PyMySQL 安装

在使用 PyMySQL 之前，我们需要确保 PyMySQL 已安装。

PyMySQL 下载地址：https://github.com/PyMySQL/PyMySQL。 如果还未安装，

我们可以使用以下命令安装最新版的 PyMySQL：

```
$ pip install PyMySQL
```

以下实例链接 Mysql 的 zufang 数据库：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","root","zhaolong","zufang" )

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 使用 execute()  方法执行 SQL 查询
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法获取单条数据.
data = cursor.fetchone()
print ("Database version : %s " % data)

# 关闭数据库连接
db.close()
```

创建数据库表

如果数据库连接存在我们可以使用execute()方法来为数据库创建表，如下所示创建表EMPLOYEE：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用 cursor() 方法创建一个游标对象 cursor
cursor = db.cursor()

# 使用 execute() 方法执行 SQL，如果表存在则删除
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# 使用预处理语句创建表
sql = """CREATE TABLE EMPLOYEE (
         FIRST_NAME  CHAR(20) NOT NULL,
         LAST_NAME  CHAR(20),
         AGE INT,
         SEX CHAR(1),
         INCOME FLOAT )"""

cursor.execute(sql)

# 关闭数据库连接
db.close()
```

数据库插入操作

以下实例使用执行 SQL INSERT 语句向表 EMPLOYEE 插入记录：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 插入语句
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
         LAST_NAME, AGE, SEX, INCOME)
         VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
   # 执行sql语句
   cursor.execute(sql)
   # 提交到数据库执行
   db.commit()
except:
   # 如果发生错误则回滚
   db.rollback()

# 关闭数据库连接
db.close()
```

以上例子也可以写成如下形式：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 插入语句
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
       LAST_NAME, AGE, SEX, INCOME) \
       VALUES ('%s', '%s', '%d', '%c', '%d' )" % \
       ('Mac', 'Mohan', 20, 'M', 2000)
try:
   # 执行sql语句
   cursor.execute(sql)
   # 执行sql语句
   db.commit()
except:
   # 发生错误时回滚
   db.rollback()

# 关闭数据库连接
db.close()
```

以下代码使用变量向SQL语句中传递参数:

```
.................................
user_id = "test123"
password = "password"

con.execute('insert into Login values("%s", "%s")' % \
             (user_id, password))
.................................
```

数据库查询操作

```
Python查询Mysql使用 fetchone() 方法获取单条数据, 使用fetchall() 方法获取多条数据。
fetchone(): 该方法获取下一个查询结果集。结果集是一个对象
fetchmany(size):获取前size行
fetchall(): 接收全部的返回结果行.
rowcount: 这是一个只读属性，并返回执行execute()方法后影响的行数。
```

问题:

```
查询EMPLOYEE表中salary（工资）字段大于1000的所有数据

import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 查询语句
sql = "SELECT * FROM EMPLOYEE \
       WHERE INCOME > '%d'" % (1000)
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 获取所有记录列表
   results = cursor.fetchall()
   for row in results:
      fname = row[0]
      lname = row[1]
      age = row[2]
      sex = row[3]
      income = row[4]
       # 打印结果
      print ("fname=%s,lname=%s,age=%d,sex=%s,income=%d" % \
             (fname, lname, age, sex, income ))
except:
   print ("Error: unable to fetch data")

# 关闭数据库连接
db.close()
```

更新操作

更新操作用于更新数据表的的数据，以下实例将 TESTDB表中的 SEX 字段全部修改为 'M'，AGE 字段递增 1：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 更新语句
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1
                           WHERE SEX = '%c'" % ('M')
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 提交到数据库执行
   db.commit()
except:
   # 发生错误时回滚
   db.rollback()

# 关闭数据库连接
db.close()
```

删除操作

删除操作用于删除数据表中的数据，以下实例演示了删除数据表 EMPLOYEE 中 AGE 大于 20 的所有数据：

```
import pymysql

# 打开数据库连接
db = pymysql.connect("localhost","testuser","test123","TESTDB" )

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# SQL 删除语句
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 提交修改
   db.commit()
except:
   # 发生错误时回滚
   db.rollback()

# 关闭连接
db.close()
```

事务

```
Python DB API 2.0 的事务提供了两个方法 commit 或 rollback。
实例
# SQL删除记录语句
sql = "DELETE FROM EMPLOYEE WHERE AGE > '%d'" % (20)
try:
   # 执行SQL语句
   cursor.execute(sql)
   # 向数据库提交
   db.commit()
except:
   # 发生错误时回滚
   db.rollback()
```

格式化打印

python print格式化输出。

```
1. 打印字符串
print ("His name is %s"%("Aviad"))

2.打印整数
print ("He is %d years old"%(25))

3.打印浮点数
print ("His height is %f m"%(1.83))

