1.单引号，双引号，三引号的区别：

1).单引号和双引号主要用来表示字符串

2).三引号

三单引号:'''python ''',也可以表示字符串,一般用来输入多行文本,或者用于大段的注释。

三双引号： """python"""，一般**用在类里面,用来注释类**,这样省的写文档,直接用类的对象__doc__访问获得文档。

2.写一个函数，输入一个字符串，返回倒序排列的结果

输入: string_reverse(‘abcdef’), 返回: ‘fedcba’ ,写出你能想到的多种方法：

```
def string_reverse(text="abcdef"):
 return text[::-1]
print(string_reverse())
-----------------------------------
fedcba
```

```
def string_reverse(text="abcdef"):
 new_text=list(text)
 new_text.reverse()
 return ''.join(new_text)

print(string_reverse())
--------------------------------
fedcba
```

3.按升序合并如下两个list，并去除重复的元素：

list1 = [2, 3, 8, 4, 9, 5, 6]

list2 = [5, 6, 10, 17, 11, 2]

```
list1 = [2, 3, 8, 4, 9, 5, 6]
list2 = [5, 6, 10, 17, 11, 2]
list3=list1+list2
print(sorted(list(set(list3))))
-------------------------------------------
[2, 3, 4, 5, 6, 8, 9, 10, 11, 17]
```

4.下面的代码会不会报错?

```
>>> list = ['a', 'b', 'c', 'd', 'e']
```

```
>>> print(list[10:])
[]
```

5.说出下面list1,list2,list3的输出值

```
#新的默认列表仅仅只在函数被定义时创建一次。随后当 extendList没有
#被指定的列表参数调用的时候，其使用的是同一个列表。
#这就是为什么当函数被定义的时候，表达式是用默认参数被计算，而不是它被调用的时候。
#因此，list1 和 list3 是操作的相同的列表。
#而list2是操作的它创建的独立的列表（通过传递它自己的空列表作为list参数的值）
def extendList(val, list=[]):
 list.append(val)
 return list
list1 = extendList(10)
list2 = extendList(123,[])
list3 = extendList('a')
print("list1 = %s" % list1)
print("list2 = %s" % list2)
print("list3 = %s" % list3)
----------------------------------------
list1 = [10, 'a']
list2 = [123]
list3 = [10, 'a']
```

```
def extendList(val, list=None):
 if list is None:
 list = []
 list.append(val)
 return list
list1 = extendList(10)
list2 = extendList(123,[])
list3 = extendList('a')
print("list1 = %s" % list1)
print("list2 = %s" % list2)
print("list3 = %s" % list3)
```

```
----------------------------------------
list1 = [10]
list2 = [123]
list3 = ['a']
```

5.阅读代码，给出代码运行的结果。

```
a0=dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
a1=range(10)
a2=[i for i in a1 if i in a0]
a3=[a0[s] for s in a0]
a4=[i for i in a1 if i in a3]
a5={i:i*i for i in a1}
a6=[[i,i*i] for i in a1]
print(a0)
print(a1)
 print(a2)
print(a3)
print(a4)
print(a5)
print(a6)
----------------------------------------
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
range(0, 10)
[]
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64,
 9: 81}

[[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25],
 [6, 36], [7, 49], [8, 64], [9, 81]]
```

6.一行代码实现对列表a 中偶数位置的元素加3后求和?

a = [1,2,3,4,5]

```
list1=[1,2,3,4,5]
sums=sum(map(lambda x: x+3,list1[1::2]))
```

```
print(sums)
-----------------------
12
```

7.将列表a的元素顺序打乱，再对a 进行排序得到列表b，
然后将a和b按元素顺序构造一个字典d.

```
from random import shuffle
a=[1,2,3,4,5]
shuffle(a)
b=sorted(a,reverse=True)
d=dict(zip(a,b))
print(d)
-----------------------------------
{5: 5, 1: 4, 3: 3, 4: 2, 2: 1}
```

