

## Kosmoo的博客

记录自己学习python过程中的经验与心得

[目录视图](#)[摘要视图](#)

### 个人资料



Kosmoo

原创

52

粉丝

10

喜欢

9

评论

5



等级：博客 4

访问量：8万+

积分：1457

排名：3万+



### 计算机学校排名



### 文章搜索

### 文章分类

python基础 (12)

python爬虫 (39)

数据库 (7)

环境搭建 (4)

python-数据挖掘 (2)

selenium (8)

数据分析 (1)

数据获取 (1)

### 文章存档

2018年3月 (6)

2018年2月 (4)

2018年1月 (9)

2017年12月 (8)

2017年11月 (11)

展开

### 阅读排行

python爬虫之请求 ( url ) 构... (3157)

selenium+python设置爬虫... (3133)

python使用pymongo访问M... (3049)

python3下为scrapy爬虫添加... (2877)

python版 —— 验证码校验 ... (2848)

## python使用pymongo访问MongoDB的基本操作，以及CSV文件导出

标签：python mongodb

2017年08月29日 23:22:17

3107人阅读

评论(0)

收藏

分类：数据库 (6)

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/zwq912318834/article/details/77689568>

目录(?)

[+]

## python使用pymongo访问MongoDB的基本操作，以及CSV文件导出

### 1. 环境。

Python：3.6.1

Python IDE：pycharm

系统：win7

### 2. 简单示例

```
1 import pymongo
2
3 # mongodb服务的地址和端口号
4 mongo_url = "127.0.0.1:27017"
5
6 # 连接到mongodb，如果参数不填，默认为“localhost:27017”
7 client = pymongo.MongoClient(mongo_url)
8
9 #连接到数据库myDatabase
10 DATABASE = "myDatabase"
11 db = client[DATABASE]
12
13 #连接到集合(表):myDatabase.myCollection
14 COLLECTION = "myCollection"
15 db_coll = db[COLLECTION ]
16
17 # 在表myCollection中寻找date字段等于2017-08-29的记录，并将结果按照age从大到小排序
18 queryArgs = {'date':'2017-08-29'}
19 search_res = db_coll.find(queryArgs).sort('age',-1)
20 for record in search_res:
21     print(f"_id = {record['_id']}, name = {record['name']}, age = {record['age']}")
```

### 3. 要点

- 针对读操作，进行数据统计，尽量使用多线程，节省时间，只是要注意线程数量，会大量吃内存。

### 4. mongoDB的数据类型

- MongoDB支持许多数据类型，如下：

- 字符串** - 用于存储数据的最常用的数据类型。MongoDB中的字符串必须为UTF-8。
- 整型** - 用于存储数值。整数可以是32位或64位，具体取决于服务器。

python实现scrapy爬虫每天...	(2748)
scrapy-redis分布式爬虫的搭...	(2653)
selenium+python实现1688...	(2639)
python用最小二乘法分析数...	(2514)
Pycharm (python集成开发...	(2448)

#### 最新评论

python版 —— 验证码校验 ...  
GAOXIANG626 : [reply]zwq912318834[/reply]打码兔API的使用不太懂，试验多次都失败，希...

python版 —— 验证码校验 ...  
Kosmoo : [reply]GAOXIANG626[/reply]具体是哪一块不清楚，打码兔API的使用，还是关...

python版 —— 验证码校验 ...  
GAOXIANG626 : 看完之后还是没太懂，能否再详细说明一下。对于我刚刚写selenium自动爬取的人来说还是有些难

