一、用了哪些Python三方包 （根据你的项目看需要哪些）

http://www.360doc.com/content/18/0529/21/3175779_758069027.shtml

http://www.mamicode.com/info-detail-1922617.html

1、富文本编辑器 tinymce

```
pip install django-tinymce
```

2、全文检索 haystack whoosh

3、中文分词 jieba

4、异步执行耗时程序 celery

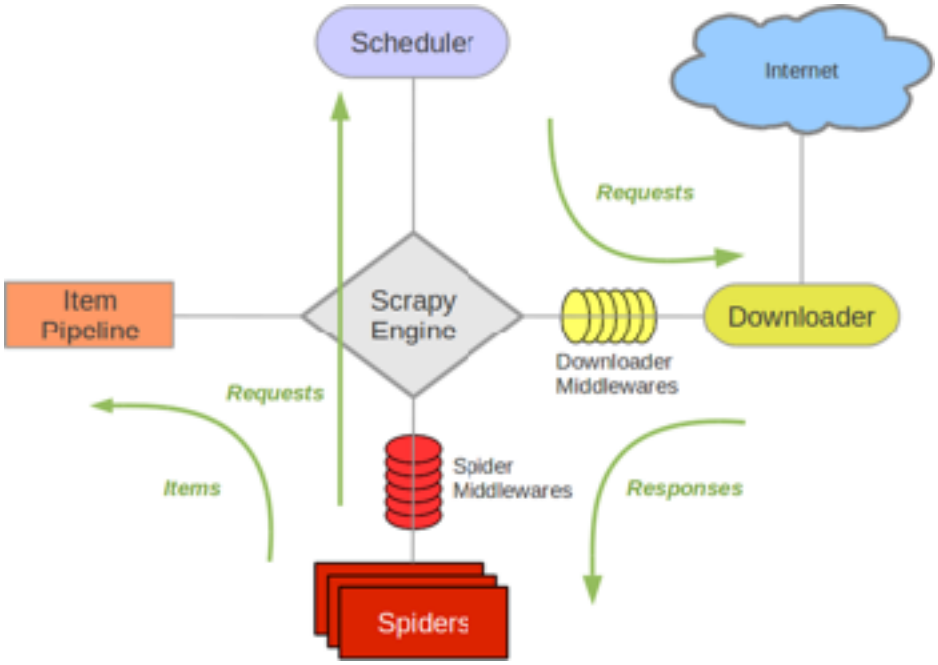
5、BeautifulSoap 网页分析程序

6、pandas, numpy, xlrd, lxml 文件的读写和分析 （包括csv文件、excel文件、xml文件等

7、PIL 图像处理

8、OpenCV 图像识别

二、Scrapy工作流程



Scrapy中的数据流由执行引擎控制，其过程如下:

1. 引擎打开一个网站(open a domain)，找到处理该网站的Spider并向该spider请求第一个要爬取的URL(s)。
2. 引擎从Spider中获取到第一个要爬取的URL并在调度器(Scheduler)以Request调度。
3. 引擎向调度器请求下一个要爬取的URL。
4. 调度器返回下一个要爬取的URL给引擎，引擎将URL通过下载中间件(请求(request)方向)转发给下载器(Downloader)。
5. 一旦页面下载完毕，下载器生成一个该页面的Response，并将其通过下载中间件(返回(response)方向)发送给引擎。
6. 引擎从下载器中接收到Response并通过Spider中间件(输入方向)发送给Spider处理。
7. Spider处理Response并返回爬取到的Item及(跟进的)新的Request给引擎。
8. 引擎将(Spider返回的)爬取到的Item给Item Pipeline，将(Spider返回的)Request给调度器。
9. (从第二步)重复直到调度器中没有更多地request，引擎关闭该网站。

三、 进程、线程、协程理解

1、进程就是一个程序在一个数据集上的一次动态执行过程。进程由程序，数据集，进程控制块三部分组成。程序用来描述进程哪些功能以及如何完成；数据集是程序执行过程中所使用的资源；进程控制块用来保存程序运行的状态。

2、线程

一个进程中可以开多个线程，为什么要有进程，而不做成线程呢？因为一个程序中，线程共享一套数据，如果都做成进程，每个进程独占一块内存，那这套数据就要复制好几份给每个程序，不合理，所以有了线程。