8.乘法表

```
for i in range(1,10):
 for j in range(1,i+1):
 print("%d*%d=%2d" % (i,j,i*j),end=" ")
 print(" ")
-------------------------------------------------
1*1= 1
2*1= 2 2*2= 4
3*1= 3 3*2= 6 3*3= 9
4*1= 4 4*2= 8 4*3=12 4*4=16
 5*1= 5 5*2=10 5*3=15 5*4=20 5*5=25
 6*1= 6 6*2=12 6*3=18 6*4=24 6*5=30 6*6=36
 7*1= 7 7*2=14 7*3=21 7*4=28 7*5=35 7*6=42 7*7=49
 8*1= 8 8*2=16 8*3=24 8*4=32 8*5=40 8*6=48 8*7=56 8*8=64
 9*1= 9 9*2=18 9*3=27 9*4=36 9*5=45 9*6=54 9*7=63 9*8=72 9*9=81
```

9.阶乘

```
from functools import reduce
a=3
b=reduce(lambda x,y:x*y,range(1,a+1))
print(b)
-----------------------
6
```

10.斐波那契数列（1-100的斐波那契数列）

```
a=0
b=1
while b < 100:
 print(b,end=",")
 a,b = b,a+b
--------------------------------
1,1,2,3,5,8,13,21,34,55,89,
```

11.一行代码实现1---100之和

```
a=sum(range(0,101))
print(a)
----------------------
5050
```

12.列表[1,2,3,4,5],请使用map()函数输出[1,4,9,16,25],并使用列表推导式提取出大于10的数，最终输出[16,25]

```
list1= [1,2,3,4,5]
def fn(x):
 return x**2

result = map(fn,list1)
result = [i for i in result if i > 10]
print(result)
--------------------------
[16, 25]
```

13.冒泡排序:

```
#冒泡排序
def bubble_sort(lists):
 len_list=len(lists)
 for i in range(len_list):
 for j in range(len_list-i-1):
 if lists[j] > lists[j+1]:
 lists[j],lists[j+1] = lists[j+1],lists[j]
 print(lists)
 return lists

```

```
lists=[45,2,3,78,1,96,23,56]
bubble_sort(lists)
-----------------------------------------------
[2, 3, 45, 1, 78, 23, 56, 96]
[2, 3, 1, 45, 23, 56, 78, 96]
[2, 1, 3, 23, 45, 56, 78, 96]
[1, 2, 3, 23, 45, 56, 78, 96]
[1, 2, 3, 23, 45, 56, 78, 96]
[1, 2, 3, 23, 45, 56, 78, 96]
[1, 2, 3, 23, 45, 56, 78, 96]
[1, 2, 3, 23, 45, 56, 78, 96]
```

14.删除列表中重复的数字：

```
l=[1,1,6,3,1,5,2]
def duplictae(lists):
 L=[]
 for i in lists:
 if i not in L:
 L.append(i)
 return L
print(duplictae(l))
--------------------------
[1, 6, 3, 5, 2]
```

15.以下代码的输出结果：

```
def f(x,l=[]):
 for i in range(x):
 l.append(i*i)
 print(l)
------------------------
>>> f(2)
[0, 1]
>>> f(3,[3,2,1])
[3, 2, 1, 0, 1, 4]
 >>> f(3)
 [0, 1, 0, 1, 4]
```

16.List = [-2,1,3,-6],如何实现以绝对值大小从小到大将List中的内容排序?

```
list1 = [-2,1,3,-6]
print(sorted(list1,key=abs))
-----------------------------------
[1, -2, 3, -6]
```

17.合并两个list?

```
# 方法1:
list1=[2,3,8]
list2=[5,6,10]
def list_union(list1,list2):
 for i in list2:
 list1.append(i)
 return(list1)
a=list_union(list1,list2)
print(a)
 -------------------------------
 [2, 3, 8, 5, 6, 10]
 -------------------------------
 # 方法二:
 list1=[2,3,8]
 list2=[5,6,10]
 def list_union(list1,list2):
  return(list1+list2)

 a=list_union(list1,list2)
 print(a)
 ----------------------------
 [2, 3, 8, 5, 6, 10]
```

18.lambda函数?

lambda函数是python中的匿名函数。它语法简单，简化代码，不会产生命名冲突，污染命名空间。

19.python在函数编程方面提供了函数和语法?

