# 笨方法学python

## Class 1

* 如何验证python安装成功：

进入powershell后键入python，显示相关信息及说明安装成功环境变量设置成功，若未成功，可输入[Environment]::SetEnvironmentVariable("Path","$env:Path;安装路径", "User")。

* Powershell相关命名：

1、New-Item -路径 -name 文件或文件夹名 -type 文件类型

2、内容 |Out-File 文件路径+文件名（利用管道给文件里传入相关内容）

* Python打印格式：

例：print "Let's talk about %s." %my\_name

%的用法：是一种将其他变量置入到字符串的特定位置以形成新的字符串的运算。

* 打印细节：

1、print "." \* 10 可打印10个点

2、，打印时可变为空格

3、当打印的字符串中含有单引号时，原来要打印的双引号会变成单引号。

4、三个双引号表示打印多行字符串。

## Class2

* Python的输入：

Raw\_put(),()里面可跟提示，以引号引起来。

* 输入python -m pydoc 命令，可查看相关命令的帮助文档。
* 参数、解包、变量：

from sys import argv ：import用来导入模组，此条语句的意思为将系统模组的参数导入到脚本中。

script, first, second, third = argv：用来解包

* 读写相关命令：

1. open（文件名）：打开文件并赋给某个变量。
2. read：读取文件内容，

例：f.read()

1. close（）：关闭文件，

例：f.close()

1. readline：用来读取文本文件中的一行
2. truncate：清空文件，一般在写入文件之前用此命令，保证文件符合预期
3. write：用来把数据写入文件

* from os.path import exists:引入了exists命令，以文件名作为参量，如果文件存在，会输出true，否则会输出false
* 函数：

函数格式：

def 函数名（）：

函数体

注意：使用函数时，函数体必须缩进，否则会进行报错。

* seek()：可用来移动指针

## class3

* 几种函数的用法：

1. split：对字符串按照指定命令进行分割
2. sort：排序
3. pop：用于移除列表中的一个元素，默认最后一个元素（排序从0开始）

* 可以通过python命令进入python进行交互，通过import命令引入需要的文件，通过函数名调用函数；可以通过help命令来获取函数的说明（前提是有）；通过from 文件名 import \*引入文件方便一些
* If语句：

1、结构：if（）：

代码

2、if……elif……，这是与其他语言不同的地方。

* For语句：

1、结构：for……in……：

代码

2、range函数：

例：range（1,5）=[1,2,3,4]顺序（0,1,2,3）或（-4，-3，-2，-1）

Array=[1,2,3,4,5],array[2:]=[4]，array[:2]=[1,2]

Array[::2]=[1,3,5],array[2::]=[3,4,5]

1. append方法：将对象添加到list末尾

* while语句：

结构：while 判断条件

代码

* 小技巧：调试时使用print打印变量，尽量不要使用debugger；if后面最好加一个else，如果该else无意义，可以加一个die函数，打印错误信息结束进程。
* Join：将字符串按照指定的符号连接成一个

## Class4

* Dict：字典，通过索引可以访问到具体内容

格式：数组名[索引名：内容，……]

Dict型数据结构是无序的。

Exit（0）无错误退出，exit（1）有错误退出

* Class：

1. 结构

Class 类名（object）：

Def……

Def……

2、Self：通用参数，代指实例，只有有了它，才能调用类里面的各种方法

例：def init（self）……a.init()……

1. pass：空命令

## class5

* 建立项目骨架：

# 创建骨架目录，每次只输入一行代码，回车

mkdir projects

cd projects/ # 进入projects 文件夹，后面要有一个 /

mkdir skeleton

cd skeleton

mkdir bin

mkdir NAME

mkdir tests

mkdir docs

# Windows环境下配置初始文件

new-item -type file NAME/\_\_init\_\_.py

new-item -type file tests/\_\_init\_\_.py

查看目录命令：ls -r

测试命令：nosetests

* 元组：用圆括号，逗号隔开，如first\_word = ('direction', 'north')
* Pip安装软件包问题：

安装pip时进入其所在目录，在命令行输入python 文件名 install，安装成功

解决方案就是在python安装目录: python27\Lib\site-packages中新建一个文件sitecustomize.py，内容为：

import sys

sys.setdefaultencoding('gb2312')

