标准库：<https://docs.python.org/3/library/index.html>

第三方库： check out the community-run repository: <http://pypi.python.org>

第一章 basic

Python IDLE 开发环境，解释程序，F5运行结果

Functions + Modules(a collection of related function) = The Standard Library，标准库使用：

Python shell: experimenting at the >>>prompt helps you work out the code you need.

“import” brings in the named function to the program, then invoke it without using the dot-notation syntax

>>> import sys >>>print(sys.platform) >>> import os >>>print(os.getenv(‘HOME’)) >>>os.environ

>>> from os import getcwd #import the function from its module

>>> where\_am\_I = getcwd()

1. Python 变量是动态分配的；类型不需要预先声明。

Python comes with all the usual operators, including <， > , <=, >=, ==, !=, as well as the = assignment operator.

>>>import time >>>print(time.strftime("%H:%M")) #格式化输出

>>>print(time.strftime("%A %p"))

>>>time.sleep(5) #the shell imports the “time” module, letting us invoke the “sleep” function

>>>from datetime import datetime >>>right\_this\_minute = datetime.today().minute

1. 块缩进：

(1)Python中的块很容易识别，因为他们总是缩进;

(2)冒号引入一个缩进的代码组，else取消缩进配对if。elif同if。可嵌套缩进，可扩展。For loop循环。

Experimenting at the >>> prompt helps you work out the code you need.

For loop缩进: for i in [1,2,3]: print(i)。for num in range(5): print(‘head first rocks!’)

1. 缩进整块区域：indent format menu；for loop 循环

Generate产生随机数：>>>import random

Use”help” to read the python docs. windows和linux时，输入Alt-p重复最后一个命令。

>>> random.randint(1,60 )

numbers : >>>wait\_time = random\_randint(1,60)

strings : >>> word = “bottles”

1. Range(start, stop, step): such as range(99, 0, -1)，>>>list(range(0,10,2))

第二章 list date顺序数据结构

objects : numbers, strings, functions, modules ----everything is an object, and any object can be assigned to a variable in python.

1. 内嵌数据结构list() : Dynamic, mutable, array . 使用[]

like an array, the object it stores are ordered sequentially in slots. 表示odds = [对象，对象，对象,…], can have differing type, help(list)。

1. List 类似arrays， lists can hold data of any type.
2. in，not in检查成员对象
3. list.append(value) #追加对象 >>>list.append([]) #an empty list add to the end of the list
4. list.remove('k') #remove a value/objects
5. list.pop() #pop the last value/objects
6. list.extend([3,4]) #provide a list of objects to append to the existing list

>>>list.extend([]) #do nothing, no object

1. list.insert(0, 'dd')
2. copyVowel = found #deep copy
3. shadowVowel = found.copy() #shadow copy

#donot use the assignment operator to copy a list, use the "copy" method instead.

1. 列表扩展方括号；lists的start、stop、step；list

>>> saying = “don’t panic” >>>letters = list(saying)

>>>letters #[‘d’,’o’,’n’,’’’,’t’,’ ’,’p’,’a’,’n’,’I’,’c’]

>>>first = letters[0] >>>last = letters[-1]

1. []方括号语法，letters[start: stop: step]，用于列表分片。指定列表分片范围和步长

start default value of 0, step default value of 1, stop takes on the maximum value allowable for the list. All three values are optional when used. 与list相反的操作’’.join(…)，二者均不改变原始数据，无损分片。

>>>firstlist = letters[0:3] #[‘d’, ‘o’, ‘n’]

>>>firstletters = ‘’.join(letters[0:5]) #don’t >>>’’.join(letters[-5:]) #panic

>>>reversedletters=’’.join([::-1]) #reversed string: "cinap t'nod"

>>>evey3letters=’’.join([::2]) #reversed string: "dntpnc"

>>>letters.insert(2,letters.pop(4))

12. if the data you want to store has an identifiable structure, consider using something other than a list.

1. 内嵌数据结构dictionary : dict()，(1)存储key / value键值对，使用{key:value,key2:value2, …}

associative array, map, symbol table, or has

1. 内嵌数据结构 tuple() ：an immutable list。常量数据结构(value,)。t=(2,3,4)
2. 内嵌数据结构set()，不允许重复元素

{key, key2, …}，创建有效集，可从序列创建set，

>>> vowel=set(‘words’), >>> word=’hello’,

第三章 structured data，结构化数据

1. dictionary : (1)存储key / value键值对，使用{key：value, key2：value2, …}

(2)associative array, map, symbol table, or hash，不维护插入顺序.

(3)能理解[]方括号里面是key，且key不允许重复，用方括号进行值查找found[key]=value,

>>> for k in found: print(k,’was found’, found[k])

>>> for k, v in sorted(found.items()): print(k,’was found’, found[k]) #output is sorted in order.

(4)dictionary动态特性，键值必须被初始化，运行时避免keyErrors(未定义key)；3种方法

方法一：对成员用”in”检查，else初始化,

方法二：用“not in”代替”in”, 追加initialize，确保在使用前初始化；

方法三：setdefault，只会在not in做一遍初始化，已经初始化的自动忽略

1. set：不允许重复，{}curly brace，内部对象没有顺序，内部对象可改变

>>>vowels = set(‘aabooeei’) #{‘a’, ’b’, ’o’, ’e’, ‘i} word=’how are you’

集合操作：交，并，差分（顺序很重要），并且没有破坏性，无损操作

>>>u=vowel.union(set(word)) # u = {'w', 's', 'h', 'l', 'd', 'r', 'e', 'o'}是vowelSet和word的并集

>>>list\_u=sorted(list(u)) #u = ['d', 'e', 'h', 'l', 'o', 'r', 'w'] 变list

>>>d=vowel.difference(set(word)) #d={'w', 'r', 'd'}，差分集，(vowel)-(word)

>>>i=vowel.intersection(set(word)) #i={‘o’}，交集

🡺字符检索，set结构如下。版本从list、dict到set到复杂结构见第2、3章code：vowels2.py-> vowels7.py

>>> vowels=set(“aeiou”)

>>> found=vowels.intersection(set(word))

1. tuple： 常量数据结构，不可变，元素不可以插入/删除/更新，除非包含的list的元素

>>>d=('w', 'r', 'd') >>>t=(‘python’ , ) #对于单个元素，必须有逗号, 否则变成str

>>> type(t) #<class 'tuple'>

1. 组合内嵌数据结构，存储，输出，可视化，访问复杂数据结构中的数据

（1）Dict包含dict，list，set等基础4中数据结构都可以相互包含嵌套

>>>people={} >>>people[‘Ford’]={‘name’: ‘Ford’, ‘gender’: ‘male’ } #dict{dict{}}

（2）打印输出 >>>import pprint >>>pprint.pprint(people) #pprint会自动排序

（3）访问数据 >>>people[‘Ford’] >>>people[‘Ford’][‘gender’]

