python入门
1.Python适合的领域：
web网站和各种网络服务；
系统工具和脚本；
作为“胶水”语言，把其他语言开发的模块包装起来方便使用。
2.Python不适合的领域：
硬件驱动程序；
移动开发；
游戏开发。
3.Python和其他语言 对比：

	类型	运行速度	代码量
C	编译为机器码	非常快	非常多
Java	编译为字节码	快	多
Python	解释执行	慢	少

注：Python的源码不能加密。
4.选择Python的版本
目前Python有两个版本：
2.7版和3.3版，两个版本不兼容。
由于第三方库对2.7版兼容的比较好，所以使用2.7版。

第二章 python变量和数据类型
一、数据类型
1.整型：
python的整数运算结果仍然是整数。
2.字符串型：
2.1 python的单双引号是有区别的：
2.2 raw字符串和多行字符串：
在字符串前加r，表示这是一个raw字符串，里面的字符就不需要转义了：
例：r'\(~_~)/' 注：这种方式不能表示多行字符串
如果表示多行字符串，可以用'''.....'''表示：
'''line
line
line'''
注：可以在多行字符串之前加r，把这个多行字符串也变成一个raw字符串。
2.3 Unicode字符串
Unicode表示的字符串用u'...'表示，例：
u'中文'
注：Unicode字符串除了多了一个u之外，与普通字符没有什么区别，转义字符和多行表示法仍然有效。

```
# -*- coding: utf-8 -*-
```

告诉python解释器，用utf-8编码读取源代码。
3.浮点型：
浮点型的运算结果仍然是浮点数。
但是整数和浮点数的运算就变成了浮点数。

4.布尔型：True/false
python把0，空字符串和None 看成False，其他数值和非空字符串都看成True。
5.空：None

》print语句：
1.print语句打印多个字符串，每个字符串用逗号隔开，python遇到逗号，会输出一个空格。
2.print可以打印整数，或者计算结果。

》注释：
单行注释：#

6.列表--list
python内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。
声明：[] 用逗号隔开；
list中包含的元素并不要求都是同一种数据类型；
按照索引访问：在未指定索引的情况下，是从0开始的。
```
print list[0]
```
按照索引倒序访问：从-1开始访问最后一个元素，以此类推。
```
print list[-1]
print list[-2]
```
添加新元素：
append()总是把新的元素添加到list的尾部；
insert()，接受两个参数，第一个是索引号，第二个是待添加的新元素。
```
list.append('paul')
list.insert(0, 'paul')
```
注：使用负数索引，python是从倒数第一个之后添加。list.insert(-1, 'paul')
删除元素：
list.pop():默认pop删除最后一个元素，可以指定索引号让其删除。
注：删除元素后，list内的元素索引会发生变化，所以删除指定的元素，需要先删除后面的，再删除前面的。
替换元素：
对list中的某一个索引值赋值，就可以直接用新的元素替换掉原来的元素，list包含的元素个数保持不变。

7.元组--tuple
tuple和list非常相似，但是tuple一旦创建完毕是无法进行修改的。
tuple没有append(),insert()和pop()方法。
创建：t=('q', 'w', 'e')
```
tuple和list一样，可以包含0个、1个或任意多个元素。
```
注：由于定义单元素的元组会产生歧义，例如：t=(1) t的实际值会解析为1
所以python规定，单元素tuple要多加一个逗号。
创建可变的tuple：
```
t = ('1', 'b' , ['s', 'y'])
```
这样的话，tuple的值是可变的。tuple所谓的不变是说，tuple的每个元素，指向永远不变。即指向a，就不能改成指向b，但是指向的这个list的本身是可变的。
所以要创建，内容也不变得tuple，就需要使用不变的值，即定值。

8.if 语句
```
if age >=18:
    print '123'
```

```
    print '456'
print 'end'
```

