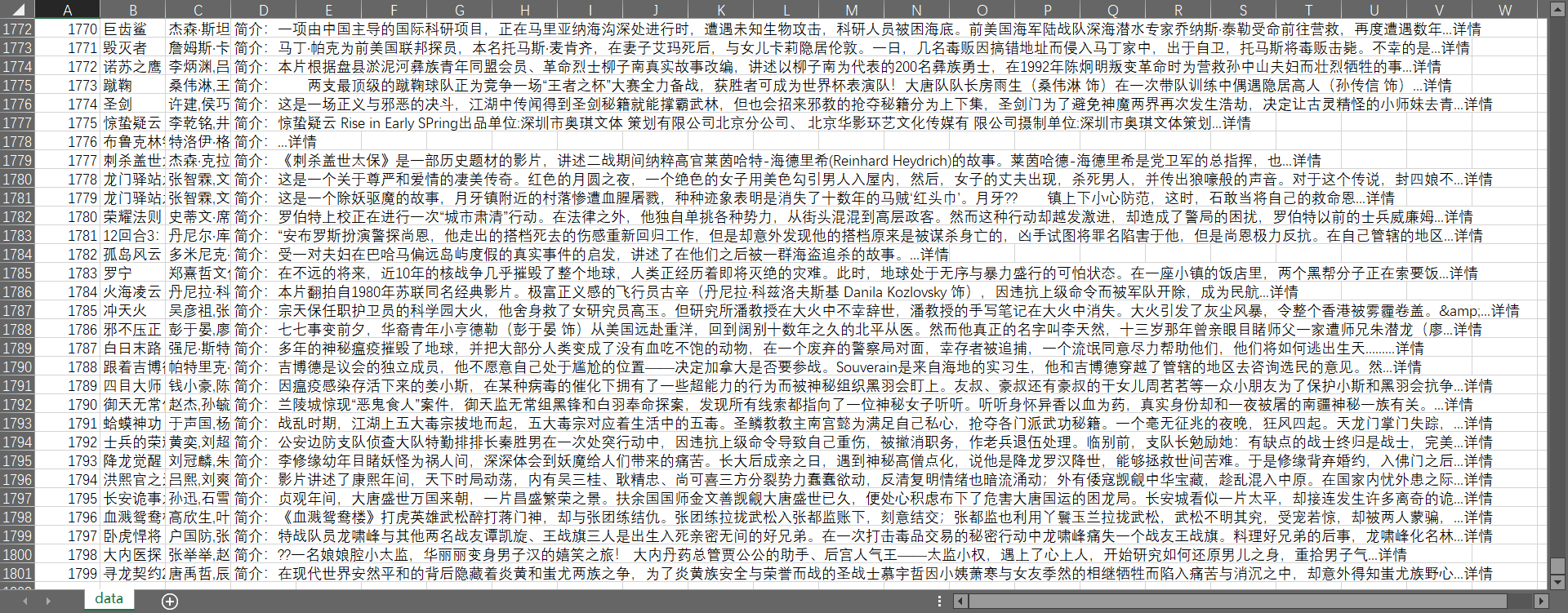
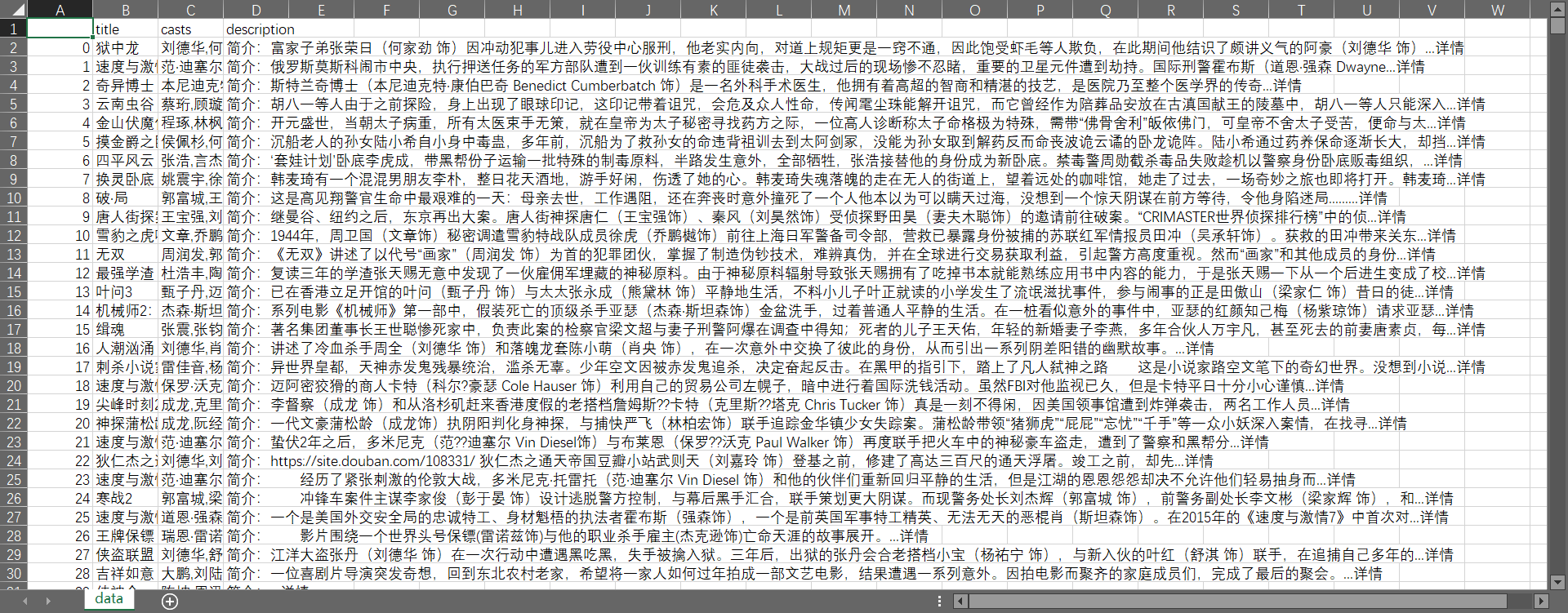
## 网络爬虫实践题