第四章 code reuse，Functions -> Modules -> standard library，函数的复用与包装

1. Function: def functionname(argus): """a documentation string """ return value

关键词: def和return，引号，描述和代码，3引号描述函数目的，help函数时显示。入参。

2. PEPs标准: <https://www.python.org/dev/peps/>.

函数返回一个值：False

>>> bool(0) >>> bool(0.0) >>> bool('') >>> bool([]) >>> bool({}) >>> bool(()) >>> bool(None)

>>>def search4vowels(word):

"""display any vowels found in an asked-for word."""

vowels=set('aeiou')

return vowels.intersection(set(word))

2. Annotations, 使用注解来改进你的文档：see PEP3107，<https://www.python.org/dev/peps/pep-3107/>.

（1）函数annotations可选的。（2）function annotation are informational。annotations帮助文档化函数，可以使用help BIF来查看到内容。

3. 建立通用的函数

(1) a more generic function name. (2) add a second argument.

(3) remove the vowels variable. (4) update the docstring. (5) a default value

>>>def search4vowels(phrase:str) -> set:

"""Return any vowels found in a supplied phrase."""

vowels = set('aeiou')

return vowels.intersection(set(phrase))

>>>def search4letters(phrase:str, letters:str=’aeiou’) -> set:

"""Return a set of the 'letters' found in 'phrase'."""

return set(letters).intersection(set(phrase))

函数调用时，两种匹配方式：位置和关键字。因此参数顺序并不重要，可以使用关键字参数

>>> search4letters('how','are you')

>>> search4letters(letters='how', phrease='are you')

4. 函数打包成模块

（1）上面两个函数保存成文件vsearch.py，make it a fully formed module。

（2）identify the module to import。要求：1.必须在当前工作目录. 2. Your interpreter’s site-packages locations. 3. The standard library locations

（3）命令行运行python，>>>import vsearch >>> vsearch.search4letters('how are you')

5. 模块放入站点包中

使用setuptools 安装到站点包中：(1)create a distribution description. (2)generate a distribution file. (3)install the distribution file. Python3.4之后的版本才使用setuptools a breeze

Step1：创建setup文件，setup.py，内容如下：

from setuptools import setup

#this is an invocation of the ‘setup’ function. We’re spreading its arguments over many lines.

setup(

name='vsearch', #indentified the distribution. It’s common practice to name the distribution after the module.

version='1.0',

description='The Head First Python Search Tools',

author='ying xin',

author\_email='395488553@qq.com',

url='qq.com',

py\_modules=['vsearch'], # py\_modules=['vsearch']: a list of ‘py’ files to include in the package. we only have one ’vsearch’. )

Step2: setup.py和vsearch.py在同一个目录下，命令行进入这个目录。如下：

D:\python\code> py -3 setup.py sdist

Run python -3，执行setup.py，pass ‘sdist’ as an argument。生成文件vsearch-1.0.tar.gz。

如果运行结果有不对的，检查python版本，和setup.py文件是否正确。

6. 使用’pip’安装包

D:\python\code\dist> py -3 –m pip install vsearch-1.0.tar.gz

任何python代码都可使用pip 来安装你的模块

分享代码：<https://pypi.python.org/pypi> <https://www.pypa.io>

7. 函数参数传值？arg = arg\*2， 传引用？arg.append(‘nore data’)

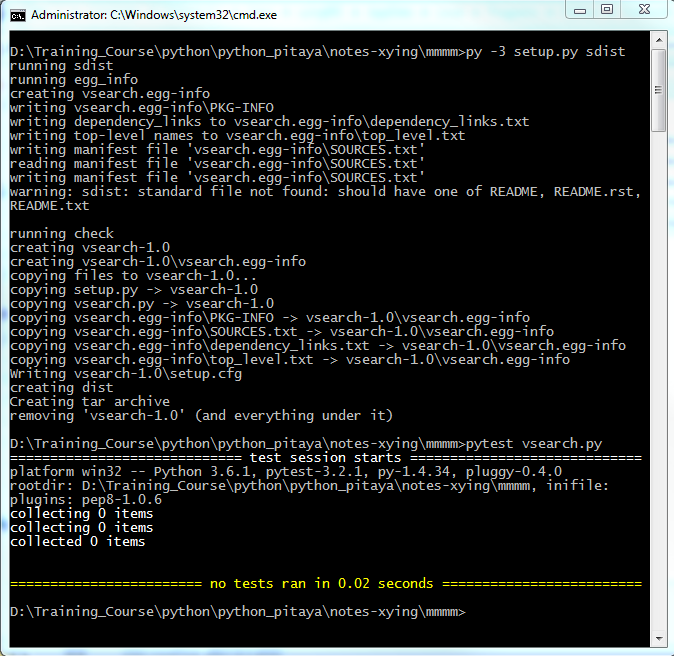
8. 检查PEP 8服从性， more ref: <https://doc.pytest.org/en/latest/>.

Step1: 管理员模式下，安装测试开发工具: $ py -3 –m pip install pytest

Step2: 管理员模式下，安装测试开发工具: $ py -3 –m pip install pytest-pep8

注：冒号后有空格，->前后都有空格，代码最后有一个空行

备注：非Windows下，”py -3”替换成”sudo python3”



(step1：生成dist文件夹（文件夹有vsearch压缩包，供后面install用），和vsearch.egg-info。Step2：生成\_\_pycache\_\_文件夹，里面有pytest结果)

