PDF Notebook

将PDF自动拆解为图片再按照一定的顺序生成PDF文档，并在每四张图片后自动添加条形码。

1. 依赖环境

- Python 3.x
- reportlab
- PyPDF2
- tqdm
- fitz
- Pillow

2. 问题分析

本题的主要难点在于如何将4页PDF文档合并在一页中。如果使用PyPDF2库读取PDF，会返回一个PDF Object。同时经过查阅资料，PyPDF2库中没有能够合并Page Object的方法。因此采用PyPDF2库完成该题的方案被我否决了。

经过多次的查询资料(实现多页pdf合并为单页的资料极少)，我最终采用了**reportlab**库来实现该题所需功能。

本题需要完成的核心任务拆解为以下若干点：

1. PDF文件转换为PNG图片。
2. 实现对四张图片的左排版。
3. 实现对若干条横线的右排版。
4. 实现对条形码的右上排版。

3. 实现思路

本项目的核心思路是采用reportlab库的Canvas类来实现对PDF文档的生成。
ReportLab可以用于生成各种PDF文档，包括报告、合同、内部文件等。它的功能包括：

- 生成PDF文档：ReportLab可以创建PDF文档，支持文本、图片、表格、线条、图表、页眉页脚等元素。

- PDF文档样式：ReportLab可以设置页面尺寸、背景颜色、字体、字号、对齐方式、颜色、边距等细节样式。
- PDF文档布局：ReportLab可以控制文档的排版和布局，包括页面的大小和方向、内容的位置和大小、多列和分页等。

基于reportlab强大的编辑功能，才可以满足本题中对pdf高度自定义的需求。

3.1 PDF文件转换为PNG图片

此处采用 `fitz` 库实现对pdf转图片的功能，保存为jpg图片格式到指定目录。具体使用的代码如下:

```
def pdf_to_image(pdf_file, output_folder=None, image_format='png', dpi=600):

    if output_folder is None:
        output_folder = os.path.splitext(pdf_file)[0]
    os.makedirs(output_folder, exist_ok=True)

    # extract pdf pages and convert to image
    pdf = fitz.open(pdf_file)
    for i, page in tqdm(enumerate(pdf), total=len(pdf), desc='Converting PDF to image'):
        pixel_map = page.get_pixmap(dpi=dpi, alpha=False)
        pixel_map.save(os.path.join(output_folder, '{}.{}'.format(str(i), image_format)))
    pdf.close()

```

该函数读取pdf，并自动将pdf的每一页转换为jpg文件存在 `output_folder` 目录下。其中，dpi参数表示图片的分辨率，dpi越大，图片越清晰，但是图片的大小也会随之增大，默认为600dpi。

3.2 实现对四张图片的左排版

对图片的排版主要使用了reportlab中canvas类的 `drawInlineImage()` 函数，用于在页面内指定位置绘制图像。样例代码如下:

```
img = Image.open(i)
c.drawInlineImage(img,
                  img_pos[index % 4][0],
                  img_pos[index % 4][1],
                  width=0.45 * PAGE_WIDTH,
                  height=0.25 * PAGE_HEIGHT)
```

3.3 实现对若干条横线的右排版

采用canvas类中的 `line()` 函数，用于在页面内指定位置绘制直线。样例代码如下:

```
c.line(i[0], i[1], i[0] + 0.4 * PAGE_WIDTH, i[1])

