[1. Python 历史和快速教程 4](#_Toc423422719)

[1-1. Learn Python—the hard way 4](#_Toc423422720)

[2. Python 语法 4](#_Toc423422721)

[2-1. 数据类型和变量 4](#_Toc423422722)

[2-1-1. 变量不需要声明，可以直接使用 4](#_Toc423422723)

[2-1-2. 基本数据类型 4](#_Toc423422724)

[2-1-3. 序列 4](#_Toc423422725)

[2-1-4. 只有一个元素的元组 6](#_Toc423422726)

[2-1-5. 字符串 6](#_Toc423422727)

[2-1-6. 运算 7](#_Toc423422728)

[2-1-7. 字典 7](#_Toc423422729)

[2-2. 缩进 8](#_Toc423422730)

[2-3. 循环 9](#_Toc423422731)

[2-3-1. Range() 9](#_Toc423422732)

[2-3-2. Enumerate 9](#_Toc423422733)

[2-3-3. Zip 10](#_Toc423422734)

[2-4. 函数 10](#_Toc423422735)

[2-4-1. 定义 10](#_Toc423422736)

[2-4-2. 值传递和指针传递 10](#_Toc423422737)

[2-4-3. 关键字传值 11](#_Toc423422738)

[2-4-4. 参数的默认值 11](#_Toc423422739)

[2-4-5. 包裹传递 11](#_Toc423422740)

[2-4-6. 解包裹 12](#_Toc423422741)

[2-4-7. Lambda函数 12](#_Toc423422742)

[2-4-8. Map函数 13](#_Toc423422743)

[2-4-9. Filter函数 13](#_Toc423422744)

[2-4-10. Reduce函数 14](#_Toc423422745)

[2-4-11. 函数作为参数传递 14](#_Toc423422746)

[2-5. 异常处理 15](#_Toc423422747)

[2-5-1. 抛出异常 15](#_Toc423422748)

[2-6. 类和对象 15](#_Toc423422749)

[2-6-1. 类定义 15](#_Toc423422750)

[2-6-2. 类属性 16](#_Toc423422751)

[2-6-3. \_\_init\_\_()方法 16](#_Toc423422752)

[2-6-4. 对象属性 16](#_Toc423422753)

[2-6-5. Dir（）用来查看类和对象的所有属性 17](#_Toc423422754)

[2-7. 类定义里除了类变量，还允许语句 17](#_Toc423422755)

[2-8. IO 18](#_Toc423422756)

[2-8-1. 文件读写 18](#_Toc423422757)

[2-9. 模块 19](#_Toc423422758)

[2-9-1. 定义和使用 19](#_Toc423422759)

[2-9-2. 模块的搜索路径 19](#_Toc423422760)

[2-9-3. 模块包 20](#_Toc423422761)

[2-10. 正则表达式 20](#_Toc423422762)

[2-10-1. 字符串中的\必须转义 20](#_Toc423422763)

[2-10-2. Python 中的正则表达式也用\来表示转义 20](#_Toc423422764)

[2-10-3. 当正则表达式需要用python中的字符串表示时，会带来双重转义 20](#_Toc423422765)

[2-10-4. 阅读和编写正则表达式的方式 21](#_Toc423422766)

[2-10-5. Search，match，findall 21](#_Toc423422767)

[2-11. Match 和Search返回的是MatchObject对象，findall返回的是字符串集合 22](#_Toc423422768)

[3. Python 标准库(standard library) 25](#_Toc423422769)

[3-1. Python增强 Python advanced 25](#_Toc423422770)

[3-1-1. 文字处理 25](#_Toc423422771)

[3-1-2. 数据对象 25](#_Toc423422772)

[3-1-3. 日期和时间 25](#_Toc423422773)

[3-1-4. 数学运算 25](#_Toc423422774)

[3-1-5. 存储 25](#_Toc423422775)

[3-2. 系统互动 25](#_Toc423422776)

[3-2-1. Python运行控制 25](#_Toc423422777)

[3-2-2. 操作系统 25](#_Toc423422778)

[3-2-3. 线程与进程 25](#_Toc423422779)

[3-3. 网络 25](#_Toc423422780)

[3-3-1. 基于socket层的网络应用 25](#_Toc423422781)

[3-3-2. 互联网应用 25](#_Toc423422782)

[3-4. Os 25](#_Toc423422783)

[3-4-1. Os.path.dirname 25](#_Toc423422784)

[3-5. Where is sys.py 25](#_Toc423422785)

[4. 为easy\_install 和pip设置代理 27](#_Toc423422786)

[5. 为Python安装第三方包 27](#_Toc423422787)

[6. Django 28](#_Toc423422788)

[6-1. 安装 28](#_Toc423422789)

[6-1-1. Windows7下使用django-admin 而不要用django-admin.py 28](#_Toc423422790)

[6-2. 启动 29](#_Toc423422791)

[6-3. 连接PostgreSQL 29](#_Toc423422792)

