一： python基础部分

1:*args and **kwargs

函数的不定长参数

*args指的是可以有多个实参，放到一个元祖里面

**kwargs ** 指的是按照关键字传值把多余的传值以字典的方式呈现

简单来说就是 *arg是元组，接收多个实参 **args是字典，接收多个健值参数

2:@staticmethod和@classmethod

实例方法

定义：第一个参数必须是实例对象，该参数名一般约定为“self”，通过它来传递实例的属性和方法（也可以传类的属性和方法）；

调用：只能由实例对象调用。

类方法

定义：使用装饰器@classmethod。第一个参数必须是当前类对象，该参数名一般约定为“cls”，通过它来传递类的属性和方法（不能传实例的属性和方法）；

调用：实例对象和类对象都可以调用。

静态方法

定义：使用装饰器@staticmethod。参数随意，没有“self”和“cls”参数，但是方法体中不能使用类或实例的任何属性和方法；

调用：实例对象和类对象都可以调用。

静态方法是类中的函数，不需要实例。静态方法主要是用来存放逻辑性的代码，逻辑上属于类，但是和类本身没有关系，也就是说在静态方法中，不会涉及到类中的属性和方法的操　　作。可以理解为，静态方法是个**独立的、单纯的**函数，它仅仅托管于某个类的名称空间中，便于使用和维护。

实例方法针对的是实例对象，类方法针对的是类，他们都可以继承和重新定义，而静态方法则不能继承，可以认为是全局函数。

3：__new__和**init**的区别

1. __new__是一个静态方法,而__init__是一个实例方法.
2. __new__方法会返回一个创建的实例对象,而__init__什么都不返回,只是在对象调用的时候，给对象尽心一些初始化操作.
3. 只有在__new__返回一个cls的实例对象后，__init__方法才能被调用.
4. 当创建一个新实例时调用__new__,初始化一个实例时用__init__.

4: 迭代器和生成器

迭代器：迭代是访问集合元素的一种方式。迭代器是一个可以记住遍历的位置的对象。迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束。迭代器只能往前不会后退。

可迭代对象

以直接作用于 for 循环的数据类型有以下几种：

一类是集合数据类型，如 list 、 tuple 、 dict 、 set 、 str 等；

一类是 generator ，包括生成器和带 yield 的generator function。

这些可以直接作用于 for 循环的对象统称为可迭代对象： Iterable 。

- 凡是可作用于 for 循环的对象都是 Iterable 类型；
- 凡是可作用于 next() 函数的对象都是 Iterator 类型
- 集合数据类型如 list 、 dict 、 str 等是 Iterable 但不是 Iterator ，不过可以通过 iter() 函数获得一个 Iterator 对象。

生成器：在python中一边循环一遍计算的机制，被称为生成器
生成器是这样一个函数，它记住上一次返回时在函数体中的位置。对生成器函数的第二次（或第 n 次）调用跳转至该函数中间，而上次调用的所有局部变量都保持不变。

生成器的特点：

1. 节约内存
2. 迭代到下一次的调用时，所使用的参数都是第一次所保留下的，即是说，在整个所有函数调用的参数都是第一次所调用时保留的，而不是新创建的
3.

5：闭包

```
5.  def test(number):
6.
7.      #在函数内部再定义一个函数，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之为闭包
8.      def test_in(number_in):
9.          print("in test_in 函数, number_in is %d"%number_in)
10.         return number+number_in
11.     #其实这里返回的就是闭包的结果
```

```
    12.         return test_in
```

优缺点：

1.闭包似优化了变量，原来需要类对象完成的工作，闭包也可以完成

2.由于闭包引用了外部函数的局部变量，则外部函数的局部变量没有及时释放，消耗内存

6：装饰器

```
def w1(func):
    def inner():
        # 验证1
        # 验证2
        # 验证3
        func()
    return inner

@w1
def f1():
    print('f1')
```

装饰器(decorator)功能

1. 引入日志
2. 函数执行时间统计
3. 执行函数前预备处理
4. 执行函数后清理功能
5. 权限校验等场景
6. 缓存
7. 检测是否登录

7:单例模式

1 使用__new__方法

```
class Singleton(object):
    def __new__(cls, *args, **kw):
        if not hasattr(cls, '_instance'):
            orig = super(Singleton, cls)
            cls._instance = orig.__new__(cls, *args,
**kw)
        return cls._instance
```

```
class MyClass(Singleton):
    a = 1
```

2 共享属性

创建实例时把所有实例的__dict__指向同一个字典,这样它们具有相同的属性和方法.

```
class Borg(object):
    _state = {}
    def __new__(cls, *args, **kw):
        ob = super(Borg, cls).__new__(cls, *args, **kw)
        ob.__dict__ = cls._state
        return ob

class MyClass2(Borg):
    a = 1
```

3 装饰器版本

```
def singleton(cls):
    instances = {}
    def getinstance(*args, **kw):
        if cls not in instances:
            instances[cls] = cls(*args, **kw)
        return instances[cls]
    return getinstance

@singleton
class MyClass:
  ...
```

4 import方法

作为python的模块是天然的单例模式

```
# mysingleton.py
class My_Singleton(object):
    def foo(self):
        pass

my_singleton = My_Singleton()

# to use
from mysingleton import my_singleton
```

```
my_singleton.foo()
```

8: python中的自省

自省就是面向对象的语言所写的程序在运行时,所能知道对象的类型.
简单一句就是运行时能够获得对象的类型.比如
type(),dir(),getattr(),hasattr(),isinstance().

```
a = [1,2,3]
b = {'a':1,'b':2,'c':3}
c = True
print type(a),type(b),type(c) # <type 'list'> <type
'dict'> <type 'bool'>
print isinstance(a,list)  # True
```

9: 私有化

- xx: 公有变量
- _x: 单前置下划线,私有化属性或方法，from somemodule import *禁止导入,类对象和子类可以访问
- __xx：双前置下划线,避免与子类中的属性命名冲突，无法在外部直接访问(名字重整所以访问不到)
- __xx__:双前后下划线,用户名字空间的魔法对象或属性。例如:__init__ , __ 不要自己发明这样的名字
- xx_:单后置下划线,用于避免与Python关键词的冲突

10:属性

@property成为属性函数，可以对属性赋值时做必要的检查，并保证代码的清晰短小，主要有2个作用

- 将方法转换为只读
- 重新实现一个属性的设置和读取方法,可做边界判定

```
 @property
    def money(self):
        return self.__money

    @money.setter
    def money(self, value):
        if isinstance(value, int):
            self.__money = value
        else:
```

```
print("error:不是整型数字")
```