4.打印浮点数（指定保留小数点位数）
print ("His height is %.2f m"%(1.83))
```

python爬虫网络爬虫何时有用

假设我有一个鞋店， 并且想要及时了解竞争对手的价格。我可以每天 访问他们的网 站 ， 与 我店铺中鞋子的价格进行对比。但是， 如果我店铺中
的鞋类品种繁多，或是希望能够更加频繁地查看价格变化的话， 就需要花 费 大 量 的 时 间 ， 甚 至 难 以 实 现 。 再 举 一 个 例 子 ， 我 看 中 了 一 双 鞋 ，
想 等它 促销时再购买。我可能需要每天访问这家鞋店的网 站 来查看这双鞋是否降 价， 也许需要等待几个月的时间，我才能如愿盼到这双鞋促销。 上述这两个重复性的手工流程，都可以利用本书介绍的网络爬虫技术实现自动化 处理。

理想状态下， 网络爬虫并不是必须品， 每个网 站 都应该提供 API， 以结构 化的格式共享它们的数据。然而现实情况中， 虽然一些网站已经提供了这种API，但是它们通常会限制可以抓 取的数据，以及访问这些数据的频率。 另外， 对于网站的开发者而言， 维护前端界面比维护后端API接口优先级更高。

总之， 我们不能仅仅依赖于API去访问我们所需的在线数据，而是应该学习一 些网 络爬虫技术的相关知识。

大多数网站 都会定义robots.txt文件， 这样可以让爬虫了解爬取该网站 时存在哪些限制。 这些限制虽然仅仅作为建议给出，但是良好的网 络公民都应 当 遵 守 这 些 限 制 。 在 爬 取 之 前 ， 检 查 r o b o t s . t x t 文 件 这 一 宝 贵 资 源 可 以 最小 化爬虫被封禁的可能，而且还能发现 和网站结构相关的线索。

urllib模块的使用

1.基本方法

urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)

> `url：` 需要打开的网址

> `data：Post`提交的数据

> `timeout：`设置网站的访问超时时间

直接用urllib.request模块的urlopen（）获取页面，page的数据格式为bytes类型，需要decode（）解码，转换成str类型。

```
1 from urllib import request

2 response = request.urlopen(r'http://python.org/') # <http.client.HTTPResponse ob
ject at 0x00000000048BC908> HTTPResponse类型

3 page = response.read()

4 page = page.decode('utf-8')
```

urlopen返回对象提供方法：

> read() , readline() ,readlines() , fileno() , close() ：对HTTPResponse类型数据进行操作

> info()：返回HTTPMessage对象，表示远程服务器返回的头信息

> getcode()：返回Http状态码。如果是http请求，200请求成功完成;404网址未找到

> geturl()：返回请求的url

2.使用Request

urllib.request.Request(url, data=None, headers={}, method=None)

使用request（）来包装请求，再通过urlopen（）获取页面。

复制代码

```
 1 url = r'http://www.lagou.com/zhaopin/Python/?labelWords=label'
 2 headers = {
 3     'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHT
ML, like Gecko) '
 4                   r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',
 5     'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',
 6     'Connection': 'keep-alive'
 7 }
 8 req = request.Request(url, headers=headers)
 9 page = request.urlopen(req).read()
10 page = page.decode('utf-8')
```

复制代码 用来包装头部的数据：

User-Agent ： 这个头部可以携带如下几条信息：浏览器名和版本号、操作系统名和版本号、默认语言

Referer：可以用来防止盗链，有一些网站图片显示来源http://***.com，就是检查Referer来鉴定的

Connection：表示连接状态，记录Session的状态。

3.Post数据

urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)

urlopen（）的data参数默认为None，当data参数不为空的时候，urlopen（）提交方式为Post。

复制代码

```
 1 from urllib import request, parse
 2 url = r'http://www.lagou.com/jobs/positionAjax.json?'
 3 headers = {
 4     'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHT
ML, like Gecko) '
 5                   r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',
 6     'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',
 7     'Connection': 'keep-alive'
 8 }
 9 data = {
10     'first': 'true',
11     'pn': 1,
12     'kd': 'Python'
13 }
14 data = parse.urlencode(data).encode('utf-8')
15 req = request.Request(url, headers=headers, data=data)
16 page = request.urlopen(req).read()
17 page = page.decode('utf-8')
```

复制代码

urllib.parse.urlencode(query, doseq=False, safe='', encoding=None, errors=None)

urlencode（）主要作用就是将url附上要提交的数据。

复制代码

```
1 data = {
2     'first': 'true',
3     'pn': 1,
4     'kd': 'Python'
5 }
6 data = parse.urlencode(data).encode('utf-8')
```

复制代码 经过urlencode（）转换后的data数据为?first=true?pn=1?kd=Python，最后提交的url为

http://www.lagou.com/jobs/positionAjax.json?first=true?pn=1?kd=Python

Post的数据必须是bytes或者iterable of bytes，不能是str，因此需要进行encode（）编码

```
1 page = request.urlopen(req, data=data).read()
```

当然，也可以把data的数据封装在urlopen（）参数中

4.异常处理

复制代码

```
 1 def get_page(url):
 2     headers = {
 3         'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) '
 4                       r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',
 5         'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',
 6         'Connection': 'keep-alive'
 7     }
 8     data = {
 9         'first': 'true',
10         'pn': 1,
11         'kd': 'Python'
12     }
13     data = parse.urlencode(data).encode('utf-8')
14     req = request.Request(url, headers=headers)
15     try:
16         page = request.urlopen(req, data=data).read()
17         page = page.decode('utf-8')
18     except error.HTTPError as e:
19         print(e.code())
20         print(e.read().decode('utf-8'))
21     return page
```