Python与金融数据挖掘(9)

文欣秀

wenxinxiu@ecust.edu.cn

案例分析


baidu.com/s?rtt=1&bsst=1&cl=2&tn=news&ie=utf-8&word=阿里巴巴
oogle 翻译
进入空间
创新实践育人计划
Welcome to Pyth...
PyTorch（二）—...
Python Program...
度
阿里巴巴
百度一下
网页
资讯
贴吧
知道
文库
图片
视频
地图
采购
更多
百度为您找到相关资讯234个
按焦点排序
全部资讯
宣亚国际:公司与阿里巴巴集团旗下公司有互联网广告投放业务等项目...
39分钟前
宣亚国际4月12日在互动平台回答投资者提问时表示,公司与阿里巴巴集团旗下公司有互联网广告投放业务等项目合作。 原标题:宣亚国际:公司与阿里巴巴集团旗下公司有互联网广告投放业务等项...
东方财富网

舆情数据爬虫

```
import requests
import re
import time
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36'}
def baidu(company):
    url = 'http://www.baidu.com/s?tn=news&rtt=1&wd=' + company
    res = requests.get(url, headers=headers, timeout=10).text
    p_href = '<h3 class="news-title_1YtI1 "><a href="(.*?)"'
    href = re.findall(p_href, res, re.S)
    #…(接下页)
```

输出搜索到的标题、日期、来源

```
    #…(接上页)
    p_title = '<h3 class="news-title_1YtI1 ">.*?>(.*?)</a>'
    title = re.findall(p_title, res, re.S)
    p_date = '<span class="c-color-gray2 c-font-normal c-gap-right-\
xsmall" .*?>(.*?)</span>'
    date = re.findall(p_date, res)
    p_source = '<span class="c-color-gray" .*?>(.*?)</span>'
    source = re.findall(p_source, res)
```

搜索结果清洗及输出

```
    for i in range(len(date)):
        title[i] = title[i].strip()
        title[i] = re.sub('<.*?>', '', title[i])
        if ('小时' in date[i]) or ('分钟' in date[i]):
            date[i] = time. strftime("%Y-%m-%d")
        else:
            date[i] = date[i]
        print(str(i + 1) + '.' + title[i] + '(' + date[i] + '-' + source[i] + ')')
        print(href[i])
baidu('阿里巴巴')
```

问题：如何自动生成舆情数据分析报告？

自动生成舆情数据报告

```
fobj = open('E:\\分析报告.txt', 'a', encoding='utf-8')
for i in range(len(date)):
    title[i] = title[i].strip()
    #...清洗标题与时间
    fobj. write(str(i + 1) + '.' + title[i] + '(' + date[i] + '-' + source[i] + ')' + '\n')
    fobj. write(href[i] + '\n')
fobj.close()
```

问题：如何爬取多个公司的数据？

爬取多个公司数据

```
companys = ['阿里巴巴', '万科集团', '腾讯', '京东']
for each in companys:
    baidu(each)
    print(str(each)+'成功！ ')
```

问题： 如何爬取多个公司的多页数据?

爬取多公司多页数据

```
# 爬取多个公司的多页, 可以给函数传入两个参数
def baidu(company, page):
    num = page * 10  # 参数规律是页数*10，页数从0开始
    url = 'http://www.baidu.com/s?tn=news&rtt=4&wd='+company+'&pn='+str(num)
    res = requests. get(url, headers=headers, timeout=10).text
    #…
```

爬取多公司多页数据

```
companys = ['阿里巴巴', '万科集团', '腾讯', '京东']
for each in companys:
    for i in range(10):  # 这里一共爬取了10页
        baidu(each, i)  # i表示第几页，从0开始
        print(each + '第' + str(i+1) + '页爬取成功')
        time. sleep(3)
```

问题： 如何解决爬取网页时程序崩掉？

错误处理

```
try:
    <语句块1>
except <异常类型1>:
    <语句块2>
else:
    <语句块3>
finally:
    <句块4>
```

错误处理案例

```
try:
    num1=int(input("The first number:"))
    num2=int(input("The second number:"))
    num3=num1/num2
    print(num3)
except ZeroDivisionError:
    print("除数为零错误")
```

错误处理案例

```
try:
    filename = input("input file name:")
    fobj = open(filename,"r")
    for line in fobj:
        print(line.strip())
except IOError:
    print("文件不存在")
else:
    fobj.close()
```

