# 200道python面试题

1. Python是如何进行内存管理的？

Python内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。

引用计数增加的情况：

1，一个对象分配一个新名称

2，将其放入一个容器中（如列表、元组或字典）

引用计数减少的情况：

1，使用del语句对对象别名显示的销毁

2，引用超出作用域或被重新赋值

sys.getrefcount( )函数可以获得对象的当前引用计数

多数情况下，引用计数比你猜测得要大得多。对于不可变数据（如数字和字符串），解 释器会在程序的不同部分共享内存，以便节约内存。

1. 什么是lambda函数？它有什么好处?

lambda 表达式，通常是在需要一个函数，但是又不想费神去命名一个函数的场合下使 用，也就是指匿名函数

lambda函数：首要用途是指点短小的回调函数

1. Python里面如何实现tuple和list的转换？

直接使用tuple和list函数就行了，type()可以判断对象的类型

1. 请写出一段Python代码实现删除一个list里面的重复元素

1,使用set函数，set(list)

2，使用字典函数，

>>>a=[1,2,4,2,4,5,6,5,7,8,9,0]

>>> b={}

>>>b=b.fromkeys(a)

>>>c=list(b.keys())

>>> c

1. 编程用sort进行排序，然后从最后一个元素开始判断

a=[1,2,4,2,4,5,7,10,5,5,7,8,9,0,3]

a.sort()

last=a[-1]

for i inrange(len(a)-2,-1,-1):

if last==a[i]:

del a[i]

else:last=a[i]

print(a)

1. Python里面如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）

赋值（=），就是创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。

浅拷贝：创建一个新的对象，但它包含的是对原始对象中包含项的引用（如果用引用的 方式修改其中一个对象，另外一个也会修改改变）{1,完全切片方法；2，工厂函数，如 list()；3，copy模块的copy()函数}

深拷贝：创建一个新的对象，并且递归的复制它所包含的对象（修改其中一个，另外一 个不会改变）{copy模块的deep.deepcopy()函数}

1. 介绍一下except的用法和作用？

try…except…except…[else…][finally…]

执行try下的语句，如果引发异常，则执行过程会跳到except语句。对每个except分支 顺序尝试执行，如果引发的异常与except中的异常组匹配，执行相应的语句。如果所 有的except都不匹配，则异常会传递到下一个调用本代码的最高层try代码中。

try下的语句正常执行，则执行else块代码。如果发生异常，就不会执行

如果存在finally语句，最后总是会执行。

1. Python中pass语句的作用是什么？

pass语句不会执行任何操作，一般作为占位符或者创建占位程序，whileFalse:pass

1. 介绍一下Python下range()函数的用法？

列出一组数据，经常用在for in range()循环中

1. 如何用Python来进行查询和替换一个文本字符串？

可以使用re模块中的sub()函数或者subn()函数来进行查询和替换，

格式：sub(replacement, string[,count=0])（replacement是被替换成的文本，string是需 要被替换的文本，count是一个可选参数，指最大被替换的数量）

import re

>>>p=re.compile(‘blue|white|red')

>>>print(p.sub(‘colour','blue socks and red shoes'))

colour socks and colourshoes

>>>print(p.sub(‘colour','blue socks and red shoes',count=1))

colour socks and redshoes

1. python里面match()和search()的区别？

re模块中match(pattern,string[,flags]),检查string的开头是否与pattern匹配。

re模块中research(pattern,string[,flags]),在string搜索pattern的第一个匹配值。

>>>print(re.match(‘super', ‘superstition').span())

(0, 5)

>>>print(re.match(‘super', ‘insuperable'))

None

>>>print(re.search(‘super', ‘superstition').span())

(0, 5)

>>>print(re.search(‘super', ‘insuperable').span())

(2, 7)

1. 用Python匹配HTML tag的时候，<.\*>和<.\*?>有什么区别？

术语叫贪婪匹配( <.\*> )和非贪婪匹配(<.\*?> )

例如:

test

<.\*> :

test

<.\*?> :

1. Python里面如何生成随机数？

random模块

随机整数：random.randint(a,b)：返回随机整数x,a<=x<=b

random.randrange(start,stop,[,step])：返回一个范围在(start,stop,step)之间的随机整数， 不包括结束值。

随机实数：random.random( ):返回0到1之间的浮点数

random.uniform(a,b):返回指定范围内的浮点数。

1. 有没有一个工具可以帮助查找python的bug和进行静态的代码分析？

PyChecker是一个python代码的静态分析工具，它可以帮助查找python代码的bug, 会 对代码的复杂度和格式提出警告

Pylint是另外一个工具可以进行codingstandard检查

1. 如何在一个function里面设置一个全局的变量？

解决方法是在function的开始插入一个global声明：

def f()

global x

1. 单引号，双引号，三引号的区别?

单引号和双引号是等效的，如果要换行，需要符号(\),三引号则可以直接换行，并且可 以包含注释

如果要表示Let's go 这个字符串

单引号：s4 = ‘Let\'s go'

双引号：s5 = “Let's go”

s6 = ‘I realy like“python”!'

这就是单引号和双引号都可以表示字符串的原因了

1. 谈谈对 Python 和其他语言的区别?

Python 是一门语法简洁优美,功能强大无比,应用领域非常广泛,具有强大完备的第三方库，他是一门强类型的可移植、可扩展，可嵌入的解释型编程语言，属于动态语言。

拿 C 语言和 Python 比： Python 的第三方类库比较齐全并且使用简洁,很少代码就能实现一些功能，如果用 C 去实现相同的功能可能就比较复杂。但是对于速度来说 Python 的运行速度相较于 C 就比较慢了。所以有利的同时也有弊端，毕竟我们的学习成本降低了。

1. 简述解释型和编译型编程语言?

解释型语言是在运行程序的时候才翻译，每执行一次，要翻译一次，效率较低。 编译型就是直接编译成机型可以执行的，只翻译一次，所以效率相对来说较高.

1. Python 的解释器种类以及相关特点？

CPython c 语言开发的，使用最广的解释器

IPython 基于 cPython 之上的一个交互式计时器，交互方式增强功能和 cPython 一样

PyPy 目标是执行效率，采用 JIT 技术。对 Python 代码进行动态编译，提高执行效率

JPython 运行在 Java 上的解释器，直接把 Python 代码编译成 Java 字节码执行

IronPython 运行在微软 .NET 平台上的解释器，把 Python 编译成 . NET 的字节码。

1. Python3 和 Python2 的区别？

print 在 Python3 中是函数必须加括号，Python2 中 print 为 class。

Python2 中使用 xrange，Python3 使用 range。

Python2 中默认的字符串类型默认是 ASCII，Python3 中默认的字符串类型是 Unicode。

Python2 中/的结果是整型，Python3 中是浮点类型。

Python2 中声明元类：metaclass = MetaClass,Python3 中声明元类：class newclass(metaclass=MetaClass)：pass。

1. Python3 和 Python2 中 int 和 long 区别？

Python2 有 int 和 long 类型。int 类型最大值不能超过 sys.maxint，而且这个最大值是平台相关的。可以通过在数字的末尾附上一个Ｌ来定义长整型，显然，它比 int 类型表示的数字范围更大。在 Python3 里，只有一种整数类型 int，大多数情况下，和 Python２中的长整型类似。

1. xrange 和 range 的区别？

xrange 是在 Python2 中的用法，Python3 中只有 range xrange 用法与 range 完全相同，所不同的是生成的不是一个 list 对象，而是一个生成器。

1. 什么是 PEP8?

PEP8 通常会听别人提到，但是具体的指什么内容呢，简单介绍下。 《Python Enhancement Proposal #8》（8 号 Python 增强提案）又叫 PEP8，他针对的 Python 代码格式而编订的风格指南。

1. 了解 Python 之禅么？

通过 import this 语句可以获取其具体的内容。它告诉大家如何写出高效整洁的代码。

1. 了解 DocStrings 么？

DocStrings 文档字符串是一个重要工具，用于解释文档程序，帮助你的程序文档更加简单易懂。主要是解释代码作用的。

1. 了解类型注解么？

PEP 484 引入了类型提示，这使得可以对 Python 代码进行静态类型检查。 在使用 Ide 的时候可以获取到参数的类型，更方便传入参数。使用格式如下

def foo(num： int) -> None：

print(f"接收到的数字是：{num}")

1. 例举你知道 Python 对象的命名规范，例如方法或者类等

类：总是使用首字母大写单词串，如 MyClass。内部类可以使用额外的前导下划线。 变量：小写，由下划线连接各个单词。方法名类似 常量：常量名所有字母大写 等

1. Python 中的注释有几种？

总体来说分为两种，单行注释和多行注释。

单行注释在行首是 #。

多行注释可以使用三个单引号或三个双引号，包括要注释的内容。

1. 如何优雅的给一个函数加注释？

可以使用 docstring 配合类型注解

1. 如何给变量加注释？

可以通过变量名：类型的方式如下

a： str = "this is string type"

1. Python 代码缩进中是否支持 Tab 键和空格混用。

不允许 tab 键和空格键混用，这种现象在使用 sublime 的时候尤为明显。

一般推荐使用 4 个空格替代 tab 键。

1. 是否可以在一句 import 中导入多个库?

可以是可以，但是不推荐。因为一次导入多个模块可读性不是很好，所以一行导入一个模块会比较好。同样的尽量少用 from modulename import \*，因为判断某个函数或者属性的来源有些困难，不方便调试，可读性也降低了。

1. 在给 Py 文件命名的时候需要注意什么?

给文件命名的时候不要和标准库库的一些模块重复，比如 abc。 另外要名字要有意义，不建议数字开头或者中文命名。

1. 例举几个规范 Python 代码风格的工具

pylint 和 flake8

数据类型-字符串

1. 列举 Python 中的基本数据类型？

Python3 中有六个标准的数据类型：字符串（String）、数字（Digit）、列表（List）、元组（Tuple）、集合（Sets）、字典（Dictionary）。

1. 如何区别可变数据类型和不可变数据类型?

从对象内存地址方向来说

可变数据类型：在内存地址不变的情况下，值可改变（列表和字典是可变类型，但是字典中的 key 值必须是不可变类型）

不可变数据类型：内存改变，值也跟着改变。（数字，字符串，布尔类型，都是不可变类型）可以通过 id() 方法进行内存地址的检测。

1. 将"hello world"转换为首字母大写"Hello World"

这个得看清题目是要求两个单词首字母都要大写，如果只是第一个单词首字母大小的 话，只使用 capitalize 即可，但是这里是两个单词，所以用下面的方法。

arr = "hello world".split(" ")

new\_str = f"{arr[0].capitalize()} {arr[1].capitalize()}"

print(new\_str)

后来评论中有朋友提到了下面的方法，这里感谢这位朋友提醒。方案如下

"hello world".title()

1. 如何检测字符串中只含有数字?