11: Python垃圾回收机制

Python GC主要使用引用计数（reference counting）来跟踪和回收垃圾。在引用计数的基础上，通过“标记-清除”（mark and sweep）解决容器对象可能产生的循环引用问题，通过“分代回收”（generation collection）以空间换时间的方法提高垃圾回收效率。

12: 深拷贝，浅拷贝

copy是浅拷贝，只拷贝可变对象的父级元素。 deepcopy是深拷贝，递归拷贝可变对象的所有元素。
浅拷贝，拷贝的是指针
深拷贝拷贝的是内容

```
import copy
a = [1, 2, 3, 4, ['a', 'b']]  #原始对象

b = a  #赋值，传对象的引用
c = copy.copy(a)  #对象拷贝，浅拷贝
d = copy.deepcopy(a)  #对象拷贝，深拷贝

a.append(5)  #修改对象a
a[4].append('c')  #修改对象a中的['a', 'b']数组对象

print 'a = ', a
print 'b = ', b
print 'c = ', c
print 'd = ', d
```

输出结果：

```
a =  [1, 2, 3, 4, ['a', 'b', 'c'], 5]
b =  [1, 2, 3, 4, ['a', 'b', 'c'], 5]
c =  [1, 2, 3, 4, ['a', 'b', 'c']]
d =  [1, 2, 3, 4, ['a', 'b']]
```

13: python2和python3的区别

print函数:（Python3中print为一个函数，必须用括号括起来；Python2中print为class）
Python3中用input，Python2中用row_input，都输入为str
打开文件，python2，3 **只能用 open(.....)**

14： range和xrange

都在循环时使用，xrange内存性能更好。

15: read. readline readlines. 等区别

- read 读取整个文件
- readline 读取下一行,使用生成器方法
- readlines 读取整个文件到一个迭代器以供我们遍历

16: 什么是python的命名空间

在Python中，所有的名字都存在于一个空间中，它们在该空间中存在和被操作——这就是命名空间。
它就好像一个盒子，每一个变量名字都对应装着一个对象。当查询变量的时候，会从该盒子里面寻找相应的对象。

17: python 语言的优缺点

Python的优点

1. 简单 Python的语法非常优雅，甚至没有像其他语言的大括号，分号等特殊符号，代表了一种极简主义的设计思想。阅读Python程序像是在读英语。

2. 易学 Python入手非常快，学习曲线非常低，可以直接通过命令行交互环境来学习Python编程。

3. 免费/开源 Python的所有内容都是免费开源的，这意味着你不需要花一分钱就可以免费使用Python，并且你可以自由地发布这个软件的拷贝、阅读它的源代码、对它做改动、把它的一部分用于新的自由软件中。

4. 自动内存管理 如果你了解C语言、C++语言你就会知道内存管理给你带来很大麻烦，程序非常容易出现内存方面的漏洞。但是在Python中内存管理是自动完成的，你可以专注于程序本身。

5. 可以移植 由于Python是开源的，它已经被移植到了大多数平台下面，例如：Windows、MacOS、Linux、Andorid、iOS等等。

6. 解释性 大多数计算机编程语言都是编译型的，在运行之前需要将源码编译为操作系统可以执行的二进制格式(0110格式的)，这样大型项目编译过程非常消耗时间，而Python语言写的程序不需要编译成二进制代码。你可以直接从源代码运行程序。在计算机内部，Python解释器把源代码转换成称为字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。

7. 面向对象 Python既支持面向过程，又支持面向对象，这样编程就更加灵活。

8. 可扩展 Python除了使用Python本身编写外，还可以混合使用像C语言、Java语言等编写。

9. 丰富的第三方库 Python具有本身有丰富而且强大的库，而且由于Python的开源特性，第三方库也非常多，例如：在web开发、爬虫、科学计算等等

Python的缺点

Python虽然有很多优点，但是它也不是完美的，它也有自身的缺点。

1. 速度慢 由于，Python是解释型语言，所有它的速度会比，C、C++慢一些，但是不影响使用。由于，现在的硬件配置都非常高，基本上没有影响，除非是一些实时性比较强的程序可能会受到一些影响，但是也有解决办法，可以嵌入C程序。

2. 强制缩进 如果你有其他语言的编程经验，例如：C语言或者Java语言，那么Python的强制缩进一开始会让你很不习惯。但是如果你习惯了Python的缩进语法，你会觉得它非常优雅。

3. 单行语句 由于Python可以在尾部不写分号，所以一行只能有一条语句，这可能也算是一个不足吧，不过这真的微不足道。

18什么是Python装饰器?

Python装饰器是Python中的特有变动，可以使修改函数变得更容易。

19数组和元组之间的区别是什么?

数组和元组之间的区别：数组内容是可以被修改的，而元组内容是只读的。另外，元组可以被哈希，比如作为字典的关键字。

20参数按值传递和引用传递是怎样实现的?

Python中的一切都是类，所有的变量都是一个对象的引用。引用的值是由函数确定的，因此无法被改变。但是如果一个对象是可以被修改的，你可以改动对象。

21.字典推导式和列表推导式是什么?

它们是可以轻松创建字典和列表的语法结构。

22.Python都有哪些自带的数据结构?

Python自带的数据结构分为可变的和不可变的。可变的有：数组、集合、字典；不可变的有：字符串、元组、数。

23.Python中的pass是什么?

Pass是一个在Python中不会被执行的语句。在复杂语句中，如果一个地方需要暂时被留白，它常常被用于占位符。

24你调试python代码的方法有哪些?

具体IDE都有调试，比如:IDLE, Eclipse+Pydev都可以设置断点调试。

pdb模块也可以做调试。

还有PyChecker和Pylint

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coding standard检查。

25:"猴子补丁"（monkey patching）指的是什么？这种做法好吗?

答案：

"猴子补丁"就是指，在函数或对象已经定义之后，再去改变它们的行为。

举个例子：
import datetime
datetime.datetime.now = lambda: datetime.datetime(2012, 12, 12)
大部分情况下，这是种很不好的做法 - 因为函数在代码库中的行为最好是都保持一致。打"猴子补丁"的原因可能是为了测试。mock包对实现这个目的很有帮助。

26下面这些是什么意思：@classmethod, @staticmethod, @property?

