1. 深拷贝、浅拷贝？

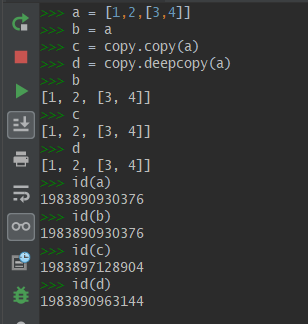
对于不可变对象（数字、字符串、元组），深拷贝和浅拷贝地址值都是一样的。

对于可变对象（列表、字典、集合）：

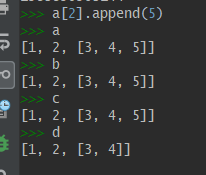
=浅拷贝：值相等、地址值相等

Copy.copy():值相等、地址值不相等

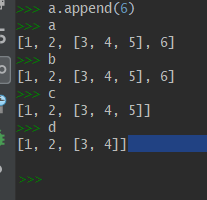
Copy.deepcopy():直相等、地址值不相等



当可变对象内部变化时：浅拷贝后会随着变化、深拷贝不会变化



当可变对象外部变化时：深浅拷贝都不会变化（浅拷贝只是把外部地址拷贝了，内部可变对象没有拷贝；深拷贝不仅拷贝了外部地址，内部地址也拷贝了）。



1. Python新式类和经典类的区别？
2. 在python里凡是继承了object的类都是新式类
3. Python3中只有新式类
4. Python2里继承object的是新式类，没有写父类的是经典类
5. 经典类目前在python中基本没有应用
6. Python中内置的数据结构有几种？

Int long float complex str list tuple dict set

1. 大数据的文件读取：
2. 利用生成器generator
3. 利用迭代器进行迭代遍历: for line in file
4. 单例模式的应用场景？

资源共享情况下，避免由于资源操作时导致的性能或损耗等，如日志文件，应用配置。

资源控制情况下，方便资源置键的互相通信，如线程池等。

网站的计数器、应用配置、多线程池、数据库配置，数据库连接池、应用程序的日志应用。

1. 解释下什么时闭包？

在函数内部在定义一个函数，并且这个函数用到了外边函数的变量，那么将这个函数以及用到的一些变量称为闭包。

7、迭代器和生成器的区别：

生成器能做到迭代器能做的所有的事，而且因为自动创建了\_iter\_\_() 和\_\_next\_\_(),生成器显得特别简洁，生成器也是高效的，使用生成器表达式取代列表解析可以同时节省内存。

除了创建和保存程序最后一次的执行位置和所有的数据值时，还会自动抛出StopIteration异常。

8 、装饰器的作用：

可以在让其他函数不需要做任何代码变动的前提下增加额外的功能。装饰器的返回值也是一个函数对象。

引入日志、函数执行时间统计、执行函数前预备处理、执行函数后的清理功能、权限校验、缓存等。

9、is和== 区别？

Is 比较的是两个对象的id值是否相等，也就是比较两对象是否为同一个实例对象，是否指向同一内存地址。

== 比较的是两个对象的内容是否相等，默认会调用对象的\_\_eq\_\_()方法。

10、作用域？

LEGB原则：L(局部变量)>E(嵌套)>G(全局)>B(内置作用域)

11、Cpython、PyPy、Jython?

Cpython是c语言实现python，python最新的语言特性在这个上面已实现，linux、OS、等自带也是这个版本，基本包含了所有第三方库支持。但Cpython有几个缺陷，1、全局锁使python在多线程效能上表现不佳，2、Cpython无法支持JIT(即时编译)，导致速度不及java和js等语言。

Pypy是python自身实现的解释器。针对cpython的缺点进行了改良，最重要一点是集成了JIT。但Pypy无法支持官方的CPython的api，导致无法使用例如Numpy、scrapy等重要的第三方库。

12、内存泄漏？怎么避免？

内存泄露并非指内存在物理上的消失，而是应用程序分配某段内存后，由于设计错误，失去了对该段内存的控制，因而造成了内存的浪费。导致程序速度减慢，甚至系统崩溃等后果。

当不使用对象时，使用del object来删除一个对象的引用计数就可以有效防止内存泄漏的问题。

13、read、readline、readlines()区别？

read():一次性读取整个文件内容。

Readline() : 每次读取一行内容。

Readlines():一次性读取整个文件内容，并按行返回到列表中。

14、编写函数的四个原则？

函数设计尽量短小，嵌套不宜过深(最好三层)

函数声明应该做到合理，简单，易于使用;

函数参数设计应该考虑向下兼容；

一个函数只能做一件事，尽量保证函数语句粒度的一致性。

15、函数调用参数的传递方式是值传递还是引用传递？

Python不允许程序员选择采用传值还是串引用。Python参数传递采用的是“传引用”的方式。这种方式相当于传值和引用的一种的综合。如果函数收到的是一个可变对象(list，dict)的引用，就能修改对象的原始值-----这相当于“传引用”来传递对象。

如果函数收到的是一个不可变对象（数字、字符串或元组）的引用，就不能直接修改原始对象。-----这相当于“传值”来传递对象。

16、缺省参数？

是指在调用函数的时候没有传入参数的情况下，调用默认的参数；在调用函数的同时赋值，所传入的参数会代替默认参数。

\*args 是不定长参数，表示输入参数不确定，可以是任意多个；

\*\*kwargs 是关键字参数，赋值是以键=值的方式。

17、mysql限制IP访问？

**GRANT** ALL **PRIVILEGES** **ON** \*.\* **TO** 'username'@'172.29.8.72' **IDENTIFIED** **BY** 'password' **WITH** **GRANT** **OPTION**;

18、pass的作用？

1、pass语句什么都不做，作为占位符或占位程序；

2、保持格式完整

3、保持语义完整

19、回调函数？

一个函数作为参数传给另一个函数，第一个函数称为回调函数。

20、什么是lambda函数？ 有什么好处？

Lambda是一个可以接收任意多个参数并且返回单个表达式值的函数。

好处：

Lambda函数比较轻便，即用即扔；

匿名函数，一般用来给filter、map、reduce这样的函数式编服务；

作为回调函数，传递给某些应用，比如消息处理。

21、yield的用法？

Yield就是保存当前程序执行状态。你用for循环是，每次取一个元素就会计算一次。用yield函数叫generator，和iterator一样，他的好处就是不用一次计算所有元素，而是用一次算一次，可以节省很多空间，generator每次计算需要上一次计算结果，所以用yield，否则一return，上次计算结果就没了。

22、match和search的区别？

Match() 从第一字符开始找，如果第一个字符不匹配就返回None,u不继续匹配。用于判断字符串开头或整个字符串是否匹配，速度块。

Search()会整个字符串查找，知道匹配一个。

23、多线程、多进程、协程，项目是否可用到？

进程：一个运行的程序(代码)就是一个进程，，没有运行的代码叫程序，进程是系统资源分配的最小单位，进程拥有自己的独立内存空间，所以进程间数据不共享，开销大。

线程： 调度执行的最小单位，也叫执行路径，不能独立存在，依赖进程，存在一个进程至少有一个线程，叫主线程，而多线程共享内存（数据共享，共享全局变量），从而极大提高了程序的运行效率。

