目录

[一、平台体系 5](#_Toc58186615)

[python基本概念 5](#_Toc58186616)

[Windows下如何运行python 6](#_Toc58186617)

[vs code如何运行python 6](#_Toc58186618)

[vs code常用插件 8](#_Toc58186619)

[Windows下设置环境变量 8](#_Toc58186620)

[Windows下查看版本号 8](#_Toc58186621)

[Windows下如何安装pip 9](#_Toc58186622)

[Windows下如何某个包是否安装 9](#_Toc58186623)

[模块导入 12](#_Toc58186624)

[模块重名 13](#_Toc58186625)

[from xx import方式 14](#_Toc58186626)

[import numpy 和 from numpy import \* 的区别 14](#_Toc58186627)

[语法糖 15](#_Toc58186628)

[打包成独立执行的exe可执行文件 15](#_Toc58186629)

[二、基本语法 17](#_Toc58186630)

[和.Net等的区别 17](#_Toc58186631)

[{} 17](#_Toc58186632)

[变量 18](#_Toc58186633)

[标准数据类型 20](#_Toc58186634)

[标准数据类型 20](#_Toc58186635)

[type-获取数据类型 20](#_Toc58186636)

[id-获取变量内存地址 20](#_Toc58186637)

[数字 21](#_Toc58186638)

[字符串 21](#_Toc58186639)

[索引index 21](#_Toc58186640)

[切片 22](#_Toc58186641)

[split 22](#_Toc58186642)

[join 22](#_Toc58186643)

[列表 22](#_Toc58186644)

[可以通过max函数求最大值 24](#_Toc58186645)

[排序 24](#_Toc58186646)

[移除元素 24](#_Toc58186647)

[添加元素 25](#_Toc58186648)

[Round()方法 25](#_Toc58186649)

[注释 26](#_Toc58186650)

[元组 26](#_Toc58186651)

[以上三种类型的区别 26](#_Toc58186652)

[字典 27](#_Toc58186653)

[类 28](#_Toc58186654)

[\* 29](#_Toc58186655)

[变量打印 29](#_Toc58186656)

[pass 29](#_Toc58186657)

[流程控制 30](#_Toc58186658)

[函数 31](#_Toc58186659)

[异常 32](#_Toc58186660)

[数值判断 33](#_Toc58186661)

[列表推导式(List Comprehension) 33](#_Toc58186662)

[变量作用域 34](#_Toc58186663)

[数据类型转换 35](#_Toc58186664)

[三、数据计算 35](#_Toc58186665)

[numpy 35](#_Toc58186666)

[arrary 35](#_Toc58186667)

[二维数组numpy 36](#_Toc58186668)

[list 39](#_Toc58186669)

[必了解知识 39](#_Toc58186670)

[scipy 41](#_Toc58186671)

[pandas 42](#_Toc58186672)

[引用 42](#_Toc58186673)

[数据结构 42](#_Toc58186674)

[Series 42](#_Toc58186675)

[IO工具 42](#_Toc58186676)

[四、数据可视化 42](#_Toc58186677)

[matplotlib 43](#_Toc58186678)

[五、爬虫 44](#_Toc58186679)

[scrapy 44](#_Toc58186680)

[urllib 48](#_Toc58186681)

[第三方库 50](#_Toc58186682)

[requests 53](#_Toc58186683)

[访问后台 53](#_Toc58186684)

[Post 54](#_Toc58186685)

[爬取meizitu上美图的完整代码(防盗链，多线程) 54](#_Toc58186686)

[防盗链 58](#_Toc58186687)

[XPath 60](#_Toc58186688)

[抓包 61](#_Toc58186689)

[Fiddle 62](#_Toc58186690)

[六、运维 63](#_Toc58186691)

[获取操作系统信息 63](#_Toc58186692)

[七、HTTP访问和接口测试 63](#_Toc58186693)

[get api 63](#_Toc58186694)

[get请求 64](#_Toc58186695)

[post请求 65](#_Toc58186696)

[八、json 65](#_Toc58186697)

[基本 65](#_Toc58186698)

[module 'json' has no attribute 'dumps' 65](#_Toc58186699)

[转换json.dumps() 65](#_Toc58186700)

[读取json 67](#_Toc58186701)

[九、语音朗读 67](#_Toc58186702)

[pyttsx3 67](#_Toc58186703)

[十、多线程 67](#_Toc58186704)

[基本 67](#_Toc58186705)

[协程 68](#_Toc58186706)

[十一、Chrome插件 68](#_Toc58186707)

