

如何使用python提取pdf表格及文本，并保存到excel

Python绿色通道 1 week ago

The following article is from Python大数据分析 Author 朱卫军



Python大数据分析

分享python编程、可视化设计、大数据分析、机器学习等技术以及数据分析案例，包括但不...

↑ 关注 + 星标，每天学Python新技能

后台回复【大礼包】送你Python自学大礼包



pdf是一种便携式文档格式，由Adobe公司设计。因为不受平台限制，且方便保存和传输，所以pdf非常受欢迎。

目前市场上有很多pdf工具，大部分是阅读类，也有支持对pdf的修改、转换等功能，但这部分工具不少是收费的。

这次介绍一个开源python工具库-pdfplumber，可以方便地获取pdf的各种信息，包括文本、表格、图表、尺寸等。

pdfplumber在github上有英文官方文档，后面我们会捡重点讲解，先看下如何用pdfplumber提取pdf表格？

以NBA 2020-2021 常规赛数据作为范例，pdf表格如下：

排名	球队	场数	场均得分	场均篮板	场均助攻	场均抢断	场均盖帽	犯规
1	犹他 爵士	32	116	48.4	23.7	6.3	5.6	18.8
2	密尔沃基 雄鹿	33	120.1	48.3	26.2	8.2	4.8	18.1
3	纽约 尼克斯	33	104.7	47.1	20.9	6.5	4.8	21
4	奥兰多 魔术	33	104.8	46.8	22.5	7.1	4.3	16.9
5	新奥尔良 鹈鹕	32	115.5	46.1	25.3	7.5	4	17.5
6	费城 76人	33	113.9	45.8	23.3	8.5	6.1	20
7	亚特兰大 老鹰	32	113.7	45.4	24.4	6.8	4.8	20
8	洛杉矶 湖人	33	111.5	45.2	24.7	7.1	6.2	18.6
9	圣安东尼奥 马刺	28	110.7	45.1	25	7.3	5	17
10	俄克拉荷马城 雷霆	32	106.7	44.7	23.2	7.1	4.3	18.3
11	华盛顿 奇才	30	114.5	44.6	25.1	7.4	3.6	22
12	孟菲斯 灰熊	28	110.8	44.6	27.4	9.6	4.4	18.5
13	波特兰 开拓者	31	114.8	44.5	19.9	7.2	5.1	19.7
14	芝加哥 公牛	31	115.3	44.3	26.2	7.2	4.4	20.3
15	夏洛特 黄蜂	31	111.3	44.3	26.9	8	4.7	18.5
16	洛杉矶 快船	34	115.2	44.2	24.2	7.2	4.5	19
17	菲尼克斯 太阳	31	113.5	43.8	26.6	6.4	4.3	20.1
18	波士顿 凯尔特人	32	110.7	43.8	22.1	8.2	5.1	21.2
19	布鲁克林 篮网	34	121.4	43.7	27	6.6	5.4	19.2
20	金州 勇士	33	114.2	43.5	27.8	7.8	5.2	21.8
21	丹佛 掘金	32	115.2	43.4	26.3	8.3	4.4	19.7
22	克利夫兰 骑士	33	104.3	43.4	23.4	8.3	5.3	19.2
23	休斯顿 火箭	30	108.6	43.3	22.8	8	5.7	20.1
24	明尼苏达 森林狼	33	108.2	43	25.4	8.5	5.7	21.7
25	印第安纳 步行者	30	113.2	42.8	26.6	8.4	6.1	20.6
26	底特律 活塞	32	108	42.7	24.2	7.8	5.1	20.7
27	萨克拉门托 国王	32	114.1	42.6	25.7	6.8	5	19.8
28	达拉斯 独行侠	31	111.9	42.6	22.5	6.3	4.5	20.8
29	迈阿密 热火	32	107.2	42.6	26.1	6.9	3.9	19.8
30	多伦多 猛龙	33	112.6	41.7	25	8.7	5.6	22

第一步：使用pdfplumber提取表格文本

```
# 导入pdfplumber
import pdfplumber

# 读取pdf文件，保存为pdf实例
pdf = pdfplumber.open("E:\\nba.pdf")

# 访问第二页
first_page = pdf.pages[1]

# 自动读取表格信息，返回列表
table = first_page.extract_table()

table
```