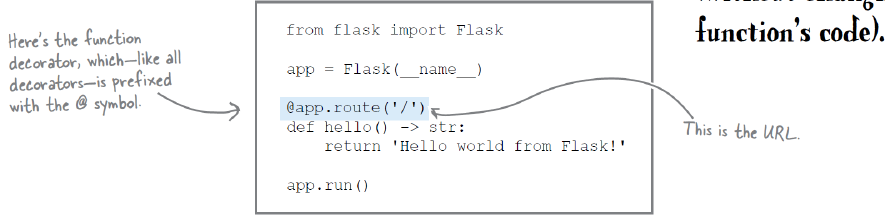
第五章 building a webapp

1. Install python
2. Check pip have installed

Python的安装路径C:\Users\xying001\AppData\Local\Programs\Python\Python36-32

1. Install flask

D:\Training\_Course\python-> py -3 –m pip install flask



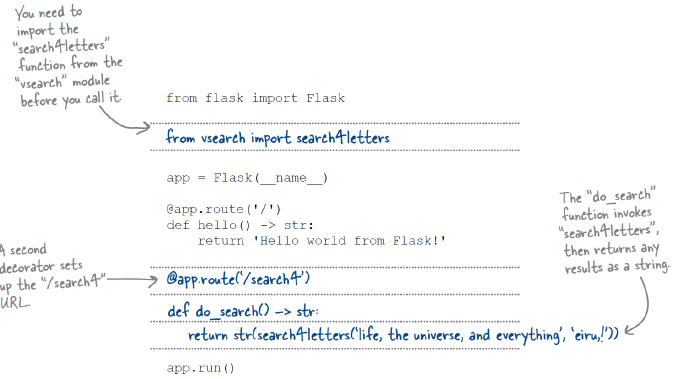
Create an instance of a Flask object and assign it to “app”.

前端：Javascript，Angularjs，Python Flask

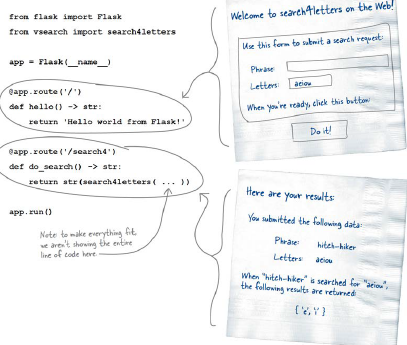
后端：nodejs，Flask python

框架： vs，nodejs，flask

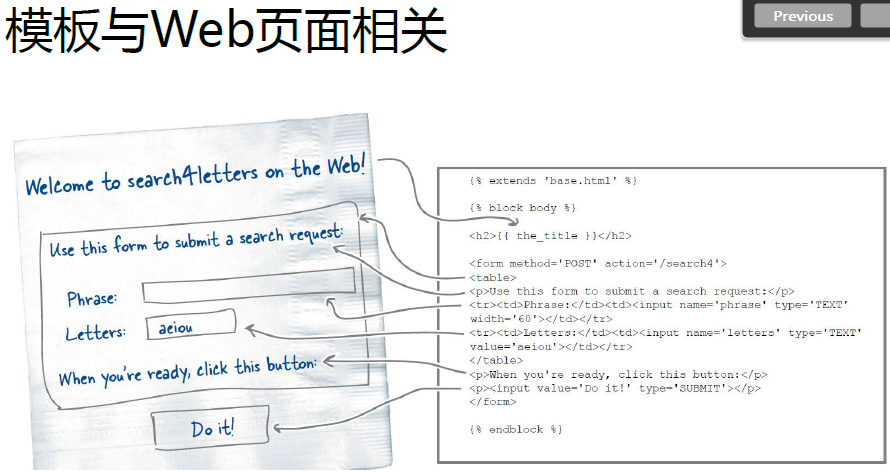
导入module，然后route到已有的。因此下面的，第一个route收集用户的输入，第二个route返回用户的结果。As：小黄车



任何一个服务都有一个前端和一个后端，下图中左侧是后端，右侧是前端



1，构建HTML表单，构造前端



有一个input，value是do it，是一个submit类型，点完do it，会发出一个submit，触发post，将会跳转到/sesrch4页面。基本html模板信息如下：(1). base.html页面

<!doctype html> 标准html5 markup。

{{the\_title}} 模板参数，用2个大括号括起来的，都是可以在后端实现的，动态的，必须自己提供的参数。

{%block body%} {% endblock %}一组格式是指一个HTML block，在展示前被替换，可以被任何子类页面重写。

(2). entry.html 页面，继承base.html页面

{% extends ‘base.html’ %} 是指html模板继承于base.html。后面是当前html的详细实现部分

2. 从Flask渲染模板，in python

（1）import render\_template 函数。（2）创建新URL，/entry （3）创建函数返回正确的渲染HTML

@app.route(‘/entry’)

def entry\_page() -> ‘html’: 返回的是html

return render\_template(‘entry.html’, the title=’welcome to search letters on the web!’)

提供render模板，提供html页面中相关参数的值

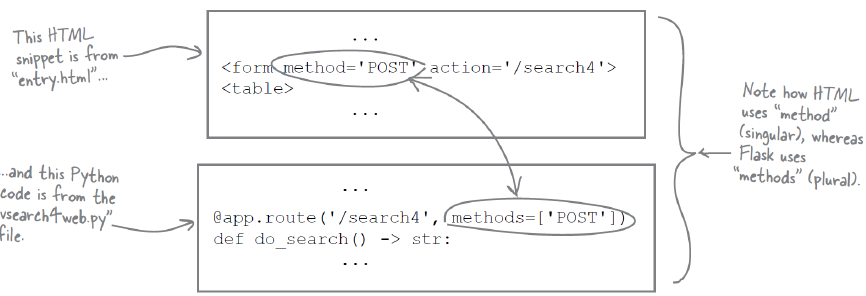
注意：所有的html必须放在templates文件夹下面，由于render\_template搜索决定的

3， Python中动作： get, post, delete

@app.route默认的行为是get，如果html模板需要post动作，就要修改method=”post”

即@app.route('/search4', methods=['POST'])

如下图，上面一个是html5页面代码，下面是python代码



4，使用flask访问HTML表单数据，搜索页面，in python

entry.html页面中的<input name='phrase'…>，对应的就是search4对应的数据，而目前仅仅是返回str硬编码。

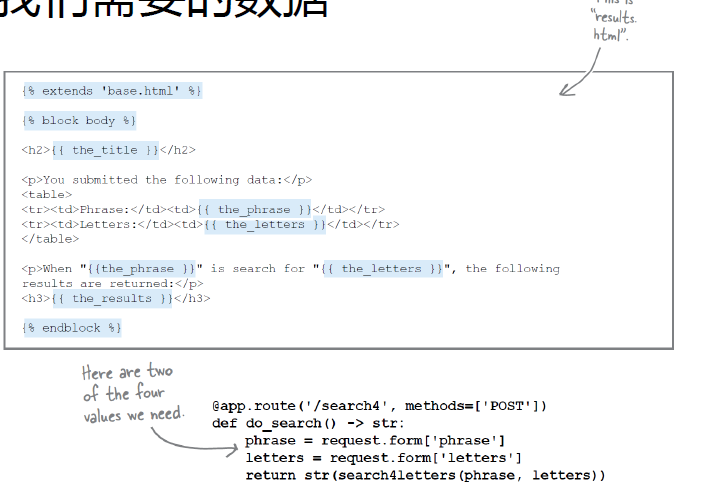
5, 从flask使用request请求数据，优化硬编码

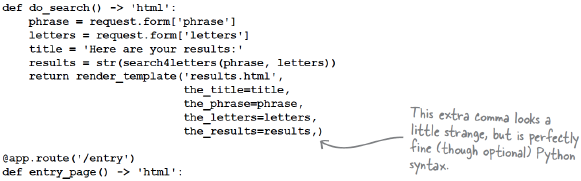
phrase = request.form['phrase'] letters = request.form['letters']

return str(search4letters(phrase, letters))