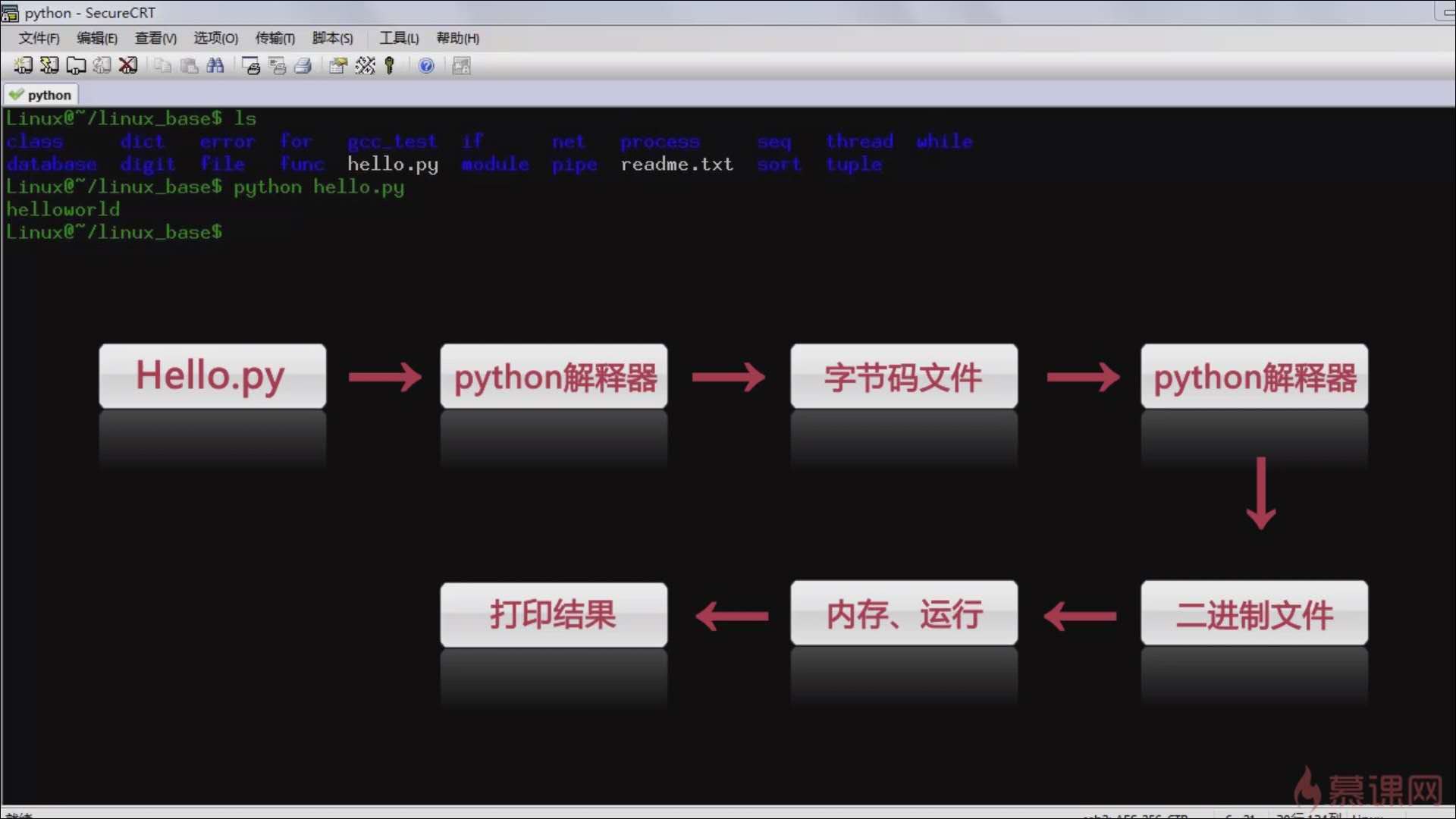
[解决方法： 70](#_Toc58186708)

# 一、平台体系

python基本概念

**Python是解释型语言：**解释型语言，是在运行的时候将程序翻译成机器语言。

解释型语言的程序不需要在运行前编译，在运行程序的时候才翻译，专门的解释器负责在每个语句执行的时候解释程序代码。这样解释型语言每执行一次就要翻译一次，效率比较低。



## Windows下如何运行python

进入[Python官方网站](https://www.python.org/)下载安装包，安装python 2.X或3.X，设置环境变量Path，添加值： ;C:\Program Files\Python36。

运行IDLE，在弹出的窗口中输入pr按tab，会智能联想，输入print (“Hello World!”)，3.X必须得输入()。

也可以安装pycharm，激活方式请查看：<http://www.imsxm.com/jetbrains-license-server.html>

## vs code如何运行python

首先，创建一个空文件夹''hello''，然后使用VS Code打开它。通过VS Code打开文件夹，该文件夹就变成了你的”工作区“。VS Code在.vscode/settings.json中存储该工作去的特殊配置，与用户的全局设定相分开。

使用Ctrl+Shift+P打开命令板，输入Python: Select Interpreter进行搜索。接下来会显示VS Code所能找到的全部解释器，选择你需要的哪个就好。

在Hello文件夹下新建文件, 命名为hello.py。

msg = "Hello World"

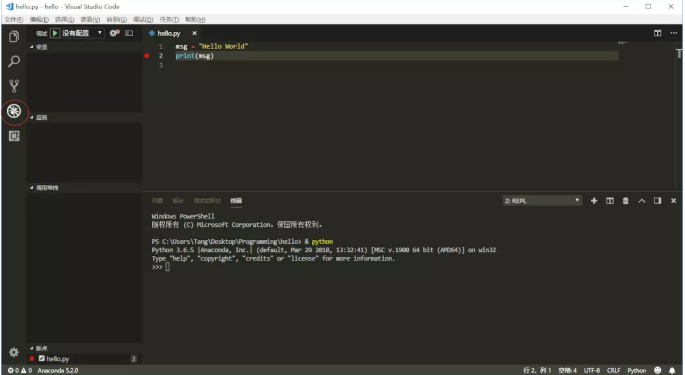
print(msg)