3.进程线程的关系

（1）一个线程只能属于一个进程，而一个进程可以有多个线程，但至少有一个线程

（2）资源分配给进程，进程是程序的主体，同一进程的所有线程共享该进程的所有资源

（3）cpu分配给线程，即真正在cpu上运行的是线程

（4）线程是最小的执行单元，进程是最小的资源管理单元

4、协程

协程，又称微线程，纤程。英文名Coroutine

协程看上去也是子程序，但执行过程中，在子程序内部可中断，然后转而执行别的子程序，在适当的时候再返回来接着执行。

四、django F与Q函数

F() ---- 专门取对象中某列值的操作

```
from django.db.models import F
from core.models import Order

order = Order.objects.get(orderid='123456789')
order.amount = F('amount') - 1
order.save()
```

生成的SQL语句

```
UPDATE `core_order` SET ..., `amount` = `core_order`.`amount` - 1
WHERE `core_order`.`orderid` = '123456789'
```

Q() ---- 对对象的复杂查询，组合查询

```
Order.objects.get(
    Q(desc__startswith='Who'),
    Q(create_time=date(2016, 10, 2)) |
    Q(create_time=date(2016,10,6))
)
```

sql语句

```
SELECT * from core_order WHERE desc LIKE 'Who%' AND (create_time =
'2016-10-02' OR create_time = '2016-10-06')
```

五、Mysql myisam 与 innodb的区别

1、MyISAM：不是事务安全的，而且不支持外键，如果执行大量的select，insert MyISAM比较适合。
2、InnoDB：支持事务安全的引擎，支持外键、行锁、事务是他的最大特点。如果有大量的update和insert，建议使用InnoDB，特别是针对多个并发较高的情况。

六、django用到的装饰器

https://blog.csdn.net/qq_37049050/article/details/80388269

返回操作成功的json数据 @response_success
返回操作失败的json数据 @response_failure
拦截非get请求 @get
拦截非post请求 @post
参数检查 @params

六、常见http响应码

https://blog.csdn.net/GarfieldEr007/article/details/77984065

```
200 请求成功
403 禁止访问
500 内部服务器错误
404 没有发现文件、查询或URL
```

七 %与format的区别

format是python2.6新增的一个格式化字符串的方法，相对于老版的%格式方法，它有很多优点。

1.不需要理会数据类型的问题，在%方法中%s只能替代字符串类型

2.单个参数可以多次输出，参数顺序可以不相同

```
print('hello {0} i am {1} . my name is
{0}'.format('Kevin','Tom'))

print('hello {name1}  i am
{name2}'.format(name1='Kevin',name2='Tom'))
```

八、urllib与urllib2的区别

urllib2可以接受一个**Request**类的实例来设置URL请求的headers，urllib仅可以接受URL。这意味着，你不可以通过urllib模块伪装你的User Agent字符串。

urllib提供**urlencode**方法用来GET查询字符串的产生，而urllib2没有。这是为何urllib常和urllib2一起使用的原因。

> response.text与response.content的区别

response.text返回的类型是str

response.content返回的类型是bytes，可以通过decode()方法将bytes类型转为str类型

推荐使用：response.content.decode()的方式获取相应的html页面

> re/xpath/beautifulsoap需要导入哪些包

```
import re
from lxml import etree
from bs4 import BeautifulSoup
```

> mongodb导包，连接、插入、更新、查找代码

```
from pymongo import MongoClient

conn = MongoClient('192.168.0.113', 27017)
db = conn.mydb  #连接mydb数据库，没有则自动创建
```

```
my_set = db.test_set    #使用test_set集合，没有则自动创建
```

插入数据

```
my_set.insert({"name":"zhangsan","age":18})
#或
my_set.save({"name":"zhangsan","age":18})

#查询全部
for i in my_set.find():
    print(i)
#查询name=zhangsan的
for i in my_set.find({"name":"zhangsan"}):
    print(i)
print(my_set.find_one({"name":"zhangsan"}))

my_set.update({"name":"zhangsan"},{'$set':{"age":20}})

#删除name=lisi的全部记录
my_set.remove({'name': 'zhangsan'})

#删除name=lisi的某个id的记录
id = my_set.find_one({"name":"zhangsan"})["_id"]
my_set.remove(id)

#删除集合里的所有记录
db.users.remove()
```

> POST与PUT的区别

POST相同的数据重复提交每次结果不一样，PUT相同的数据重复提交每次结果都是一样的

> 手写一个装饰器，计算函数运行时间

```
import time

def set_func(func):
    def call_func(*args, **kwargs):
        start_time = time.time()
        func(*args, **kwargs)
        end_time = time.time()
        print('use time:%i' % int(end_time - start_time))
    return call_func
```

```
@set_func
def test():
        time.sleep(3)
```