函数：map,reduce,filter等函数。

语法：装饰器，闭包等。

20.list对象alist[{'name':'a','age':20},{'name':'b','age':30},{'name':'c','age':25}]请按照alist中元素的age由大到小排序?

```
alist=[{'name':'a','age':20},{'name':'b','age':30},
{'name':'c','age':25}]
a=sorted(alist,key=lambda x:x['age'],reverse=True)
print(a)
----------------------------------------
[{'name': 'b', 'age': 30}, {'name': 'c', 'age': 25}, {'name':
'a', 'age': 20}]
```

21.将字符串“k:1|k1:2|k2:3|k3:4”处理成python字典

{ ‘k':'1','k1':'2','k2':'3','k3':'4'}?

2019/12/05 08:59

```
str1 = "k:1|k1:2|k2:3|k3:4"
str_list=str1.split("|")
print(str_list)
d={}
for i in str_list:
 key,value = i.split(":")
 d[key]=value
print(d)
----------------------------
['k:1', 'k1:2', 'k2:3', 'k3:4']
{'k': '1', 'k1': '2', 'k2': '3', 'k3': '4'}
```

22.简述位和字节之间的关系?

位：计算机的计算单位，代表0或者1.

字节：一字节相当于8位。

23.简述ascii,unicode,utf-8,gbk的关系?

- ascii 是最早美国用的标准信息交换码，把所有的字母的大小写，各种符号用 二进制来表示，共有256种，加入些拉丁文等字符，1bytes代表一个字符。
- Unicode是为了统一世界各国语言的不用，统一用2个bytes代表一个字符，可以表达2**16=65556个，称为万国语言，特点：速度快，但浪费空间。可以用在内存处理中，兼容了utf-8，gbk，ASCII。
- utf-8 为了改变Unicode的这种缺点，规定1个英文字符用1个字节表示，1个中文字符用3个字节表示，特点；节省空间，速度慢，用在硬盘数据传输，网络数据传输，相比硬盘和网络速度，体现不出来的。
- gbk 是中文的字符编码，用2个字节代表一个字符。

24.请写出“王港”分别用utf-8和 gbk编码所占的位数?

```
name="王港"
print(len(bytes(name,encoding='utf-8')))
print(len(bytes(name,encoding='gbk')))
----------------------------
6
4
```

25.声明变量时的注意事项?

- 由字母，数字和下划线构成，不能以数字开头，不能任意特殊字符
- 变量定义规范，使用驼峰式或者下划线式格式
- 变量定义尽量简明，易懂，方便使用者应用

26.利用内置函数chr(),ord(),以及random模块写一个简单随机4位验证码?

```
import random
tmp = '' #最后生成的随机码
for i in range(4):
 n=random.randrange(0,2) #生成随机数1或0，用来判断下面是生成随机数字还是字母
 if n == 0:
 num = random.randrange(65,91) #为0时，生成大写字母
 tmp +=chr(num)
 else:
 k = random.randrange(0,10) #为1时，生成数字
  tmp +=str(k)
print(tmp)
-----------------------------------
2XK7
```

27.执行python程序时，自动生成的.pyc文件的作用是什么?

python执行前生成的编译字节码文件。

28.实现用户输入用户名和密码，当密码为123用户名为seven时，显示登录成功，否则登录失败。

```
username = input("请输入用户名：")
password = input("请输入密码：")
if username == 'seven' and password == '123':
```

```
 print('登录成功')
else:
 print('登录失败')
----------------------------------
请输入用户名：seven
请输入密码：123
登录成功
```