编写完成后按ctrl+s保存。在空白处右键选择**在终端运行Python文件：python f.py**，就可以看到运行结果了。

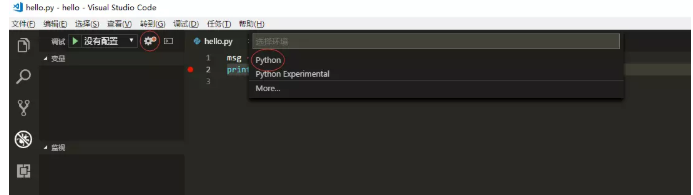
此外，VS Code中还有一些调试Python代码的方式：

把光标移到第二行然后按F9，就可以设置一个断点。同样，也可以在行号左边双击设置。

接下来，在侧边栏打开Debug视图。



然后点击配置按钮，选择Python：



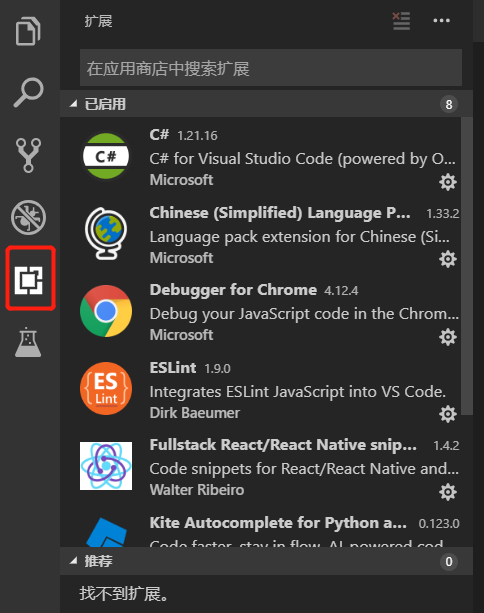
然后Python插件会自动创建包含一系列配置的launch.json文件，可以在下拉列表里面选择，现在选择Python: Current File即可。

为了让调试器在自动在程序开始时停在第一行，添加一条配置stopOnEntry": true，然后保存。

在编辑器中跳转回hello.py，点击绿色箭头或者按F5启动调试器。调试器会停留在文件的第一行。

## vs code常用插件

搜索插件

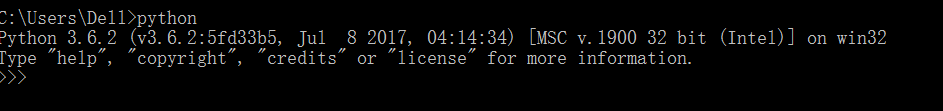


扩展测试

首先，创建一个空文件夹''hello''，

## Windows下设置环境变量

进入Path中添加python安装目录：C:\Program Files (x86)\Python36-32。



## Windows下查看版本号

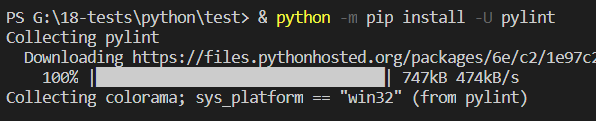
python -V

V大写

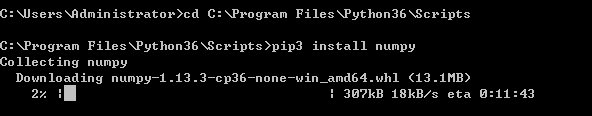
## Windows下如何安装pip

首先电脑已经安装好了python，如果配置好了python的环境变量，无需进入安装目录，直接pip3 numpy即可。

在VS Code里也可以python -m pip install pylint：



找到python的安装目录，接着找到pip.exe，一般而言它会在Scripts文件夹下，我这里选择的是C:\Program Files\Python36\Scripts\pip3.6.exe。



如果出现红色的Exception，则记得将文件夹的everyone读写权限都加上。再重新执行上面的操作，应该是没有问题的，安装完成即可。



## Windows下如何某个包是否安装

你从系统cmd命令行窗口 输入python回车进入python的命令行，然后import pandas试试。

1、如果import失败，说明pandas未安zhi装成功，dao或者pandas安装的路径不在syspath里。请排查。

2、如果import成功，说明pandas安装的没有问题，问题处在IDE上，试着重启IDE或者重新导入本地python路径试试。

## Windows下如何某个库是否安装以及版本

pip show 库名



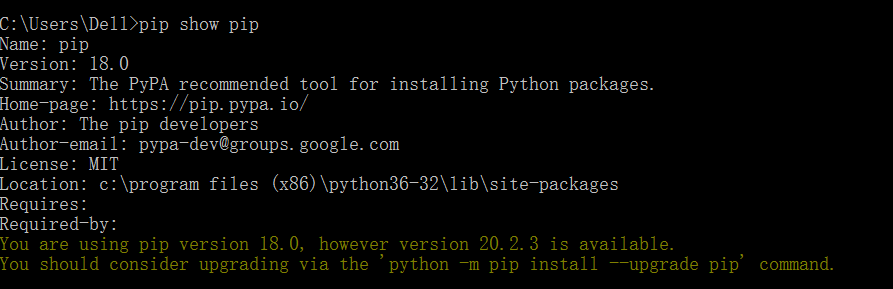
## 更换安装包的版本

pip uninstall xlrd

pip install xlrd==1.2.0

显示pip相关信息，包括pip安装的默认路径

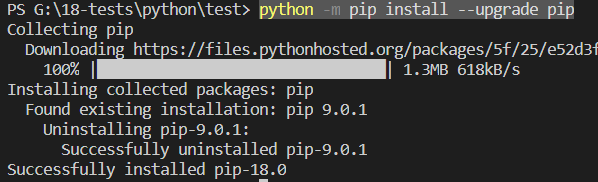
使用pip show pip



升级pip

使用pip install提示：You should consider upgrading via the 'python -m pip install --upgrade pip' command.需要升级pip：

