

• 共性问题

- 将内容存储至csv文件时，需要注意python的open方法以及csv库是不保证线程安全的，需要 加锁，或者单独开一个线程把爬到的信息写入(参考自曹思涵同学的作业)，不推荐都抓完才 统一写，中间断掉数据会都存不下来。
- 爬虫在定位特定元素或者请求网络资源的部分可以用 try except 做处理(在保证代 码正确的前提下)，防止因为网络问题或者页面有小的差异(比如个别页面没有某个元素)就 使程序异常
- <https://music.163.com/#/discover/playlist/?order=hot&cat=说唱&limit=35&offset=35>



School of Economics and Management, Beihang University

现代程序设计技术

赵吉昌

jichang@buaa.edu.cn

- 面向对象编程
 - 数据库连接与操作
 - ORM

- 数据库
 - 关系数据库
 - PostgreSQL
 - MySQL
 - 非关系数据库
 - MongoDB
- 所需第三方库
 - 关系数据库
 - psycopg
 - pymysql (建议自学并了解)
 - <https://github.com/PyMySQL/PyMySQL/>
 - 非关系数据库
 - pymongo

- **psycopg**

- 一般的使用逻辑 (Demo: `pys.py`)

- `import psycopg`

- 创建数据库连接 (会话)

- `conn = psycopg.connect("dbname=test
user=postgres password=secret")`

- 创建游标并通过游标执行SQL语句

- `cur = conn.cursor()`

- 执行SQL语句 (创建, 插入, 查询等)

- `cur.execute("CREATE TABLE test (id serial
PRIMARY KEY, num integer, data varchar);")`

- `cur.execute("INSERT INTO test (num, data)
VALUES (%s, %s)", (100, "abc'def'))`

- `cur.execute("SELECT * FROM test;")`

- **psycopg**
 - 通过会话完成事务的提交或回滚
 - `conn.commit()`
 - `conn.rollback()`
 - 关闭数据库会话
 - `cur.close()`
 - `conn.close()`

- **psycopg**
 - 需要通过try-finally来确保连接被关闭
 - `conn = psycopg.connect(DSN)`
 - **try:**
 - `with conn:`
 - `with conn.cursor() as curs:`
 - `curs.execute(SQL1)`
 - `with conn:`
 - `with conn.cursor() as curs:`
 - `curs.execute(SQL2)`
 - **finally:**
 - `conn.close()`

- **psycopg**

- 利用with语句进行连接和游标的管理

- `with psycopg.connect(DSN) as conn:`

- `with conn.cursor() as curs:`

- ...

- **#注意退出with上下文时关闭连接**

- `with psycopg.connect(autocommit=True) as conn:`

- `cur = conn.cursor()`

- `with conn.transaction():`

- `cur.execute("INSERT INTO data VALUES (%s)",`
`("Hello",))`

- `cur.execute("INSERT INTO times VALUES (now())")`

- `# These two operation run atomically in the same`
`transaction`

- `# COMMIT is executed at the end of the block.`

- `# The connection is in idle state again.`

- `# The connection is closed at the end of the block.`

- psycopg

- 给SQL语句传参

- using **%s placeholders** in the SQL statements
 - cur.execute("""
INSERT INTO some_table (an_int, a_date, a_string)
VALUES (%s, %s, %s);
""",
(10, datetime.date(2020, 11, 18), "O'Reilly"))
 - cur.execute("""
– INSERT INTO some_table (an_int, a_date,
another_date, a_string)
– VALUES %(int)s, %(date)s, %(date)s, %(str)s; """,
– **{'int': 10, 'str': "O'Reilly", 'date': datetime.date(2020,
11, 18)}**)

- psycopgpg

- 给SQL语句传参

- Python字符串操作的%不能使用

- cur.execute("INSERT INTO numbers VALUES (%s, %s)" (10, 20)) #错误的写法

- the execute() method accepts a tuple or dictionary of values as second parameter

- cur.execute("INSERT INTO foo VALUES (%s)" , ("bar" ,))

- The placeholder *must not be quoted*

- cur.execute("INSERT INTO numbers VALUES ('%s')", (10,))

- *must always be a %s*

- 表名或者列名等不能直接作为参数传入（动态查询）

- import psycopgpg.sql

- cur.execute(SQL("INSERT INTO {} VALUES (%s)").format(Identifier('numbers')),(10,))

- psycopg
 - 结果的获取
 - **cursor实例本身可迭代**
 - `cur.execute("SELECT * FROM test;")`
 - `for record in cur:`
 - » `print(record)`
 - `fetchone()`
 - 返回tuple
 - `fetchmany([size])`
 - 返回tuple的list, 若无数据即返回[]
 - `fetchall()`
 - 返回tuple的list