输出：

```
[[['排名', '球队', '场数', '得分', '篮板', '助攻', '抢断', '盖帽', '犯规'],
 ['1', '犹他 爵士', '32', '116', '48.4', '23.7', '6.3', '5.6', '18.8'],
 ['2', '密尔沃基 雄鹿', '33', '120.1', '48.3', '26.2', '8.2', '4.8', '18.1'],
 ['3', '纽约 尼克斯', '33', '104.7', '47.1', '20.9', '6.5', '4.8', '21'],
 ['4', '奥兰多 魔术', '33', '104.8', '46.8', '22.5', '7.1', '4.3', '16.9'],
 ['5', '新奥尔良 鹈鹕', '32', '115.5', '46.1', '25.3', '7.5', '4', '17.5'],
 ['6', '费城 76人', '33', '113.9', '45.8', '23.3', '8.5', '6.1', '20'],
 ['7', '亚特兰大 老鹰', '32', '113.7', '45.4', '24.4', '6.8', '4.8', '20'],
 ['8', '洛杉矶 湖人', '33', '111.5', '45.2', '24.7', '7.1', '6.2', '18.6'],
 ['9', '圣安东尼奥 马刺', '28', '110.7', '45.1', '25', '7.3', '5', '17'],
 ['10', '俄克拉荷马城 雷霆', '32', '106.7', '44.7', '23.2', '7.1', '4.3', '18.3'],
 ['11', '华盛顿 奇才', '30', '114.5', '44.6', '25.1', '7.4', '3.6', '22'],
 ['12', '孟菲斯 灰熊', '28', '110.8', '44.6', '27.4', '9.6', '4.4', '18.5'],
 ['13', '波特兰 开拓者', '31', '114.8', '44.5', '19.9', '7.2', '5.1', '19.7'],
 ['14', '芝加哥 公牛', '31', '115.3', '44.3', '26.2', '7.2', '4.4', '20.3'],
 ['15', '夏洛特 黄蜂', '31', '111.3', '44.3', '26.9', '8', '4.7', '18.5'],
 ['16', '洛杉矶 快船', '34', '115.2', '44.2', '24.2', '7.2', '4.5', '19'],
 ['17', '菲尼克斯 太阳', '31', '113.5', '43.8', '26.6', '6.4', '4.3', '20.1'],
 ['18', '波士顿 凯尔特人', '32', '110.7', '43.8', '22.1', '8.2', '5.1', '21.2'],
 ['19', '布鲁克林 篮网', '34', '121.4', '43.7', '27', '6.6', '5.4', '19.2'],
 ['20', '金州 勇士', '33', '114.2', '43.5', '27.8', '7.8', '5.2', '21.8'],
 ['21', '丹佛 掘金', '32', '115.2', '43.4', '26.3', '8.3', '4.4', '19.7'],
 ['22', '克利夫兰 骑士', '33', '104.3', '43.4', '23.4', '8.3', '5.3', '19.2'],
 ['23', '休斯顿 火箭', '30', '108.6', '43.3', '22.8', '8', '5.7', '20.1'],
 ['24', '明尼苏达 森林狼', '33', '108.2', '43', '25.4', '8.5', '5.7', '21.7'],
 ['25', '印第安纳 步行者', '30', '113.2', '42.8', '26.6', '8.4', '6.1', '20.6'],
 ['26', '底特律 活塞', '32', '108', '42.7', '24.2', '7.8', '5.1', '20.7'],
 ['27', '萨克拉门托 国王', '32', '114.1', '42.6', '25.7', '6.8', '5', '19.8'],
 ['28', '达拉斯 独行侠', '31', '111.9', '42.6', '22.5', '6.3', '4.5', '20.8'],
 ['29', '迈阿密 热火', '32', '107.2', '42.6', '26.1', '6.9', '3.9', '19.8'],
 ['30', '多伦多 猛龙', '33', '112.6', '41.7', '25', '8.7', '5.6', '22']]]
```

第二步：整理成dataframe格式，保存为excel

```
import pandas as pd

# 将列表转为df
table_df = pd.DataFrame(table_2[1:], columns=table_2[0])

# 保存excel
table_df.to_excel('test.xlsx')
```