29.实现用户输入用户名和密码，当密码为123用户名为seven时，显示登录成功，否则登录失败，失败时允许重新输入三次?

```
error_num = 0
while True:
 username = input("请输入用户名：")
 password = input("请输入密码：")
 if username == 'seven' and password == '123':
 print('登录成功')
 break
 else:
 print('登录失败')
  error_num +=1
  if error_num == 3:
  exit()
  else:
  continue
-----------------------------------------------------
```

30.实现用户输入用户名和密码，当密码为123且用户名为seven或者alex时，显示登录成功，否则登录失败，失败时允许重新输入三次?

```
error_num = 0
while True:
 username = input("请输入用户名：")
 password = input("请输入密码：")
 if username == 'seven' or username == 'alex' and password == '123':
 print('登录成功')
 break
 else:
```

```
 print('登录失败')
  error_num +=1
  if error_num == 3:
  exit()
  else:
  continue
 ----------------------------------------------------
请输入用户名：1
请输入密码：1
登录失败
请输入用户名：1
请输入密码：1
登录失败
请输入用户名：seven
请输入密码：123
登录成功
>>>
```

31.使用while循环实现输出2-3+4-5+6.....+100的和?

```
s = 0
for i in range(2,101):
 if i % 2 == 0:
 s +=i
 else:
 s -=i

print('sum:',s)
--------------------------
 51
 ------------------方法二--------------------------
 i=2
 s = 0
 while i < 101:
  if i % 2 ==0:
  s +=i
  else:
```

```
 s -=i
 i +=1
print(s)
-----------------
51
```

32.求1-2+3-4+5...+99的和?

```
------------------方法1--------------------
s = 0
for i in range(1,100):
 if i % 2 == 0:
 s -=i
 else:
 s +=i

print('sum:',s)
 --------------------------------
 50
 ------------------方法2---------------------
 s1=0
 s2=0
 for i in range(1,100,2):
  s1 += i
 for j in range(2,100,2):
  s2 += j
 s3 = s1 - s2
 print(s3)
 ---------------------
 50
```

33.使用while循环输出1，2，3，4，5，6，8，9，10.

```
i = 0
while i < 10:
 i +=1
 if i == 7:
 continue
 else:
```

```
 print(i,end=',')
----------------------------
1,2,3,4,5,6,8,9,10,
```

34.输出1到100内所有的奇数?

```
for i in range(1,100,2):
 print(i)
```

35.输出1到100内所有的偶数?

```
for i in range(2,100,2):
 print(i)
```

36.字典如何删除键和合并两个字典?

```
>>> dic = {"name":"wg","age":18}
>>> del dic["name"]
>>> dic
{'age': 18}
>>> dic2 = {"name":"zw"}
>>> dic.update(dic2)
>>> dic
{'age': 18, 'name': 'zw'}
>>>
```

37.用lambda函数实现两数相乘?

```
s = lambda a,b:a*b
print(s(5,4))
----------------------
20
```

38.字符串a = "not 404 found 张三 99 深圳"，每个词中间是空格，用正则过滤掉英文和数字，最终输出"张三 深圳"?

```

```

39.列表和元组的区别?

LIST	TUPLES
列表是可变的，即可以编辑。	元组是不可变的。
列表比元组慢。	元组比列表快。

40.

```
# 当不确定你的函数中将要传递多少个参数时，使用*args.它可以传递任意数量的参数。
def print_everything(*args):
 for count,thing in enumerate(args):
 print("{}. {}".format(count,thing))

print_everything("apple","banana","cabbage")



# 相似的，**kwargs允许你使用没有事先定义的参数名
def table_things(**kwargs):
 for name,value in kwargs.items():
 print("{}={}".format(name,value))

table_things(apple="fruit",cabbage = "vagetable")

```

41.join()函数可以将指定的字符添加到字符串。

```
>>> "abc".join("123456")
'1abc2abc3abc4abc5abc6'
```

42.split()函数用指定的字符分割字符串。

```
>>> "1,1,2,2,3,4".split(",")
['1', '1', '2', '2', '3', '4']
```

43.将字符串转换为大小写?

```
>>> "WANGGANG".lower()
'wanggang'
----------------------------------------
>>> "wAnGganG".upper()
'WANGGANG'
```

44.简述同源策略?