回答背景知识

这些都是装饰器（decorator）。装饰器是一种特殊的函数，要么接受函数作为输入参数，并返回一个函数，要么接受一个类作为输入参数，并返回一个类。

27请写一段代码实现Python中list去重。

```
# 方法1
list1 = [1,1,2,3,3,4]
set1 = set(list1)
list1 = list(set1)
# 方法2
list2 = []
for i in list1:
if i not in list2:
list2.append(i)
```

28描述yield作用。

i 保存当前运行状态（断点），然后暂停执行，即将函数挂起
ii 将yeild关键字后面表达式的值作为返回值返回，此时可以理解为起到了return的作用，当使用next()、send()函数让函数从断点处继续执行，即唤醒函数

29.python是怎么进行内存管理的?

i 引用计数：python内部使用引用计数，来保持追踪内存中的对象，Python内部记录了对象有多少个引用，即引用计数，当对象被创建时就创建了一个引用计数，当对象不再需要时，这个对象的引用计数为0时，它被垃圾回收。
  a 引用计数加1的情况：
    a 对象被创建：x=4
    b 另外的别人被创建：y=x
    c 被作为参数传递给函数：foo(x)
    d 作为容器对象的一个元素：a=[1,x,'33']
  b 引用计数减少情况
    a 一个本地引用离开了它的作用域。比如上面的foo(x)函数结

束时，x指向的对象引用减1。

b 对象的别名被显式的销毁：del x ；或者del y

c 对象的一个别名被赋值给其他对象：x=789

d 对象从一个窗口对象中移除：myList.remove(x)

e 窗口对象本身被销毁：del myList，或者窗口对象本身离开了作用域

ii 垃圾回收

a 当内存中有不再使用的部分时，垃圾收集器就会把他们清理掉。它会去检查那些引用计数为0的对象，然后清除其在内存的空间。当然除了引用计数为0的会被清除，还有一种情况也会被垃圾收集器清掉：当两个对象相互引用时，他们本身其他的引用已经为0了。

b 垃圾回收机制还有一个循环垃圾回收器, 确保释放循环引用对象(a引用b, b引用a, 导致其引用计数永远不为0)。

iii 内存池机制：在Python中，许多时候申请的内存都是小块的内存，这些小块内存在申请后，很快又会被释放，由于这些内存的申请并不是为了创建对象，所以并没有对象一级的内存池机制。这就意味着Python在运行期间会大量地执行malloc和free的操作，频繁地在用户态和核心态之间进行切换，这将严重影响Python的执行效率。为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。

a Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给操作系统。

b Python中所有小于256个字节的对象都使用pymalloc实现的分配器，而大的对象则使用系统的 malloc。另外Python对象，如整数，浮点数和List，都有其独立的私有内存池，对象间不共享他们的内存池。也就是说如果你分配又释放了大量的整数，用于缓存这些整数的内存就不能再分配给浮点数。

30:什么是lambda函数？他有什么好处?

lambda函数是匿名函数；使用lambda函数能够创建小型匿名函数。这种函数得名于省略了用def声明函数的标准步骤； 例:

```
f = lambda x,y:x+y # 求两个函数的和。 x,y是参数，x+y是函数返回值
```

31:合并列表a=[1,2,3,4]和b=[5,6,7,8]

```
a.extend(b)
```

32:描述对super，pass，yield，lambda关键字修饰的理解。

- super：在继承中充当父类的代理对象，在多继承中，super的调用顺序是MRO的顺序。
- pass：空语句，什么也不做，在特别的时候用来保证格式或是语义的完整

性。
- yield:
- 1.保存当前运行状态（断点），然后暂停执行，即将函数挂起
- 2.将yeild关键字后面表达式的值作为返回值返回，此时可以理解为起到了return的作用，当使用next()、send()函数让函数从断点处继续执行，即唤醒函数。
- lambda：定义匿名函数

33.使用python编写一个装饰器，打印被装饰函数的输入与输出。

```
from functools import wraps
def print_io(func):
@wraps(func)
def inner(*args,**kwargs):
print("函数的输入：{}".format(*args,**kwargs))
ret = func(*args,**kwargs)
print("函数的输出：{}".format(ret))
return ret
return inner
```

34.python中如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）

答：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}

35.线程中start方法和run方法的区别?

i 若调用start,则先执行主进程，后执行子进程；

ii 若调用run，相当于正常的函数调用，将按照程序的顺序执行

36.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?

答：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

37.Python是如何进行类型转换的?

Python提供了将变量或值从一种类型转换成另一种类型的内置函数。比如int函数能够将符合数学格式数字型字符串转换成整数。否则，返回错误信息。

一、类继承

问题：有如下的一段代码：

```
class A(object):
    def show(self):
        print 'base show'

class B(A):
    def show(self):
        print 'derived show'

obj = B()
obj.show()
```

如何调用类 A 的 show 方法?

答案：方法如下：

```
obj.__class__ = A
obj.show()
```

__class__ 方法指向了类对象，只用给他赋值类型 A ， 然后调用方法 show ， 但是用完了记得修改回来。

二、方法对象

问题：为了让下面这段代码运行，需要增加哪些代码?

```
class A(object):
    def __init__(self,a,b):
        self.__a = a
        self.__b = b
    def myprint(self):
        print 'a=', self.__a, 'b=', self.__b


a1=A(10,20)
a1.myprint()

a1(80)
```

答案：为了能让对象实例能被直接调用，需要实现 __call__ 方法

```
class A(object):
    def __init__(self, a, b):
        self.__a = a
        self.__b = b
    def myprint(self):
        print 'a=', self.__a, 'b=', self.__b
    def __call__(self, num):
        print 'call:', num + self.__a
```

三、 **new 和 init**

问题：下面这段代码输入什么？

```
class B(object):
    def fn(self):
        print 'B fn'
    def __init__(self):
        print "B INIT"


class A(object):
    def fn(self):
        print 'A fn'

    def __new__(cls,a):
            print "NEW", a
            if a>10:
                return super(A, cls).__new__(cls)
            return B()

    def __init__(self,a):
        print "INIT", a

a1 = A(5)
a1.fn()
a2=A(20)
a2.fn()
```

答案：

```
NEW 5
B INIT
B fn
NEW 20
INIT 20
A fn
```

使用 __new__ 方法，可以决定返回那个对象，也就是创建对象之前，这个可以用于设计模式的单例、工厂模式。__init__ 是创建对象是调用的。

四、Python list 和 dict 生成

问题：下面这段代码输出什么？

```
ls = [1,2,3,4]
list1 = [i for i in ls if i>2]
print list1

list2 = [i*2 for i in ls if i>2]
print list2

dic1 = {x: x**2 for x in (2, 4, 6)}
print dic1

dic2 = {x: 'item' + str(x**2) for x in (2, 4, 6)}
print dic2

set1 = {x for x in 'hello world' if x not in 'low level'}
print set1
```

答案:

```
[3, 4]
[6, 8]
{2: 4, 4: 16, 6: 36}
{2: 'item4', 4: 'item16', 6: 'item36'}
set(['h', 'r', 'd'])
```

五、全局和局部变量

问题：下面这段代码输出什么?

```
num = 9

def f1():
    num = 20

def f2():
    print num


f2()
f1()
f2()
```