python实现scrapy爬虫每...  
TianMine : 谢谢分享！

python3下为scrapy爬虫...  
qqzhuimengren : 很棒，赞一个



#### 联系我们



请扫描二维码联系客服  
✉ webmaster@csdn.net  
☎ 400-660-0108  
👤 QQ客服 🗣 客服论坛

关于 招聘 广告服务 百度  
©1999-2018 CSDN版权所有  
京ICP证09002463号

经营性网站备案信息  
网络110报警服务  
中国互联网举报中心  
北京互联网违法和不良信息举报中心

- **布尔类型** - 用于存储布尔值(true / false)值。
- **双精度浮点数** - 用于存储浮点值。
- **最小/最大键** - 用于将值与最小和最大BSON元素进行比较。
- **数组** - 用于将数组或列表或多个值存储到一个键中。
- **时间戳** - timestamp, 当文档被修改或添加时, 可以方便地进行录制。
- **对象** - 用于嵌入式文档。
- **对象** - 用于嵌入式文档。
- **Null** - 用于存储Null值。
- **符号** - 该数据类型与字符串相同; 但是, 通常保留用于使用特定符号类型的语言。
- **日期** - 用于以UNIX时间格式存储当前日期或时间。您可以通过创建日期对象并将日, 月, 年的 - 行指定自己需要的日期时间。
- **对象ID** - 用于存储文档的ID。
- **二进制数据** - 用于存储二进制数据。
- **代码** - 用于将JavaScript代码存储到文档中。
- **正则表达式** - 用于存储正则表达式。
- 不支持的数据类型 :
  - python中的集合 ( set )

## 5. 对表 ( 集合collection ) 的操作

```

1 import pymongo
2
3 # mongodb服务的地址和端口号
4 mongo_url = "127.0.0.1:27017"
5
6 # 连接到mongodb, 如果参数不填, 默认为"localhost:27017"
7 client = pymongo.MongoClient(mongo_url)
8 #连接到数据库myDatabase
9 DATABASE = "amazon"
10 db = client[DATABASE]
11
12 #连接到集合(表):myDatabase.myCollection
13 COLLECTION = "galance20170801"
14 db_coll = db[COLLECTION]
```

### 5.1. 查找记录 : find

- ( 5.1.1 ) 指定返回哪些字段

```

1 # 示例一: 所有字段
2 # select * from galance20170801
3 searchRes = db_coll.find()
4 # 或者searchRes = db_coll.find({})

1 # 示例二: 用字典指定要显示的哪几个字段
2 # select _id,key from galance20170801
3 queryArgs = {}
4 projectionFields = {'_id':True, 'key':True} # 用字典指定
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # 结果{'_id': 'B01EYCLJ04', 'key': 'pro audio'}
```

```

1 # 示例三: 用字典指定去掉哪些字段
2 queryArgs = {}
3 projectionFields = {'_id':False, 'key':False} # 用字典指定
4 searchRes = db_coll.find(queryArgs, projection = projectionFields)
5 # 结果{'activity': False, 'avgStar': 4.3, 'color': 'Yellow & Black', 'date': '2017-08-01'}

1 # 示例四: 用列表指定要显示哪几个字段
2 # select _id,key,date from balance20170801
3 queryArgs = {}
4 projectionFields = ['key','date'] # 用列表指定, 结果中一定会返回_id这个字段
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # 结果{'_id': 'B01EYCLJ04', 'date': '2017-08-01', 'key': 'pro audio'}