可以通过 isdigit 方法，例子如下

s1 = "12223".isdigit()

print(s1)

s2 = "12223a".isdigit()

print(s2)

#结果如下：

#True

#False

1. 将字符串"ilovechina"进行反转

s1 = "ilovechina"[：：-1]

print(s1)

1. python高并发解决方案

gevent

代码看起来好看一些,但是维护比较差,patch没有规律,而且里面封装了C,对python3的支持最差.

twisted

稳定性是最好的,但是需要较长时间的学习.对python3的支持较差.

tornado

兼容性最好.但是过于简单了,功能不强,另外没有python数据库适配器能和tornado无缝对接,因此调用数据库很麻烦,而且只支持web.

1. 谈谈python的迭代器,生成器,装饰器,yield?

迭代器

(Iterator)迭代器是带状态的对象,它会记录当前迭代所在的位置,以方便下次迭代的时候获取正确的元素.

生成器

(generator)生成器和装饰器是python中最吸引人的两个黑科技,生成器虽没有装饰器那么常用,但在某些针对的情境下十分有效.

装饰器

(Decorator)装饰器是python中最吸引人的特性,装饰器本质上还是一个函数,它可以让已有的函数不做任何改动的情况下增加功能.

1. 标准库线程安全的队列是哪一个？不安全的是哪一个？logging是线程安全的吗？

线程

线程安全和非线程安全这些概念在其他的编程语言也同样使用.

所谓线程安全：就是对于多线程同时操作是是安全的而不会发生写冲突,比如python的Queue

相反非线程安全：就是多线成同时操作时会发生写冲突,比如python的其他list,set,dict

队列

Python的Queue模块中提供了同步的.线程安全的队列类,包括FIFO（先入先出)队列Queue,LIFO（后入先出）队列LifoQueue,和优先级队列PriorityQueue.这些队列都实现了锁原语,能够在多线程中直接使用.可以使用队列来实现线程间的同步.

三种队列

Python queue模块有三种队列:

1.FIFO队列先进先出.(线程安全)

2.LifoQueue类似于堆,即先进后出(线程安全)

3.PriorityQueue优先级队列,级别越低,越先出来(线程安全)

队列构造函数

针对这三种队列分别有三个构造函数:

1.class Queue.Queue(maxsize) FIFO

2.class Queue.LifoQueue(maxsize) LIFO

3.class Queue.PriorityQueue(maxsize) 优先级队列

logging

logging是Python标准库里的日志模块(线程安全)

1. 简述一下爬虫的步骤？

确定需求；

确定资源；

通过url获取网站的返回数据；

定位数据；

存储数据。

1. 遇到反爬机制怎么处理？

headers方向

判断User-Agent、判断Referer、判断Cookie。

将浏览器的headers信息全部添加进去

注意：Accept-Encoding；gzip,deflate需要注释掉

1. 说一说redis-scrapy中redis的作用?

它是将scrapy框架中Scheduler替换为redis数据库，实现队列管理共享。

优点：

可以充分利用多台机器的带宽；

可以充分利用多台机器的IP地址。

1. scrapy分为几个组成部分？分别有什么作用？

分为5个部分；Spiders(爬虫类)，Scrapy Engine(引擎),Scheduler(调度器),Downloader(下载器),Item Pipeline(处理管道)。

Spiders:开发者自定义的一个类，用来解析网页并抓取指定url返回的内容。

Scrapy Engine:控制整个系统的数据处理流程，并进行事务处理的触发。

Scheduler：接收Engine发出的requests，并将这些requests放入到处理列队中，以便之后engine需要时再提供。

Download：抓取网页信息提供给engine，进而转发至Spiders。

Item Pipeline:负责处理Spiders类提取之后的数据。

比如清理HTML数据、验证爬取的数据(检查item包含某些字段)、查重(并丢弃)、将爬取结果保存到数据库中

1. 为什么会选择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，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

1. 描述一下scrapy框架的运行机制？

从start\_urls里面获取第一批url发送请求，请求由请求引擎给调度器入请求对列，获取完毕后，调度器将请求对列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，如果提取出需要的数据，则交给管道处理，如果提取出url，则继续执行之前的步骤，直到多列里没有请求，程序结束。

1. 字典推导式

d = {key:value for (key,value) in iterable}

1. python新式类和经典类的区别？

a. 在python里凡是继承了object的类，都是新式类

b. Python3里只有新式类

c. Python2里面继承object的是新式类，没有写父类的是经典类

d. 经典类目前在Python里基本没有应用

1. 列出5个python标准库？

os：提供了不少与操作系统相关联的函数

sys: 通常用于命令行参数

re: 正则匹配

math: 数学运算

datetime:处理日期时间

1. 谈下python的GIL？

GIL 是python的全局解释器锁，同一进程中假如有多个线程运行，一个线程在运行python程序的时候会霸占python解释器（加了一把锁即GIL），使该进程内的其他线程无法运行，等该线程运行完后其他线程才能运行。如果线程运行过程中遇到耗时操作，则解释器锁解开，使其他线程运行。所以在多线程中，线程的运行仍是有先后顺序的，并不是同时进行。

多进程中因为每个进程都能被系统分配资源，相当于每个进程有了一个python解释器，所以多进程可以实现多个进程的同时运行，缺点是进程系统资源开销大

1. python2和python3的range（100）的区别？

python2返回列表，python3返回迭代器，节约内存

1. 避免转义给字符串加哪个字母表示原始字符串？

r , 表示需要原始字符串，不转义特殊字符

1. 10个Linux常用命令？

ls pwd cd touch rm mkdir tree cp mv cat more grep echo

1. 列出python中可变数据类型和不可变数据类型，并简述原理？

不可变数据类型：数值型、字符串型string和元组tuple

不允许变量的值发生变化，如果改变了变量的值，相当于是新建了一个对象，而对于相 同的值的对象，在内存中则只有一个对象（一个地址）

可变数据类型：列表list和字典dict；

允许变量的值发生变化，即如果对变量进行append、+=等这种操作后，只是改变了变 量的值，而不会新建一个对象，变量引用的对象的地址也不会变化，不过对于相同的值 的不同对象，在内存中则会存在不同的对象，即每个对象都有自己的地址，相当于内存 中对于同值的对象保存了多份，这里不存在引用计数，是实实在在的对象。

1. 用python删除文件和用linux命令删除文件方法？

python：os.remove(文件名)

linux: rm 文件名

1. 正则表达式匹配中，（.\*）和（.\*?）匹配区别？

（.\*）是贪婪匹配，会把满足正则的尽可能多的往后匹配

（.\*?）是非贪婪匹配，会把满足正则的尽可能少匹配

1. 简述Django的orm？

ORM，全拼Object-Relation Mapping，意为对象-关系映射

实现了数据模型与数据库的解耦，通过简单的配置就可以轻松更换数据库，而不需要修 改代码只需要面向对象编程,orm操作本质上会根据对接的数据库引擎，翻译成对应的 sql语句,所有使用Django开发的项目无需关心程序底层使用的是MySQL、Oracle、 sqlite....，如果数据库迁移，只需要更换Django的数据库引擎即可

1. 写5条常用sql语句？

Select \* from 表明;

Show databases;

Show tables;

Desc 表名;

Delete from 表名 where id = 2;

1. 提高python运行效率的方法?

1、使用生成器，因为可以节约大量内存

2、循环代码优化，避免过多重复代码的执行

3、核心模块用Cython PyPy等，提高效率

4、多进程、多线程、协程

5、多个if elif条件判断，可以把最有可能先发生的条件放到前面写，这样可以减少程序 判断的次数，提高效率

1. 讲述redis和mysql数据库的区别？

Redis是属于非关系型数据库，数据保存在内存中，读取速度快

Mysql是关系型数据库，数据存储在本地磁盘中，读取速度相对慢

1. 遇到bug如何处理？

1、细节上的错误，通过print（）打印，能执行到print（）说明一般上面的代码没有问 题，分段检测程序是否有问题，如果是js的话可以alert或console.log

2、如果涉及一些第三方框架，会去查官方文档或者一些技术博客。

3、对于bug的管理与归类总结，一般测试将测试出的bug用teambin等bug管理工具 进行记录，然后我们会一条一条进行修改，修改的过程也是理解业务逻辑和提高自己编 程逻辑缜密性的方法，我也都会收藏做一些笔记记录。

1. 导包问题、城市定位多音字造成的显示错误问题
2. 最主要的是先看报错的内容，找到具体的报错方向最终再去处理这些错误信息
3. 列举一下在网页上常见的状态码并说明一下它的意义？

200 OK

请求正常处理完毕

204 No Content

请求成功处理，没有实体的主体返回

206 Partial Content

GET范围请求已成功处理

301 Moved Permanently

永久重定向，资源已永久分配新URI

302 Found

临时重定向，资源已临时分配新URI

303 See Other

临时重定向，期望使用GET定向获取

304 Not Modified

发送的附带条件请求未满足

307 Temporary Redirect

临时重定向，POST不会变成GET

400 Bad Request

请求报文语法错误或参数错误

401 Unauthorized

需要通过HTTP认证，或认证失败

403 Forbidden

请求资源被拒绝

404 Not Found

无法找到请求资源（服务器无理由拒绝）

500 Internal Server Error

服务器故障或Web应用故障

503 Service Unavailable

服务器超负载或停机维护

1. 分别从前端、后端、数据库阐述web项目的性能优化？

前端优化：

1、减少http请求、例如制作精灵图

2、html和CSS放在页面上部，javascript放在页面下面，因为js加载比HTML和Css加 载慢，所以要优先加载html和css,以防页面显示不全，性能差，也影响用户体验差

精灵图网址：<https://blog.csdn.net/wangwenkai76/article/details/82556642>

后端优化：

1、缓存存储读写次数高，变化少的数据，比如网站首页的信息、商品的信息等。应用 程序读取数据时，一般是先从缓存中读取，如果读取不到或数据已失效，再访问磁盘数 据库，并将数据再次写入缓存。

