|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | Github账号 | 分工 | 成绩 |
| 鲁义军 | 201592067 | BlackPeachLawn | 组长 |  |
| 毕文韬 | 201592385 | BiWenTao201592385 | 组员 |  |
| 王旭昇 | 201592090 | blueeda | 组员 |  |
| 孙振航 | 201592200 | a231819 | 组员 |  |
| 雷廷 | 201592174 | LT201592174 | 组员 |  |

**开源软件基础课程报告**

|  |  |
| --- | --- |
| 报告题目 | 开源软件基础大作业 |
| 项目网址 | https://github.com/BlackPeachLawn/KYRJ\_HomeWork |
| 完成日期 | 2018.01.07 |

目 录

[1 生命周期 1](#_Toc420487560)

[2 需求分析 2](#_Toc420487560)

[2.1 爬取词条 2](#_Toc420487561)

[2.2 爬取票房 2](#_Toc420487561)

[2.3 存储数据 2](#_Toc420487561)

[3 概要设计 2](#_Toc420487560)

[3.1 抓取数据 2](#_Toc420487561)

[3.2 数据下载 2](#_Toc420487561)

[3.3 可视化 2](#_Toc420487561)

[4 详细设计 3](#_Toc420487560)

[4.1 信息来源 4](#_Toc420487561)

[4.2 信息获取及处理 5](#_Toc420487561)

[4.3 数据信息展示 6](#_Toc420487561)

[5 编码实现 7](#_Toc420487560)

[5.1 下载网页 7](#_Toc420487561)

[5.2 提取电影信息 8](#_Toc420487561)

[5.3 检索功能主函数 9](#_Toc420487561)

[5.4 可视化 10](#_Toc420487561)

[6 测试 11](#_Toc420487560)

[6.1 根据检索条件获取电影信息 11](#_Toc420487561)

[6.2 信息图表化 12](#_Toc420487561)

[7 总结 1](#_Toc420487560)4

**引语：**

为了快速地获取到海量电影中的数据，根据电影网页的特点，设计出了一种基于Python的爬虫程序。通过模拟登录百度糯米电影网站，实时抓取电影网站中指定电影内容，该工具匹配符合规定条件的电影，并抓取相关内容；最后使用该工具对部分电影数据作了分析。

**1．生命周期：**

整个爬虫的生命周期(链接提取、页面下载、内容抽取、可视化呈现)

针对本项目的开发特点，参考相似软件过程体系，决定采用瀑布模型。

本项目具备瀑布模型的特点：

a.阶段间具有顺序性和依赖性

b.推迟实现，避免急于求成而导致的返工，尽可能推迟程序的物理实现

c.质量保证的观点

d.提供了一种规范，使得分析、设计、编码、测试工作可以在该规范的指导下有序地展开，避免了开发、维护过程中的随意状态

e.需求确定、不随时间发生变化

需求分析

验证

编码

测试

设计

验证

综合测试

产品提交

2.生命期中的各阶段如下：  
需求分析阶段  
阶段目标：确定目标的需求  
输入： 项目计划，  
过程： 需求获取，需求分析

输出： 需求规格

设计阶段

阶段目标：总体系统结构设计

输入： 需求规格

过程： 总体设计

输出： 系统设计规划

编码阶段

阶段目标：实现描述的功能

输入： 系统设计规划

过程： 编码，代码走查，代码评审

输出： 源代码，可运行版本

集成测试

阶段目标：通过集成环境下的软件测试

输入： 测试计划，测试案例

过程： 集成测试，系统测试

输出： 测试报告，产品说明书

**2.需求分析**

1.需求背景：过去几十年里，web的迅速发展，大量的数据通过web发布，使其成为世界上规模最大的公共数据源。随着网络的高速发展，互联网成为海量信息的载体，如何有效地提取并利用这些信息成为研发人员一个巨大的挑战。为了更好地吸引开发者，以及和开发者更好的交互，百度糯米等平台提供了一些数据访问编程接口(OpenAPI)供研发人员获取数据，但是，由于各方面的考虑，利用OpenAPI进行数据抓取时总是有各种各样的限制…（以豆瓣为例，豆瓣提供的API对普通用户的权限和抓取频率都进行了较为严格地限制，而且无法对豆瓣内容进行搜索口）。

2.需求整理

提出的项目需求如下：

设计、开发一套基于主要基于windows平台的具有爬取电影网站信息的爬虫，能够实现收集并分析电影数据以及最新电影票房动态，能让电影爱好者和电影数据分析者了解最新的电影相关信息与动态，为广大电影关注者提供便利的数据收集方式。该爬虫要求功能完善，操作友好，可以适配windows主流系统。

