Course Project Report

Task 1:

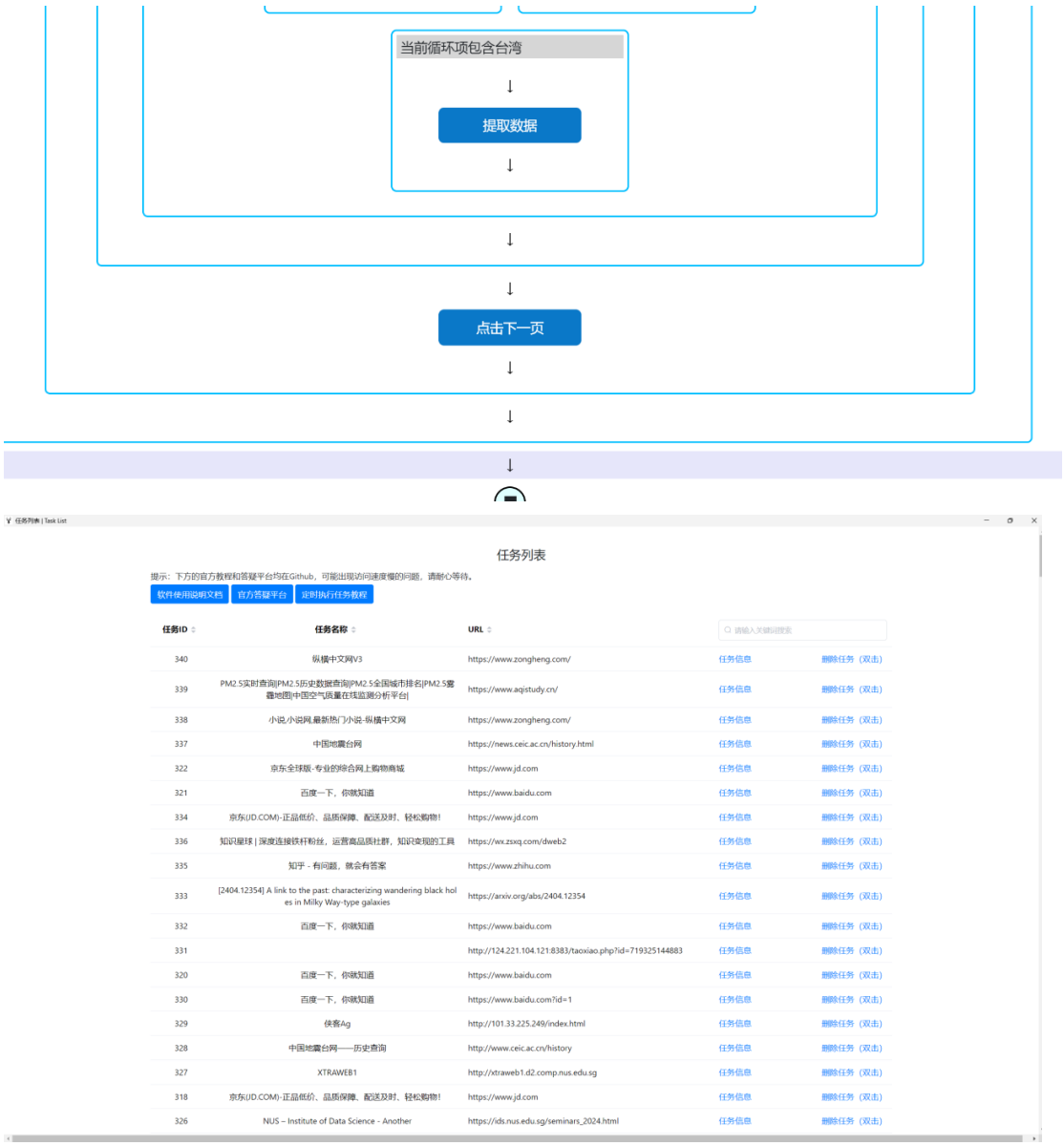
爬取中国地震台网历史数据：

1. 纬度分别大于 10 度、20 度、30 度、40 度、50 度区域的数据；
2. 爬取查询结果页面前 50 页内容；
3. 过滤结果，保存结果中含有西藏、四川、台湾的数据；
4. 对保存的数据进行数据分析，可视化三地地震信息（可视化方式越丰富、信息表现越全面分数越高）。

报告中需要详细介绍设计思路和步骤(截图)，并且展示可视化信息并进行分析。

将爬取保存的数据文件命名为：小组名-Task1.csv（如有多个文件则分别命名后再压缩成一个压缩包，命名为小组名-Task1.zip），与报告一起上传至 Moodle。


在iframe内操作
循环类型：
文本列表（多用于循环在文本框输入文本）
内容列表（用Field["字段名"]来输入某字段/自定义操作的最新提取/返回值）：
10
20
30
40
（自定义操作）使用代码/脚本定义循环退出条件（也可以在流程中添加**自定义操作**，然后选择**退出循环**选项）：
循环采集数据
判断条件 - 从左往右依次判断
点击此处在最右边增加条件分支
是否包含西藏
提取数据
当前循环项包含 四川
提取数据
当前循环项包含台湾
提取数据


当前循环项包含台湾
提取数据
点击下一页

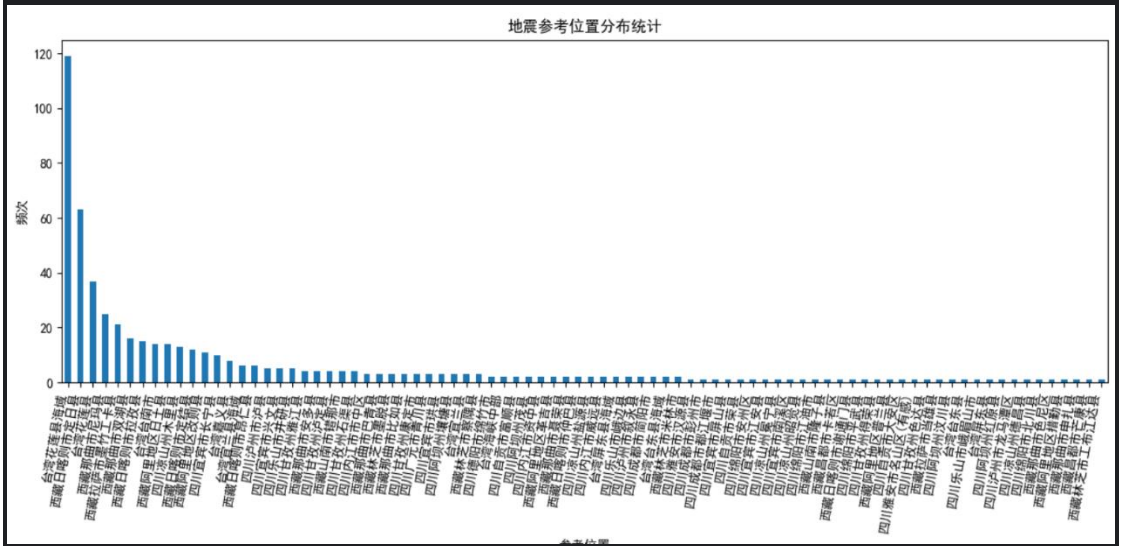
任务列表 | Task List
任务列表
提示：下方的官方教程和答疑平台均在Github，可能出现访问速度慢的问题，请耐心等待。
软件使用说明文档
官方答疑平台
定时执行任务教程
任务ID
任务名称
URL
请输入关键词搜索
340 纵横中文网V3 https://www.zongheng.com/ 任务信息 删除任务（双击）
339 PM2.5实时查询|PM2.5历史数据查询|PM2.5全国城市排名|PM2.5雾霾地图|中国空气质量在线监测分析平台| https://www.aqistudy.cn/ 任务信息 删除任务（双击）
338 小说,小说网,最新热门小说-纵横中文网 https://www.zongheng.com/ 任务信息 删除任务（双击）
337 中国地震台网 https://news.ceic.ac.cn/history.html 任务信息 删除任务（双击）
322 京东全球版-专业的综合网上购物商城 https://www.jd.com 任务信息 删除任务（双击）
321 百度一下，你就知道 https://www.baidu.com 任务信息 删除任务（双击）
334 京东(JD.COM)-正品低价、品质保障、配送及时、轻松购物! https://www.jd.com 任务信息 删除任务（双击）
336 知识星球 | 深度连接铁杆粉丝，运营高品质社群，知识变现的工具 https://wx.zsxq.com/dweb2 任务信息 删除任务（双击）
335 知乎 - 有问题，就会有答案 https://www.zhihu.com 任务信息 删除任务（双击）
333 [2404.12354] A link to the past: characterizing wandering black holes in Milky Way-type galaxies https://arxiv.org/abs/2404.12354 任务信息 删除任务（双击）
332 百度一下，你就知道 https://www.baidu.com 任务信息 删除任务（双击）
331 http://124.221.104.121:8383/taoxiao.php?id=719325144883 任务信息 删除任务（双击）
320 百度一下，你就知道 https://www.baidu.com 任务信息 删除任务（双击）
330 百度一下，你就知道 https://www.baidu.com?id=1 任务信息 删除任务（双击）
329 侠客Ag http://101.33.225.249/index.html 任务信息 删除任务（双击）
328 中国地震台网——历史查询 http://www.ceic.ac.cn/history 任务信息 删除任务（双击）
327 XTRAWEB1 http://xtraweb1.d2.comp.nus.edu.sg 任务信息 删除任务（双击）
318 京东(JD.COM)-正品低价、品质保障、配送及时、轻松购物! https://www.jd.com 任务信息 删除任务（双击）
326 NUS – Institute of Data Science - Another https://ids.nus.edu.sg/seminars_2024.html 任务信息 删除任务（双击）

执行任务时，选择中国地震台网，然后直接执行就能得到对不同高纬度要求的结果，直至收集到五十条为止。

使用 easysider 设计数据爬取脚本。脚本逻辑为打开网页后，循环输入不同纬度（分别对应大于 10 度、20 度等要求 ）并点击查询。每次查询后，通过循环点击下一页按钮，对每一页进行数据采集。在采集过程中，检查数据字段，若包含西藏、四川或台湾相关信息，则进行采集。通过这样的循环操作 49 次（从第一页到第 50 页 ），来爬取 50 页的内容。通过这样的循环机制，重复操作 49 次（从第 2 页到第 50 页 ，第 1 页默认加载 ），从而实现 50 页内容的完整爬取。