> Python里面如何拷贝一个对象 deepcopy与copy的区别

1. copy 浅拷贝 只拷贝父对象，不会拷贝对象的内部的子对象。
2. deepcopy 深拷贝 拷贝对象及其子对象

> 阅读以下Python程序

```
for i in range(5, 0, -1):
  print(i)
```

写出打印结果

5
4
3
2
1

> post与get有什么区别

- GET在浏览器回退时是无害的，而POST会再次提交请求。
- GET产生的URL地址可以被Bookmark，而POST不可以。
- GET请求会被浏览器主动cache，而POST不会，除非手动设置。
- GET请求只能进行url编码，而POST支持多种编码方式。
- GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。
- GET请求在URL中传送的参数是有长度限制的，而POST没有。
- 对参数的数据类型，GET只接受ASCII字符，而POST没有限制。
- GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。
- GET参数通过URL传递，POST放在Request body中。
- GET产生一个TCP数据包；POST产生两个TCP数据包。（对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据））

> 给一个字符数组，字符包含a-z, 1-9, 比如abc4b2ac113, 求只出现一次的第一次出现的字符

```
from collections import Counter

def str_find(src_str):
        counter = Counter(src_str)
        list1 = []
```

```
    for (key,value) in counter.items():
        if value < 2:
            list1.append(key)
    print(list1)

str_find('abc4b2ac113')
```

> 什么是lambda
lambda是Python中定义匿名函数的方法

```
add = lambda x, y : x+y
add(1,2)
```

> 定一张班级学生数据表，包含学生姓名(name)、班级(class_id)、年纪(grade_id)三列，请写出sql语句按年纪、班级统计出学生总人数
select count(*) from students group by grade_id, class_id

> TCP与UDP有什么区别?
TCP提供的是面向连接的、可靠的数据流传输。
UDP提供的是非面向连接的、不可靠的数据流传输。
TCP提供可靠的服务，通过TCP连接传送的数据，无差错、不丢失，不重复，按序到达；UDP尽最大努力交付，即不保证可靠交付。
TCP面向字节流；
UDP面向报文。
TCP连接只能是点到点的；
UDP支持一对一、一对多、多对一和多对多的交互通信。
TCP的逻辑通信信道是全双工的可靠信道；
UDP的逻辑通信信道是不可靠信道。

> epoll，select的区别？边缘触发，水平触发的区别

epoll的解决方案不像select或poll一样每次都把current轮流加入fd对应的设备等待队列中，而只在epoll_ctl时把current挂一遍（这一遍必不可少）并为每个fd指定一个回调函数，当设备就绪，唤醒等待队列上的等待者时，就会调用这个回调函数

Linux IO多路复用有水平触发和边缘触发两种方式
只要有数据就绪就会水平触发（select和poll)
有新的数据到来才会边缘触发

epoll既可用于水平触发也可用于边缘触发。

> with语句底层实现

with语句，它仅能工作于支持上下文管理协议（context management protocol）的对象。
也就是说，只有内建了"上下文管理"的对象可以和with一起工作，目前支持该协议的对象有：

- file
- decimal.Context
- thread.LockType
- threading.Lock
- threading.RLock
- threading.Condition
- threading.Semaphore
- threading.BoundedSemaphore

当with语句执行时，便执行上下文表达式（context_expr）来获得一个上下文管理器，上下文管理器的职责是提供一个上下文对象，用于在with语句块中处理细节：
一旦**获得了上下文对象，就会调用它的__enter__()方法**，将完成with语句块执行前的所有准备工作，如果with语句后面跟了as语句，则用__enter__()方法的返回值来赋值；
当with语句块结束时，无论是正常结束，还是由于异常，都会调用上下文对象的__exit__()方法，__exit__()方法有3个参数，如果with语句正常结束，三个参数全部都是 None；如果发生异常，三个参数的值分别等于调用sys.exc_info()函数返回的三个值：类型（异常类）、值（异常实例）和跟踪记录（traceback），相应的跟踪记录对象。
因为上下文管理器主要作用于共享资源，__enter__()和__exit__()方法干的基本是需要分配和释放资源的低层次工作

```
class A:
    def __enter__(self):
        print('__enter__() is called')

    def __exit__(self, e_t, e_v, t_b):
        print('__exit__() is called')


with A() as a:
    print('got instance')
```