```

## • ( 5.1.2 ) 指定查询条件

### • ( 5.1.2.1 ) . 比较 : = , != , > , < , >= , <=

```

1 $ne: 不等于(not equal)
2 $gt: 大于(greater than)
3 $lt: 小于(less than)
4 $lte: 小于等于(less than equal)
5 $gte: 大于等于(greater than equal)

1 # 示例一: 相等
2 # select _id,key,sales,date from balance20170801 where key = 'TV & Video'
3 queryArgs = {'key':'TV & Video'}
4 projectionFields = ['key','sales','date']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # 结果: {'_id': '0750699973', 'date': '2017-08-01', 'key': 'TV & Video', 'sales': 0}

1 # 示例二: 不相等
2 # select _id,key,sales,date from balance20170801 where sales != 0
3 queryArgs = {'sales':{'$ne':0}}
4 projectionFields = ['key','sales','date']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # 结果: {'_id': 'B01M996469', 'date': '2017-08-01', 'key': 'stereos', 'sales': 2}

1 # 示例三: 大于
2 # where sales > 100
3 queryArgs = {'sales':{'$gt':100}}
4 # 结果: {'_id': 'B0100YASRG', 'date': '2017-08-01', 'key': 'Sound Bar', 'sales': 124}

1 # 示例四: 小于
2 # where sales < 100
3 queryArgs = {'sales':{'$lt':100}}
4 # 结果: {'_id': 'B011798DKQ', 'date': '2017-08-01', 'key': 'pro audio', 'sales': 0}

1 # 示例五: 指定范围
2 # where sales > 50 and sales < 100
3 queryArgs = {'sales':{'$gt':50, '$lt':100}}
4 # 结果: {'_id': 'B008D2IHES', 'date': '2017-08-01', 'key': 'Sound Bar', 'sales': 66}

1 # 示例六: 指定范围, 大于等于, 小于等于
2 # where sales >= 50 and sales <= 100
3 queryArgs = {'sales':{'$gte':50, '$lte':100}}
4 # 结果: {'_id': 'B01M6DHW26', 'date': '2017-08-01', 'key': 'radios', 'sales': 50}

```

### • ( 5.1.2.2 ) . and

```

1 # 示例一: 不同字段, 并列条件
2 # where date = '2017-08-01' and sales = 100
3 queryArgs = {'date':'2017-08-01', 'sales':100}
4 # 结果: {'_id': 'B01BW2YYC', 'date': '2017-08-01', 'key': 'Video', 'sales': 100}

```

```

1 # 示例二: 相同字段, 并列条件
2 # where sales >= 50 and sales <= 100
3 # 正确: queryArgs = {'sales':{'$gte':50, '$lte':100}}
4 # 错误: queryArgs = {'sales':{'$gt':50}, 'sales':{'$lt':100}}
5 # 结果: {'_id': 'B01M6DHW26', 'date': '2017-08-01', 'key': 'radios', 'sales': 50}

```

#### • ( 5.1.2.3 ) . or

```

1 # 示例一: 不同字段, 或条件
2 # where date = '2017-08-01' or sales = 100
3 queryArgs = {'$or':[{'date':'2017-08-01'}, {'sales':100}]}
4 # 结果: {'_id': 'B01EYCLJ04', 'date': '2017-08-01', 'key': 'pro audio', 'sales': 0}

```

```

1 # 示例二: 相同字段, 或条件
2 # where sales = 100 or sales = 120
3 queryArgs = {'$or':[{'sales':100}, {'sales':120}]}
4 # 结果:
5 #   {'_id': 'B00X5RV14Y', 'date': '2017-08-01', 'key': 'Chargers', 'sales': 120}
6 #   {'_id': 'B0728GGX6Y', 'date': '2017-08-01', 'key': 'Glasses', 'sales': 100}

```

#### • ( 5.1.2.4 ) . in , not in , all

```

1 # 示例一: in
2 # where sales in (100,120)
3 queryArgs = {'sales':{'$in':[100,120]}}
4 # 结果:
5 #   {'_id': 'B00X5RV14Y', 'date': '2017-08-01', 'key': 'Chargers', 'sales': 120}
6 #   {'_id': 'B0728GGX6Y', 'date': '2017-08-01', 'key': 'Glasses', 'sales': 100}

```

```

1 # 示例二: not in
2 # where sales not in (100,120)
3 queryArgs = {'sales':{'$nin':[100,120]}}
4 # 结果: {'_id': 'B01EYCLJ04', 'date': '2017-08-01', 'key': 'pro audio', 'sales': 0}

```

```

1 # 示例三: 匹配条件内所有值 all
2 # where sales = 100 and sales = 120
3 queryArgs = {'sales':{'$all':[100,120]}} # 必须同时满足
4 # 结果: 无结果