另外，在Windows系统下，wordcloud直接安装可能会虚，试试在Python Extension Packages for Windows的第三方平台下载wordcloud的whl文件，放在pip所在的Scripts文件夹中，然后在本地用pip安装。

注意，cp27的意思就是匹配python2.7 ，win\_amd64即64位的操作系统。下载好之后在cmd中：

pip install wordcloud‑1.3.1‑cp27‑cp27m‑win\_amd64.whl

在这里，我的电脑是64位的，但是安装时说版本不匹配……于是又试了一下win32的版本，安装成功。

# 廖雪峰教程

## Class6

* …代表另起一行，例：

print '''line1

... line2

... line3'''

* 解决中文乱码问题：

电脑采用Unicode编码，而在传输时，为了节省资源提高传输速率，会使用UTF-8编码，为了不使文件里面汉字乱码，可在开头添加

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

有时需要转码，示例如下：myname.decode(‘UTF-8’).encode(‘GBK’)

Unicode方法也是编码的方法

* Insert函数：在列表的制定位置添加元素

classmates.insert(1, 'Jack')

* 定义元组tuple时，如果只有一个元素，应写成以下形式：

t = (1,)

元组所谓的不变，是指向的地址不变

* 以raw\_input输入的数据永远是字符串形式，如果要进行其他处理，需要进行数据转换
* 在字典中，判断索引是否存在有两种方法，一种是通过in，例如：'Thomas' in d；另一种是通过get方法，例如：d.get('Thomas')
* Set：使用示例

S = Set([1,2,3])

添加元素：s.add(4)

删除元素：s.remove(4)

* 数据类型检查可以使用isinstance,例如：isinstance(x, (int, float))
* 函数设置参数注意事项：必要参数在前，默认参数在后；变化大的参数在前，变化小的参数在后

注意：默认参数需要指向一个不变的数据

* 可变参数：示例：def calc(\*numbers):这样的话，默认输入的是一个元组
* 关键字参数：示例：def person(name, age, \*\*kw):

可用来扩展函数，根据用户的需要传递更多的参量，默认输入的是一个dict

注意：参数定义的顺序必须是：必选参数、默认参数、可变参数和关键字参数

## Class7

* 如何判断一个对象是否是迭代对象：使用collections模块的Iterable类型判断，例：

isinstance('abc', Iterable)

* Python内置的enumerate 函数可以把一个list变成索引元素对
* 列表生成式
* Generator：一种形式是列表生成式的中括号变为小括号；另一种形式是函数中定义了关键字yield，这种函数的执行顺序是next（）方法到了yield处终止，再次执行时再从yield的地方开始。其实质保存的是算法，其容量可以算是无限。
* 高阶函数：函数作为一个参量输入到另一个函数中。

1. map（）：传入两个参量，第一个是函数，第二个是list，其作用是将函数作用于list中的每一个元素，结果作为一个list输出
2. reduce（）：传入两个参量，第一个是函数，第二个是list，其作用是将函数作用于第一个跟第二个元素（姑且如此描述），然后函数作用在结果跟第三个元素上，以此类推。
3. filter（）：传入两个参量，第一个是函数，第二个是list，其作用是将函数作用于list的每个元素，如果结果为0则丢弃，是1则保留
4. sort（）：可以作为高阶函数使用，可自定义排序函数

* 闭包：函数之中又定义了函数，且内部函数引用了外部的参数和局部变量，又返回了这个函数，因此，这个函数已经包括了相关变量以及参数
* 返回闭包时牢记的一点就是：返回函数不要引用任何循环变量，或者后续会发生变化的变量
* 匿名函数：示例：lambda x: x \* x，lambda代表匿名函数，x代表传入的参量，只能有一个表达式，无return函数
* 装饰器：本质上一个返回函数的高阶函数，在代码运行期间动态的增加函数的功能，利用@语法，置于函数的定义处，示例：