错误处理案例

```
try:
    first=int(input("第一个数：  "))
    second=int(input("第二个数：  "))
    print(first+second)
except:
    print("未知错误")
else:
    print("成功运行")
finally:
     print("程序结束")
```

爬取多个公司多页数据(修改)

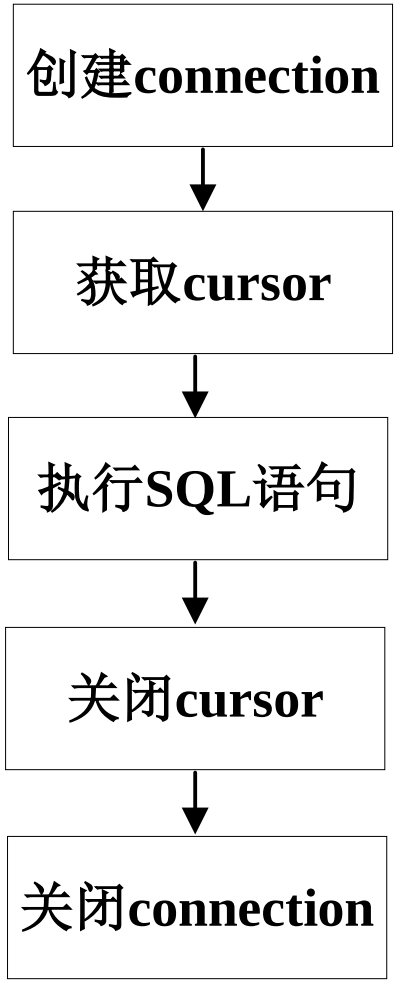
```
companys = ['阿里巴巴', '万科集团', '腾讯', '京东']
for each in companys:
    for i in range(10):
        try:
            baidu(each,i)
            print(each + '第' + str(i+1) + '页爬取成功')
            time. sleep(3)
        except:
            print("{}".format(each+'爬虫失败！'))
```

问题： 如何将爬虫结果存入MySQL数据库中？

常用数据库二

MySQL：是一个关系型数据库管理系统，是最流行的关系型数据库管理系统之一，在WEB 应用方面，MySQL是最好的RDBMS应用软件。

MySQL数据库



访问数据库常用方法

Method	Description
connect()	Open the database connection
close()	Close the database connection
cursor()	Create a cursor object
execute()	Execute an SQL statement
executemany()	Execute a parameterized SQL command
commit()	Commit the current transaction

MySQL创建表例子

```
#!/usr/bin/env python3
import pymysql
# 打开数据库连接
conn = pymysql.connect(host="localhost", user="root",
                       password="123456", database="test")
# 使用 cursor() 方法创建一个游标对象 cursor
cur = conn.cursor()
# 使用 execute() 方法执行 SQL，如果表存在则删除
cur.execute("DROP TABLE IF EXISTS staff")
```

MySQL创建表例子

```
# 使用预处理语句创建表
sql = """CREATE TABLE staff (
                    number  CHAR(10) primary key,
                    name  CHAR(20),  age INT,
                    sex CHAR(1),salary FLOAT )"""
cur.execute(sql)
cur.close()
# 关闭数据库连接
conn.close()
```

MySQL表中插入数据

```
#!/usr/bin/env python3
import pymysql
conn = pymysql.connect(host="localhost", user="root",
password="123456", database="test")
cur = conn.cursor()
sql = """INSERT INTO staff(number,
        name, age, sex, salary)
        VALUES ('1001', '张三', 28, 'M', 7078.5)"""
```

MySQL表中插入数据

```
try:
    cur.execute(sql)    # 执行sql语句
    conn.commit()       # 提交到数据库执行
except:
    conn.rollback()     # 如果发生错误则回滚
cur.close()
conn.close()            # 关闭数据库连接
```

MySQL表中插入多条数据

```
aList=[("1002","Mike",35,"M",10000),("1003","Jack",45,"F",12000)]
sql = """INSERT INTO staff(number, name, age, sex, salary)
        VALUES (%s,%s,%s,%s,%s)"""
try:
   cur.executemany(sql,aList) # 执行sql语句
   conn.commit()      # 提交到数据库执行
except:
   conn.rollback()    # 如果发生错误则回滚
```

MySQL表中取出多条数据

```
#!/usr/bin/env python3
import pymysql
conn = pymysql.connect(host="localhost", user="root",
                                        password="123456", database="test")
cur = conn.cursor()
condition=float(input("请输入工资:"))
sql = """SELECT * FROM staff WHERE salary > %f""" % condition
```