爬虫内部需要实现的主要功能具体要求如下：

2.1.可以爬取关于感兴趣的电影的相关词条（包括但不限于电影名称、电影评分，导演、主演、剧情简介等）。

2.2.可以实时查询热映电影的票房数据（包括但不限于总票房，上映天数，实时票房，票房占比，排片占比、上座率、观影人次、平均票价等）

2.3.可以将爬取的数据存储到本地，来进行进一步的分析或用可视化方式进行呈现等。

**3.概要设计**

1.解决方案概述

我们提出了一款基于Python的电影数据爬虫程序，为电影数据获取提供技术支持。本文提出的程序通过爬虫模拟登录移动端百度糯米电影并获取相关电影数据，并将这些数据保存到本地，方便进一步的数据挖掘与分析。使用本文爬虫程序能够节省电影数据分析人员的数据收集时间，使得他们可以将更多的精力放在数据分析上面，同时也可以对海量数据起到过滤作用。

2.业务处理流程

爬虫

爬取数据

数据下载

可视化表现

3.1.抓取数据

分析网页html文本和页面规则后，制定步骤与相关函数来抓取数据

3.2.数据下载

将爬取到的数据解析后，下载到本地，存取到excel中

3.3.可视化呈现

将存取在excel中的数据用表格，扇形图等可视化方式进行整合呈现

**4．详细设计**

4.1电影信息资源

糯米电影（www.nuomi.com）是百度旗下的综合娱乐O2O平台，聚焦电影、演出两个行业，凭借领强大的平台资源、深厚的技术积累，深度布局演出市场，将在大数据指导优质项目挖掘、资金投放，以及票务、宣传、互动资源整合方面给予行业更多的动能，为演出行业的正向发展带来新的想象空间。实时的显示电影评分，票房，口碑，上座率等用户关注的信息。

4.2信息获取及处理

整合数据

信息展示

提取网站固定标签信息

信息来源

导入csv文件

添加样式

提取筛选信息

糯米电影

在糯米电影网页的电影信息模块中,截取对应标签内容,筛选出电影名称，评价，票房，上座率，排场占比等实时信息及对应电影的URL地址,将提取的信息放入List中.遍历List,用for循环遍历List中的信息并写入.csv文件中，读取.csv中的文件的信息，分析并绘制了条形图，雷达图，饼状图更加直观简洁的表示爬取信息。

4.3爬取信息展示

首先导入pygal库，创建一个图表实例，提取csv文件中的数据，对数据字符串进行处理（例如票房数据）将处理完的数据放入列表中，将几个列表作为参数传入到以上图表实例方法中，生成条形图，雷达图，饼状图三种图像，并以.svg的文件格式保存到本地文件夹。

**5．编码实现**

5.1下载糯米电影网页

13 download\_url = 'http://dianying.nuomi.com/movie/boxoffice'

14 def download\_page(url):

15 headers = {

16 'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_11\_2) Apple WebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.80 Safari/537.36'

17 }

18

19 r = requests.get(url, headers=headers)

20 r.encoding = 'UTF-8'

21 data =r.text

22 return data

5.2提取第一部分电影名，票房，上座率，排场数等信息。

25 def parse\_html(html):

26 soup=BeautifulSoup(html,'lxml')

27 movie=[]

28 movie\_list\_soup= soup.find('div',attrs={'class':'detail-container'})

29 for l in movie\_list\_soup.find\_all('dd',attrs={'class':'each-movie clearfix'}):

30 movie\_title=l.find('h5',attrs={'class':'movie-title'}).getText()

31 movie\_day=l.find('li',attrs={'class':'days'}).getText()

32 movie\_total=l.find('span').getText()

33 movie\_today=l.find('div',attrs={'class':'column colm-2'}).getText()

34 movie\_pfzb=l.find('div',attrs={'class':'column colm-3'}).getText()

35 movie\_ppzb=l.find('div',attrs={'class':'column colm-4'}).getText()

36 movie\_szl=l.find('div',attrs={'class':'column colm-5'}).getText()

37 movie\_pzzb=l.find('div',attrs={'class':'column colm-6'}).getText()

38 movie\_cc=l.find('div',attrs={'class':'column colm-7'}).getText()

39 movie\_rc=l.find('div',attrs={'class':'column colm-8'}).getText()

40 movie\_cjrc=l.find('div',attrs={'class':'column colm-9'}).getText()

41 movie\_cjsr=l.find('div',attrs={'class':'column colm-10'}).getText()