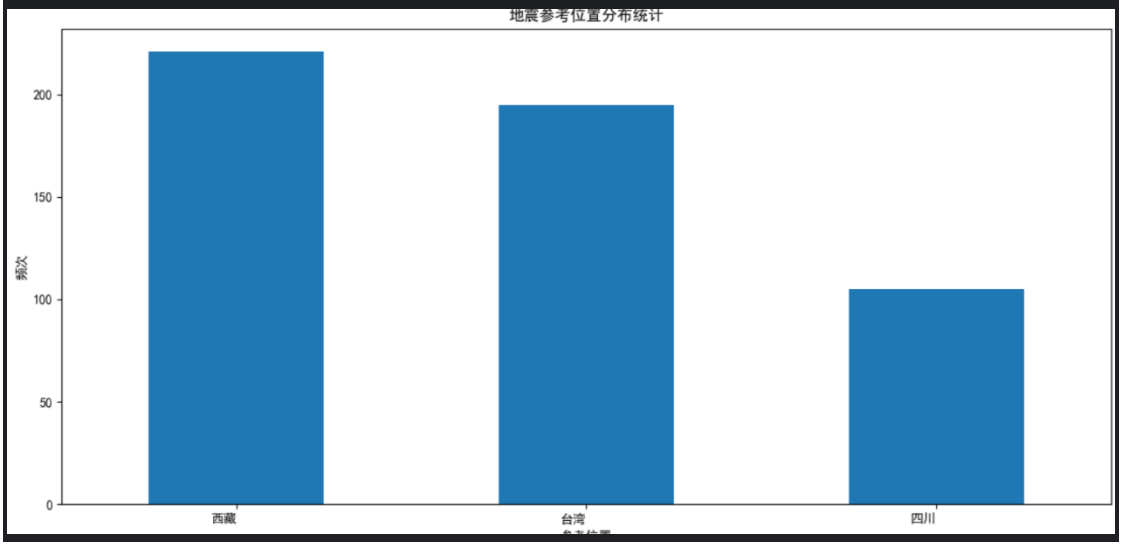
```
import pandas as pd
import matplotlib.pyplot as plt


# 读取数据

df = pd.read_csv("./中国地震台数据.csv")


df = df.iloc[:,0:6]

df.drop_duplicates(inplace=True)
# df.info()
# print( df.head() )



location_counts = df['参考位置'].value_counts()


plt.figure(figsize=(12, 6))
location_counts.plot(kind='bar')
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.title('地震参考位置分布统计')

plt.xlabel('参考位置')

plt.ylabel('频次')
plt.xticks(rotation=80, ha='right')
plt.tight_layout()
plt.show()
```

```
plt.close()
```




```
import pandas as pd
import matplotlib.pyplot as plt


df = pd.read_csv("./中国地震台数据.csv")

df = df.iloc[:,0:6]
df.drop_duplicates(inplace=True)
# df.info()
print( df.head() )

location = df.pop('参考位置')

location.astype(str)
location=location.str[:2]

df.insert(1,'参考位置',location)

print(df.head())


location_counts = df['参考位置'].value_counts()


print("\n参考位置统计结果: ")

print(location_counts)

plt.figure(figsize=(12, 6))
location_counts.plot(kind='bar')
plt.rcParams['font.sans-serif'] = ['SimHei']

plt.title('地震参考位置分布统计')
```

```
plt.xlabel('参考位置')

plt.ylabel('频次')
plt.xticks(rotation=0, ha='right')
plt.tight_layout()
plt.show()

plt.close()
```



从代码 location_counts = df['参考位置'].value_counts() 和 location_counts.plot(kind='bar') 生成的“地震参考位置分布统计” 柱状图来看：

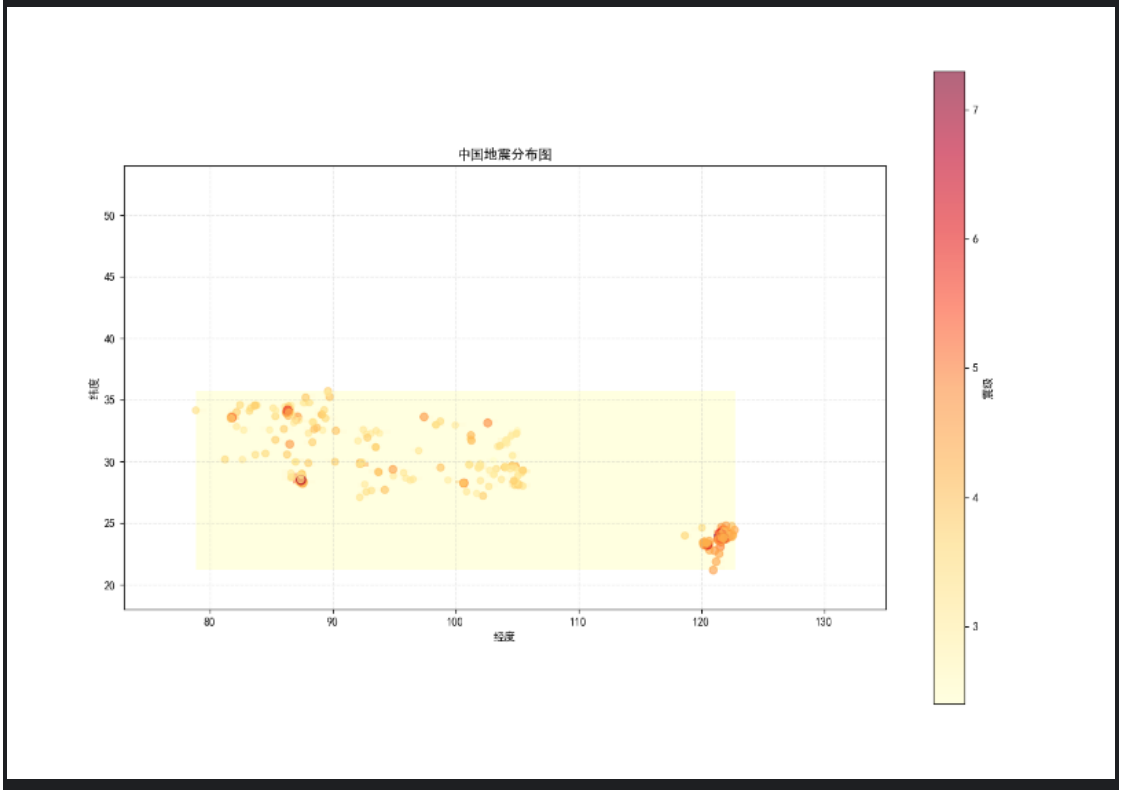
高频地震区域：通过代码统计出的结果直观显示，西藏、四川、台湾地区在柱状图中对应的柱子高度显著高于其他地区。这是因为 value_counts 函数对“参考位置” 列中的每个地区出现的次数进行了准确计数，反映在柱状图上就是高度代表频次。西藏位于印度洋板块与亚欧板块的强烈碰撞地带，板块的强烈挤压使得地壳岩石变形、破裂，从而引发频繁的地震活动。四川处于多个活跃的地质构造带，如龙门山断裂带等，板块间的应力积累和突然释放导致地震频发。台湾地处环太平洋地震带，太平洋板块与亚欧板块在此相互作用强烈，地壳运动极为活跃，因此地震发生的频次较高。

低频地震区域：除了上述三个地区外，其他地区的柱子高度相对较低。这表明这些地区的地震发生次数较少，但并不意味着它们没有地震风险。代码的统计结果清晰地展示了这种差异，有助于我们对不同地区的地震活动程度有一个初步的量化认识。

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
from matplotlib.colors import LinearSegmentedColormap

# 读取数据

df = pd.read_csv("./中国地震台数据.csv")

df = df.iloc[:,0:6]
df.drop_duplicates(inplace=True)

# 设置中文字体

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False

# 创建图形

plt.figure(figsize=(15, 10))

# 绘制中国地图底图

plt.plot([73.66, 135.05, 135.05, 73.66, 73.66], [53.55,
53.55, 18.15, 18.15, 53.55], 'k-', linewidth=1)
=

x = df['经度'].values

y = df['纬度'].values

heatmap, xedges, yedges = np.histogram2d(x, y, bins=50)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
plt.imshow(heatmap.T, extent=extent, origin='lower',
cmap='YlOrRd', alpha=0.6)
=

scatter = plt.scatter(df['经度'], df['纬度'],

                      c=df['震级'],

                      cmap='YlOrRd',

                      s=df['震级']*10,  # 点的大小根据震级变化

                      alpha=0.6)

cbar = plt.colorbar(scatter)

cbar.set_label('震级')
```

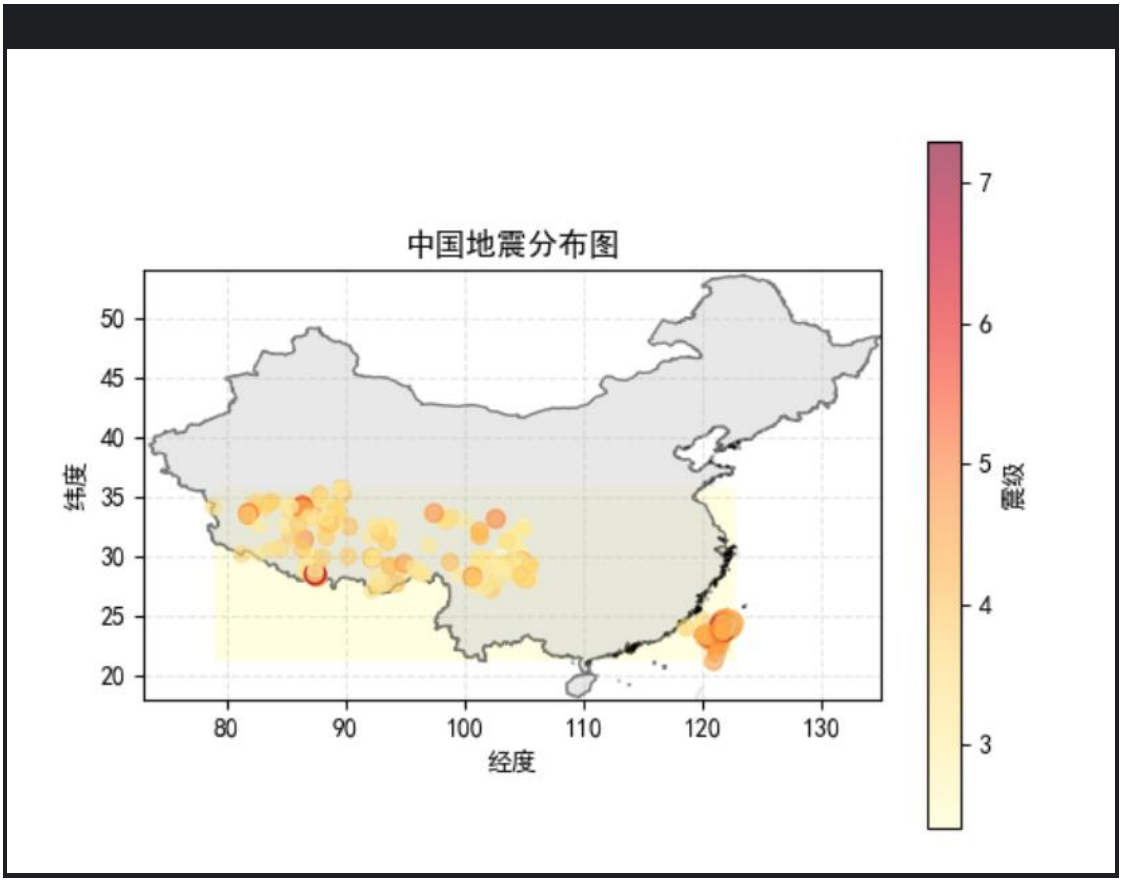
```
plt.title('中国地震分布图')

plt.xlabel('经度')

plt.ylabel('纬度')


plt.xlim(73, 135)
plt.ylim(18, 54)

plt.grid(True, linestyle='--', alpha=0.3)
plt.show()

i=300, bbox_inches='tight')
```




```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LinearSegmentedColormap
import cnmaps
import geopandas as gpd
from shapely.geometry import shape
```

```
# 读取数据

df = pd.read_csv("./中国地震台数据.csv")

df = df.iloc[:,0:6]
df.drop_duplicates(inplace=True)


# 设置中文字体

plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


# 创建图形

plt.figure(figsize=(15, 10))