协程：是一种比线程更加轻量级的存在。正如一个进程可以拥有多个线程一样，一个线程也可以拥有多个协程。最重要的是，协程不是被操作系统内核所管理，而完全是由程序所控制。协程拥有自己的寄存器上下文和栈。协程调度时，将寄存器上下文和栈保存到其他地方，在切回来的时候，恢复先前保存的寄存器上下文和栈，直接操作栈基本没有内核切换的开销，可以不加锁的访问全局变量，所以上下文切换非常快。

24、异步使用场景有哪些？

1、不涉及共享资源，即互斥操作。

2、没有时序上的严格关系。

3、不需要原子操作或通过其他方式控制原子性。

4、常用于IO操作等耗时操作，因为比较影响客户体验和使用性能

5、不影响主线程逻辑。

25、什么是多线程竞争？

线程是非独立的，同一个进程里线程是数据共享的，当各个线程访问数据资源时会出现竞争状态即：数据几乎同步会被多个线程占用，造成数据混乱，即所谓的线程不安全。那么怎么解决多线程竞争问题呢？锁； 锁的好处：确保了某段关键代码（共享数据资源）只能由一个线程从头到尾完整的执行，能解决多线程资源竞争下的原子操作问题。锁的坏处：阻止了多线程并发执行，包含锁的某段代码实际上只能以单线程模式执行，效率就大大地下降。

锁的致命问题：死锁。

26、介绍下python的线程同步？

1、setDaemon(False) 当一个线程启动后，会默认产生一个主线程，因为线程是程序执行的最小单位，当设置多线程时，主线程会创建多个子线程，在python中，默认情况下，就是setDaemon(False),主线程执行完自己的任务后，就退出了，此时子线程会继续执行自己的任务，知道自己的任务结束。

2、setDaemon(True) 当我们使用setDaemon(True)时，这时子线程为守护线程，主线程一旦结束，则全部子线程被强制终止。

3、join() 即主线程任务结束后，进入阻塞状态，一直等待所有的子线程结束以后，主线程在终止。同时，当也有设置守护线程时，主线程会等待timeout秒后将会杀死子线程。如果没有设置守护线程，主线程会在timeout秒后自动结束，子线程仍然可以继续执行，直到子线程结束。

27、解释什么是锁？有哪几种锁？

锁是python提供的对线程控制的对象。有互斥锁，可重入锁，死锁。

28、什么是死锁？

若干子线程在系统资源竞争时，都在等待对方对某部分资源解除占用状态，结果是谁也不愿先解锁，互相干等着，程序无法执行下去，就是死锁。 GIL锁（全局解释器锁），只在Cpython里才有，作用：限制多线程同时执行，保证同一时间只有一个线程执行，所以cpython里的多线程其实是伪多线程!! 所以python里常常使用协程技术来代替多线程，协程是一种轻量级的线程。进程和线程的切换由系统决定，而协程由我们程序员自己决定，而gevent下切换是遇到了耗时操作时才会切换；进程里有线程，线程里有进程。

29、多线程交互访问数据，如果访问到了就不访问了？

怎么避免重读？创建一个已访问的数据列表，用于存储已经访问过的数据，并加上互斥锁，在多线程访问数据的时候先查看数据是否在已访问的列表中，如存在就直接跳过。

30、什么是互斥锁？

每个对象都对应于一个可称为“互斥锁”的标记，这个标记用来保证在任意时刻，只能有一个线程访问该对象。同一进程中的多线程之间是共享系统资源的，多个线程同时对一个对象进行操作，一个线程操作尚未结束，另一个线程已经对其进行操作，导致最终结果出现错误，此时需要对被操作对象添加互斥锁，保证每个线程对该对象的操作都得到正确的结果。

31、同步、异步、阻塞、非阻塞？

同步：多个任务之间有先后顺序执行，一个执行完下个才能执行。

异步：多个任务之间没有先后顺序，可以同时执行。

阻塞： 如果卡住了调用者，调用者不能继续往下执行，就是说调用者阻塞了

非阻塞： 如果不会卡住，可以继续执行，就是说非阻塞的。

同步异步相对于多任务而言，阻塞非阻塞相对于代码执行而言。

32、什么是僵尸进程和孤儿进程？怎么避免僵尸进程？

孤儿进程：父进程退出，子进程还在运行的这些子进程都是孤儿进程，孤儿进程将被init进程(进程号为1)所收养，并由init进程对他们完成状态收集工作。僵尸进程：进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait获得子进程的状态信息，那么子进程的进程描述仍然保存在系统中的这些进程是僵尸进程。避免僵尸进程的方法：1、fork两次用孙子进程去完成子进程的任务 2、用wait() 函数使父进程阻塞 3、.使用信号量，在signal handler 中调用waitpid,这样父进程不用阻塞。

33、python 中线程和进程的使用场景？

多进程适合在cpu密集操作（cpu指令比较多，如位多的浮点运算）。

多线程适合在io密集型操作（读写数据操作多的，如爬虫）

34、线程是并发还是并行，进程是并发还是并行？

线程是并发，进程是并行；

进程之间相互独立，是系统分配资源的最小单位，同一个进程中的所有线程共享资源。

35、并行、并发？

并行： 同一时刻多个任务同时在运行

并发: 在同一时间间隔内多个任务都在运行，但是并不会在同一时刻同时运行，存在交替执行的情况。

实现并行的库有： multiprocessing 实现并发的库有: threading 程序需要执行较多的读写、请求和回复任务的需要大量的IO操作，IO密集型操作使用并发更好。 CPU运算量大的程序，使用并行会更好。

36、IO密集型 和cpu密集型？

io密集型：系统运行，大部分状况是cpu在等io的读写。

Cpu密集型：大部分时间在做计算，逻辑判断等。

CPU动作的程序称之cpu密集型。

5、简单谈下GIL：

Global Interpreter Lock(全局解释器锁)