- 协议相同
- 域名相同
- 端口相同

45.简述cookie和session的区别?

- session在服务器端，cookie在客户端（浏览器）。

- cookie安全性比session差。
- 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。

46.异常

- IOError:输入输出异常
- AttributeError:试图访问一个对象没有的属性。
- ImportError: 无法导入模块或者包，基本是路径的问题。
- IndentationError:语法错误
- IndexError:下标索引超出序列边界。
- KeyError:试图访问字典里不存在的键。
- SyntaxError:python代码逻辑语法错误。
- NameError:使用一个还未赋予对象的变量。

47.使用lambda函数对list排序 foo = [-5,8,0,4,9,-4,-20,-2,8,2,-4],输出结果为：[0,2,4,8,8,9,-2,-4,-4,-5,-20],正数从小到大，负数从大到小?

```
foo = [-5,8,0,4,9,-4,-20,-2,8,2,-4]
a = sorted(foo,key = lambda x:(x<0,abs(x)))
print(a)
----------------------------------------------------
[0, 2, 4, 8, 8, 9, -2, -4, -4, -5, -20]
```

48.递归求和

```
def get_sum(num):
 if num>=1:
 res = num+get_sum(num-1)

 else:
 res = 0

 return res
res = get_sum(10)
print(res)
-------------------------
55
```

49.统计字符串中某个字符出现的次数。

```
str = "zhangsan lisi wangwu hahaha zhangsan"
res = str.count("zhangsan")
```

```
print(res)
---------------------------------
2
```

50.正则匹配不是以4和7结束的手机号?

```
import re
tels = ["18428021065","15225025120","18428021064","15825202102","15729759477","10086"]
for tel in tels:
 ret = re.match("1\d{9}[0-3,5-6,8-9]",tel)
 if ret:
 print("想要的结果为: ",ret.group())
 else:
 print("%s 不是想要的手机号"% tel)
-------------------------------------
想要的结果为:  18428021065
想要的结果为:  15225025120
18428021064 不是想要的手机号
想要的结果为:  15825202102
15729759477 不是想要的手机号
10086 不是想要的手机号
>>>
```

51.正则匹配中文

```
import re
title = "你好，HELLO，世界"
pattern = re.compile(r'[\u4e00-\u9fa5]+')
result = pattern.findall(title)

print(result)
--------------------------------
['你好', '世界']
```

52.求两个列表的交集，差集，并集

```
a=[1,2,3,4]
b=[4,3,5,6]
jj1=[i for i in a if i in b]
```

```
jj2=list(set(a).intersection(set(b)))

bj1=list(set(a).union(set(b)))

cj1=list(set(b).difference(set(a)))
cj2=list(set(a).difference(set(b)))

print("交集：",jj1)
print("交集：",jj2)

print("并集：",bj1)

print("差集：",cj1)
print("差集：",cj2)
------------------------------
交集：  [3, 4]
交集：  [3, 4]
并集：  [1, 2, 3, 4, 5, 6]
差集：  [5, 6]
差集：  [1, 2]
>>>
```