> re/xpath/beautifulsoap用法，现场给一个页面html,让你匹配出需要的数据（手写代码）

1. ... ul里面有随机个li标签，用xpath取出最后一个li标签
2. ^*^*$%^^%$^$%...aabbcc...$&$&###**%& 用re取出aabbcc多一个字母少一个字母都不行
3. ^*^*&%^^%$^$%...href='www.baidu.com'...$&$&###**%& 用re取出url

> https和http的区别，requests里面如何取消ssl验证

超文本传输协议HTTP协议被用于在Web浏览器和网站服务器之间传递信息，HTTP协议以明文方式发送内容，不提供任何方式的数据加密

安全套接字层超文本传输协议HTTPS，为了数据传输的安全，HTTPS在HTTP的基础上加入了SSL协议，SSL依靠证书来验证服务器的身份，并为浏览器和服务器之间的通信加密。

> Python3 requests 关闭ssl证书验证

```
html = requests.get(item_url, headers=headers, verify=False)
```

> Mongodb与Mysql的区别

Mysql是关系型数据库，Mongodb是非关系型（nosql)数据库。

Mongodb的适用范围

更高的写入负载
默认情况下，MongoDB更侧重高数据写入性能，而非事务安全，MongoDB很适合业务系统中有大量"低价值"数据的场景。但是应当避免在高事务安全性的系统中使用MongoDB，除非能从架构设计上保证事务安全。
高可用性
MongoDB的复副集(Master-Slave)配置非常简洁方便，此外，MongoDB可以快速响应的处理单节点故障，自动、安全的完成故障转移。这些特性使得MongoDB能在一个相对不稳定（如云主机）的环境中，保持高可用性。
数据量很大或者未来会变得很大
依赖数据库(MySQL)自身的特性，完成数据的扩展是较困难的事，在MySQL中，当一个单达表到5-10GB时会出现明显的性能降级，此时需要通过数据的水平和垂直拆分、库的拆分完成扩展，使用MySQL通常需要借助驱动层或代理层完成这类需求。而MongoDB内建了多种数据分片的特性，可以很好的适应大数据量的需求。
基于位置的数据查询
MongoDB支持二维空间索引，因此可以快速及精确的从指定位置获取数据。

表结构不明确，且数据在不断变大
在一些传统RDBMS中，增加一个字段会锁住整个数据库/表，或者在执行一个重负载的请求时会明显造成其它请求的性能降级。通常发生在数据表大于1G的时候（当大于1TB时更甚）。 因MongoDB是文档型数据库，为非结构货的文档增加一个新字段是很快速的操作，并且不会影响到已有数据。另外一个好处当业务数据发生变化时，是将不在需要由DBA修改表结构。

> redis的适用场景
https://www.cnblogs.com/NiceCui/p/7794659.html

1、缓存——热数据

select 数据库前查询redis，有的话使用redis数据，放弃select 数据库，没有的话，select 数据库，然后将数据插入redis

update或者delete数据库钱，查询redis是否存在该数据，存在的话先删除redis中数据，然后再update或者delete数据库中的数据

2、计数器
3、队列
4、位操作（大数据处理）
5、分布式锁与单线程机制
6、最新列表
7、排行榜

> 代码注释和版本提交信息 （检查下github里面代码规范）
代码注释应该包括类用途、方法注释（方法用途、参数、返回结果）、属性注释、复杂语句注释

版本提交信息里面应该包括 本次提交的原因、任务管理系统里面相关编号。

> 生成器、生成式

1、列表生成式
```
li = [i*i for i in xrange(1,10) if i%2==0]
print li
```

2、列表生成器
最简单的办法就把原来的[]换成()就可以了,不过输出变成了迭代对象
```
li1 = (i*i for i in xrange(1,10) if i%2==0)
for i in li1:
    print i
```

3、函数生成器

```
def f(n):
    sum = 0
    i = 0
    while(i<n):
        sum +=i
        i +=1
        yield(sum)
print (type(f(5)))
for i in f(5):
    print(i)
```

> range与xrange的区别

Python 2 中 range会生成一个列表，xrange返回生成器,占用内存更小。Python3中合并成了range函数

> celery的作用

执行异步任务的框架，将Web应用中的一些耗时的作业转交给工作池，让工作池中的worker以异步的方式执行这些作业

Celery 主要包含以下几个模块：

任务模块 Task

包含异步任务和定时任务。其中，异步任务通常在业务逻辑中被触发并发往任务队列，而定时任务由 Celery Beat 进程周期性地将任务发往任务队列。

消息中间件 Broker

Broker，即为任务调度队列，接收任务生产者发来的消息（即任务），将任务存入队列。Celery 本身不提供队列服务，官方推荐使用 RabbitMQ 和 Redis 等。

