目录

[1 概述 2](#_Toc491969821)

[1.1 参考教程 2](#_Toc491969822)

[1.2 优秀框架 2](#_Toc491969823)

[2 安装环境 3](#_Toc491969824)

[3 数据类型 3](#_Toc491969825)

[3.1 字典 3](#_Toc491969826)

[3.1.1 Dictionary 3](#_Toc491969827)

[3.1.2 Tuple 4](#_Toc491969828)

[3.1.3 List 4](#_Toc491969829)

[3.1.4 布尔型 5](#_Toc491969830)

[3.1.5 映射 5](#_Toc491969831)

[3.1.6 字符串 5](#_Toc491969832)

[3.2 技巧 6](#_Toc491969833)

[4 基础 6](#_Toc491969834)

[4.1 常用方法 6](#_Toc491969835)

[4.1.1 Print 6](#_Toc491969836)

[4.1.2 String 6](#_Toc491969837)

[4.2 内置函数 6](#_Toc491969838)

[4.3 类型转化 8](#_Toc491969839)

[4.4 自省 8](#_Toc491969840)

[4.5 异常 9](#_Toc491969841)

[4.6 面向对象 10](#_Toc491969842)

[4.6.1 基本类 10](#_Toc491969843)

[4.6.2 专用类方法 12](#_Toc491969844)

[4.6.3 私有属性，方法 12](#_Toc491969845)

[4.6.4 方法名字特别 13](#_Toc491969846)

[4.6.5 调用父类 13](#_Toc491969847)

[4.6.6 与java区别 14](#_Toc491969848)

[4.7 常用知识 14](#_Toc491969849)

[4.7.1 格式化数字 14](#_Toc491969850)

[4.7.2 列推到式 14](#_Toc491969851)

[4.7.3 序列 14](#_Toc491969852)

[4.7.4 and or 15](#_Toc491969853)

[4.7.5 Lambda函数 15](#_Toc491969854)

[4.7.6 可变参数\*args ，\*\*kwargs 15](#_Toc491969855)

[4.7.7 空判断 15](#_Toc491969856)

[4.7.8 \_\_all\_\_, \_\_slots\_\_ 16](#_Toc491969857)

[4.7.9 全局变量 16](#_Toc491969858)

[4.7.10 修饰符 16](#_Toc491969859)

[4.7.11 定时调用 17](#_Toc491969860)

[5 常用 17](#_Toc491969861)

[5.1 常用模块 17](#_Toc491969862)

[5.1.1 Print 17](#_Toc491969863)

[5.1.2 文件操作 17](#_Toc491969864)

[5.1.3 xml操作 18](#_Toc491969865)

[5.1.4 正则表达式 18](#_Toc491969866)

[5.1.5 JSON 18](#_Toc491969867)

[5.1.6 单元测试 19](#_Toc491969868)

[5.1.7 目录操作 19](#_Toc491969869)

[5.1.8 调用DLL 19](#_Toc491969870)

[5.1.9 C生成DLL 20](#_Toc491969871)

[5.1.10 操作WPSoffice 21](#_Toc491969872)

[5.1.11 操作Excel 21](#_Toc491969873)

[5.1.12 C++ 21](#_Toc491969874)

[5.2 常用代码 22](#_Toc491969875)

[5.2.1 2，3区别 23](#_Toc491969876)

[6 第三方模块 23](#_Toc491969877)

[6.1.1 excel 23](#_Toc491969878)

[6.1.2 itertools 迭代模块 23](#_Toc491969879)

[6.1.3 itertools 迭代模块 23](#_Toc491969880)

[7 问题 24](#_Toc491969881)

[7.1.1 Mysql安装 24](#_Toc491969882)

[7.1.2 加速镜像 24](#_Toc491969883)

# 概述

## 参考教程

<http://www.liaoxuefeng.com/wiki/0014316089557264a6b348958f449949df42a6d3a2e542c000>