注：Python代码的缩进规则：具有相同缩进的代码被视为代码块。
缩进为四个空格，不能混合空格和tab。

```
if-else:
if age >= 18:
    print 'you are old'
else:
    print 'you are young'
print 'end'

if-elif-else:
if age >= 18:
    print 'adult'
elif age >=6:
    print 'teenager'
elif age >= 3:
    print 'kid'
else:
    print 'baby'
```

9.for循环：

```
l=['adam', 'lisa', 'bart']
for item in l:
    print item
```

10.whie循环：

```
while i<100:
    print '123'
```

11.break 跳出当前循环。
continue跳出本次循环，进行下次循环。

第六章 dict
6.1 什么是dict
类似于PHP中的关联数组。list和tuple可以用来表示顺序集合，但是要根据某个值获取另一个，就没法办到了。

```
d= {
'adam':95,
'lisa':85,
'bart':59
}
```

len()函数可以计算任意集合的大小。dict也属于集合。

6.2访问dict
1.可以简单的使用d[key]的方式；
通过key访问dict的value，只要key存在，dict就返回对应的value。如果key不存在，会直接报错：KeyError
2.在取值之前可以先判断是否存在：用in来判断：

```
if 'paul' in d:
    print d['paul']
```

可以使用Python提供的get方法，在key不存在的时候，返回none：

```
print d.get('bart')
```

6.3 dict的特点
1.查找速度快，无论dict有十个元素还是十万个元素，查找速度都一样。缺点是占用内存大。这个和list相反，list占用内存小，但是查找速度快。在一个dict中，key不能重复。
2.存储的key-value是没有顺序的。
3.作为key的元素必须是不可变的，Python的基本类型如字符串、整数、浮点数都是不可变的，都可以作为key。

6.4 更新dict
dict是可变的，可以随时往dict中添加新的key-value。
如果key已经存在，则赋值会用新的value替换掉原来的value
6.5遍历dict
```
for key in d:
    print key
    print d.get(key)
```

6.6 set
1.set持有一系列元素，但是set的元素没有重复，而且是无序的。
```
s = set(['A', 'B', 'C'])
```
6.7 访问set
1.由于set存储的是无序集合，所以没办法通过索引来访问。
访问set中的元素实际上就是判断一个元素是否在set中。
```
'Adam' in s
```
6.8 set的特点
1.set的内部结构和dict很像，唯一的区别是不存储value。
2.set存储的元素和dict的key类似，必须是不变对象。任何可变对象是不能放入set中的
3.set中存储的元素是没有顺序的。
6.9 遍历set
由于set也是一个集合，所以遍历set和遍历list类似，都可以通过for循环实现。
```
for name in s:
    print name
```
6.10 更新set
1.把新的元素添加到set中。
添加元素时，用set的add()方法：
```
s = set(['1', '2'. 3''])
s.add(4)
```
注：如果添加的元素已经存在于set中，add()不会报错，但是也不会加进去。
2.删除set中的元素时，用set的remove()方法：
```
s = set(['1', '2'. 3''])
s.remove(4)
```
注：如果删除的元素不存在set中，remove()会报错。所以删除前需要判断。

第七章 函数
7.1 函数编写
定义一个函数使用def语句，依次写出函数名、括号、括号中的参数和冒号。
```
def myFunction():
    if x >=0:
        return x
    else:
        return -x
```
7.2 返回多个值
math包提供了sin()和cos()函数，需要先用import 引用它：
```
import math
def move(x, y, step, angle):
    nx = x + step * math.cos(angle)
    ny = y + step * math.sin(angle)
    return nx, ny
x, y = move(100, 100, 60, math.pi/6)
```
注：Python的函数返回多值其实就是返回一个tuple，然后按位置赋值给对应的值。
7.3 递归函数
如果一个函数在内部调用自身，这个函数就是递归函数。
理论上，所有的递归函数都可以写成循环的方式，但循环的逻辑不如递归清晰。