# Python 历史和快速教程

<http://www.cnblogs.com/vamei/archive/2013/02/06/2892628.html>

里面有一套非常不错的Python&Django入门教程

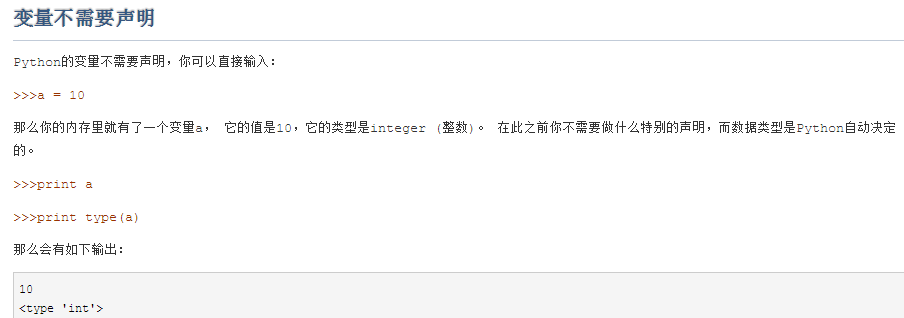
## Learn Python—the hard way

http://learnpythonthehardway.org/book/

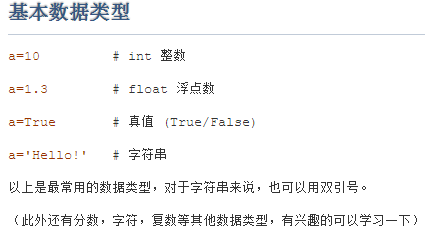
# Python 语法

## 数据类型和变量

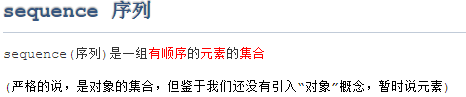
### 变量不需要声明，可以直接使用



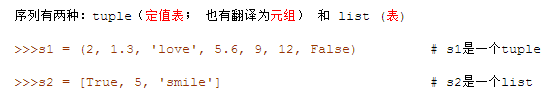
### 基本数据类型



### 序列

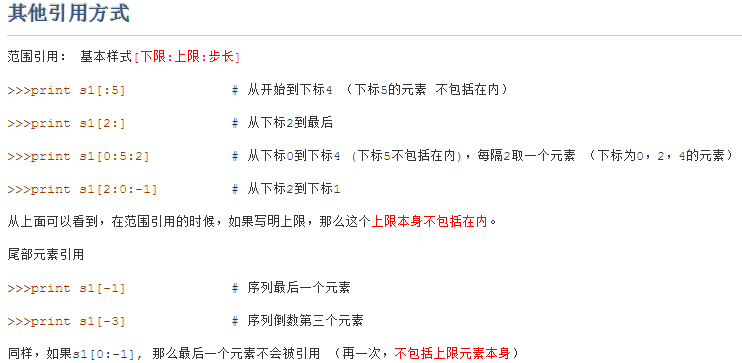


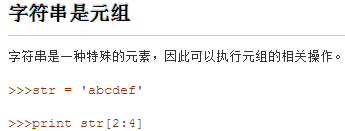




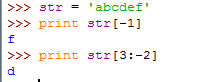






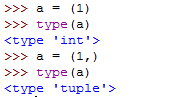


注意[]里面的index，但只有一个值时，就是取某个index的值，不存在包括不包括上限的问题。只有当是用[a:b]的形式时，才存在b是上限，只包括b-1的形式



### 只有一个元素的元组

当一个元组只有一个元素时，如(1), 在定义时应该在元素的末尾再显示加一个, 以表示这是一个元组类型。否则，编译器可能认为它只是一个括号包括的int值



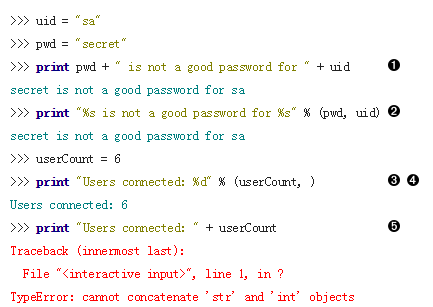
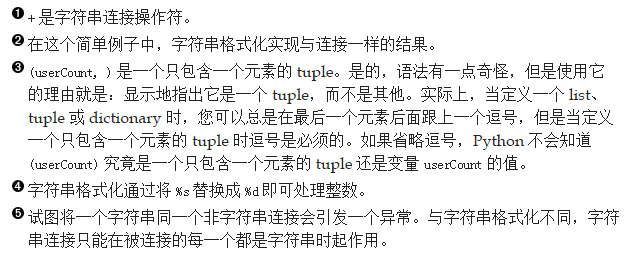
List不存在这样的问题



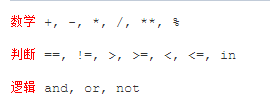
### 字符串

字符串之所以能用[]取某个特定index处的字符，是因为字符串也是一个元组

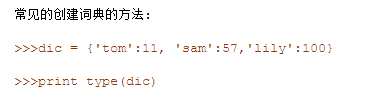
字符串的连接：

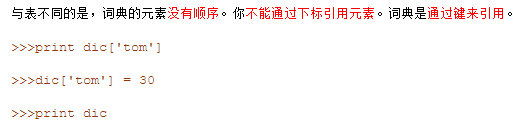
 

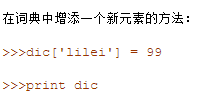
### 运算

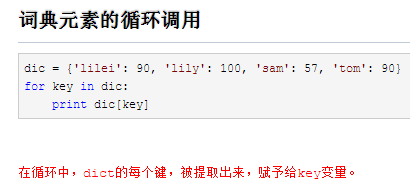


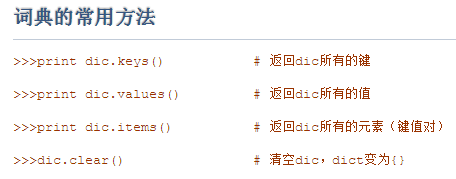
### 字典

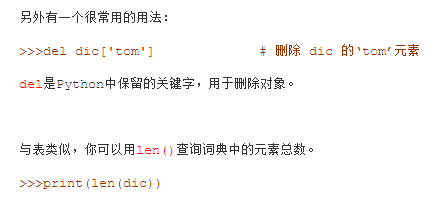




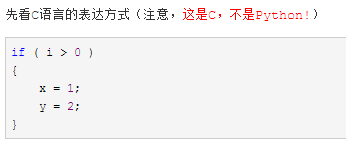
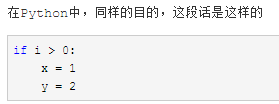








