python面试题搜集（五）： 整理的最全 python常见面试题（基本必考）

1、大数据的文件读取

① 利用生成器generator

②迭代器进行迭代遍历：for line in file

2、迭代器和生成器的区别

1）迭代器是一个更抽象的概念，任何对象，如果它的类有next方法和iter方法返回自己本身。对于string、list、dict、tuple等这类容器对象，使用for循环遍历是很方便的。在后台for语句对容器对象调用iter()函数，iter()是python的内置函数。iter()会返回一个定义了next()方法的迭代器对象，它在容器中逐个访问容器内元素，next()也是python的内置函数。在没有后续元素时，next()会抛出一个StopIteration异常

2）生成器（Generator）是创建迭代器的简单而强大的工具。它们写起来就像是正规的函数，只是在需要返回数据的时候使用yield语句。每次next()被调用时，生成器会返回它脱离的位置（它记忆语句最后一次执行的位置和所有的数据值）

区别：生成器能做到迭代器能做的所有事,而且因为自动创建了**iter**()和next()方法,生成器显得特别简洁,而且生成器也是高效的，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状态的自动方法,当发生器终结时,还会自动抛出StopIteration异常

3、装饰器的作用和功能：

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

权限校验等场景

缓存

4、简单谈下GIL：

Global Interpreter Lock(全局解释器锁)

Python代码的执行由Python 虚拟机(也叫解释器主循环，CPython版本)来控制，Python 在设计之初就考虑到要在解释器的主循环中，同时只有一个线程在执行，即在任意时刻，只有一个线程在解释器中运行。对Python 虚拟机的访问由全局解释器锁（GIL）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 虚拟机按以下方式执行：

\1. 设置GIL \2. 切换到一个线程去运行 \3. 运行： a. 指定数量的字节码指令，或者

b. 线程主动让出控制（可以调用time.sleep(0)） \4. 把线程设置为睡眠状态 \5. 解锁GIL \6. 再次重复以上所有步骤

在调用外部代码（如C/C++扩展函数）的时候，GIL 将会被锁定，直到这个函数结束为止（由于在这期间没有Python 的字节码被运行，所以不会做线程切换）。

5、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通常用来再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。

6、线上服务可能因为种种原因导致挂掉怎么办?

linux下的后台进程管理利器 supervisor

每次文件修改后再linux执行 service supervisord restart

7、如何提高python的运行效率

使用生成器；关键代码使用外部功能包（Cython，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

8、常用Linux命令：

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

9、Python中的yield用法

yield简单说来就是一个生成器，这样函数它记住上次返 回时在函数体中的位置。对生成器第 二次（或n次）调用跳转至该函 次）调用跳转至该函 数。

10、Python是如何进行内存管理的

一、垃圾回收：python不像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语言来讲，对象的类型和内存都是在运行时确定的。这也是为什么我们称Python语言为动态类型的原因（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）。

二、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，都维护这一个对指向该对对象的引用的计数。当变量被绑定在一个对象上的时候，该变量的引用计数就是1，(还有另外一些情况也会导致变量引用计数的增加),系统会自动维护这些标签，并定时扫描，当某标签的引用计数变为0的时候，该对就会被回收。

三、内存池机制Python的内存机制以金字塔行，-1，-2层主要有操作系统进行操作，

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

11、描述数组、链表、队列、堆栈的区别?

数组与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，而堆栈是后进先出；队列和堆栈可以用数组来实现，也可以用链表实现。

12、你知道几种排序,讲一讲你最熟悉的一种?

排序方法	平均情况	最好情况	最坏情况	辅助空间	稳定性
冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
插入排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
希尔排序	$O(n \log n)$~ $O(n^2)$	$O(n^{1.3})$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	不稳定
归并排序	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$	稳定
快速排序	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$~ $O(n)$	不稳定

你是最棒的!

web框架部分

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应用服务器登陆的session数据没有共享到B应用服务器，纳米之前的登录状态就没有了。

2.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

4.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?

答：中间件是介于request与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的输入与输出。

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：只需要使用起ORM，做简单的对象定义，它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程度高。

应用程序必须使用Django内置的ORM，否则就不能享受到框架内提供的种种基于其ORM的便利；理论上可以切换掉其ORM模块，但这就相当于要把装修完毕的房子拆除重新装修，倒不如一开始就去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，都需要对其进行重构，才能满足性能的要求。

Django适用的是中小型的网站，或者是作为大型网站快速实现产品雏形的工具。

Django模板的设计哲学是彻底的将代码、样式分离； Django从根本上杜绝在模板中进行编码、处理数据的可能。

\7. Django重定向你是如何实现的？用的什么状态码？

使用HttpResponseRedirect

redirect和reverse

状态码：302,301

8.ngnix的正向代理与反向代理？

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。

反向代理正好相反，对于客户端而言它就像是原始服务器，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，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端，就像这些内容原本就是它自己的一样。

\9. Tornado 的核是什么？

Tornado 的核心是 ioloop 和 iostream 这两个模块，前者提供了一个高效的 I/O 事件循环，后者则封装了 一个无阻塞的 socket 。通过向 ioloop 中添加网络 I/O 事件，利用无阻塞的 socket ，再搭配相应的回调 函数，便可达到梦寐以求的高效异步执行。