42 movie\_pjpj=l.find('div',attrs={'class':'column colm-11'}).getText()

43 next\_page=l.find('div',attrs={'class':'column colm-1'})

44 next\_url='http://dianying.nuomi.com'+next\_page['to-url']

45 movie\_details=[]

46 movie\_details.append(movie\_title)

47 movie\_details.append(movie\_day)

48 movie\_details.append(movie\_total)

49 movie\_details.append(movie\_today)

50 movie\_details.append(movie\_pfzb)

51 movie\_details.append(movie\_ppzb)

52 movie\_details.append(movie\_szl)

53 movie\_details.append(movie\_pzzb)

54 movie\_details.append(movie\_cc)

55 movie\_details.append(movie\_rc)

56 movie\_details.append(movie\_cjrc)

57 movie\_details.append(movie\_cjsr)

58 movie\_details.append(movie\_pjpj)

59 movie\_details.append(next\_url)

60 movie.append(movie\_details)

61 return movie

5.3提供检索功能的主函数。

def main():

87 movie\_details=parse\_html(download\_page(download\_url))

88 with open("movie\_details.csv","w",newline='')as csvfile:

89 detalimiter='\t'

90 writer=csv.writer(csvfile,dialect='excel')

91 list1=["电影名","上映天数","总票房","实时票房","票房占比","排片占比","上座率","排座占比","场次","人次","场均人次","场均收入","平均票价","网址"]

92 writer.writerow(list1)

93 for i in movie\_details:

94 writer.writerow(i)

95 with open(r"movie\_details.csv")as datacs:

96 read1=csv.reader(datacs,delimiter=',')

97 for j in read1:

98 print(j[0])

99 with open(r"movie\_details.csv")as datacsv:

100 read=csv.reader(datacsv,delimiter=',')

101 flag=0

102 print()

103 str=input("请输入你想了解的电影全名：")

104 print()

105 for i in read:

106 if str==i[0]:

107 data=download\_page(i[13])

108 dict1=dict(zip(list1,i))

109 for key in dict1.keys():

110 print('{}: \t{}'.format(key,dict1[key]))

111 print()

112 detail\_mes=parse\_detail(data)

113 flag=1

114 if flag==0:

115 print("电影名有误")

116 if \_\_name\_\_ == '\_\_main\_\_':

117 main()

5.4将提取信息图形化

#encoding=utf-8

2 import csv

3 import pygal

4 from matplotlib import pyplot as plt

6 filename = 'movie\_details.csv'

7 with open(filename) as f:

8 reader = csv.reader(f)

9 head\_row = next(reader)

11 first\_lie = []

12 second\_lie = []

13 third\_lie = []

14 fourth\_lie = []

15 fifth\_lie = []

16 sixth\_lie = []

17 seventh\_lie =[]

18 for row in reader:

19 first\_lie.append(row[0]) #电影名称

21 if (row[3])[-1] == '亿': #实时票房

22 s = (row[3])[0:-1]

23 s1 = 10000 \* float(s)

24 second\_lie.append(s1)

25 if (row[3])[-1] == '万':

26 s2 = float((row[3])[0:-1])

27 second\_lie.append(s2)

29 if (row[2])[-1] == '亿': #总票房

30 s = (row[2])[0:-1]

31 s1 = 10000 \* float(s) #将字符串由字符转为float再乘以10000

32 third\_lie.append(s1)

33 if (row[2])[-1] == '万':

34 s2 = float((row[2])[0:-1])

35 third\_lie.append(s2)

36 four = float((row[4])[0:-1])/100 #票房占比

37 fourth\_lie.append(four)

38 four = float((row[5])[0:-1])/100 #排片占比

39 fifth\_lie.append(four)

40 four = float((row[6])[0:-1])/100 #上座率

41 sixth\_lie.append(four)

42 four = float((row[7])[0:-1])/100 #占座比率

43 seventh\_lie.append(four)

44 print(fourth\_lie)

45 print(second\_lie)

47 hist = pygal.Bar()

48 hist.title = "Result of the movies "

49 hist.x\_title = "电影名称"

50 hist.y\_title = "电影票房数 (单位：万)"

51 hist.x\_labels = first\_lie

52 hist.add("总票房",third\_lie)

53 hist.render\_to\_file('总票房.svg')

55 pie\_chart = pygal.Pie()

56 pie\_chart.title = '票房所占比例'

57 i = 0

58 for x in first\_lie:

59 pie\_chart.add(first\_lie[i], fourth\_lie[i])