53.python垃圾回收机制：

python垃圾回收主要以引用计数为主，标记-清除和分代清除为辅的机制，其中标记-清除和分代回收主要是为了处理循环引用的难题。

54.django 创建项目的命令?

```
django-admin startproject 项目名称
```

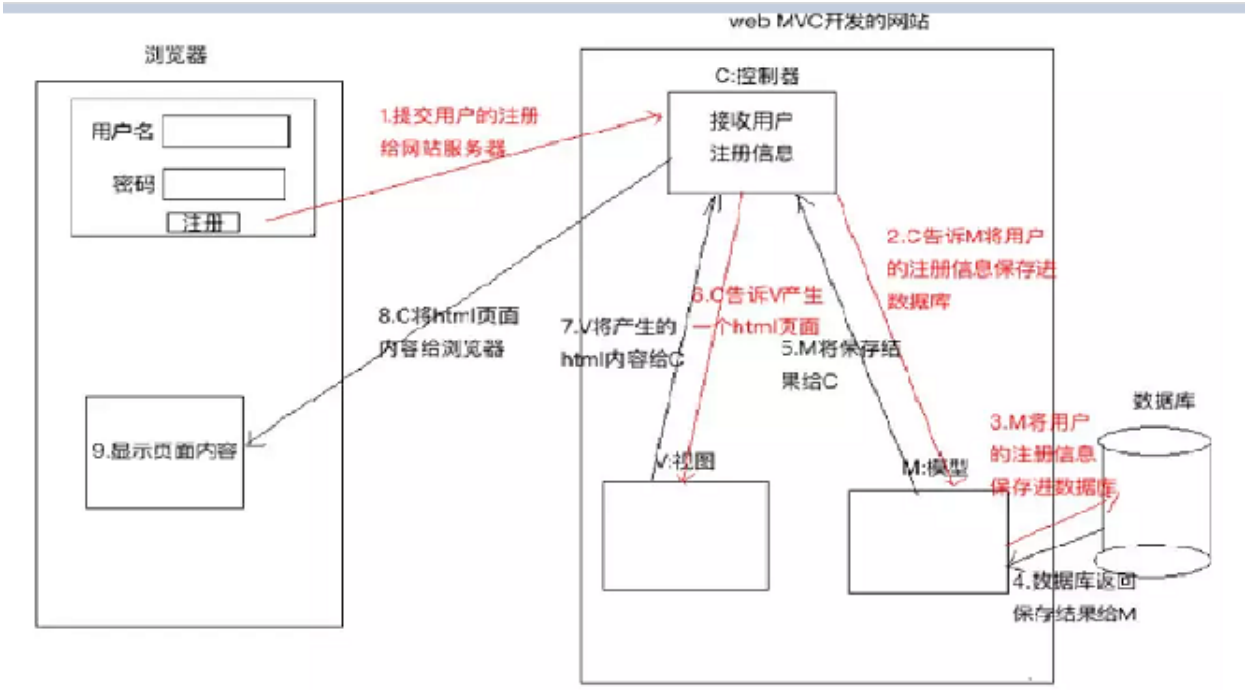
55.2.Django创建项目以后，项目文件夹下的组成部分（对mvt的理解）？

- **manage.py：是项目运行的入口，指定配置文件路径。**
- **__init__.py：是一个空文件，作用是这个目录可以被当做包使用，也可以在这个文件中做一些初始化的操作。**
- **settings.py：是项目的整体配置文件。**
- **urls.py：是项目的URL配置文件。**
- **wsgi.py：是项目与WSGI兼容的web服务器。**
- **与项目同名的目录：包含项目的配置文件、子应用之类的。**

56.对MVC的理解?

- **M：Model，模型，和数据库进行交互。**
- **V：View，视图，负责产生HTML页面。**
- **C：Controller，控制器，接收请求，进行处理，与M和V进行交互，返回应答。**