```

```

1 # 示例四: 匹配条件内所有值 all
2 # where sales = 100 and sales = 100
3 queryArgs = {'sales':{'$all':[100,100]}} # 必须同时满足
4 # 结果: {'_id': 'B01BW2YYVC', 'date': '2017-08-01', 'key': 'Video', 'sales': 100}

```

#### • ( 5.1.2.5 ) . 字段是否存在

```

1 # 示例一: 字段不存在
2 # where rank2 is null
3 queryArgs = {'rank2':None}
4 projectionFields = ['key', 'sales', 'date', 'rank2']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # 结果: {'_id': 'B00ACOKQTY', 'date': '2017-08-01', 'key': '3D TVs', 'sales': 0}

```

```

1 # 示例二: 字段存在
2 # where rank2 is not null
3 queryArgs = {'rank2':{'$ne':None}}
4 projectionFields = ['key', 'sales', 'date', 'rank2']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields).limit(100)
6 # 结果: {'_id': 'B014I8SX4Y', 'date': '2017-08-01', 'key': '3D TVs', 'rank2': 4.0, 'sales': 0}

```

#### • ( 5.1.2.6 ) . 正则表达式匹配 : \$regex ( SQL : like )

```

1 # 示例一: 关键字key包含audio子串
2 # where key like "%audio%"
3 queryArgs = {'key':{'$regex':'.*audio.*'}}
4 # 结果: {'_id': 'B01M19FGTZ', 'date': '2017-08-01', 'key': 'pro audio', 'sales': 1}

```

#### • ( 5.1.2.7 ) . 数组中必须包含元素 : \$all

```

1 # 查询记录, linkNameLst是一个数组, 指定linkNameLst字段必须包含 'Electronics, Computers & Off
2 db.getCollection("2018-01-24").find({'linkNameLst': {'$all': ['Electronics, Computers & C
3
4 # 查询记录, linkNameLst是一个数组, 指定linkNameLst字段必须同时包含 'Wearable Technology' 和 '
5 db.getCollection("2018-01-24").find({'linkNameLst': {'$all': ['Wearable Technology', 'Ele

```

#### • ( 5.1.2.8 ) . 按数组大小查询

- 两个思路 :
- 第一个思路 : 使用\$where ( 具有很大的灵活性 , 但是速度会慢一些 )

```

1 # priceLst是一个数组, 目标是查询 len(priceLst) < 3
2 db.getCollection("20180306").find({'$where: "this.priceLst.length < 3"})

```

- 关于\$where , 请参考官方文档 : <http://docs.mongodb.org/manual/reference/operator/query/where/> .
- 第二个思路 : 判断数组中的某个指定索引的元素是否存在 ( 会比较高效 )
- 例如 : 如果要求 len(priceLst) < 3 : 就意味着 num[ 2 ]不存在

```

1 # priceLst是一个数组, 目标是查询 len(priceLst) < 3
2 db.getCollection("20180306").find({'priceLst.2': {'$exists: 0}})

```

- 例如 : 如果要求 len(priceLst) > 3 : 就意味着 num[ 3 ]存在

```

1 # priceLst是一个数组, 目标是查询 len(priceLst) > 3
2 db.getCollection("20180306").find({'priceLst.3': {'$exists: 1}})

```

#### • ( 5.1.3 ) 指定查询条件

- ( 5.1.3.1 ) . 限定数量 : limit

```

1 # 示例一: 按sales降序排列, 取前100
2 # select top 100 _id,key,sales from balance20170801 where key = 'speakers' order by sales
3 queryArgs = {'key':'speakers'}
4 projectionFields = ['key','sales']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 topSearchRes = searchRes.sort('sales',pymongo.DESCENDING).limit(100)