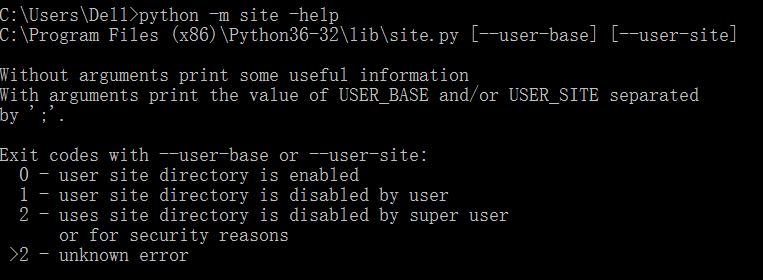
python -m pip install --upgrade pip

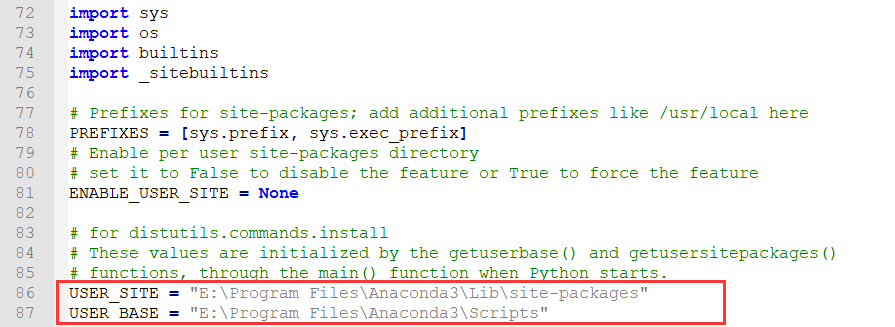


修改pip默认路径

查看pip配置文件，找到site.py：

python -m site -help





<https://www.cnblogs.com/maggieq8324/p/12099068.html>

pip安装某个库

使用pip install wget



安装第三方库失败的情况，配置国内镜像仓库

<https://blog.csdn.net/nulijiushimeili/article/details/79982974>

上面链接的方法亲测可用：

配制成默认的  
在你的“C:\Users\你的用户名\”目录下创建“pip”目录，“pip”目录下创建“pip.ini”文件（注意：以UTF-8 无BOM格式编码）；  
“pip.ini”文件内容：  
[global]    
index-url=http://mirrors.aliyun.com/pypi/simple/    
[install]    
trusted-host=mirrors.aliyun.com

安装第三方库

cmd和vs code中都可以用：

通过wheel

可以自己下载.whl文件进行安装

wheel文件本质上就是zip或者rar,只不过他更加方便python的安装以及使用。在之前的图片中我们只要使用pip install wheel 就可以安装wheel。

之前直接使用pip install beautifulsoup来安装beautifulsoup, 在安装了wheel之后我们可以使使用pip install XXX.whl来安装.whl的文件了。

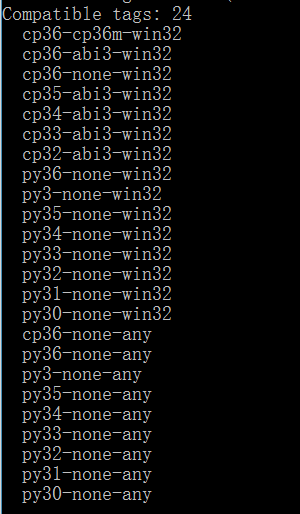
通过wheel安装的问题

会提示：is not a supported wheel on this platform

主要是whl版本问题。

通过<https://blog.csdn.net/xiuxiuen_michelle/article/details/81080694>查询支持的版本，如果不行：

在命令行里：**python -m pip debug –verbose**



然后去找满足要求的whl命名的文件，然后再安装就可以了。

## 模块导入

模块(module)：python中每个python文件就是一个模块，每个python文件中，封装类似功能的变量、函数、类型等等，可以被其他的python模块通过import关键字引入重复使用！

包(pakcage)：包含多个python文件/模块的文件夹，并且文件夹中有一个名称为\_\_init\_\_.py的特殊声明文件，那么这个文件夹就是一个包(模块包)，可以将大量功能相关的python模块包含起来统一管理，同样也可以被其他模块通过import关键字引入重复使用封装的模块和代码！

import math

print math.pow(2,10)

如果我们只希望导入用到的math模块的某几个函数，而不是所有函数，可以用下面的语句：

from math import pow, sin, log

这样，可以直接引用 pow, sin, log 这3个函数，但math的其他函数没有导入进来：

>>> pow(2, 10) 1024.0 >>> sin(3.14) 0.0015926529164868282

如果遇到名字冲突怎么办？比如math模块有一个log函数，logging模块也有一个log函数，如果同时使用，如何解决名字冲突？

如果使用import导入模块名，由于必须通过模块名引用函数名，因此不存在冲突：

import math, logging print math.log(10) # 调用的是math的log函数 logging.log(10, 'something') # 调用的是logging的log函数