2、异步方式，如果有耗时操作，可以采用异步，比如celery

3、代码优化，避免循环和判断次数太多，如果多个if else判断，优先判断最有可能先 发生的情况

数据库优化：

1、如有条件，数据可以存放于redis，读取速度快

2、建立索引、外键等

1. 简述cookie和session的区别？

1，session 在服务器端，cookie 在客户端（浏览器）

2、session 的运行依赖 session id，而 session id 是存在 cookie 中的，也就是说，如果 浏览器禁用了 cookie ，同时 session 也会失效，存储Session时，键与Cookie中的 sessionid相同，值是开发人员设置的键值对信息，进行了base64编码，过期时间由开 发人员设置

3、cookie安全性比session差

1. 简述多线程、多进程？

进程：

1、操作系统进行资源分配和调度的基本单位，多个进程之间相互独立

2、稳定性好，如果一个进程崩溃，不影响其他进程，但是进程消耗资源大，开启的进程数量有限制

线程：

1、CPU进行资源分配和调度的基本单位，线程是进程的一部分，是比进程更小的能独立运行的基本单位，一个进程下的多个线程可以共享该进程的所有资源

2、如果IO操作密集，则可以多线程运行效率高，缺点是如果一个线程崩溃，都会造成进程的崩溃

应用：

IO密集的用多线程，在用户输入，sleep 时候，可以切换到其他线程执行，减少等待的时间

CPU密集的用多进程，因为假如IO操作少，用多线程的话，因为线程共享一个全局解释器锁，当前运行的线程会霸占GIL，其他线程没有GIL，就不能充分利用多核CPU的优势

1. 简述any()和all()方法？

any():只要迭代器中有一个元素为真就为真

all():迭代器中所有的判断项返回都是真，结果才为真

1. python字典和json字符串相互转化方法？

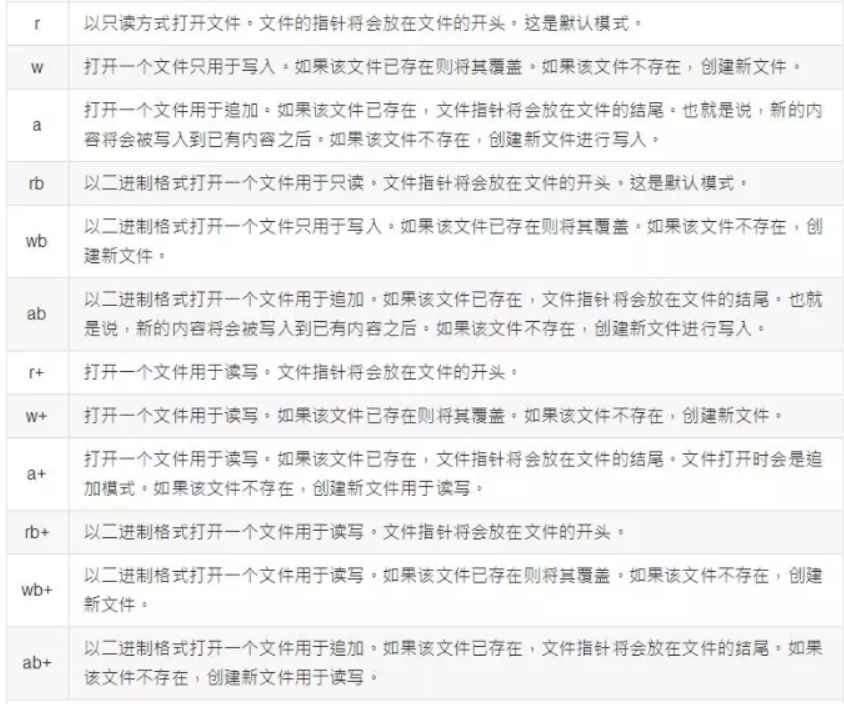
json.dumps()字典转json字符串，json.loads()json转字典

1. 简述乐观锁和悲观锁?

悲观锁, 就是很悲观，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block直到它拿到锁。传统的关系型数据库里 边就用到了很多这种锁机制，比如行锁，表锁等，读锁，写锁等，都是在做操作之前先 上锁。

乐观锁，就是很乐观，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，可以使用版本号等机 制，乐观锁适用于多读的应用类型，这样可以提高吞吐量

1. r、r+、rb、rb+文件打开模式区别?



1. 常见的网络传输协议?

UDP、TCP、FTP、HTTP、SMTP等等

1. 单引号、双引号、三引号用法?

1、单引号和双引号没有什么区别，不过单引号不用按shift，打字稍微快一点。表示字 符串的时候，单引号里面可以用双引号，而不用转义字符,反之亦然。

'She said:"Yes." ' or "She said: 'Yes.' "

2、但是如果直接用单引号扩住单引号，则需要转义，像这样：

' She said:\'Yes.\' '

3、三引号可以直接书写多行，通常用于大段，大篇幅的字符串

"""

hello

world

"""

1. HTTP请求中get和post区别?

1、GET请求是通过URL直接请求数据，数据信息可以在URL中直接看到，比如浏览器 访问；而POST请求是放在请求头中的，我们是无法直接看到的；

2、GET提交有数据大小的限制，一般是不超过1024个字节，而这种说法也不完全准确， HTTP协议并没有设定URL字节长度的上限，而是浏览器做了些处理，所以长度依据浏 览器的不同有所不同；POST请求在HTTP协议中也没有做说明，一般来说是没有设置限 制的，但是实际上浏览器也有默认值。总体来说，少量的数据使用GET，大量的数据使 用POST。

3、GET请求因为数据参数是暴露在URL中的，所以安全性比较低，比如密码是不能暴 露的，就不能使用GET请求；POST请求中，请求参数信息是放在请求头的，所以安全 性较高，可以使用。在实际中，涉及到登录操作的时候，尽量使用HTTPS请求，安全 性更好。

1. python中读取Excel文件的方法?

Import pandas as pd

Df = pd.read\_excel(“xxx.xlsx”)

Print(df)

1. python正则中search和match的区别？

import re

print(re.match("func", "function"))

# 打印结果 <\_sre.SRE\_Match object; span=(0, 4), match='func'>

print(re.match("func", "function").span())

# 打印结果 (0, 4)

print(re.match("func1", "function"))

# 打印结果 None

1. 简述Python的深浅拷贝以及应用场景？

深浅拷贝用法来自copy模块。

导入模块：import copy

浅拷贝：copy.copy

import copy

n1 = {'k1':'wu','k2':123,'k3':['alex',678]}

n3 = copy.copy(n1)

深拷贝：copy.deepcopy

import copy

n1 = {'k1':'wu','k2':123,'k3':['alex',678]}

n4 = copy.deepcopy(n1)

深拷贝的时候python将字典的所有数据在内存中新建了一份，所以如果你修改新的模 版的时候老模版不会变。相反，在浅copy 的时候，python仅仅将最外层的内容在内存 中新建了一份出来，字典第二层的列表并没有在内存中新建，所以你修改了新模版，默 认模版也被修改了

1. 列举常见的内置函数？

max(list) : 求取list最大值。max([1,2,3])

min(list) : 求取list最小值。min([1,2,3])

int(str) : 转换为int型。int('1') >>> 1

float(int/str) : 将int型或字符型转换为浮点型。float('1') >>> 1.0

exec() : 执行python语句。 exec('print("Python")') >>> Python

filter(func, iterable) : 通过判断函数fun，筛选符合条件的元素。 filter(lambda x: x>3, [1,2,3,4,5,6]) >>> <filter object at 0x0000000003813828>

type()：返回一个对象的类型。

id()： 返回一个对象的唯一标识值。

1. 如何安装第三方模块？以及用过哪些第三方模块？

pip install 模块名

一、Python爬虫

1. 请求

requests（第三方模块）

2. 解析：

bs4（即beautifulsoup，第三方模块）

3. 储存：

pymongo（第三方模块）：

　　把数据写入MongoDB

MySQL-python（第三方模块）：

　　把数据写入MySQL里面。

协程：gevent（第三方模块）

1：pip包管理器

2：源码下载

-下载

-解压

-python setup.py build

-python setup.py install

1. Python的优势？

1、Python 易于学习;

2、用少量的代码构建出很多功能;（高效的高级数据结构）

3、Python 拥有最成熟的程序包资源库之一;

4、Python完全支持面向对象;

5、Python 是跨平台且开源的。

6、动态类型

1. 简述解释型和编译型编程语言？

解释型：就是边解释边执行（Python，php）

编译型：编译后再执行（c、java、c#）

1. 三元运算写法和应用场景？

应用场景：简化if语句

# 关于三元运算

# 结果+　if + 条件 + else + 结果

result='gt' if 1>3 else 'lt'

print(result) # lt

# 理解：如果条件为真，把if前面的值赋值给变量，否则把else后面的值赋值给变量。

lambda 表达式

temp = lambda x,y:x+y

print(temp(4,10)) # 14

可替代：

def foo(x,y):

return x+y

print(foo(4,10)) # 14

1. 用一行代码实现数值交换？

a = 1

b = 2

a,b = b,a

1. 列举布尔值为False的常见值？

0，“”，{}，[],（），set（）

0 Flask 负数 不成立的表达式 None 等

1. 字符串、列表、元组、字典每个常用的5个方法？

字符串：

字符串用单引号(')或双引号(")括起来，不可变

1，find通过元素找索引，可切片，找不到返回-1

2，index，找不到报错。

3，split 由字符串分割成列表，默认按空格。

4，captalize 首字母大写，其他字母小写。

5，upper 全大写。

6，lower 全小写。

7，title，每个单词的首字母大写。

8，startswith 判断以什么为开头，可以切片，整体概念。

9，endswith 判断以什么为结尾，可以切片，整体概念。

字典：

1无序（不能索引）2：数据关联性强3:键值对，键值对。唯一一个映射数据类型。

#字典的键必须是可哈希的 不可变类型。

在同一个字典中，键(key)必须是唯一的。

列表是有序的对象集合，字典是无序的对象集合。两者之间的区别在于：字典当中的元 素是通过键来存取的，而不是通过偏移存取

key： 输出所有的键

clear：清空

dic：删除的键如果没有则报错

pop：键值对删，有返回，没有原来的键会报错（自行设置返回键就不会报错）

popitem：随机删键值对

del：删除的键如果没有则报错

改 update

查 用get时。不会报错# 没有可以返回设定的返回值

注意：

1、字典是一种映射类型，它的元素是键值对。

2、字典的关键字必须为不可变类型，且不能重复。

3、创建空字典使用 { }。

列表：

索引，切片，加，乘，检查成员。

增加：有三种，

append：在后面添加。

Insert按照索引添加，

expend：迭代着添加。

list.extend(seq) - 在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）

pop 删除 (pop 有返回值)

remove 可以按照元素去删

clear 清空列表

del 1、可以按照索引去删除 2、切片 3、步长（隔着删）

改 1、索引 2、切片：先删除，再迭代着添加

list.count(obj) - 统计某个元素在列表中出现的次数

list.index(obj) - 从列表中找出某个值第一个匹配项的索引位置

list.reverse() - 反向列表中元素

list.sort([func]) - 对原列表进行排序

注意：

1、List写在方括号之间，元素用逗号隔开。

2、和字符串一样，list可以被索引和切片。

3、List可以使用+操作符进行拼接。

4、List中的元素是可以改变的。

元组：

（）元组的元素不能修改

1、cmp(tuple1, tuple2)：比较两个元组元素。

2、len(tuple)：计算元组元素个数。

3、max(tuple)：返回元组中元素最大值。

4、min(tuple)：返回元组中元素最小值。

5、tuple(seq)：将列表转换为元组。

注意

1、与字符串一样，元组的元素不能修改。

2、元组也可以被索引和切片，方法一样。

3、注意构造包含0或1个元素的元组的特殊语法规则。

4、元组也可以使用+操作符进行拼接。

Set（集合）

：集合（set）是一个无序不重复元素的序列。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而 不是 { }，因为 { } 是用来创建一个空字典。