## 优秀框架

科学计算：matplotlib、numpy、scipy  
web框架：flask、django  
GUI：PyQt、WxPython  
[网络爬虫](https://www.baidu.com/s?wd=%E7%BD%91%E7%BB%9C%E7%88%AC%E8%99%AB&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLmWNhPj0suWmYnyf3PjbY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1nvrH6vnjbzn1nYPWm3Pjf3P0)：Scrapy  
HTML解析：BeautifulSoup、lxml

# 安装环境

第三方包下载地址： <https://pypi.python.org/pypi>

使用pip 安装： pip install package-name

安装pip前需要安装setuptools,  window执行里面 ez\_setup.py 安装

pip 目录里执行  setup.py install

安装mysql:   pip install mySQL-python

修改django  http端口为9000，在运行参数里 增加  runserver --noreload 9000

manage.py inspectdb

manage.py sqlall blog

manage.py validate

# 数据类型

## 字典

### Dictionary

d = {"server":"mpilgrim", "database":"master"}

d["retrycount"] = 3

del d[42]

d.clear()

params.keys()

params.values()

params.items() //(key，value)组成的 tuple 的 list

合并字典：

方法1：

dictMerged1=dict(dict1.items()+dict2.items())  
方法2：

dictMerged2=dict(dict1,\*\*dict2)  
方法2等同于：

dictMerged=dict1.copy()  
 dictMerged.update(dict2)

### Tuple

不可变list，创建后不可改变

t = ("a", "b", "mpilgrim", "z", "example")

### List

li = ["a", "b", "mpilgrim", "z", "example"]

li[0]

li[-1]

li.append("new")

li.insert(2, "new")

li.extend(["two", "elements"])

li.index("example")

"c" in li //返回true or false

li.remove("z")

li.pop()

li = li + ['example', 'new']

li += ['two']

li = [1, 2] \* 3

//切片返回的是一个新的list

li[1:3]

li[:3]

li[:]

### 布尔型

·0 为 false; 其它所有数值皆为 true。

·空串 ("") 为 false; 其它所有字符串皆为 true。

·空 list ([]) 为 false; 其它所有 list 皆为 true。

·空 tuple (()) 为 false; 其它所有 tuple 皆为 true。

·空 dictionary ({}) 为 false; 其它所有 dictionary 皆为 true。

### 映射

li = [1, 9, 8, 4]

[elem\*2 for elem in li]

["%s=%s" % (k, v) for k, v in params.items()]

return ";".join(["%s=%s" % (k, v) for k, v in params.items()])

>>> li = ['server=mpilgrim', 'uid=sa', 'database=master', 'pwd=secret']

>>> s = ";".join(li)

>>> s

'server=mpilgrim;uid=sa;database=master;pwd=secret'

>>> s.split(";") (1)

['server=mpilgrim', 'uid=sa', 'database=master', 'pwd=secret']

>>> s.split(";", 1) (2)

['server=mpilgrim', 'uid=sa;database=master;pwd=secret']

### 字符串

<http://www.douban.com/note/327175040/>

http://www.jb51.net/article/33631.htm

## 技巧

# 基础

## 常用方法

### Print

print pwd + " is not a good password for " + uid

print "%s is not a good password for %s" % (pwd, uid)

print "Users connected: %d" % (userCount, )