如果使用 from...import 导入 log 函数，势必引起冲突。这时，可以给函数起个“别名”来避免冲突：

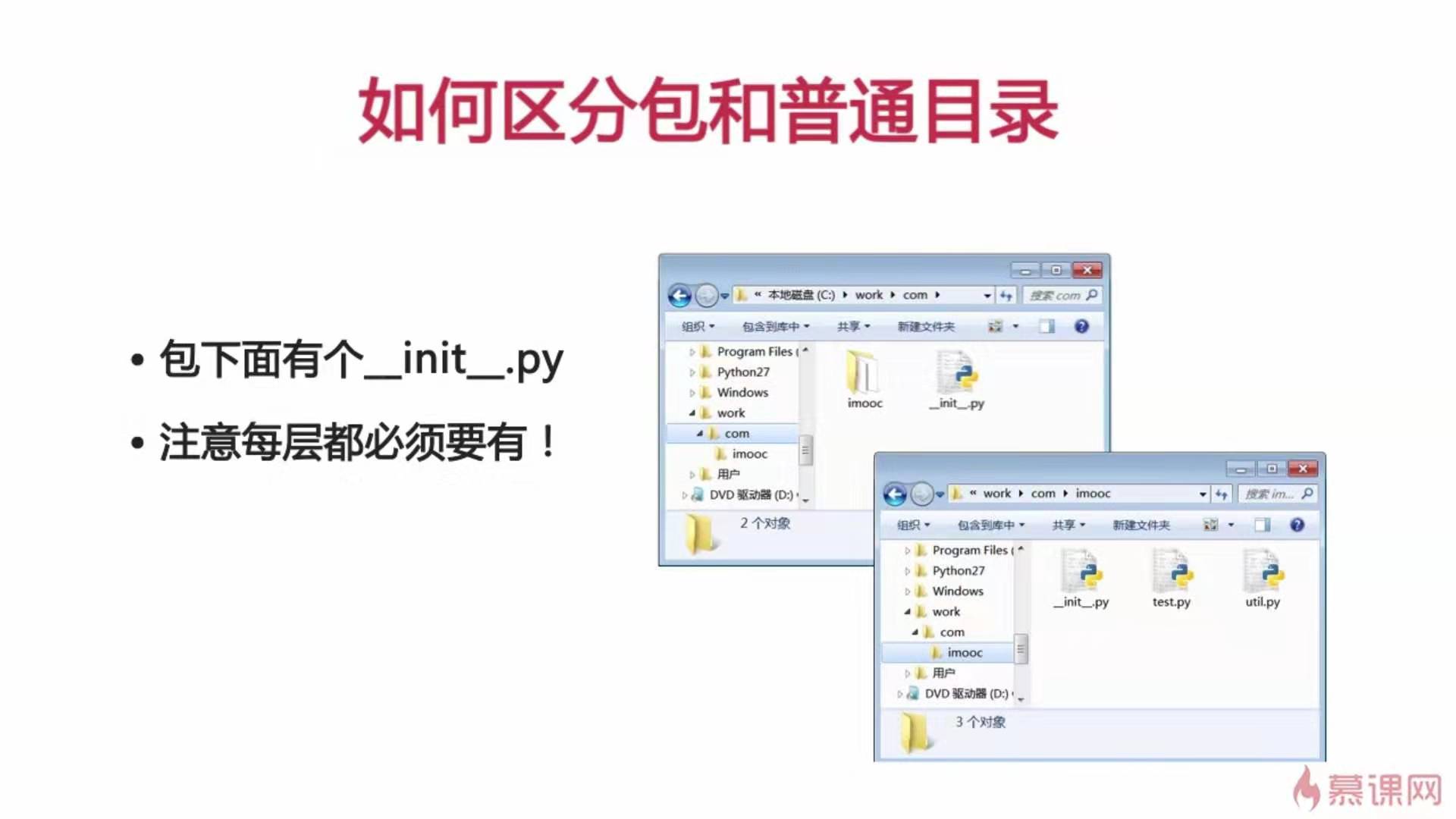
from math import log from logging import log as logger # logging的log现在变成了logger print log(10) # 调用的是math的log logger(10, 'import from logging')。

## 模块重名

加包名

如: import p1.until

python中，每个包下面都有个\_\_init\_\_.py



## from xx import方式

from xx import方式基本语法如下

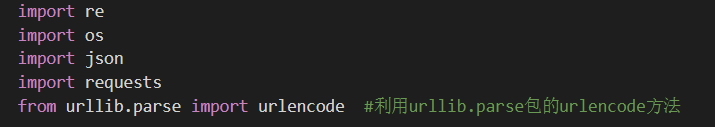
|  |
| --- |
| from . import xxx # 从当前模块路径下，引入xxx模块  from .. import xxx # 从当前模块的父级路径下，引入xxx模块  from pkg import module  # 从 pkg包中引入一个模块module  from pkg.module import vars, func, clazz#   从指定的模块中直接引入 |

## import numpy 和 from numpy import \* 的区别

两种方式都是引入numpy库中的所有函数、函数、对象、变量等，两者的区别在于调用其中内容时不同。

    以调用numpy中的random模块为例，第一种方式要用numpy.random，第二种方式只用random即可。

**但是请特别注意**：pep标准推荐使用第一种方式，请在日常使用中尽量使用第一种方法，就比如numpy中random  标准库中也有random，但是两者的功能是不同的，使用第二种方式容易造成混淆。



举例import tkinter与from tkinter import \*的区别

from tkinter import \*

class DirList(object):

def \_\_init\_\_(self,initdir=None):

self.top= Tk() ##可以不加tkinter前缀

self.label = Label(self.top, text="Directory Lister V1.1") ##可以不加tkinter前缀

self.label.pack()

import tkinter

class DirList(object):

def \_\_init\_\_(self,initdir=None):

self.top= tkinter.Tk() ##必须加tkinter前缀

self.label = tkinter.Label(self.top, text="Directory Lister V1.1") ##必须加tkinter前缀

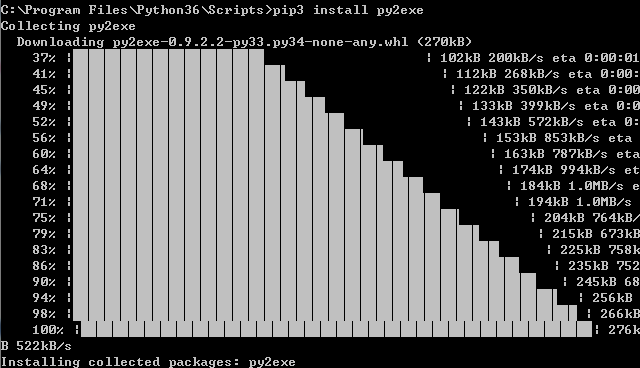
self.label.pack()

## 语法糖

这是一种编程语法，它并不会带来新的特性，但能使代码更易读、更易写。

## 打包成独立执行的exe可执行文件

py2exe，顾名思义，就是将.py文件转换成可执行的.exe文件，这样就可以摆脱编译环境和预载模块的限制，在任一windows环境下运行。



py2exe太过陈旧缺少维护操作不便，推荐PyInstaller项目 [www.pyinstaller.org](https://link.zhihu.com/?target=http%3A//www.pyinstaller.org/),可用一句命令打包：pyinstaller -F -w -i manage.ico app.py