table_df

输出：

	排名	球队	场数	场均得分	场均篮板	场均助攻	场均抢断	场均盖帽	犯规
0	1	犹他 爵士	32	116	48.4	23.7	6.3	5.6	18.8
1	2	密尔沃基 雄鹿	33	120.1	48.3	26.2	8.2	4.8	18.1
2	3	纽约 尼克斯	33	104.7	47.1	20.9	6.5	4.8	21
3	4	奥兰多 魔术	33	104.8	46.8	22.5	7.1	4.3	16.9
4	5	新奥尔良 鹈鹕	32	115.5	46.1	25.3	7.5	4	17.5
5	6	费城 76人	33	113.9	45.8	23.3	8.5	6.1	20
6	7	亚特兰大 老鹰	32	113.7	45.4	24.4	6.8	4.8	20
7	8	洛杉矶 湖人	33	111.5	45.2	24.7	7.1	6.2	18.6
8	9	圣安东尼奥 马刺	28	110.7	45.1	25	7.3	5	17
9	10	俄克拉荷马城 雷霆	32	106.7	44.7	23.2	7.1	4.3	18.3
10	11	华盛顿 奇才	30	114.5	44.6	25.1	7.4	3.6	22
11	12	孟菲斯 灰熊	28	110.8	44.6	27.4	9.6	4.4	18.5
12	13	波特兰 开拓者	31	114.8	44.5	19.9	7.2	5.1	19.7
13	14	芝加哥 公牛	31	115.3	44.3	26.2	7.2	4.4	20.3
14	15	夏洛特 黄蜂	31	111.3	44.3	26.9	8	4.7	18.5
15	16	洛杉矶 快船	34	115.2	44.2	24.2	7.2	4.5	19
16	17	菲尼克斯 太阳	31	113.5	43.8	26.6	6.4	4.3	20.1
17	18	波士顿 凯尔特人	32	110.7	43.8	22.1	8.2	5.1	21.2
18	19	布鲁克林 篮网	34	121.4	43.7	27	6.6	5.4	19.2
19	20	金州 勇士	33	114.2	43.5	27.8	7.8	5.2	21.8
20	21	丹佛 掘金	32	115.2	43.4	26.3	8.3	4.4	19.7
21	22	克利夫兰 骑士	33	104.3	43.4	23.4	8.3	5.3	19.2
22	23	休斯顿 火箭	30	108.6	43.3	22.8	8	5.7	20.1
23	24	明尼苏达 森林狼	33	108.2	43	25.4	8.5	5.7	21.7
24	25	印第安纳 步行者	30	113.2	42.8	26.6	8.4	6.1	20.6
25	26	底特律 活塞	32	108	42.7	24.2	7.8	5.1	20.7
26	27	萨克拉门托 国王	32	114.1	42.6	25.7	6.8	5	19.8
27	28	达拉斯 独行侠	31	111.9	42.6	22.5	6.3	4.5	20.8
28	29	迈阿密 热火	32	107.2	42.6	26.1	6.9	3.9	19.8
29	30	多伦多 猛龙	33	112.6	41.7	25	8.7	5.6	22

一个小小的脚本，不到十行代码，便将pdf表格提取并转化为dataframe格式，最终保存到excel。

有个初步认知后，接下来详细讲讲pdfplumber的安装、导入、api接口等信息。

pdfplumber简介

前面已经介绍过pdfplumber的用途，也用一个小案例展示了如何提取表格，我觉得对于pdfplumber只需要了解三点就可以。

- 1、它是一个纯python第三方库，适合python 3.x版本
- 2、它用来查看pdf各类信息，能有效提取文本、表格
- 3、它不支持修改或生成pdf，也不支持对pdf扫描件的处理