- **psycopg2**

- 以字典形式返回执行结果

- with conn.cursor(**row_factory=dict_row**) as dict_cur:
 - dict_cur.execute('SELECT * FROM test;')
 - **res=dict_cur.fetchone()**
 - print(res['id'])
 - print(res['data'])
 - Demo: psy_dict_cur.py

- **psycopg**
 - 以类实例的形式返回结果
 - `from dataclasses import dataclass`
 - `@dataclass`
 - `class Test:`
 - `id: int`
 - `num: int`
 - `data: str`
 - `with conn.cursor(row_factory=class_row(Test))`
`as class_cur:`
 - Demo: `psy_class_cur.py`

- **psycopg**
 - 批量操作
 - 使用 `cursor.executemany()`
 - 但对 `insert`, `update` 等操作效率并不佳
 - 建议使用 `copy`
 - Demo: `psy_batch.py`
 - 高级特征
 - 异步IO
 - Demo: `apsy.py`

- pymongo

- 创建连接

- `from pymongo import MongoClient`
 - `client = MongoClient()`
 - `client = MongoClient('localhost', 27017)`
 - `client = MongoClient('mongodb://localhost:27017/')`

- 指定数据库

- `db = client.test_database`
 - `db = client['test-database']`

- 指定集合(collection)

- `collection = db.test_collection`
 - `collection = db['test-collection']`
 - `Demo: mongo.py`

- pymongo
 - 插入文档
 - `insert_one()`
 - 批量插入
 - `insert_many()`
 - Demo: `mongo_insert_batch.py`

- pymongo
 - returns a single document matching a query (or None if there are no matches)
 - `find_one([query])`
 - 返回结果为字典
 - To get more than a single document as the result of a query
 - `find([query])`
 - returns a Cursor instance that can be iterated
 - `find().limit(size)` #控制返回的数目
 - Demo: `mongo_find.py`

- 计数
 - 返回满足要求的文档数
 - `count_documents({})`
 - Demo: `mongo_count.py`
- 排序
 - 对查询结果进行排序
 - `sort("name" ,1) #ascending`
 - `sort(" name" ,-1) #descending`
 - Demo: `mongo_sort.py`

- pymongo

- 删除文档

- delete_one(myquery)
 - delete_many(myquery)
 - delete_many({})#删除collection中的所有文档
 - drop()#删除整个collection

- 更新





- myquery = { "address": "Valley 345" }
 - newvalues = { "\$set": { "address": "Canyon 123" } }
 - update_one(myquery, newvalues)
 - myquery = { "address": { "\$regex": "^S" } }
 - newvalues = { "\$set": { "name": "Minnie" } }
 - x = mycol.update_many(myquery, newvalues)

- 面向关系数据库的ORM
 - sqlalchemy
 - peewee
 - PonyORM
 - Django ORM
- 一般的逻辑
 - 创建Mapping
 - 业务逻辑中的实体类与数据库的表建立对应关系
 - 构建数据类和会话后进行存储或查询
 - Demo: `sqla.py`
 - [由于psycopg升级, sqlalchemy尚未兼容]

- 面向非关系数据库的ORM
 - Django ORM
 - **MongoEngine**
 - MongoKit
 - Ming

补充：Web开发框架简介



web framework	Bottle	Flask	Flask	Django
ORM	Peewee	Pony ORM	SQLAlchemy	Django ORM
database connector	psycopg	psycopg	psycopg	psycopg
relational database	 PostgreSQL	 PostgreSQL	 PostgreSQL	 PostgreSQL

- 以Bilibili热榜为例，练习协程与非关系数据库的使用
 - 1. 获取某一分区视频的排行榜信息，并从中解析出前 10000 条视频信息。要求使用协程爬取数据。注意协程应用的方式：发送http请求后，可以跳转到数据处理的函数中，而不要跳到发送另一个http请求的函数中，因为由于b站的风控较为严格，短时间大量的请求可能会导致ip被封禁。
 - 2. 将视频信息存入MongoDB数据库，并记录此视频当时的排名和数据创建时间。
 - 3. 爬取同一分区一周之后的热榜，并从中解析出前 10000 条视频信息。对比两次结果并更新数据库，要求第一次结果需从数据库中取出。根据以下规则更新数据库：对于仅在第二次排行榜中的视频，将信息存入数据库；对于仅在第一次排行榜中的视频，将信息从数据库中删除；对于两次排行榜都存在的视频，更新数据库，保留创建时间，增加更新时间字段，并更新排名。
 - 4. (附加) 可否从数据库中查询出两次排行榜中上升和下降排名前10的视频信息？
 - 5. (附加) 由于数据量比较大，数据库处理的时间可能较长，可以考虑使用SMTP协议在数据库处理完成时通过邮件的形式通知你。