```

#### • ( 5.1.3.2 ) . 排序 : sort

```

1 # 示例二: 按sales降序, rank升序
2 # select _id,key,date,rank from balance20170801 where key = 'speakers' order by sales des
3 queryArgs = {'key':'speakers'}
4 projectionFields = ['key','sales','rank']
5 searchRes = db_coll.find(queryArgs, projection = projectionFields)
6 # sortedSearchRes = searchRes.sort('sales',pymongo.DESCENDING) # 单个字段
7 sortedSearchRes = searchRes.sort([('sales', pymongo.DESCENDING),('rank', pymongo.ASCENDING)
8 # 结果:
9 # {'_id': 'B000289DC6', 'key': 'speakers', 'rank': 3.0, 'sales': 120}
10 # {'_id': 'B001VRJ5D4', 'key': 'speakers', 'rank': 5.0, 'sales': 120}

```

- ( 5.1.3.3 ). 统计 : count

```
1 # 示例三: 统计匹配记录总数
2 # select count(*) from balance20170801 where key = 'speakers'
3 queryArgs = {'key':'speakers'}
4 searchResNum = db_coll.find(queryArgs).count()
5 # 结果:
6 # 106
```

## 5.2. 添加记录

### 5.2.1. 单条插入

```
1 # 示例一: 指定 _id, 如果重复, 会产生异常
2 ID = 'firstRecord'
3 insertDate = '2017-08-28'
4 count = 10
5 insert_record = {'_id':ID, 'endDate': insertDate, 'count': count}
6 insert_res = db_coll.insert_one(insert_record)
7 print(f"insert_id={insert_res.inserted_id}: {insert_record}")
8 # 结果: insert_id=firstRecord: {'_id': 'firstRecord', 'endDate': '2017-08-28', 'count': 10}

1 # 示例二: 不指定 _id, 自动生成
2 insertDate = '2017-10-10'
3 count = 20
4 insert_record = {'endDate': insertDate, 'count': count}
5 insert_res = db_coll.insert_one(insert_record)
6 print(f"insert_id={insert_res.inserted_id}: {insert_record}")
7 # 结果: insert_id=59ad356d51ad3e2314c0d3b2: {'endDate': '2017-10-10', 'count': 20, '_id':
```

### 5.2.2. 批量插入

```
1 # 更高效, 但要注意如果指定_id, 一定不能重复
2 # ordered = True, 遇到错误 break, 并且抛出异常
3 # ordered = False, 遇到错误 continue, 循环结束后抛出异常
4 insertRecords = [{'i':i, 'date':'2017-10-10'} for i in range(10)]
5 insertBulk = db_coll.insert_many(insertRecords, ordered = True)
6 print(f"insert_ids={insertBulk.inserted_ids}")
7 # 结果: insert_ids=[ObjectId('59ad3ba851ad3e1104a4de6d'), ObjectId('59ad3ba851ad3e1104a4d
```

## 5.3. 修改记录

```
1 # 根据筛选条件_id, 更新这条记录。如果找不到符合条件的记录, 就插入这条记录 (upsert = True)
2 updateFilter = {'_id': item['_id']}
3 updateRes = db_coll.update_one(filter = updateFilter,
4                                 update = {'$set': dict(item)},
5                                 upsert = True)
6 print(f"updateRes = matched:{updateRes.matched_count}, modified = {updateRes.modified_cou

1 # 根据筛选条件, 更新部分字段: i是原有字段, isUpdated是新增字段
2 filterArgs = {'date':'2017-10-10'}
3 updateArgs = {'$set':{'isUpdated':True, 'i':100}}
4 updateRes = db_coll.update_many(filter = filterArgs, update = updateArgs)
5 print(f"updateRes: matched_count={updateRes.matched_count}, "
6       f"modified_count={updateRes.modified_count} modified_ids={updateRes.upserted_id}")
7 # 结果: updateRes: matched_count=8, modified_count=8 modified_ids=None
```

balance20171010 0.014 s 9 Docs

_id	endDate	count	i	date	isUpdated
Objectid("59ad356d51ad3e2314c0d3b2")	2017-10-10	20			
Objectid("59ad3ba851ad3e1104a4de6d")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de6e")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de6f")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de70")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de71")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de72")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de75")			100	2017-10-10	true
Objectid("59ad3ba851ad3e1104a4de76")			100	2017-10-10	true