## 缩进

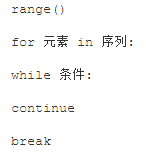
 



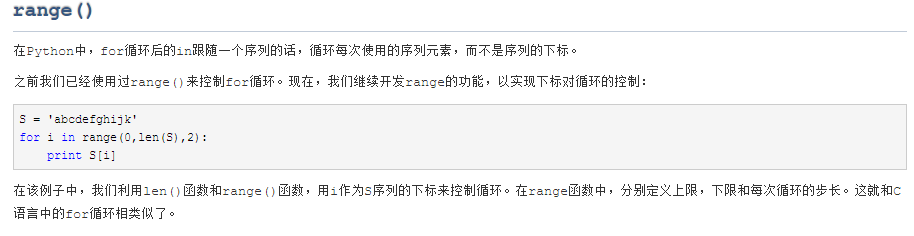




## 循环



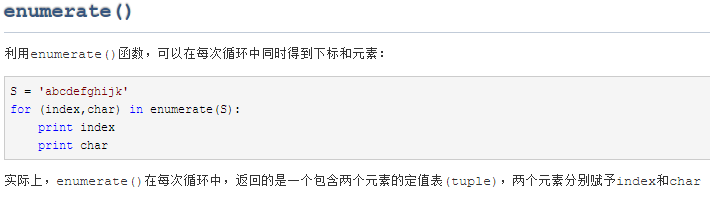
### Range()