Python 在任意时刻，只有一个线程在解释器内运行。对python虚拟机的访问由全局解释器锁来控制，正是这个锁能保证同一时刻只有一个线程在运行。

在多线程环境中，python虚拟机按以下方式运行：

1. 设置GI
2. 切换到一个线程运行
3. 运行：

指定数量的字节码指令

或者线程主动让出控制

1. 把线程设置为睡眠状态
2. 解锁GIL
3. 再次重复以上步骤。

在调用外部代码（c、c++）的时候，GIL会被锁定，直到这个函数结束为止（由于这段时间没有python的字节码被运行，所以不会做线程切换）。

6、find 和grep：

grep命令是一种强大的文本搜索工具，grep搜索内容串可以是正则表达式，允许对文本文件进行模式查找。如果找到匹配模式，grep打印包含模式的所有行。

find命令通常用来在特定的目录下搜索符合条件的文件，也可以用来搜索特定用户属主的文件。

1. 线上服务可能因为各种原因导致挂掉怎么办？

Linux下的后台进程管理利器supervisor

每次修改文件后在linux执行service supervisord restart

1. 如何提高python运行效率：

使用生成器；关键代码使用外部功能包(cpython,pylnlne,pypy,pyrex);针对循环的优化，--尽量避免在循环中访问变量的属性。

1. 常用linux命令：

ls、help cd more clear mkdir touch pwd rm grep find mv su date

1. yield的用法：

yield简单来说是一个生成器，他记住上次返回时在函数体的位置。对生成器下一次跳转到该函数的指定位置。

1. python内存管理？
2. 垃圾回收：python不想c++、java等语言一样，他们可以不用事先声明变量类型而直接对变量进行赋值。对python来讲，对象的类型和内存都是在运行时确定的。这也是为什么我们程python语言为动态类型语言的原因。（这里我们把动态类型可以简单的归结为对变量内存地址的分配是在运行时自动判断变量类型并对变量进行赋值）
3. 引用计数。Python采用了类似windows内核对象一样的方式来对内存进行管理。每一个对象，都维护一个指向该对象的引用的计数。当变量被绑定在一个对象上时，该变量的引用计数就是1，系统会自动维护这些标签，并定时扫描，当标签的引用计数为0时，该对象就会被回收
4. 内存池机制。python的内存机制以金字塔行，-1，-2层主要由操作系统进行操作。

第0层是C中的malloc，free等内存分配和释放函数进行操作；

第1层和第2层是内存池，有Python的接口函数PyMem\_Malloc函数实现，当对象小于256K时有该层直接分配内存；

第3层是最上层，也就是我们对Python对象的直接操作；

在C 中如果频繁的调用 malloc 与 free 时,是会产生性能问题的.再加上频繁的分配与释放小块的内存会产生内存碎片. Python 在这里主要干的工作有:

如果请求分配的内存在1~256字节之间就使用自己的内存管理系统,否则直接使用 malloc.

这里还是会调用malloc 分配内存,但每次会分配一块大小为256k的大块内存.

经由内存池登记的内存到最后还是会回收到内存池,并不会调用 C 的 free 释放掉.以便下次使用.对于简单的Python对象，例如数值、字符串，元组（tuple不允许被更改)采用的是复制的方式(深拷贝?)，也就是说当将另一个变量B赋值给变量A时，虽然A和B的内存空间仍然相同，但当A的值发生变化时，会重新给A分配空间，A和B的地址变得不再相同

1. 描述数组、链表、队列、堆栈的区别？

数组和链表是数据存储方式的概念，数组在连续的空间中存储数据，而链表可以在非连续的空间中存储数据。

队列和堆栈是描述数据存取方式的概念，队列是先进先出，栈是先进后出；队列和栈可以用数组来实现，也可以用链表来实现。

1. 排序方式？



Django部分：

1. django 中当一个用户登录 A 应用服务器（进入登录状态），然后下次请求被 nginx 代理到 B 应用服务器会出现什么影响？

如果用户在A服务器登陆的session数据没有共享到B服务器，那么之前登录的状态就没有了。

1. 跨域请求问题django怎么解决的（原理）？

启用中间件

Post请求

验证码

表单中添加{%csrf\_token%}标签

3、 请解释或描述一下Django的架构

对于Django框架遵循MVC设计，并且有一个专有名词：MVT

M全拼为Model，与MVC中的M功能相同，负责数据处理，内嵌了ORM框架

V全拼为View，与MVC中的C功能相同，接收HttpRequest，业务处理，返回HttpResponse

T全拼为Template，与MVC中的V功能相同，负责封装构造要返回的html，内嵌了模板引擎。

1. 、django对数据查询结果排序怎么做，降序怎么做，查询大于某个字段怎么做

排序使用order\_by()

降序需要在排序字段名前加-

查询字段大于某个值：使用filter(字段名\_gt=值)

5、.说一下Django，MIDDLEWARES中间件的作用？

中间件是介于request和response处理之间的一道处理过程，相对比较轻量级，并且在全局上改变django的输入和输出。

6、你对Django的认识？

Django是走大而全的方向，它最出名的是其全自动化的管理后台：只需要使用ORM，做简单的对象定义，他就能自动生成数据库结构、以及全功能的管理后台。

Django内置的ORM跟框架内的其他模块耦合度高。

应用程序必须使用Djang内置的ORM，否则就不能享受到框架内提供的种种基于ORM的便利；理论上可以切换掉其ORM模块，这就相当于把装修完毕的房子拆除重新装修，倒不如一开始就去做毛胚房做全新的装修。

Django的卖点是超高的开发效率，其性能扩展有限；采用Django的项目，在流量达到一定规模后，需要进行重构，才能满足性能的需求。

Django适合中小型网站、或者作为大型网站快速实现产品雏形的工具。

7、Django重定向是如何实现的？用的什么状态码？

HttpResponseRedirect

Redirect和reverse

重定向：302、301

8、nginx的正反向代理？

正向代理： 是一个位于客户端和原始服务器(origin server)之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。客户端必须要进行一些特别的设置才能使用正向代理。

反向代理：对于客户端而言他就像原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理的命名空间中的内容发送普通请求，接着反响代理将判断向何处转交请求，并将获得的内容返回给客户端。

9、Django 本身提供了 runserver，为什么不能用来部署？

runserver 方法是调试 Django 时经常用到的运行方式，它使用 Django 自带的；

WSGI server运行，主要在测试和开发中使用，并且runserver开启的方式是单线程。

uWSGI 是一个web服务器，它实现了WSGI协议、uwsgi、http等协议。uwsgi是一种通信协议，而uWSGI是实现uwsgi协议和WSGI协议的web服务器。uWSGI具有超快的性能、低内存占用和多app管理等优点，并配置Nginx。相比较来讲，支持的并发量更高，方便管理多进程，发挥多核的优势，提升性能。