Github地址<https://github.com/jsvine/pdfplumber>

pdfplumber安装和导入

同其他python库一样，pdfplumber支持使用pip安装，在命令行输入：

```
pip install pdfplumber
```

如果遇到安装慢的问题，可以替换镜像源，会快很多。

pdfplumber安装后，用import导入即可使用：

```
import pdfplumber  
....
```

pdfplumber简单使用

pdfplumber中有两个基础类，PDF和Page。看字面意思能猜出，前者是处理整个文档，后者是处理页面。

「pdfplumber.PDF类」

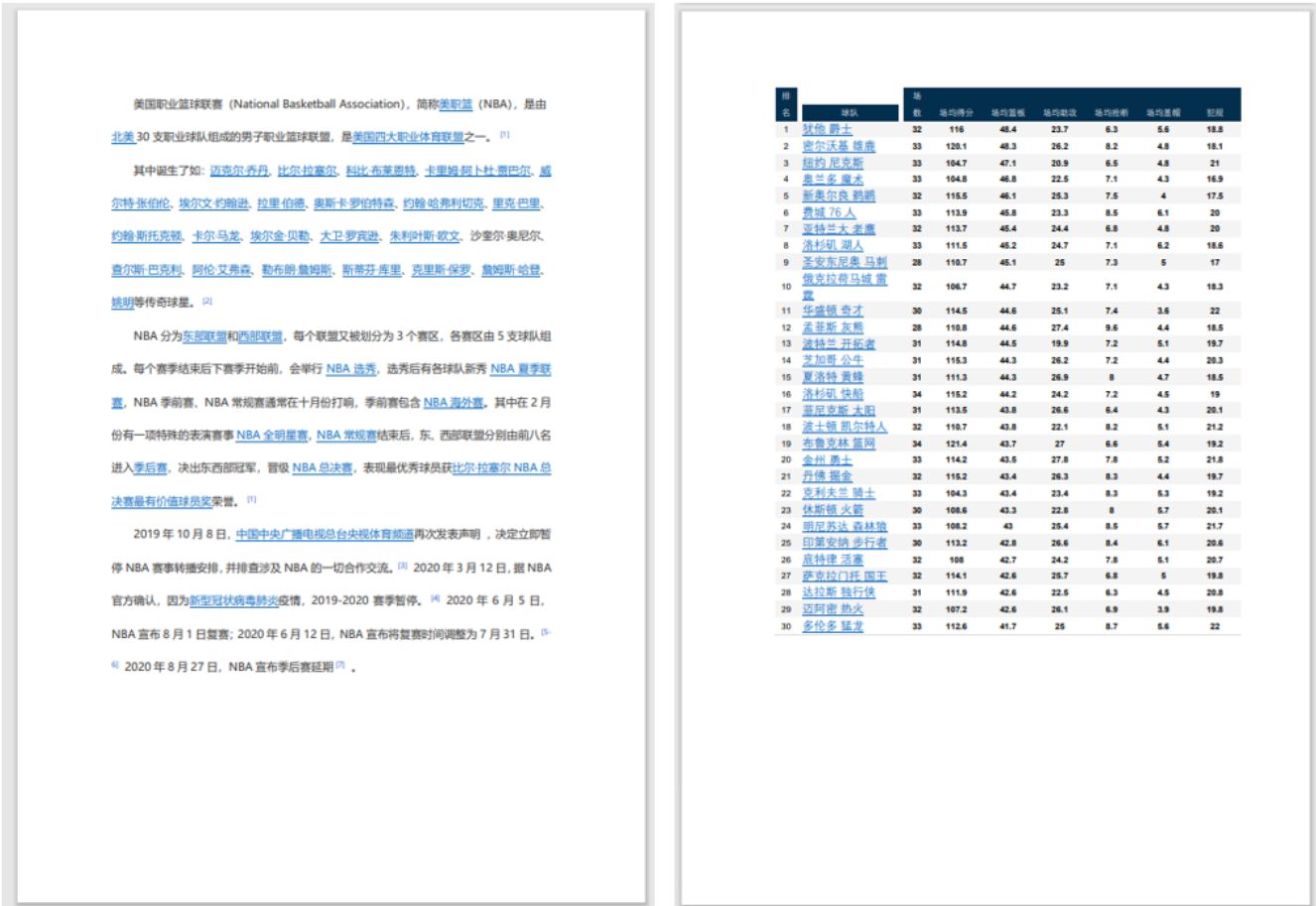
属性	描述
<code>.metadatas</code>	获取pdf基础信息，返回字典
<code>.pages</code>	一个包含pdfplumber.Page实例的列表，每一个实例代表pdf每一页的信息。

「pdfplumber.Page类」

这是pdfplumber的核心功能，对pdf的大部分操作都是基于这个类，包括提取文本、表格、尺寸等。

这里暂不一一列举它的属性和方法。

通过一个简单的案例，就可以明白它们的作用。示例pdf文档，共两页：



1. 读取pdf

```
# 导入pdfplumber
import pdfplumber

# 读取pdf文件，返回pdfplumber.PDF类的实例
pdf = pdfplumber.open("e:\\nba2.pdf")
```

2. 获取该pdf文档的信息

```
# 通过pdfplumber.PDF类的metadata属性获取pdf信息
pdf.metadata
```

输出：

```
{'Author': 'Zhu Weijun',
 'Creator': 'Microsoft® Word 适用于 Microsoft 365',
 'CreationDate': "D:20210227174711+08'00'",
 'ModDate': "D:20210227174711+08'00'",
 'Producer': 'Microsoft® Word 适用于 Microsoft 365'}
```

这些是pdf的基础信息，包括作者、来源、日期等。

3. 总页数

```
# 通过pdfplumber.PDF类的metadata属性获取pdf页数
len(pdf.pages)
```

4. 读取第一页的页宽、页高等信息

```
# 第一页pdfplumber.Page实例
first_page = pdf.pages[0]

# 查看页码
print('页码: ', first_page.page_number)

# 查看页宽
print('页宽: ', first_page.width)
```

```
# 查看页高
print('页高: 'first_page.height)
```