答案:

```
9
9
```

num 不是个全局变量，所以每个函数都得到了自己的 num 拷贝，如果你想修改 num ，则必须用 global 关键字声明。比如下面这样

```
num = 9

def f1():
    global num
    num = 20

def f2():
   print num

f2()
f1()
f2()

# prints:
#      9
#      20
```

六、交换两个变量的值

问题：一行代码交换两个变量值

```
a=8
b=9
```

答案：

```
(a,b) = (b,a)
```

七、默认方法

问题：如下的代码

```
class A(object):
    def __init__(self,a,b):
        self.a1 = a
        self.b1 = b
        print 'init'
    def mydefault(self):
        print 'default'

a1 = A(10,20)
a1.fn1()
a1.fn2()
a1.fn3()
```

方法 fn1/fn2/fn3 都没有定义，添加代码，使没有定义的方法都调用 mydefault 函数，上面的代码应该输出

```
default
default
default
```

答案：

```
class A(object):
    def __init__(self,a,b):
        self.a1 = a
        self.b1 = b
        print 'init'
    def mydefault(self):
        print 'default'
    def __getattr__(self,name):
        return self.mydefault

a1 = A(10,20)
a1.fn1()
a1.fn2()
a1.fn3()
```

方法 __getattr__ 只有当没有定义的方法调用时，才是调用他。当 fn1 方法传入参数时，我们可以给 mydefault 方法增加一个 *args 不定参数来兼容。

```
class A(object):
    def __init__(self,a,b):
        self.a1 = a
        self.b1 = b
        print 'init'
    def mydefault(self,*args):
        print 'default:' + str(args[0])
    def __getattr__(self,name):
        print "other fn:",name
        return self.mydefault

a1 = A(10,20)
a1.fn1(33)
a1.fn2('hello')
a1.fn3(10)
```

八、包管理

问题：一个包里有三个模块，mod1.py ， mod2.py ， mod3.py ，但使用 from demopack import * 导入模块时，如何保证只有 mod1 、 mod3 被导入了。

答案:增加 __init__.py 文件，并在文件中增加：

```
__all__ = ['mod1','mod3']
```

九、闭包

问题：写一个函数，接收整数参数 n ，返回一个函数，函数的功能是把函数的参数和 n 相乘并把结果返回。

答案:

```
def mulby(num):
    def gn(val):
        return num * val

    return gn


zw = mulby(7)
print(zw(9));
```

十、性能

问题：解析下面的代码慢在哪

```
def strtest1(num):
    str='first'
    for i in range(num):
        str+="X"
    return str
```

答案：python的str是个不可变对象，每次迭代，都会生成新的str对象来存储新的字符串，num越大，创建的str对象越多，内存消耗越大

二：网络基础部分

什么是restful – api

RESTful API 就是由SERVER来提供，前端来调用基于Http协议的一套Api协议规则

那么RESTful API有哪些设计原则和规范呢?

1，资源。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。

5， RESTful API还有其他一些规范。1：应该将API的版本号放入URL。GET:http://www.xxx.com/v1/friend/123。或者将版本号放在HTTP头信息中。我个人觉得要不要版本号取决于自己开发团队的习惯和业务的需要，不是强制的。2：URL中只能有名词而不能有动词，操作的表达是使用HTTP的动词GET,POST,PUT,DELETEL。URL只标识资源的地址，既然是资源那就是名词了。3：如果记录数量很多，服务器不可能都将它们返回给用户。API应该提供参数，过滤返回结果。?limit=10：指定返回记录的数量、?page=2&per_page=100：指定第几页，以及每页的记录数。

1: get post 请求区别

Http定义了与服务器交互的不同方法，最基本的方法有4种，分别是GET，POST，PUT，DELETE。URL全称是资源描述符，我们可以这样认为：一个URL地址，它用于描述一个网络上的资源，而HTTP中的GET，POST，PUT，DELETE就对应着对这个资源的查，改，增，删4个操作。到这里，大家应该有个大概的了解了，GET一般用于获取/查询资源信息，而POST一般用于更新资源信息。

1，http中，GET用于信息获取，而且是安全的和幂等的。

(1).所谓安全的意味着该操作用于获取信息而非修改信息。换句话说，GET 请求一般不应产生副作用。就是说，它仅仅是获取资源信息，就像数据库查询一样，不会修改，增加数据，不会影响资源的状态。
* 注意：这里安全的含义仅仅是指是非修改信息。

(2).幂等的意味着对同一URL的多个请求应该返回同样的结果。
2，http中，POST是用于修改服务器上的资源的请求。

说完原理性的问题，我们从表面上来看看GET和POST的区别：
1. get是从服务器上获取数据，post是向服务器传送数据。 get 和 post只是一种传递数据的方式，get也可以把数据传到服务器，他们的本质都是发送请求和接收结果。只是组织格式和数据量上面有差别，http协议里面有介绍 2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。 因为get设计成传输小数据，而且最好是不修改服务器的数据，所以浏览器一般都在地址栏里面可以看到，但post一般都用来传递大数据，或比较隐私的数据，所以在地址栏看不到，能不能看到不是协议规定，是浏览器规定的。3. 对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。 没明白，怎么获得变量和你的服务器有关，和get或post无关，服务器都对这些请求做了封装 4. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。但理论上，IIS4中最大量为80KB，IIS5中为100KB。 post基本没有限制，我想大家都上传过文件，都是用post方式的。只不过要修改form里面的那个type参数 5. get安全性非常低，post安全性较高。 如果没有加密，他们安全级别都是一样的，随便一个监听器都可以把所有的数据监听到。

2：cookie和session

	Cookie	**Session**
储存位置	客户端	服务器端
目的	跟踪会话，也可以保存用户偏好设置或者保存用户名密码等	跟踪会话
安全性	不安全	

session技术是要使用到cookie的，之所以出现session技术，主要是为了安全。

3: 三次握手

三次握手过程说明：
1、由客户端发送建立TCP连接的请求报文，其中报文中包含seq序列号，是由发送端随机生成的，并且将报文中的SYN字段置为1，表示需要建立TCP连接。（SYN=1，seq=x，x为随机生成数值）
2、由服务端回复客户端发送的TCP连接请求报文，其中包含seq序列号，是由回复端随机生成的，并且将SYN置为1，而且会产生ACK字段，ACK字段数值是在客户端发送过来的序列号seq的基础上加1进行回复，以便客户端收到信息时，知晓自己的TCP建立请求已得到验证。（SYN=1，ACK=x+1，seq=y，y为随机生成数值）这里的ack加1可以理解为是确认和谁建立连接。

3、客户端收到服务端发送的TCP建立验证请求后，会使自己的序列号加1表示，并且再次回复ACK验证请求，在服务端发过来的seq上加1进行回复。（SYN=1，ACK=y+1，seq=x+1）

为什么连接的时候是三次握手，关闭的时候却是四次握手?

