PDF文件收集与拼接实验

1. 实验目的

本实验旨在指导用户掌握如何通过 Python 编写脚本实现对多个实验文件夹中 PDF 文档的自动化收集、中心化存档以及带有逻辑索引和书签的文档拼接。通过本实验，参与者将学习并掌握以下核心技能：

1. **自动化办公技能**：掌握利用 Python 的 `pathlib` 库进行文件系统级的递归搜索与操作。
2. **PDF 文档处理**：学习使用 `pypdf` 库对 PDF 文件进行读取、合并及添加交互式书签。
3. **数据清洗与排序**：理解并实现自然排序算法，确保处理后的文件能够按照人类阅读习惯（如 Exp1, Exp2, Exp10）进行逻辑排序。
4. **代码鲁棒性设计**：学习如何通过自动重命名机制防止重名文件覆盖，建立稳健的文件管理体系。

2. 实验要求

此编写实验目的

- 软件要求：Windows 10及以上版本；RflySim工具链[1]。
- 硬件要求：笔记本/台式电脑1台[2]。

3. 实验地址

例程目录：[安装目录]\RflySimAPIs\1.RflySimIntro\0.ApiExps\10.PDF_File_Processing
例程目录内包含以下文件，用于实现 PDF 文件的自动化处理：

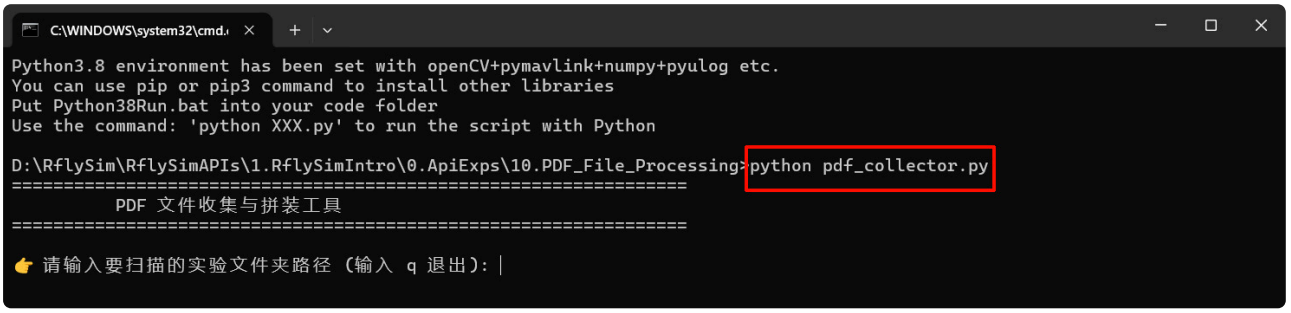
文件目录结构说明：

```
├── pdf_collector.py                # 实现核心逻辑的 Python 源代码
├── Python38Run.bat                 # 用于快捷运行 Python 脚本的批处理文件
```

4. 实验内容或步骤

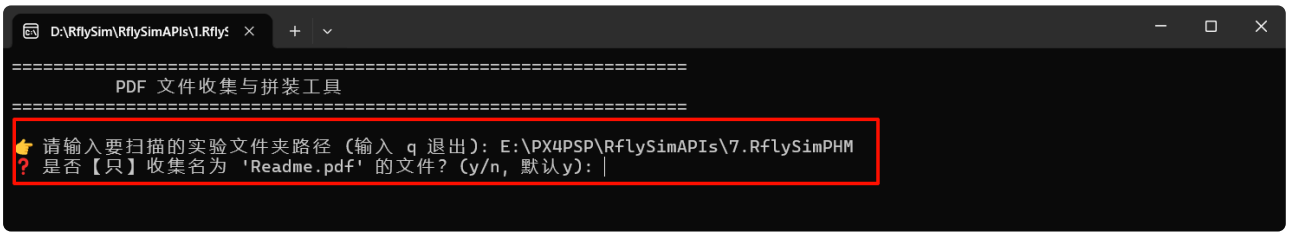
4.1 程序运行

双击 Python38Run.bat 文件，在弹出的对话框中输入：`python pdf_collector.py`。



4.2 输入路径

在弹出的对话框中输入具体的绝对路径，按下`Enter`即可快速收集具体的pdf文件，另外，也可以整针对性选择收集Readme.pdf文件，如下图所示：



5. 关键知识点

本项目通过 Python 实现了对超大规模实验体系下指导书的自动化归档。以下是本项目的技术架构与核心逻辑解析：

5.1 整体实现思路

程序的处理流程遵循以下五个阶段：

1. **环境预检**：程序启动时自动检测当前 Python 环境是否包含 `pypdf` 库。若缺失，将通过 `subprocess` 调用当前 Python 解释器自动完成静默安装。
2. **递归扫描**：在用户指定的根目录下，递归搜索所有子文件夹中的 PDF 文件。
3. **逻辑过滤与自然排序**：根据用户选择过滤出目标文件（如仅收集 Readme.pdf），并执行自然排序（Natural Sorting），确保存放在不同编号文件夹（如 1.Intro,