1. is和==的区别?

a = 'lishi'

str1 = "li"

str2 = "shi"

str3 = str1 + str2

print("a == str3",a == str3)

print("a is str3",a is str3)

print("id(a)",id(a))

print("id(str3)",id(str3))

# a == str3 True == ---> 只需要内容相等

# a is str3 False is ---> 只需要内存地址相等

# id(a) 38565848

# id(str3) 39110280

is 比较的是两个实例对象是不是完全相同，它们是不是同一个对象，占用的内存地址 是否相同。

== 比较的是两个对象的内容是否相等，即内存地址可以不一样，内容一样就可以了。 默认会调用对象的 \_\_eq\_\_()方法。

1. 一行代码实现9\*9乘法表?

print('\n'.join([' '.join(['%s\*%s=%-2s' % (j, i, i \* j) for j in range(1, i + 1)]) for i in range(1, 10)]))

1. 一行代码实现删除列表中重复的值?

list(set([1, 2, 3, 4, 45, 1, 2, 343, 2, 2]))

1. 如何用一行代码生成[1,4,9,16,25,36,49,64,81,100]?

[i\*i for i in range(1,11)]

推导式

1. 如何在函数中设置一个全局变量？

x = 2

def func():

global x

x = 1

return x

func()

print(x) # 1

1. drf序列化器与反序列化器？

**序列化：**

将对象的状态信息转换为可以存储或传输的形式的过程。（百度定义）

对应到drf中,序列化即把模型对象转换为字典形式, 再返回给前端,主要用于输出 序列化返回

序列化器，就是对数据进行校验、对数据对象进行转化

将Django中的模型类对象装换为JSON字符串，这个转换过程我们称为序列化。

**反序列化：**

把其他格式转化为程序中的格式。

对应drf中接收前段数据经过验证转为Python中的字典，主要用于输入，接收前段数据

**反序列化存储**

使用序列化器进行反序列化时，需要对数据进行验证后，才能获取验证成功的数据或保存成模型类对象。

在获取反序列化的数据前，必须调用is\_valid()方法进行验证，验证成功返回True，否则返回False。

将JSON字符串转换为Django中的模型类对象，这个过程我们称为反序列化。

loads

1. jwt认证的特点和使用？

1）jwt token由服务器生成，交给客户端进行存储，不占用服务器的存储空间；

​ 2）适合于分布式站点的应用场景。

​ 3）使用时payload中不要存储敏感信息，因为这部分内容很容易被客户端解码。

​ 4）服务器端的密钥需要保存好，泄露密钥之后jwt token数据可以被随意伪造。

我们保存jwt,token在浏览器本地存储中,返回给前端，前端保存token。私钥是django 项目一创建出来就出现在配置文件里的）再用base64编码，构成最终的jwt.

1. 怎么实现跨域？

安装

pip install django-cors-headers

添加应用

INSTALLED\_APPS = (

…

‘corsheaders’,

…

)

中间层设置

MIDDLEWARE = [

‘corsheaders.middleware.CorsMiddleware’,

…

]

添加白名单

# CORS

CORS\_ORIGIN\_WHITELIST = (

‘127.0.0.1:8080’,

‘localhost:8080’,

‘www.meiduo.site:8080’,

‘api.meiduo.site:8000’

)

CORS\_ALLOW\_CREDENTIALS = True # 允许携带cookie

1. docker你用的挺熟的嘛？

docker的话一般提起来都喜欢和虚拟机联系起来，提起docker的话，我脑海里有3个 关键字，就是镜像、容器、仓库，就是它会把一些镜像在虚拟机里运作起来，那么它就 是一个容器。为什么说它是一个仓库呢，就是它负责管理存储这些镜像文件。然后docker 它有一个很大的好处就是，对于我们开发人员来说，这个代码在我这里能跑，但是在运 维那里就跑不了，然后运维就说你的代码有问题，但是再试的话就又能跑了，没有问题。 但是我们用了docker的话就不会出现这种问题，因为它的环境是一样的。所以运维也 蛮喜欢用这个的，因为说用docker的话它会把你的运行程序隔离起来，就不会有影响。 它是用go写的。

1. flask与Django有什么区别？

个人更倾向flask，因为它更方便，更灵活，用什么就装什么。不想django那样，都放 在那里。就好比你买辆车吧，django就是什么车胎发动机都给你定下来装好了，而flask 的话我可以改改发动机呀改改轮胎呀什么的，根据个人的需求来定制化操作。flask更 轻便。

1. dumps与loads？

项目接口里用到

dumps:将dict（字典）转换为str（字符串）。

loads:用于将str（字符串）转换为dict（字典）

1. 说出git常用命令并说出该命令的作用？

作用：对源代码进行分布式,版本控制，方便跟踪源代码的修改过程，备份源代码,在团 队开发中经常使用

常用命令：

i.git init 初始化本地git仓库

ii.git add . 将当前工作区的所有文件添加到暂存区

iii.git commit -m “”将当前暂存区文件添加到本地仓库进行版本管理

iv.git push 将本地仓库内容添加到远程仓库中

v.git pull 将远程仓库代码拉取到本地

vi.git clone 拷贝远程仓库内容到本地

**可以使用Sourcetree软件代替命令**

1. nginx的使用？

Nginx

采用 C 语言编写

实现分流、转发、负载均衡

服务器

WSGI（Web Server Gateway Interface），翻译为Python web服务器网关接口，即Python 的Web应用程序（如Flask）和Web服务器(如Nginx)之间的一种通信协议。也就是说， 如果让你的Web应用在任何服务器上运行，就必须遵循这个协议。

那么实现WSGI协议的web服务器有哪些呢？

就比如uWSGI与gunicorn。两者都可以作为Web服务器。可能你在许多地方看到的都 是采用Nginx + uWSGI（或gunicorn）的部署方式。实际上，直接通过uWSGI或gunicorn 直接部署也是可以让外网访问的，那你可能会说，那要Nginx何用？那么接来下介绍另 一个Web服务器——Nginx

Nginx作为一个高性能Web服务器，具有负载均衡、拦截静态请求、高并发…等等许多 功能，你可能要问了，这些功能和使用Nginx + WSGI容器的部署方式有什么关系？

首先是负载均衡，如果你了解过OSI模型的话，其实负载均衡器就是该模型中4~7层交 换机中的一种，它的作用是能够仅通过一个前端唯一的URL访问分发到后台的多个服 务器，这对于并发量非常大的企业级Web站点非常有效。在实际应用中我们通常会让 Nginx监听（绑定）80端口，通过多域名或者多个location分发到不同的后端应用。

其次是拦截静态请求，简单来说，Nginx会拦截到静态请求（静态文件，如图片），并 交给自己处理。而动态请求内容将会通过WSGI容器交给Web应用处理;

Nginx还有其他很多的功能，这里便不一一介绍。那么前面说了，直接通过uWSGI或 gunicorn也可以让外网访问到的，但是鉴于Nginx具有高性能、高并发、静态文件缓存、 及以上两点、甚至还可以做到限流与访问控制，所以选择Nginx是很有必要的

这里可以说明，如果你选择的架构是：Nginx + WSGI容器 + web应用，WSGI容器相当 于一个中间件；如果选择的架构是uWSGI + web应用，WSGI容器则为一个web服务器

1. 什么是wsgi？

WSGI是Python在处理HTTP请求时，规定的一种处理方式。如一个HTTP Request过来 了，那么就有一个相应的处理函数来进行处理和返回结果。WSGI就是规定这个处理函 数的参数长啥样的，它的返回结果是长啥样的？至于该处理函数的名子和处理逻辑是啥 样的，那无所谓。简单而言，WSGI就是规定了处理函数的输入和输出格式。

1. 列举django的内置组件？

.Admin是对model中对应的数据表进行增删改查提供的组件

.model组件：负责操作数据库

.form组件：1.生成HTML代码2.数据有效性校验3校验信息返回并展示

.ModelForm组件即用于数据库操作,也可用于用户请求的验证

1. 如何在一个function里设置一个全局的变量？

先说概念，全局变量是指定义在函数外部的变量。全局变量的作用域为全局。

局部变量是指定义在函数内部的变量。局部变量的作用域为函数内，除了函数就无效了。

这里举个例子，如果把函数比作国家，那么全局就是全球，全局变量好比是阿拉伯数字， 每个国家都认识。

1. 简单描述一下Django 、Flask、Tornado三个框架？

1.Django走的是大而全的方向,开发效率高。它的MTV框架,自带的ORM,admin后台管 理,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

