Day10-爬虫程序vs数据可视化

今日课程学习目标

```
掌握爬虫爬取图片数据和GDP数据
熟悉pyecharts模块绘制饼图
掌握日志的作用和级别
熟悉logging模块的基本使用
```

今日课程内容大纲

```
# 1. 爬虫程序
    爬取单张图片数据
    爬取多张图片数据
    爬取GDP数据
    多任务爬虫
# 2. 数据可视化
    pyecharts绘制饼图
# 3. 程序日志记录
    日志的作用和级别
    logging模块的使用
    数据埋点简介
```

知识点1：爬虫示例-爬取单张图片数据【掌握】

需求：使用 requests 编写爬虫程序，爬取 **http://127.0.0.1:8080/images/1.jpg** 图片数据并保存。

示例代码：

```
"""
爬虫示例-爬虫单张图片数据
学习目标：能够使用 requests 爬取单张图片数据并保存
"""

import requests

# 准备请求的 URL 地址
url = 'http://127.0.0.1:8080/images/1.jpg'
```

```python

# 发送请求
response = requests.get(url)

# 获取响应图片内容
image_content = response.content

# 将响应图片内容保存成本地图片文件
with open('./spider/1.jpg', 'wb') as f:
    f.write(image_content)
```

知识点2：爬虫示例-爬取多张图片数据【掌握】

需求：访问 http://127.0.0.1:8080/index.html 网址，获取页面上的所有图片保存到本地。

示例代码：

```python
"""
爬虫示例-爬虫多张图片数据
学习目标：能够使用 requests 爬取多张图片数据并保存
"""

import requests
import re

# 思路
# ① 先请求 http://127.0.0.1:8080/index.html，获取响应内容
# ② 从上一步的响应内容中提取所有图片的地址
# ③ 遍历每一个图片地址，向每个图片地址发送请求，并将响应的内容保存成图片文件


def get_images():
    # ① 先请求 http://127.0.0.1:8080/index.html，获取响应内容
    url = 'http://127.0.0.1:8080/index.html'
    # 发送请求
    response = requests.get(url)

    # 获取响应的内容
    html_str = response.content.decode()
    print(html_str)

    # ② 从上一步的响应内容中提取所有图片的地址
    image_url_list = re.findall(r'<img src="(.*?)"', html_str)
    print(image_url_list)

    # ③ 遍历每一个图片地址，向每个图片地址发送请求，并将响应的内容保存成图片文件
    base_url = 'http://127.0.0.1:8080'

    for i, image_url in enumerate(image_url_list):
        # 拼接完整的图片地址
        image_url = base_url + image_url[1:]
        # 发送请求
        image_response = requests.get(image_url)

        # 将响应内容保存成本地图片
        with open(f'./spider/{i}.jpg', 'wb') as f:
            f.write(image_response.content)

```

```
        print('保存图片完毕!!!')


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

知识点3：爬虫示例-爬取GDP数据【掌握】

需求：访问 http://127.0.0.1:8080/gdp.html 网址，提取页面上的国家和GDP数据并保存到本地。

示例代码：

```
"""
爬虫示例-爬取GDP数据
学习目标：能够使用 requests 爬取GDP数据并保存
"""

# 思路
# ① 先请求 http://127.0.0.1:8080/index.html，获取响应内容
# ② 使用正则提取页面上的国家和GDP数据
# ③ 将提取的 GDP 保存到文件 gdp.txt 中

import requests
import re


def get_gdp_data():
    # ① 先请求 http://127.0.0.1:8080/gdp.html，获取响应内容
    url = 'http://127.0.0.1:8080/gdp.html'
    # 发送请求
    response = requests.get(url)

    # 获取响应的内容
    html_str = response.content.decode()
    print(html_str)

    # ② 使用正则提取页面上的国家和GDP数据
    gdp_data = re.findall('<a href=""><font>(.*?)</font></a>.*?<font>￥(.*?)亿元</font>', html_str, flags=re.S)
    print(gdp_data)

    # ③ 将提取的 GDP 保存到文件 gdp.txt 中
    with open('./spider/gdp.txt', 'w', encoding='utf8') as f:
        f.write(str(gdp_data))

    print('保存GDP数据完毕!!!')