60 i = i+1

61 pie\_chart.render\_to\_file('票房所占比例.svg')

63 j = 0

64 radar\_chart = pygal.Radar()

65 radar\_chart.title = '电影总数据雷达图'

66 radar\_chart.x\_labels = head\_row[2:7]

67 radar\_chart.x\_labels[0] = radar\_chart.x\_labels[0]+"(单位：十亿)"

68 radar\_chart.x\_labels[1] = radar\_chart.x\_labels[0]+"(单位：亿)"

69 while j < 20:

70 radar\_chart.add(first\_lie[j], [third\_lie[j]/100000, second\_lie[j]/10000, fourth\_lie[j],fifth\_lie[j],sixth\_lie[j],seventh\_lie[j]])

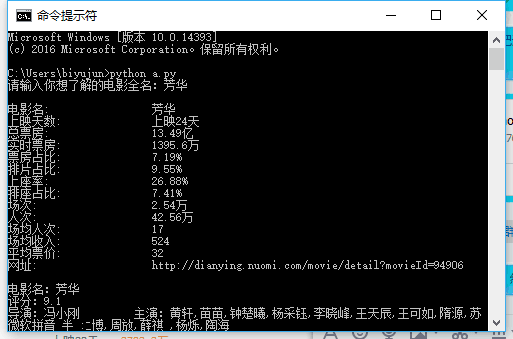
71 j = j+1

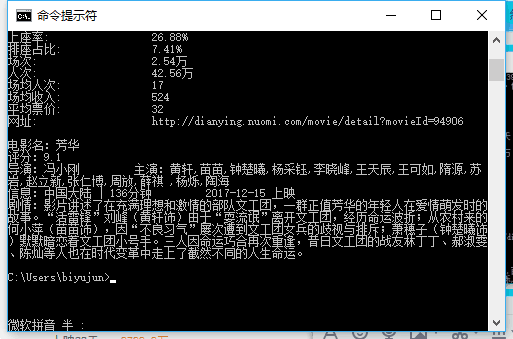
72 print(j)

73 radar\_chart.render\_to\_file('总数据雷达图.svg')

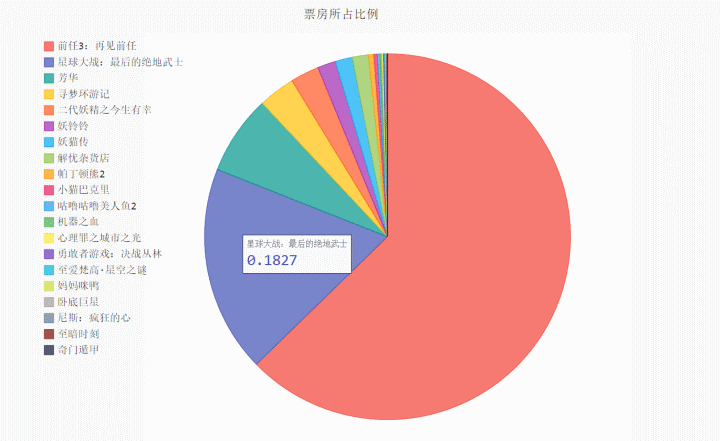
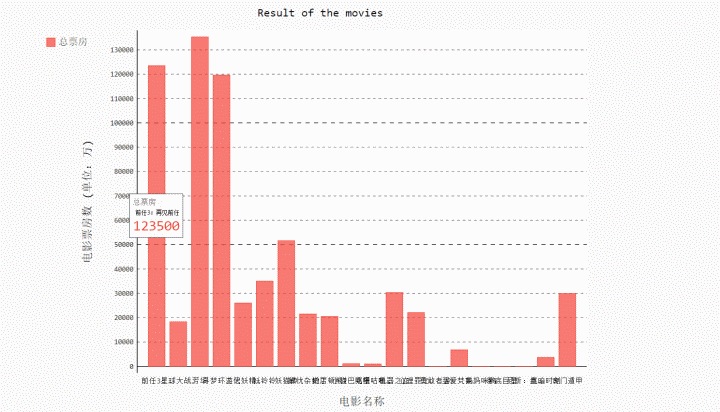
**6．测试**