输出：

```
页码： 1
页宽： 595.320
页高： 841.920
```

5. 读取第一页的文本

```
# 读取文本
text = first_page.extract_text()
print(text)
```

输出：

美国职业篮球联赛（National Basketball Association），简称美职篮（NBA），是由北美30支职业球队组成的男子职业篮球联盟，是美国四大职业体育联盟之一。 [1]
其中诞生了如：迈克尔·乔丹、比尔·拉塞尔、科比·布莱恩特、卡里姆·阿卜杜·贾巴尔、威尔特·张伯伦、埃尔文·约翰逊、拉里·伯德、奥斯卡·罗伯特森、约翰·哈弗利切克、里克·巴里、约翰·斯托克顿、卡尔·马龙、埃尔金·贝勒、大卫·罗宾逊、朱利叶斯·欧文、沙奎尔·奥尼尔、查尔斯·巴克利、阿伦·艾弗森、勒布朗·詹姆斯、斯蒂芬·库里、克里斯·保罗、詹姆斯·哈登、姚明等传奇球星。 [2]
NBA分为东部联盟和西部联盟，每个联盟又被划分为3个赛区，各赛区由5支球队组成。每个赛季结束后下赛季开始前，会举行 NBA 选秀，选秀后有各球队新秀 NBA 夏季联赛，NBA季前赛、NBA常规赛通常在十月份打响，季前赛包含NBA海外赛。其中在2月份有一项特殊的表演赛事NBA全明星赛，NBA常规赛结束后，东、西部联盟分别由前八名进入季后赛，决出东西部冠军，晋级NBA总决赛，表现最优秀球员获比尔·拉塞尔NBA总决赛最有价值球员奖荣誉。 [1]
2019年10月8日，中国中央广播电视总台央视体育频道再次发表声明，决定立即暂停NBA赛事转播安排，并排查涉及NBA的一切合作交流。 [3] 2020年3月12日，据NBA官方确认，因为新型冠状病毒肺炎疫情，2019-2020 赛季暂停。 [4] 2020 年 6 月 5 日，NBA宣布8月1日复赛；2020年6月12日，NBA宣布将复赛时间调整为7月31日。 [5-6] 2020年8月27日，NBA宣布季后赛延期 [7] 。

6. 读取第二页的表格

```
import pandas as pd

# 第二页pdfplumber.Page实例
first_page = pdf.pages[1]
```



```
# 自动读取表格信息，返回列表
table = first_page.extract_tables()

# 将列表转为df
table_df = pd.DataFrame(table_2[1:],columns=table_2[0])

table_df
```

	排名	球队	场数	场均得分	场均篮板	场均助攻	场均抢断	场均盖帽	犯规
0	1	犹他 爵士	32	116	48.4	23.7	6.3	5.6	18.8
1	2	密尔沃基 雄鹿	33	120.1	48.3	26.2	8.2	4.8	18.1
2	3	纽约 尼克斯	33	104.7	47.1	20.9	6.5	4.8	21
3	4	奥兰多 魔术	33	104.8	46.8	22.5	7.1	4.3	16.9
4	5	新奥尔良 鹈鹕	32	115.5	46.1	25.3	7.5	4	17.5
5	6	费城 76人	33	113.9	45.8	23.3	8.5	6.1	20
6	7	亚特兰大 老鹰	32	113.7	45.4	24.4	6.8	4.8	20
7	8	洛杉矶 湖人	33	111.5	45.2	24.7	7.1	6.2	18.6
8	9	圣安东尼奥 马刺	28	110.7	45.1	25	7.3	5	17
9	10	俄克拉荷马城 雷霆	32	106.7	44.7	23.2	7.1	4.3	18.3
10	11	华盛顿 奇才	30	114.5	44.6	25.1	7.4	3.6	22
11	12	孟菲斯 灰熊	28	110.8	44.6	27.4	9.6	4.4	18.5
12	13	波特兰 开拓者	31	114.8	44.5	19.9	7.2	5.1	19.7
13	14	芝加哥 公牛	31	115.3	44.3	26.2	7.2	4.4	20.3
14	15	夏洛特 黄蜂	31	111.3	44.3	26.9	8	4.7	18.5
15	16	洛杉矶 快船	34	115.2	44.2	24.2	7.2	4.5	19
16	17	菲尼克斯 太阳	31	113.5	43.8	26.6	6.4	4.3	20.1
17	18	波士顿 凯尔特人	32	110.7	43.8	22.1	8.2	5.1	21.2
18	19	布鲁克林 篮网	34	121.4	43.7	27	6.6	5.4	19.2
19	20	金州 勇士	33	114.2	43.5	27.8	7.8	5.2	21.8
20	21	丹佛 掘金	32	115.2	43.4	26.3	8.3	4.4	19.7
21	22	克利夫兰 骑士	33	104.3	43.4	23.4	8.3	5.3	19.2
22	23	休斯顿 火箭	30	108.6	43.3	22.8	8	5.7	20.1
23	24	明尼苏达 森林狼	33	108.2	43	25.4	8.5	5.7	21.7
24	25	印第安纳 步行者	30	113.2	42.8	26.6	8.4	6.1	20.6
25	26	底特律 活塞	32	108	42.7	24.2	7.8	5.1	20.7
26	27	萨克拉门托 国王	32	114.1	42.6	25.7	6.8	5	19.8
27	28	达拉斯 独行侠	31	111.9	42.6	22.5	6.3	4.5	20.8
28	29	迈阿密 热火	32	107.2	42.6	26.1	6.9	3.9	19.8
29	30	圣路易 红雀	33	110.7	42.5	25.5	7.5	5.5	20.5