6，将结果生成为HTML

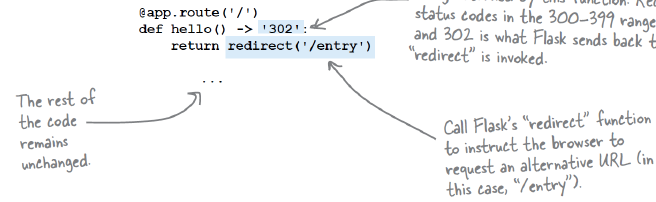
result.html页面，其中阴影部分都是需要python实现的。如下：python扔回的的界面search4，和实现的phrase…





注意：最后的the\_results=results,的最后有一个逗号，加逗号表示这是个元组，不加逗号表示的是元素，仅是个类型元素

出错的时候都是返回到entry里面



作用域：如果之前的作用域是main，才可以起这个服务。

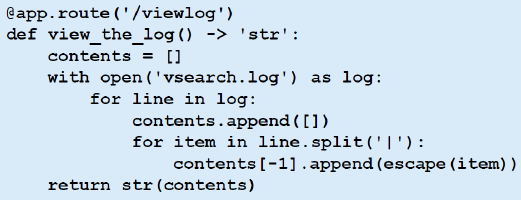
if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

第六章 数据处理

数据处理方法：’’.join, list.split(‘’), list.append([]), escape。无损数据处理。

Html5编程+python



解释：@一个路由到某个h5页面；def函数，返回值str；定一个空list；打开log文件，仅仅打开，仅是作为一个对象；for line查看是否有内容；在备用list里添加一个空list；查看并删除当前行内容中的|；将数据追加到备用list的尾部的刚追加的空list中。

查看指定目录下的文件

>>> import os

>>> os.listdir()

替换字符并重写文件

import os

os.chdir('D:\Training\_Course\javascript\RFID\RFID-node.js-master\python\_analysis\_homecss')

lines = open('home.txt').readlines()

fp = open('home2.txt', 'w')

for s in lines:

fp.write(s.replace('}', '\*\*')

fp.close()

字符串换行：

if \_\_name\_\_ == '\_\_main\_\_':

#第一种：三个单引号

    print ''' 我是一个程序员

        我刚开始学习[Python](http://lib.csdn.net/base/python)'''

#第二种：三个双引号

    print """ 我是一个程序员

        我刚开始学习[python](http://lib.csdn.net/base/python)"""

#第三种：\结尾

    print "我是一个程序员，\

        我刚开始学python"

第7章 webapp --windows

1．mysql下载mysql-5.7.19-winx64.zip

<https://dev.mysql.com/downloads/mysql/>

（官网->downloads->my community->my community server->mysql-5.7.19-winx64.zip）

2. 配置mysql

（1）解压缩，在bin的目录下创建一个my.ini配置文件，并添加内容utf8

（2）添加系统环境变量，D:\mysql-5.7.19-winx64\bin;

（3）管理员权限开启cmd

（4）mysqld --initialize-insecure --user=mysql 【第一次安装】

mysqld install MySQL --default-file=”D:\mysql-5.7.19-winx64\bin\my.ini” 【第二次安装】

mysqld remove

（5）net start mysql

或者在search program and files 搜索框里面，输入services.cvs，查找并点击mysql，进入点击start，手动启动服务，

（6）net stop mysql

3. 创建数据库，并将数据库grant给特定的用户，创建数据表。

> create database vsearchlogDB;

> grant all on vsearchlogDB.\* to ‘vsearch’ identified by ‘vsearchpasswd’;

>quit

> mysql –u vsearch –p vsearchlogDB

> use vsearchlogDB;

mysql> create table log (…);

4. 为python 按时装mysql驱动程序，一共有3种方法

方法一：下载mysql-connector的zip文件，其中2.1.6版本对应python3.4。

解压缩mysql-connector，并安装，按shift+鼠标右键，当前目录下打开cmd >py -3 setup.py install

如果是linux或者mac，$ sudo –H python3 setup.py install

方法二：下载mysql-connector的msi安装包，这个安装包针对Windows系统的。

方法三：直接在线用pip安装， cmd下> pip install mysql-connector #自动安装最新的默认的驱动

本地操作：python交互界面操作

Step1：配置dbconfig

Step2：import mysql.connector，连接数据库

Step3：打开游标

Step4：写sql语句，执行sql语句。（如果insert数据，还需要执行游标commit()）

Step5：关闭游标，关闭数据库连接。

云上操作

[https://www.pythonanywhere.com/user/yingxin/consoles/#](https://www.pythonanywhere.com/user/yingxin/consoles/) 注册用户名密码。

网页上有5个窗口: Consoles, Files, Web, Schedule, Databases

1. Files窗口，上传vsearch-1.0.tar.gz

注意：网页上consoles窗口，点bash进入linux的shell；点3.6进入python idle，即python shell；点mysql进入mysql的shell界面

1. 安装上传的依赖包 ：consoles窗口-> bash, $ python –version #获取版本号，后面要使用

$ python3 #进入py3 >>>python3 -m pip install vsearch-1.0.tar.gz --user

1. MySQL操作创建数据库。

方法一：网页上databases窗口，设置数据库密码“vsearchpasswd”，数据库名键入：vsearchlogDB，点击Create。然后会自动跳转到mysql的命令行界面。可以创建table了。

方法二：右键命令行，consoles窗口-> bash。推荐方法一。

>show databases; >use databasename; >describe log; >select \* from log;