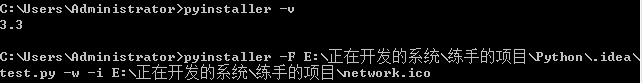
-F：打包为单文件；

-w：Windows程序，不显示命令行窗口；

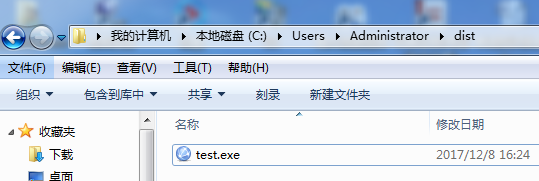
-i：是程序图标，app.py是你要打包的py文件。

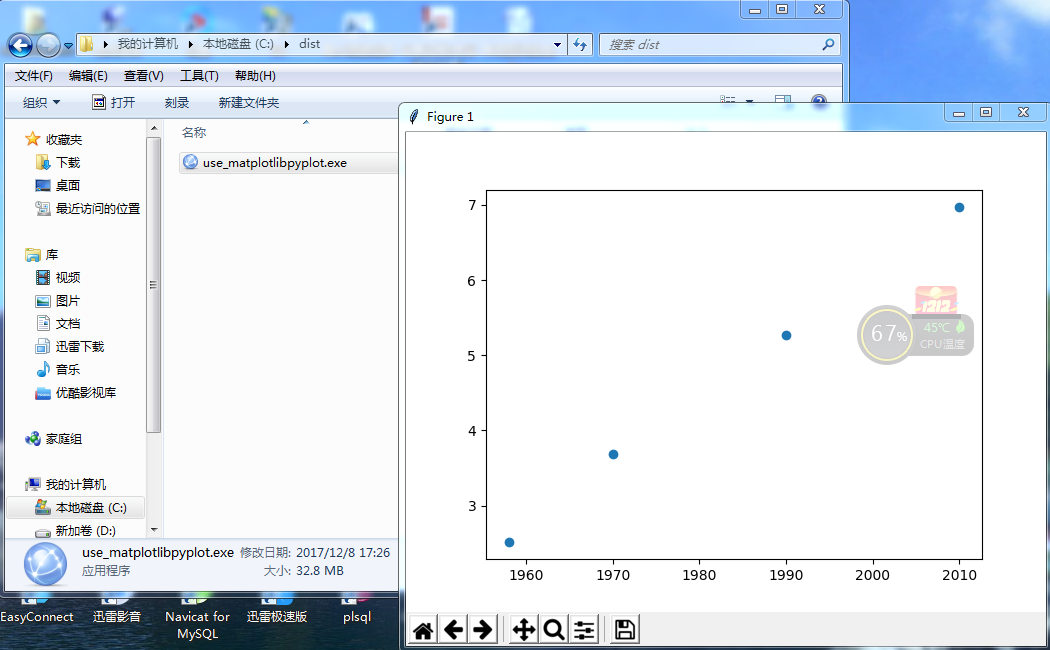
安装方法：先跑pip3 install pypiwin32再跑pip3 install pyinstaller即可。

cmd中运行” pyinstaller -v”如果提示“pyinstaller不是内部或外部的命令”，安装后的目录是：C:\Program Files\Python36\Lib\site-packages\PyInstaller，但是需要将目录(C:\Program Files\Python36\Scripts)加入环境变量中。









貌似print的生成exe，运行后没反应，用matplotlib库生成统计图，双击就没问题。

具体的命令：pyinstaller -F E:\正在开发的系统\练手的项目\Python\.idea\use\_matplotlibpyplot.py -w -i E:\正在开发的系统\练手的项目\network.ico

卸载：pip3 uninstall Pyinstaller

# 二、基本语法

和.Net等的区别

### {}

用:替代。

如def 函数名:

函数体

又如：

if(os.name=='nt'):  
 print('你正在使用windows平台！')

语句后不用;

变量

变量存储在内存中的值。这就意味着在创建变量时会在内存中开辟一个空间。基于变量的数据类型，解释器会分配指定内存，并决定什么数据可以被存储在内存中。

Python 中的变量赋值不需要类型声明。

每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建(和.net不同，类似lazy)。

a = 123 # a是整数

print(a)

a = 'ABC' # a变为字符串

print(a)

这种变量本身类型不固定的语言称之为动态语言，与之对应的是静态语言。静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如Java是静态语言，赋值语句如下（// 表示注释）：

**int** a = 123; *// a是整数类型变量*

a = "ABC"; *// 错误：不能把字符串赋给整型变量*

a = 'ABC'

b = a

a = 'XYZ'

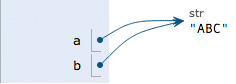
print(b)

最后一行打印出变量b的内容到底是'ABC'呢还是'XYZ'？如果从数学意义上理解，就会错误地得出b和a相同，也应该是'XYZ'，但实际上b的值是'ABC'，让我们一行一行地执行代码，就可以看到到底发生了什么事：

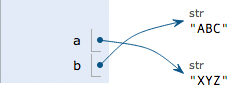
执行a = 'ABC'，解释器创建了字符串'ABC'和变量a，并把a指向'ABC'：

py-var-code-1

执行b = a，解释器创建了变量b，并把b指向a指向的字符串'ABC'：



执行a = 'XYZ'，解释器创建了字符串'XYZ'，并把a的指向改为'XYZ'，但b并没有更改：



所以，最后打印变量b的结果自然是'ABC'了。

Python允许你同时为多个变量赋值。例如：

a = b = c = 1

以上实例，创建一个整型对象，值为1，三个变量被分配到相同的内存空间上。

您也可以为多个对象指定多个变量。例如：

a, b, c = 1, 2, "john"