## 5.4. 删除记录

### 5.4.1. 删除一条记录

```

1 # 示例一: 和查询使用的条件一样
2 queryArgs = {'endDate': '2017-08-28'}
3 delRecord = db_coll.delete_one(queryArgs)
4 print(f"delRecord={delRecord.deleted_count}")
5 # 结果: delRecord=1

```

### 5.4.2. 批量删除

```

1 # 示例二: 和查询使用的条件一样
2 queryArgs = {'i': {'$gt': 5, '$lt': 8}}
3 # db_coll.delete_many({}) # 清空数据库
4 delRecord = db_coll.delete_many(queryArgs)
5 print(f"delRecord={delRecord.deleted_count}")
6 # 结果: delRecord=2

```

## 6. 将数据库文档写入csv文件。

### 6.1. 标准代码

#### • 读csv文件

```

1 import csv
2
3 with open("phoneCount.csv", "r") as csvfile:
4     reader = csv.reader(csvfile)
5     # 这里不需要readlines
6     for line in reader:
7         print(f"# line = {line}, typeOfLine = {type(line)}, lenOfLine = {len(line)}")
8 # 输出结果如下:
9 line = ['850', 'rest', '43', 'NN'], typeOfLine = <class 'list'>, lenOfLine = 4
10 line = ['9865', 'min', '1', 'CD'], typeOfLine = <class 'list'>, lenOfLine = 4

```

#### • 写csv文件

```

1 # 导出数据库所有记录的标准模版
2 import pymongo
3 import csv
4
5 # 初始化数据库
6 mongo_url = "127.0.0.1:27017"
7 DATABASE = "databaseName"
8 TABLE = "tableName"
9
10 client = pymongo.MongoClient(mongo_url)
11 db_des = client[DATABASE]
12 db_des_table = db_des[TABLE]
13
14 # 将数据写入到CSV文件中
15 # 如果直接从mongod booster导出, 一旦有部分出现字段缺失, 那么会出现结果错位的问题

```

```

16
17 # newline='' 的作用是防止结果数据中出现空行，专属于python3
18 with open(f"{DATABASE}_{TABLE}.csv", "w", newline='') as csvfileWriter:
19     writer = csv.writer(csvfileWriter)
20     # 先写列名
21     # 写第一行，字段名
22     fieldList = [
23         "_id",
24         "itemType",
25         "field_1",
26         "field_2",
27         "field_3",
28     ]
29     writer.writerow(fieldList)
30
31     allRecordRes = db_des_table.find()
32     # 写入多行数据
33     for record in allRecordRes:
34         print(f"record = {record}")
35         recordValueLst = []
36         for field in fieldList:
37             if field not in record:
38                 recordValueLst.append("None")
39             else:
40                 recordValueLst.append(record[field])
41         try:
42             writer.writerow(recordValueLst)
43         except Exception as e:
44             print(f"write csv exception. e = {e}")

```

## 6.2. 可能出现的问题以及解决方案

### 6.2.1. 写csv文件编码问题

- 参考文章：Python UnicodeEncodeError: 'gbk' codec can't encode character 解决方法：
- <http://www.jb51.net/article/64816.htm>

- **重要点：**目标文件的编码是导致标题所指问题的罪魁祸首。如果我们打开一个文件，在windows文件的默认编码是gbk，这样的话，python解释器会用gbk编码去解析我们的网络数据流txt，然而已经是decode过的unicode编码，这样的话就会导致解析不了，出现上述问题。解决的办法就是目标文件的编码。