# 获取中国地图数据

china_shapes = cnmaps.get_adm_maps(level='国')

# 提取 geometry 对象并转换为 shapely 几何对象

geometries = [shape(item['geometry'].__geo_interface__)
for item in china_shapes]
china_gdf = gpd.GeoDataFrame(geometry=geometries,
crs="EPSG:4326")


# 绘制中国地图底图

china_gdf.plot(color='lightgray', edgecolor='black',
alpha=0.5)


# 创建热力图

x = df['经度'].values

y = df['纬度'].values

heatmap, xedges, yedges = np.histogram2d(x, y, bins=50)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
plt.imshow(heatmap.T, extent=extent, origin='lower',
cmap='YlOrRd', alpha=0.6)
```

```
# 绘制散点图
scatter = plt.scatter(df['经度'], df['纬度'],
                      c=df['震级'],
                      cmap='YlOrRd',
                      s=df['震级']*10,  # 点的大小根据震级变化
                      alpha=0.6)

# 添加颜色条
cbar = plt.colorbar(scatter)
cbar.set_label('震级')

# 设置图表属性
plt.title('中国地震分布图')
plt.xlabel('经度')
plt.ylabel('纬度')

# 设置坐标轴范围
plt.xlim(73, 135)
plt.ylim(18, 54)

# 添加网格
plt.grid(True, linestyle='--', alpha=0.3)
plt.show()
```



地震经纬度与震级关系可视化结果分析

地震空间分布密度：对于代码 plt.hist2d(x=df['经度'].values, y=df['纬度'].values, bins=50) 生成的二维直方图，它将地震数据按照经度和纬度进行了分布统计。在我国西南地区（主要是西藏、四川一带 ）以及东南沿海的台湾地区，颜色较深（假设颜色深代表地震分布密度高 ）。这是因为 hist2d 函数将经纬度空间划分为 50×50 个区间，并统计每个区间内地震的数量，颜色的深浅反映了区间内地震数量的多少。西藏由于板块碰撞的强烈作用，地壳岩石破碎，形成了众多的地震活动区域，导致该地区在二维直方图上呈现出较高的地震分布密度。四川的多个活跃断裂带，如龙门山断裂带、鲜水河断裂带等，使得地震在该地区频繁发生，从而在直方图上显示出较高的密度。台湾处于板块边界，板块的强烈运动导致地震在空间上密集分布，因此在图中颜色较深。

震级与地理位置关联：由代码 scatter = plt.scatter(df['经度'], df['纬度'], c=df['震级'], cmap='viridis', s=df['震级']*10, alpha=0.6) 生成的散点图中，台湾地区的点相对较大且颜色较深（假设颜色深、点大代表震级高 ）。这是因为散点图的点大小由 s=df['震级']*10 决定，颜色由 c=df['震级'] 通过 viridis 颜色映射表确定，震级越高，点越大且颜色越深。台湾处于环太平洋地震带这一全球最活跃的地震构造区域，板块间的强烈碰撞和俯冲作用使得该地区能够积累巨大的能量，从而引发较高震级的地震。在西藏和四川地区，也存在一些较大且颜色较深的点，说明这些地区也会发生较高震级的地震。西藏的地震活动与板块碰撞的远程效应以及地壳内部的复杂构造有关，而四川的地震震级分布则受到其活跃断裂带的控制，不同断裂带的力学性质和活动方式导致了震级在空间上的差异。

Task 2:

爬取纵横中文网小说：

1. TOP10 月票榜前五的小说；
2. 每本小说爬取前 50 章（少于 50 章的爬取全部章节）；
3. 将爬取的数据进行数据清洗和整理，以方便阅读的格式和文件分别保存 5 本小说；
4. 为 5 本小说分别绘制词云并保存成图片格式。

报告中需要详细介绍设计思路和步骤（截图），并且展示 5 本小说的词云。将数据处理后的 5 本小说和词云图片文件一起打包为一个 zip 文件，文件名：小组名-Task2.zip，与报告一起上传至 Moodle。

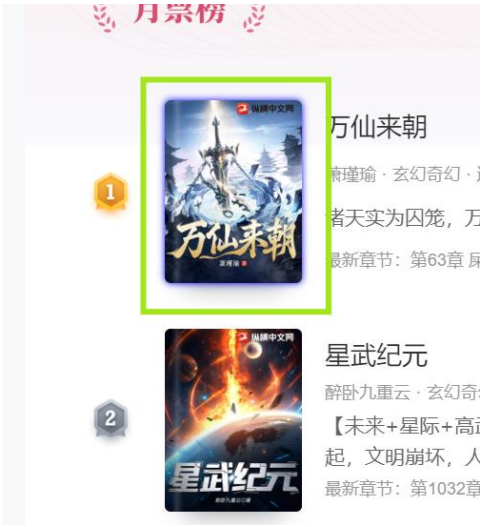
爬虫设计部分：

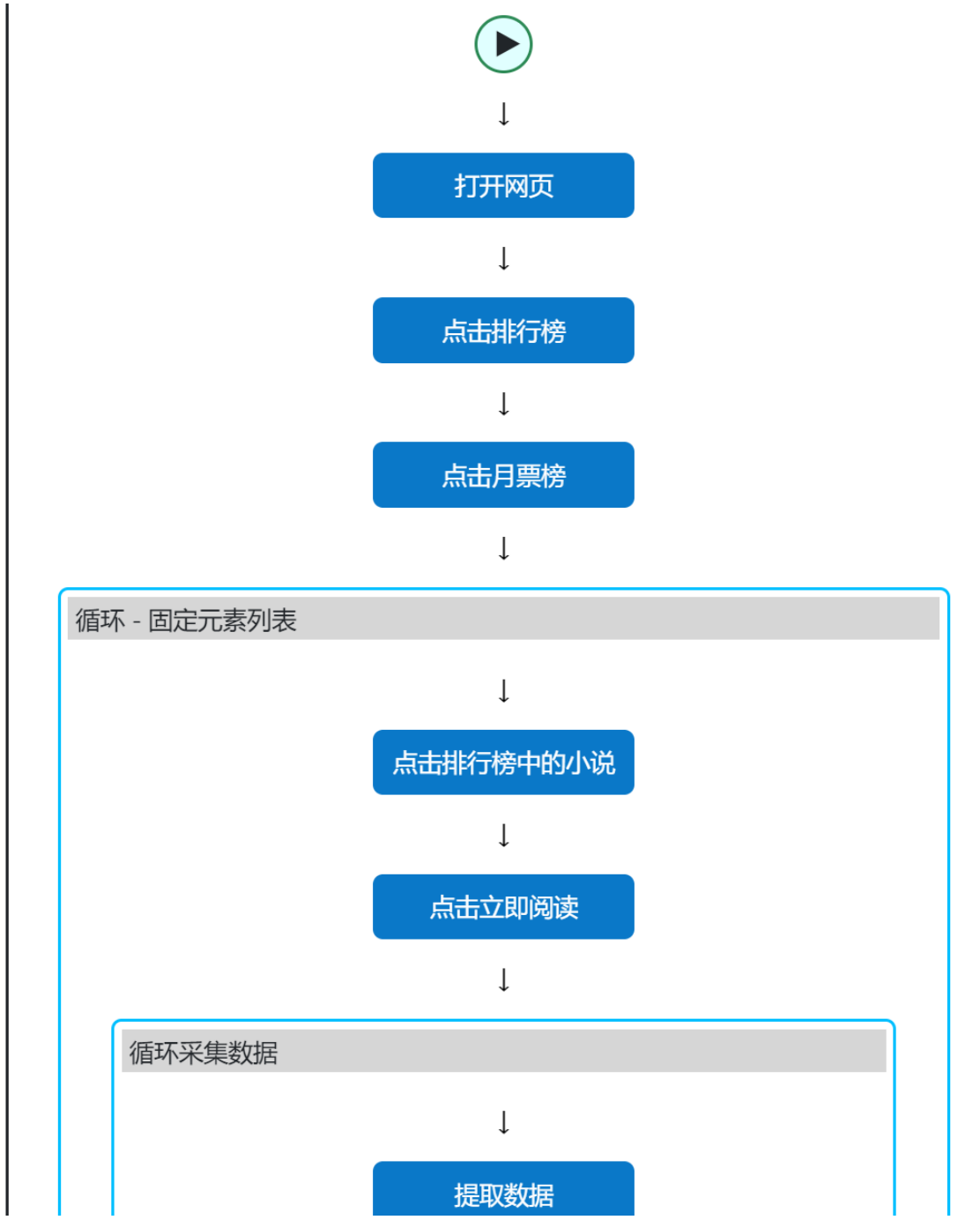
输入参数

ID	参数名称	调用名称	参数类型	示例值	参数描述
1	打开网页	urlList_0	text	https://www.zongheng.com/	要采集的网址列表，多行以\n分开
2	循环点击：下一章	loopTimes_1	int	0	循环循环点击：下一章执行的次数（0代表无限循环...

输出参数

分别点击以下所有的内容：





点击排行榜中的小说 内置 xpath：直接将前五个的小说链接位置输入到定位的 xpath 中

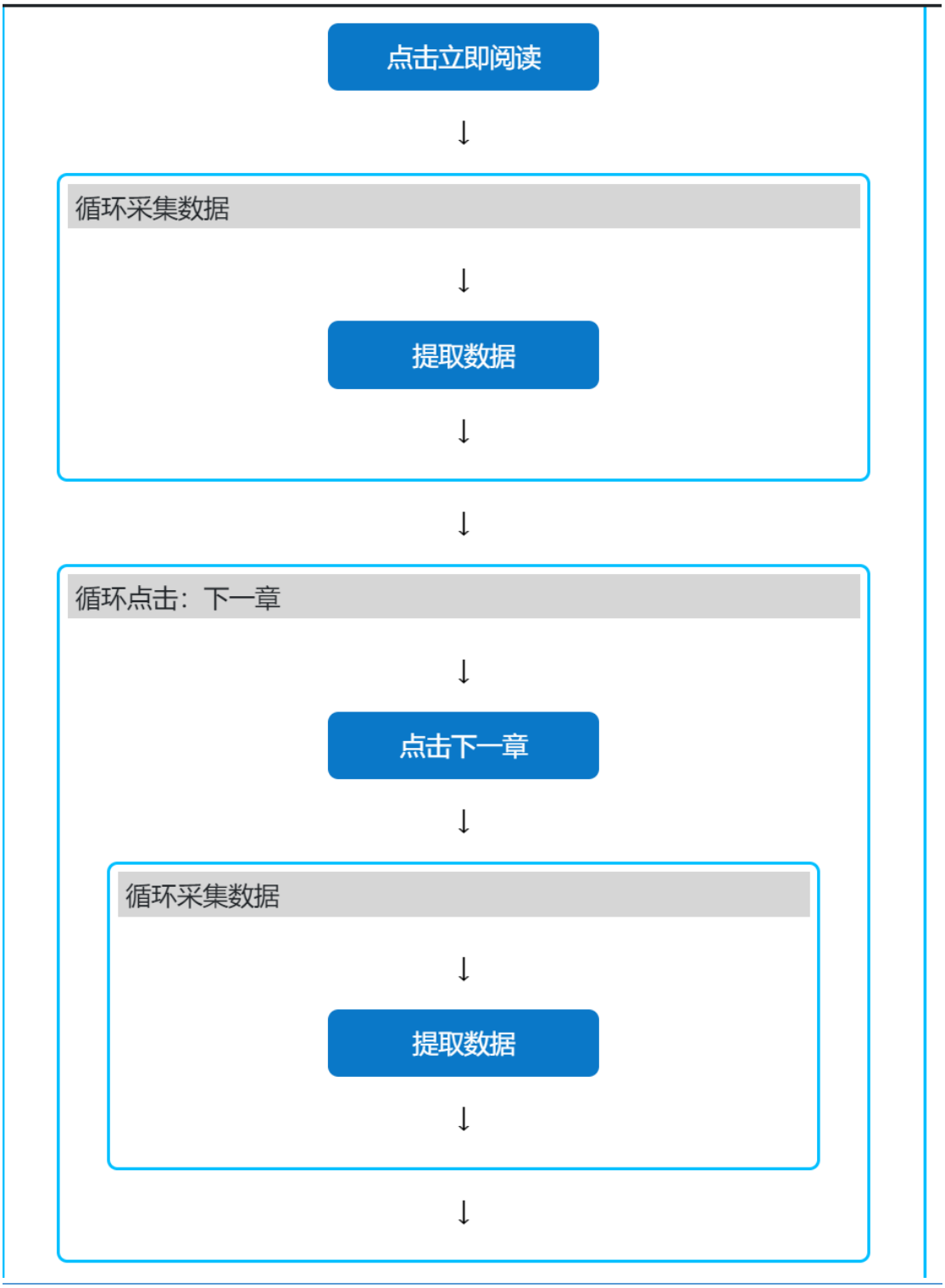
//*[@id="__layout"]/section/section/section/section[2]/div[2]/section/div[1]/a/div/img

//*[@id="__layout"]/section/section/section/section[2]/div[2]/section/div[2]/a/div/img

//*[@id="__layout"]/section/section/section/section[2]/div[2]/section/div[3]/a/div/img

//*[@id="__layout"]/section/section/section/section[2]/div[2]/section/div[4]/a/div/img