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

知识点4：爬虫示例-多任务版爬虫【了解】

真正的工作环境中，我们爬取的数据可能非常的多，如果还是使用单任务实现，这时候就会让我们爬取数据的时间很长，那么显然使用多任务可以大大提升我们爬取数据的效率。

需求：使用多线程实现分别爬取图片数据和 GDP 数据。

示例代码：

```python
"""
爬虫示例-爬虫多任务版
学习目标：能够使用多线程的方式执行多任务爬虫
"""

import requests
import re
import threading


def get_images():
    # ① 先请求 http://127.0.0.1:8080/index.html，获取响应内容
    url = 'http://127.0.0.1:8080/index.html'
    # 发送请求
    response = requests.get(url)

    # 获取响应的内容
    html_str = response.content.decode()
    # print(html_str)

    # ② 从上一步的响应内容中提取所有图片的地址
    image_url_list = re.findall(r'<img src="(.*?)"', html_str)
    # print(image_url_list)

    # ③ 遍历每一个图片地址，向每个图片地址发送请求，并将响应的内容保存成图片文件
    base_url = 'http://127.0.0.1:8080'

    for i, image_url in enumerate(image_url_list):
        # 拼接完整的图片地址
        image_url = base_url + image_url[1:]
        # 发送请求
        image_response = requests.get(image_url)

        # 将响应内容保存成本地图片
        with open(f'./spider/{i}.jpg', 'wb') as f:
            f.write(image_response.content)

    print('保存图片完毕!!!')


def get_gdp_data():
    # ① 先请求 http://127.0.0.1:8080/gdp.html，获取响应内容
    url = 'http://127.0.0.1:8080/gdp.html'
    # 发送请求
    response = requests.get(url)

    # 获取响应的内容
    html_str = response.content.decode()
    # print(html_str)

    # ② 使用正则提取页面上的国家和GDP数据
    gdp_data = re.findall('<a href=""><font>(.*?)</font></a>.*?<font>￥(.*?)亿元</font>', html_str, flags=re.S)
    # print(gdp_data)

    # ③ 将提取的 GDP 保存到文件 gdp.txt 中
    with open('./spider/gdp.txt', 'w', encoding='utf8') as f:
```

```
            f.write(str(gdp_data))

        print('保存GDP数据完毕!!!')


if __name__ == '__main__':
    # 创建两个线程，分别执行爬取图片和爬取GDP数据的任务
    image_thread = threading.Thread(target=get_images)
    gdp_thread = threading.Thread(target=get_gdp_data)

    # 启动线程
    image_thread.start()
    gdp_thread.start()
```

知识点5：pyecharts 模块简介和安装【了解】

问题：

```
1. 什么是 pyecharts？有什么作用？
```

echarts 是个由百度开源的 javascript 语言实现数据可视化框架，凭借着良好的交互性，精巧的图表设计，得到了众多开发者的认可。pyecharts 是 echarts 的 python 版本的实现，在 python 中，可以使用 pyecharts 进行数据可视化操作。

官网地址：https://pyecharts.org/#/

特性：

1. 简洁的API设计，使用如丝滑般流畅，支持链式调用
2. 囊括了30+种常见图表，应有尽有
3. 支持主流Notebook 环境， Jupyter Notebook 和JupyterLab
4. 可轻松集成至Flask, Django等主流Web框架
5. 高度灵活的配置项，可轻松搭配出精美的图表
6. 详细的文档和示例，帮助开发者更快的上手项目
7. 多达400+地图文件以及原生的百度地图，为地理数据可视化提供强有力的支持