1. 什么是wsgi，uwsgi，uWSGI？

网络和前端：

1. 简述TCP 和UDP的区别以及优缺点？

UDP是面向无连接的通信协议，udp数据包括目的端口和原端口信息。

优点：UDP速度快，操作简单，要求系统资源少，由于通讯不需要连接，可以实现广播发送。

缺点：udp传送数据前并不与对方建立连接，对接收的数据也不发送确认信号，发送端不知道数据是否正确接收，也不重发发送，不可靠。

TCP 是面向连接的通讯协议，通过三次握手建立连接，通讯完成时四次挥手。

优点：tcp在数据传递时，有确认、窗口、重传、阻塞等控制机制，能保证数据正确性，较为可靠。

缺点：tcp相对于udp速度慢一点，要求系统资源较多。

1. 简述浏览器通过WSGI请求动态资源的过程？
2. 发送http请求动态资源给web服务器。
3. Web服务器收到请求后通过WSGI调用一个属性给（应用程序框架）
4. 应用程序框架通过引用WSGI调用web服务器的方法，设置返回的状态和头信息。
5. 调用后返回，此时web服务器保存了刚刚设置的信息。
6. 应用程序查询数据库，生成动态页面的body信息。
7. 把生成的body的信息返回给web服务器。
8. Web服务器把数据返回给浏览器。
9. 描述浏览器访问[www.baidu.com](http://www.baidu.com)过程。
10. 先要解析出baidu.com对应的ip地址。
11. 要先使用arp获取默认网关的mac地址。
12. 组织数据发送给默认网关。（ip还是dns服务器的ip，mac地址是默认网关的mac地址）。
13. 默认网关拥有转发数据的能力，把数据转发给路由器。
14. 路由器根据自己的路由协议，来选择一个合适的较快的路径转发数据给目的网关。
15. 目的网关（dns服务器所在的网关），把数据转发给dns服务器。
16. dns服务解析器解析出baidu.com对应的IP地址，并原路发挥请求这个域名的client。
17. 得到baidu.com对应的ip地址后，会发送tcp的三次握手，进行连接。
18. 使用http协议发送请求数据的web服务器。
19. Web服务器收到请求数据后，通过查询自己的服务器得到响应的结果，原路返回给浏览器。
20. 浏览器收到数据之后通过浏览器自己的渲染功能来显示这个网页。
21. 浏览器关闭tcp连接，即四次挥手结束，完成整个访问过程。

1. tcp的三次握手和四次挥手？

三次握手：1、客户端将报文中的SYN=1,并选择一个seq=x将这个报文发送到服务器，此时客户端进入同步发送状态（SYN-SEND）。

2、服务器收到请求报文后，如同意建立，则回复报文，SYN=1,ACK=1,并选择一个seq=y，并且报文中确认号为x+1，序号为y，此时服务器进入同步已接收状态（SYN-RCVD）。

3、客户端收到服务器的同步确认后，对服务器发送确认的确认。将ACK=1,确认号为y+1，将报文发送出后，客户端进入已连接状态。（ESTABLISHED）。

5、http和https区别？

https协议需要ca申请书，需要收费。

http是超文本传输协议，信息是明文传输，https则是具有安全性的ssl加密传输协议。

http和https使用的是完全不同的连接方式，用的端口有也不一样，前者是80，后者是443.

http的连接很简单，是无状态的；https是由ssl+http协议构建的可进行加密传输、身份认证的网络协议。

https相对http的优缺点：

优点：更安全，在一定程度上保护了服务端。

缺点：技术门槛高，加重了服务端的负担，降低了访问速度。

物理层、数据链路层、网络层、传输层、会话层、表示层、应用层。

使用socket套接字需要传哪些参数？

Address Family 和 Type，分别表示套接字应用场景和类型。

family的值可以是AF\_UNIX(Unix域，用于同一台机器上的进程间通讯)，也可以是AF\_INET（对于IPV4协议的TCP和 UDP），至于type参数，SOCK\_STREAM（流套接字）或者SOCK\_DGRAM（数据报文套接字）,SOCK\_RAW（raw套接字）。

http常见请求头？

1. Host (主机和端口号)

2. Connection (链接类型)

3. Upgrade-Insecure-Requests (升级为 HTTPS 请求)

4. User-Agent (浏览器名称)

5. Accept (传输文件类型)

6. Referer (页面跳转处)

7. Accept-Encoding（文件编解码格式）

8. Cookie （Cookie）

9. x-requested-with :XMLHttpRequest (是 Ajax 异步请求)

6、WSGI、uwsgi、uWSGI的概念？

WSGI是一种web服务器网关接口。他是web服务器与web应用通信的一种规范。

uwsgi是一种线路协议而不是通信协议，常用于uWSGI服务器和其他网络服务器的数据通信。

uWSGI是实现了WSGI和uwsgi两种协议的web服务器。

7、Django请求的生命周期？

客户端发送请求、根据请求的url，在路由系统查询相应的视图函数、视图函数根据调用数据库获取相应数据、视图函数根据模板渲染生成页面、渲染页面反馈回客户端。

8、Django、Flask、Tornado对比？

Django走的是大而全的方向。开发效率高。他的MTV框架，自带的ORM，admin管理后台，自带的sqlite数据库和开发测试用的服务器，给开发者提高了超高的开发效率。

应用场景：主要来搞快速开发，他的亮点是快速开发，节约成本，正常并发量不过10000，如果实现高并发的话，就要自己写socket和http通信，将orm框架干掉，自己编写与数据库交互的框架。因为自带的orm虽然能操作数据库，但他的效率很低，使用外键来联系表与表之间的查询。

Flask是轻量级的框架，自由、灵活、可扩展性强，可以灵活使用第三方扩展，核心基于werkzeug、WSGI工具和jinja2模板引擎。

Tornado走的是少而精的方向，性能优越。他最出名的是异步非阻塞的设计方式。

他的两大核心模块：iostream：对非阻塞的socket进行简单的封装。ioloop：对i/o多路复用的封装，它实现了一个单例。

应用场景：是实时web服务的一个理想框架。

9、ngnix、tomcat、apache、uWSGI有什么区别？

Ngnix常用来做静态内容服务器和代理服务器，外来请求直接转发给后面的应用服务器，tomcat更多用来做一个应用容器，装java web app。严格上讲，apache和ngnix应该叫做HTTP server，而tomcat是一个Application server。

客户端通过http server访问服务器上存储的资源（html文件，图片文件等），http server只是把服务器上的文件通过http协议转发给客户端。

应用服务器往往是运行在http server的背后，执行应用，将动态的内容转化为静态内容后，通过http server分发给客户端。

Apache是同步多进程模型，一个连接对应一个进程，而ngnix是多个连接可对应一个进程。

Ngnix轻量级，抗并发，处理静态文件好。Apache超稳定，对php支持比较简单，ngnix需要配合其他后端用，处理动态请求有优势。

建议使用前端ngnix抗并发，后端apache集群，配合起来效果更好。

uWSGI也是一个web服务器，它实现了WSGI通信协议，uwsgi线路协议、http等协议。Ngnix中HttpUwsgiModule的作用是与uWSGI服务器进行交换。

**nginx 和 uWISG 服务器之间如何配合工作的？**

首先浏览器发起http请求到ngnix服务器，ngnix收到请求包，进行url分析，判断访问的资源类型，如果是静态资源，直接读取静态资源返回给浏览器，如果请求的是动态资源就转发给uWSGI服务器，然后uWSGI服务器根据自身的uwsgi和WSGI协议，找到对应的Django进行逻辑处理，将返回值发送给uWSGI服务器，然后返回给ngnix，最后ngnix将返回值返回给浏览器进行渲染给用户。