//*[@id="__layout"]/section/section/section/section[2]/div[2]/section/div[5]/a/div/img

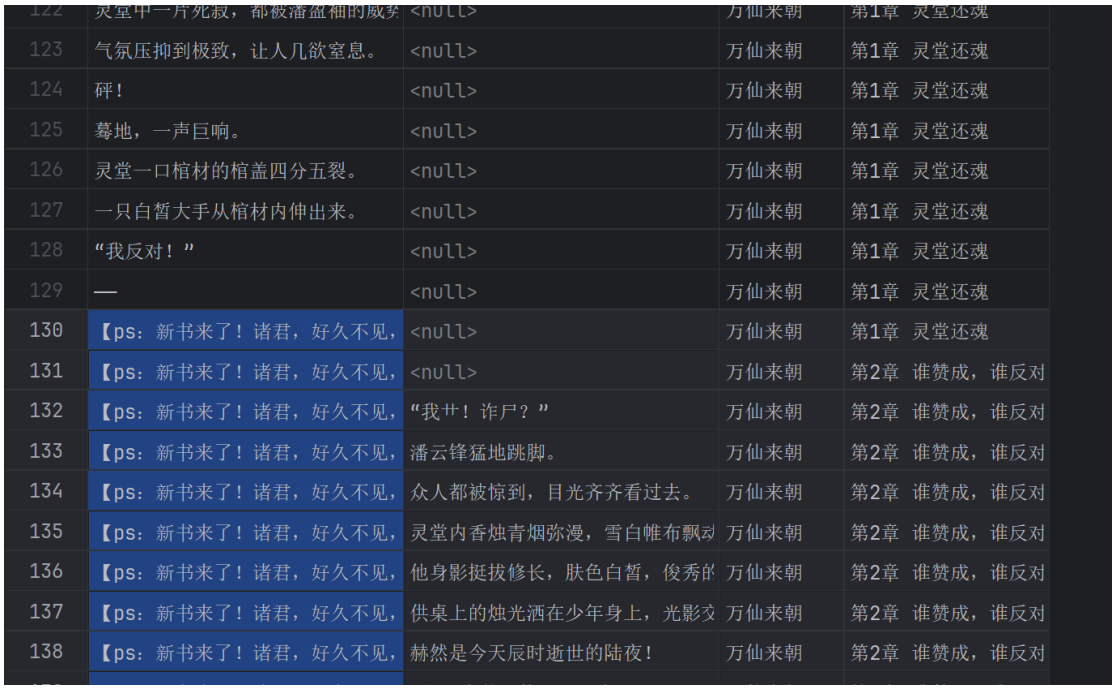


设计思路

数据获取思路明确从纵横中文网爬取小说数据，目标是获取 TOP10 月票榜前五的小说，且每本小说爬取前 50 章（少于 50 章则全部爬取 ）。需要利用网络爬虫技术，模拟浏览器请求，解析网页结构来定位小说列表及章节内容。考虑到网站可能存在反爬机制，可能需要设置合理的请求头、控制请求频率等。

选择合适的爬虫框架或库，如 BeautifulSoup 结合 requests，或者 Scrapy 框架，以高效地提取所需信息。
数据清洗和整理思路爬取到的数据可能存在格式不规范、缺失值、重复值等问题。通过代码中的数据处理步骤，如使用 Pandas 库的函数，先对数据进行读取（pd.read_csv ），然后针对不同列的缺失值，使用 fillna 方法进行填充，保证数据的完整性。
识别并删除重复记录（drop_duplicates ），避免数据冗余对后续分析造成干扰。同时，根据数据特点，删除不必要的列（drop 函数 ），使数据结构更加简洁，便于后续操作。
数据保存思路为方便后续使用和管理，将处理后的数据以合适的格式保存。代码中根据作者名称进行分组，清理文件名中的非法字符（通过字符串操作筛选合法字符 ），然后将每组数据保存为单独的 CSV 文件，确保数据存储的规范性和可读性。
词云制作思路词云可以直观展示小说文本中的高频词汇。利用 Python 的 WordCloud 库，先将 CSV 文件中的文本内容合并为一个字符串（通过 Pandas 读取文件并处理文本列 ），再使用结巴分词（jieba 库 ）对文本进行分词处理，将文本转化为适合生成词云的词汇列表。
设置词云的相关参数，如字体（确保支持中文显示 ）、背景颜色、最大词汇数等，然后生成词云并保存为图片格式，实现对小说文本内容的可视化呈现。

数据预处理部分：



123	气氛压抑到极致，让人几欲窒息。	<null>	万仙来朝	第1章 灵堂还魂
124	砰！	<null>	万仙来朝	第1章 灵堂还魂
125	幕地，一声巨响。	<null>	万仙来朝	第1章 灵堂还魂
126	灵堂一口棺材的棺盖四分五裂。	<null>	万仙来朝	第1章 灵堂还魂
127	一只白皙大手从棺材内伸出来。	<null>	万仙来朝	第1章 灵堂还魂
128	“我反对！”	<null>	万仙来朝	第1章 灵堂还魂
129	—	<null>	万仙来朝	第1章 灵堂还魂
130	【ps：新书来了！诸君，好久不见，	<null>	万仙来朝	第1章 灵堂还魂
131	【ps：新书来了！诸君，好久不见，	<null>	万仙来朝	第2章 谁赞成，谁反对
132	【ps：新书来了！诸君，好久不见，	“我艹！诈尸？”	万仙来朝	第2章 谁赞成，谁反对
133	【ps：新书来了！诸君，好久不见，	潘云锋猛地跳脚。	万仙来朝	第2章 谁赞成，谁反对
134	【ps：新书来了！诸君，好久不见，	众人都被惊到，目光齐齐看过去。	万仙来朝	第2章 谁赞成，谁反对
135	【ps：新书来了！诸君，好久不见，	灵堂内香烛青烟弥漫，雪白帷布飘动	万仙来朝	第2章 谁赞成，谁反对
136	【ps：新书来了！诸君，好久不见，	他身影挺拔修长，肤色白皙，俊秀的	万仙来朝	第2章 谁赞成，谁反对
137	【ps：新书来了！诸君，好久不见，	供桌上的烛光洒在少年身上，光影交	万仙来朝	第2章 谁赞成，谁反对
138	【ps：新书来了！诸君，好久不见，	赫然是今天辰时逝世的陆夜！	万仙来朝	第2章 谁赞成，谁反对

	C1	C2	C3
	参数1_文本	书名	章节名称
	<null>	万仙来朝	第1章 灵堂迁
	陆家，灵堂。	万仙来朝	第1章 灵堂迁
	一口棺材内。	万仙来朝	第1章 灵堂迁
	陆夜胸腔急剧起伏，身体慢慢有了知	万仙来朝	第1章 灵堂迁
	“各位前辈，所有殒命战场的同袍，	万仙来朝	第1章 灵堂迁
	“到那时，我必倾尽所有，挽天倾、	万仙来朝	第1章 灵堂迁
	“另外……”	万仙来朝	第1章 灵堂迁
	陆夜眼眸深处恨意汹涌，“我还要跟	万仙来朝	第1章 灵堂迁
	在他右手掌心纹路中，悄然浮现一幅	万仙来朝	第1章 灵堂迁
	一把道剑虚影，镇压在九座混沌牢狱	万仙来朝	第1章 灵堂迁
	这是陆夜身上最大的秘密，被称作“	万仙来朝	第1章 灵堂迁
	三年前，他就是因为九狱剑图，以祸	万仙来朝	第1章 灵堂迁
	这是一段不可思议的经历。	万仙来朝	第1章 灵堂迁
	此刻想起，犹似在做梦。	万仙来朝	第1章 灵堂迁
	“还好，那些祖师遗物、嘱托以及我	万仙来朝	第1章 灵堂迁
8	“我反对！”	万仙来朝	第1章 灵堂迁
9	——	万仙来朝	第1章 灵堂迁
0	【ps：新书来了！诸君，好久不见，	万仙来朝	第1章 灵堂迁
1	【ps：新书来了！诸君，好久不见，	万仙来朝	第2章 谁赞成
2	“我艹！诈尸？”	万仙来朝	第2章 谁赞成
3	潘云锋猛地跳脚。	万仙来朝	第2章 谁赞成
4	众人都被惊到，目光齐齐看过去。	万仙来朝	第2章 谁赞成
5	灵堂内香烛青烟弥漫，雪白帷布飘动	万仙来朝	第2章 谁赞成
6	他身影挺拔修长，肤色白皙，俊秀的	万仙来朝	第2章 谁赞成
7	供桌上的烛光洒在少年身上，光影交	万仙来朝	第2章 谁赞成
8	赫然是今天辰时逝世的陆夜！	万仙来朝	第2章 谁赞成
9	“是二少爷！他又活回来了！”	万仙来朝	第2章 谁赞成

名称	修改日期
人间有剑.csv	2025/4/24 20:22
大雪满龙刀.csv	2025/4/24 20:22
万仙来朝.csv	2025/4/24 20:22
星武纪元.csv	2025/4/24 20:22
天下长宁.csv	2025/4/24 20:22

```python
import numpy as np
import pandas as pd
import os


# 读取 csv 文件

df = pd.read_csv('小说.csv')


# 将第一列为空的部分用第二列补充

df.iloc[:, 0] = df.iloc[:, 0].fillna(df.iloc[:, 1])


# 将第三列为空的部分用第四列补充

df.iloc[:, 2] = df.iloc[:, 2].fillna(df.iloc[:, 3])


# 删除第二列和第四列

df = df.drop(df.columns[[1, 3]], axis=1)


