1. a=1，b=2，如何不用中间变量交换a和b的值

方法1,：

a, b = b, a

方法2：使用加法操作：

a+=b;

b=a-b;

a -= b;

方法3：使用异或操作：

a^=b;

b^=a;

a^=b;

2. 请描述set的用途并举例说明

是一个无序不重复元素集, 基本功能包括关系测试和消除重复元素。集合对象还支持union(联合), intersection(交), difference(差)和sysmmetric difference(对称差集)等数学运算。

old\_list = [1,1,1,3,4]

new\_list = list(set(old\_list))

3. 请简述python2.x与python3.x的主要区别

1.基本类型，Python2.x中有两种整数类型，一般的32位整数和长整数，长整数都是以L或者l（不建议用小写l，容易跟1混淆），超过32位长度之后会自动转换为长整型。

Python3.x中，只有一种类型，没有长度限制。

2.运算符 ，python2.x中不等于有两种写法，!=和<>

Python3.x中，去掉了<>，只有 != 一种写法。

3.Pyhon2.x中反引号``相当于repr函数的作用，python3.x中去掉了``这种写法。

4.Print语句 ，python3中print是一个内置函数，有多个参数。Python2中print是一个语法结构

5.编码格式，Python3中默认使用utf-8编码，不需要再在文件头上声明 #-\*-coding:utf-8 -\*-

6.Input函数，python3取消了raw\_input\

7.迭代器，Python3取消了xrange

8.import包时的路径搜索顺序，python3是import sys，python2是sys.path #查看搜索路径

4. Python里面re模块search()和match()的区别？

>>> import re

>>> re.match(r'python','Programing Python, should be pythonic')

>>> obj1 = re.match(r'python','Programing Python, should be pythonic') #返回None

>>> obj2 = re.search(r'python','Programing Python, should be pythonic') #找到pythonic

>>> obj2.group()

'python'

#re.match只匹配字符串的开始，如果字符串开始不符合正则表达式，则匹配失败，函数返回None；

#re.search匹配整个字符串，直到找到一个匹配。

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)

Match()函数只检测re是不是在string的开始位置匹配

Search()会扫描整个string查找匹配

也就是说match()只有在0位置匹配成功的话才有返回

如果不是开始位置匹配成功的话，match()就返回none

5. 写正则表达式，匹配一个http

url请求，该请求以abc.com为一级域名，包含多种二级域名，请求以.js结尾，并可能包含参数，例如：http://123.abc.com/qwerty.js或http://www.abc.com/asdfgh.js?version=1

Http\:\/\/\w{3}\.com\/\w{6}\.js

6. 请简述线程\进程\协程的特性

进程：是资源分配的最小单位，创建和销毁开销较大；

线程：是CPU调度的最小单位，开销小，切换速度快；

协程：轻量级的线程，又称之为 纤程

是一种用户态的轻量级线程，实际上只有一个单线程完成。

操作系统将CPU时间片分配给多个线程，每个线程在指定放到时间片内完成。操作系统不断从一个线程切换到另一个线程执行，宏观上看就好像是多个线程一起执行。

Python中由于全局锁 (GIL)的存在导致，同一时间只有一个获得GIL的线程在跑，其他线程则处于等待状态，这导致了多线程只是在做分时切换，并不能利用多核。

多线程与多进程的区别：（1）多进程中同一个变量各自有一份拷贝在每个进程中，互不影响；（2）多线程中，所有变量都由所有线程共享，任何一个变量都可被任何一个线程修改。线程之间共享数据的最大危险在于多个线程同时更改一个变量，把内容改乱。

协程优点：创建协程资源消耗非常少，协程的优点是可以用作IO高并发处理

无需上下文切换的开销，没有临界资源的争夺，保证对数据的原子操作，缺点是无法利用多核。

进程拥有自己独立的堆和栈，既不共享堆，也不共享栈，进程由操作系统调度

线程拥有自己独立的栈，和共享的堆，不共享栈，线程也是由操作系统调度

协程和线程一样共享堆，不共享栈，协程由程序员在协程的代码里显示调度

协程避免了无意义的调度，由此可以提高性能，但也因此，程序员必须自己承担调度的责任，同时，协程也失去了标准线程使用多cpu的能力

7. 请简述你对python闭包的理解

闭包是词法闭包的简称，是引用了自由变量的函数。是函数式编程的重要语法结构

闭包也是一种组织代码的结构。它同样提高了代码的可重复性

特性： 1.函数嵌套函数 2.函数内部可以引用外部的参数和变量 3.参数和变量不会被垃圾回收机制回收

闭包的缺点就是常驻内存，会增大内存使用量，使用不当很容易造成内存泄露。

为什么要使用闭包：

为了设计私有方法和变量，避免全局变量污染希望一个变量长期驻扎在内存中

闭包(closure)是函数式编程的重要的语法结构。闭包也是一种组织代码的结构，它同样提高了代码的可重复使用性。

当一个内嵌函数引用其外部作用域的变量,我们就会得到一个闭包. 总结一下,创建一个闭包必须满足以下几点:

必须有一个内嵌函数

内嵌函数必须引用外部函数中的变量

外部函数的返回值必须是内嵌函数

重点是函数运行后并不会被撤销,这个功能类似类里的类变量,只不过迁移到了函数上.