10、请给出你熟悉关系数据库范式有那些，有什么作用？

11、系统中日志的作用？

查找系统或软件或项目的错误或异常错误。

侦察系统被莫名人员进入的蛛丝马迹。

12、Django中间件？

中间件工作在试图函数之前，在和试图函数return之后。

1. 做ip限制。放在中间件列表中，组织某些ip访问。
2. url访问过滤。如果访问login视图（放过），如果访问其他视图（登陆的用户放过）。
3. 缓存。客户端请求来了，中间件去缓存看看有没有数据，有的话直接返回给用户，没有再去逻辑层执行视图函数。

13、有过部署经验？用的什么技术？可以满足多少压力？

有部署经验，在阿里云服务器上部署的。

Ngnix+uWSGI的方式来部署Django项目。

压力测试一到两千。

14、Django哪里用到了线程？哪里用到了协程？哪里用到了进程？

Django中耗时的任务用一个进程或线程来执行，比如发邮件，使用celery。

部署Django项目的时候，配置文件中设置了进程和协程的相关配置。

15、csrf攻击原理？

简单来说就是: 你访问了信任网站A，然后A会用保存你的个人信息并返回给你的浏览器一个cookie，然后呢，在cookie的过期时间之内，你去访问了恶意网站B，它给你返回一些恶意请求代码，要求你去访问网站A，而你的浏览器在收到这个恶意请求之后，在你不知情的情况下，会带上保存在本地浏览器的cookie信息去访问网站A，然后网站A误以为是用户本身的操作，导致来自恶意网站C的攻击代码会被执：发邮件，发消息，修改你的密码，购物，转账，偷窥你的个人信息，导致私人信息泄漏和账户财产安全收到威胁。

16、什么是restful api ，谈谈你的理解？（表现层状态转换）

REST是设计风格而不是标准。是客户端和服务器的交互形式。

REST的特点：

1. 具象的。一般指表现层，要表现的对象就是资源。
2. 表现：资源的表现形式。Txt格式、html格式、json格式、jpg格式等。浏览器通过url确定资源的位置，但需要在http请求中，用Accept和Content-Type字段指定，这两个字段是对资源表现的描述。
3. 状态转换：客户端和服务器交互的过程。在这个过程中，一定会有数据和状态的转化，这个转化叫做状态转换。GET获取资源，POST创建资源，DELETE删除资源，PUT更新资源。

RESTful架构：

每个url代表一种资源。

客户端和服务器之间，传递这种资源的某种表现层。

客户端通过4个http动词，对服务器资源进行操作，实现表现层状态转换。

##### 17、对cookie与session的了解？他们能单独用吗

首先需要搞清楚的是session是存储在服务器上的，cookie是存储在客户端浏览器上的两者是相辅相成的用户首次访问服务器，服务器会为每个用户单独创建一个session对象(HttpSession)，并为每个session分配唯一一个id(sessionId)，sessionId通过cookie保存到用户端，当用户再次访问服务器时，需将对应的sessionId携带给服务器，服务器通过这个唯一sessionId就可以找到用户对应的session对象，从而达到管理用户状态

##### 18、Celery分布式任务队列？

情景：用户发起request，并等待response返回。在本些views中，可能需要执行一段耗时的程序，那么用户就会等待很长时间，造成不好的用户体验，比如发送邮件、手机验证码等。 使用celery后，情况就不一样了。解决：将耗时的程序放到celery中执行。将多个耗时的任务添加到队列queue中，也就是用redis实现broker中间人，然后用多个worker去监听队列 里的任务去执行。

任务task：就是一个Python函数。

队列queue：将需要执行的任务加入到队列中。

工人worker：在一个新进程中，负责执行队列中的任务。

代理人broker：负责调度，在布置环境中使用redis。

正向代理：请求经过代理服务器从局域网发出，然后到达互联网上的服务器。

特点：服务端并不知道真正的客户端是谁。

反向代理：请求从互联网发出，先进入代理服务器，再转发给局域网内的服务器。

特点：客户端并不知道真正的服务端是谁。

区别：正向代理的对象是客户端。反向代理的对象是服务端。

##### 19、简述Django下的（内建的）缓存机制?

开发调试

内存

文件

数据库

Memcache缓存(python-memcached模块)

Memcache缓存(pylibmc模块)

除此之外还可使用redis缓存

由于Django是动态网站，所有每次请求均会去数据进行相应的操作，当程序访问量大时，耗时必然会更加明显，

最简单解决方式是使用：缓存，缓存将一个某个views的返回值保存至内存或者memcache中，5分钟内(默认配置)再有人来访问时，

则不再去执行view中的操作，而是直接从内存或者Redis中之前缓存的内容拿到，并返回

1.AJAX是什么，如何使用AJAX？

ajax(异步的javascript 和xml) 能够刷新局部网页数据而不是重新加载整个网页。

第一步，创建xmlhttprequest对象，var xmlhttp =new XMLHttpRequest（);XMLHttpRequest对象用来和服务器交换数据。

第二步，使用xmlhttprequest对象的open（）和send（）方法发送资源请求给服务器。

第三步，使用xmlhttprequest对象的responseText或responseXML属性获得服务器的响应。

第四步，onreadystatechange函数，当发送请求到服务器，我们想要服务器响应执行一些功能就需要使用onreadystatechange函数，每次xmlhttprequest对象的readyState发生改变都会触发onreadystatechange函数。

2. 常见的HTTP状态码有哪些？

200 OK

301 Moved Permanently

302 Found

304 Not Modified

307 Temporary Redirect

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

410 Gone

500 Internal Server Error

501 Not Implemented

3. Post和get区别？

GET请求，请求的数据会附加在URL之后，以?分割URL和传输数据，多个参数用&连接。URL的编码格式采用的是ASCII编码，而不是uniclde，即是说所有的非ASCII字符都要编码之后再传输。

POST请求：POST请求会把请求的数据放置在HTTP请求包的包体中。上面的item=bandsaw就是实际的传输数据。

因此，GET请求的数据会暴露在地址栏中，而POST请求则不会。

4.cookie 和session 的区别？

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用服务器的性能考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、建议：  
   将登陆信息等重要信息存放为SESSION  
   其他信息如果需要保留，可以放在COOKIE中

5、创建一个简单tcp服务器需要的流程

1.socket创建一个套接字

2.bind绑定ip和port

3.listen使套接字变为可以被动链接

4.accept等待客户端的链接

5.recv/send接收发送数据

爬虫：

1. scrapy和scrapy-redis有什么区别？为什么选择redis数据库？

1、Scrapy是一个python爬虫框架，爬取效率极高，具有高度定制性，但不支持分布式。而scrapy-redis是一套基于redis，运行在scrapy框架之上的组件，可以让scrapy支持分布式策略，slaver端共享master端redis里的item队列、请求队列和请求指纹集合。