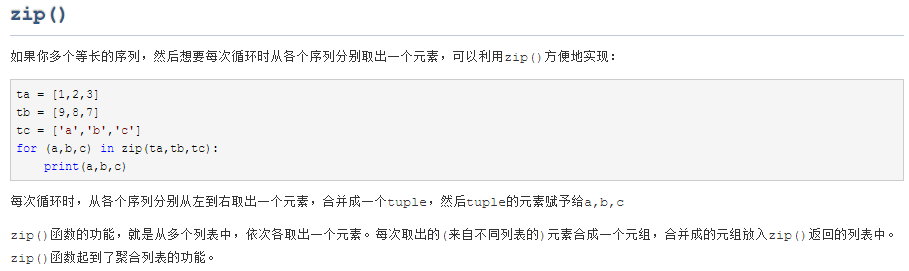
Range 的上限不包含在结果中，如

### Enumerate

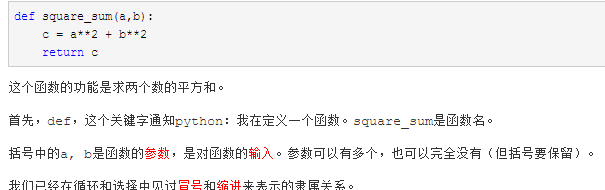


### Zip



## 函数

### 定义



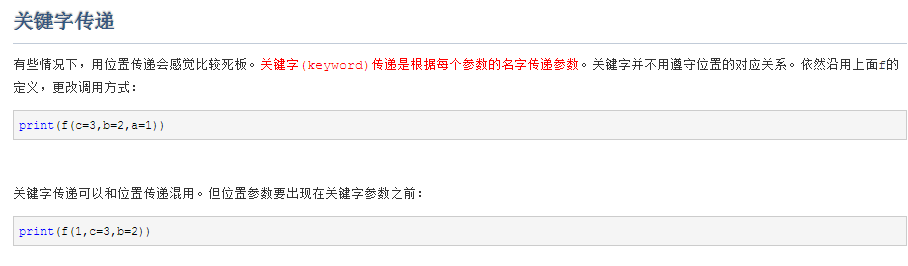




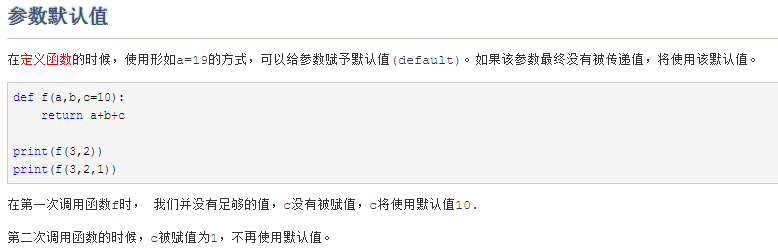
### 值传递和指针传递



### 关键字传值



### 参数的默认值



### 包裹传递

分两种：tuple 和 dictionary

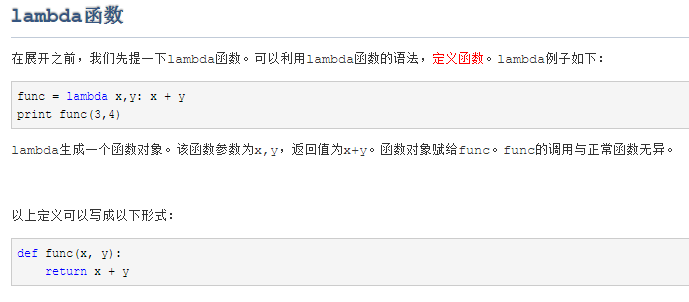


### 解包裹

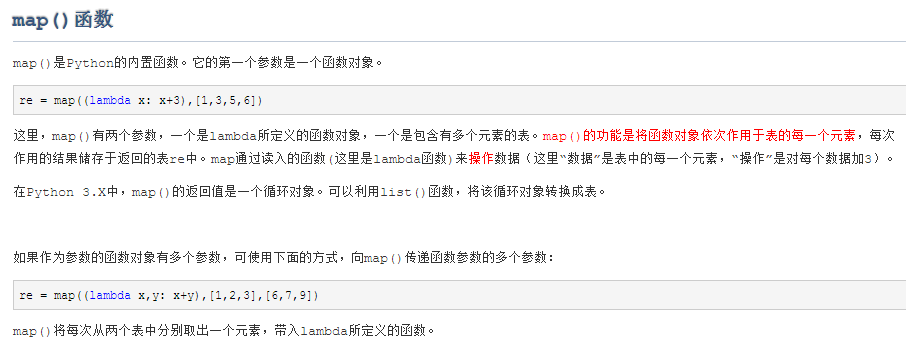


### Lambda函数

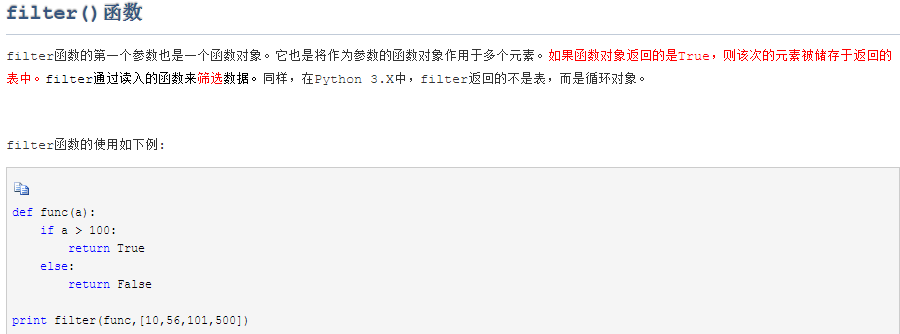




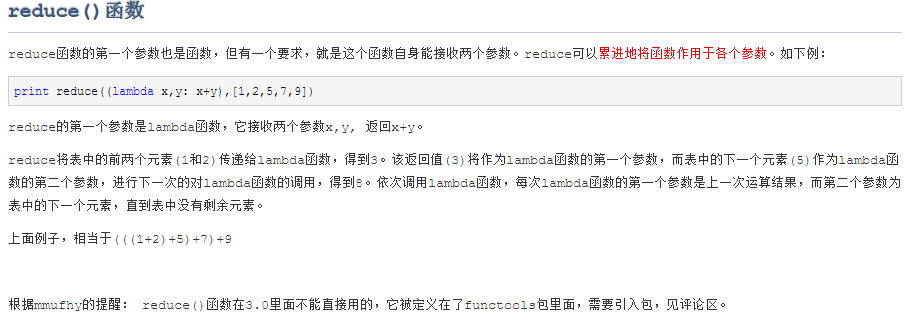
### Map函数



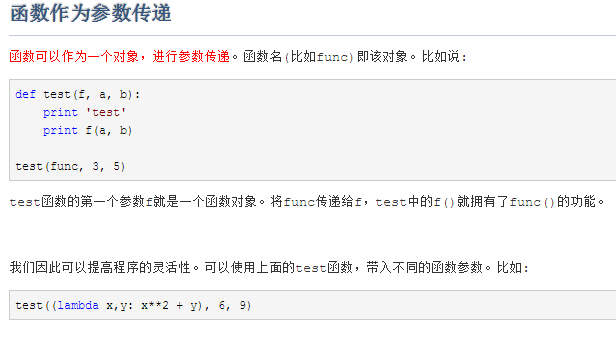
### Filter函数



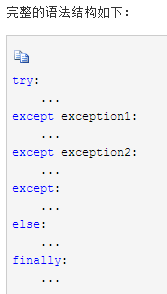
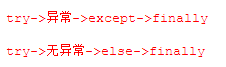
### Reduce函数