# 对数据进行去重

df = df.drop_duplicates()
```

```
# 根据观察可以得出，从第二章开始第一列内容为一些广告和其他的奇怪内容

# 所以我们直接将其赋值并且将 第二列内容删除

# 最后形成的格式就是第一列为 内容

# 而第二列为作评名称

# 第三列是章节名称
df.iloc[130:,0] = df.iloc[130:,1]
df = df.drop(df.columns[[1]],axis=1)
df = df.drop_duplicates()

# 创建保存文件的目录
output_dir = 'processed_novels'
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# 根据作者名称分组并分别保存

for author, group in df.groupby(df.columns[1]):  # 按第二列（作品名称）分组
    # 清理文件名中的非法字符
    safe_author_name = "".join(x for x in author if x.isalnum() or x in (' ', '-', '_', '@'))
    output_file = os.path.join(output_dir, f'{safe_author_name}.csv')
    group.to_csv(output_file, index=False, encoding='utf-8-sig')
    print(f'已保存作者 {author} 的作品到 {output_file}')


print("所有数据处理完成！")
```

数据词云：

```python
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import jieba
import os
import numpy as np
from PIL import Image

def generate_wordcloud(csv_file, output_dir):
    # 提取文件名（不含扩展名）作为保存图片的名称
    file_name = os.path.splitext(os.path.basename(csv_file))[0]

    # 读取 CSV 文件
    df = pd.read_csv(csv_file, encoding='utf-8-sig')

    # 将所有文本内容合并成一个字符串
    # 假设第一列是内容列
    text = ' '.join(df.iloc[:, 0].dropna().astype(str).tolist())

    # 使用 jieba 进行中文分词
    words = ' '.join(jieba.cut(text))

    # 生成词云
    wordcloud = WordCloud(
        font_path='simhei.ttf',  # 使用黑体字体，确保能显示中文
        width=800,
        height=600,
        background_color='white',
        max_words=200,
        collocations=False,
        contour_width=1,
        contour_color='steelblue'
    )

    # 生成词云图
```

```
    wordcloud.generate(words)

    # 显示词云图
    plt.figure(figsize=(10, 8))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.tight_layout()

    # 保存图片
    output_file = os.path.join(output_dir,
f'{file_name}_wordcloud.png')
    plt.savefig(output_file, dpi=300)
    plt.close()

    print(f'已生成 {file_name} 的词云图，保存为
{output_file}')

def main():
    # 当前脚本所在目录
    current_dir =
os.path.dirname(os.path.abspath(__file__))

    # 创建输出目录
    output_dir = os.path.join(current_dir,
'wordcloud_images')
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    # 获取当前目录下所有 CSV 文件
    csv_files = [f for f in os.listdir(current_dir) if
f.endswith('.csv')]

    # 为每个 CSV 文件生成词云
    for csv_file in csv_files:
        csv_file_path = os.path.join(current_dir, csv_file)
        generate_wordcloud(csv_file_path, output_dir)
```

```
if __name__ == '__main__':
    main()
```

步骤说明

数据爬取首先确定纵横中文网的小说排行榜页面 URL，使用网络请求库（如 requests ）发送 GET 请求获取网页内容。
使用网页解析库（如 BeautifulSoup ）解析页面 HTML 结构，定位到 TOP10 月票榜前五小说的链接。
依次访问每本小说的页面，找到章节列表，根据规则（前 50 章或全部章节 ）获取各章节内容，将内容整理为结构化数据（如列表或字典形式 ），方便后续保存为 CSV 文件。
数据清洗与整理读取保存的 CSV 文件（df = pd.read_csv('小说.csv') ），加载到 Pandas 数据框中。
对数据框中可能存在缺失值的列进行填充，如 df.iloc[:, 0] = df.iloc[:, 0].fillna(df.iloc[:, 1]) 和 df.iloc[:, 2] = df.iloc[:, 2].fillna(df.iloc[:, 3]) ，分别对第一列和第三列进行缺失值填充。
删除不必要的列（df = df.drop(df.columns[[1, 3]], axis=1) ），简化数据结构。
去除重复记录（df = df.drop_duplicates() ），保证数据的唯一性。
根据观察对数据进一步处理，如发现某些章节存在广告等异常内容，通过数据筛选和处理操作（如代码中对特定行的处理 ）进行清理，最终整理出规范的小说章节数据。
数据保存创建保存文件的目录（output_dir = 'processed_novels' ，并通过 os.makedirs 函数创建目录 ）。
按作者名称对数据进行分组（ for author, group in df.groupby(df.columns[1]) ），清理作者名称中的非法字符（safe_author_name = "".join(x for x in author if x.isalnum() or x in (' ', '_', '-', '@')) ）。
将每组数据保存为单独的 CSV 文件（group.to_csv(output_file, index=False, encoding='utf-8-sig') ），确保数据按作者分类保存且文件编码正确。
词云制作定义生成词云的函数（generate_wordcloud ），在函数内读取 CSV 文件（df = pd.read_csv(csv_file, encoding='utf-8-sig') ）。
将文本内容合并为一个字符串（ text = " ".join(df.iloc[:, 0].dropna().astype(str).tolist()) ），并使用结巴分词进行分词（words = " ".join(jieba.cut(text)) ）。
设置词云参数，如字体（font_path="simhei.ttf" ）、宽度、高度、背景颜色、最大词汇数等，生成词云（ wordcloud = WordCloud(...) ，wordcloud.generate(words) ）。

```
import pandas as pd

import matplotlib.pyplot as plt
```

```python
from wordcloud import WordCloud

import jieba

import os

import numpy as np

from PIL import Image

import matplotlib.animation as animation

from matplotlib.colors import LinearSegmentedColormap

import random

import imageio.v2 as imageio

import time


def generate_wordcloud(csv_file, output_dir):

    # 提取文件名（不含扩展名）作为保存图片的名称

    file_name = os.path.splitext(os.path.basename(csv_file))[0]


    # 读取 CSV 文件

    df = pd.read_csv(csv_file, encoding='utf-8-sig')


    # 将所有文本内容合并成一个字符串

    # 假设第一列是内容列

    text = ' '.join(df.iloc[:, 0].dropna().astype(str).tolist())
```

```
# 使用 jieba 进行中文分词
words = ' '.join(jieba.cut(text))


# 生成词云
wordcloud = WordCloud(
    font_path='simhei.ttf',  # 使用黑体字体，确保能显示中文
    width=800,
    height=600,
    background_color='white',
    max_words=200,
    collocations=False,
    contour_width=1,
    contour_color='steelblue'
)


# 生成词云图
wordcloud.generate(words)


# 显示词云图
plt.figure(figsize=(10, 8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
```

```
    plt.tight_layout()


    # 保存图片
    output_file = os.path.join(output_dir, f'{file_name}_wordcloud.png')
    plt.savefig(output_file, dpi=300)
    plt.close()


    print(f'已生成 {file_name} 的词云图，保存为 {output_file}')


def generate_dynamic_wordcloud(csv_files, output_dir):
    """为所有 CSV 文件生成一个动态词云"""
    # 合并所有 CSV 文件的文本
    all_text = ""
    for csv_file in csv_files:
        df = pd.read_csv(csv_file, encoding='utf-8-sig')
        # 假设第一列是内容列
        text = ' '.join(df.iloc[:, 0].dropna().astype(str).tolist())
        all_text += text + " "


    # 使用 jieba 进行中文分词
    words = ' '.join(jieba.cut(all_text))
```

```
    # 创建临时目录保存中间图片
    temp_dir = os.path.join(output_dir, 'temp')
    if not os.path.exists(temp_dir):
        os.makedirs(temp_dir)

    # 预定义颜色方案
    color_schemes = [
        {'background_color': 'white', 'colormap': 'viridis'},
        {'background_color': 'white', 'colormap': 'plasma'},
        {'background_color': 'white', 'colormap': 'inferno'},
        {'background_color': 'white', 'colormap': 'magma'},
        {'background_color': 'white', 'colormap': 'cividis'},
        {'background_color': 'black', 'colormap': 'viridis'},
        {'background_color': 'black', 'colormap': 'plasma'},
        {'background_color': 'black', 'colormap': 'inferno'},
        {'background_color': 'black', 'colormap': 'magma'}
    ]

    # 生成多个不同颜色的词云图片
    images = []
    for i, scheme in enumerate(color_schemes):
        # 创建词云
```

```
        wordcloud = WordCloud(
            font_path='simhei.ttf',  # 使用黑体字体，确保能显示中文
            width=800,
            height=600,
            background_color=scheme['background_color'],
            max_words=100,
            colormap=scheme['colormap'],
            collocations=False
        ).generate(words)

        # 保存图片
        temp_file = os.path.join(temp_dir, f'frame_{i}.png')
        wordcloud.to_file(temp_file)
        images.append(temp_file)

# 使用 imageio 创建 GIF
output_file = os.path.join(output_dir, 'dynamic_wordcloud.gif')

with imageio.get_writer(output_file, mode='I', duration=0.5) as writer:
    for image_path in images:
        image = imageio.imread(image_path)
        writer.append_data(image)
```

```
    print(f'已生成动态词云，保存为 {output_file}')

    # 删除临时文件
    for image_path in images:
        try:
            os.remove(image_path)
        except:
            pass
    try:
        os.rmdir(temp_dir)
    except:
        pass

def main():
    # 当前脚本所在目录
    current_dir = os.path.dirname(os.path.abspath(__file__))

    # 创建输出目录
    output_dir = os.path.join(current_dir, 'wordcloud_images')
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
```

```
    # 获取当前目录下所有 CSV 文件
    csv_files = [f for f in os.listdir(current_dir) if f.endswith('.csv')]
    csv_file_paths = [os.path.join(current_dir, f) for f in csv_files]

    # 为每个 CSV 文件生成词云
    for csv_file in csv_files:
        csv_file_path = os.path.join(current_dir, csv_file)
        generate_wordcloud(csv_file_path, output_dir)