2、选择redis数据库，因为redis支持主从同步，而且数据都是缓存在内存的，所以基于redis的分布式爬虫，对请求和数据的高频读取效率非常高的。

2、你用过的爬虫框架或者模块有哪些？谈谈他们的区别或者优缺点？

Python自带：urlib，urllib2

第三方：requests

框架：scrapy

urlib和urlib2模块都做与请求url相关的操作，但他们提供不同的功能。

urlib2:urlib2.urlopen可以接收一个request对象或url（在接收request对象时，并以此来设置headers），urlib.urlopen只接收一个url。

urlib有urlencode,urlib2没有，因此只是urlib、urlib2一起使用。

Scrapy是封装起来的框架，他包含了下载器、解析器，日志及异常处理。基于多线程，twisted的方式处理，对于固定单个网站的爬取开发，有优势，但对于多网站，100个网站，并发及分布式处理方面，不够灵活，不便调整与扩展。

Requests是一个http库，他只是用来，进行请求，对于http请求，他是一个强大的库，下载解析全部自己处理，灵活性高，高并发与分布式部署也灵活。

Scrapy优缺点：

优点：scrapy是异步的，采取可读性更强的xpath代替正则

强大的统计和log系统

同时在不同的url爬行

支持shell方式，方便独立调试

写middleware，方便写一些统一的过滤器。

通过管道的方式写入数据库。

缺点：基于python的爬虫框架，扩展性比较差。

基于twisted框架，运行中的exception是不会干掉reactor，并且异步框架出错后是不会停掉其他任务的，数据出错后难以察觉。

1. 你常用的mysql引擎有哪些？各引擎间有什么区别？
2. InnoDB 支持事务，MyISAM 不支持。
3. MyISAM 适合查询以及插入为主的应用，InnoDB 适合频繁修改以及涉及到安全性较高的应用；
4. InnoDB 支持外键，MyISAM 不支持；
5. MyISAM 是默认引擎，InnoDB 需要指定。
6. InnoDB 中不保存表的行数，如 select count(\*) from table 时，InnoDB；需要扫描一遍整个表来计算有多少行，但是 MyISAM 只要简单的读出保存好的行数即可。注意的是，当 count(\*)语句包含 where 条件时 MyISAM 也需要扫描整个表。
7. InnoDB 支持行锁（某些情况下还是锁整表，如 update table set a=1 where user like '%lee%'
8. 描述下scrapy框架运行的机制？

Scrapy是封装起来的框架，他包含了下载器、解析器，日志及异常处理。基于多线程，twisted的方式处理，对于固定单个网站的爬取开发，有优势，但对于多网站，100个网站，并发及分布式处理方面，不够灵活，不便调整与扩展。

从start\_urls里获取第一批url并发送请求，请求由引擎交给调度器请求队列，获取完毕后，调度器讲请求队列的请求交给下载器取获取请求对应的响应资源，并将响应交给自己编写的解析方法做提取处理：1、如果是取出需要的数据，则交给管道文件处理；2、如果是提取出url，则继续进行之前的步骤，直到请求队列里没有请求，程序结束。

1. 写爬虫是用多进程好？还是多线程好？ 为什么？

IO密集型（文件处理、网络爬虫等），多线程有效提高效率（单线程下有io操作会进行io等待，造成不必要的时间浪费，而开启多线程能在线程A等待时，自动切换到线程B，可以不浪费cpu的资源，从而提升程序执行效率）。在实际的数据采集过程中，及考虑网速和响应的问题，也需要考虑自身机器的硬件情况，来设置进程或线程）。

1. 数据库的优化？
2. 优化索引、sql语句、分析慢查询。
3. 设计表的时候严格根据数据库的范式来设计数据库。
4. 使用缓存，把经常访问的数据而且不经常变化的数据放在缓存中，能节约磁盘io。
5. 优化硬件；采用ssd，使用磁盘队列技术等
6. 垂直分表；把一些不经常读的数据放在一张表里，节约磁盘io；
7. 主从分离读写；采用主从复制把数据库的读操作和写入操作分离开来。
8. 选择合适的表引擎，参数上的优化；
9. 进行架构级别的缓存，静态化和分布式；
10. 不采用全文检索；
11. 采用更快的存储方式，如NOSQL存储经常访问的数据库。
12. 常见的反爬虫和应对方法？

1）通过Headers反爬虫

从用户请求的Headers反爬虫是最常见的反爬虫策略。很多网站都会对Headers的User-Agent进行检测，还有一部分网站会对Referer进行检测（一些资源网站的防盗链就是检测Referer）。如果遇到了这类反爬虫机制，可以直接在爬虫中添加Headers，将浏览器的User-Agent复制到爬虫的Headers中；或者将Referer值修改为目标网站域名。对于检测Headers的反爬虫，在爬虫中修改或者添加Headers就能很好的绕过。

2）基于用户行为反爬虫

还有一部分网站是通过检测用户行为，例如同一IP短时间内多次访问同一页面，或者同一账户短时间内多次进行相同操作。

大多数网站都是前一种情况，对于这种情况，使用IP代理就可以解决。可以专门写一个爬虫，爬取网上公开的代理ip，检测后全部保存起来。这样的代理ip爬虫经常会用到，最好自己准备一个。有了大量代理ip后可以每请求几次更换一个ip，这在requests或者urllib2中很容易做到，这样就能很容易的绕过第一种反爬虫。

对于第二种情况，可以在每次请求后随机间隔几秒再进行下一次请求。有些有逻辑漏洞的网站，可以通过请求几次，退出登录，重新登录，继续请求来绕过同一账号短时间内不能多次进行相同请求的限制。

3）动态页面的反爬虫

上述的几种情况大多都是出现在静态页面，还有一部分网站，我们需要爬取的数据是通过ajax请求得到，或者通过JavaScript生成的。首先用Fiddler对网络请求进行分析。如果能够找到ajax请求，也能分析出具体的参数和响应的具体含义，我们就能采用上面的方法，直接利用requests或者urllib2模拟ajax请求，对响应的json进行分析得到需要的数据。

能够直接模拟ajax请求获取数据固然是极好的，但是有些网站把ajax请求的所有参数全部加密了。我们根本没办法构造自己所需要的数据的请求。这种情况下就用selenium+phantomJS，调用浏览器内核，并利用phantomJS执行js来模拟人为操作以及触发页面中的js脚本。从填写表单到点击按钮再到滚动页面，全部都可以模拟，不考虑具体的请求和响应过程，只是完完整整的把人浏览页面获取数据的过程模拟一遍。

用这套框架几乎能绕过大多数的反爬虫，因为它不是在伪装成浏览器来获取数据（上述的通过添加 Headers一定程度上就是为了伪装成浏览器），它本身就是浏览器，phantomJS就是一个没有界面的浏览器，只是操控这个浏览器的不是人。利selenium+phantomJS能干很多事情，例如识别点触式（12306）或者滑动式的验证码，对页面表单进行暴力破解等。

