作为一名程序员，经常要搜一些教程，有的教程是在线的，不提供离线版本，这就有些局限了。那么同样作为一名程序员，遇到问题就应该解决它，今天就来将在线教程保存为PDF以供查阅。

1、网站介绍

2、准备工作

2.1 软件安装

2.2 库安装

3、爬取内容

3.1 获取教程名称

3.2 获取目录及对应网址

3.3 获取章节内容

3.4 保存pdf

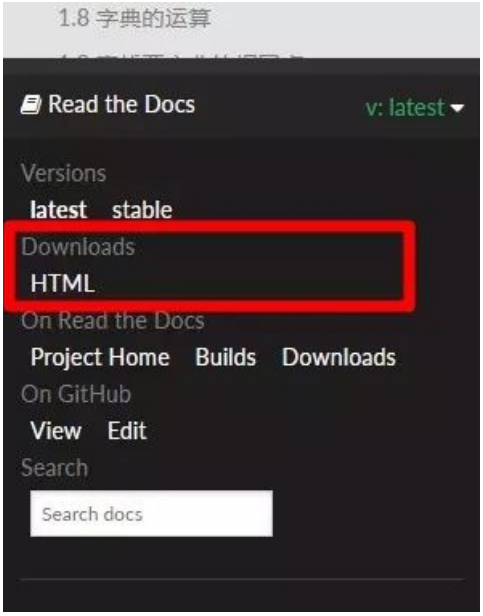
3.5 合并pdf

# 1、网站介绍

之前再搜资料的时候经常会跳转到如下图所示的在线教程：


包括一些github的项目也纷纷将教程链接指向这个网站。经过一番查找，该网站是一个可以创建、托管和浏览文档的网站，其网址为：https://readthedocs.org 。在上面可以找到很多优质的资源。

该网站虽然提供了下载功能，但是有些教程并没有提供PDF格式文件的下载，如图：




该教程只提供了 HTML格式文件的下载，还是不太方便查阅，那就让我们动手将其转成PDF吧！

# 2、准备工作

## 2.1 软件安装

由于我们是要把html转为pdf，所以需要手动**wkhtmltopdf** 。Windows平台直接在 http://wkhtmltopdf.org/downloads.html 下载稳定版的 wkhtmltopdf 进行安装，安装完成之后把该程序的执行路径加入到系统环境 $PATH 变量中，否则 pdfkit 找不到 wkhtmltopdf 就出现错误 “No wkhtmltopdf executable found”。Ubuntu 和 CentOS 可以直接用命令行进行安装

```
$ sudo apt-get install wkhtmltopdf  # ubuntu
$ sudo yum intsall wkhtmltopdf      # centos
```

## 2.2 库安装

- pip install requests # 用于网络请求
- pip install beautifulsoup4 # 用于操作html
- pip install pdfkit # wkhtmltopdf 的Python封装包
- pip install PyPDF2 # 用于合并pdf