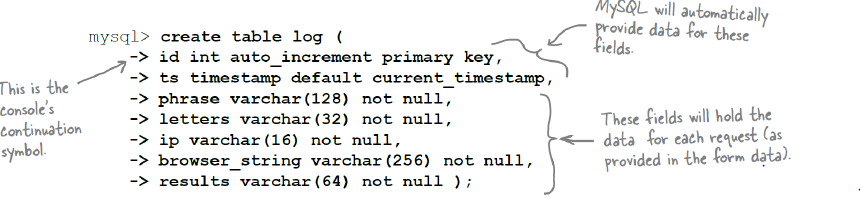
1. Create table。

mysql> show databases;

mysql> use yingxin$vsearchlogDB;

mysql> show tables;

mysql> create table log (…); detail as below



mysql> show tables;

mysql> describe log;

1. 云上打通mysql和python

Consoles窗口：点bash。>>>python3

>>>dbconfig = {‘host’: ‘yingxin.mysql.pythonanywhere-services.com’,

‘user’: ‘yingxin,

‘password’: ‘vsearchpasswd’,

'database': 'yingxin$vsearchlogDB', } //yingxin是注册用户名, password是数据库时设置的mysql的密码

>>> import mysql.connector //如果云上没有，就可以pip install mysql-connector

>>>conn = mysql.connector.connect(\*\*dbconfig) //两个\*代表dict，connect上连接

>>>cursor = conn.cursor() //打开游标

>>> \_\_SQL = “””show tables””” //sql语句，是3引号

>>>cursor.execute(\_\_SQL) //执行sql语句

>>>res = cursor.fetchall() //fetch结果

>>>res //查看fetch到的sql=show tables的内容

>>> \_\_SQL = “””describe log””” //sql语句里面是3引号

>>>cursor.execute(\_\_SQL)

>>>res = cursor.fetchall()

>>>import pprint

>>>pprint.pprint(res) //换一种输出方式

1. 上传本地webapp代码到网上。架自己的网站。

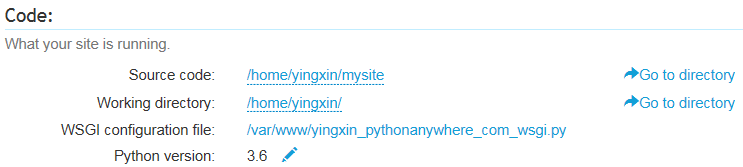
主页是第七章的vsearch4web.py修改dbconfig 如上，第九章的DBcm.py挪到第七章 webapp里面（因为这个里面有import DBcm）。把整个webapp打包成webapp\_clond.zip，上传到网页的Files窗口。

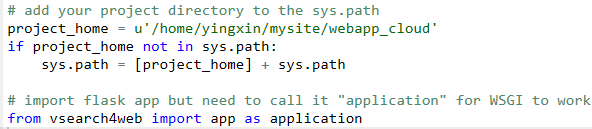
1. 进入网页consoles的Python的bash。退出python shell到linux shell界面$ ls –s查看文件是否都在，vsearch-1.0.zip和webapp\_clond.zip。
2. 解压缩并move到mysite里，并验证vsearch4web.py是否已经修改好了dbconfig

$unzip webapp\_clond.zip

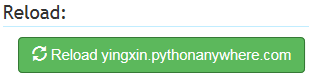
$ mv webapp\_clond mysite

1. 进入网页的web窗口，点‘add a new web app’,next, 选Flask，next，选python3.6 flask0.11.1（这里就是刚才查看的python的版本号），next
2. 网页输入<http://yingxin.pythonanywhere.com> 查看默认网站。
3. 在web窗口，找到运维文件WSGI，点开，把替换自己的网站





注意project\_home路径，保存并reload。或者在web窗口，点击最上面的reload重新加载网站。



Mysql 数据库基本操作

一、清除mysql表中数据

delete from 表名; truncate table 表名;

不带where参数的delete语句可删除mysql表中所有内容，用truncate table也可以清空mysql表中所有内容。

效率上truncate比delete快，但truncate删除后不记录mysql日志，不可以恢复数据。

delete的效果有点像将mysql表中所有记录一条一条删除到删完，而truncate相当于保留mysql表的结构，重新创建了这个表，所有的状态都相当于新表。

二、删除表中的某些数据

delete from命令格式：delete from 表名 where 表达式

例如，删除表 MyClass中编号为1 的记录：

mysql> delete from MyClass where id=1;

三、修改表

1.选择数据库：>use 数据库名;

2.查询所有数据表：>show tables;

3.查询表的字段信息：>desc 表名称;

3.1.修改某个表的字段类型及指定为空或非空

>ALTER TABLE 表名称 change 字段名称 字段名称 字段类型 [是否允许非空];

> ALTER TABLE 表名称 modify 字段名称 字段类型 [是否允许非空];

>ALTER TABLE 表名称 drop 字段名称;

mysql>ALTER TABLE person CHANGE number id BIGINT; //number名字修改为id，类型为BIGINT

mysql>ALTER TABLE person MODIFY number BIGINT NOT NULL; //number类型修改为BIGINT

3.2.修改某个表的字段名称及指定为空或非空

>ALTER TABLE 表名称 change 字段原名称 字段新名称 字段类型 [是否允许非空];

例如: 修改表expert\_info中的字段birth,允许其为空

mysql> ALTER TABLE expert\_info change birth birth varchar(20) null;

1.增加、删除一个字段(一列)

>ALTER TABLE <table\_name> add column <column\_name> type default <value>;

type指该字段的类型,value该字段的默认值

>ALTER TABLE form1 drop column 列名;

mysql>ALTER TABLE mybook add column publish\_house varchar(10) default ”;

mysql>ALTER TABLE person DROP COLUMN birthday;

2.更改一个字段名字(也可以改变类型和默认值)

>ALTER TABLE <table\_name> change <sorce\_col\_name> <dest\_col\_name> <type> default <value>;

source\_col\_name指原来的字段名称,dest\_col\_name指改后的字段名称

mysql>ALTER TABLE Board\_Info change IsMobile IsTelphone int(3) unsigned default 1;

3.改变一个字段的默认值

>ALTER TABLE <table\_name> alter <column\_name> set default <value>;

mysql>ALTER TABLE book alter flag set default '0′;

4.改变一个字段的数据类型

>ALTER TABLE <table\_name> change column <column\_name> <column\_name> <type>;

Mysql>ALTER TABLE userinfo change column username username varchar(20);

5.向一个表中增加一个列做为主键，或删除主键

ALTER TABLE <table\_name> add column <column\_name> <type> <auto\_increment> PRIMARY KEY;

mysql>ALTER TABLE book add column id int(10) auto\_increment PRIMARY KEY;

mysql>ALTER TABLE person ADD PRIMARY KEY (id);

mysql>ALTER TABLE person DROP PRIMARY KEY (id);