pdfplumber提取表格有很多的细节需要处理，这里给到的范例表格线框比较规范，所以能很简单的提取，但对于线框不完全（包含无线框）的表格，其效果就差了不少。

在实际项目所需处理的pdf文档中，线框完全及不完全的表格都比较多，为了能够理解pdfplumber实现表格抽取的原理和方法，我们需要去细究相关参数的设置。

正如案例所示，pdfplumber.Page对象的`.extract_table()`方法可以提取表格，返回从页面上最大的表中提取的文本，以列表列表的形式显示，结构为row -> cell。


「表格抽取参数设置」

默认情况下，`extract_table`使用页面的垂直和水平线（或矩形边缘）作为单元格分隔符。该方法可以通过`table_settings`参数进行高度自定义。可能的设置及其默认值：

```
{
    "vertical_strategy": "lines",
    "horizontal_strategy": "lines",
    "explicit_vertical_lines": [],
    "explicit_horizontal_lines": [],
    "snap_tolerance": 3,
    "join_tolerance": 3,
    "edge_min_length": 3,
    "min_words_vertical": 3,
    "min_words_horizontal": 1,
    "keep_blank_chars": False,
    "text_tolerance": 3,
    "text_x_tolerance": None,
    "text_y_tolerance": None,
    "intersection_tolerance": 3,
    "intersection_x_tolerance": None,
    "intersection_y_tolerance": None,
}
```

设置	描述
"vertical_strategy"	"lines", "lines_strict", "text", 或 "explicit", 具体含义见下文。
"horizontal_strategy"	"lines", "lines_strict", "text", 或 "explicit", 具体含义见下文。
"explicit_vertical_lines"	垂直线列表，用于明确划分表格中的单元格。可以与以上任何策略结合使用。列表中的项目应为数字（表示页面的整个高度的线条的 x 坐标）或 line / rect / curve 对象。
"explicit_horizontal_lines"	明确划分表中单元格的水平线列表。可以与以上任何策略结合使用。列表中的项目应为数字（表示页面的整个高度的线条的 y 坐标）或 line / rect / curve 对象。
"snap_tolerance"	snap_tolerance 像素内的平行线将被“捕捉”到相同的水平或垂直位置。
"join_tolerance"	同一条直线上的线段（其末端在彼此的 join_tolerance 之内）将被“拼接”为单个线段。
"edge_min_length"	短于 edge_min_length 的边将在尝试重建表之前被丢弃。
"min_words_vertical"	使用 "vertical_strategy": " text" 时，至少 min_words_vertical 个单词必须共享相同的对齐方式。
"min_words_horizontal"	使用 "horizontal_strategy": " text" 时，至少 min_words_horizontal 个单词必须共享相同的对齐方式。
"keep_blank_chars"	使用 text 策略时，将 " " 字符作为单词的一部分而不是单词分隔符。
"text_tolerance", "text_x_tolerance", "text_y_tolerance"	当 text 策略搜索单词时，它将期望每个单词中的各个字母相差不超过 text_tolerance 像素。
"intersection_tolerance", "intersection_x_tolerance", "intersection_y_tolerance"	将边缘合并为单元格时，正交边缘必须在 intersection_tolerance 像素内才能被视为相交。