8、分布式爬虫主要解决什么问题？

1)ip

2)带宽

3）cpu

4）io

9、爬虫过程中验证码怎么处理？

1.输入式验证码：这种验证码主要是通过用户输入图片中的字母、数字、汉字等进行验证。使用OCR识别技术，直接使用python的第三方库，tesserocr。

2、滑动式验证码：模拟人去拖动验证码的行为，点击按钮，弹出没有缺口的图片，然后获取没有缺口的图片，点击滑动按钮，弹出带缺口的图片，获取带缺口的图片，对比两张图片的所有RBG像素点，得到不一样像素点的x值，即要移动的距离，模拟人的习惯，把需要拖动的总距离分成一段一段小的轨迹，然后使用[selenium](https://www.baidu.com/s?wd=selenium&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)拖到缺口位置。

3、点击式的图文验证和图标选择：借助第三方识别接口识别出相应的内容，推荐超级鹰，把验证码发送过去，然后使用selenium模拟点击即可。

4、宫格验证码：但是我们发现不一样的验证码个数是有限的，这里采用模版匹配的方法。我觉得就好像暴力枚举，把所有出现的验证码保存下来，然后挑出不一样的验证码，按照拖动顺序命名，我们从左到右上下到下，设为1,2,3,4。上图的滑动顺序为4,3,2,1所以我们命名4\_3\_2\_1.png，这里得手动搞。当验证码出现的时候，用我们保存的图片一一枚举，与出现这种比较像素，方法见上面。如果匹配上了，拖动顺序就为4,3,2,1。然后使用selenium模拟即可。

10、列举您使用过的python网络爬虫所用到的网络数据包？

requests、urlib、urlib2

11、列举您使用过的Python网络爬虫所用到的解析数据包？

Beautifulsoup、xpath、lxml

12、解析网页的解析器使用最多的是哪几个?

re正则表达式（模糊匹配）、python自带的html.parser模块、第三方库beautifulSoup以及lxml库。（后面三个都是结构化解析）

13、cookie过期的处理问题？

采用selenium自动登陆获取cookie，保存到文件

读取cookie，比较cookie的有效期，若过期则再次重新获取cookie。

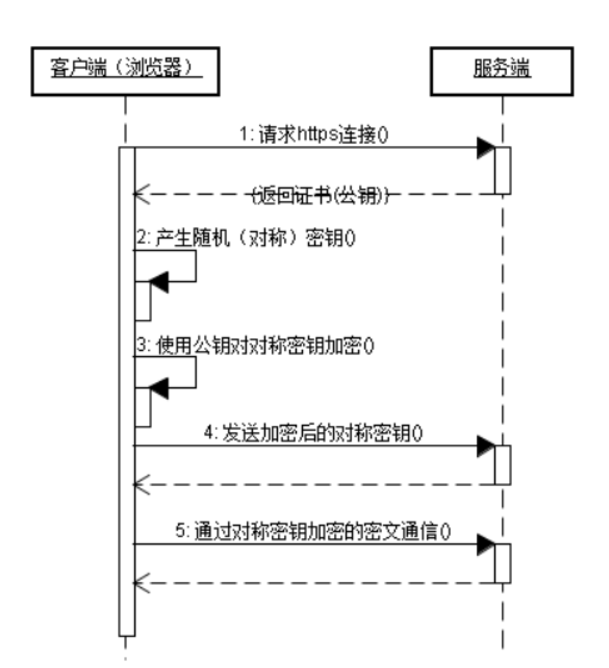
在请求其他网页时，填入cookie，实现登陆状态的保持。

1. 动态加载又对及时性要求很高怎么处理？

使用selenium和Phantomjs

尽量不使用sleep，而使用WebDriverWait

1. https是如何实现安全传输数据的？



16、TTL，MSL，RTT各是什么？

TTL:生存时间

MSL：报文最大生存时间

RTT：客户到服务器往返所花时间。

17、谈谈你对selenium和phantomjs的了解？

Selenium是一个web自动化测试工具，最初是为网站自动化测试而开发的。可以根据我们的指令，让浏览器自动加载页面，获取需要的页面，甚至页面截屏。他支持所有主流的浏览器（包括phanthomjs这些无界面的浏览器）。

Phantomjs是一个基于webkit的无界面的浏览器，他会把网站加载到内存并执行页面上的js，因为不会展示图形界面，所以运行起来比完整的浏览器更高效。

18、爬虫中间件？

Spider Middlewares(主要功能是在爬虫运行过程中进行一些处理) 和Downloader Middleware(主要功能在请求网页后，页面被下载时进行一些处理)。

将fake-useragent和ip代理集成到Middlewares中， 使spider在发送request时主动更换User-Agent和IP。

19、代理问题？

网站的反爬虫策略会检测到同一个ip访问次数频率过快，从而禁止该ip的访问。因此爬虫过程中需要ip代理避免该问题。

在spiderMiddles中添加获取代理的ip方法，为了在给网站发送请求前获得有效的代理ip。

失效？事先用检测代码检测可用的代理，每隔一段时间更换一次代理，如果出现302等状态码，则立即更换下一个可用的ip。

20、scrapy的去重原理？

对于每一个url请求，调度器都会根据请求的相关信息加密得到一个指纹信息，并且将指纹信息和set集合中的指纹信息进行比对，如果set集合中已经存在这个数据中，就不再将这个request放入队列中。如果set集合中没有存在这个加密后的数据，就将这个request对象放入队列中，等待被调度。

21、数据提取问题？

主要使用什么样的结构化数据提取方式？

Json文件：先转换为python类型，然后用json.dumps()转换。

Xml文件：转换为python类型(xmltodict)、xpath、css选择器、正则表达式

动态加载的数据如何获取？

Selenium获取，分析请求页面，获取它的js请求文件。

Json数据如何提取？

采用正则表达式解析：获取整个json数据后，采用正则表达式匹配到关键词；

基于json格式进行获取：使用json.dumps()将python的dict转成json串。Json.loads()将json串转化为dict类型。

22、怎么监控爬虫的状态?

获取要展示的数据，包含当前的时间数据，存到influxDb(目前比较流行的时间序列数据库)里面，然后再到Grafana(可视化面板)里面进行相应的配置即可。

23、怎样让scrapy框架发送一个post请求？

可以使用FormRequest，也可使用FormRequest.from\_response()来实现。





24、怎么判断网站是否更新？

1、304http状态码：当第二次请求页面，如果页面没更新，则会反馈304状态码。

2、通过Last-Modified文件最后修改时间判断。

3、使用MD5数字签名：每次下载网页时，把服务器返回的数据流ResponseStream先放在内存缓冲区，然后对数据流生成MD5数字签名s1，下次下载同样生成签名s2，比较s1和s2，如果相同，则页面没有更新，否则更新。

4、比较文件的大小。

25、图片、视频爬取怎么绕过防盗连接？

在请求头上添加Referer属性就好（可以填爬取网站的地址），referer携带的数据，是用来服务器当前请求是从哪个页面请求过来的。

26、你爬出来的数据量大概有多大？大概多长时间爬一次？

一个晚上爬取了大约1000部小说35W个章节的信息，还在继续爬取中

一台机器10小时抓25万数据+6GB图，而且网站限制并发10，200次后需要等待15分钟。而且断线率超高，只有70%左右的数据是正常，所以要断线重抓，我是各种hack手段才搞到数据的。