6. 添加、删除普通索引；

>ALTER TABLE person ADD INDEX birthday\_index(‘birthday’);

>ALTER TABLE person drop index birthday\_index;

7. 激活、禁用非唯一索引；

>ALTER TABLE person enable keys;

>ALTER TABLE person disable keys;

8. 数据库某表的备份,在命令行中输入:

mysqldump -u root -p <database\_name> <table\_name> > <bak\_file\_name>

mysql>mysqldump -u root -p f\_info user\_info > user\_info.dat

9. 导出数据

select\_statment into outfile“dest\_file”;

mysql>select cooperatecode,createtime from publish limit 10 into outfile”/home/mzc/temp/tempbad.txt”;

10. 导入数据

load data infile“file\_name” into table <table\_name>;

mysql>load data infile“/home/mzc/temp/tempbad.txt” into table pad;

11. 将两个表里的数据拼接后插入到另一个表里。下面的例子说明将t1表中的com2和t2表中的com1字段的值拼接后插入到tx表对应的字段里。

mysql>insert into tx select t1.com1,concat(t1.com2,t2.com1) from t1,t2;

12. 修改表默认字符集和所有字符列改为新的字符集；修改表某一列的编码；仅改变表的默认字符集。

>ALTER TABLE person CONVERT TO CHARACTER SET utf8;

>ALTER TABLE person CHANGE name name varchar(255) CHARACTER SET utf8;

>ALTER TABLE person DEFAULT CHARACTER SET utf8;

13. 修改表名

>RENAME TABLE person TO person\_other;

14. 移动表到其他数据库

>RENAME TABLE current\_db.tbl\_name TO other\_db.tbl\_name;

第八章 A little bit of class

类似于c++中的this，python使用的是self，其中带有\_\_是系统默认的函数，不带\_\_的是自定义的function。

class CountFromBy:

def \_\_init\_\_(self, v: int=0, i: int=1) -> None:

self.val = v

self.incr = i

def increase(self) -> None:

self.val += self.incr

def \_\_repr\_\_(self) -> str:

return str(self.val)

d = CountFromBy()

d.increase() #看起来没有参数，实际上increase的原型应该是有self的那个. When writing code in a class, think of “self” as an alias to the current object.

class CountFromBy:

pass #其中pass是类占位符，满足编译通过，便于后面实现.

作用域：def中的形参的生存期只是def内部，作用域内部用self约束，class中的self的作用域就是在这个class定义中；class中的def，如果没有self约束将会报nameError，因为调用def的方法时，就是出了def increase函数，参数就无效了。

带默认值的构造def \_\_init\_\_(self):，做基本初始化，并且参数提供合理的默认值（方便调用时必须给初值），参数赋值，self.val = v，self.incr = i

查看地址：id(d), hex(id(d))，可确认作用域

常用：help(), dir(), id()

复用webapp代码

conn=mysql.connector.connect(\*\*dbconfig) ##其中dbconfig是dict类型，两个\*，如果是list类型是一个\*

teardown code，必须按照这个顺序

conn.commit() #提交为保存的数据

cursor.close() #关闭数据库游标

conn.close() #关闭连接

第九章 上下文管理协议

如何在运行时进行配置python的with声明

1. 分析复用webapp代码，最好的方法？在class第六章基础上继续第七章
2. 管理上下文，打造一个通用的类，自定义的类必须有如下3个方法。类定义这3个方法，就是管理上下文

\_\_enter\_\_ （perform setup）， \_\_exit\_\_ （does teardown）， \_\_init\_\_ （perform initialize）

构造析构的任务是

1. import mysql.connector #Using camelcase when naming a class in python

class UseDatabase:

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

pass

def \_\_enter\_\_(self):

pass

def \_\_exit\_\_(self):

pass

1. 作业 ppt466， base on webapp\_cloud，重新上传到网上

(1)select count(\*) from log;

(2) select count(letters) as ’count’, letters from log group by letters order by count desc limit 1;

(3)select distinct ip from log;

(4) select browser\_string, count(browser\_string) as ’count’ from log group by browser\_string, order by count desc limit 1;

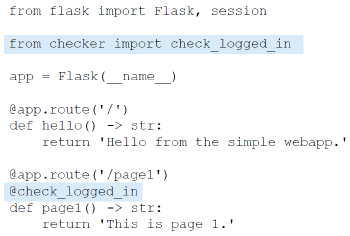
第十章 function decorators

装饰器模式@，session

注意：尽量用cmd打开，然后运行python，进入的是main域。而使用python idle并不是在main域执行python.

Session，参考第10章code的quick\_session.py。改进版本simple\_webapp2.py…，最后使用包装器的checker，装饰器@@check\_logged\_in，得到simple\_webapp4.py

装饰器，wraps 包装器，详细实例参看第10章code的checker.py，就是一个标准的模板。把check\_logged\_in包装成一个包装器wrapper，最后以装饰器@注入依赖信息到自己的代码。最后实际上返回的就是一个函数func。第一步第二步是让装饰器起作用。



装饰器语法：以@的方式注入依赖。引入python函数模板。@xxx就是使用的装饰模式。

满足如下4个条件才能满足装饰器，也就是创建函数装饰器的方法（ppt502）：1. 创建一个函数；2. 把函数当参数传给别的函数；3. 函数返回值是一个函数；4. 能处理任何数量和类型的函数参数，或者装饰器包含一个装饰函数的签名

对于C++11函数指针。。。

1. 函数指针 std::function<T> typedef (void\*)(\*func)(void\*) using func=(void\*)(\*)(void\*)
2. 函数对象class func{ operator()(){…}}
3. lamda表达式[](int i){return i>=100;}，
4. 函数回调方法2个，装饰器是面向过程的包装，包装函数有如下两个方法

方法一：函数作为其他函数的参数，python参看第10章code的checker.py，or>>>apply(type, apply)

方法二：函数可以嵌套在函数内，python参看ppt506，inner(),

同样，python 函数模板，面向过程的封装，机械的封装。

例子1. 包装器，函数是参数和/或返回值

def apply(func: object, value: object) -> object:

return func(value)

apply是一个包装器，可以做任何事情，他包装了func，而func只能做一件事情。

做到了：第一，将函数作为参数，第二点，返回函数作为返回值

>>> apply(id, 'a')

>>> apply(type, 'rf')

例子2：函数可以嵌套函数见ppt506，调用，inner()

例子3：从函数返回一个函数见ppt507，return inner，返回函数对象给调用者

例子4：可变参模板：参看第10章code的myfunc.py，idle运行后

>>> myfunc(10) #10

>>> myfunc()

>>> myfunc(10,20,30) #10 20 30

>>> value=[1,2,3,4]

>>> myfunc(value) #[1, 2, 3, 4] 直接把value整体当参数

>>> myfunc(\*value) #1 2 3 4 依次萃取value中每个值，依次执行，这又叫产生式

说明：

myfunc(\*args): 一个\*， 代表可以接收任意多个参数

myfunc2(\*\*kwargs) #两个\*\*，二维，接收参数字典，一个表示key，一个代表value

myfunc3(\*args, \*\*kwargs): 既可以接收一维，也可以接收二维，或者可以同时接收一维二维。参数第一个是list，第二个是dict。推荐尽可能写myfunc3.