pdfplumber支持对图表进行可视化调试，能输出图像，显示如何提取表。

						
Notice Date	Effective	Received	Company	City	No. Of	Layoff/Closure
06/22/2015	03/25/2016	07/01/2015	Maxim Integrated Product	San Jose	150	Closure Permanent
06/30/2015	08/29/2015	07/01/2015	McGraw-Hill Education	Monterey	137	Layoff Unknown at this time
06/30/2015	08/30/2015	07/01/2015	Long Beach Memorial Medical Center	Long Beach	90	Layoff Permanent
07/01/2015	09/02/2015	07/01/2015	Leidos	El Segundo	72	Layoff Permanent
07/01/2015	09/30/2016	07/01/2015	Bosch Healthcare Systems, Inc.	Palo Alto	55	Closure Permanent
06/29/2015	09/01/2015	07/02/2015	Encompass Digital Media, Inc.	Los Angeles	41	Closure Permanent
07/02/2015	07/06/2015	07/02/2015	Alphatec Spine	Carlsbad	99	Layoff Permanent
06/30/2015	08/07/2015	07/06/2015	Symantec Corporation	Mountain View	60	Layoff Permanent
06/30/2015	08/31/2015	07/06/2015	Fusion Contacts Centers, LLC	Santa Maria	50	Closure Permanent
06/30/2015	09/15/2015	07/06/2015	KLA-Tencor Corporation	Milpitas	213	Layoff Permanent
07/01/2015	09/04/2015	07/06/2015	Southern California Edison Company	San Clemente	100	Closure Permanent
07/02/2015	09/01/2015	07/06/2015	State Fish Company, Inc.	Wilmington	76	Closure Permanent
07/02/2015	09/04/2015	07/06/2015	Boeing Company	Long Beach	56	Layoff Unknown at this time
07/06/2015	09/04/2015	07/06/2015	Bridgepoint Education, Inc.	San Diego	7	Layoff Permanent
07/06/2015	09/04/2015	07/06/2015	Bridgepoint Education, Inc.	San Diego	15	Layoff Permanent
07/01/2015	06/29/2015	07/07/2015	BAE Systems	San Francisco	4	Layoff Temporary
07/01/2015	06/29/2015	07/07/2015	BAE Systems	San Francisco	78	Layoff Temporary
07/01/2015	09/07/2015	07/07/2015	Bay Bread LLC dba Bakery Los Angeles	San Fernando	50	Closure Permanent
07/01/2015	09/25/2015	07/07/2015	Bay Bread LLC dba New French Bakery	South San	121	Closure Permanent
07/02/2015	06/12/2015	07/07/2015	Hewlett-Packard Company	Palo Alto	65	Layoff Permanent
07/08/2015	09/06/2015	07/08/2015	Microsoft Corporation	San Diego	129	Layoff Permanent
06/25/2015	10/09/2015	07/10/2015	Aramark Healthcare Support Services,	Culver City	53	Closure Permanent
07/01/2015	09/10/2015	07/10/2015	Maxim Integrated Product	San Jose	20	Layoff Permanent
07/06/2015	09/04/2015	07/10/2015	ProCourier, Inc.	San Diego	22	Layoff Unknown at this time
07/06/2015	09/04/2015	07/10/2015	ProCourier, Inc.	Los Angeles	71	Layoff Unknown at this time
07/07/2015	09/04/2015	07/10/2015	ProCourier, Inc.	Irvine	22	Layoff Unknown at this time
07/09/2015	07/22/2015	07/10/2015	Berkeley Pyramid Alehouse	Berkeley	63	Closure Permanent
07/09/2015	09/14/2015	07/10/2015	Fireman's Fund Insurance Company	Novato	35	Layoff Permanent
06/30/2015	08/31/2015	07/13/2015	First Transit	San Bernardino	127	Layoff Permanent
06/30/2015	08/31/2015	07/13/2015	First Transit	Rancho	71	Layoff Permanent
07/10/2015	07/14/2015	07/13/2015	11 Main LLC	San Mateo	35	Closure Permanent
07/10/2015	07/14/2015	07/13/2015	11 Main LLC	Chico	44	Layoff Permanent
07/15/2015	07/15/2015	07/15/2015	TaylorMade Golf Company	Carlsbad	64	Layoff Permanent
07/08/2015	09/06/2015	07/16/2015	Southern California Edison Company	Rosemead	38	Layoff Permanent
07/14/2015	09/18/2015	07/20/2015	Actavis, Inc.	Corona	45	Layoff Permanent
07/17/2015	07/13/2015	07/21/2015	American Management Services LLC	Monterey	56	Closure Permanent