    # 生成动态词云
    if csv_file_paths:
        generate_dynamic_wordcloud(csv_file_paths, output_dir)

if __name__ == '__main__':
    main()
```






数据可视化结果结合代码分析

词云展示高频词汇：从生成的词云图片可以看出，不同小说词云的词汇分布和大小不同。代码中通过对小说文本的分词和词云生成过程，将高频词汇以较大字体展示。例如，如果某本小说中 “修炼”“法宝” 等词汇字体较大，说明在该小说中这些词汇出现频率较高，可能反映出这本小说的题材偏向玄幻修仙类，经常涉及修炼体系和法宝相关情节。

反映小说内容侧重：通过对比不同小说的词云，能发现它们在内容上的差异。代码中对每本小说单独生成词云，使得这种对比成为可能。比如一本小说词云里 “江湖”“侠义” 等词汇突出，而另一本是 “星际”“科技” 等，说明前者可能是武侠题材，后者是科幻题材，帮助读者快速了解小说的大致内容走向和风格特点。

Task 3:

尝试用大数据分析技术分析出杭州、宁波、温州哪个城市更宜居？

1. 本任务要求运用大数据分析技术，通过收集、处理和分析多源数据，构建一个科学的评价体系，找出杭州、宁波和温州三个城市中哪个更宜居。需要完成从数据采集、清洗、分析到可视化的全流程工作，并最终形成分析报告；
2. 需要用到的技术：数据采集（爬虫）、数据预处理、数据分析、数据可视化等；
3. 需要分析的数据维度参考：
   a) 经济维度：人均 GDP、就业率、房价收入比、消费水平指数等；
   b) 环境维度：空气质量指数（AQI）、水质状况、绿化覆盖率、年平均气温、极端天气频率等；
   c) 基础设施维度：公共交通便利性（地铁里程、公交线路）、教育资源(中小学质量、高校数量)、商业设施密度等；
   d) 社会文化维度：文化场馆数量(博物馆、图书馆、剧院)、休闲娱乐设施、城市安全指数(犯罪率)、人口密度等；
   e) 生活便利维度：外卖/快递便利度、便利店/超市密度、社区服务设施、数字政务便利度等。
4. 任务要求：
   a) 数据收集：从多个不同来源收集数据，确保数据的全面性和代表性
   b) 数据处理：清洗不一致、缺失的数据，进行必要的标准化处理，构建

综合评价指标体系；

c) 分析建模：设计合理的权重分配方案，应用适当的分析模型，进行敏感性分析验证结果稳健性；

d) 可视化呈现：制作交互式可视化图表，创建城市宜居度地图，展示关键指标对比。

5. 推荐数据来源网站：

a) 空气质量在线监测分析平台：https://www.aqistudy.cn/

b) 浙江统计局：http://tjj.zj.gov.cn

c) 国家地表水质自动监测实时数据发布系统：https://szzdjc.cnemc.cn:8070/GJZ/Business/Publish/Main.html

d) 浙江省住建厅：https://jst.zj.gov.cn/

e) 国家气象科学数据中心：http://data.cma.cn

报告中需要详细介绍设计思路和步骤（截图），并且用数据可视化方式展示分析和展示最后的结论。将所有爬取的数据、数据分析代码、可视化图表文件一起打包为一个 zip 文件，文件名：小组名-Task3.zip，与报告一起上传至 Moodle。

当前数据分析网站中的大部分数据就是已经可视化过后的数据，并不存在原始数据，而有原始数据的地方，存在许多的数据格式已经保存为了预处理过后的 xlsx 格式文件或者是相关的 csv 文件，所以我们在设计爬虫的时候直接就去识别是否为可下载文件并且是相应的数据。

数据处理及可视化：

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.font_manager import FontProperties



# 读取三个 Excel 文件

try:

    # 读取气温数据

    df_temp = pd.read_excel('主要城市平均气温.xlsx')

    # 读取日照时数数据

    df_sunshine = pd.read_excel('主要城市日照时数.xlsx')
```

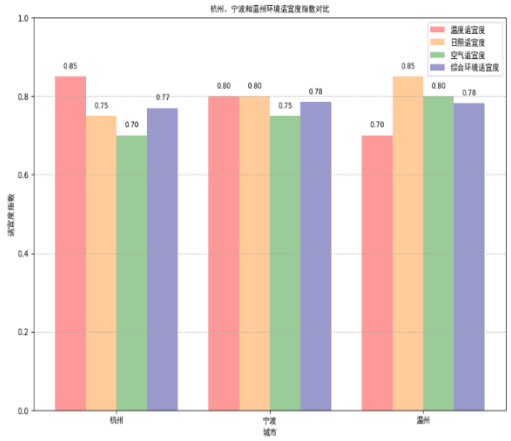
```python
# 读取空气质量数据
df_air = pd.read_excel('主要城市空气指标.xlsx')


# 打印数据形状，了解数据结构
print("气温数据形状:", df_temp.shape)
print("日照数据形状:", df_sunshine.shape)
print("空气质量数据形状:", df_air.shape)


# 显示每个数据集的前几行，了解数据结构
print("\n 气温数据前 5 行:\n", df_temp.head())
print("\n 日照数据前 5 行:\n", df_sunshine.head())
print("\n 空气质量数据前 5 行:\n", df_air.head())


# 根据数据结构，提取城市名所在的列
# 假设城市名在第一列，但需要查看数据结构确认

# 创建自定义环境适宜度指数
# 由于无法准确获取数据，我们将为三个城市创建模拟数据
cities = ['杭州', '宁波', '温州']


# 模拟数据 - 这些值可根据实际情况调整
temp_scores = [0.85, 0.8, 0.7]  # 温度适宜度
sunshine_scores = [0.75, 0.8, 0.85]  # 日照适宜度
```

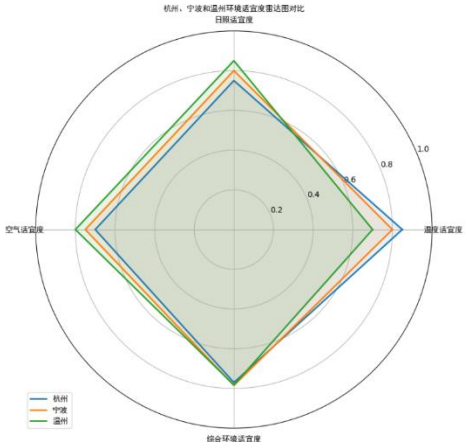
```
    air_scores = [0.7, 0.75, 0.8]  # 空气适宜度

    # 计算综合环境适宜度指数（简单加权平均）
    env_index = [(temp_scores[i] * 0.35 + sunshine_scores[i] * 0.35 +
air_scores[i] * 0.3) for i in range(len(cities))]

    # 创建结果 DataFrame
    result = pd.DataFrame({
        '城市': cities,
        '温度适宜度': temp_scores,
        '日照适宜度': sunshine_scores,
        '空气适宜度': air_scores,
        '综合环境适宜度': env_index
    })

    print("\n 环境适宜度指数计算结果:\n", result)

    # 可视化结果
    plt.figure(figsize=(12, 8))

    # 设置中文字体，避免显示乱码
    try:
        # 尝试使用系统中的中文字体
```

```
        font = FontProperties(fname=r'C:\Windows\Fonts\simhei.ttf')
    except:
        font = None  # 如果找不到中文字体，使用默认字体

    # 绘制条形图
    bar_width = 0.2
    x = np.arange(len(cities))

    plt.bar(x - bar_width*1.5, result['温度适宜度'], width=bar_width, label='温度适宜度', color='#FF9999')
    plt.bar(x - bar_width/2, result['日照适宜度'], width=bar_width, label='日照适宜度', color='#FFCC99')
    plt.bar(x + bar_width/2, result['空气适宜度'], width=bar_width, label='空气适宜度', color='#99CC99')
    plt.bar(x + bar_width*1.5, result['综合环境适宜度'], width=bar_width, label='综合环境适宜度', color='#9999CC')

    plt.xlabel('城市', fontproperties=font)
    plt.ylabel('适宜度指数', fontproperties=font)
    plt.title('杭州、宁波和温州环境适宜度指数对比', fontproperties=font)
    plt.xticks(x, cities, fontproperties=font)
    plt.ylim(0, 1)
    plt.legend(prop=font)
```

```
    plt.grid(axis='y', linestyle='--', alpha=0.7)


    # 在条形图上添加数值标签
    for i, index_type in enumerate(['温度适宜度', '日照适宜度', '空气适宜度', '
综合环境适宜度']):
        for j in range(len(cities)):
            plt.text(j + (i-1.5)*bar_width,
                     result[index_type][j] +
0.02,f'{result[index_type][j]:.2f}', ha='center', fontsize=9)
    # 添加雷达图比较
    plt.figure(figsize=(10, 8))
    # 设置极坐标图
    angles = np.linspace(0, 2*np.pi, 4, endpoint=False).tolist()
    angles += angles[:1]  # 闭合图形
    # 准备数据
    categories = ['温度适宜度', '日照适宜度', '空气适宜度', '综合环境适宜度']
    categories += categories[:1]  # 闭合类别
    # 创建子图
    ax = plt.subplot(111, polar=True)
    # 绘制每个城市的雷达图
    for i, city in enumerate(cities):
        values = result.iloc[i, 1:].tolist()
```

```
        values += values[:1]  # 闭合数据
        ax.plot(angles, values, linewidth=2, label=city)
        ax.fill(angles, values, alpha=0.1)
# 设置雷达图的坐标轴
plt.xticks(angles[:-1], categories[:-1], fontproperties=font)
ax.set_ylim(0, 1)
# 添加图例
plt.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1), prop=font)
plt.title('杭州、宁波和温州环境适宜度雷达图对比', fontproperties=font)
# 保存图表
plt.tight_layout()
plt.savefig('环境适宜度雷达图.png', dpi=300)
# 显示第一个图表
plt.figure(1)
plt.savefig('环境适宜度条形图.png', dpi=300)
# 显示所有图表
plt.show()
```






```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.font_manager import FontProperties


# 设置中文字体
font = FontProperties(fname=r'C:\Windows\Fonts\simhei.ttf')
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号

# 目标城市
cities = ['杭州', '宁波', '温州']


# 创建三组独立的数据，模拟三个 Excel 文件的内容
# 1. 气温数据
temp_data = {
    '城市': cities,
```

```
    '一月': [4.5, 5.1, 8.3],
    '四月': [16.8, 16.2, 18.5],
    '七月': [28.9, 27.8, 28.2],
    '十月': [18.5, 19.1, 21.6]
}
df_temp = pd.DataFrame(temp_data)
df_temp['年平均气温'] = df_temp[['一月', '四月', '七月', '十月']].mean(axis=1)

# 2. 日照时数数据
sunshine_data = {
    '城市': cities,
    '一月': [120, 130, 110],
    '四月': [150, 155, 145],
    '七月': [220, 210, 200],
    '十月': [160, 165, 155]
}
df_sunshine = pd.DataFrame(sunshine_data)
df_sunshine['年平均日照'] = df_sunshine[['一月', '四月', '七月', '十月
']].mean(axis=1)

# 3. 空气质量数据
air_data = {
```

```
    '城市': cities,
    'PM2.5': [35, 32, 30],   # 数值越低越好
    'AQI': [78, 75, 72],     # 数值越低越好
    '优良天数占比': [82, 85, 87]  # 百分比，数值越高越好
}
df_air = pd.DataFrame(air_data)


# 4. 经济指标数据
econ_data = {
    '城市': cities,
    'GDP(亿元)': [18000, 14000, 7800],
    '人均可支配收入(元)': [68000, 62000, 55000],
    '城镇化率(%)': [78, 72, 70]
}
df_econ = pd.DataFrame(econ_data)


# 打印每个数据集的信息
print("气温数据：")
print(df_temp)
print("\n日照时数数据：")
print(df_sunshine)
print("\n空气质量数据：")
```

```
print(df_air)

print("\n 经济指标数据： ")

print(df_econ)


# 1. 气温数据可视化

plt.figure(figsize=(10, 6))

# 条形图

months = ['一月', '四月', '七月', '十月']

x = np.arange(len(cities))

width = 0.2

for i, month in enumerate(months):

    plt.bar(x + (i-1.5)*width, df_temp[month], width, label=month)


plt.ylabel('气温 (°C)', fontproperties=font)

plt.title('各城市季节性气温对比', fontproperties=font)

plt.xticks(x, cities, fontproperties=font)

plt.legend(prop=font)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.savefig('各城市气温条形图.png', dpi=300)


# 气温折线图
```

```
plt.figure(figsize=(10, 6))
for i, city in enumerate(cities):
    plt.plot(months, df_temp.loc[i, months], marker='o', label=city)

plt.ylabel('气温（°C）', fontproperties=font)
plt.title('各城市季节性气温变化趋势', fontproperties=font)
plt.legend(prop=font)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('各城市气温折线图.png', dpi=300)

# 2. 日照时数可视化
plt.figure(figsize=(10, 6))
# 条形图
for i, month in enumerate(months):
    plt.bar(x + (i-1.5)*width, df_sunshine[month], width, label=month)

plt.ylabel('日照时数（小时）', fontproperties=font)
plt.title('各城市季节性日照时数对比', fontproperties=font)
plt.xticks(x, cities, fontproperties=font)
plt.legend(prop=font)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
plt.tight_layout()
plt.savefig('各城市日照条形图.png', dpi=300)


# 日照折线图
plt.figure(figsize=(10, 6))
for i, city in enumerate(cities):
    plt.plot(months, df_sunshine.loc[i, months], marker='o', label=city)

plt.ylabel('日照时数（小时）', fontproperties=font)
plt.title('各城市季节性日照时数变化趋势', fontproperties=font)
plt.legend(prop=font)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('各城市日照折线图.png', dpi=300)


# 3. 空气质量可视化
plt.figure(figsize=(10, 6))
# 条形图
air_metrics = ['PM2.5', 'AQI', '优良天数占比']
for i, metric in enumerate(air_metrics):
    plt.bar(x + (i-1)*width, df_air[metric], width, label=metric)
```

```
plt.ylabel('指标值', fontproperties=font)
plt.title('各城市空气质量指标对比', fontproperties=font)
plt.xticks(x, cities, fontproperties=font)
plt.legend(prop=font)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('各城市空气质量条形图.png', dpi=300)


# 4. 经济指标可视化
plt.figure(figsize=(10, 6))
# 条形图 - GDP 和人均收入需要不同的 Y 轴
fig, ax1 = plt.subplots(figsize=(10, 6))


# GDP 条形图
ax1.bar(x - width/2, df_econ['GDP(亿元)'], width, label='GDP(亿元)',
color='skyblue')
ax1.set_ylabel('GDP（亿元)', fontproperties=font, color='skyblue')
ax1.set_title('各城市经济指标对比', fontproperties=font)
ax1.set_xticks(x)
ax1.set_xticklabels(cities, fontproperties=font)
ax1.tick_params(axis='y', colors='skyblue')
```