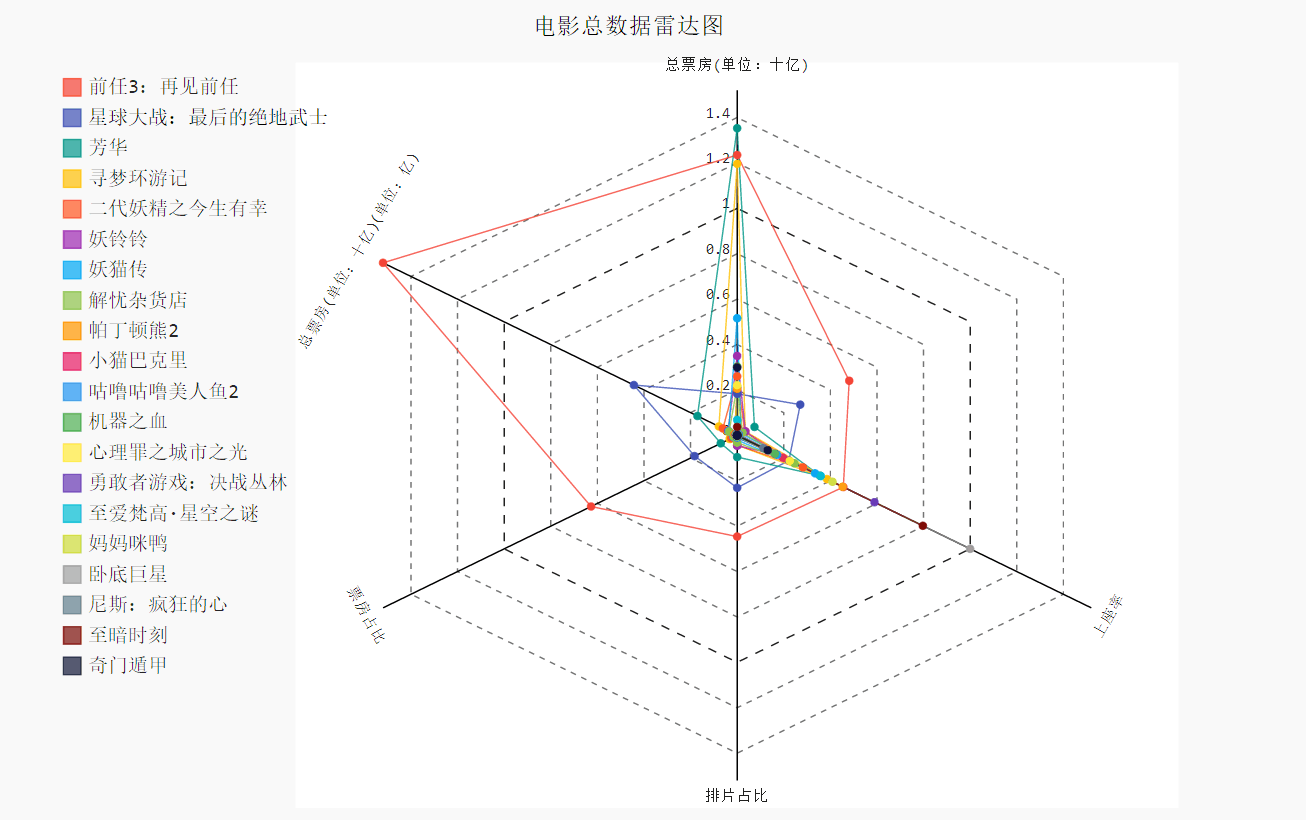
6.1根据检索条件获得相应信息，例如电影《芳华》。





6.2将上面提取到的信息图形化展示。





**7．总结**

观看电影是现在大学生选择最多的消遣方式，所以我们小组讨论决定制作一个关于整合热门电影相关信息的爬虫小程序，并以可视化的方式，简洁明了的展现给大家。通过这次做项目，使我们对编程有了进一步的认识。

本次项目所完成的功能中，主要是用python对数据的获取，相对缺乏的是对数据的细化处理。而针对网络爬虫来说，数据分析和挖掘是比数据获取更重要也更有难度的。所以如何将获取到的信息可视化的展现出来变得尤为重要。考虑到这一点，图表是一个很好的将数据可视化展示的工具，因此我们将获取到的信息做了三种不同的图表来更直观地展示各个电影相关事实的信息。

对于一些布局和细节设计没有考虑清楚就动手，导致了遇到了错误就不停地修改，这是项目实现过程中应该竭力避免的。如果一开始对此模块到底需要实现的功能与细节分析得不彻底，那么就最好直接停止，避免在开始编程的时候就手足无措，进而使这部分的代码出现漏洞，浪费时间影响进度。

另外,文档的书写也非常讲究规范化。团队合作之间除了口头上的交流，文档的信息传递功能不可小视。组员之间的工作交接,接口的定义都需要合格的文档来体现，这样才不会出现团队配合之间的失误,更有利于项目的规范化实施。