10.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 开启的方式也是单进程 。

uWSGI 是一个 Web 服务器，它实现了 WSGI 协议、uwsgi、http 等协议。注意 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，并且搭配着 Nginx

就是一个生产环境了，能够将用户访问请求与应用 app 隔离开，实现真正的部署 。相比来讲，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。

你是最棒的！

网络编程和前端部分

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。

2、传输数据的大小

在HTTP规范中，没有对URL的长度和传输的数据大小进行限制。但是在实际开发过程中，对于GET，特定的浏览器和服务器对URL的长度有限制。因此，在使用GET请求时，传输数据会受到URL长度的限制。

对于POST，由于不是URL传值，理论上是不会受限制的，但是实际上各个服务器会规定对POST提交数据大小进行限制，Apache、IIS都有各自的配置。

3、安全性

POST的安全性比GET的高。这里的安全是指真正的安全，而不同于上面GET提到的安全方法中的安全，上面提到的安全仅仅是不修改服务器的数据。比如，在进行登录操作，通过GET请求，用户名和密码都会暴露再URL上，因为登录页面有可能被浏览器缓存以及其他人查看浏览器的历史记录的原因，此时的用户名和密码就很容易被他人拿到了。除此之外，GET请求提交的数据还可能会造成Cross-site request frogery攻击。

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

你是最棒的!

爬虫和数据库部分

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，但是不支持分布式。而scrapy-redis一套基于redis数据库、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数据库里的item队列、请求队列和请求指纹集合。

2) 为什么选择redis数据库，因为redis支持主从同步，而且数据都是缓存在内存中的，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。

\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：urllib，urllib2

第 三 方：requests

框 架：Scrapy

urllib和urllib2模块都做与请求URL相关的操作，但他们提供不同的功能。

urllib2.：urllib2.urlopen可以接受一个Request对象或者url，（在接受Request对象时候，并以此可以来设置一个URL 的headers），urllib.urlopen只接收一个url

urllib 有urlencode,urllib2没有，因此总是urllib，urllib2常会一起使用的原因

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，但是对于多网站爬取 100个网站，并发及分布式处理方面，不够灵活，不便调整与括展。

request 是一个HTTP库， 它只是用来，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，灵活性更高，高并发与分布式部署也非常灵活，对于功能可以更好实现.

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上爬行

支持shell方式，方便独立调试

写middleware,方便写一些统一的过滤器

通过管道的方式存入数据库

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。

3.你常用的mysql引擎有哪些？各引擎间有什么区别？

主要 MyISAM 与 InnoDB 两个引擎，其主要区别如下：

一、InnoDB 支持事务，MyISAM 不支持，这一点是非常之重要。事务是一种高

级的处理方式，如在一些列增删改中只要哪个出错还可以回滚还原，而 MyISAM

就不可以了；

二、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；

三、InnoDB 支持外键，MyISAM 不支持；

四、MyISAM 是默认引擎，InnoDB 需要指定；

五、InnoDB 不支持 FULLTEXT 类型的索引；

六、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；

七、对于自增长的字段，InnoDB 中必须包含只有该字段的索引，但是在 MyISAM

表中可以和其他字段一起建立联合索引；

八、清空整个表时，InnoDB 是一行一行的删除，效率非常慢。MyISAM 则会重

建表；

九、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%'

4.描述下scrapy框架运行的机制？

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入请求队列，获取完毕后，调度器将请求队列里的请求交给下载器去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. 如果提取出需要的数据，则交给管道文件处理；2. 如果提取出url，则继续执行之前的步骤（发送url请求，并由引擎将请求交给调度器入队列...），直到请求队列里没有请求，程序结束。

5.什么是关联查询，有哪些？

将多个表联合起来进行查询，主要有内连接、左连接、右连接、全连接（外连接）

6.写爬虫是用多进程好？还是多线程好？ 为什么？

IO密集型代码(文件处理、网络爬虫等)，多线程能够有效提升效率(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，从而能提升程序执行效率)。在实际的数据采集过程中，既考虑网速和响应的问题，也需要考虑自身机器的硬件情况，来设置多进程或多线程

7.数据库的优化？

\1. 优化索引、SQL 语句、分析慢查询；

\2. 设计表的时候严格根据数据库的设计范式来设计数据库；

\3. 使用缓存，把经常访问到的数据而且不需要经常变化的数据放在缓存中，能

节约磁盘IO；

\4. 优化硬件；采用SSD，使用磁盘队列技术(RAID0,RAID1,RDID5)等；

\5. 采用MySQL 内部自带的表分区技术，把数据分层不同的文件，能够提高磁

盘的读取效率；

\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I/O；

\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. 分库分表分机器（数据量特别大），主要的的原理就是数据路由；

\9. 选择合适的表引擎，参数上的优化；

\10. 进行架构级别的缓存，静态化和分布式；

\11. 不采用全文索引；

\12. 采用更快的存储方式，例如 NoSQL存储经常访问的数据

8.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。

9.分布式爬虫主要解决什么问题?

1)ip

2)带宽

3）cpu

4）io

10.爬虫过程中验证码怎么处理?

1.scrapy自带

2.付费接口