任务执行单元 Worker

Worker 是执行任务的处理单元，它实时监控消息队列，获取队列中调度的任务，并执行它。

任务结果存储 Backend

Backend 用于存储任务的执行结果，以供查询。同消息中间件一样，存储也可使用 RabbitMQ, redis 和 MongoDB 等。

> 把列表中的None变成0

```
list1 = [None, 1, 2, 3, None]
```

```
list2 = [0 if item == None else item for item in list1]
```

> inner join 和 left join的区别

left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录
右表只会显示符合搜索条件的记录，右表记录不足的地方均为NULL.

right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录
以右表为基础的,左表不足的地方用NULL填充

inner join(内连接) 只返回两个表中联结字段相等的行

> 在一个列表中如何快速查询到指定元素的位置
list1.index方法

> 爬虫无界面浏览器（无头浏览器）

1、Phantomjs
2、selenium chrome driver

```
from selenium import webdriver
import os

url = 'http://jandan.net/ooxx'
chrome_options = webdriver.ChromeOptions()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
driver = webdriver.Chrome(chrome_options=chrome_options)
driver.get(url)
```

> nginx+uwsgi的理解，及还了解其他的服务器么

nginx 是一个高性能的HTTP和反向代理服务器。

uWSGI：实现了`WSGI`协议（Web Server Gateway Interface），它是nginx web 服务器与应用服务器之间的桥梁。

其他的服务器

Apache http server, nginx, lighttpd, gunicorn

> 反爬有哪些策略，你是如何克服的?

1. 限制IP地址单位时间的访问次数

减少单位时间的访问次数，减低采集效率

2. 屏蔽ip

使用代理ip

3. 用户登录才能访问网站内容

模拟用户提交登录表单

4. header User-Agent 检查用户所用客户端的种类和版本

设置User-Agent

5. Referer 是检查此请求由哪里来，通常可以做图片的盗链判断

自定义Referer字段

6. Cookies

网站可能会检测 Cookie 中 session_id 的使用次数，如果超过限制，就触发反爬策略

定时向目标网站发送不带 Cookies 的请求，提取响应中 Set-cookie 字段信息并保存。爬取网页时，把存储起来的 Cookies 带入 Headers 中

7. 动态加载

网站使用 ajax 动态加载内容

可以先截取 ajax 请求分析一下，有可能根据 ajax 请求构造出相应的 API 请求的 URL 就可以直接获取想要的内容，通常是 json 格式，反而还不用去解析 HTML。

然而，很多时候 ajax 请求都会经过后端鉴权，不能直接构造 URL 获取。这时就可以通过 PhantomJS+Selenium 模拟浏览器行为，抓取经过 js 渲染后的页面

> 集群与分布式的区别
集群是个物理形态，分布式是个工作方式
分布式是指将不同的业务分布在不同的地方。 而集群指的是将几台服务器集中在一起，实现同一业务。

`> Restful`
Rest架构的主要原则
网络上的所有事物都被抽象为资源
每个资源都有一个唯一的资源标识符
同一个资源具有多种表现形式(xml,json等)
对资源的各种操作不会改变资源标识符
所有的操作都是无状态的
符合REST原则的架构方式即可称为RESTful

RESTful用法：
http://127.0.0.1/user/1 GET 根据用户id查询用户数据
http://127.0.0.1/user POST 新增用户
http://127.0.0.1/user PUT 修改用户信息
http://127.0.0.1/user DELETE 删除用户信息

> django如何判断form表单提交格式是否正确?
form.is_valid()

> 鉴权
是指验证用户是否拥有访问系统的权利
常用的鉴权有四种：
HTTP Basic Authentication
session-cookie
Token 验证
OAuth(开放授权)

https://blog.csdn.net/wang839305939/article/details/78713124/

> 用Python编写一个线程安全的单例模式实现

```
import threading
class Singleton(object):
    _instance_lock = threading.Lock()

    def __init__(self):
        pass


    def __new__(cls, *args, **kwargs):
        if not hasattr(Singleton, "_instance"):
            with Singleton._instance_lock:
                if not hasattr(Singleton, "_instance"):
                    Singleton._instance = object.__new__(cls)
        return Singleton._instance

obj1 = Singleton()
obj2 = Singleton()
print(obj1,obj2)
```