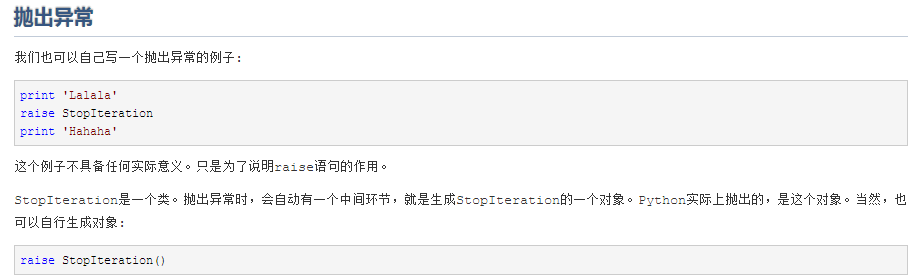
### 函数作为参数传递



## 异常处理

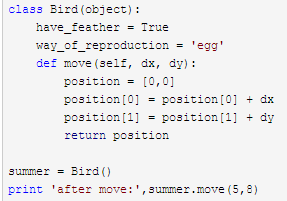
 

### 抛出异常



## 类和对象

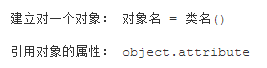
### 类定义

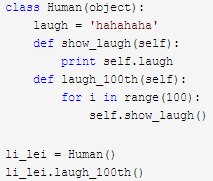


生成对象时不需要用new关键字



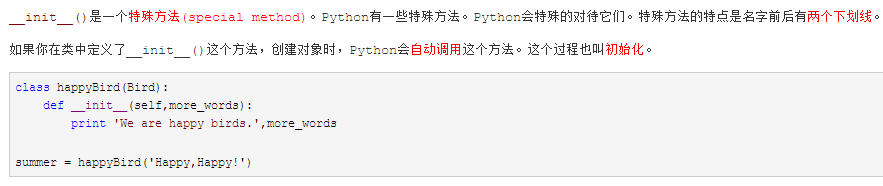


### 类属性

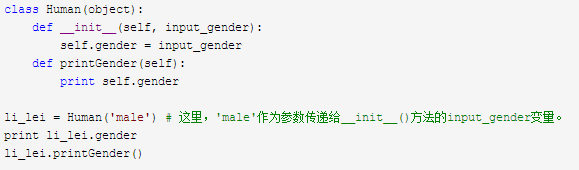


### \_\_init\_\_()方法



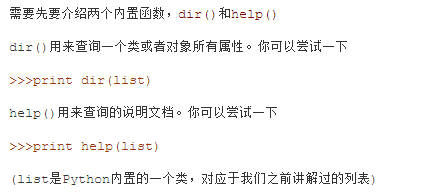
### 对象属性





意味着可以动态给对象增加属性

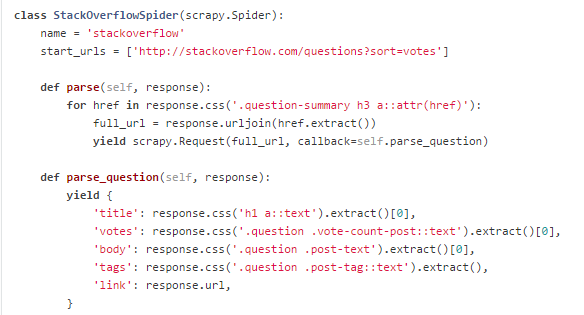
### Dir（）用来查看类和对象的所有属性



## 类定义里除了类变量，还允许语句

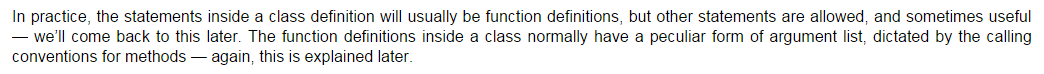
这个问题主要来自于在scrapy里写spider的时候

我是用的一个for循环语句填充的start\_urls



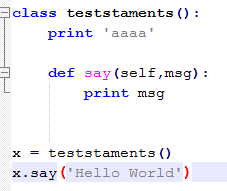
那这个for循环语句在什么时候时候被执行呢。

首先，这是允许的



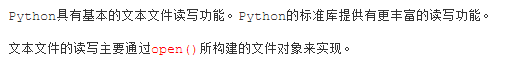
<https://docs.python.org/2/tutorial/classes.html#class-definition-syntax>

执行时机好像是实例化对象的就会被先调用

## IO

### 文件读写





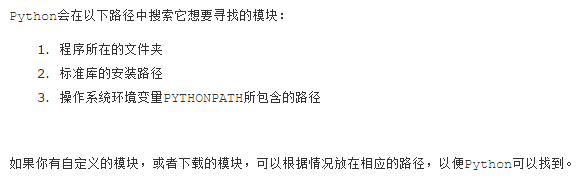
## 模块

### 定义和使用

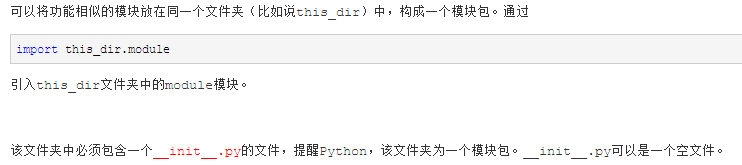




### 模块的搜索路径



### 模块包

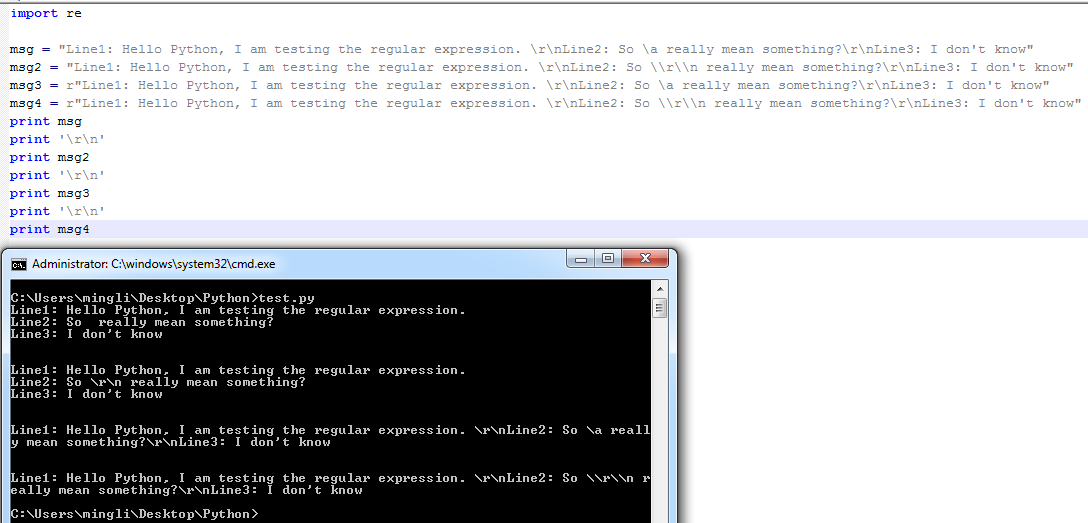


## 正则表达式

### 字符串中的\必须转义

首先我们看到，就字符串本身而言，反斜杠\必须转义。即如果要在字符串中包含一个字面上的\，必须用两个\ \ 表示。

否则的话，\就不是字面上的反斜杠的意思，而是一个转义字符，和后面的一个字符一起表示一个特定的字符。如果这个组合不是一个有效的组合，那么反斜杠和后面的字符一起变成了一个不可打印字符。比如msg中的\a