@log

def now():

print '2013-12-25'

* 偏函数：利用functools.partial，将函数的某些参数固定，即设定默认值，使用示例：

>>> import functools

>>> int2 = functools.partial(int, base=2)

* 导入模块使用别名：

try:

import cStringIO as StringIO

except ImportError: # 导入失败会捕获到ImportError

import StringIO

try:

import json # python >= 2.6

except ImportError:

import simplejson as json # python <= 2.5

* 特殊变量，例如：\_ \_name\_ \_

Private变量，例如：\_abc/\_ \_abc

外部不需要引用的函数全部定义成private，只有外部需要引用的函数才定义为public。

* 修改模块搜索路径：

>>> import sys

>>> sys.path.append('/Users/michael/my\_py\_scripts')

* \_future\_:在高版本中试验低版本的语法，例如：

from \_\_future\_\_ import division

* 继承与多态
* 判断对象类型：使用type（）函数，可以利用types模块来简化type的使用方法

判断类的类型：使用isinstance方法，

获得一个对象所有属性和方法：使用dir（）方法，配合getattr() 、setattr() 以及hasattr() ，我们可以直接操作一个对象的状态

## Class8

* 给实例绑定属性与方法：绑定属性直接在实例后面加.属性名；绑定方法需要编写方法，具体使用如下：

>>> def set\_age(self, age):

... self.age = age

...

>>> from types import MethodType

>>> s.set\_age = MethodType(set\_age, s, Student)

>>> s.set\_age(25) # 调用实例方法

>>> s.age # 测试结果

给一个实例绑定方法后，对另一个实例是不起作用的，如果要对所有的实例起作用，可以绑定类，如：Student.set\_score = MethodType(set\_score, None, Student)

* 限制绑定类的属性：使用\_slots\_,示例：

>>> class Student(object):

... \_\_slots\_\_ = ('name', 'age') # 用tuple定义允许绑定的属性名称

...

注意：slots只对父类起作用，对继承的子类不起作用

* 将方法变成属性：使用@property，可以设置setter属性变为可读，也可不设置变为只读
* 多重继承
* 打印有关的方法：\_ \_str\_ \_，\_ \_repr\_ \_
* 如果一个类想要被用于for循环，必须实现\_\_iter\_\_()方法，例如：

class Fib(object):

def \_\_init\_\_(self):

self.a, self.b = 0, 1 # 初始化两个计数器a，b

def \_\_iter\_\_(self):

return self # 实例本身就是迭代对象，故返回自己

def next(self):

self.a, self.b = self.b, self.a + self.b # 计算下一个值

if self.a > 100000: # 退出循环的条件

raise StopIteration();

return self.a # 返回下一个值

* 要表现得像list那样按照下标取出元素，需要实现\_\_getitem\_\_() 方法：
* \_\_getattr\_\_() 方法，动态返回一个属性
* \_\_call\_\_()方法：可以将一个示例变得跟函数一样，方便调用

通过callable() 函数，我们就可以判断一个对象是否是“可调用”对象。

* Type()：可以用来动态的创建类，共传入三个参数，第一个是类名，第二个是继承的父类，以元组的形式，第三个是绑定函数。如下所示：

Hello = type('Hello', (object,), dict(hello=fn))

* Metaclass():类可以看做其创建出的实例，所以先定义metaclass，再创建类，最后创建实例

Metaclass创建类，必须从type类型派生。

举例：

# metaclass是创建类，所以必须从`type`类型派生：

class ListMetaclass(type):

def \_\_new\_\_(cls, name, bases, attrs):

attrs['add'] = lambda self, value: self.append(value)

return type.\_\_new\_\_(cls, name, bases, attrs)

class MyList(list):