27、用什么数据库存爬下来的数据？部署是你做的吗？怎么部署？

存到生产环境的PG数据库。

将脚本部署到Centos服务器，并设置定时任务（corntab），自动启动脚本。

28、scrapy实现增量爬虫？

主要有两种方法：1、是在获得页面解析的内容后判断该内容是都已经被爬取过；2、是在发送请求之前判断被请求的url是否已经被爬取过。前一种可感知每个页面的变化，能获取新增或变化的内容，但要对每个url发送请求，所以速度比较慢，服务器压力大。后一种则无法感知页面的变化，不用爬取已经爬过的url，服务器压力小，速度快，适用于爬取新增网页。

以下几种方式：

1. 缓存；通过开启缓存，将每个请求缓存至本地，下次爬取时，scrapy会优先从本地缓存中获得response，这种模式下，再次请求已爬取的网页不用从网络中获得响应，所以不受带宽影响，对服务器也不会造成额外的压力，但是无法获取网页变化的内容，速度也没有第二种方式快，而且缓存的文件会占用比较大的内存，在setting.py的以下注释用于设置缓存。适合内存大的主机。
2. 对item实现去重；
3. 对url进行去重；速度快，服务器压力下。自定义下载中间件。

29、爬取下来的数据如何去重，说下具体的算法数据。

通过MD5生成电子指纹来判断页面是否改变。

Nutch去重。Nutch中的digest是对采集的每一个页面内容的32哈希值，如果每个页面内容完全一样，他们的digest值肯定会一样。

30、怎么设置爬取的深度？

通过在seeting.py中设置DEPTH\_LIMIT的值可以限制爬取的深度。这个深度是与start\_url中定义的相对值。也就是相对url的深度。例如定义url为：http://www.domz.com/game/,DEPTH\_LIMIT=1那么限制爬取的只能是此url下一级的网页。深度大于设置值的将被ignore。

31、什么是分布式存储？

分布式存储简单来说，就是将数据分散存储到多个存储服务器上，并将这些分散的存储资源构成一个虚拟的存储设备，实际上数据分散的存储在企业的各个角落。

数据库：

1. 视图的作用？可以更改吗？

视图是虚拟的表，与包含数据的表不一样，视图只包含使用时动态检索数据的查询；不包含任何列或数据。使用视图可以简化复杂的sql操作。隐藏具体的细节，保护数据；视图创建后，可以使用与表相同的方式利用它们。

对于某些视图比如未使用联结子查询分组聚集函数等，是可以对其更新的，对试图的更新将对基表进行更新；但是视图主要用于简化索引，保护数据，并不用于更新，而且大部分视图都不可以更新。

1. drop、delete、truncate的区别？

Truncate 和delete只删除记录不删除表的结构，而drop语句将删除表的结构依赖的约束，触发器和索引。

truncate之后的自增字段从头开始计数了，而delete之后的自增字段仍保留原来的数值。

速度上，drop>truncate>delete

1. 索引的工作原理和种类？

索引是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B树及其B+树。

种类：唯一索引，主键索引，聚集索引

1. 连接种类？

内连接，自然链接，外连接(左外，右外，全外)

1. 存储过程和触发器？

存储过程是一组为了完成特定功能的sql语句集，存储在数据库中，第一次经过编译后再次调用不需要再次编译。优点：允许模块化程序设计、更快执行，减少网络流量，更好的安全机制。

触发器是一种特殊的存储过程，它又不同于存储过程，触发器主要是通过事件进行触发而被执行的，而存储过程可以通过存储过程名字而被直接调用。

1. 乐观锁和悲观锁？

悲观锁：总是假设最坏的情况，每次去拿数据的时候都认为别人会修改，所以每次在拿数据的时候都会上锁，这样别人想拿这个数据就会阻塞直到它拿到锁（共享资源每次只给一个线程使用，其他线程阻塞，用完后再把资源转让给其他线程）。比如：行锁、表锁、读锁、写锁等都是在操作之前先上锁。

乐观锁：总是假设最好的情况，每次去拿数据的时候都认为别人不会修改，所以不会上锁，但是在更新的时候会判断一下在此期间别人有没有去更新这个数据，可以用版本号机制和CAS算法实现。乐观锁多用于多读的应用类型，这样可以提高吞吐量，像数据库提供的类似于**write\_condition机制**，其实都是提供的乐观锁。

1. Redis宕机怎么解决?

如果一台redis，数据会丢失，无法挽救。

1. slave从redis宕机。

在redis中从库重新启动后会自动加入到主从架构中，自动完成同步数据。

如果从数据库实现了持久化，可以直接连接到主的上面，只要实现增量备份，重新连接到主从架构中会实现增量同步。

1. Master宕机。

假如主从都没数据持久化，此时千万不要立马重启服务，否则可能会造成数据丢失。正确操作如下：

1. 在slave数据上执行SLAVEOF ON ONE ,来断开主从关系并把slave升级为主库。
2. 此时重启主数据库，执行SLAVEOF，把他设置为从库，连接到主的redis上面做主从复制，自动备份数据。
3. redis和mecached的区别，以及使用场景？

存储方式：memecache把数据全部存在内存中，断掉电后会挂掉，数据不能超过内存大小。redis有部分存在硬盘上，这样能保证数据的持久性，支持数据的持久化。（笔者注：有快照和AOF日志两种持久化方式，在实际应用的时候，要特别注意配置文件快照参数，要不就很有可能[服务器](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)频繁满载做dump）。

数据支持类型：redis在数据支持上要比memecache多的多。Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。memcache支持简单的数据类型，String。redis还支持数据的备份，即slave-master备份。

redis的速度比memecache快的多。memcache是多线程，非阻塞io服用的网络模型；redis使用单线程的io复用模型。

运行环境上：redis目前官方只支持LINUX上运行，省去了对与其他系统的支持，可以把更好的精力用于本系统环境上的优化。



有持久化需求或者对数据结构和处理有高级要求的应用，选择redis，其他简单的key/value存储，选择memcache。

1. Redis集群方案该怎么做?都有哪些方案?

官方cluster方案、twemproxy方案、哨兵模式、codis、客户端分片

1. redis回收算法：LRU算法
2. 、MongoDB中对多条记录做更新操作命令是什么？

db.collection.update(criteria,objNew,upsert,multi)

criteria：查询条件

objNew：update对象和一些更新操作符

upsert：如果不存在update的记录，是否插入objNew这个新的文档，true为插入，默认为false，不插入。

multi：默认是false，只更新找到的第一条记录。如果为true，把按条件查询出来的记录全部更新。

1. 、MongoDB如何才会拓展到多个shard（分片）里？

mongodb分片是基于区域的，所以一个集合的所有对象都放置在放置在同一个块中，只有当存在多余一个块的时候，才会有多个分片获取数据的选项。