# 3、爬取内容

本文的目标网址为：http://python3-cookbook.readthedocs.io/zh_CN/latest/ 。

## 3.1 获取教程名称

页面的左边一栏为目录，按F12调出开发者工具并按以下步骤定位到目录元素：

① 点击开发者工具左上角"**选取页面元素**"按钮；

② 用鼠标点击左上角教程名称处。

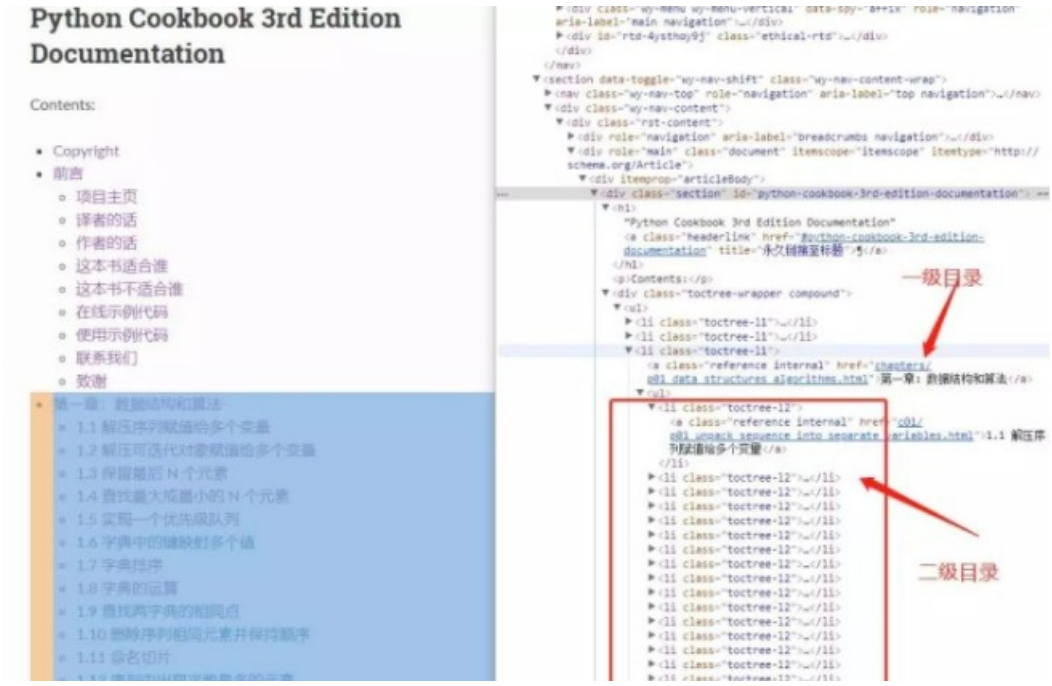
通过以上步骤即可定位到目录元素，用图说明：


从图看到我们需要的教程名称包含在 之间的a标签里。假设我们已经获取到了网页内容为html，可以使用以下代码获取该内容：

```
book_name = soup.find('div', class_='wy-side-nav-search').a.text
```

# 3.2 获取目录及对应网址

使用与 2.1 相同的步骤来获取：




从图看到我们需要的目录包含在 之间，<li></li>标签里为一级目录及网址；<li></li>标签里为二级目录及网址。当然这个url是相对的url，前面还要拼接http://python3-cookbook.readthedocs.io/zh_CN/latest/。

使用BeautifulSoup进行数据的提取：

```python
# 全局变量
base_url = 'http://python3-cookbook.readthedocs.io/zh_CN/latest/'
book_name = ''
chapter_info = []

def parse_title_and_url(html):
    """
    解析全部章节的标题和url
    :param html: 需要解析的网页内容
    :return None
    """
    soup = BeautifulSoup(html, 'html.parser')

    # 获取书名
    book_name = soup.find('div', class_='wy-side-nav-search').a.text
    menu = soup.find_all('div', class_='section')
    chapters = menu[0].div.ul.find_all('li', class_='toctree-l1')
    for chapter in chapters:
        info = {}
        # 获取一级标题和url
        # 标题中含有'/'和'*'会保存失败
        info['title'] = chapter.a.text.replace('/', '').replace('*', '
        info['url'] = base_url + chapter.a.get('href')
        info['child_chapters'] = []
```

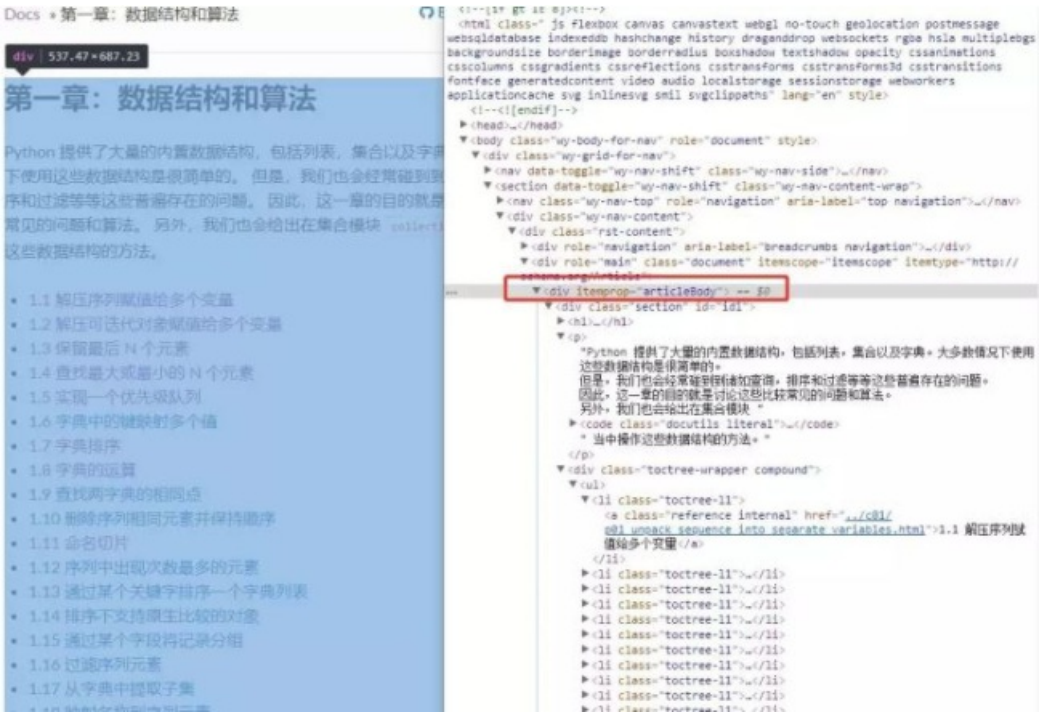
```python
        # 获取二级标题和url
        if chapter.ul is not None:
            child_chapters = chapter.ul.find_all('li')
            for child in child_chapters:
                url = child.a.get('href')
                # 如果在url中存在'#'，则此url为页面内链接，不会跳转到其他页
                # 所以不需要保存
                if '#' not in url:
                    info['child_chapters'].append({
                        'title': child.a.text.replace('/', '').replace
                        'url': base_url + child.a.get('href'),
                    })

        chapter_info.append(info)
```