闭包可以实现先将一个参数传递给一个函数，而并不立即执行，以达到延迟求值的目的。

defdelay\_fun(x, y):

defcaculator():

returnx+y

returncaculator

print('返回一个求和的函数，并不求和')

msum = delay\_fun(3,4)

print('调用并求和:')

print(msum())

8. 什么是静态方法，类方法，有什么区别

静态方法：静态方法是定义在类内部的方法，只能借助于该类访问

静态方法需要用 @staticmethod装饰器定义

静态方法写普通函数的定义相同，不需要传入self和cls 参数

通常是在类中使用静态方法，要求在类成员函数前面加上@staticmethod标记符，以表示下面的成员函数是静态函数。使用静态方法的好处是，不需要定义实例即可使用这个方法。另外，多个实例共享此静态方法。

类方法：一个类方法就可以通过类或它的实例来调用的方法, 不管你是用类来调用这个方法还是类实例调用这个方法,该方法的第一个参数总是定义该方法的类对象。

区别：

虽然静态方法和类方法调用方式都一样，都可以通过[类.方法名]来使用，但是类方法中有默认属性cls，指向自身的类，所以一般如果是单一的工具函数定义成静态方法，如果还需要调用类中其他的静态方法，或者类方法的函数定义成类方法

主要区别在于参数传递上的区别，实例方法悄悄传递的是self引用作为参数，而类方法悄悄传递的是cls引用作为参数。

静态方法只是名义上归类管理，实际上在静态方法里访问不了类或实例中的任何属性。

类方法只能访问类变量，不能访问实例变量

class A(object):

def fun\_a(self):#实例方法

pass

@staticmethod

def fun\_b():#静态方法

pass

@classmethod

def fun\_c(cls):#类方法

pass

9. 描述你知道的设计模式及各模式特点

1.创建模式，提供实例化的方法，为适合的状况提供相应的对象创建方法。

2.结构化模式，通常用来处理实体之间的关系，使得这些实体能够更好地协同工作。

3.行为模式，用于在不同的实体间进行通信，为实体之间的通信提供更容易，更灵活的通信方法。

10. 解释一下Django和Tornado这两种web框架的关系、差别

Django源自一个在线新闻 Web站点，于 2005 年以开源的形式被释放出来。

Django框架的核心组件有：

用于创建模型的对象关系映射为最终用户设计的完美管理界面一流的 URL 设计设计者友好的模板语言缓存系统等等.它鼓励快速开发,并遵循MVC设计。Django遵守 BSD版权，最新发行版本是Django1.4，于2012年03月23日发布.Django的主要目的是简便、快速的开发数据库驱动的网站。它强调代码复用,多个组件可以很方便的以“插件”形式服务于整个框架，Django有许多功能强大的第三方插件，你甚至可以很方便的开发出自己的工具包。这使得Django具有很强的可扩展性。它还强调快速开发和DRY(Do Not RepeatYourself)原则。

Tornado是FriendFeed使用的可扩展的非阻塞式 web 服务器及其相关工具的开源版本。这个 Web 框架看起来有些像 web.py 或者 Google 的webapp，不过为了能有效利用非阻塞式服务器环境，这个 Web 框架还包含了一些相关的有用工具和优化。

Tornado 和现在的主流 Web 服务器框架（包括大多数Python 的框架）有着明显的区别：它是非阻塞式服务器，而且速度相当快。得利于其非阻塞的方式和对epoll的运用，Tornado 每秒可以处理数以千计的连接，这意味着对于实时 Web服务来说，Tornado 是一个理想的 Web 框架。我们开发这个 Web 服务器的主要目的就是为了处理FriendFeed的实时功能 ——在FriendFeed的应用里每一个活动用户都会保持着一个服务器连接。

11. 解释下Http协议

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。

HTTP协议的主要特点可概括如下：

1.支持客户/服务器模式。

2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。

3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。

4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

5.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

12. 解释下Http请求头和常见响应状态码

Accept:指浏览器或其他客户可以接爱的MIME文件格式。可以根据它判断并返回适当的文件格式。

Accept-Charset：指出浏览器可以接受的字符编码。英文浏览器的默认值是ISO-8859-1.