以上实例，两个整型对象1和2的分配给变量 a 和 b，字符串对象 "john" 分配给变量 c。

标准数据类型

Python有五个标准的数据类型：

Numbers（数字）

String（字符串）

List（列表）

Tuple（元组）

Dictionary（字典）



type-获取数据类型

a=3.14

print(type(a))

classmates = ['Michael', 'Bob', 'Tracy']

print(type(classmates))

<class 'float'>

<class 'list'>

id-获取变量内存地址

id() 函数返回对象的唯一标识符，标识符是一个整数。CPython 中 id() 函数用于获取对象的内存地址。

数字

数字数据类型用于存储数值。他们是不可改变的数据类型，这意味着改变数字数据类型会分配一个新的对象。

当你指定一个值时，Number对象就会被创建，您可以通过使用del语句删除单个或多个对象的引用。例如：del var\_a, var\_b

Python支持四种不同的数字类型：int（有符号整型）、long（长整型[也可以代表八进制和十六进制]）、float（浮点型）、complex（复数）

字符串

字符串或串(String)是由数字、字母、下划线组成的一串字符。

如果你要实现从字符串中获取一段子字符串的话，可以使用变量 [头下标:尾下标]，就可以截取相应的字符串。s = 'ilovepython'，s[1:5]的结果是love。上面的结果包含了s[1]的值l，而取到的最大范围不包括上边界，就是s[5]的值p。索引从0开始。

3.x中unicode就是str。

索引index

print(s.index('o'))

print(s.index('o',2))

print(s.index('o',9))

结果：

3

3

11

切片

s="I love python"

print(s[1:5]) # lov 从1到5，包含1包含4不包含5,list的索引是从1开始

print(s[1:5:2]) #2是步长

print(s[::-1]) #步长-1代表倒序排

print(s[-1:-5:-1]) #起始要从-1开始才行

结果：

lov

o

nohtyp evol I

noht

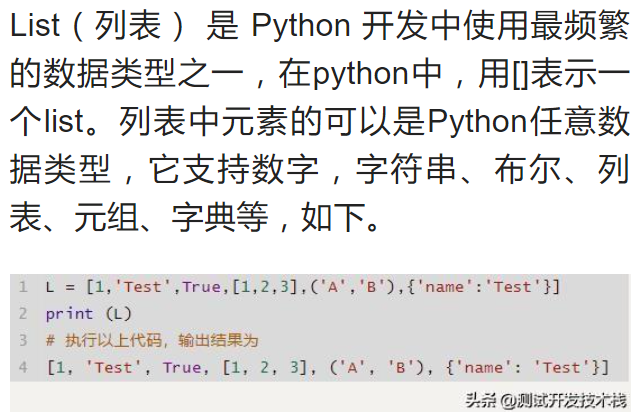
split

合并字符串

join

连接字符串

列表



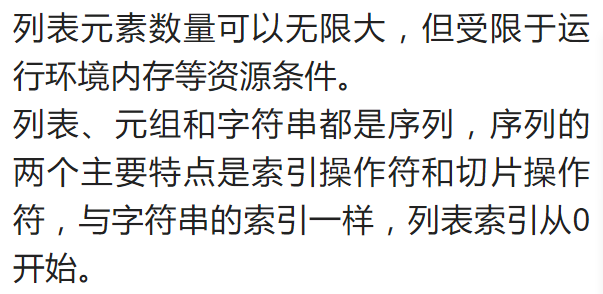
fam=[1,2,3,4,5]

print(fam[4])

列表中可以是任意类型，感觉就是个object的列表，注意看一下效率。

区别：在C#中，数组/列表是用{}包含的。

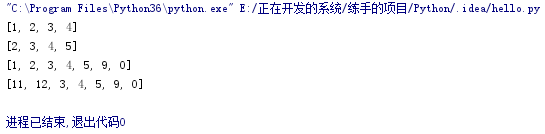
fam=[1,2,3,4,5]  
print(fam[:4])



从第1个元素（index=0）到index=4（不包含4）

hello=fam+[9,0];  
print(hello);  
hello[0:2]=[11,12];  
print(hello);

结果如下：



空列表：lst=[]或者lst=list()

i=int()，int不是列表，此时i=0

列表中操作其实是内存中的引用。

ab=hello;

ab[0:2]=[21,22];

print(ab)

print(hello)

结果：



如果只是拷贝值，而不是引用。

ae=list(hello)  
ae[0:2]=[1,2]  
print(hello)  
print(ae)

结果：



可以通过max函数求最大值

tallest=max(hello)

print(tallest)

print(type(tallest))

结果：



排序

fam=[2,1,0,5,7,9,2,10]

print(id(fam))

print(fam.sort(reverse=True))

print(fam) #和-1那种方式不同，不会新建一个list

sort()，默认reverse=False

移除元素

fam.pop(3) #默认索引是-1，也就是删除最后一个

print(fam)

fam.remove(2) #代表删除元素2，如果有两个2，就删除第一个

print(fam)

添加元素

fam.append(100)

print(fam)