代码中定义了两个全局变量来保存信息。章节内容保存在**chapter_info**列表里，里面包含了层级结构，大致结构为：

```
[
    {
        'title': 'first_level_chapter',
        'url': 'www.xxxxxx.com',
        'child_chapters': [
            {
                'title': 'second_level_chapter',
                'url': 'www.xxxxxx.com',
            }
            ...
        ]
    }
    ...
]
```

3.3 获取章节内容

还是同样的方法定位章节内容：




05.获取章节内容

代码中我们通过itemprop这个属性来定位，好在一级目录内容的元素位置和二级目录内容的元素位置相同，省去了不少麻烦。

```
html_template = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
</head>
<body>
{content}
</body>
</html>
"""

def get_content(url):
    """
    解析URL，获取需要的html内容
    :param url: 目标网址
    :return: html
    """
    html = get_one_page(url)
    soup = BeautifulSoup(html, 'html.parser')
    content = soup.find('div', attrs={'itemprop': 'articleBody'})
    html = html_template.format(content=content)
    return html
```

## 3.4 保存pdf

```
def save_pdf(html, filename):
    """
    把所有html文件保存到pdf文件
    :param html:  html内容
    :param file_name: pdf文件名
    :return:
    """
    options = {
        'page-size': 'Letter',
        'margin-top': '0.75in',
        'margin-right': '0.75in',
        'margin-bottom': '0.75in',
        'margin-left': '0.75in',
        'encoding': "UTF-8",
        'custom-header': [
            ('Accept-Encoding', 'gzip')
        ],
        'cookie': [
            ('cookie-name1', 'cookie-value1'),
            ('cookie-name2', 'cookie-value2'),
        ],
        'outline-depth': 10,
    }

    pdfkit.from_string(html, filename, options=options)
```

```python
def parse_html_to_pdf():
    """
    解析URL，获取html，保存成pdf文件
    :return: None
    """
    try:
        for chapter in chapter_info:
            ctitle = chapter['title']
            url = chapter['url']
            # 文件夹不存在则创建（多级目录）
            dir_name = os.path.join(os.path.dirname(__file__), 'gen',
            if not os.path.exists(dir_name):
                os.makedirs(dir_name)

            html = get_content(url)

            padf_path = os.path.join(dir_name, ctitle + '.pdf')
            save_pdf(html, os.path.join(dir_name, ctitle + '.pdf'))

            children = chapter['child_chapters']
            if children:
                for child in children:
                    html = get_content(child['url'])
                    pdf_path = os.path.join(dir_name, child['title'] +
                    save_pdf(html, pdf_path)

    except Exception as e:
        print(e)
```

3.5 合并pdf

经过上一步，所有章节的pdf都保存下来了，最后我们希望留一个pdf，就需要合并所有pdf并删除单个章节pdf。

```python
from PyPDF2 import PdfFileReader, PdfFileWriter

def merge_pdf(infnList, outfn):
    """
    合并pdf
    :param infnList: 要合并的PDF文件路径列表
    :param outfn: 保存的PDF文件名
    :return: None
    """
    pagenum = 0
    pdf_output = PdfFileWriter()

    for pdf in infnList:
        # 先合并一级目录的内容
        first_level_title = pdf['title']
        dir_name = os.path.join(os.path.dirname(
            __file__), 'gen', first_level_title)
        padf_path = os.path.join(dir_name, first_level_title + '.pdf')

        pdf_input = PdfFileReader(open(padf_path, 'rb'))
        # 获取 pdf 共用多少页
        page_count = pdf_input.getNumPages()
        for i in range(page_count):
            pdf_output.addPage(pdf_input.getPage(i))

        # 添加书签
        parent_bookmark = pdf_output.addBookmark(
            first_level_title, pagenum=pagenum)

        # 页数增加
        pagenum += page_count
```

```python
        # 存在子章节
        if pdf['child_chapters']:
            for child in pdf['child_chapters']:
                second_level_title = child['title']
                padf_path = os.path.join(dir_name, second_level_title

                pdf_input = PdfFileReader(open(padf_path, 'rb'))
                # 获取 pdf 共用多少页
                page_count = pdf_input.getNumPages()
                for i in range(page_count):
                    pdf_output.addPage(pdf_input.getPage(i))

                # 添加书签
                pdf_output.addBookmark(
                    second_level_title, pagenum=pagenum, parent=parent_
                # 增加页数
                pagenum += page_count

    # 合并
    pdf_output.write(open(outfn, 'wb'))
    # 删除所有章节文件
    shutil.rmtree(os.path.join(os.path.dirname(__file__), 'gen'))
```

本来PyPDF2库中有一个类`PdfFileMerger`专门用来合并pdf，但是在合并过程中会抛出异常，网上有人也遇到同样的问题，解

决办法是修改库源码，本着“不动库源码”的理念，毅然选择了上面这种比较笨的办法，代码还是比较好理解的。

经过以上几个步骤，我们想要的pdf文件已经生成，一起来欣赏一下劳动成果：