python常见面试题（一）

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

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

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

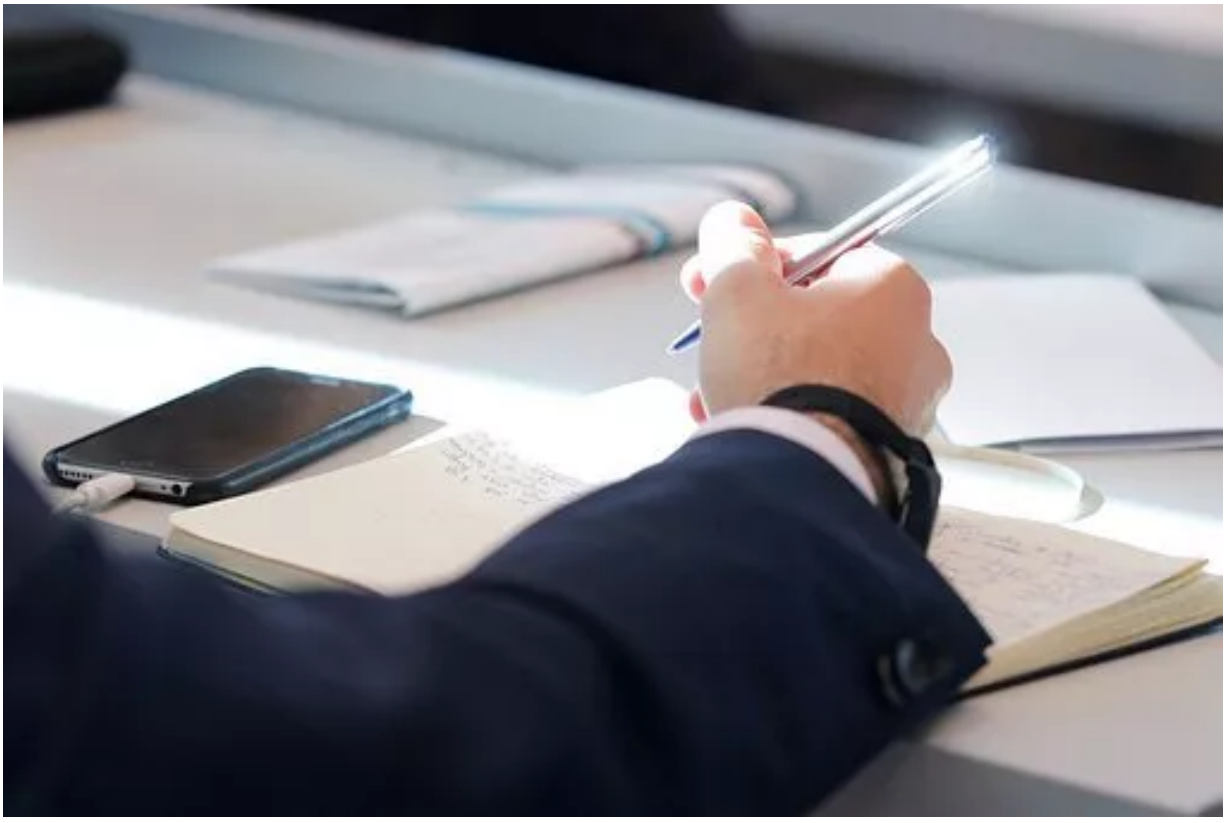
执行函数后的清理功能

权限校验等场景

缓存

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）。

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。

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

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

常用Linux命令

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

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。

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。

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（n log n）~O（n）	不稳定