创建函数装饰器：1. 一个函数，参数是函数，返回值是一个函数，python参看第10章code的checker.py，典型装饰器模板

让装饰器起作用：import一个装饰器，删除不必要的代码，使用@装饰器 as require，参看simple\_webapp4.py

第十一章 第一部分exception handing

4个措施对待错误：防错，查错，改错，容错

恢复方式：向前恢复和向后恢复

继承，为了复用

多态，用了一个名字多重能力，静态多态，动态多态

父类对象的引用、指针指向子类的对象，调用的是子类的方法。

针对我们的webapp\_cloud，Python错误检测和容易出错的地方

1. dbconfig写错，数据库连接失败，多数是用户名密码数据库名字..写错了
2. Sql语句写错
3. 执行sql语句不成功，cursor.execute(\_SQL, …)执行时间长
4. log\_require调用失败

可能发现的问题：

1. 数据库临时不可用，数据库连接失败
2. 输入端输入攻击，恶意代码插入
3. 输出缓慢，代码执行太长时间
4. 调用函数失败，读写日志失败

解决这些异常错误的办法：增加健壮性

1. What happens if the database connection fails?
2. Is our webapp protected from web attacks?
3. What happens if something take a long time?
4. What happens if a function call fails?
5. 打开文件不确定是否失败？带有try的with结构，针对with失败，参考第11章的code的try\_examples2.py

try：with open(‘file.txt’) as fh:… except FileNotFoundError: …

1. 打开权限不够问题？函数调用失败？异常处理？参考第11章的code的try\_examples3~5.py

类似c++中的万能异常处理方式catch(…)cout<<”other excep”; 有exception类位于STL里。

Python里也有很多exception，各个语言都有自己的exception类。如果exception不够，python万能的exception是except: print(‘some other error occurred. ’ )

\*\* Exception类型：reset、undefined instruction、software interrupt、data abort、prefetch abort、reserved，interrupt、fast interrupt共1~64位。如下2种方式就可以包含所有的异常。

>>>except Exception as err: print(str(err))

>>>except: …

1. sys的异常处理方式，在with闭包内就能处理error。抛出异常

refer: <https://docs.python.org/3/library/sys.html>.

针对webapp\_cloud，log\_request失败，try：… except Exception as err: print(str(err))

下面4/5针对webapp\_cloud，DBcm.py的异常处理，自定义异常

1. 主动raise exception，类似c++ throw…
2. 创建自定义异常，子类继承自Exception，参考ppt591

>>>class ConnectionError(Exception): pass #定义异常类，继承自Exception

>>> try: ……

[except mysql.connector.errors.InterfaceError as err:]

raise ConnectionError("hhhhhhhhh!" 或err) #合适地方，合适except，raise出异常异常内容

>>>try… #合适地方，使用自定义异常

except ConnectionError as err: print('got :', str(err))

1. 数据库连接失败问题，DBcm可能出的问题的处理，例子参考第11张webapp的code的new\_DBcm.py

（1）连接数据库失败，没有连上mysql，数据库有问题，没有start等

import mysql.connector

class ConnectionError(Exception):

pass

class UseDatabase:

def \_\_init\_\_(self, config: dict):

"""Add the database configuration parameters to the object.

This class expects a single dictionary argument which needs to assign

the appropriate values to (at least) the following keys:

host - the IP address of the host running MySQL/MariaDB.

user - the MySQL/MariaDB username to use.

password - the user's password.

database - the name of the database to use.

For more options, refer to the mysql-connector-python documentation. """

self.configuration = config

def \_\_enter\_\_(self) -> 'cursor':

"""Connect to database and create a DB cursor.

Return the database cursor to the context manager. """

try:

self.conn = mysql.connector.connect(\*\*self.configuration)

self.cursor = self.conn.cursor()

return self.cursor

except mysql.connector.errors.InterfaceError as err:

raise ConnectionError(err)

def \_\_exit\_\_(self, exc\_type, exc\_value, exc\_traceback): #”exe\_info” should be clear what these method arguments refer to: exception data.

"""Destroy the cursor as well as the connection (after committing). """

self.conn.commit()

self.cursor.close()

self.conn.close()

（2）认证失败，就是mysql的conn用的dbconfig的用户名/密码/数据库名等问题

except mysql.connector.errors.ProgrammingError as err:

raise CredentialsError(err)

（3）SQL语句有问题。语句执行失败。这个就需要在游标关闭数据库连接关闭后，抛出异常。

if exc\_type is mysql.connector.errors.ProgrammingError:

raise SQLError(exc\_value)

elif exc\_type:

raise exc\_type(exc\_value)

1. 运行慢的问题，并行处理，多线程处理

第十一章 第二部分Threading线程处理

Cpu密集型工作，主线程

IO密集型工作，子线程

Refer: <https://docs.python.org/3/library/concurrency.html>

有两个模块用于支持在Python 3中使用线程:

\_thread

threading

thread模块已被“不推荐”了很长一段时间。 鼓励用户使用threading模块。 因此，在Python 3中，thread模块不再可用。 但是，thread模块已被重命名为“\_thread”，用于Python 3中的向后兼容性。

要使用threading模块实现新线程，必须执行以下操作：

（1）定义Thread类的新子类。

（2）覆盖\_\_init \_\_(self [，args])方法添加其他参数。

（3）然后，重写run(self [，args])方法来实现线程在启动时应该执行的操作

在webapp例子里面，读写日志的读写日志库是最慢的，可以放子线程

问题1：游标写日志数据库，cursor.execute(…)。问题2：游标读日志库

分析：写数据库不能影响response用户；读数据库是必备的路径，才能显示数据。

改在step1，结果：需要写数据库线程化。使用线程思路 >>>from threading import Thread

>>>t= Thread(…) >>>t.start() DBcm改造见ppt618和后面的小测试和答案，主要看try…except部分的代码的改造

from threading import Thread

try:

t = Thread(target=log\_request, args=(request, results)) #回调入口地址，入口函数的参数列表

t.start() #函数相关的t线程被threading模块 调度执行。调用start()方法来调用run()启动新线程。

except Exception as err:

print('\*\*\*\*\* Logging failed with this error:', str(err))