1. 代码功能简要说明：

爬虫程序由6个子程序和主函数组成，子程序功能依次为：

1. 获取第1~end页的url ：getAllUrl；
2. 通过requests库请求获得目标网页的HTML页面内容：getHTMLText；
3. 解析html页面，初始化BS对象：analyzeHTMLText；
4. 在bs对象内检索电影演职员表和详情页面的链接，再调用函数getDescription进入详情页检索电影简介，并使用正则表达式去除html标签，最后将获得的电影名、信息分别存在列表中：getDetails；
5. 进入指定的电影详情页url获取电影简介：getDescription；
6. 使用pandas库保存获取的内容到工作目录下的data.csv：saveContents

子程序由主函数依次调用，爬取到的信息保存在爬虫程序工作目录下的data.csv中，设置爬取网站50页时文件首尾截图如下：（共1798条数据）



由于该网站数据量不大，csv文件只有几百kb，因此爬取到的数据一并提交了。

1. 关键代码说明：
2. 分析得到不同页码的url格式后，可以自动化生成所需页数的url，保存在列表中，且页数便于修改。

for i in range(1, end):

url.append("https://www.dytt8.net.cn/index.php/vod/show/id/6/page/"+str(i)+".html")

1. 获得详情页url

# css类->属性组合查找获取链接html标签，并使用.get()获取链接纯文本

for item in bs.select(".stui-vodlist\_\_box > a"):

link = item.get('href')

# 分析获取到的链接和详情页url的对应关系，拼接字符串获得详情页完整url

detail\_url = "https://www.dytt8.net.cn/index.php/vod/" + link[15:]

1. 使用正则表达式删除html标签

regl = re.compile("<[^>]\*>")

item1 = regl.sub('', str(item))

1. 研究网页源码后我发现，有且仅有电影标题和演员表属于类“text-overflow”，且二者交替出现，因此只需一次beautifulsoup选择，再将获取的内容交替保存至name和detail列表中，即可在保证数据正确性的同时提高效率。
2. csv编码问题：使用pandas保存至csv时，默认编码格式为UTF-8，但excel不支持；为便于查看，查阅pandas官方文档后设置编码格式为gb18030。

df.to\_csv("data.csv", encoding="gb18030") # encoding gbk, gb2312均报错，又win10不支持UTF-8，所以只有这个了

1. 设计的不足
2. 由于整个爬虫程序单线程工作，效率不高；若将其改进为多个线程同时提取不同电影的信息，应当能有所改善。
3. 获取的简介信息不完整。原网页中简介信息在两部分中显示，其中完整的简介——第二部分格式较为复杂（如图），而且内容过多，为保证运行效率使用的是不完整的第一部分。