- 解决方案：

```

1 ##### 确实最推荐的做法是在open文件时，指定编码格式：
2 with open(f"{DATABASE}_{TABLE}.csv", "w", newline='', encoding='utf-8') as csvfileWriter:
3 # 就像我们在windows环境下，写csv文件时，默认编码是'gbk'，而从网上获取的数据大部分是'utf-8'，这就

```

### 6.2.2. 写csv文件出现空白行（存在一行间一行）

- python2.x 版本

- 描述及解决方案，请参考：<https://www.cnblogs.com/China-YangGISboy/p/7339118.html>

```

1 # 为了解决这个问题，查了下资料，发现这是和打开方式有关，将打开的方法改为wb，就不存在这个问题了，也
2 在read/write csv 文件是要以binary的方式进行。
3 with open('result.csv', 'wb') as cf:
4     writer = csv.writer(cf)
5     writer.writerow(['shader', 'file'])
6     for key, value in result.items():
7         writer.writerow([key, value])

```

```

1 python2.x要用'wb'模式写入的真正原因：
2 python2.x中写入CSV时，CSV文件的创建必须加上'b'参数，即open('result.csv', 'wb')，不然会出现隔行

```



```
3
4 而且在python2.x中，str和bytes是存在很多隐性转换的，所以虽然CSV是文本文件，也是可以正常写入。
```

#### • python3 版本

在python3中，str和bytes有了清晰的划分，也没有任何隐性的转换，csv 是文本格式的文件，不支持二进制写入，所以不要用二进制模式打开文件，数据也不必转成bytes。

- 描述及解决方案，请参考：[https://segmentfault.com/q/1010000006841656?\\_ea=1148776](https://segmentfault.com/q/1010000006841656?_ea=1148776)

```
1 # 解决方案就是 newline 配置成空即可
2 with open('result.csv', 'w', newline='') as csvfile:
```

- 总结一下：出现空白行的根本原因是Python版本问题，解决方案上python2.x中要求用'wb'，python3.0以上用'w'和newline参数。
- 拓展：关于python3中bytes和string之间的互相转换：<http://www.jb51.net/article/105064.htm>

- [上一篇](#) python实现自动监测目标网站的爬取速度\_以及整体网络环境分析
- [下一篇](#) python实现scrapy爬虫每天定时抓取数据

TeamCity - 官网下载试用



TeamCity持续集成解决方案,JetBrains官网下载试用

注册电气工程师收入将大幅上涨



据悉，2018年随着政策变化，注册电气工程师，年薪收入将大幅增加，从平均30万/年的水准

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

## pymongo 介绍和使用示例



xsdxs 2016年09月17日 21:17

背景最近项目中用到了mongodb，并且用python的pymongo包操作。本文就把目前遇到的问题和学习经历做个小结，查询。...

## Pymongo Tutorial & Pymongo入门教



Yelbosh 2015年05月24日 17:21

阅读目录 前置条件通过MongoClient建立一个连接。获取一个数据库获取一个Collection文件 ( Documents ) 文件插入文件获取 find\_one()按照ObjectId查...

## 阿里到底要招什么样的人？

各部门主管&校招师兄齐上阵，海量ppt+视频详解，助你获取offer！

## MongoDb随笔，PyMongo简单使用



callinglove 2015年05月12日 13:53

安装MongoDbMongoDb下载对应的系统版本的可执行文件 本人系统环境:rhel-server-6.2-x86\_64 解压缩包tar zxvf rr-linux-x86\_64-rh...

## PyMongo基本使用



suwei19870312 2013年12月18日 13:18

引用PyMongo >>> import pymongo 创建连接Connection >>> import pymongo >>> conn = pymongo.Conn

## pymongo 学习总结



woshixiaosimao 2017年01月17日 11:26

老规矩，英文文档：http://api.mongodb.com/python/current/examples/authentication.html一、mongodb1、简介  
oDB是一...

## Python 使用pymongo操作mongodb库



mchdba 2016年12月31日 21:55

#!/usr/bin/env python # -\*- coding: utf-8 -\*- import pymongo import datetime def get\_db(): # ...