我们可以以用户注册的一个案例来说明一下三者之间的关系，结合图片进行说明：



1.用户输入完注册信息之后，点击按钮，将信息提交给网站的服务器。

2.Controller控制器接收用户的注册信息，Controller会告诉Model层将用户的注册信息保存到数据库中。

3.Model层接收到指令之后，将用户的注册信息保存进数据库。

4.数据库返回保存的结果给Model模型。

5.Model层再将保存的结果的返回给Controller控制器。

6.Controller控制器收到保存的结果之后，告诉VIew视图，View视图产生一个html页面。

7.View将产生的html页面的内容交给Controller控制器。

8.Controller控制器将html页面内容返回给浏览器。

9.浏览器接收到服务器Controller返回的html页面之后进行解析展示。

57.使用自定义迭代器输出斐波那契数列的前10项。

```
class Fib(object):
 def __init__(self,num):
 self.num = num
 self.a = 0
 self.b = 1
 self.current_index = 0

 def __next__(self):
 if self.current_index < self.num:
  result = self.a
  self.a,self.b = self.b,self.a+self.b
  self.current_index +=1
  return result
  else:
  raise StopIteration

  def __iter__(self):
  return self

if __name__=="__main__":
  fib = Fib(10)
  for value in fib:
  print(value,end=" ")
--------------------------------
0 1 1 2 3 5 8 13 21 34
```

58.使用生成器输出斐波那契数列前10项。

```
def Fib(num):
 a = 0
 b = 1
 current_index = 0
```

```
    while current_index < num:
    result = a
    a,b = b,a+b
    current_index +=1
    yield result
if __name__=="__main__":
    fib = Fib(10)
    for value in fib:
    print(value,end=" ")
--------------------------------
0 1 1 2 3 5 8 13 21 34
```

59.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。
  - 三大范式：
  - 1.表字段的原子性（不可拆分）；
  - 2.满足第一范式的基础上，有主键依赖；
  - 3.满足第一二范式的基础上，非主属性之间没有依赖关系。
- 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，使用磁盘队列技术等。
- 5.采用MySQL内部自带的表分区技术，把数据分成不同的文件，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IO。
- 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.分库分表机器（数据量特别大），主要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。

60.列出至少4种HTTP请求返回状态码，并解释其意思。

状态码	解释说明
302	跳转，新的url在响应的loaction头中给出
303	浏览器对于POST的响应进行重定向

	问
307	浏览器对于GET的响应重定向至新的URL
503	服务器维护或者负载过重未应答

61.在Linux系统中查看程序的端口号的命令?

netstat -tnulp

- -t：显示tcp端口。
- -u：显示udp端口。
- -l：仅显示套接字。
- -p：显示进程标识符和程序名称。
- -n：不进行DNS轮询，显示IP

62.判断一个字符串是不是回文（字符串是否对称）

```
def is_palindrom(s):
 """判断回文数，递归法"""
 if len(s) < 2:
 return True
 if s[0] == s[-1]:
 return is_palindrom(s[1:-1])
 else:
 return False
a=is_palindrom("abccba")
 b=is_palindrom("abcdex")
 c=is_palindrom("abcdcba")
 print(a)
 print(b)
 print(c)
 ----------------------------------
 True
 False
 True
```

63.统计在一个数列中的数字，有多少个正数，多少个负数?

```
# 方法1
a = [1,3,5,7,0,-1,-9,-4,-5,-8]

b = [i for i in a if i > 0]
```

```
5
6 print("大于0的个数：%s" %len(b))
7
8 c = [i for i in a if i < 0]
9
10 print("小于0的个数：%s"% len(c))
11 ------------------------------------
12 大于0的个数：4
13 小于0的个数：5
```

```
1 # 方法2
2 a = [1,3,5,7,0,-1,-9,-4,-5,-8]
3
4 m = 0
5 n = 0
6 for i in a:
7  if i > 0:
8  m +=1
9  elif i < 0:
10  n +=1
11  else:
12  pass
13 print("大于0的个数：%s" %m)
14 print("小于0的个数：%s" %n)
15 ---------------------------------
16 大于0的个数：4
17 小于0的个数：5
```

64.格式输出：已知一个数字为1，如何输出“0001”.

```
1 >>> a=1
2 >>> print("%04d" % a)
3 0001
```

65.已知一个队列,如： [1, 3, 5, 7], 如何把第一个数字，放到第三个位置，得到：[3, 5, 1, 7]。

```
1 a =[1,3,5,7]
2 a.insert(3,a[0])
```

```
3 print(a[1:])
4 ------------------------
5 [3, 5, 1, 7]
```

2019/12/10 19:59