2.Flask是轻量级的框架,自由,灵活,可扩展性很强,核心基于Werkzeug WSGI工具和jinja2 模板引擎

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

Tornado的两大核心模块：

1.iostraem：对非阻塞式的socket进行简单的封装

2.ioloop：对I/O多路复用的封装，它实现了一个单例

1. 谈谈你对restful规范的认识？

描述：首先restful是一种软件架构风格或者说是一种设计风格，并不是标准，它只是 提供了一组设计原则和约束条件，主要用于客户端和服务器交互类的软件。

就像设计模式一样，并不是一定要遵循这些原则，而是基于这个风格设计的软件可以更 简洁，更有层次，我们可以根据开发的实际情况，做相应的改变。

1、restful 提倡面向资源编程,在url接口中尽量要使用名词，不要使用动词

2、在url接口中推荐使用Https协议，让网络接口更加安全 <https://baidum/v1/mycss？> page=3

（Https是Http的安全版，即HTTP下加入SSL层，HTTPS的安全基础是SSL， 因此加 密的详细内容就需要SSL（安全套接层协议））

3、在url中可以体现版本号 https://v1.bootcss.com/mycss 不同的版本可以有不同的接 口，使其更加简洁，清晰

4、url中可以体现是否是API接口 https://baidum/api/mycss

5、url中可以添加条件去筛选匹配

https://baidum/v1/mycss？page=3

6、可以根据Http不同的method，进行不同的资源操作 （5种方法：GET

/ POST / PUT / DELETE / PATCH）

7、响应式应该设置状态码

8、有返回值，而且格式为统一的json格式

9、返回错误信息

10、返回结果中要提供帮助链接，即API最好做到接口文档

1. 简单描述一下支付宝支付流程？

第一、发方首先有一个公钥/私钥对，它将要签名的报文作为一个单向散列函数的输入， 产生一个定长的散列码，一般称为消息摘要。

第二、使用发放的私钥对散列码进行加密生成签名。将报文和签名一同发出去。

第三、收方用和发放一样的散列函数对报文运算生成一个散列码，同时用发放的公钥对 签名进行解密。

第四、如果收方计算得到的散列码和解密的签名一致，那么说明的确是发方对报文进行 了签名而且报文在途中没有被篡改。

1. celery？

Celery是由Python开发、简单、灵活、可靠的分布式任务队列，其本质是生产者消费 者模型，生产者发送任务到消息队列，消费者负责处理任务。Celery侧重于实时操作， 但对调度支持也很好，其每天可以处理数以百万计的任务

特点：简单：熟悉celery的工作流程后，配置使用简单

高可用：当任务执行失败或执行过程中发生连接中断，celery会自动尝试重新执行任务

快速：一个单进程的celery每分钟可处理上百万个任务

灵活：几乎celery的各个组件都可以被扩展及自定制

工作原理：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 中被触发并发往消息队列，而定时任务由Celery Beat进程周期性地将任务发往消息队 列；任务执行单元Worker实时监视消息队列获取队列中的任务执行；Woker执行完任 务后将结果保存在Backend中。

1. 下列哪个语句在Python中是非法的？

A、x = y = z =1 B、x = (y = z + 1)

1. x, y = y, x D、x += y

答案：B

1. 关于Python内存管理，下列说法错误的是？

A、变量不必事先声明 B、变量无须先创建和赋值而直接使用

1. 变量无须指定类型 D、可以使用del释放资源

答案：B. (不先赋值会报错，is not defined)

1. 下面哪个不是Python合法的标识符？
2. int32 B、40XL C、self D、name

答案：B（合法的标识符不能以数字开头）

1. 下列哪种说法是错误的？

A、除字典类型外，所有标准对象均可以用于布尔测试

B、空字符串的布尔值是False

C、空列表对象的布尔值是False

D、值为0的任何数字对象的布尔值是False

答案：D （a=0,0, bool(a)为True）

1. 下列表达式的值为True的是?

A、5+4j >2-3j B、3>2>2

1. (3,2)<('a','b'） D、’abc’ > ‘xyz’

答案：C (在Py2.x版本中正确，在Py3.x运行错误)

1. Python不支持的数据类型有?

A、char B、int C、float D、list

答案：A（python里无char型数据，有string字符串类型;但C语言中有char数据类型）

1. 关于Python中的复数，下列说法错误的是?

A、表示复数的语法是real + imagej B、实部和虚部都是浮点数

C、虚部必须后缀j，且必须是小写 D、方法conjugate返回复数的共轭复数

答案：C（复数虚部的后缀也可以是大写的J）

1. 关于字符串下列说法错误的是?

A、字符应该视为长度为1的字符串

B、字符串以\0标志字符串的结束

C、既可以用单引号，也可以用双引号创建字符串

D、在三引号字符串中可以包含换行回车等特殊字符

答案：B（python因为字符串有长度限制，到了长度就标志字符串的结束）

1. 以下不能创建一个字典的语句是?

A、dict1 = {}                       B、dict2 = { 3 : 5 }

C、dict3 ={[1,2,3]: “uestc”}         D、dict4 = {(1,2,3): “uestc”}

答案：C（字典的键必须是不变的，而列表是可变的）

1. 下列Python语句正确的是?

A、min = x if x< y = y

B、max = x > y ?x:y

C、if (x >y) print x

D、while True :pass

答案：D

1. 用一条SQL语句 查询出每门课都大于80分的学生姓名。表scores如下:

name kecheng fenshu

张三 语文 81

张三 数学 75

李四 语文 76

答案：SELECT DISTINCT name FROM grade WHERE name NOT IN(SELECT DISTINCTname FROM grade WHERE score <=80);

更简单的：SELECT name FROMgrade GROUP BY name HAVING MIN(score) > 80;

1. python常见的命令行交互自动化模块有哪些？

答案：

a)Import module

b) Import module1,module2

c) From module import \*

d) Frommodule import m1,m2,m3

e) From module import logger asmodule\_logger

1. python的底层网络交互模块有哪些？

答案：socket, urllib,urllib3 , requests, grab, pycurl

1. python网络交互时，二进制打包解包的模块有哪些?

答案：打包：pack(), 解包：upk()

1. python的测试框架有哪些？试列出常用的3个或更多?

答案：unittest, nose,unittest2, doctest, pytest

1. 一行把[1,2,3,1,2,3] 中的重复元素剔除?

答案：List(set([1,2,3,1,2,3]))

1. 现在要你使用pyDes(DES加密）和标准库中的namedtuple，假设你之前没有接触过，你如何快速上手？

答案：仔细阅读官方文档中namedtuple库和pyDes的使用。下载pyDes和namedtuple 库，借鉴网上的使用教程，摸索实践就OK了

1. 如何在Windows上安装Python并设置路径变量？

从以下链接安装python：https：//http://www.python.org/downloads/

下载之后，将其安装在您的PC上。在命令提示符下使用以下命令查找PC上安装PYTHON 的位置：cmd python。

然后转到高级系统设置并添加新变量并将其命名为PYTHON\_NAME并粘贴复制的路径。

查找路径变量，选择其值并选择“编辑”。

如果值不存在，请在值的末尾添加分号，然后键入％PYTHON\_HOME％

1. Python中的self是什么？

self是类的实例或对象。在Python中，self包含在第一个参数中。但是，Java中的情况 并非如此，它是可选的。它有助于区分具有局部变量的类的方法和属性。init方法中的 self变量引用新创建的对象，而在其他方法中，它引用其方法被调用的对象。

1. Python中help()和dir()函数的用法是什么？

Help()和dir()这两个函数都可以从Python解释器直接访问，并用于查看内置函数的合并 转储。

help()函数：help()函数用于显示文档字符串，还可以查看与模块，关键字，属性等相关 的使用信息。

dir()函数：dir()函数用于显示定义的符号。

1. 当Python退出时，为什么不清除所有分配的内存？

当Python退出时，尤其是那些对其他对象具有循环引用的Python模块或者从全局名称 空间引用的对象并没有被解除分配或释放。

无法解除分配C库保留的那些内存部分。

退出时，由于拥有自己的高效清理机制，Python会尝试取消分配/销毁其他所有对象。

1. 如何在python中使用三元运算符？

三元运算符是用于显示条件语句的运算符。这包含true或false值，并且必须为其评估 语句。其基本语法为：

三元运算符是用于显示条件语句的运算符。这包含true或false值，并且必须为其评估 语句。其基本语法为：

[on\_true] if [expression] else [on\_false] x，y = 25,50big = x if x <y else y

1. 在Python中split()，sub()，subn()功能？

如果要修改字符串，Python的“re”模块提供了3种方法

split() - 使用正则表达式模式将给定字符串“拆分”到列表中。

sub() - 查找正则表达式模式匹配的所有子字符串，然后用不同的字符串替换它们

subn() - 它类似于sub()，并且还返回新字符串。

1. 如何在Python中删除文件？

要在Python中删除文件，就需要导入模块OS，然后os.remove(需要删除的文件地址)

1. NumPy中有哪些操作Python列表的函数？

NumPy不仅效率更高; 它也更方便。你可以免费获得大量的向量和矩阵运算，这有时可 以避免不必要的工作。它们也得到有效实施。

NumPy数组更快，你可以使用NumPy，FFT，卷积，快速搜索，基本统计，线性代数， 直方图等内置。

1. Ajax的使用步骤？

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

如何使用ajax?

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。

var xhttp;

if (window.XMLHttpRequest) {

//现代主流浏览器

xhttp = new XMLHttpRequest();

} else {

// 针对浏览器，比如IE5或IE6

xhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

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。

1. undefined和null的区别？

null： Null类型，代表“空值”，代表一个空对象指针，使用typeof运算得到 “object”，

所以你可以认为它是一个特殊的对象值。

undefined： Undefined类型，当一个声明了一个变量未初始化时，得到的就是undefined。

null是javascript的关键字，可以认为是对象类型，它是一个空对象指针，

和其它语言一样都是代表“空值”，不过 undefined 却是javascript才有的。

undefined是在ECMAScript第三版引入的，为了区分空指针对象和未初始化的变量，

它是一个预定义的全局变量。没有返回值的函数返回为undefined，

没有实参的形参也是undefined。

1. 如何进行网站性能优化？

目的： 从用户角度而言，优化能够让页面加载得更快、对用户的操作响应得更及时， 能够给用户提供更为友好的体验。从服务商角度而言，优化能够减少页面请求数、或者 减小请求所占带宽，能够节省可观的资源。

措施：

减少HTTP请求数。

使用缓存。

脚本的无阻塞加载。

内联脚本的位置优化等。

Javascript中的DOM 操作优化、CSS选择符优化。

图片编码优化，懒加载。

使用负载均衡方案。

1. 什么是mvvm，mvc？

MVC是比较直观的架构模式，用户操作

->View（负责接收用户的输入操作）

->Controller（业务逻辑处理）->Model（数据持久化）->View（将结果反馈给View）。

MVC使用非常广泛，比如JavaEE中的SSH框架。

如果说MVP是对MVC的进一步改进，那么MVVM则是思想的完全变革。

它是将“数据模型数据双向绑定”的思想作为核心，因此在View和Model之间没有联 系。

通过ViewModel进行交互，而且Model和ViewModel之间的交互是双向的。

因此视图的数据的变化会同时修改数据源，而数据源数据的变化也会立即反应view。

微信小程序前端使用mvvm

1. 从输入url到显示页面，都经历了什么？简单的描述一下

1、首先，在浏览器地址栏中输入url。

2、浏览器先查看浏览器缓存-系统缓存-路由器缓存，如果缓存中有，会直接在屏幕中 显示页面内容。若没有，则跳到第三步操作。

3、在发送http请求前，需要域名解析(DNS解析)(DNS（域名系统，Domain Name System） 是互联网的一项核心服务，它作为可以将域名和IP地址相互映射的一个分布式数据库， 能够使人更方便的访问互联网，而不用去记住IP地址。)，解析获取相应的IP地址。

