异常处理与爬取数据的保存

1. 异常处理机制的原理和用法
2. 异常的捕获与处理
3. 异常触发
4. 自定义异常

异常捕获

参考 https://docs.python.org/zh-cn/3.6/library/exceptions.html

所有内置的非系统退出的异常都派生自 Exception 类

StopIteration 异常示例：

```
gennumber = ( i for i in range(0,2))
print(next(gennumber))
print(next(gennumber))

try:
    print(next(gennumber))

except StopIteration:
    print('最后一个元素')
```

异常处理机制的原理

- 异常也是一个类
- 异常捕获过程：
  1. 异常类把错误消息打包到一个对象
  2. 然后该对象会自动查找到调用栈
  3. 直到运行系统找到明确声明如何处理这些类异常的位置

- 所有异常继承自 BaseException
- Traceback 显示了出错的位置，显示的顺序和异常信息对象传播的方向是相反的

异常信息与异常捕获

- 异常信息在 Traceback 信息的最后一行，有不同的类型
- 捕获异常可以使用 try...except 语法
- try...except 支持多重异常处理

常见的异常类型主要有：

1. LookupError 下的 IndexError 和 KeyError
2. IOError
3. NameError
4. TypeError
5. AttributeError
6. ZeroDivisionError

抛出异常和自定义异常

使用 raise 语句抛出异常

自定义异常建议从 Exception 继承

新的需求

获取豆瓣电影 TOP250 排名第一的电影短评

反爬虫

1. 验证码识别实战
2. Selenium 模拟登录实战
3. Selenium 访问 JavaScript 加密链接实战

Scrapy 的中间件

下载中间件

爬虫中间件

使用 Redis 实现分布式 Scrapy

下载中间件

如何编写一个下载中间件？一般需要重写下面四个主要方法：

process_request(request, spider)
Request 对象经过下载中间件时会被调用，优先级高的先调用

process_response(request, response, spider)
Response 对象经过下载中间件时会被调用，优先级高的后调用

process_exception(request, exception, spider)
当 process_exception() 和 process_request() 抛出异常时会被调用

from_crawler(cls, crawler)
使用 crawler 来创建中间器对象，并（必须）返回一个中间件对象

下载中间件

更换代理 IP

更换 Cookies

更换 User-Agent

自动重试

分布式爬虫

Scrapy 原生不支持分布式，多机之间需要 Redis 实现队列和管道的共享。

scrapy-redis 很好地实现了 Scrapy 和 Redis 的集成

使用 scrapy-redis 之后 Scrapy 的主要变化：

1. 使用了 RedisSpider 类替代了 Spider 类
2. Scheduler 的 queue 由 Redis 实现
3. item pipeline 由 Redis 实现

安装并启动：pip install scrapy-redis

THANKS!