10.Advanced）中的文件能按数字大小递增。

4. **安全收集**：将分散的文件复制到单一汇总文件夹中。为了防止重名，采用 `{三位编号}_{父目录名}_{原文件名}` 的格式重命名。
5. **带书签拼接**：将收集到的文件合并为单一 PDF，并根据其来源路径自动生成侧边栏书签，提升阅读效率。

5.2 核心代码解析

(0) 自动依赖修复与环境隔离

为了降低用户的使用门槛，脚本内置了自动安装功能，并带有 `sys.frozen` 识别逻辑防止打包后的 EXE 出现死循环：

```
def check_dependencies():
    # 如果是在打包后的 EXE 中运行，跳过自动安装逻辑，防止死循环
    if getattr(sys, 'frozen', False): return
    try:
        import pypdf
    except ImportError:
        # 使用当前运行脚本的 Python 解释器进行自动安装
        subprocess.check_call([sys.executable, "-m", "pip", "install", "pypdf", "-i", "https://pypi.tuna.tsinghua.edu.cn/simple"])
```

(1) 自然排序算法的实现

为了解决传统字符串排序中 "10" 排在 "2" 之前的问题，程序引入了正则表达式解析数值：

```
def natural_sort_key(s, _nsre=re.compile('([0-9]+)')):
    """自然排序算法：使包含数字的字符串按人类逻辑排序"""
    return [int(text) if text.isdigit() else text.lower()
            for text in _nsre.split(str(s))]
```

这段代码确保了在合并 5v5 仿真与 10v10 仿真的文档时，顺序符合预期的实验梯度。

(2) 递归检索与安全过滤

利用 `pathlib.Path.rglob` 优雅地实现跨层级搜索，并规避生成的汇总文件夹，防止“套娃”循环：

```python
# 递归扫描获取所有 PDF 文件
all_pdf_files = list(source_path.rglob("*.pdf"))

# 过滤逻辑
for f in all_pdf_files:
    # 防止扫描到程序本身生成的汇总输出文件夹
    if str(output_dir) in str(f.resolve()):
        continue
    # 针对性收集 Readme.pdf
    if only_readme and f.name.lower() != "readme.pdf":
        continue
    pdf_files.append(f)
```

(3) 带书签的 PDF 合并

利用 `pypdf` 的 `PdfWriter` 对象在合并的同时注入树状书签：

```python
merger = PdfWriter()
for idx, pdf_path in enumerate(pdf_files, 1):
    # 提取上一级的目录名称，作为书签的关键标识
    parent_name = pdf_path.parent.name
    bookmark_title = f"{idx}. {parent_name} - {pdf_path.stem}"

    # 将文件载入合并器并添加书签项
    merger.append(pdf_path, outline_item=bookmark_title)
```

这种设计最大程度保留了文件的“地理信息”，即用户可以从合并后的文档一眼看出该章节属于哪个实验。

6. 参考资料

1. RflySim官方文档： https://rflysim.com/doc/zh/
2. pypdf 库官方技术文档： https://pypdf.readthedocs.io/
3. Python pathlib 模块文件操作指南：
   https://docs.python.org/3/library/pathlib\.html
4. 自然排序 (Natural Sort Order) 技术原理：
   https://en.wikipedia.org/wiki/Natural_sort_order

7. 常见问题

Q1：运行脚本时提示错误: 缺少必要的库。

A1：脚本已经内置了自动安装机制。如果您在运行 `.py` 源码时发现缺失库，程序会尝试自动为您安装。如果自动安装由于网络原因失败，请手动打开命令行窗口，并确保在 `Python38Run.bat` 设置的环境下运行：

```
pip install pypdf -i https://pypi.tuna.tsinghua.edu.cn/simple
```

Q2：输入的实验文件夹路径明明存在，但程序提示“路径不存在”。

A2：这通常是由于路径中的反斜杠字符被系统误识别，或者路径两端包含了多余的空格/引号。建议直接在弹出的输入栏右键粘贴路径，脚本已内置了对引号和首尾空格的自动清洗机制。

Q3：生成的 Merged.pdf 无法打开，或者提示“正在被占用”。

A3：请检查您的设备是否已经打开了之前生成的同名合并文档。在 Windows 系统下，如果该 PDF 正在被阅读器（如 Adobe Acrobat 或 Chrome 浏览器）查看，程序将无法对其进行覆盖写入。请关闭已打开的 PDF 后重试。

Q4：合并后的文档顺序与我期望的不一致。

A4：程序依赖于文件夹名称中的数字进行排序。请确保您的实验文件夹命名包含清晰的数字前缀（例如 e1.xxx，e2.xxx），程序将根据这些数字进行升序排列。

---

1. https://rflysim.com/ ↩
2. 推荐配置请见： https://rflysim.com/doc/zh/HowToInstall.pdf ↩