print "Users connected: " + userCount //error

### String

## 内置函数

内置函数都归组到了 \_\_builtin\_\_

<http://python.usyiyi.cn/python_343/library/index.html#library-index>

* str 将数据强制转换为字符串。每种数据类型都可以强制转换为字符串。
* type 函数返回任意对象的数据类型
* help(obj) 在线帮助, obj可是任何类型
* callable(obj) 查看一个obj是不是可以像函数一样调用
* repr(obj) 得到obj的表示字符串，可以利用这个字符串eval重建该对象的一个拷贝
* eval\_r(str) 表示合法的python表达式，返回这个表达式
* dir(obj) 查看obj的name space中可见的name
* hasattr(obj,name) 查看一个obj的name space中是否有name
* getattr(obj,name) 得到一个obj的name space中的一个name
* setattr(obj,name,value) 为一个obj的name space中的一个name指向vale这个object
* delattr(obj,name) 从obj的name space中删除一个name
* vars(obj) 返回一个object的name space。用dictionary表示
* locals() 返回一个局部name space,用dictionary表示
* globals() 返回一个全局name space,用dictionary表示
* type(obj) 查看一个obj的类型
* isinstance(obj,cls) 查看obj是不是cls的instance
* issubclass(subcls,supcls) 查看subcls是不是supcls的子类
* callable　是不是可执行方法
* iter 变成序列
* def is\_class(o): return isinstance(o, (types.ClassType, type))
* isinstance(f, basestring)
* dir([obj]) 显示对象的属性，如果没有提供参数， 则显示全局变量的名字
* int(obj) 将一个对象转换为整数
* len(obj) 返回对象的长度
* open(fn, mode)
* range([[start,]stop[,step]) 返回一个整数列表。起始值为 start, 结束值为 stop - 1; start默认值为 0， step默认值为1。
* raw\_input(str) 等待用户输入一个字符串， 可以提供一个可选的参数 str 用作提示信息。

>>> from apihelper import info

>>> import \_\_builtin\_\_

>>> info(\_\_builtin\_\_, 20)

## 类型转化

* chr(i) 把一个ASCII数值,变成字符
* ord(i) 把一个字符或者unicode字符,变成ASCII数值
* oct(x) 把整数x变成八进制表示的字符串
* hex(x) 把整数x变成十六进制表示的字符串
* str(obj) 得到obj的字符串描述
* list(seq) 把一个sequence转换成一个list
* tuple(seq) 把一个sequence转换成一个tuple
* dict(),dict(list) 转换成一个dictionary
* int(x) 转换成一个integer
* long(x) 转换成一个long interger
* float(x) 转换成一个浮点数
* complex(x) 转换成复数
* max(...) 求最大值
* min(...) 求最小值

## 自省

method=getattr(self,callback,None)

if callable(method):

#print 'call'

method(self.sheet.row(rx));

def info(object, spacing=10, collapse=1):

"""Print methods and doc strings.

Takes module, class, list, dictionary, or string."""

methodList = [method for method in dir(object) if callable(getattr(object, method))]

processFunc = collapse and (lambda s: " ".join(s.split())) or (lambda s: s)

print "\n".join(["%s %s" %

(method.ljust(spacing),

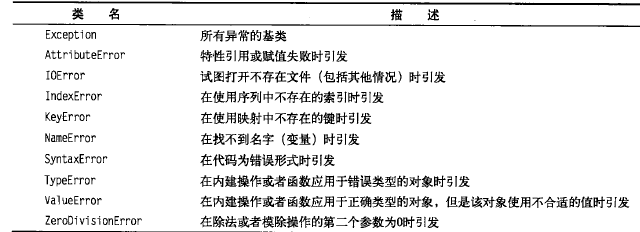
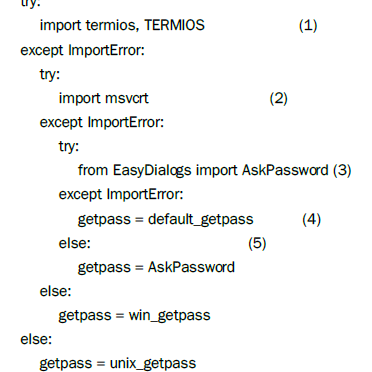
processFunc(str(getattr(object, method).\_\_doc\_\_)))

for method in methodList])

if \_\_name\_\_ == "\_\_main\_\_":

print info.\_\_doc\_\_

## 异常



## 面向对象

### 基本类

class people:

    '这是个表示人的类'  #class注释

    ''' 类属性   实例可以改变自己的，但是改变不了别人的

        通过类名可以改变所有实例的

        实例名.name 会为实例增加一个数据属性覆盖类属性，通过实例名.\_\_class\_\_.name才能调用到

    '''

    name='jack'

    #私有变量，外部访问不到

    \_\_age=12

    #constructor

    def \_\_init\_\_(self):

            #改变类属性值

            self.\_\_class\_\_.name='wang'

            #数据属性，公用变量，外部可访问

            self.pname='wang'

    def getAge(self):