4、浏览器向服务器发起tcp连接，与浏览器建立tcp三次握手。（TCP即传输控制协议。 TCP连接是互联网连接协议集的一种。）

5、握手成功后，浏览器向服务器发送http请求，请求数据包。

6、服务器处理收到的请求，将数据返回至浏览器。

7、浏览器收到HTTP响应。

8、读取页面内容，浏览器渲染，解析html源码。

9、生成Dom树、解析css样式、js交互。

10、客户端和服务器交互。

11、ajax查询。

1. $(document).ready()方法和window.onload有什么区别？

(1)、window.onload方法是在网页中所有的元素(包括元素的所有关联文件)完全加载到 浏览器后才执行的。

(2)、$(document).ready() 方法可以在DOM载入就绪时就对其进行操纵，并调用执行 绑定的函数。

1. 字符串、字典、元组、列表常用方法？

字符串常用的是split、replace、join、find、strip、just

列表常用 pop 、append、 remove、 insert 、clear 、len 、sort、 reverse

字典常用 get、 index、 keys、 values、 update

1. 数据库的读写分离？

读写分离，基本的原理是让主数据库处理事务性增、改、删操作（INSERT、UPDATE、 DELETE），而从数据库处理SELECT查询操作。数据库复制被用来把事务性操作导致的 变更同步到集群中的从数据库。

1. 面向对象优缺点？

解决了程序的扩展性，对某个对象单独修改 会立刻反映到整个体系，在调用时只需要 实例化一个对象，对象就会拥有所有的属性

缺点是可控性差，无法像面向过程的程序设计一样精准的预测问题的处理结果

1. 面向对象继承时要注意什么？

继承是一种创建新类的方式，在python中，新建的类可以继承一个或多个父类

继承分单继承和多继承 多继承需要注意继承的顺序，

而且子类继承父类后如果方法命名与父类相同 则会覆盖

但在python3中。可以使用super函数来调用父类方法

1. 什么是数据库锁？

由于数据库是多用户共享资源，所以需要处理并发问题，而数据库锁的机制就是为了处 理这一问题，当出现并发的时候，如果不做控制就会出现各种问题 比如脏数据。修改 丢失等 所有数据库并发需要事务来控制，事务并发问题需要数据库锁来控制

事务四个特性。持久型 原子性 一致性 隔离性

数据库锁有 乐观锁 悲观锁 死锁。活锁 行锁 表锁 页级锁

排它锁有称为写锁。共享锁又称为读锁

1. 谈谈你知道的对于数据库的优化方案？

尽量避免使用select \* 能用字段名就用字段名。避免查询无用字段

select count(\*)会查全表 尽量避免

建表时字段类型能用varchar/nvarchar就不要用char/ncahr

避免频繁的创建和删除临时表 会耗费性能资源 产生大量log

如果使用临时表 在使用的最后一定要显示删除 先trancate table 再drop table

尽量避免大事务操作，提高并发效率

避免向客户端返回大数据量 数据量过大 应考虑需求是否合理

比如你在一个在线网站使用delete和update操作。必然会引发数据库锁

1. 简单描述一下wsgi的工作原理？

请求：

请求方法

请求地址

请求内容

请求头

请求环境

响应：

状态吗

响应数据

响应头

wsgi的作用就是将上面数据在server端和python程序间进行传递 它是一个标准 一个需要遵从的规范 才能正常工作

1. 说说你对Django框架的理解？

Django遵从mvc框架

Django是由python编写的开源WEB框架

目录urls.py 路由分发 将url与view一一映射

wsgi.py 关于程序与服务器端的规范 或者说统一的接口

1. Django 中间件的作用是什么？

中间件是服务器端与应用程序的一个中间层，它将个管道一样。将接受到的请求进行一 些处理。然后传递到客户端 然后把客户端处理的结果再返回

它的应用场景是

根据url把请求给到不同的客户端程序

允许多个客户端

负载均衡和远程处理

应答的过滤处理

1. 协程的优缺点是什么？

优点：

　　上下文切换消耗小

　　方便切换控制流，简化编程模型

　　高并发，高扩展性，低成本

缺点：

　　无法利用多核

　　进行阻塞操作时会阻塞掉整个程序

　　单纯的协程是没有意义的，只是人为的控制执行一下这个，执行一下那个，如果想 监测是否有IO操作，需要结合IO多路复用

1. 进程与线程之间的关系？

（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

1. 什么是rest api？

RESTful API由后台也就是SERVER来提供前端来调用。前端调用API向后台发起HTTP请 求，后台响应请求将处理结果反馈给前端。也就是说RESTful 是典型的基于HTTP的协 议。

1，资源。首先是弄清楚资源的概念。资源就是网络上的一个实体，一段文本，一张图 片或者一首歌曲。资源总是要通过一种载体来反应它的内容。文本可以用TXT，也可以 用HTML或者XML、图片可以用JPG格式或者PNG格式，JSON是现在最常用的资源 表现形式。

     2，统一接口。RESTful风格的数据元操CRUD（create,read,update,delete）分别对应 HTTP方法：GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用 来更新资源，DELETE用来删除资源，这样就统一了数据操作的接口。

     3，URI。可以用一个URI（统一资源定位符）指向资源，即每个URI都对应一个特定 的资源。要获取这个资源访问它的URI就可以，因此URI就成了每一个资源的地址或识 别符。一般的，每个资源至少有一个URI与之对应，最典型的URI就是URL。

     4，无状态。所谓无状态即所有的资源都可以URI定位，而且这个定位与其他资源无 关，也不会因为其他资源的变化而变化。有状态和无状态的区别，举个例子说明一下， 例如要查询员工工资的步骤为第一步：登录系统。第二步：进入查询工资的页面。第三 步：搜索该员工。第四步：点击姓名查看工资。这样的操作流程就是有状态的，查询工 资的每一个步骤都依赖于前一个步骤，只要前置操作不成功，后续操作就无法执行。如 果输入一个URL就可以得到指定员工的工资，则这种情况就是无状态的，因为获取工 资不依赖于其他资源或状态，且这种情况下，员工工资是一个资源，由一个URL与之 对应可以通过HTTP中的GET方法得到资源，这就是典型的RESTful风格。

1. 说说你对Python这门语言的看法？

单说python这门语言的话我们首先得了解它的特性与优势：python是一门具有解释性、 编译性、互动性和面向对象的脚本语言

python有很强的可读性，对初学者十分友好

因为开源，所以也有很丰富的扩展库、与Linux Unix Windows兼容良好,可移植性好，现 在像mac终端已经自带python了

python与php相比可以处理多进程多线程任务

语法没有java那么严格 java的可移植性是需要代价的

1. Docker有什么好处？

<https://www.cnblogs.com/lys_013/p/5682153.html>

参考网址

1. python下多线程的限制以及多进程中传递参数的方式？

python多线程有个全局解释器锁（global interpreter lock），这个锁的意思是任一时间 只能有一个线程使用解释器，跟单cpu跑多个程序一个意思，大家都是轮着用的，这叫 “并发”，不是“并行”。

多进程间共享数据，可以使用 multiprocessing.Value 和 multiprocessing.Array

1. 分别从前端、后端、数据库阐述web项目的性能优化？

前端优化：

1、减少http请求、例如制作精灵图

2、html和CSS放在页面上部，javascript放在页面下面，因为js加载比HTML和Css加 载慢，所以要优先加载html和css,以防页面显示不全，性能差，也影响用户体验差

后端优化：

1、缓存存储读写次数高，变化少的数据，比如网站首页的信息、商品的信息等。应用 程序读取数据时，一般是先从缓存中读取，如果读取不到或数据已失效，再访问磁盘数 据库，并将数据再次写入缓存。

2、异步方式，如果有耗时操作，可以采用异步，比如celery

3、代码优化，避免循环和判断次数太多，如果多个if else判断，优先判断最有可能先 发生的情况

数据库优化：

1、如有条件，数据可以存放于redis，读取速度快

2、建立索引、外键等

1. 简述同源策略？

同源策略需要同时满足以下三点要求：

1）协议相同

 2）域名相同

3）端口相同

 http:www.test.com与https:www.test.com 不同源——协议不同

 http:www.test.com与http:www.admin.com 不同源——域名不同

 http:www.test.com与http:www.test.com:8081 不同源——端口不同

 只要不满足其中任意一个要求，就不符合同源策略，就会出现“跨域”

1. 简述cookie和session的区别？

1、session 在服务器端，cookie 在客户端（浏览器）

2、session 的运行依赖 session id，而 session id 是存在 cookie 中的，也就是说，如果浏览器禁用了 cookie ，同时 session 也会失效，存储Session时，键与Cookie中的sessionid相同，值是开发人员设置的键值对信息，进行了base64编码，过期时间由开发人员设置

3、cookie安全性比session差

1. IOError、AttributeError、ImportError、IndentationError、IndexError、KeyError、SyntaxError、NameError分别代表什么异常？

IOError：输入输出异常

AttributeError：试图访问一个对象没有的属性

ImportError：无法引入模块或包，基本是路径问题

IndentationError：语法错误，代码没有正确的对齐

IndexError：下标索引超出序列边界

KeyError:试图访问你字典里不存在的键

SyntaxError:Python代码逻辑语法出错，不能执行

NameError:使用一个还未赋予对象的变量

1. 列出几种魔法方法并简要介绍用途？

\_\_init\_\_:对象初始化方法

\_\_new\_\_:创建对象时候执行的方法，单列模式会用到

\_\_str\_\_:当使用print输出对象的时候，只要自己定义了\_\_str\_\_(self)方法，那么就会打印 从在这个方法中return的数据

\_\_del\_\_:删除对象执行的方法

1. MyISAM 与 InnoDB 区别？

1、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；

2、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；

3、InnoDB 支持外键，MyISAM 不支持；

4、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；

5、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；

1. int("1.4"),int(1.4)输出的结果？

int("1.4")报错，int(1.4)输出1

1. 列举3条以上PEP8编码规范？

1、顶级定义之间空两行，比如函数或者类定义。

2、方法定义、类定义与第一个方法之间，都应该空一行

3、三引号进行注释

4、使用Pycharm、Eclipse一般使用4个空格来缩进代码

1. r、r+、rb、rb+文件打开模式的区别分别是什么?