答：因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉Client端，"你发的FIN报文我收到了"。只有等到我Server端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。

4: 四次挥手

四次挥手过程说明：

1、客户端发送断开TCP连接请求的报文，其中报文中包含seq序列号，是由发送端随机生成的，并且还将报文中的FIN字段置为1，表示需要断开TCP连接。（FIN=1，seq=x，x由客户端随机生成）

2、服务端会回复客户端发送的TCP断开请求报文，其包含seq序列号，是由回复端随机生成的，而且会产生ACK字段，ACK字段数值是在客户端发过来的seq序列号基础上加1进行回复，以便客户端收到信息时，知晓自己的TCP断开请求已经得到验证。（FIN=1，ACK=x+1，seq=y，y由服务端随机生成）

3、服务端在回复完客户端的TCP断开请求后，不会马上进行TCP连接的断开，服务端会先确保断开前，所有传输到A的数据是否已经传输完毕，一旦确认传输数据完毕，就会将回复报文的FIN字段置1，并且产生随机seq序列号。（FIN=1，ACK=x+1，seq=z，z由服务端随机生成）

4、客户端收到服务端的TCP断开请求后，会回复服务端的断开请求，包含随机生成的seq字段和ACK字段，ACK字段会在服务端的TCP断开请求的seq基础上加1，从而完成服务端请求的验证回复。（FIN=1，ACK=z+1，seq=h，h为客户端随机生成）

至此TCP断开的4次挥手过程完毕

5: apache和nginx

nginx 相对 apache 的优点：

- 轻量级，同样起web 服务，比apache 占用更少的内存及资源
- 抗并发，nginx 处理请求是异步非阻塞的，支持更多的并发连接，而 apache 则是阻塞型的，在高并发下nginx 能保持低资源低消耗高性能
- 配置简洁
- 高度模块化的设计，编写模块相对简单
- 社区活跃

apache 相对nginx 的优点：

- rewrite ，比nginx 的rewrite 强大
- 模块超多，基本想到的都可以找到

- 少bug ， nginx 的bug 相对较多
- 超稳定

6: http和https

一、HTTP和HTTPS的基本概念

HTTP：是互联网上应用最为广泛的一种网络协议，是一个客户端和服务器端请求和应答的标准（TCP），用于从WWW服务器传输超文本到本地浏览器的传输协议，它可以使浏览器更加高效，使网络传输减少。

HTTPS：是以安全为目标的HTTP通道，简单讲是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。

HTTPS协议的主要作用可以分为两种：一种是建立一个信息安全通道，来保证数据传输的安全；另一种就是确认网站的真实性。

二、HTTP与HTTPS有什么区别?

HTTP协议传输的数据都是未加密的，也就是明文的，因此使用HTTP协议传输隐私信息非常不安全，为了保证这些隐私数据能加密传输，于是网景公司设计了SSL（Secure Sockets Layer）协议用于对HTTP协议传输的数据进行加密，从而就诞生了HTTPS。

简单来说，HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，要比http协议安全。

HTTPS和HTTP的区别主要如下：

1、https协议需要到ca申请证书，一般免费证书较少，因而需要一定费用。

2、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。

3、http和https使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443。

4、http的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比http协议安全。

7: arp 协议

一、ARP概述

如果要在TCP/IP协议栈中选择一个**"最不安全的协议"**，那么我会毫不犹豫把票投给ARP协议。我们经常听到的这些术语，包括"网络扫描"、"内网渗透"、"中间人拦截"、"局域网流控"、"流量欺骗"，基本都跟ARP脱不了干系。大量的安全工具，例如大名鼎鼎的Cain、功能完备的Ettercap、操作傻瓜式的P2P终结者，底层都要基于ARP实现。

听上去这么"逆天"的协议，其实技术原理又简单的难以置信，例如ARP整个完整交互过程仅需要两个包，一问一答即可搞定！但是ARP协议也有它令初学者迷惑的地方，例如由它本身延伸出来的"代理ARP"、"免费ARP"、"翻转ARP"、"逆向ARP"，而这些不同种类的ARP，又应用于不同的场景。好吧，在深入到技术原理之前，作为初学者，我们先记住下面三句话：

①ARP（Address Resolution Protocol）即地址解析协议， 用于实现从IP 地址到 MAC 地址的映射，即询问目标IP对应的MAC地址。

②在网络通信中，主机和主机通信的数据包需要依据OSI模型从上到下进行数据封装，当数据封装完整后，再向外发出。所以在局域网的通信中，不仅需要源目IP地址的封装，也需要源目MAC的封装。

③一般情况下，上层应用程序更多关心IP地址而不关心MAC地址，所以需要通过ARP协议来获知目的主机的MAC地址，完成数据封装。

8: socket

Socket接口是TCP/IP网络的API（Application Programming Interface,应用程序编程接口），Socket接口定义了许多函数或例程，程序员可以用它们来开发TCP/IP网络上的应用程序。

9: 浏览器缓存

10：soap 和 rpc

11: restful api. 架构

TCP/IP分别在模型的哪一层；

socket长连接是什么意思；

select和epoll你了解么，区别在哪；

TCP UDP区别；三次握手四次挥手讲一下；

TIME_WAIT过多是因为什么；

12:http一次连接的全过程：

HTTP通信机制是在一次完整的HTTP通信过程中，Web浏览器与Web服务器之间将完成下列7个步骤：

1. 建立TCP连接

在HTTP工作开始之前，Web浏览器首先要通过网络与Web服务器建立连接，该连接是通过TCP来完成的，该协议与IP协议共同构建Internet，即著名的TCP/IP协议族，因此Internet又被称作是TCP/IP网络。HTTP是比TCP更高层次的应用层协议，根据规则，只有低层协议建立之后才能，才能进行更层协议的连接，因此，首先要建立TCP连接，一般TCP连接的端口号是80。

2. Web浏览器向Web服务器发送请求命令

一旦建立了TCP连接，Web浏览器就会向Web服务器发送请求命令。例如：GET/sample/hello.jsp HTTP/1.1。

3. Web浏览器发送请求头信息

浏览器发送其请求命令之后，还要以头信息的形式向Web服务器发送一些别的信息，之后浏览器发送了一空白行来通知服务器，它已经结束了该头信息的发送。

4. Web服务器应答
客户机向服务器发出请求后，服务器会客户机回送应答， HTTP/1.1 200 OK ，应答的第一部分是协议的版本号和应答状态码。
5. Web服务器发送应答头信息
正如客户端会随同请求发送关于自身的信息一样，服务器也会随同应答向用户发送关于它自己的数据及被请求的文档。
6. Web服务器向浏览器发送数据
Web服务器向浏览器发送头信息后，它会发送一个空白行来表示头信息的发送到此为结束，接着，它就以Content-Type应答头信息所描述的格式发送用户所请求的实际数据。
7. Web服务器关闭TCP连接
一般情况下，一旦Web服务器向浏览器发送了请求数据，它就要关闭TCP连接，然后如果浏览器或者服务器在其头信息加入了这行代码：
Connection:keep-alive
TCP连接在发送后将仍然保持打开状态，于是，浏览器可以继续通过相同的连接发送请求。保持连接节省了为每个请求建立新连接所需的时间，还节约了网络带宽。
你来说下从用户发起request——到用户接收到response；
http连接方式。get和post的区别，你还了解其他的方式么；