```

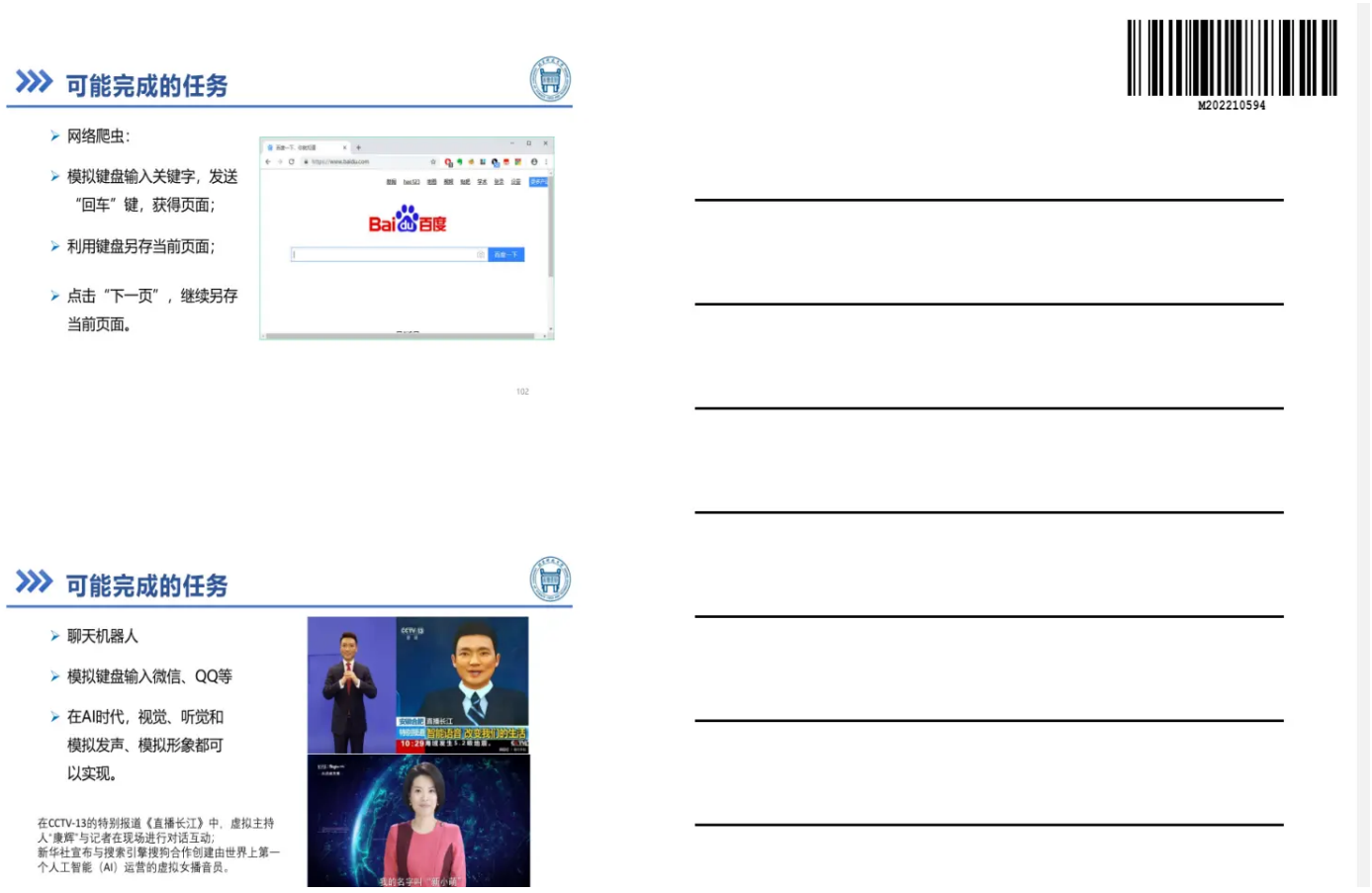
3.4 实现对条形码的右上排版。

与3.3类似，采用canvas类中的 `drawInlineImage()` 函数，用于在页面内指定位置绘制图像。样例代码如下：

```
c.drawInlineImage(bar_code_path,
                  0.84 * PAGE_WIDTH,
                  0.94 * PAGE_HEIGHT,
                  width=0.15 * PAGE_WIDTH,
                  height=0.05 * PAGE_HEIGHT)
```

4. 效果展示

1. 局部放大图，包含pdf，横线，右上角带有学号的条形码



2. 整体效果图

读取表格

- 假设存在Excel文件：example.xlsx

	A		
1	2019-4-1	Li Lei	52
	2019-4-2	Han Meimei	300
	2019-4-5	Li Lei	230
	2019-4-8	Li Lei	170
	2019-4-10	Han Meimei	96

取得工作簿及工作表

```
>>> import openpyxl
>>> wb = openpyxl.load_workbook('example.xlsx')
>>> wb.sheetnames
['Sheet1', 'Sheet2', 'Sheet3']
>>> sheet = wb['Sheet3']
>>> sheet
<Worksheet "Sheet3">
>>> type(sheet)
<class 'openpyxl.worksheet.worksheet.Worksheet'>
>>> sheet.title
'Sheet3'
>>> anotherSheet = wb.active
>>> anotherSheet
<Worksheet "Sheet1">
```

打开工作簿

获取所有sheet的名称

按名称或活跃单元格获取Worksheet对象。

取得单元格

	A	B	C
1	2019-4-1	Li Lei	52
2	2019-4-2	Han Meimei	300

```
>>> sheet = wb['Sheet3']
>>> sheet['A1']
<Cell Sheet1.A1>
>>> sheet['A1'].value
datetime.datetime(2019, 4, 1, 0, 0)
>>> c = sheet['B1']
>>> c.value
'Li Lei'
>>> f'Row {c.row}, Column {c.column} = {c.coordinate}'
'Row 1, Column B = B1'
>>> sheet['C1'].value
52
```

取得单元格

	A	B	C
1	2019-4-1	Li Lei	52
2	2019-4-2	Han Meimei	300

```
>>> for i in range(1, 5):
    print(i, sheet.cell(row=i, column=2).value)
1 Li Lei
2 Han Meimei
3 Li Lei
4 Li Lei
>>> print(sheet.max_row, sheet.max_column)
5 3
```

cell对象也可以通过行、列的序号来访问，均从1开始。

通过max_row属性获得最大的行，通过max_column属性获得最大的列。