改造step2：copy当前主线程context给子线程，才可以sleep(15)，同步子线程和主线程的作用域，否则将崩溃，参考第11章code的vsearch4web.not.slow.with.threads.fixed.py

def do\_search() -> 'html':

@copy\_current\_request\_context #copy当前主线程context给子线程

def log\_request(req: 'flask\_request', res: str) -> None:

改造结果：第一：“log\_request”函数定义在“do\_search”函数内；第二：装饰器应用到“log\_request”，copy主线程上下文到子线程

第十二章 Advanced iteration

参考第12章code，内容：总结之前内容，并优化和迭代器，高级迭代，推导式和产生式

类似c++中的lamda表达式

1. 处理CSV文件处理，转换成python文件处理。

（1）数据的加载，Python里面有CVS module，因此 import cvs，open file

读出来: 一堆字符串，一堆list，一堆dict都是不好的。最好应该是读出来只有一个list

（2）这就是数据的清洗，和整理。最后剪出来一个dict，见ppt638

line.split, escapt, strip。注意：链式的方法调用顺序，先strip后split

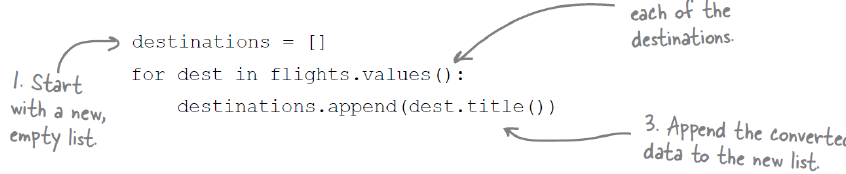
（3）数据的链式和转化，数据变换，转换成字典

数据格式:https://docs.python.org/3/library/datetime.html#strftime-and-strptime-behavior

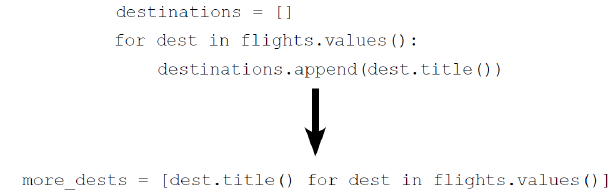
1. 推导式 ppt650 使用[] 把所有事情都做完了才产生，统一做，list的推导式，效率高，替代一个多个for

For k，v in flights.items():

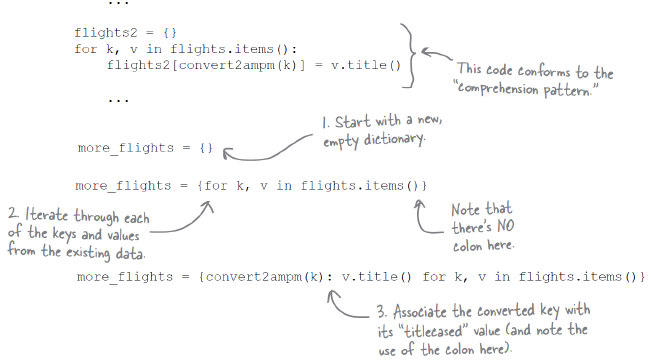
如下是标准推导式



1. List的推导式🡺，使用[]，提高了效率，没有那么多临时变量。因为for的迭代器每一次都会new出来一个，就会导致不停的new、delete。但是推导式的方式是一次申请到一定量的内存，用完释放。使用的是[]



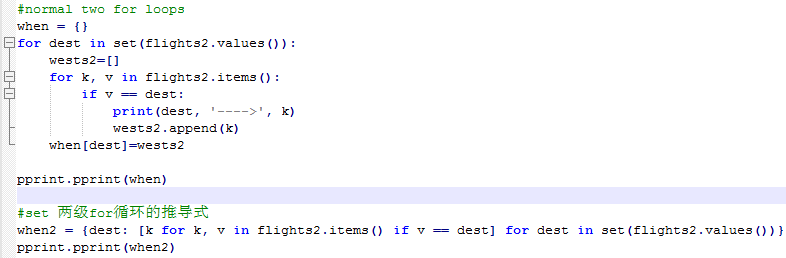
2. 一个字典“推导式”ppt659 使用的是{}



3. set的推导式 ppt662

Set最大特点：没有重复，没有冒号。使用{}，set(flight2.values())

转换成set后，再转换成一个list最后提取出的见ppt677. 推导式可以嵌套，双层for



如上list, dict, set的整个推导式推导过程见第12章xying\_csvprocess.py

4. “元组” (tuple)推导式 ppt681. It is immutable成员不可改变

1. Generator产生式，使用listcomp处理url ppt684，使用()，每一次联通一次产生一个，逐个做

需要插件$ python - m pip install requests

安装结果在C:\Users\xying001\AppData\Local\Programs\Python\Python36-32\Lib\site-packages

用产生式做的实例见ppt689

for i in [x\*3 for x in [1,2,3,4,5]]: print(i) # a listcomp,

for i in (x\*3 for x in [1,2,3,4,5]): print(i) #a generator can be used anywhere a listcomp is used, and produces the same results.

#define function

def gen\_from\_urls(urls: tuple) -> tuple:

for resp in (requests.get(url) for url in urls):

yield len(resp.content), resp.status\_code, resp.url

#return len(resp.content), resp.status\_code, resp.url

#use yield instead return. the "yield" return each line of results from the GEN response to the waiting for loop

#remember: donot use "return"

第3章：复杂的数据结构

第4章：函数的复用和包装

第5章Web

数据库

类

With

函数和修饰符

爬虫实践

1．urllib库的使用

#encoding:UTF-8

import urllib.request

data= urllib.request.urlopen("http://www.baidu.com").read()

data = data.decode('UTF-8')

print(data)

2．认证身份，设置Headers

有些网站不会同意程序直接用上面的方式进行访问，如果识别有问题，那么站点根本不会响应，所以为了完全模拟浏览器的工作，我们需要设置一些Headers 的属性。这个头中包含了许许多多是信息，有文件编码啦，压缩方式啦，请求的agent啦