1. Linux命令重定向 > 和 >>?

Linux 允许将命令执行结果 重定向到一个 文件

将本应显示在终端上的内容 输出／追加 到指定文件中

> 表示输出，会覆盖文件原有的内容

>> 表示追加，会将内容追加到已有文件的末尾

用法示例：

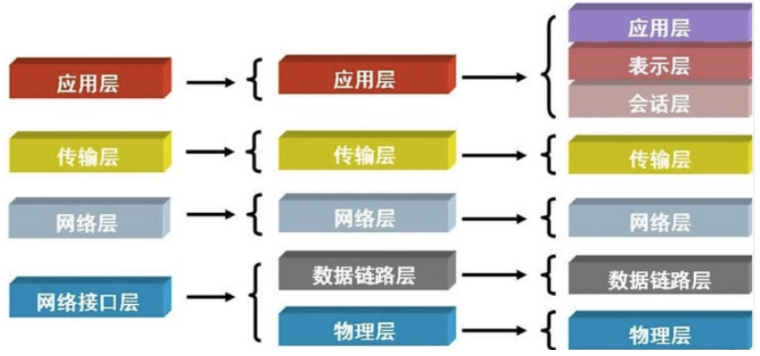
将 echo 输出的信息保存到 1.txt 里echo Hello Python > 1.txt

将 tree 输出的信息追加到 1.txt 文件的末尾tree >> 1.txt

1. 说说常见的网络传输协议都有哪些？

UDP、TCP、FTP、HTTP、SMTP等等。

1. 简述一下 OSI 七层协议？



物理层：主要是基于电器特性发送高低电压（电信号），高电压对应数字1，低电压对 应数字0。

数据链路层：定义了电信号的分组方式。

网路层：引入一套新的地址用来区分不同的广播域/子网，这套地址即网络地址。

传输层：建立端口到端口的通信。

会话层：建立客户端与服务端连接。

表示层：对来自应用层的命令和数据进行解释，按照一定格式传给会话层。如编码、数 据格式转换、加密解密、压缩解压。

应用层：规定应用程序的数据格式。

1. TCP/UDP区别？

TCP协议是面向连接，保证高可靠性传输层协议。

UDP：数据丢失，无秩序的传输层协议（qq基于udp协议）。

1. 什么是防火墙？及它的作用是什么？

防火墙是一个分离器、一个限制器，也是一个分析器，有效地监控了内部网和Internet 之间的任何活动，保证了内部网络的安全。

作用：

防火墙是网络安全的屏障。

可以强化网络安全策略。

对网络存取和访问进行监控审计。

防止内部信息的外泄。

除了安全作用，防火墙还支持具有Internet服务特性的企业内部网络技术体系VPN （虚拟专用网）。

1. 什么是并发和并行？

并发：同一时刻只能处理一个任务，但一个时段内可以对多个任务进行交替处理(一个 处理器同时处理多个任务)。

并行：同一时刻可以处理多个任务(多个处理器或者是多核的处理器同时处理多个不同 的任务)。

类比：并发是一个人同时吃三个馒头，而并行是三个人同时吃三个馒头。

1. 解释什么是异步非阻塞？

非阻塞：不等待。

即：遇到IO阻塞不等待(setblooking=False),（可能会报错->捕捉异常）

- sk=socket.socket()

- sk.setblooking(False)

异步：回调，当达到某个指定的状态之后，自动调用特定函数。

实例：

nb\_async.py 实现异步非阻塞的模块。

1. 路由器和交换机的区别？

1：交换机：是负责内网里面的数据传递（arp协议）根据MAC地址寻址

路由器：在网络层，路由器根据路由表，寻找该ip的网段。

2：路由器可以处理TCP/IP协议。

3：路由器可以把一个IP分配给很多个主机使用，这些主机对外只表现出一个IP。

交换机可以把很多主机连起来，这些主机对外各有各的IP。

4：交换机是做端口扩展的，也就是让局域网可以连进来更多的电脑。

路由器是用来做网络连接，也就是；连接不同的网络。

1. 什么是域名解析？

在互联网上，所有的地址都是ip地址，现阶段主要是IPv4（比如：110.110.110.110）。

但是这些ip地址太难记了，所以就出现了域名（比如http://baidu.com）。

域名解析就是将域名，转换为ip地址的这样一种行为。

1. 如何修改本地hosts件？

Hosts是一个没有扩展名的系统文件，可以用记事本等工具打开，其作用就是将一些常 用的网址域名与其对应的IP地址建立一个关联“数据库”，

当用户在浏览器中输入一个需要登录的网址时，系统会首先自动从Hosts文件中寻找对 应的IP地址，

一旦找到，系统会立即打开对应网页，如果没有找到，则系统会再将网址提交DNS域 名解析服务器进行IP地址的解析。

文件路径：C:\WINDOWS\system32\drivers\etc。

将127.0.0.1 www.163.com 添加在最下面

修改后用浏览器访问“www.163.com”会被解析到127.0.0.1，导致无法显示该网页。

1. 生产者消费者模型应用场景及优势？

应用场景：用户提交订单，订单进入引擎的阻塞队列中，由专门的线程从阻塞队列中获 取数据并处理。

优势：

1；解耦

假设生产者和消费者分别是两个类。如果让生产者直接调用消费者的某个方法，那么生 产者对于消费者就会产生依赖（也就是耦合）。

将来如果消费者的代码发生变化，可能会影响到生产者。而如果两者都依赖于某个缓冲 区，两者之间不直接依赖，耦合也就相应降低了。

2：支持并发

生产者直接调用消费者的某个方法，还有另一个弊端。由于函数调用是同步的（或者叫 阻塞的），在消费者的方法没有返回之前，生产者只能一直等着

而使用这个模型，生产者把制造出来的数据只需要放在缓冲区即可，不需要等待消费者 来取。

3：支持忙闲不均

缓冲区还有另一个好处。如果制造数据的速度时快时慢，缓冲区的好处就体现出来了。

当数据制造快的时候，消费者来不及处理，未处理的数据可以暂时存在缓冲区中。等生 产者的制造速度慢下来，消费者再慢慢处理掉。

1. 什么是cdn？

目的是使用户可以就近到服务器取得所需内容，解决 Internet网络拥挤的状况，提高用 户访问网站的响应速度。

cdn 即内容分发网络。

1. LVS是什么及它的作用？

LVS ：Linux虚拟服务器

作用：

LVS主要用于多服务器的负载均衡。

它工作在网络层，可以实现高性能，高可用的服务器集群技术。

它廉价，可把许多低性能的服务器组合在一起形成一个超级服务器。

它易用，配置非常简单，且有多种负载均衡的方法。

它稳定可靠，即使在集群的服务器中某台服务器无法正常工作，也不影响整体效果。另 外可扩展性也非常好。

1. 列举常见的关系型数据库和非关系型都有那些？

关系型数据库(需要有表结构)

mysql、oracle 、 spl、server、db2、sybase

非关系型数据库（是以key-value存储的，没有表结构）（NoSQL）

MongoDB

MongoDB 是一个高性能，开源，无模式的文档型数据库，开发语言是C++。它在许多 场景下可用于替代传统的关系型数据库或键/值存储方式。

Redis

Redis 是一个开源的使用ANSI C语言编写、支持网络、可基于内存亦可持久化的日志型、 Key-Value数据库，并提供多种语言的API。

1. 简述数据库三大范式？

数据库的三大特性：

'实体':表

'属性'：表中的数据(字段)

'关系'：表与表之间的关系

数据库设计三大范式：

1：确保每列保持原子性（即数据库表中的所有字段值是不可分解的原子值）

2：确保表中的每列都是和主键相关（表中只能保存一种数据，不可以把多种数据保存 在同一张表中）--->完全属于当前表的数据

3：确保每列都和主键直接相关，而不是间接相关（在一个数据库表中保存的数据只能 与主键相关）----> 消除传递依赖（间接）

比如在设计一个订单数据表的时候，可以将客户编号作为一个外键和订单表建立相应的 关系。

而不可以在订单表中添加关于客户其它信息（比如姓名、所属公司等）的字段。

1. 数据库的五大约束是什么？

1.primary KEY:设置主键约束；

2.UNIQUE：设置唯一性约束，不能有重复值；

3.DEFAULT 默认值约束；

4.NOT NULL：设置非空约束，该字段不能为空；

5.FOREIGN key :设置外键约束；

1. 什么是事务？

事务用于将某些操作的多个SQL作为原子性操作，一旦有某一个出现错误，即可回滚到 原来的状态，从而保证数据库数据完整性。

事务的特性：

原子性: 确保工作单元内的所有操作都成功完成，否则事务将被中止在故障点，和以前 的操作将回滚到以前的状态。

一致性: 确保数据库正确地改变状态后，成功提交的事务。

隔离性: 使事务操作彼此独立的和透明的。

持久性: 确保提交的事务的结果或效果的系统出现故障的情况下仍然存在。

1. 简述数据库设计中一对多和多对多的应用场景？

FK(一对多)

下拉框里面的数据就需要用FK关联另一张表

M2M（多对多）

多选的下拉框，或者checkbox

1. 常见的SQL？

group by 分组对聚合的条件进行筛选需要通过havhing

SQL的left join 、right join、inner join之间的区别

left join (左连接) 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

right join(右连接) 返回包括右表中的所有记录1和左表中联结字段相等的记录

inner join（内连接）： 只返回两个表中联结字段相等的行

1. 主键和外键的区别？

主键是能确定一条记录的唯一标示。例如，身份证证号

外键：用于与另一张表的关联，是能确定另一张表记录的字段，用于保持数据的一致性

|  |  |  |
| --- | --- | --- |
|  | 主键 | 外键 |
| 定义 | 唯一标识一条记录，不能有重复的，不允许为空 | 表的外键是另一张表的主键，外键可以有重复的，可以为空 |
| 作用 | 用来保证数据完整性 | 用来与其他表建立联系的 |
| 个数 | 主键只能有一个 | 一个表可以有多个外键 |

1. MySQL常见的函数？

聚合函数

max/sum/min/avg

时间格式化

date\_format

字符串拼接

concat（当拼接了null，则返回null）

截取字符串

substring

返回字节个数

length

1. 如何开启慢日志查询？

修改配置文件

slow\_query\_log = OFF 是否开启慢日志记录

long\_query\_time = 2 时间限制，超过此时间，则记录

slow\_query\_log\_file = /usr/slow.log 日志文件

log\_queries\_not\_using\_indexes = OFF 为使用索引的搜索是否记录

下面是开启

slow\_query\_log = ON

long\_query\_time = 2

log\_queries\_not\_using\_indexes = OFF

log\_queries\_not\_using\_indexes = ON

注：查看当前配置信息：

　　 show variables like '%query%'