Accept-Language：指出浏览器可以接受的语言种类，如en或en-us，指英语。

Accept-Encoding：指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到Web响应之后先解码，然后再检查文件格式。

Cache-Control：设置关于请求被代理服务器存储的相关选项。一般用不到。

Connection：用来告诉服务器是否可以维持固定的HTTP连接。HTTP/1.1使用Keep-Alive为默认值，这样，当浏览器需要多个文件时(比如一个HTML文件和相关的图形文件)，不需要每次都建立连接。

Content-Type：用来表名request的内容类型。可以用HttpServletRequest的getContentType()方法取得。

Cookie：浏览器用这个属性向服务器发送Cookie。Cookie是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能。

状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息–表示请求已接收，继续处理

2xx：成功–表示请求已被成功接收、理解、接受

3xx：重定向–要完成请求必须进行更进一步的操作

4xx：客户端错误–请求有语法错误或请求无法实现

5xx：服务器端错误–服务器未能实现合法的请求

常见状态代码、状态描述、说明：

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

13. 解释下REST API接口的设计原则

REST API是使用统一资源标识符（url）来搜寻资源 有七大设计原则：

1. url结尾不应包含（/）

2. 正斜杠分隔符（/）必须用来指示层级关系

3. 应使用连字符（-）来提高url的可读性

4. 不得在url中使用下划线（\_）

5. url路径中首选小写字母

6. 文件扩展名中不应包含在url中

7. 端点名称是单数（但是实际上为了保持url格式的一致性建议使用复数形式）

正在构建的服务中的每个资源将至少有一个url标识它，这个url最好是有意义的，且能充分藐视资源。url应遵循可预测的层次结构，用来提高其可理解性，可用性：可预测的意义在于他们是一致的，他的层次结构在数据关系上时有意义的。

REST API是使用者编写的，url的名称和结构应该能够向使用者传达更清晰的含义。通过遵循上述规则，您将创建一个更清晰的REST API与更友好的客户端。

14. django models中如何实现反向查找？如a表中通过外键关联了b表，那通过b表如何反查到a表中有多少记录关联到了b表中的某条记录

先定义两个模型，一个是A，一个是B，是一对多的类型。

class A(models.Model):

name= models.CharField('名称', max\_length=32)

class B(models.Model):

a= models.ForeignKey(A, verbose\_name='A类',related\_name = "test")

name = models.CharField('称呼', max\_length=16)

如果我们要查询一个A拥有哪些B的话， 需要通过主表来查询子表

A.objects.get(id=A\_id).test.all().order\_by('-created'),

django 默认每个主表的对象都有一个是外键的属性，可以通过它来查询到所有属于主表的子表的信息。这个属性的名称默认是以子表的名称小写加上\_set()来表示(上面默认以b\_set访问)，默认返回的是一个querydict对象。

related\_name 可以给这个外键定义好一个别的名称

通过子表来查询主表

B.objects.filter(a=A\_id).order\_by('-created')

15. 简单说下关系型数据库和非关系型数据库的区别，你用过哪些关系型和非关系型数据库

关系型数据库通过外键关联来建立表与表之间的关系，

非关系型数据库通常指数据以对象的形式存储在数据库中，而对象之间的关系通过每个对象自身的属性来决定

关系型数据库的优势：

1. 复杂查询

可以用SQL语句方便的在一个表以及多个表之间做非常复杂的数据查询。

2. 事务支持

使得对于安全性能很高的数据访问要求得以实现。

用过的关系型数据库：SQLite、Oracle、mysql

用过的非关系型数据库：MongoDb、redis、HBase

非关系型数据库的优势：

1. 性能

NOSQL是基于键值对的，可以想象成表中的主键和值的对应关系，而且不需要经过SQL层的解析，所以性能非常高。

2. 可扩展性

同样也是因为基于键值对，数据之间没有耦合性，所以非常容易水平扩展。

关系型数据库（MySQL），由数据库、表、记录三个层次组成，所用语句为传统的SQL语句，但是在海量数据处理时效率会有所下降。

非关系型数据库（MongoDB），由数据库、集合、文档三个层次构成，数据结构由键值对组成，适用于事件地记录、内容管理或博客平台等，不太稳定，且不支持事务操作，占用空间大。

16. 将一个字符串逆序，不能使用反转函数

>>> S = ‘abcde’

>>> S[::-1]

17. 求从10到100中能被3或5整除的数的和

L = []

for i in range(10,101):

if i % 3 == 0 and i % 5 ==0:

L.append(i)

Print(sum(L))