### Python 中的正则表达式也用\来表示转义

比如 \d 表示数字[0-9]

### 当正则表达式需要用python中的字符串表示时，会带来双重转义

比如我们可以说 \d 是正则表达式中用来表示数字的

但问题是如果\d 想用在python中，必须写成字符串的形式，即”\d”

但问题来了，在python的字符串中，\如果想表示字面的\，必须被转义，否则\d已经先被字符串转义解释了，轮不到正则表达式了。所以，为了获得正则表达式的字面的\d, 匹配字符串必须写成“\ \d”

比较极端的情况出现在正则表达式需要匹配\的情况。在正则表达式中，必须用两个\ \来匹配反斜杠。但是，为了获得字面的\ \, 每个反斜杠都要先符合字符串的转义，即要写成” \ \ \ \”。 稍微丑陋了点。

为了解决双重转义的问题，在字符串前加r，即raw ， 告诉字符串，里面的\都是字面的了，不需要再经过转义了。

所以，r其实是针对字符串的，而不是针对正则表达式的。类似于C#字符串前的@

### 阅读和编写正则表达式的方式

编写时，写考虑正则表达式的字面形式，如\d , 然后再将其转化为字符串。此时可以有两种形式，将\转义或者添加前缀r

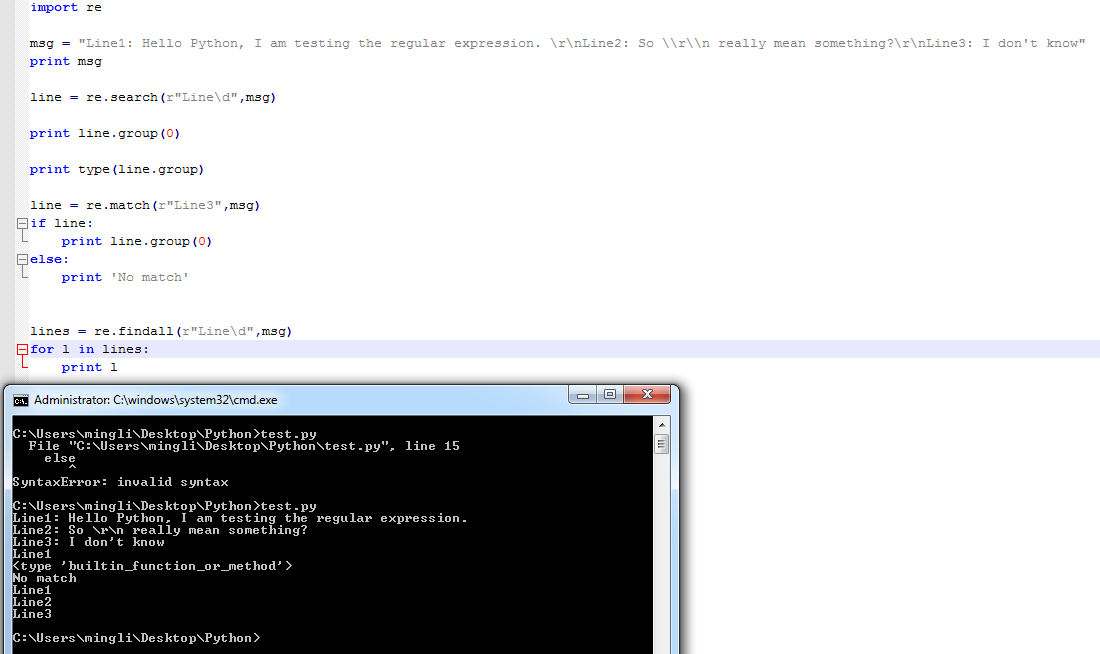
阅读时，先将字符串转化为正则表达式的字面形式，即如果有r，那本身就是正则表达式的字面形式，否则，将\ \先替换为\,然后再按照正则表达式的规则解析