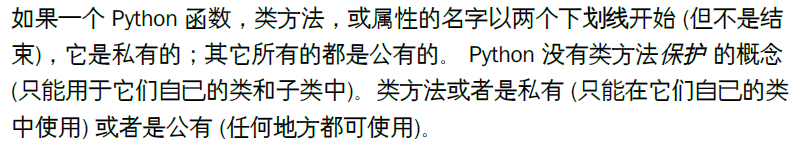
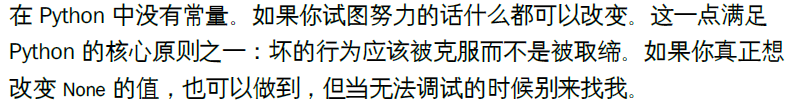
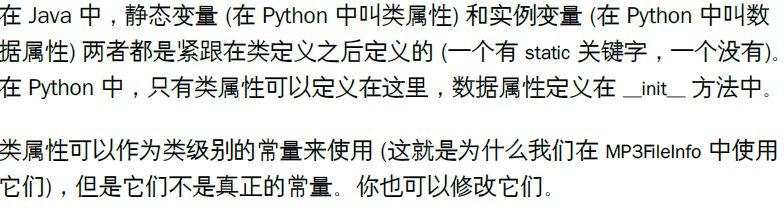
        return self.\_\_age;

    def setAge(self,age):

        self.\_\_age=age

a=people()

print a.pname



### 专用类方法

\_\_getitem\_\_("name")

def \_\_setitem\_\_(self, key, item):

### 私有属性，方法

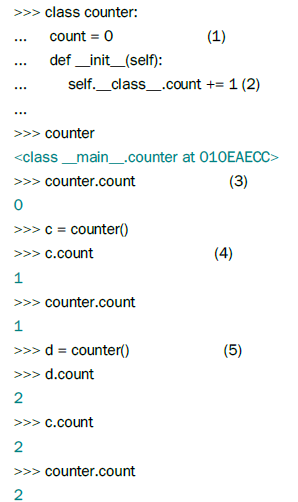
前面带2个下划线的方法为私有, \_\_parse

私有函数不可以从它们的模块外面被调用

·私有类方法不能够从它们的类外面被调用

·私有属性不能够从它们的类外面被访问

类属性：



### 方法名字特别

前后带有两个下划线的为专有方法或者内置属性 \_\_doc\_\_

前面只有下划线的为私有 \_\_parse

·

### 调用父类

<http://blog.163.com/wangbo_tester/blog/static/128067921200981741646428/>

初始化：

#UserDict.\_\_init\_\_(self,{"us":"test"})

super(FileInfo,self).\_\_init\_\_()

调用：

UserDict.clear(self)

super(FileInfo,self).clear()

Python3:

super().\_\_init\_\_()

super().clear()

### 与java区别

1，不支持函数重载，函数名字不能一样，子类同样函数名的覆盖父类的即使参数列表不一样

Python 的原作者 Guido 是这样解释方法覆盖的：“子类可以覆盖父类中的方法。因为方法没有特殊的优先级设置，父类中的一个方法在调用同类中的另一方法时，可能其实调用到的却是一个子类中覆盖父类同名方法的方法。

## 常用知识

### 格式化数字

round(2.665, 2) a =("%.2f"%2.635) 格式化浮点数

from　math import ceil, floor 取整数

### 列推到式

[i for i in range(10) if i%2==0]

[elem for elem in li if len(elem) > 1]

[elem for elem in li if elem != "b"]

[elem for elem in li if li.count(elem) == 1]

### 序列

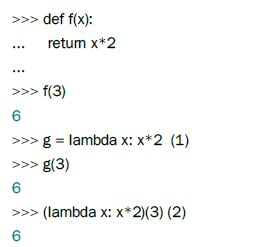
seq=['one','two','three']

for i,ele in enumerate(seq):

seq[i]='%d:%s' %(i,ele)

### and or

### Lambda函数



总的来说，lambda 函数可以接收任意多个参数 (包括可选参数) 并且返回单个表达式的值。lambda 函数不能包含命令，包含的表达式不能超过一个。不要试图向 lambda 函数中塞入太多的东西；如果你需要更复杂的东西，应该定义一个普通函数，然后想让它多长就多长。