使用递归函数需要注意防止栈溢出。
在计算机中，函数调用是通过栈这种数据结构实现的，每当进入一个函数调用栈就会加一层栈帧，每当函数返回，栈就会减一层帧。由于栈的大小不是无限的，所以递归调用的次数过多，会导致栈溢出。

汉诺塔问题：

7.4 定义默认参数
```
def power(x, a=2):
    print x^2
```
默认参数只能定义在必须参数的后面。

7.5 定义可变参数
如果一个函数能接受任意个参数，就可以定义为可变参数：
```
def fn(*args):
    print args
```
可变参数的名字后边有个*号，我们可以传入0个、1个或多个参数给可变参数。

Python解释器会把传入的一组参数组装成一个tuple传递给可变参数。

第八章 list切片
8.1 对list进行切片
python提供了切片(Slice)：
L[0:3]:表示，从索引0开始取，直到索引3为止，但不包括索引3.即索引0，1，2.
注：如果第一个索引是0，还可以省略。
L[:]只用一个：,表示从头到尾。
切片操作还可以指定第三个参数：
L[::2]:表示每N个取一个

range()函数可以创建一个数列。

8.2 倒序切片
list的最后一个元素倒序索引为-1.
L[-2:]:从倒数第二个取到最后一个
L[0:-2]:从第一个取到倒数第二个
L[-4:-1:2]

8.3 对字符串切片
字符串'XXX'和Unicode字符串u'XXX'也可以看成是一种list，每个元素就是一个字符。
字符串进行切片后，操作结果仍然是字符串。
'ABCDE'[:3]
'ABCDE'[-3:]
'AbCDe'[::2]

第九章 迭代
9.1 在Python中，如果给定一个list或tuple，我么可以通过for循环来遍历这个list或tuple，这种遍历我们称为迭代。
在Python中，迭代是通过for。。。in来完成的。
Python的for循环不仅可以用在list或tuple上，还可以作用在其他任何可迭代对象上。
集合：
1.有序集合：list，tuple，str和unicode
2.无序集合：set
3.无序集合并且具有key-value对：dict
迭代与按下标访问集合最大的不同是，后者是一种具体的迭代实现方式，而前者只关心迭代结果，根本不关

心迭代内部是如何实现的。

9.2 索引迭代

Python中，迭代永远是取出元素的本身，而非元素的索引。

如果需要取出元素的索引，可以使用enumerate()函数。

```
for inde,name in enumerate(L)
    print index,'-',name
```

索引迭代不是真的按索引访问，而是由enumerate（）函数自动把每个元素变成(index, element)这样的tuple，再迭代，这样就同时获得了索引和元素本身。

使用 enumerate() 函数，我们可以在for循环中同时绑定索引index和元素name。但是，这不是 enumerate() 的特殊语法。实际上，enumerate() 函数把：

```
['Adam', 'Lisa', 'Bart', 'Paul']
```

变成了类似：

```
[(0, 'Adam'), (1, 'Lisa'), (2, 'Bart'), (3, 'Paul')]
```

因此，迭代的每一个元素实际上是一个tuple：

```
for t in enumerate(L):
    index = t[0]
    name = t[1]
    print index, '-', name
```

zip()函数可以把两个list合并为一个list。

9.3 迭代dict的value

用for循环直接迭代dict，可以每次拿到dict的一个key。

如果希望拿到迭代对象dict的value，可以使用values()方法或itervalues()。

不同：

1.values()方法实际上把一个dict转换成了包含value的list。

2.itervalues()方法不会转换，它会在迭代过程中依次从dict中取出value，所以itervalues()方法比values()方法节省生成list的内存。

3.打印 itervalues() 发现它返回一个 <dictionary-valueiterator> 对象，这说明在Python中，for 循环可作用的迭代对象远不止 list，tuple，str，unicode，dict等，任何可迭代对象都可以作用于for循环，而内部如何迭代我们通常并不用关心。