MySQL表中取出多条数据

```
try:
   cur.execute(sql)            # 执行SQL语句
   results = cur.fetchall()   # 获取所有记录列表
   for row in results:
      print ("number=%s,name=%s,age=%d,sex=%s,salary=%f" % \
            (row[0], row[1], row[2], row[3], row[4]))
except:
   print ("Error: unable to fetch data")
cur.close()   # 关闭游标
conn.close()  # 关闭数据库连接
```

MySQL表中更新多条数据

```
#!/usr/bin/env python3
import pymysql
conn = pymysql.connect(host="localhost", user="root",
                       password="123456", database="test")
cur = conn.cursor()
# SQL 更新语句
sql = "UPDATE staff SET age = age + 10 WHERE SEX = '%c'" % 'M'
```

MySQL表中更新多条数据

```
try:
    cur.execute(sql)     # 执行SQL语句
    conn.commit()        # 提交到数据库执行
except:
    conn.rollback()      # 发生错误时回滚
cur.close()
conn.close()             # 关闭数据库连接
```

MySQL表中删除多条数据

```
#!/usr/bin/env python3
import pymysql
conn = pymysql.connect(host="localhost", user="root",
                       password="123456", database="test")
# 使用cursor()方法获取操作游标
cur = conn.cursor()
# SQL 删除语句
sql = "DELETE FROM staff WHERE age > %d" % 30
```

MySQL表中删除多条数据

```
try:
    cur.execute(sql)   # 执行SQL语句
    conn.commit()      # 提交修改
except:
    conn.rollback()    # 发生错误时回滚
cur.close()
conn.close()           # 关闭连接
```

爬取多个公司数据存入MySQL数据库中

MySQL创建表（主程序中）

```
import pymysql
conn = pymysql.connect(host="localhost", user="root",
password="123456", database="test")
cur = conn.cursor()
sql = """CREATE TABLE pachong (company  CHAR(20),
                    title  CHAR(100),  href CHAR(200),
                    source CHAR(20), date CHAR(20))"""
cur.execute(sql)
conn.commit()
conn.close()
```

存入多条数据（子函数中）

```
conn = pymysql.connect(host='localhost', port=3306, user='root',
password='123456', database='test')
cur = conn.cursor()
for i in range(len(date)):
        sql = '''INSERT INTO pachong(company,title,href,source,date)
                                        VALUES (%s,%s,%s,%s,%s) '''
        cur.execute(sql, (company, title[i], href[i], source[i], date[i]))
        conn.commit()
cur.close()
conn.close()
```

常用数据库三

MongoDB： MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。 **https://www.mongodb.com/**