\_\_metaclass\_\_ = ListMetaclass # 指示使用ListMetaclass来定制类

* 导入模块：

>>>importsys

>>>sys.path.append(main所在路径)

执行模块

Import 模块名

## Class9

* 错误处理机制：try……except……finally……

Finally一定会执行，所有的错误类型都继承于BaseException

引入logging模块可以记录错误信息，并让程序正常执行，使用如下：

try:

bar('0')

except StandardError, e:

logging.exception(e)

* 可以使用raise，抛出自己定义的错误，但是要选择好继承关系；也可以利用此语句将错误主动抛出，交由调用者来处理
* 可以使用assert语句来断言输出错误，logging可以输出错误信息到文本，使用示例：

import logging

logging.basicConfig(level=logging.INFO)

logging.getLogger().setLevel(logging.WARNING)

…

logging.info()

* Pdb：调试命令，输入python -m pdb 程序名即可启动，n命令可执行下一行代码，p 变量名可查看变量，q命令可以退出；利用引入pdb包，在代码中使用pdb.set\_trace()设置断点，程序运行到断点处自动进入调试环境，命令c继续执行
* 编写单元测试模块：需要引入unittest模块，测试类需要从unittest.TestCase继承，通常使用assertEquals()来断言输出是否是我们所需要的，或者使用assertRaises()来指定抛出的类型；运行测试模块，可以在测试模块下编写如下代码：

If \_\_name\_\_ == ‘\_\_main\_\_’

Unittest.main()

或者利用-m unittest参数直接运行测试文件

With关键字定义的语句无论执行时发生什么异常，最终都会关闭文件

* setUp打开数据库，tearDown关闭数据库
* doctest:可以提取注释中的代码并加以测试，在最后，可加上如下代码执行测试：

if \_\_name\_\_ == ‘\_\_main\_\_’:

import doctest

doctest.testmod()

## class10

* 流的概念，同步IO与异步IO的概念
* 读写文件：open,read,close……；with关键词；read(size),readline,readlines
* 读文件时自动转换编码：示例如下：

Import codecs

With codecs.open(‘……’) as f:

f.read()

* 操作文件或目录可以通过os模块，通过os.name可以查看当前操作系统，os.environ可以查看环境变量，os.getenv()可以查看指定环境变量的值；os.path.join用来拼接目录，os.mkdir用来创建目录，os.rmdir用来删除目录，os.path.split可以用来拆分目录，os.path.splitext可得到文件的扩展名，os,rename用来重命名，os.remove用来删除文件
* 序列化：指将变量从内存中变成可存储或可传输的过程，需要引入pickle包，利用如下代码：

try:

import cPickle as pickle

except ImportError:

import pickle

利用pickle.dumps将变量序列化成字符串，使用pickle.dump将变量序列化，并输出到文件中，pickle.load用于反序列化

* JSON:用于在不同的编程语言间传递对象，可使用json模块将python语言中的变量变为标准形式，使用dumps与load方法

将类序列化json可用如下代码：print(json.dumps(s, default=lambda obj: obj.\_\_dict\_\_))

## Class11

* 多进程：window环境下通过引入multiprocessing这个包来实现；利用pool来实现多个子进程
* 多线程：需要引入thread/threading包，使用current\_thread()方法可以查看当前线程的实例；可以通过使用lock.aquire与lock.release来加锁与释放锁。
* Threadlocal：类似于全局变量，可以被各个线程访问，附加属性。
* 正则表达式：给字符串定下规则看是否合法，用\*表示任意个字符，用+表示不止一个字符，用{n}表示n个字符，用{n，m}表示n-m个字符，用？表示0个或1个字符，更高级可以用[]来限定范围，如果想要使用非贪婪匹配，可以在后面加？；当需要预编译的时候，使用re.compile方法
* Collections：可以利用namedtuple创建一个元组对象，示例如下：n = namedtuple( ‘n’,’[x,y]’);deque是用来向列表中高效插入或删除的双向列表；defaultdict可用来创建字典，但其中可定义当索引不存在时的返回值；Ordererdict是用来生成有序key的字典；Counter是一个计数器
* Base64是一种可以将二进制到字符串的编码方式
* Struct：可以用来转换字符串
* Hashlib：摘要算法，可以通过分析得到摘要，判断是否被篡改；可以使用md5，例如md5 = hashlib.md5(),然后利用update方法传入数据，hexdigest方法得到摘要；sha1与md5类似；可以通过加盐来提高安全性+‘the-Salt’
* Itertools: chain() 可以把一组迭代对象串联起来，形成一个更大的迭代器, groupby() 把迭代器中相邻的重复元素挑出来放在一起;imap()针对无穷序列
* 解析Html：