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

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。

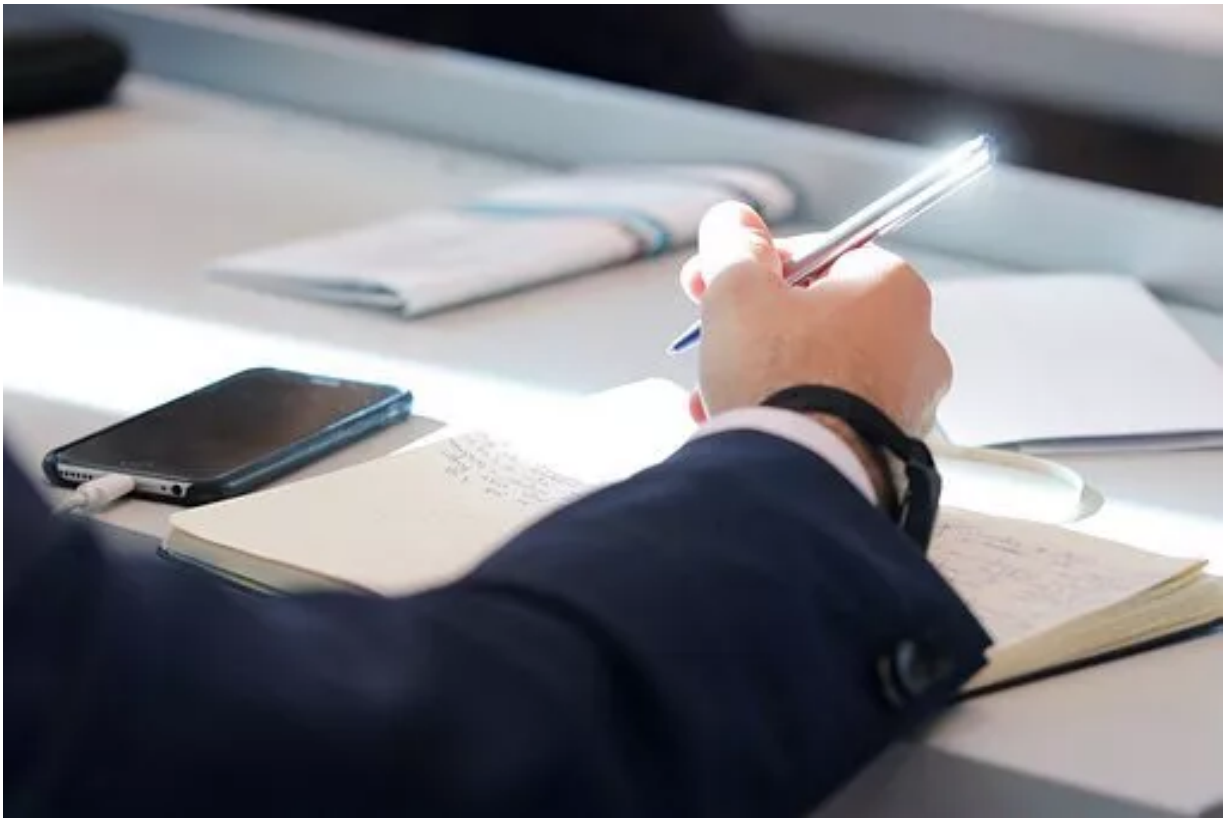
2.跨域请求问题django怎么解决的（原理）

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。

你对Django的认识?

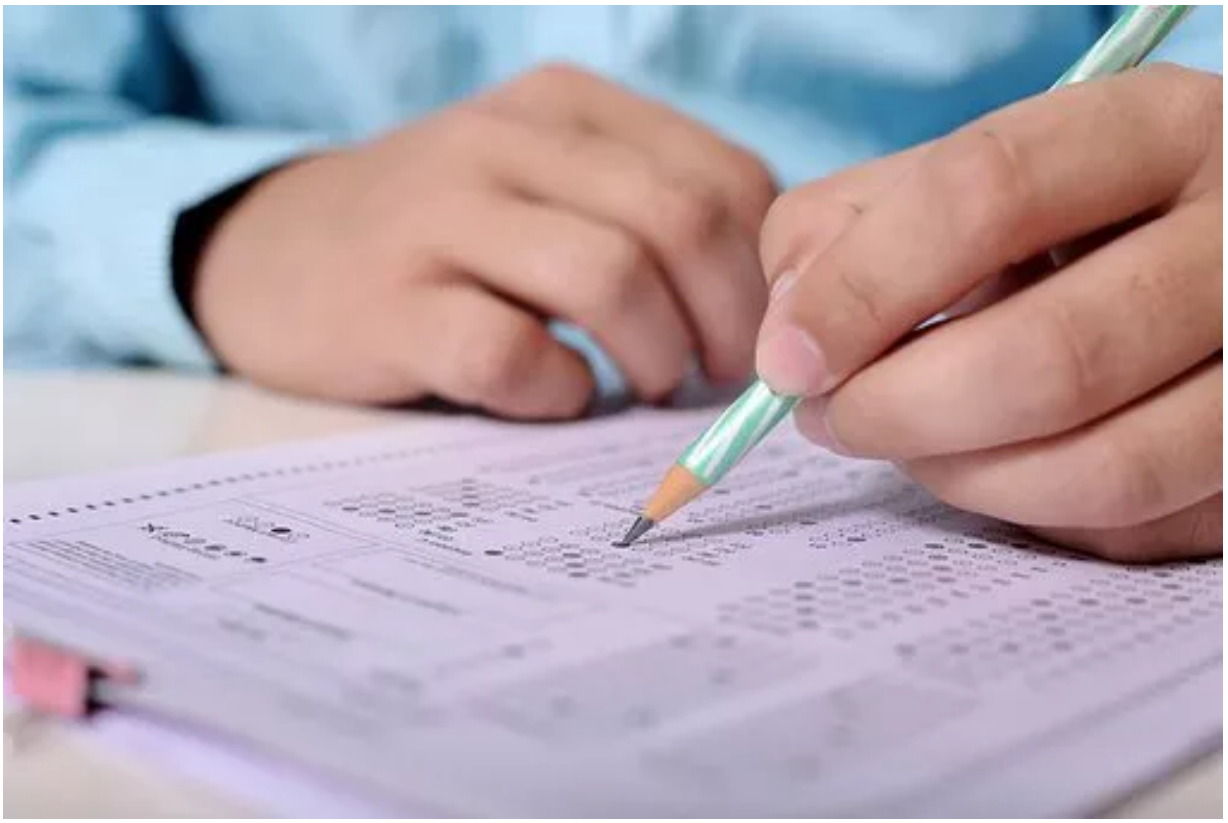
Django是走大而全的方向，它最出名的是其全自动化的管理后台：只需要使用起ORM，做简单的对象定义，它就能自动生成数据库结构、以及全功能的管理后台。
Django内置的ORM跟框架内的其他模块耦合程度高。

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。

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



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。

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。

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。