修改当前配置：

　　　　set global 变量名 = 值

1. 数据库导入导出命令（结构+数据）？

导出现有数据库数据：（当有提示出入密码。-p就不用加密码）

mysqldump -u用户名 -p密码 数据库名称 >导出文件路径 # 结构+数据

mysqldump -u用户名 -p密码 -d 数据库名称 >导出文件 路径 # 结构

导入现有数据库数据：

mysqldump -uroot -p密码 数据库名称 < 文件路径

1. char和varchar的区别？

char 和 varchar 的区别(char可变, varchar不可变 )

1. 1000w条数据，使用limit offset 分页时，为什么越往后翻越慢？如何解决？

答案一：

先查主键，在分页。

select \* from tb where id in (

select id from tb where limit 10 offset 30

)

答案二：

按照也无需求是否可以设置只让用户看200页

答案三：

记录当前页 数据ID最大值和最小值

在翻页时，根据条件先进行筛选；筛选完毕之后，再根据limit offset 查询。

select \* from (select \* from tb where id > 22222222) as B limit 10 offset 0

如果用户自己修改页码，也可能导致慢；此时对url种的页码进行加密（rest framework ）

1. redis和memcached比较？

1：redis不仅支持简单的key\_value类型，还支持字典，字符串，列表，集合，有序集 合类型

2：内存使用效率对比，使用简单的key-value存储的话，

Memcached的内存利用率更高而如果Redis采用hash结构来做key-value存储，由 于其组合式的压缩，其内存利用率会高于Memcached。

3.性能对比：由于Redis只使用单核，而Memcached可以使用多核，.

所以平均每一个核上Redis在存储小数据时比Memcached性能更高。而在100k以 上的数据中，Memcached性能要高于Redis，

4.Redis虽然是基于内存的存储系统，但是它本身是支持内存数据的持久化的，而且提 供两种主要的持久化策略：RDB快照和AOF日志。

而memcached是不支持数据持久化操作的。

5.集群管理不同，Memcached本身并不支持分布式，因此只能在客户端通过像一致性哈 希这样的分布式算法来实现Memcached的分布式存储。

1. redis中数据库默认是多少个db 及作用？

Redis默认支持16个数据库，可以通过配置databases来修改这一数字。客户端与Redis 建立连接后会自动选择0号数据库，不过可以随时使用SELECT命令更换数据库

Redis支持多个数据库，并且每个数据库的数据是隔离的不能共享，并且基于单机才有， 如果是集群就没有数据库的概念。

1. python操作redis的模块？

- 连接

- 直接连接：

import redis

r = redis.Redis(host='10.211.55.4', port=6379)

r.set('foo', 'Bar')

print r.get('foo')

- 连接池：

import redis

pool = redis.ConnectionPool(host='10.211.55.4', port=6379)

r = redis.Redis(connection\_pool=pool)

r.set('foo', 'Bar')

print r.get('foo')

1. vuex的作用？

多组件之间共享：vuex

补充luffyvue

1:router-link / router-view

2:双向绑定，用户绑定v-model

3：循环展示课程：v-for

4：路由系统，添加动态参数

5：cookie操作：vue-cookies

6：多组件之间共享：vuex

7：发送ajax请求：axios (js模块)

1. vue中的路由的拦截器的作用？

vue-resource的interceptors拦截器的作用正是解决此需求的妙方。

在每次http的请求响应之后，如果设置了拦截器如下，会优先执行拦截器函数，获取 响应体，然后才会决定是否把response返回给then进行接收。

1. axios的作用？

发送ajax请求：axios (js模块)

1. 列举vue的常见指令？

1、v-if指令:判断指令，根据表达式值得真假来插入或删除相应的值。

2、v-show指令:条件渲染指令，无论返回的布尔值是true还是false，元素都会存在 在html中，只是false的元素会隐藏在html中，并不会删除.

3、v-else指令:配合v-if或v-else使用。

4、v-for指令:循环指令，相当于遍历。

5、v-bind:给DOM绑定元素属性。

6、v-on指令:监听DOM事件。

1. 简述jsonp及实现原理？

JSONP

JSONP是json用来跨域的一个东西。原理是通过script标签的跨域特性来绕过同源策略。

JSONP的简单实现模式，或者说是JSONP的原型：创建一个回调函数，然后在远程服务 上调用这个函数并且将JSON 数据形式作为参数传递，

完成回调。

1. 什么是cors ？

CORS

浏览器将CORS请求分成两类：简单请求和赋复杂请求

简单请求(同时满足以下两大条件)

（1）请求方法是以下三种方法之一:

HEAD

GET

POST

（2）HTTP的头信息不超出以下几种字段：

Accept

Accept-Language

Content-Language

Last-Event-ID

Content-Type ：只限于三个值application/x-www-form-urlencoded、multipart/form-data、 text/plain

凡是不同时满足上面两个条件，就属于非简单请求

1. 列举Http请求中常见的请求头？

- user-agent

- host

- referer

- cookie

- content-type

1. django、flask、tornado框架的比较？

对于django，大而全的框架它的内部组件比较多，内部提供：ORM、Admin、中间件、 Form、ModelForm、Session、

缓存、信号、CSRF；功能也都挺完善的

- flask，微型框架，内部组件就比较少了，但是有很多第三方组件来扩展它，

　　比如说有那个wtform（与django的modelform类似，表单验证）、flask-sqlalchemy （操作数据库的）、

　　flask-session、flask-migrate、flask-script、blinker可扩展强，第三方组件丰富。所以 对他本身来说有那种短小精悍的感觉

- tornado，异步非阻塞。

django和flask的共同点就是，他们2个框架都没有写socket，所以他们都是利用第三 方模块wsgi。

但是内部使用的wsgi也是有些不同的：django本身运行起来使用wsgiref，而flask使用 werkzeug wsgi

还有一个区别就是他们的请求管理不太一样：django是通过将请求封装成request对象， 再通过参数传递，而flask是通过上下文管理机制

Tornado

# 是一个轻量级的Web框架，异步非阻塞+内置WebSocket功能。

'目标'：通过一个线程处理N个并发请求(处理IO)。

'内部组件

#内部自己实现socket

#路由系统

#视图

#模板

　　 #cookie

#csrf

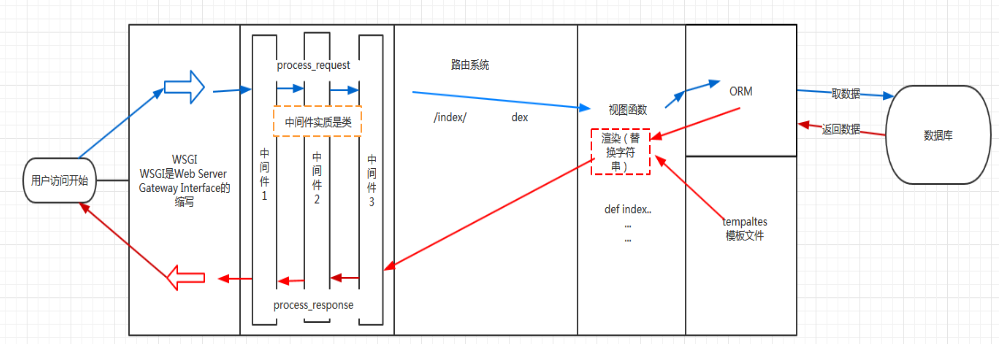
1. django请求的生命周期？

用户请求进来先走到 wsgi 然后将请求交给 jango的中间件 穿过django中间件 （方法是process\_request）

接着就是 路由匹配 路由匹配成功之后就执行相应的 视图函数

在视图函数中可以调用orm做数据库操作 再从模板路径 将模板拿到 然后在后 台进行模板渲染

模板渲染完成之后就变成一个字符串 再把这个字符串经过所有中间件（方法： process\_response） 和wsgi 返回给用户



1. F和Q的作用?

F:主要用来获取原数据进行计算。

Django 支持 F() 对象之间以及 F() 对象和常数之间的加减乘除和取模的操作。

修改操作也可以使用F函数,比如将每件商品的价格都在原价格的基础上增加10

from django.db.models import F

from app01.models import Goods

Goods.objects.update(price=F("price")+10) # 对于goods表中每件商品的价格都在原价 格的基础上增加10元

F查询专门对对象中某列值的操作，不可使用\_\_双下划线！

Q:用来进行复杂查询

Q查询可以组合使用 “&”, “|” 操作符，当一个操作符是用于两个Q的对象, 它产生一个新的Q对象，

　　Q对象可以用 “~” 操作符放在前面表示否定，也可允许否定与不否定形式的组 合。

　　Q对象可以与关键字参数查询一起使用，不过一定要把Q对象放在关键字参数查 询的前面。

Q(条件1) | Q(条件2) 或

Q(条件1) & Q(条件2) 且

Q(条件1) & ~Q(条件2) 非

1. values和values\_list的区别？

def values(self, \*fields):

# 获取每行数据为字典格式

def values\_list(self, \*fields, \*\*kwargs):

# 获取每行数据为元祖

1. django的Model中的ForeignKey字段中的on\_delete参数有什么作用？

在django2.0后，定义外键和一对一关系的时候需要加on\_delete选项，此参数为了避 免两个表里的数据不一致问题，不然会报错：

TypeError: \_\_init\_\_() missing 1 required positional argument: 'on\_delete'

举例说明：

user=models.OneToOneField(User)

owner=models.ForeignKey(UserProfile)

需要改成：

user=models.OneToOneField(User,on\_delete=models.CASCADE) --在老版本这 个参数（models.CASCADE）是默认值

owner=models.ForeignKey(UserProfile,on\_delete=models.CASCADE) --在老版本这个参 数（models.CASCADE）是默认值

参数说明：

on\_delete有CASCADE、PROTECT、SET\_NULL、SET\_DEFAULT、SET()五个可选择的值

CASCADE：此值设置，是级联删除。

PROTECT：此值设置，是会报完整性错误。

SET\_NULL：此值设置，会把外键设置为null，前提是允许为null。

SET\_DEFAULT：此值设置，会把设置为外键的默认值。

SET()：此值设置，会调用外面的值，可以是一个函数。

一般情况下使用CASCADE就可以了。

### 外加一些真实面试题（图片形式）：