MySQL与MongoDB区别

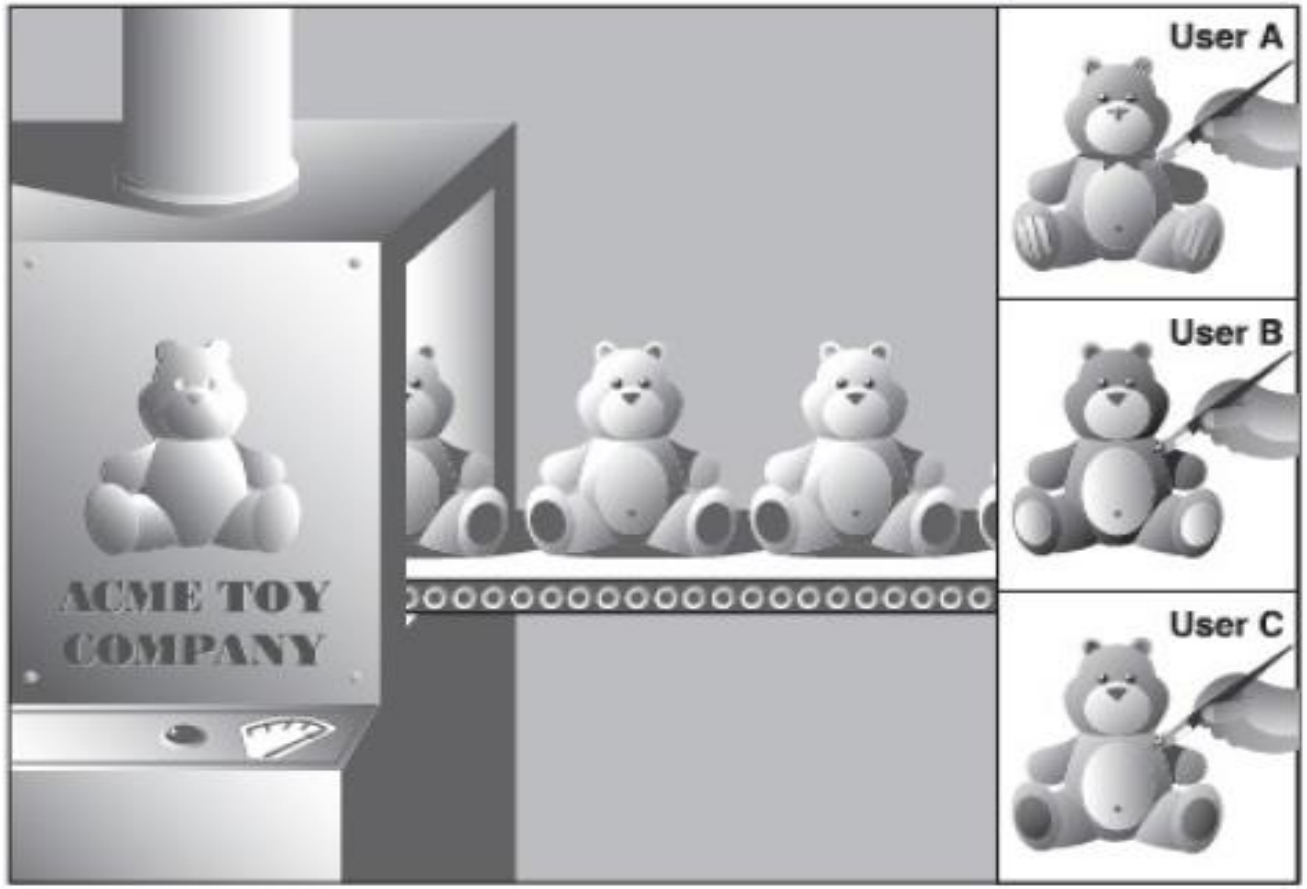
- MySQL是关系型数据库， MongoDB是非关系型数据库；
- MySQL中支持多种引擎，不同引擎有不同存储方式，MongoDB以类JSON的文档的格式存储；
- MySQL使用传统SQL语句进行查询， MongoDB有自己的查询方式(类似JavaScript的函数)。

如何单击按钮调用爬虫程序?

面向对象程序设计

面向对象程序设计将**数据**以及对**数据的操作**放在一起，作为一个相互依存、不可分割的整体进行处理。**对象**（包含**属性**和**方法**）是程序的**基本单元**，每个对象都可以与程序中其它对象进行交互，从而提高软件的重用性、灵活性和扩展性。

类与对象图解



类与对象

类：建立对象的模板，它定义了事物的**属性**和事物可以执行的**行为**；利用类模板所创建的对象称为**类的实例**，类与实例之间是**抽象**与**具体**的关系。

同一类的不同实例之间具有如下特点：

- 相同的**操作**集合
- 相同的**属性**集合
- **不**同的对象**名**

类的定义

class 类名**[(父类)]:**

类的属性

类的方法

```
class nameoftheclass([parent_class]):
    statement1
    statement2
    …
nameofinstance=nameoftheclass([arguments])
```

类应用示例一

```
class Dog:
    def __init__(self):
        # a new instance of class Dog
        self. sound = "wang~wang~~ "
    def bark(self):
        print(self. sound)

if __name__=='__main__':
    bob = Dog()   #define an object of class Dog
    bob. bark()
```

类应用示例二

```
class Animal(object):
    def __init__(self, voice='miao'):
        self. voice=voice

    def say(self):
        print(self. voice)
```

```
if __name__=='__main__':
    kitty=Animal()
    kitty. say()

    bob=Animal('wow')
    bob. say()
```

类应用示例三

```
class animals:
    def breath(self):
        print('breathing')
class dog (animals):
    def eat(self):
        print('eating')
```

```
if __name__=='__main__':
    bob=dog()
    bob. breath()
    bob. eat()
```

类的三种特征

封装性：将基本类结构的细节（如实例变量）隐藏起来，通过**方法接口**实现对实例变量的所有必要访问。

继承性：基于类的特征创建子类，子类可以继承父类的**属性和方法**。

多态性：使用运算符或方法时，根据调用它们的对象类型，执行**不同**的操作过程。

课堂练习

哪个选项用于描述对象的静态特性（　）。

A、方法

B、类型

C、属性

D、消息

谢　谢