pdfplumber的独特之处

python中有很多库可以处理pdf，比如PyPDF2、pdfminer等，那pdfplumber的优势在哪呢？

首先，pdfplumber能轻松访问有关PDF对象的所有详细信息，且用于提取文本和表格的方法高级可定制，使用者可根据表格的具体形式来调整参数。

最关键的是pdfplumber作者持续在维护该库，而同样受欢迎的PyPDF2已经不再维护了。

获取源码



程序员导师



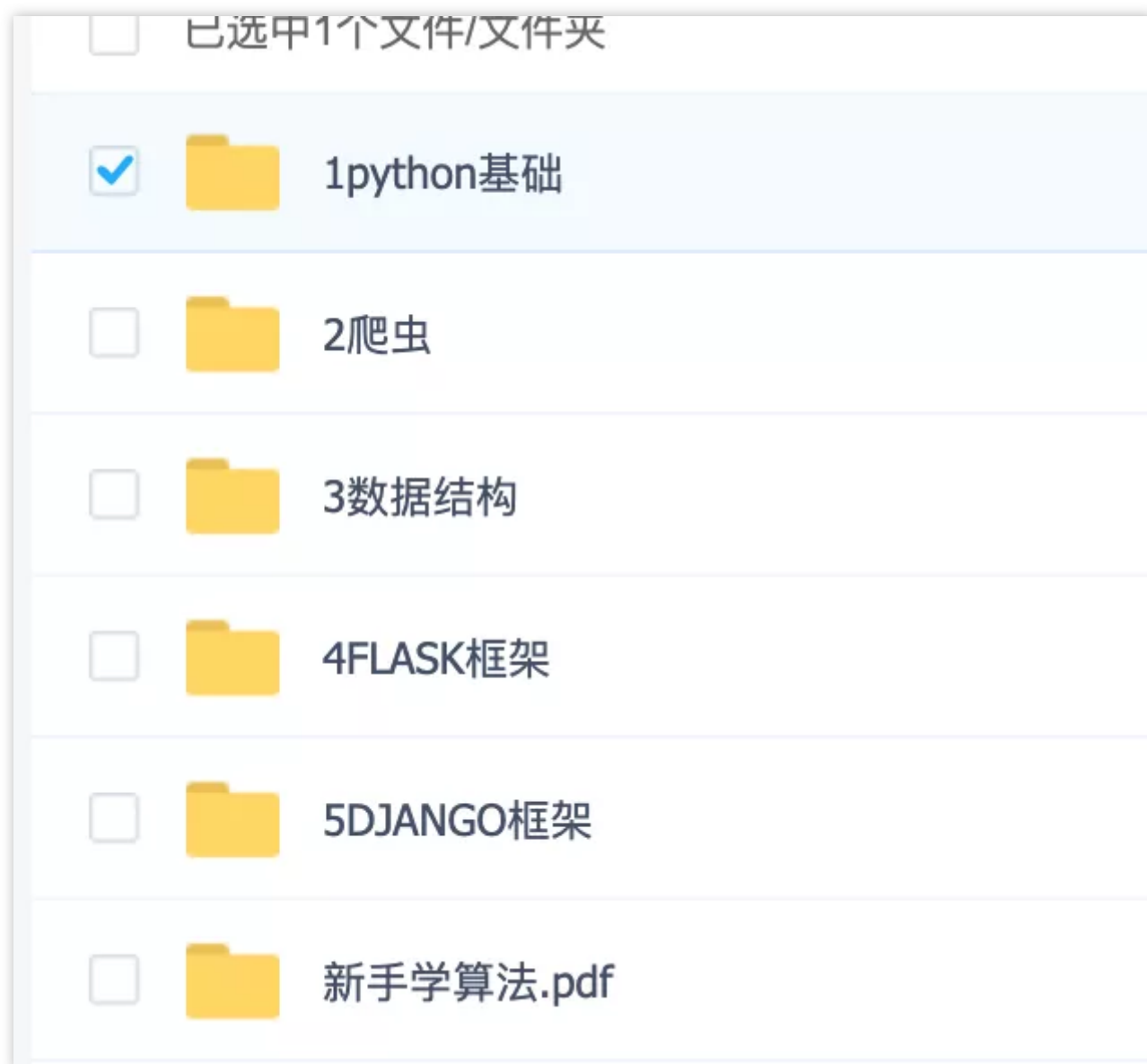
回复「1024」有福利，每天分享Java，Python,热点新闻等，带领20W程序员...
1 篇原创内容

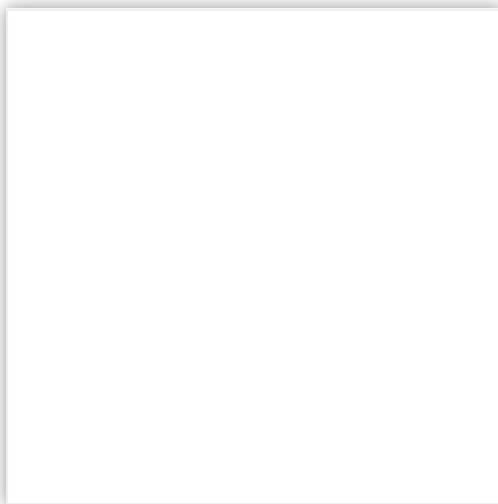
Official Account

点击上方公众号名片回复「PDF」，获取文中案例及代码。

— The End —

送大家一份Python学习大礼包，从Python基础，爬虫，数据分析Web开发等全套资料，吃透资料，这些资料都是视频，新人学起来非常友好。





扫码加微信后备注「Python新手」方便我给你发送资料

推荐阅读

1. 卧槽，一个牛逼的Python 可视化库：PyG2Plot
2. IT大佬廖雪峰带你玩转Python数据分析（内附资源）
3. 不得了了！Python 又爆出重大 Bug~
4. 新功能！微信可以开“小号”了



扫码回复「大礼包」后获取大礼

喜欢此内容的人还喜欢

WebAssembly 介绍 | Linux 中国

Linux中国

全网最全面的 Node.js 资源汇总推荐，4W Star!

前端技术江湖