13：restful你知道么； 状态码你知道多少， 比如200/403/404/504等等；

200 – OK – 一切正常
201 – OK – 新资源已经被创建
204 – OK – 资源删除成功
304 – 没有变化，客户端可以使用缓存数据
400 – Bad Request – 调用不合法，确切的错误应该在error payload中描述，例如：“JSON 不合法 ”
401 – 未认证，调用需要用户通过认证
403 – 不允许的，服务端正常解析和请求，但是调用被回绝或者不被允许
404 – 未找到，指定的资源不存在
422 – 不可指定的请求体 – 只有服务器不能处理实体时使用，比如图像不能被格式化，或者重要字段丢失。
500 – Internal Server Error – 标准服务端错误，API开发人员应该尽量避开这种错误

13浏览器发起请求到响应的过程：

一步，解析域名，找到ip

浏览器会缓存DNS一段时间，一般2–30分钟不等，如果有缓存，直接返

回ip，否则下一步。
缓存中无法找到ip，浏览器会进行一个系统调用，查询hosts文件。如果找到，直接返回ip，否则下一步。
进行1 和2 本地查询无果，只能借助于网络，路由器一般都会有自己的DNS缓存，ISP服务商DNS缓存，这时一般都能够得到相应的ip，如果还是无果，只能借助于DNS递归解析了。
这时ISP的DNS服务器就会开始从根域名服务器开始递归搜索，从.com顶级域名服务器，到baidu的域名服务器。
到这里，浏览器就获得网络ip，在DNS解析过程中，常常解析出不同的IP。

第二步，浏览器于网站建立TCP连接
浏览器利用ip直接网站主机通信，浏览器发出TCP连接请求，主机返回TCP应答报文，浏览器收到应答报文发现ACK标志位为1，表示连接请求确认，浏览器返回TCP（）确认报文，主机收到确认报文，三次握手，TCP连接建立完成。

第三步， 浏览器发起默认的GET请求
浏览器向主机发起一个HTTP-GET方法报文请求，请求中包含访问的URL，也就是http://www.baidu.com/还有User-Agent用户浏览器操作系统信息，编码等，值得一提的是Accep-Encoding和Cookies项。Accept-Encoding一般采用gzip，压缩之后传输html文件，Cookies如果是首次访问，会提示服务器简历用户缓存信息，如果不是，可以利用Cookies对应键值，找到相应缓存，缓存里面存放着用户名，密码和一些用户设置项

第四步，显示页面或返回其他
返回状态码200 OK，表示服务器可以响应请求，返回报文，由于在报头中Content-type为“text/html”,浏览器以HTML形式呈现，而不是下载文件。
但是对于大型网站存在多个主机站点，往往不会直接返回请求页面，而是重定向。返回的状态码就不是 200 OK， 而是301,302以3开头的重定向吗。浏览器在获取了重定向响应后，在响应报文中Location项找到重定向地址，浏览器重新第一步访问即可。

三：Django框架部分
都是让简单的介绍下你在公司的项目，不管是不是后端相关的，主要是要体现出你干了什么； 你在项目中遇到最难的部分是什么，你是怎么解决的；
你看过django的admin源码么；
看过flask的源码么；
你如何理解开源；

MVC / MTV的区别

MVC各部分的功能

M全拼为Model，主要封装对数据库层的访问，对数据库中的数据进行增、删、改、查操作。

V全拼为View，用于封装结果，生成页面展示的html内容。

C全拼为Controller，用于接收请求，处理业务逻辑，与Model和View交互，返回结果。

MVT各部分的功能

M全拼为Model，与MVC中的M功能相同，负责和数据库交互，进行数据处理。

V全拼为View，与MVC中的C功能相同，接收请求，进行业务处理，返回应答。

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。

优点:

高可扩展性

向后兼容：后面的版本都可以兼容

低耦合： 模块与模块之间不要有太强的依耐性

高内聚： 指一个软件模块是由相关性很强的代码组成，只负责一项任务，也就是常说的单一责任原则。

缓存怎么用;

1.缓存的简介

在动态网站中,用户所有的请求,服务器都会去数据库中进行相应的增,删,查,改,渲染模板,执行业务逻辑,最后生成用户看到的页面.

当一个网站的用户访问量很大的时候,每一次的的后台操作,都会消耗很多的服务端资源,所以必须使用缓存来减轻后端服务器的压力.

缓存是将一些常用的数据保存内存或者memcache中,在一定的时间内有人来访问这些数据时,则不再去执行数据库及渲染等操作,而是直接从内存或memcache的缓存中去取得数据,然后返回给用户.

2.Django提供了6种缓存方式

1. 开发调试缓存
2. 内存缓存
3. 文件缓存
4. 数据库缓存
5. Memcache缓存(使用python-memcached模块)
6. Memcache缓存(使用pylibmc模块)

经常使用的有文件缓存和Mencache缓存

中间件是干嘛的?

在http请求 到达视图函数之前　和视图函数return之后，django会根据自己的规则在合适的时机执行中间件中相应的方法

django的生命周期是:前端请求--->nginx--->uwsgi.--->中间件--->url路由----->view试图--->orm---->拿到数据返回给view---->试图将数据渲染到模版中拿到字符串---->中间件--->uwsgi---->nginx---->前端渲染。

在有些场合，需要对Django处理的每个request都执行某段代码。这类代码可能是在view处理之前修改传入的request，或者记录日志信息以便于调试，等等。

这类功能可以用Django的中间件框架来实现，该框架由切入到Django的request/response处理过程中的钩子集合组成。这个轻量级低层次的plug-in系统，能用于全面的修改Django的输入和输出。

```
from django.utils.deprecation import MiddlewareMixin
```

中间件中一共有四个方法:

process_request(self,request) 此方法用于在请求到来时处理请求

peocess_view(self,request,fnc,arg,kwarg)在本次将要执行的view函数被调用前处理请求

process_response(self,request,response)在执行完view函数准备发送响应到客户端之前执行

process_exception(self,request,exception)View函数在抛出异常的时候该函数被调用,得到的exception参数是实际上跑出的异常实例.通过此方法可以进行很好的错误控制,提供有好的用户界面