### 可变参数\*args ，\*\*kwargs

当函数的参数不确定时，可以使用\*args 和\*\*kwargs，

\*args 没有key值是个list，

\*\*kwargs有key值,是tuple

### 空判断

可以使用 == None 或者 或者 或者 is None 进行比较，但是 is None 更快。

### \_\_all\_\_, \_\_slots\_\_

\_\_all\_\_可用于模块导入时限制，如：  
from module import \*  
此时被导入模块若定义了\_\_all\_\_属性，则只有all内指定的属性、方法、类可被导入~  
若没定义，则模块内的所有将被导入。  
  
\_\_slots\_\_用于限定类属性，如：  
class A(object):  
  \_\_slots\_\_ = ['var']  
  
此时外部调用时，如：  
a = A()  
a.var = 4    #不会报错  
a.other = 4  #此时则会抛出异常AttributeError

### 全局变量

WPSAPP = None

def **initApp**():

global WPSAPP

WPSAPP=1

### 修饰符

import sys

import functools

debug\_log=sys.stderr

//带参数

def **trace**(name):

def **funcWrap**(func):

*@functools.wraps*(func)

def **call**(\*args,\*\*kwargs):

debug\_log.write(name+*" call function:{}\n"*.format(func.\_\_name\_\_))

rs=func(\*args,\*\*kwargs)

debug\_log.write(*"end call function:{}\n"*.format(func.\_\_name\_\_))

return rs

return call

return funcWrap

def **trace2**(func):

*@functools.wraps*(func)

def **call**(\*args,\*\*kwargs):

debug\_log.write(*" call function:{}\n"*.format(func.\_\_name\_\_))

rs=func(\*args,\*\*kwargs)

debug\_log.write(*"end call function:{}\n"*.format(func.\_\_name\_\_))

return rs

return call

### 常用修饰符

<http://blog.csdn.net/openxmpp/article/details/8779691>

@classmethod

用classmethod修饰表示这是一个类方法,如果没有用@classmethod修饰，则表示这是一个对象方法，使用时必须与一个对象绑定，如下

@staticmethod修饰符

被staticmethod修饰符的表示这是一个类的静态方法，可以被类直接调用，与@classmethod的区别在于classmethod对应的方法的第一个参数为cls,为类的类型，而staticmethod不是

### 定时调用

http://www.pythontab.com/html/2013/pythonjichu\_0119/146.html

# 常用

## 常用模块

### Print

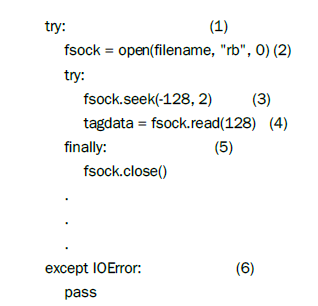
### 文件操作

f = open("/music/\_singles/kairo.mp3", "rb")

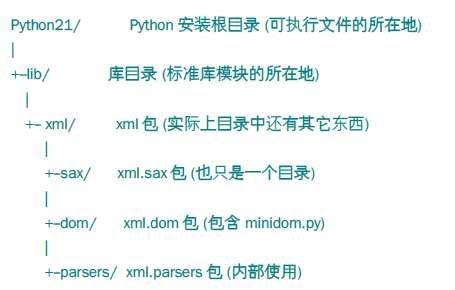
f.tell()

f.read(128)

F.close()



### xml操作



### 正则表达式

### JSON

### 单元测试

### 目录操作

os.path

os.listdir

Glob.glob 搜索子文件

Os.stat 获取状态

Os.chdir

os.getCwd

Os.path.realpath

### 调用DLL

<http://www.pythonfan.org/thread-10794-1-1.html>

http://www.linuxidc.com/Linux/2011-10/44837.htm

from ctypes import \*

# Simple Test on c\_int object

i=c\_int(5)

print i

print i.value

i.value=10

print i.value

# Import Dll

TestDll=CDLL('TestDll.dll')