### Search，match，findall

Search 返回任意一个匹配的内容，不管从什么位置开始，不过找到一个匹配项目后就返回，即最多只匹配一个，用来check是否存在

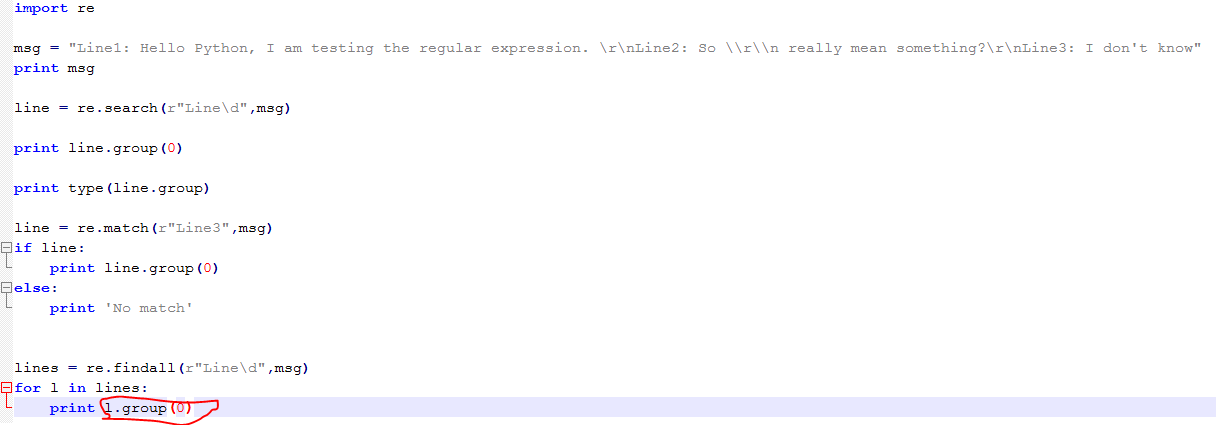
Match： 必须从头开始匹配，中间的匹配不算。

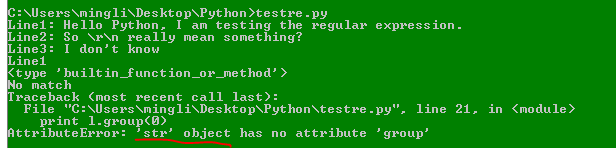
Findall：返回所有匹配的项目。



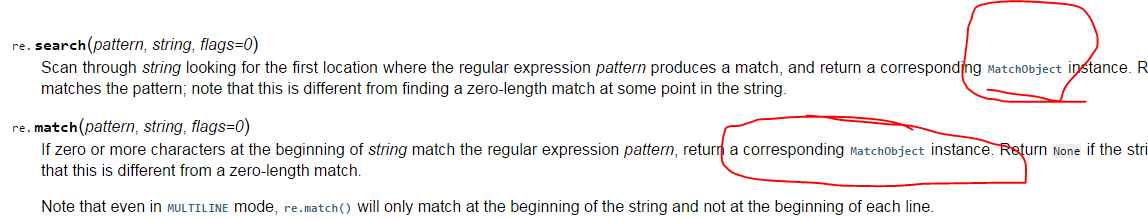
## Match 和Search返回的是MatchObject对象，findall返回的是字符串集合

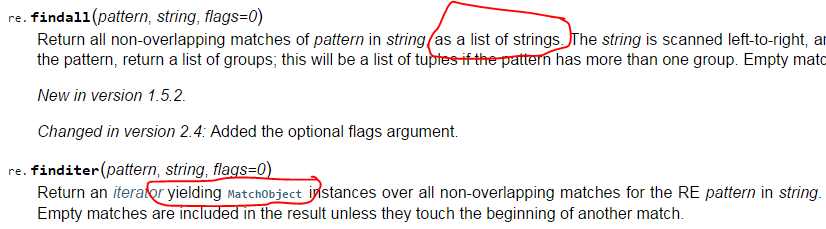
只有MatchObject对象才有group方法，如下面，当试图对l，也就是字符串取group(0)时，出错了。 如果实在想用findall，但又想取group，换用finditer方法

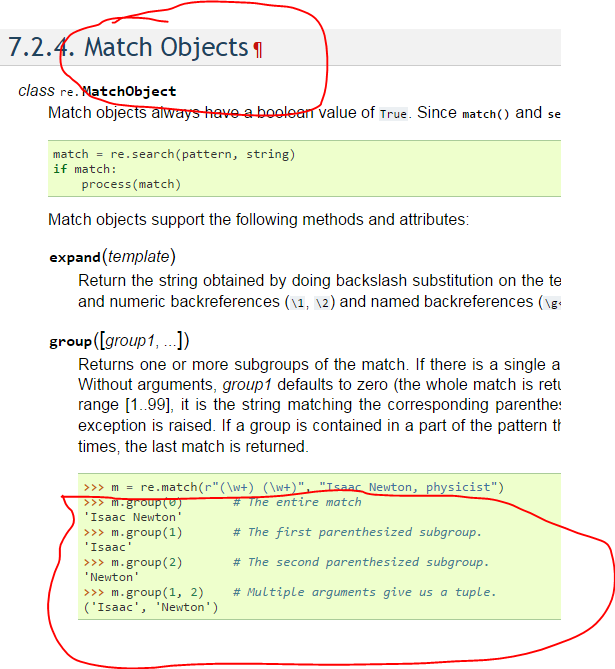


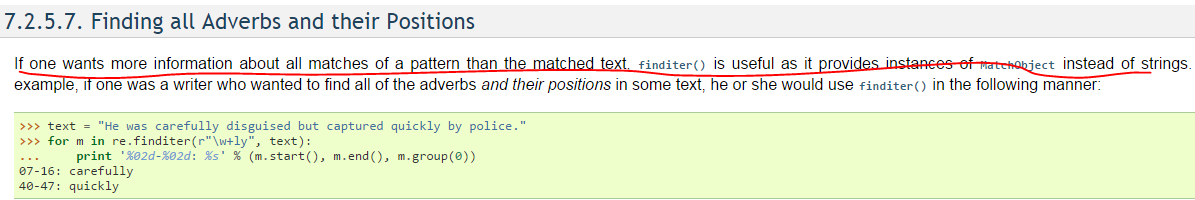


详见<https://docs.python.org/2/library/re.html>









# Python 标准库(standard library)



## Python增强 Python advanced

### 文字处理

### 数据对象

### 日期和时间

### 数学运算

### 存储

## 系统互动

### Python运行控制

### 操作系统

### 线程与进程

## 网络

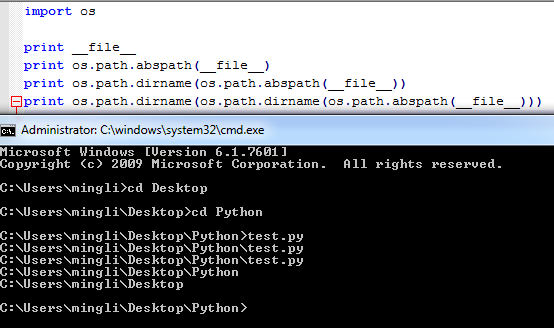
### 基于socket层的网络应用

### 互联网应用

## Os

### Os.path.dirname

返回的是父目录，所以如果参数本身是目录，则返回其父目录



## Where is sys.py

Lisi wrote:

> *If sys.py is a file, it must be somewhere; but I can't find it. Where is it?*

> *I would like to look at it.*

Others have already answered this, but consider how you might explore

the answer at the interactive interpreter:

>>> import os

>>> os.\_\_file\_\_

'/usr/lib/python2.5/os.pyc'

>>> import sys

>>> sys.\_\_file\_\_

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

AttributeError: 'module' object has no attribute '\_\_file\_\_'

**So the sys module has no \_\_file\_\_ attribute, and therefore there is no**

**sys.py.** Looking deeper:

>>> import inspect

>>> inspect.getfile(sys)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "/usr/lib/python2.5/inspect.py", line 347, in getfile

raise TypeError('arg is a built-in module')

TypeError: arg is a built-in module

Hmmm, not the most user-friendly of messages ("arg" is a built-in

module? how about \*naming\* it, all modules have a name!) but the answer

is clear: **sys is part of the Python virtual machine itself, and does not**

**live in a file.**

(Except, of course, that Python itself is compiled from files, and the

code creating sys will be in those.)

Furthermore:

>>> import math

>>> math.\_\_file\_\_

'/usr/lib/python2.5/lib-dynload/mathmodule.so'

So here we have a module that does live in a file, but it is a C

library, not a .py python file. (On Windows, it may be a .dll.)

> *The modules within it must surely be somewhere too. But since I can't find*

> *sys, I obviously can't find the modules. Again, I'd like to look at them.*

I don't understand what you mean by "modules within it".

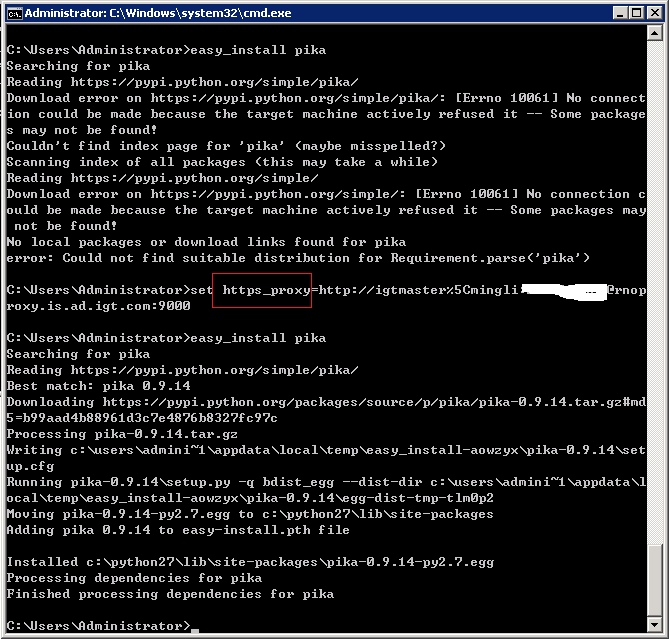
Do you mean sys.modules?

That's easy:

>>> sys.modules['inspect']

<module 'inspect' from '/usr/lib/python2.5/inspect.pyc'>

# 为easy\_install 和pip设置代理

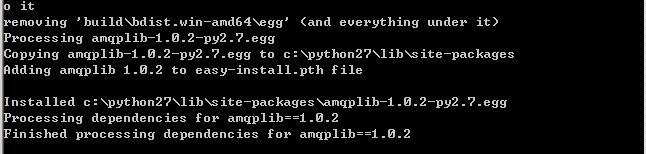


注意用set https\_proxy 而不是http\_proxy

# 为Python安装第三方包

<http://blog.sina.com.cn/s/blog_5c5d5cdf0101e20j.html>

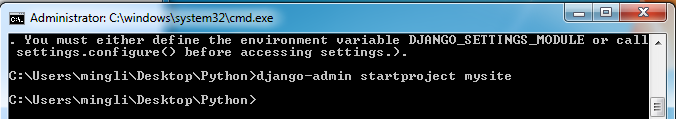
源码安装时，源码可以拷贝到任何地方，当运行install时，会自动被拷贝到python安装目录下的lib中

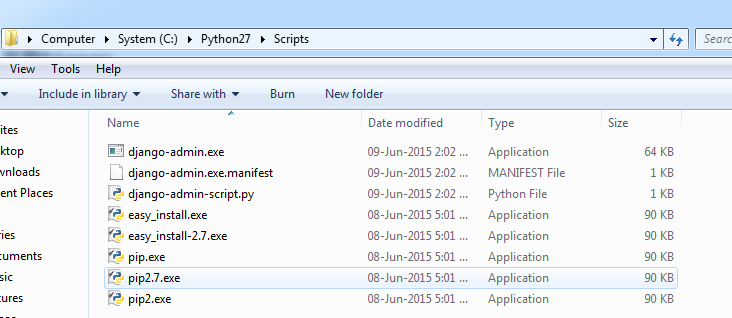


# Django

## 安装

### Windows7下使用django-admin 而不要用django-admin.py



因为python的安装目录下是一个django-admin.exe 而不是.py

## 启动



## 连接PostgreSQL