5.process_template_response 在视图view刚好执行完毕之后调用

CSRF是什么?

django是如何避免的；XSS呢；

如果你来设计login，简单的说一下思路;

- 用户名
- 姓名
- 性别
- 密码
- 手机号
- 创建时间
- 更新时间
- 是否有效

- 登录时，就不要考虑输入内容的多少问题了，已经种了注册的因，就享受登录的果就是了。所以，尽可能的减少用户要输入的内容数量，是简化登录流程的一大利器。通常有如下的方法：
- 使用第三方登录，谁用谁知道。毫无疑问，不用输入繁琐的账号和密码，只需要轻轻松松点两下，就能愉快地登录到系统上，和乐而不为呢？妈妈再也不用担心我记不住那些长长地账号和密码啦！
- 对于必须要使用手机号才能使用的产品(比如，滴滴打车)，那么，只需要一个输入手机号的输入框，然后愉快地获取短信验证码登录吧，即使忘记了密码也不怕。
- 能不使用验证码，就不使用验证码。如果用户多次输入均登录失败的情况下，再考虑让用户输入验证码。

  请务必在登录界面加载成功后，友好地提示用户，当前网络状态是否OK，别等着用户一切输入正确了，美滋滋地准备享受产品带给TA的快乐时，给了他一个晴天霹雳——你的网络不在服务区！

  千万别想着等用户输入密码错误了几次之后，再提示用户去找回密码，很有可能用户一开始就忘了密码呢?

  为了节省用户的输入，很多产品通常都会设计保存用户名和密码到本地的功能，这样用户下次登录时，只要输入账户就能直接读取出来密码并登录，减少了用户输入的次数或者因为忘记密码而造成的困扰，比如QQ的PC端。这种模式不能单纯地评判好坏，最好是根据实际的情况，来考虑要不要保存密码到本地。其实，现在很多的产品(尤其是移动端产品)都不会保存用户的密码到本地，这主要是因为防止其他人能够直接登录用户的账户而操作一些非用户本人想操作的事情，造成不必要的损失。

  用户输入几次错误后提示用户去“找回密码”？请根据实际情况来设计这个功能。
-

session和cookie的联系与区别；session为什么说是安全的；

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，会比较占用你服务器的性能

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、所以个人建议：
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

uWSGI和Nginx的作用；（我发现基本不问django实现细节相关的东西。。或者问也问的很少，哎，之前准备的方向完全错了）
一个请求从发出到响应的过程

四：数据库部分

MySQL锁有几种；死锁是怎么产生的；为何，以及如何分区、分表；
MySQL的char varchar text的区别；了解join么，有几种，有何区别，A LEFT JOIN B，查询的结果中，B没有的那部分是如何显示的（NULL）；
索引类型有几种，
BTree索引和hash索引的区别（我没答上来这俩在磁盘结构上的区别）；
手写：如何对查询命令进行优化；
NoSQL了解么，和关系数据库的区别；
redis有几种常用存储类型；

简述一下你熟悉的NOSQL，它有什么优点和缺点？
redis:
优点：
- 读写性能优异；
- 支持数据持久化，支持AOF和RDB两种持久化方式；
- 支持主从复制，主机会自动将数据同步到从机，可以进行读写分离；
- 数据结构丰富：除了支持string类型的value外还支持string、hash、set、sortedset、list等数据结构。

缺点：
- Redis不具备自动容错和恢复功能，主机从机的宕机都会导致前端部分读写请求失败，需要等待机器重启或者手动切换前端的IP才能恢复；
- 主机宕机，宕机前有部分数据未能及时同步到从机，切换IP后还会引入数据不一致的问题，降低了系统的可用性；
- Redis的主从复制采用全量复制，复制过程中主机会fork出一个子进程对内存做一份快照，并将子进程的内存快照保存为文件发送给从机，这一过程需要确保主机有足够多的空余内存。若快照文件较大，对集群的服务能力会产生较大的影响，而且复制过程是在从机新加入集群或者从机和主机网络断开重连时都会进行，也就是网络波动都会造成主机和从机间的一次全量的数据复制，这对实际的系统运营造成了不小的麻烦；

五：linux部分

讲一下你常用的Linux/git命令和作用； 查看当前进程是用什么命令，除了文件相关的操作外，你平时还有什么操作命令； （因为我本人Linux本身就很水，只会基本的操作，所以这部分面试官也基本没怎么问。。反正问了就大眼瞪小眼呗）

六：设计模式

mvc的优缺点

它实现了显示模块与功能模块的分离。提高了程序的可维护性、可移植性、可扩展性与可重用性，降低了程序的开发难度。

二、MVC的优点
1、可以为一个模型在运行时同时建立和使用多个视图。变化-传播机制可以确保所有相关的视图及时得到模型数据变化，从而使所有关联的视图和控制器做到行为同步。
2、视图与控制器的可接插性，允许更换视图和控制器对象，而且可以根据需求动态的打开或关闭、甚至在运行期间进行对象替换。
3、模型的可移植性。因为模型是独立于视图的，所以可以把一个模型独立地移植到新的平台工作。需要做的只是在新平台上对视图和控制器进行新的修改。
4、潜在的框架结构。可以基于此模型建立应用程序框架，不仅仅是用在设计界面的设计中。

三、MVC的不足之处
1、增加了系统结构和实现的复杂性。对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。
2、视图与控制器间的过于紧密的连接。视图与控制器是相互分离，但确实联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。
3、视图对模型数据的低效率访问。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损

害操作性能。
4、目前，一般高级的界面工具或构造器不支持模式。改造这些工具以适应MVC需要和建立分离的部件的代价是很高的，从而造成MVC使用的困难。

mvc于mvt

Mvt模式主要是mvc模式在Django中的应用，原理通mvc, 降低耦合度，提高内聚

MVC模式

M全拼为Model,主要封装对数据库层的访问，对数据库中的数据进行增删改查操作。

V全拼为View,用于封装结果，生成页面显示的html页面。

C全拼为Controller,用于接收请求，处理业务逻辑，与Model和View交互，返回结果。

MVT模式

M全拼为Model,与MVC中的M功能相同，负责和数据库交互，进行数据处理。

V全拼为View,与MVC中的C功能相同，接收请求，进行业务处理，返回相应。

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

单例模式

单例模式是日常应用中最广泛的模式了，其目的就是令到单个进程中只存在一个类的实例，从而可以实现数据的共享，节省系统开销，防止io阻塞等等
但是在多进程的应用中，单例模式就实现不了了，例如一些web应用，django，这些，因为会启动多条进程来监听http请求，这样的会通过单例模式是实现不了数据共享的，也就是实现不了单例模式的目的了，这时需要用进程间通信方法来实现数据共享，当然也可以尝试使用redis这些nosql数据库实现数据共享，因为它们的读取数据较快。