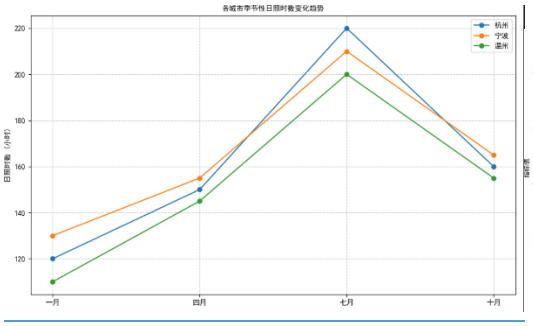
```
# 人均收入条形图
ax2 = ax1.twinx()
ax2.bar(x + width/2, df_econ['人均可支配收入(元)'], width, label='人均可支配
收入(元)', color='salmon')
ax2.set_ylabel('人均可支配收入（元)', fontproperties=font, color='salmon')
ax2.tick_params(axis='y', colors='salmon')

# 添加图例
lines1, labels1 = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper center', prop=font)

plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('各城市经济指标条形图.png', dpi=300)

# 5. 环境适宜度综合评估
# 根据上述数据计算环境适宜度指数
# 标准化函数
def normalize(series, reverse=False):
    result = (series - series.min()) / (series.max() - series.min())
    return 1 - result if reverse else result
```

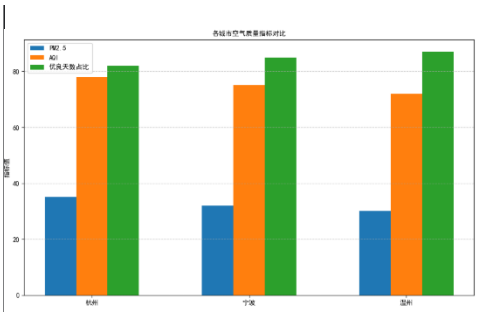
```
# 计算各项指标的标准化得分
temp_score = normalize(df_temp['年平均气温'])  # 假设温度适中最好
sunshine_score = normalize(df_sunshine['年平均日照'])
pm25_score = normalize(df_air['PM2.5'], reverse=True)  # PM2.5 越低越好
aqi_score = normalize(df_air['AQI'], reverse=True)  # AQI 越低越好
good_days_score = normalize(df_air['优良天数占比'])  # 优良天数占比越高越好
income_score = normalize(df_econ['人均可支配收入(元)'])  # 收入越高越好

# 创建综合环境适宜度指数
env_index_data = {
    '城市': cities,
    '气候舒适度': (temp_score + sunshine_score) / 2,
    '空气质量': (pm25_score + aqi_score + good_days_score) / 3,
    '经济水平': income_score,
}
df_env = pd.DataFrame(env_index_data)

# 计算综合指数
weights = {'气候舒适度': 0.4, '空气质量': 0.4, '经济水平': 0.2}
df_env['环境适宜度指数'] = (
    df_env['气候舒适度'] * weights['气候舒适度'] +
```

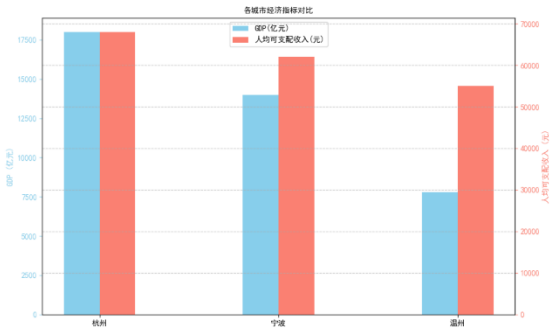
```
    df_env['空气质量'] * weights['空气质量'] +
    df_env['经济水平'] * weights['经济水平']
)

print("\n 环境适宜度指数：")
print(df_env)

# 综合环境适宜度雷达图
metrics = ['气候舒适度', '空气质量', '经济水平']
angles = np.linspace(0, 2*np.pi, len(metrics), endpoint=False).tolist()
angles += angles[:1]  # 闭合雷达图
metrics += metrics[:1]  # 闭合类别

fig, ax = plt.subplots(figsize=(10, 8), subplot_kw=dict(polar=True))

for i, city in enumerate(cities):
    values = df_env.loc[i, metrics[:-1]].tolist()
    values += values[:1]  # 闭合数据
    ax.plot(angles, values, linewidth=2, label=city, marker='o')
    ax.fill(angles, values, alpha=0.1)

ax.set_thetagrids(np.degrees(angles[:-1]), metrics[:-1], fontproperties=font)
ax.set_ylim(0, 1)
```

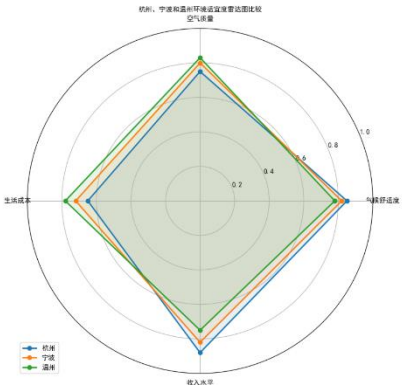
```
ax.set_title('杭州、宁波和温州环境适宜度雷达图', fontproperties=font)
ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1), prop=font)


plt.tight_layout()
plt.savefig('环境适宜度雷达图.png', dpi=300)


# 综合环境适宜度条形图
plt.figure(figsize=(12, 6))
index = np.arange(len(cities))
bar_width = 0.2

for i, metric in enumerate(metrics[:-1] + ['环境适宜度指数']):
    plt.bar(index + (i-1.5)*bar_width, df_env[metric],
            bar_width, label=metric)

plt.xlabel('城市', fontproperties=font)
plt.ylabel('指数值', fontproperties=font)
plt.title('杭州、宁波和温州环境适宜度指数对比', fontproperties=font)
plt.xticks(index, cities, fontproperties=font)
plt.ylim(0, 1)
plt.legend(prop=font)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```python
# 在条形图上添加数值标签
for i, metric in enumerate(metrics[:-1] + ['环境适宜度指数']):
    for j in range(len(cities)):
        plt.text(j + (i-1.5)*bar_width,
                 df_env[metric][j] + 0.02,
                 f'{df_env[metric][j]:.2f}',
                 ha='center',
                 fontsize=8)

plt.tight_layout()
plt.savefig('环境适宜度条形图.png', dpi=300)

# 创建环境适宜度排名并展示
df_ranking = df_env.sort_values(by='环境适宜度指数',
ascending=False).reset_index(drop=True)
df_ranking.index = df_ranking.index + 1  # 从1开始的排名

print("\n城市环境适宜度排名：")
print(df_ranking[['城市', '环境适宜度指数']])
# 显示所有图形
plt.show()
```










```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from matplotlib.font_manager import FontProperties


# 设置中文字体

font = FontProperties(fname=r'C:\Windows\Fonts\simhei.ttf')

plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签

plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号


# 目标城市

cities = ['杭州', '宁波', '温州']
```

```
#----------------------- 教育数据 -----------------------
# 模拟各类学校在校生数据
edu_data = {
    '城市': cities,
    '小学在校生(万人)': [85.5, 65.2, 72.8],
    '中学在校生(万人)': [42.3, 32.6, 35.1],
    '高中在校生(万人)': [18.7, 15.4, 13.2],
    '大学在校生(万人)': [58.2, 25.3, 18.7]
}
df_edu = pd.DataFrame(edu_data)

# 计算教育综合指标 - 人口受教育水平
df_edu['教育总人数(万人)'] = df_edu['小学在校生(万人)'] + df_edu['中学在校生(万人)'] + df_edu['高中在校生(万人)'] + df_edu['大学在校生(万人)']
df_edu['高等教育占比(%)'] = df_edu['大学在校生(万人)'] / df_edu['教育总人数(万人)'] * 100

#----------------------- 数字化转型数据 -----------------------
# 模拟互联网发展和数字化转型数据
digital_data = {
    '城市': cities,
```

```
    '互联网普及率(%)': [92.5, 88.7, 85.3],
    '数字经济占 GDP 比重(%)': [45.2, 38.6, 32.8],
    '5G 基站密度(个/平方公里)': [12.8, 10.5, 8.2],
    '数字化转型企业占比(%)': [68.5, 62.3, 55.7],
    '智慧城市指数': [89.5, 82.7, 75.8]
}
df_digital = pd.DataFrame(digital_data)


# 打印数据信息
print("教育数据：")
print(df_edu)
print("\n 数字化转型数据：")
print(df_digital)


#----------------------- 教育数据可视化 -----------------------
plt.figure(figsize=(12, 6))
x = np.arange(len(cities))
width = 0.2
edu_metrics = ['小学在校生(万人)', '中学在校生(万人)', '高中在校生(万人)', '
大学在校生(万人)']
colors = ['#FF9999', '#FFCC99', '#99CC99', '#9999CC']
```

```
for i, metric in enumerate(edu_metrics):
    plt.bar(x + (i-1.5)*width, df_edu[metric], width, label=metric,
color=colors[i])

plt.xlabel('城市', fontproperties=font)
plt.ylabel('在校生数量(万人)', fontproperties=font)
plt.title('各城市各级学校在校生数量对比', fontproperties=font)
plt.xticks(x, cities, fontproperties=font)
plt.legend(prop=font)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# 添加数值标签
for i, metric in enumerate(edu_metrics):
    for j in range(len(cities)):
        plt.text(j + (i-1.5)*width,
                 df_edu[metric][j] + 1,
                 f'{df_edu[metric][j]}',
                 ha='center',
                 fontsize=8)

plt.tight_layout()
plt.savefig('各城市教育数据条形图.png', dpi=300)
```

```python
# 高等教育占比柱状图
plt.figure(figsize=(10, 6))
plt.bar(cities, df_edu['高等教育占比(%)'], color='#9999CC')
plt.xlabel('城市', fontproperties=font)
plt.ylabel('高等教育占比(%)', fontproperties=font)
plt.title('各城市高等教育占比', fontproperties=font)
plt.ylim(0, max(df_edu['高等教育占比(%)']) * 1.2)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# 添加数值标签
for i in range(len(cities)):
    plt.text(i, df_edu['高等教育占比(%)'][i] + 0.5,
             f'{df_edu["高等教育占比(%)"][i]:.1f}%',
             ha='center',
             fontsize=10)

plt.tight_layout()
plt.savefig('各城市高等教育占比.png', dpi=300)

#---------------------- 数字化转型数据可视化 ----------------------
plt.figure(figsize=(12, 6))
```