## Class12

* Image:通过Image.open可以打开一张图片，size方法可以获得尺寸，thumbnail方法可以缩放，save方法可以保存图像；引入ImageFilter可以使用各种滤镜；
* GUI:编写GUI界面，可以引入Tkinter包，父类为Frame，每生成一个widget就利用pack方法加入父容器中
* TCP:

1. 客户端：引入socket库，然后创建一个socket，即利用socket.socket方法，即打开了一个网络链接，然后利用connect方法建立链接，利用send方法发送请求，recv方法接受数据
2. 服务器：bind方法绑定端口，listen方法监听，accept方法接受请求

* UDP:与TCP不冲突，使用9999端口，不需要建立连接，recvfrom返回数据和客户端的端口和地址，调用sendto将数据发给客户端

## Class13

* 发送邮件：从email.mime.text引入MIMEText来构造邮件对象，如：msg = MIMEText('hello, send by Python...', 'plain', 'utf-8')，引入smtplib模块来发送邮件，smtplib.SMTP方法来连接端口，set\_debuglevel方法来打印和服务器的交互信息，login() 方法用来登录SMTP服务器， sendmail() 方法就是发邮件
* 解析邮件：pop3
* 数据库：引入sqlite3模块，导入SQLite驱动，通过connect方法连接到数据库，通过cursor方法打开一个游标，execute方法执行命令

## Class14

* http格式：

GET /path HTTP/1.1

Header1:Value1

Header2:Value2

换行符为\r\n

POST格式：

POST /path HTTP/1.1

Header1:Value1

……

当遇到连续两个\r\n时进入body部分

* WSGI:首先需要编写WSGI处理函数，然后编写一个启动WSGI服务器的函数，引入相应的模块：from wsgiref.simple\_server import make\_server，再导入处理函数，之后利用make\_server创建服务器，利用serve\_forever来监听

# Python核心编程

## Class15

* \代表继续当前行
* \_\_name\_\_如果等于当前模块的名字，则代表模块是被导入，如果\_\_name\_\_=’\_\_main\_\_’，则代表模块是被执行
* Str可以将内容变为字符串表示
* 对于复数，real方法取实部，imag方法取虚部，conjugate（）方法取共轭
* Abs函数取绝对值，coerce（）类型转换，divmod（）得到商跟余数，round进行四舍五入，pow取多次方
* Dir方法可以得到一个列表的所有方法和属性
* 列表如果进行拷贝，二者互相不影响，可以使用深拷贝copy的deepcopy方法
* 正则表达式：

1. 在正则表达式中，\d与\D表示完全相反的概念，管道符|类似于或的运算；.可以匹配除了换行符\n的任意字符；\b和\B用来匹配字符边界
2. match从字符串的开始进行匹配，如果开始匹配不上，则匹配失败，search可以从字符串进行搜索进行匹配

注意：字符串前加r可防止字符串字符转义

1. findall方法类似于rearch，不过生成的是一个列表；finditer查找的结果可迭代
2. sub方法用于将正则表达式的中某字符用其他字符替代，subn方法除了替代之外，还可以给出替代的数目
3. \N在替代时表示分组的组号
4. 扩展符号（？i）可以用来做标记，其后跟需要替换或搜索的对象，（？im）同（？i），不过它是搜索的对象是多行，搜索的结果也是行