## 入门美工课程学习

ps美工学习

百度广告



## 解决Anaconda在指定虚拟环境下无法包的问题



a731062834 2017年05月08日 00:10

通过【activate py3】进入后，发现无法用【py3pip install \*\*】来安装包（py3pip是由script目录下的pip.ext更改来的，改了pip-script.py文...

## pymongo的一个注意点



keheinash 2016年12月09日 10:53

python可以通过pymongo对mongodb进行相关操作，但是有一个点需要注意：mongodb存储的数据是bson格式，bson的数据必须是有效的utf8类型。我们在把数据保存到mongod...

## Pymongo Tutorial & Pymongo入门教



wqt925497045 2018年01月16日 20:31

本教程是pymongo和Mongo的一个简单介绍，基于pymongo2.7.2的tutorial。看完后应该对Pymongo对Mongo的基本认识了。教程 这教程是pymongo和...

## PyMongo初级使用教程



hccloud 2017年06月02日 16:19

教程这篇教程主要介绍了MongoDB和PyMongo的初级使用.准备工作在我们开始前，请确保你已经安装了PyMongo的驱动。在Python命令行模式下，执行下面的指令应当不会报出异常：>>> i...

## 成为Google认证深度学习工程师

迈出百万年薪第一步，人工智能必备技能



## 用pip安装pymongo模块报错：Could not find a version that satisfies the req

安装VN.PY时，进行到安装pymongo步骤出现错误Could not find a version that satisfies the requirement pymongo<=1.3.0, >=1.3.0



dyrlovewc 2016年11月12日 20:21

18146

## Python-MongoDB连接搭建（二）：Python连接MongoDB

上篇文章我们介绍了MongoDB数据库的安装及相关配置，今天我们通过Python来操作MongoDB。（1）MongoDB数据库管理 与其他数据库（Oracle，Mysql）等类似，MongoDB...



WenWu\_Both 2017年04月14日 22:41

14345

## 使用python语言操作MongoDB



xsj\_blog 2016年07月26日 09:59

MongoDB是一个跨平台的NoSQL，基于Key-Value形式保存数据。其储存格式非常类似于Python的字典，因此用Python操作MongoDB会非常的容易。pymongo的两种安装命令pip...

## mongodb怎样导出数据为csv或者txt格式



YABIGNSHI 2015年08月14日 15:11

示例：--导出csv格式：mongoexport -h 192.168.6.52 --port 8000 --username root --csv --password mongo123

## Mongodb数据导出到json或csv

使用mongoexport，把Mongodb的数据导出到json或csv



G1Apassz 2015年02月15日 20:44



## python mongodb 设置密码前一篇ok，csv文件存入mongodb

```
# coding:utf-8 import os import csv import pymongo import time ''' mongodb 删除数据库 use test; db.dropDa.
```



xp5xp6 2016年10月24日 00:28

786

## 订单管理系统

什么是订单管理系统

百度广告



## MongoDB 导入 CSV 格式数据详细过程

主要介绍使用自带工具mongoimport工具将 CSV 格式数据导入到 MongoDB 的详细过程。...



u012318074 2017年08月30日 12:12



## MongoDB中数据导入

MongoDB的数据导入MongoDB 是一款NoSql数据库，既然是数据库，那么就需要有数据来提供给我们使用，下面我们数据。导入数据MongoDB官方提供了示例数据库文件，我们就使用官方提供...



wolf\_soul 2015年07月20日 16:18



## python小程序：把名称列表从csv文件读入mongo，再从mongo导入redis ( ana

```
#/usr/bin/env python import redis import csv import pymongo def readDictCSV(fileName...
```



weixin\_37136725 2017年10月01日 15:35

222

## python 导出mongoDB数据中的数据

```
import pymongo,urllib import sys import time import datetime reload(sys) sys.setdefaultencoding('utf...
```



h70614959 2013年11月07日 15:49