# Test Print Function

TestDll.hello()

# Test variable dilivery

a=c\_int(4)

b=c\_int(6)

c=TestDll.SumNumbers(a,b)

print c

### C生成DLL

<http://www.linuxidc.com/Linux/2011-10/44837.htm>

<test.cpp 生成动态库的源文件>

1. #include <stdio.h>
2. **extern** "C" {
3. **void** display() {
4. printf("This is Display Function\n");
5. }
6. }
7. g++ test.cpp -fPIC -shared -o libtest.so

<call.py 调用动态库的源文件>

1. **import** ctypes
2. so = ctypes.CDLL("./libtest.so")
3. so.display()

<test.cpp 生成动态库的源文件>

1. #include <Akita/Akita.h>
2. **class** TestLib{
3. **public**:
4. **void** display();
5. **void** display(**int** a);

8. };
9. **void** TestLib::display() {
10. cout<<"First display"<<endl;
11. }

14. **void** TestLib::display(**int** a) {
15. cout<<"Second display"<<endl;
16. }
17. **extern** "C" {
18. TestLib obj;
19. **void** display() {
20. obj.display();
21. }
22. **void** display\_int() {
23. obj.display(2);
24. }
25. }

### 操作WPSoffice

需要安装 pywin32程序

Word转pdf

wpsApp = win32com.client.Dispatch(*"kwps.application"*)

wpsApp.Visible=True

doc = wpsApp.Documents.Open(*"E:\\iso\\print\\pdf\\test.docx"*)

doc.ExportAsFixedFormat(*"E:\\iso\\print\\pdf\\test.pdf"*,17)

doc.Close()

### 操作Excel

http://blog.csdn.net/mr\_\_fang/article/details/7089581

### C++

宏：

|  |  |
| --- | --- |
| \_\_LINE\_\_ | 这会在程序编译时包含当前行号。 |
| \_\_FILE\_\_ | 这会在程序编译时包含当前文件名。 |
| \_\_DATE\_\_ | 这会包含一个形式为 month/day/year 的字符串，它表示把源文件转换为目标代码的日期。 |
| \_\_TIME\_\_ | 这会包含一个形式为 hour:minute:second 的字符串，它表示程序被编译的时间。 |

#define，#ifndef，#endif

## 常用代码

序列化：

try: import cPickle as pickleexcept ImportError: import pickle

webpy 生成并下载excel:

        web.header('Content-type','application/vnd.ms-excel')  #content-type需要根据实际的文件类型来指定

        web.header('Transfer-Encoding','chunked')

        web.header("Content-Disposition","attachment;filename=config.xls")

        wbk = xlwt.Workbook()

        sheet = wbk.add\_sheet('test')

        sheet.write(0,1,'test text')

        sio=StringIO.StringIO()

        wbk.save(sio)

        return sio.getvalue()

执行系统命令获取返回：

 a= os.popen("ping backend-service-dispatcher.yihaodian.com").readlines()

执行hessian

import hessian.client

 proxy = hessian.client.HessianProxy(stg\_url)

    try:

        obj= proxy.call(jsonStr)

        return strObj(obj)

    except Exception,e :

        traceback.print\_exc()

编码显示汉字：

 try:

            print sql+buildValues(dataType,title,sheet.rows(rx))

            cursor.execute(sql+buildValues(dataType,title,sheet.rows(rx)))

        except Exception as e:

            print str(e).decode("gbk").encode("utf-8")

生成临时文件：

 tfile = tempfile.mktemp('.xls')

### 2，3区别

http://blog.csdn.net/samxx8/article/details/21535901

# 第三方模块

### excel

xlrd 读取excel

xlwt 写excel

<http://www.cnblogs.com/itech/archive/2011/03/18/1986656.html>

### itertools 迭代模块

### itertools 迭代模块

# 问题

### Mysql安装

<http://blog.csdn.net/ypersistence/article/details/53188432>

### 加速镜像

~/pip/pip.ini

[global]

index-url=http://mirrors.aliyun.com/pypi/simple/

[install]

trusted-host=mirrors.aliyun.com