## Class16

* 套接字：是通信之前的相关准备，个人理解为通信协议的类型，python支持的套接字有AF\_UNIX、AF\_NETLINK、AF\_TIPC、AF\_INET

1. TCP套接字由SOCK\_STREAM创建，UDP套接字由SOCK\_DGRAM创建；所有的套接字都由socket.socket()方法创建
2. TCP服务端创建方法：

ss = socket() # 创建服务器套接字

ss.bind() # 套接字与地址绑定

ss.listen() # 监听连接

*inf\_loop*: # 服务器无限循环

cs = ss.accept() # 接受客户端连接

*comm\_loop*: # 通信循环

cs.recv()/cs.send() # 对话（接收/发送）

cs.close() # 关闭客户端套接字

ss .close() # 关闭服务器套接字#（可选）

TCP客户端创建方法：

cs = socket() # 创建客户端套接字

cs.connect() # 尝试连接服务器

*comm\_loop*: # 通信循环

cs.send()/cs.recv() # 对话（发送/接收）

cs .close() # 关闭客户端套接字

1. UDP服务器创建方法：

ss = socket() # 创建服务器套接字

ss.bind() # 绑定服务器套接字

*inf*\_*loop*: # 服务器无限循环

cs = ss.recvfrom()/ss.sendto() # 关闭（接收/发送）

ss .close() # 关闭服务器套接字

UDP客户端创建方法：

cs = socket() # 创建客户端套接字

*comm\_loop:* # 通信循环

cs.sendto()/cs.recvfrom() # 对话（发送/接收）

cs .close() # 关闭客户端套接字

1. SocketServer 模块：

服务器:

主机名、端口地址等准备

Class 类名（StreamRequestHandler）:

重写handle（）方法

TcpServ = TCP(ADDR, 类名) 创建TCP服务器

TcpServ.serve\_forever() 不停监听

客户端：

通信前准备

通信循环：

tcpCliSock = socket(AF\_INET,SOCK\_STREAM)创建套接字

tcpCliSock.connect(ADDR) 建立连接

进行通信

tcpCliSock.close()

1. Twisted框架：

服务器：

通信前准备

class TSServProtocol(protocol.Protocol):

重写connectionMade、dataReceived方法

factory = protocol.Factory() 创建工厂

factory.protocol = TSServProtocol 通过工厂创建实例

print 'waiting for connection...'

reactor.listenTCP(PORT, factory) 监听

reactor.run() 运行

客户端：

通信前准备

class **TSClntProtocol**(protocol.Protocol):

重写connectionMade、dataReceived方法

class TSClntFactory(protocol.ClientFactory):

创建工厂

reactor.connectTCP(HOST, PORT, TSClntFactory()) 连接

reactor.run() 运行

注意：import跟from……import……略有不同

## Class17

* 文件传输因特网协议：主要有FTP、UUCP、HTML、rcp等
* FTP：

连接FTP服务器的方法：

from ftplib import FTP

f = FTP('some.ftp.server')

f.login('anonymous', 'your@email.address')

:

f. quit()

* 网络新闻：

from nntplib import NNTP

n = NNTP('your.nntp.server')

r,c,f,l,g = n.group('comp.lang.python')

...

n. quit()

* 电子邮件：

1、SMTP

from smtplib import SMTP

n = SMTP('smtp.yourdomain.com')

...

n. quit()

2、POP3：

from poplib import POP3

p = POP3('pop.python.is.cool')

p.user(...)

p.pass\_(...)

...

p. quit()

3、IMAP:

from imaplib import IMAP4

s= IMAP4('imap.python.is.cool')

s.login(...)