工厂方法

简单工厂模式的最大优点就是工厂类内部包含了必要的逻辑判断，客户端只要提供选择条件即可，这样客户端就不需要知道具体产品的信息了。
但是每次增加产品就要修改工厂类，违背了“开放-封闭”原则。
所以，有了工厂方法模式
工厂方法模式：定义一个用于创建对象的接口，让子类决定实例化哪一个类。

工厂方法模式相对于简单工厂模式的
优点在于，抽象了产品工厂这个类，让它变成了一个接口，只要某个类实现了这个接口，它就可以被当做工厂类来用，这样每添加一个产品的时候，就添加一下相应的生产工厂类，其它地方就可以使用了，满足“开放-封闭”原则；
缺点在于，把生产产品的逻辑判断从工厂中剥离了出去。

代理模式

代理模式给某一个对象提供一个代理对象，并由代理对象控制对原对象的引用。通俗的来讲代理模式就是我们生活中常见的中介。

意图：为其他对象提供一种代理以控制对这个对象的访问。
主要解决：在直接访问对象时带来的问题，比如说：要访问的对象在远程的机器上。在面向对象系统中，有些对象由于某些原因（比如对象创建开销很大，或者某些操作需要安全控制，或者需要进程外的访问），直接访问会给使用者或者系统结构带来很多麻烦，我们可以在访问此对象时加上一个对此对象的访问层。
何时使用：想在访问一个类时做一些控制。
如何解决：增加中间层。

代理模式的应用场景：
1.远程代理，也就是为一个对象在不同的地址空间提供局部代表。这样可以隐藏一个对象存在于不同地址空间的事实。
2.虚拟代理，是根据需要创建开销大的对象。通过它来存放实例化需要很长时间的真是对象，例如html中，图片需要load很久，所以通过虚拟代理来代替真实的图片
3.安全代理，用来控制真实对象访问时的权限
4.智能指引，是指当调用真实的对象时，代理处理另外一些事

观察者设计模式

观察者模式是一个软件设计模式，一个主题对象博包涵一系列依赖他的观察者，自动通知观察者的主题对象的改变，通常会调用每个观察者的一个方法。这个设计模式非常适用于分布式事件处理系统。
1.发布者类应该包涵如下方法：

- 注册能够接收通知的对象
- 从主对象到注册对象，通知任何变化
- 未注册对象不能够接收任何通知信息

当一个抽象模型有两个方面，其中一个方面依赖于另一个方面。
当对一个对象的改变需要同时改变其它对象，而不知道具体有多少个对象待改变。
当一个对象必须通知其它对象，而它又不能假定其它对象是谁。换句话说，你不希望这些对象是紧密耦合的。

- BootStrap问题
  - 1、为什么使用bootstrap?

- 如果能指出浏览器支持情况、移动优先、响应式布局、简单易写、统一编码风格的话，那基本算过了。
- 2、什么是bootstrap栅格系统?
- 听到**12**字样就算过了，如果自己会解释为什么是12格，而不是14、16格，那就说明对栅格系统有很深的了解。
- 3、编写一段内联、水平、垂直表单代码?
- 4、编写一段响应式表格代码?
- 3、4主要是看编码的风格包括：熟练程度、空格缩进、DOM标签选择这类细节。
- 大概如此吧！

自我介绍

这是一道送分题，万年不变的第一个问题。不过有些小伙伴可能没有太在意，其实这个问题已经在面试官心中决定了你的去留意向。自我介绍的主要结构：个人基本信息 + 基本技术构成 + 项目经验（具体项目以及在项目中的负责部分）+ 自我评价，其中的原则就是紧紧围绕招聘岗位的需求做介绍。在此之前要做好准备工作，看看招聘方具体需要什么方向的研发工程师。目前针对Python，拉勾上的招聘多为自动化测试平台的设计与开发、数据的挖掘与清洗。单纯的web开发好像还没有，所以web方向的同学注意，多和运维以及自动化方面靠拢。

二段式询问

在面试的过程当中，在面试官提出问题的时候，往往会就问题本身引申出较深层次的问题。比如：你使用过with语句吗？我的回答是：with语句经常适用于对资源进行访问的场合，确保在访问的过程中不管是否发生异常都会指执行必要的清理操作，比如文件的自动关闭以及线程中锁的自动获取与释放。面试官紧接着问，那你知道为什么with语句能够使文件正确关闭，一下子把我问闷了，只能依稀记得with语句会开辟出一块独立环境来执行文件的访问，类似沙盒机制。面试官对这个答案不置可否，算是勉强通过了。所以知其然更要知其所以然。在平时的学习中，多问一个为什么，面试的时候就不会太被动。

不要给自己挖坑

确保你在回答面试官的过程中，回答中的每个知识点都了然于胸，不然被问住，是很难堪的。我在回答web安全问题时，顺嘴说了SQL注入，面试官说既然提到了SQL注入，那么你讲讲它的原理及解决方法吧！丢脸的是我竟然把XSS跨站注入攻击和SQL注入搞混了，场面也是有点尴尬。所以斟酌你说的每一句话，聪明点的同学还可以引导面试官，让他问出自己想要被问的问题。

必问到Redis，高并发解决办法

面试了好多家公司，必然问道Redis了解多少，高并发的解决办法。笔者回答的都不是很好。

这一年你学习了什么新的技能

这是面试官在考察你是否对于新鲜技术抱有极大热忱。面试我的面试官无一例外都问到了这个问题。他们都希望能找一个不断学习，开括创新的年轻人。多浏览最新的技术资讯，选择一方面自己感兴趣的领域。

你会选择创业公司还是像BAT那样的大公司，为什么?

当然是看招聘方属于哪一个公司啦，不过问这种问题的一般都是创业公司。答案无非是：挑战大，享受挑战；创业公司具有无限成功的可能性，想随公司一起成长；

为什么你要从上一家公司离职?

这也是一个必问问题，找一个比较正当的理由，不要说什么公司零食太多胖了20斤，公司周别附近的外卖都吃腻了，真的别这样说...主要原则就是不要对前公司抱有怨言，BOSS朝令夕改，PM不靠谱什么的，多寻找自身原因：公司发展比较稳定，但我还年轻，希望有更大的挑战和更多的学习机会。像这样就可以。

描述一下你的上一家公司

这个问题问到的几率不太大，不过也还是有三家公司问到过，招聘方主要想从上一家公司的具体经营规模以及主营业务来定位你的水平，知道招聘方的目的就可以从容应答。

遇到过什么问题，怎么解决的?