9.4 迭代dict的key和value

可以使用items()方法：

```
for key,value in d.items():
    print key,'=>',value
```

iteritems()作用和上述方法相同，只是iteritems()不把dict转换成list，而是在迭代过程中不断给出tuple，所以，iteritems()不占用额外的内存。

第十章 生成列表

列表生成式：

```
[x*x for x in range(1, 11)]
[1, 4, 9...]
```

完全可以通过一个复杂的列表生成式把它变成一个 HTML 表格：

```
tds = ['<tr><td>%s</td><td>%s</td></tr>' %
(name, score) for name, score in
d.iteritems()]
print '<table>'
print '<tr><th>Name</th><th>Score</th><tr>'
print '\n'.join(tds)
print '</table>'
```

注：字符串可以通过 % 进行格式化，用指定的参数替代 %s。字符串的join()方法可以把一个 list 拼接成一个字符串。

把打印出来的结果保存为一个html文件，就可以在浏览器中看到效果了：

10.2 条件过滤
列表生成式的for循环后面还可以加上if语句判断。
[x*x for x in range(1, 101) if x%2 == 0]

10.3 多层表达式
for循环可以嵌套，因此，在列表生成式中，也可以用多层for循环来生成列表。
[m+n for m in 'ABC' for n in '123']

Python进阶
第一章 函数式编程
2.1简介
函数式：functional，是一种编程范式。
2.2 高阶函数
1.Python中变量可以指向函数；
2.函数名其实就是指向函数的变量；
3.高阶函数：能接收函数做参数的函数；
2.3 函数实例：
1.map()函数，可以将一个函数作用在list的每个元素上
map(abs, [0, -1, 3])
2.reduce()函数是Python内置的高阶函数。传入参数与map()函数相同。reduce()函数传入的函数必须接受两个参数，reduce()对list的每个元素反复调用函数f，并返回最终的结果。
3.filter()函数：
接收一个函数F和一个list，这个函数F的作用是对每个元素进行判断，返回true或false，filter()根据判断结果自动过滤掉不符合条件的元素，返回由复合条件元素组成的新list。
注：s.strip()删除字符串中开头、结尾处的rm序列的字符。
当rm为空时，默认删除空白符（包括'\n','\r','\t'）。

4.自定义排序函数
python内置的sorted()函数可对list进行排序：
传入两个待比较的元素x,y,如果x应该排在y的前面，返回－1，如果x应该排在y的后面，返回1.如果x和y相等，返回0.

5.Python的函数不但可以返回int、str,list,dict等数据类型，还可以返回函数。

```
def f():
    print 'call f()'
    def g():
        print 'call g()'
    return g
```

```
x = f()
x()
```

6.闭包

1)定义：这种内层函数引用了外层函数的变量，然后返回内层函数的情况，称为闭包。

```
def calc_sun(lst):
    def lazy_sum():
        return sum(lst)
    return lazy_sum
```

2）特点：返回的函数还引用了外层函数的局部变量，所以，要正确使用闭包，就要确保引用的局部变量在函数返回后不能变。

返回闭包不能引用循环变量。

闭包这块不是太理解。

7.匿名函数

在python中，对匿名函数提供了有限的支持。

```
map(lambda x:x*x, [1,2,3,4])
```

关键字lambda表示匿名函数。冒号前的x表示函数参数。

注：匿名函数有个限制，就是只能有一个表达式，不写return，返回值就是该表达式的结果。

8.装饰器

希望在不修改原函数的基础之上，动态的增加函数的功能。

```
def f1(x):
    return x*2
def new_fn(f):
    def fn(x):
        print 'call'+f.__name__+'()'
        return f(x)
    return fn

f1 = new_fn(f1)
print f1()
```

python内置的@语法就是为了简化装饰器调用。

```
@new_fn
def f1(x):
    return x*2
```

装饰器的作用：

可以极大的简化代码，避免每个函数编写重复性代码。

打印日志：@log

检测性能：@performance

数据库事务：@transaction

URL路由:@post('/register')