安装：

```
pip install pyecharts
```

知识点6：pyecharts 绘制饼图-GDP数据可视化【掌握】

需求：加载爬虫抓取的 GDP 数据，使用 pyecharts 绘制饼状图显示GDP前十的国家。

示例代码：

```
"""
pyecharts-GDP数据可视化
学习目标：能够使用 pyecharts 绘制饼图
"""

# 思路
# ① 从文件中读取 GDP 数据
# ② 使用 pyecharts 绘制饼状图显示GDP前十的国家

# 导入饼图类
from pyecharts.charts import Pie
# 导入配置选项模块
import pyecharts.options as opts
```

```python


def data_view_pie():
    # ① 从文件中读取 GDP 数据
    gdp_data = []

    with open('./spider/gdp.txt', 'r', encoding='utf8') as f:
        content = f.read()
        gdp_data = eval(content)

    # ② 使用 pyecharts 绘制饼状图显示GDP前十的国家
    # 获取 GDP 前 10 的国家数据
    gdp_top10 = gdp_data[:10]

    # 创建饼图
    pie = Pie(init_opts=opts.InitOpts(width="1400px", height="800px"))
    # 给饼图添加数据
    pie.add(
        "GDP",
        gdp_top10,
        label_opts=opts.LabelOpts(formatter='{b}:{d}%')
    )
    # 给饼图设置标题
    pie.set_global_opts(title_opts=opts.TitleOpts(title="2020年世界GDP排名",
subtitle="美元"))
    # 保存结果，默认保存到 render.html 文件
    pie.render()


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

知识点7：程序日志简介及作用【掌握】

问题：

```
1. 什么是程序日志？
2. 日志有什么作用？
3. python中如何记录日志？
```

什么是程序日志?

程序的日志是记录程序在运行过程中，一些关键性的信息，比如：XX用户做了XX操作，服务器出现了XX错误等。

日志有什么作用?

1）可以很方便的了解程序的运行情况

2）方便开发人员检查bug

3）可以分析用户的操作行为、喜好等信息

python中如何记录日志?

python中，可以使用 logging 模块记录日志。

知识点8：程序日志的5个等级说明【掌握】

同其他信息一样，日志信息也是分重要程度的。

日志按重要程度从低到高，分成如下 5 个等级：

1）DEBUG：程序调试bug时使用

2）INFO：程序正常运行时使用

3）WARNING：程序未按预期运行时使用，但并不是错误，如:用户登录密码错误

4）ERROR：程序出错误时使用，如：IO操作失败

5）CRITICAL：特别严重的问题，导致程序不能再继续运行时使用，如：磁盘空间为空，一般很少使用

> 从低到高的：DEBUG < INFO < WARNING < ERROR < CRITICAL

知识点9：logging模块的基本使用【掌握】

logging 模块记录日志时，可以设置日志的等级，默认日志的等级是 WARNING。

> 记录日志信息时，<= 设置日志等级的信息，会直接被忽略(不做记录)。

示例代码：

```
"""
logging模块记录日志
学习目标：能够使用 logging 模块记录日志
"""

import logging

# 设置日志的等级、设置日志的记录格式
# level=logging.DEBUG：将日志等级设置成DEBUG
logging.basicConfig(level=logging.DEBUG,
                    format='%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s',
                    filename='log.txt',
                    filemode='a')

# logging记录日志的默认级别是：WARNING
logging.debug('这是一个debug级别的日志')
logging.info('这是一个info级别的日志')
logging.warning('这是一个warning级别的日志')
logging.error('这是一个error级别的日志')
logging.critical('这是一个critical级别的日志')
```

知识点10：数据埋点介绍【了解】

问题：

```
1. 什么是数据埋点？数据埋点有什么作用？
2. 如何做数据埋点？
```

什么是数据埋点？数据埋点有什么作用？

数据埋点是数据采集的一种方式，是针对特定业务场景进行数据采集和上报的技术方案。简单来说就是根据具体的业务分析需求，将关心的用户行为信息记录都日志中，采集数据以便进行后续的分析操作。

如何做数据埋点?

1）代码埋点：在产品源代码中添加记录日志的代码（需要开发人员配合）

2）可视化埋点：通过第三方产品，配置记录关心的用户行为数据（业务人员自己即可配置）

3）无埋点(全埋点)：通过第三方产品，配置记录用户的所有行为数据（业务人员自己即可配置）