...

s.close()

s. logout()

* 当电子邮件有多个内容时，应使用MIMEMultipart模块，参数选定alternative（防止多个内容以附件的形式添加），利用attach方法添加文本、html内容等；当有图片作为附件时，使用MIMEImage模块创建，相关参数有图片的内容及名字
* 解析电子邮件：使用email.message\_from\_string()来解析电子邮件，利用walk()方法来遍历信息附件，通过get\_content\_type()方法来获得正确的MIME类型，通过get\_payload()来从消息正文中获取特定的部分

## Class18

* Thread模块：

在thread模块中，通过start\_new\_thread(function, args, kwargs=None)来创建线程；allocate\_lock()来创建一个锁；通过acquire方法来将锁锁上，release方法来释放锁；

注意：在主线程结束之前务必保证子线程结束，否则子线程会全部清除。

* Threading模块：

在threading模块中，通过threading.Thread来创建线程；通过start方法来开始线程；join方法来等待子线程结束

* 锁与信号量，都是一种同步机制，此处对锁不做描述，对于信号量，相当于一个计时器，使用release方法，信号量计时器加1，acquire方法计时器减一，需要注意的是，除非计数器加一，否则acquire方法会堵塞
* 队列Queue

## Class19

* 在GUI中，首先要做的是创造一个根窗口，然后在其中创建相应的控件即可；首先需要引入Tkinter的包，利用其中的Tk模块创建根窗口，在所有的GUI程序写完之后，利用mainloop方法无限循环，等待在GUI窗口上的操作

+：eval函数可将字符串变为表达式进行运算并返回相应的值

* 将scrollbar与listboxs联合起来：将如listboxs的这些控件的yscrollcommand方法设为scrollbar的set方法，将scrollbar的command方法设为控件的yview方法
* 可以通过bind方法将事件与函数联系起来，通过某些动作来操作GUI
* StringVar是一种变量的定义方式，通过set跟get方法来改变或获得变量值，当组件的textvariable属性是StringVar类型时，如果值改变，组件的显示会实时改变

## Class20

* 在数据库中，底层存储指文件系统等持久化存储；用户接口指可以通过代码与数据库进行通信；数据库指管理数据的系统；
* 在数据库中，虽然不区分大小写，但习惯将关键字大写，且习惯在每行语句后面加分号；

1. 创建数据库：

CREATE DATABASE test；

GRANT ALL ON test.\* to user(s);

1. 使用数据库：

USE test;

1. 删除数据库：

DROP DATABASE test;

1. 创建新表：

CREATE TABLE users （login VARCHAR(8), userid INT， projid INT）

1. 删除表：

DROP TABLE users;

1. 插入行：

INSERT INTO users VALUES(‘leanna’,2111,1)

1. 更新行：

UPDATE users SET projid=4 WHERE projid=2;

UPDATE users SET projid=1 WHERE userid=311;

1. 删除行：

DELETE FROM users WHERE projid=%d;

DELETE FROM users;

* MySQL:

对于这种数据库，可以使用query方法来执行数据库的相关命令，但最好用建立游标的方法来执行命令

* 对于这段程序：

if isinstance(\_\_builtins\_\_, dict) and 'raw\_input' in \_\_builtins\_\_:

scanf = raw\_input

elif hasattr(\_\_builtins\_\_, 'raw\_input'):

scanf = raw\_input

else:

scanf = input

当模块被引进时，会变成dict形式，if语句的作用就是检查raw\_input是否存在于被引入的模块中，如果不存在，说明\_\_builtins\_\_是一个模块，进入剩下的判断中，判断是哪个python版本

* SQLAlchemy：使用ORM框架

from sqlalchemy.ext.declarative import declarative\_base

Base = declarative\_base()

class Users(Base):