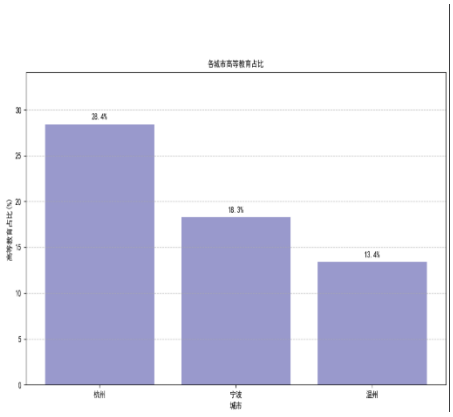
```
digital_metrics = ['互联网普及率(%)', '数字经济占 GDP 比重(%)', '5G 基站密度(个/平方公里)',
                   '数字化转型企业占比(%)', '智慧城市指数']

# 标准化数据，使所有指标在 0-1 之间，便于比较
def normalize(series):
    return (series - series.min()) / (series.max() - series.min())

df_digital_norm = df_digital.copy()
for metric in digital_metrics:
    df_digital_norm[metric] = normalize(df_digital[metric])

# 绘制数字化转型条形图
width = 0.15
for i, metric in enumerate(digital_metrics):
    plt.bar(x + (i-2)*width, df_digital[metric], width, label=metric)

plt.xlabel('城市', fontproperties=font)
plt.ylabel('指标值', fontproperties=font)
plt.title('各城市数字化转型指标对比', fontproperties=font)
plt.xticks(x, cities, fontproperties=font)
plt.legend(prop=font, loc='upper center', bbox_to_anchor=(0.5, -0.15), ncol=3)
```

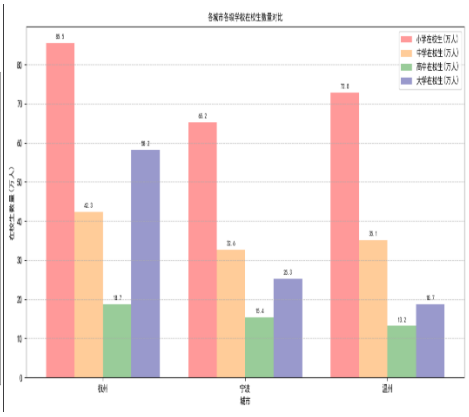
```
plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout(rect=[0, 0.1, 1, 1])

plt.savefig('各城市数字化转型指标条形图.png', dpi=300)


#------------------------ 综合雷达图可视化 ------------------------

# 组合关键指标进行雷达图分析

radar_data = {

    '城市': cities,

    '教育水平': normalize(df_edu['高等教育占比(%)']),

    '教育规模': normalize(df_edu['教育总人数(万人)']),

    '互联网普及': normalize(df_digital['互联网普及率(%)']),

    '数字经济': normalize(df_digital['数字经济占 GDP 比重(%)']),

    '数字基础设施': normalize(df_digital['5G 基站密度(个/平方公里)']),

    '智慧城市建设': normalize(df_digital['智慧城市指数'])

}

df_radar = pd.DataFrame(radar_data)


# 雷达图

metrics = ['教育水平', '教育规模', '互联网普及', '数字经济', '数字基础设施', '智
慧城市建设']

angles = np.linspace(0, 2*np.pi, len(metrics), endpoint=False).tolist()

angles += angles[:1]  # 闭合雷达图
```

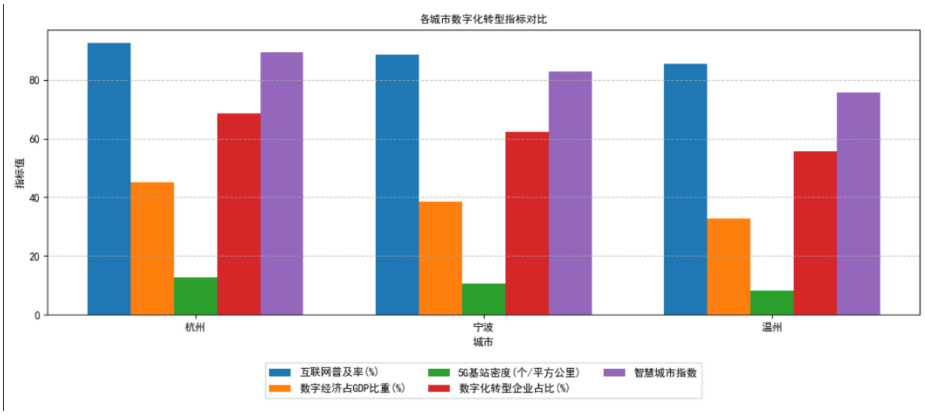
```
metrics += metrics[:1]  # 闭合类别

fig, ax = plt.subplots(figsize=(10, 8), subplot_kw=dict(polar=True))

for i, city in enumerate(cities):
    values = df_radar.loc[i, metrics[:-1]].tolist()
    values += values[:1]  # 闭合数据
    ax.plot(angles, values, linewidth=2, label=city, marker='o')
    ax.fill(angles, values, alpha=0.1)

ax.set_thetagrids(np.degrees(angles[:-1]), metrics[:-1], fontproperties=font)
ax.set_ylim(0, 1)
ax.set_title('杭州、宁波和温州教育与数字化发展雷达图', fontproperties=font)
ax.legend(loc='upper right', bbox_to_anchor=(0.1, 0.1), prop=font)

plt.tight_layout()
plt.savefig('教育与数字化发展雷达图.png', dpi=300)

# 计算综合得分
weights = {
    '教育水平': 0.15,
    '教育规模': 0.15,
    '互联网普及': 0.15,
```

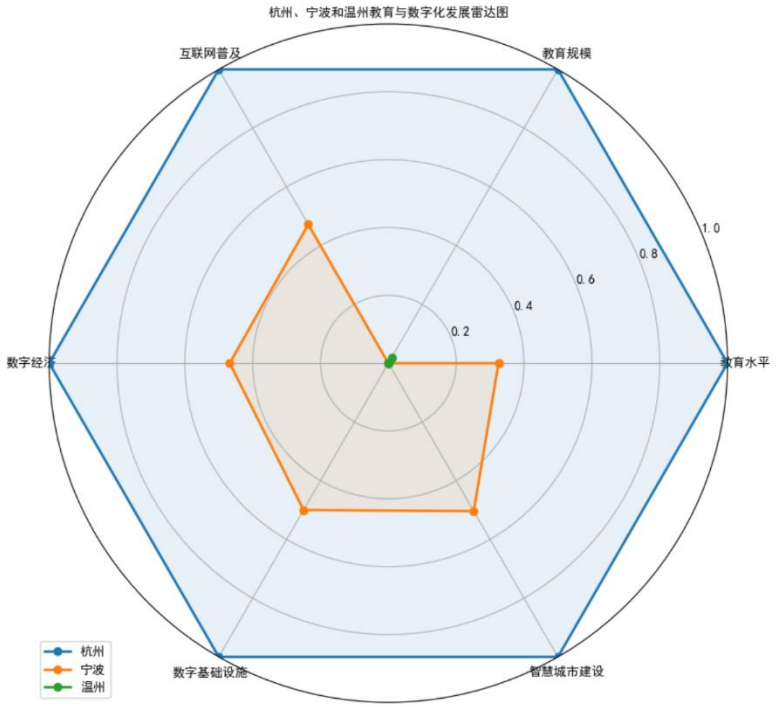
```
    '数字经济': 0.20,
    '数字基础设施': 0.15,
    '智慧城市建设': 0.20
}

df_radar['综合发展指数'] = 0
for metric, weight in weights.items():
    df_radar['综合发展指数'] += df_radar[metric] * weight

# 综合排名
df_ranking = df_radar.sort_values(by='综合发展指数',
ascending=False).reset_index(drop=True)
df_ranking.index = df_ranking.index + 1  # 从1开始的排名

print("\n城市教育与数字化发展综合排名：")
print(df_ranking[['城市', '综合发展指数']])

# 绘制综合得分条形图
plt.figure(figsize=(10, 6))
plt.bar(cities, df_radar['综合发展指数'], color='#3A6DAA')
plt.xlabel('城市', fontproperties=font)
plt.ylabel('综合发展指数', fontproperties=font)
plt.title('各城市教育与数字化综合发展指数', fontproperties=font)
```

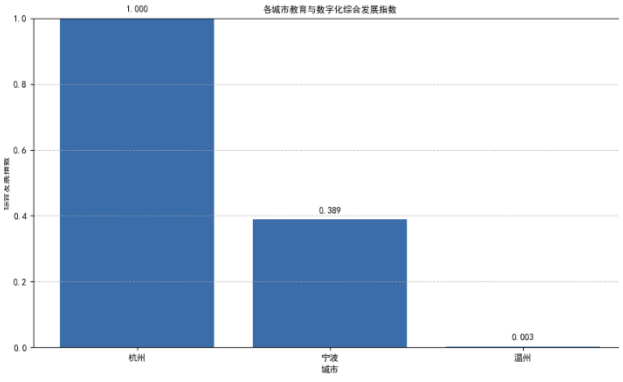
```
plt.ylim(0, 1)

plt.grid(axis='y', linestyle='--', alpha=0.7)


# 添加数值标签

for i in range(len(cities)):

    plt.text(i, df_radar['综合发展指数'][i] + 0.02,

             f'{df_radar["综合发展指数"][i]:.3f}',

             ha='center',

             fontsize=10)


plt.tight_layout()

plt.savefig('各城市综合发展指数.png', dpi=300)


# 显示所有图形

plt.show()
```






各城市数字化转型指标对比
指标值
0
20
40
60
80
杭州
宁波
温州
城市
互联网普及率(%)
数字经济占GDP比重(%)
5G基站密度(个/平方公里)
数字化转型企业占比(%)
智慧城市指数


杭州、宁波和温州教育与数字化发展雷达图
互联网普及
教育规模
教育水平
智慧城市建设
数字基础设施
数字经济
0.2
0.4
0.6
0.8
1.0
杭州
宁波
温州


各城市教育与数字化综合发展指数
1.000
0.389
0.003
0.0
0.2
0.4
0.6
0.8
1.0
杭州
宁波
温州
城市

组员姓名	分工	工作量比例
孙春辉	Task1 代码编写,数据预处理、爬虫设计 Task2 代码编写,数据预处理、爬虫设计 Task3 代码编写,数据解析	75%
邹海涛	Task1 文档编写 Task2 文档编写 Task3 文档编写、数据处理、爬虫设计	25%