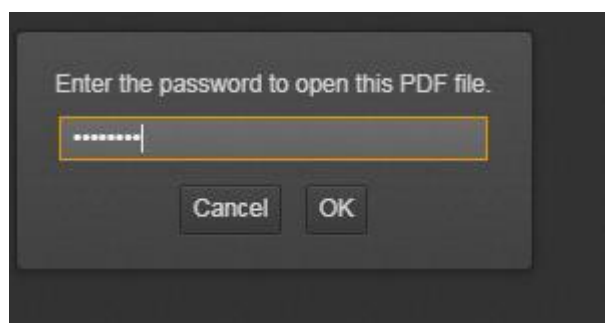


Figure 23

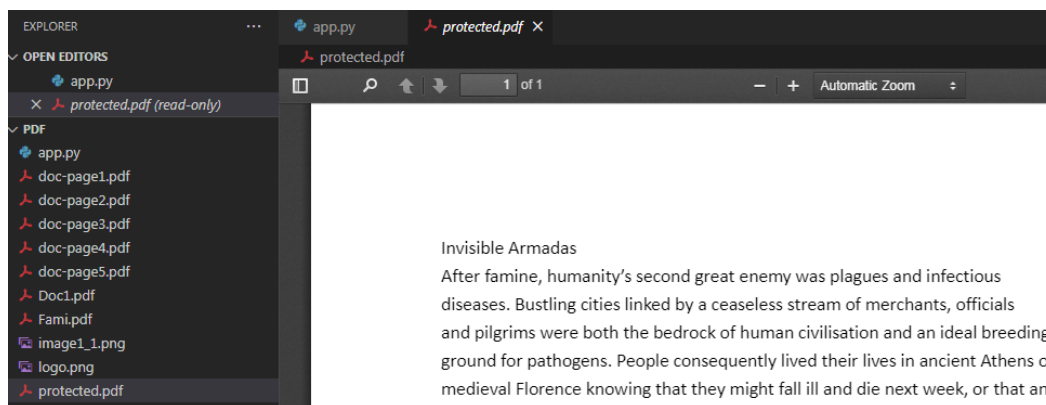


Figure 24

这是打开的文件，标题为“protected.pdf”。

5.7 最后一个功能是给文件添加水印。有一个 pdf 文件只包含水印，该文件将与选定的文件合并。我点击按钮添加水印，这将打开文件文件夹并选择 Through.pdf。对这 2 个文件进行函数处理并制作名为“watermarked”的全新文件。

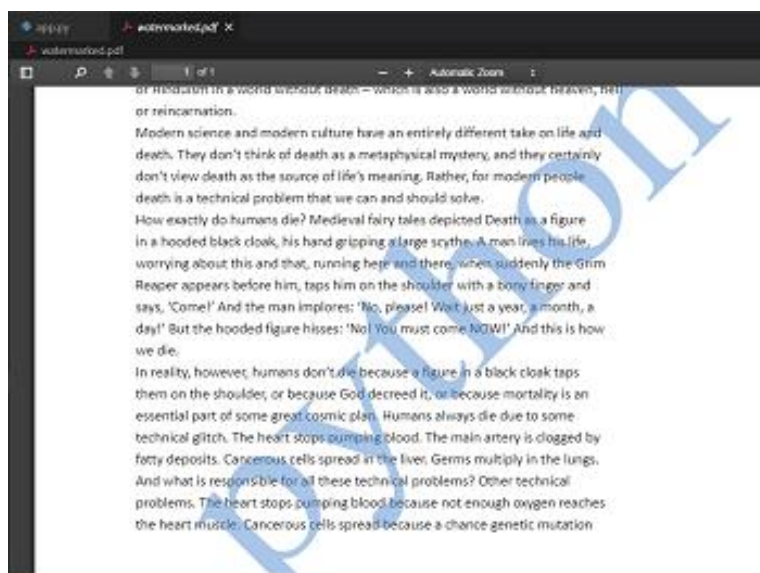


Figure 25

## 6. 结论和未来方向

我制作了简化的 acrobat pdf 桌面应用程序。由于某些功能（如合并和拆分）是真实 Acrobatpdf 中付费订阅的一部分，我真的很高兴我编写了代码，可以做我以前无法做的事情。此应用程序提取文本和图像，旋转文件页面，合并文件，将文件拆分为多个部分，加密文件并为文件添加水印。

通过这样做和报告，我学到了很多。我学会了使用 tkinter 框架来构建桌面应用程序。我学会了使用 tkinter 库制作小部件。当我犯了错误并通过它们学习时，我学到了很多关于 python 的知识。

感谢这门课程和它的作业，我成为了更好的 Python 程序员。我将了解有关 Python 语言的更多信息。将来我会成为越来越好的 python 程序员。

## 7. 致谢

首先，我要感谢教授这门课程的老师。这是获得更多知识的不可思议的旅程。其次要感谢我的同学孙金平，他给了我很大的帮助和建议。他随时准备为我提供帮助和建议。他帮助了多个职能部门，如果没有他，我将无法完成这个项目。第三，我要感谢互联网上所有的 python 专家，大师和老师，我使用了他们的知识。列表太长，所以每个人的链接都在参考文献中提到。

## 八、参考文献

### Bibliography

1.  
MariyaSha. MariyaSha/PDFextract\_text. GitHub. Published May 27, 2021. Accessed June 25, 2021. [https://github.com/MariyaSha/PDFextract\\_text](https://github.com/MariyaSha/PDFextract_text)
2.  
Rockikz A. How to Extract Images from PDF in Python - Python Code. [www.thepythoncode.com](http://www.thepythoncode.com). Published August 2020. Accessed June 25, 2021. <https://www.thepythoncode.com/article/extract-pdf-images-in-python>
3.  
Real Python. How to Work With a PDF in Python. [Realpython.com](http://Realpython.com). Published April 17, 2019. Accessed June 25, 2021. <https://realpython.com/pdf-python/>
4.  
Hofmann F. Working with PDFs in Python: Reading and Splitting Pages. Stack Abuse. Published 2017. Accessed June 25, 2021. <https://stackabuse.com/working-with-pdfs-in-python-reading-and-splitting-pages>
5.  
<https://www.facebook.com/roytuts2014>. How to encrypt PDF as Password Protected File in Python - Roy Tutorials. Roy Tutorials. Published February 22, 2021. Accessed June 27, 2021. <https://roytuts.com/how-to-encrypt-pdf-as-password-protected-file-in-python/>
6.  
monkut. split a multi-page pdf file into multiple pdf files with python? Stack Overflow. Published January 29, 2009. Accessed June 27, 2021. <https://stackoverflow.com/questions/490195/split-a-multi-page-pdf-file-into-multiple-pdf-files-with-python>
7.  
S P. Applying PDF watermark via a for loop. Stack Overflow. Published September 27, 2017. Accessed June 27, 2021. <https://stackoverflow.com/questions/46454452/applying-pdf-watermark-via-a-for-loop>
8.  
How to Add Watermark to a PDF File Using Python - CodeSpeedy. CodeSpeedy. Published December 30, 2019. Accessed June 27, 2021. <https://www.codespeedy.com/how-to-add-watermark-to-a-pdf-file-using-python/>