\_\_tablename\_\_与表名进行连接，

创建列

class SQLAlchemyTest(object):

def \_\_init\_\_(self,dsn):

try:

eng = create\_engine(dsn)创建引擎

except ImportError:

raise RuntimeError()

try:

eng.connect()连接数据库，之后可以通过对引擎的操作进行对数据库的操作

……

Session = orm.sessionmaker(bind=eng)

self.ses = Session()

self.users = Users.\_\_table\_\_

self.eng = self.users.metadata.bind = eng人为将引擎绑定数据库

……

def \_\_getattr\_\_(self,attr):

return getattr(self.users, attr)委托，当属性没有时使用别的类的属性

……

## Class21

* Excel：

Tk的withdraw方法可不让Tk顶级窗口出现

使用win32.gencache.EnsureDispatch()方法来接入excel，利用Workbooks.Add方法来新建工作簿，利用ActiveSheet来设置当前活动的工作表，利用Cells.Value方法给工作表赋值

* Word：

Word与excel基本相同，不过是用Documents.Add方法新建文档，利用InserAfter方法插入文字

## Class22

* 编写封装代码：

1. 包含头文件
2. 为函数编写形如PyObject\* Module\_func()的封装函数，利用PyArg\_ParseTuple、Py\_BuildValue来进行转码
3. 为模块编写PyMethodDef ModuleMethods[]数组
4. 添加模块初始化函数void initModule()

* 编译：

构建扩展模块，需要给每个模块创建Extension实例，所以需要引入distutils包，创建示例：Ex tension('Extest', sources=['Extest2.c'])；之后调用setup，示例如下：setup('Extest', ext\_modules=[...])。

最终，运行python setup.py build进行编译，接下来安装该扩展便可以使用

## Class23

* 防火墙，代理服务器（正向代理，反向代理）
* URL格式：prot\_sch://net\_loc/path;params?query#frag

Net\_loc可以拆分成以下部分user:passwd@host:port

* Urlparse模块：

1. urlparse.urlparse将URL分解成一个六元组，示例如下：ur lparse (urlstr, defProtSch=None, allowFrag=None)；urlunparse与之相反，示例如下：urlunparse(urltup)；urljoin示例：ur ljoin (baseurl, newurl, allowFrag=None)，取得根域名与根路径，并与newurl连接起来
2. urlopen打开一个URL，并返回文件类型，语法结构如下：urlopen (urlstr, postQueryData=None)；urlretrieve用来下载完整的html，并将其另存为文件，语法如下：urlretrieve(url, filename=None, reporthook=None, data=None)；quote和quote\_plus用来转换字符串，将不能使用的字符转换为16进制，得到有效的url，unquote与unquote\_plus与之相反；urlencode接受字典的键值对，并将其变为字符串

## class24

* 在使用CGI编程时，静态文本网页无法连接py文件，经查，改为ACTION=localhost:8000\cgi-bin\……的形式成功。

## Class26

* 使用django建立项目文件夹：django-admin.py startproject mysite；运行开发服务器：python ./manage.py runserver
* Biog：首先，创建一个应用：./manage.py startapp blog；之后将应用添加进设置里，打开settings在INSTALLED\_APPS添加应用名；打开数据模型文件，编写相应的类；在settings中的DATABASES里连接数据库（在最新版django中，没有了./manage.py syncdb命令，改为./manage.py makemigrations ,

./manage.py migrate命令）,需要开启./manage.py runserver命令才能看到admin默认界面，同时，最好注册超级用户，方面管理，命令如下：manage.py createsuperuser；之后在blog中的admin.py中进行注册，使用admin.site.register(models.BlogPost)命令；

2）编写用户端：编写模板文件，路径为模板文件的路径应该为 mysite/blog/templates/archive.html；其中，templates文件夹名固定；创建项目的urls，在urls.py中加入url(r'^blog/', include('blog.urls'))，其中inlude需要跟URL一起引入；编写应用的urls，新建urls.py文件，编写命令import blog.views

urlpatterns = [

url(r'^$', blog.views.archive),

]；创建视图函数，命令如下：

from django.shortcuts import render\_to\_response

from blog.models import BlogPost

def archive(request):

posts = BlogPost.objects.all()

return render\_to\_response('archive.html', {'posts': posts})