不允许直接通过fam[10]=100的方式添加，会提示超过索引。

初始化长度

tm\_arr=list(range(size))   #预先初始化列表长度

思想荟获取答案

#coding:utf-8

size=25                    #题目数量

tm\_arr=list(range(size))   #预先初始化列表长度

#文本里过一下：];替换为]

tm\_arr[0]=['040B1D7E-518B-4440-848C-656D8F67C2A3','2','<p>数据安全是：</p>','4','4','通过采取必要措施，保障数据得到有效保护和合法利用|通过采取必要措施，保障数据得到有效保护和合法利用，并持续处于安全状态的能力|通过采取必要措施，持续处于安全状态的能力|通过采取必要措施，保障数据活动安全。','','0','8a57235c-6af6-45d2-8eb0-bb1105d2b89c','','b','0','','3','0','100']

tm\_arr[1]=['32B764AD-36A0-4378-B38C-A5E6C73146CF','2','企业停止运营其产品或服务时，下列哪个做法是错误的。（　　）','4','4','及时停止继续收集个人信息的活动|将停止运营的通知以逐一送达或公告的形式通知用户|对其所持有的个人信息及时进行备份，以便可以在日后开展其他经营活动|对其所持有的个人信息及时进行删除或匿名化处理','','0','4a6f3239-4125-4205-abe6-38d6b3fe1e88','','c','','','1','0','100']

tm\_arr[2]=['B0920000-532E-4D5B-97BF-9D53ADD7023F','2','<p>《最高人民法院、最高人民检察院关于办理利用信息网络实施诽谤等刑事案件适用法律若干问题的解释》中，明确“利用信息网络诽谤他人”犯罪标准，同一诽谤信息实际被点击、浏览次数达到（　）以上，或被转发达次数达500次以上可判刑。</p>','4','4','500|1000|5000|10000','','0','31442085-f9bb-46cd-bbde-76b2976335d4','','c','0','','1','0','100']

tm\_arr[3]=['5F054273-9400-4333-9B33-969D3C6BE1AA','2','以下做法中正确的是（　　）','4','4','某单位开发的APP没有经过用户的同意就对用户的个人信息进行收集和使用，但告知了用户收集个人信息的目的、方式和范围。|某事业部在运营的网站中通过与用户签订在线许可协议的方式获取用户同意后收集、使用用户的个人信息。|某事业部告知用户将在提供在线教学服务时使用用户电脑及手机的摄像头，但在教学服务停止后仍然通过该摄像头收集用户的相关图像。|某培训中心在收集学员个人信息时除了必须的姓名、电话、联系地址之外，还收集了学员血型、家长政治面貌、宗教信仰等信息。','','0','41889448-9f02-4f7b-a0cb-ad48f0efa705','','b','','','1','0','100']

tm\_arr[4]=['0692C370-1749-41A7-BFD3-46C7F2288492','2','<p>根据《网络安全法》的规定，关键信息基础设施的运营者在中华人民共和国境内运营中收集和产生的个人信息和重要数据应当在（&nbsp; &nbsp; ）。因业务需要，确需向境外提供的，应当按照国家网信部门会同国务院有关部门制定的办法进行安全评估，法律、行政法规另有规定的，依照其规定。</p>','4','4','境外存储|外部存储器存储|第三方存储|境内存储','','0','2d4c5d2b-bc03-4f98-8fd4-f6ab885035d2','','d','0','','1','0','100']

tm\_arr[5]=['430AC2BF-82B5-4C4D-81EB-B70A55B844F2','2','关于安全移动存储，下列说法错误的是（　　）。','4','4','严禁将安全移动存储介质更改为普通移动存储介质|定期清理移动存储介质内容|妥善保管移动存储介质|移动存储介质可以随时借给外部单位或转借他人使用','','0','cd08a1a0-39dc-49fc-85d0-fe2b61dde8bd','','d','','','1','0','100']

tm\_arr[6]=['B2E59989-DB3F-454C-8C8A-1E352E87E98F','2','<p>数据是：</p>','4','4','任何以电子或者非电子形式对信息的记录|任何以电子形式对信息的记录|任何以非电子形式对信息的记录|以纸质形式对信息的记录','','0','2ce5d81a-77a7-4ef7-9770-d1743792d5d3','','a','0','','3','0','100']

tm\_arr[7]=['15B0A939-E3BE-4475-BB57-B6105971FD76','2','关于APP收集、使用用户个人信息的说法正确的是（　　）','4','4','APP经用户许可收集用户的个人信息后，又向其他APP的经营者提供该用户个人信息的，无须获得用户同意。|APP经用户许可收集用户的个人信息后，将该个人信息进行非标识化处理，此时可以不经过用户的同意即向其他APP经营者提供该数据。|用户个人发现APP违反法律、行政法规的规定或者双方的约定收集、使用其个人信息的，有权要求APP运营单位删除其个人信息。|用户发现APP收集、存储的其个人信息有错误的，无权要求APP运营单位进行更正。','','0','cbd46462-80ff-4804-9243-6e96560efdce','','c','','','1','0','100']

tm\_arr[8]=['C0BFA68C-B82E-4803-AFFE-16BC2E3ED0CD','2','<p>在连接互联网的Github网站上（　　）存储涉及国家秘密和企业秘密信息。</p>','4','4','可以|严禁|不确定|因为Github网络环境是安全的，所以可以','','0','77d2d90e-7e2b-4074-8c14-2031e4320afb','','b','0','','1','0','100']

tm\_arr[9]=['6FB885BB-B44D-485C-8D90-B7C9B2D5AA8D','2','<p>根据《网络安全法》的规定，（&nbsp; &nbsp; ）负责统筹协调网络安全工作和相关监督管理工作。</p>','4','4','中国电信|信息部|国家网信部门|中国联通','','0','8e32a12f-0a5e-46ef-b140-6a875e41ef6a','','c','0','','1','0','100']

tm\_arr[10]=['387246F5-7468-40CD-9D8E-7D542FA42EB2','3','<p>数据安全方面的法律责任：</p>','4','4','开展数据活动的组织、个人不履行本法第二十五条、第二十七条、第二十八条、第二十九条规定的数据安全保护义务或者未采取必要的安全措施的，由有关主管部门责令改正，给予警告，可以并处一万元以上十万元以下罚款，对直接负责的主管人员可以处五千元以上五万元以下罚款|拒不改正或者造成大量数据泄漏等严重后果的，处十万元以上一百万元以下罚款，对直接负责的主管人员和其他直接责任人员处一万元以上十万元以下罚款|数据交易中介机构未履行本法第三十条规定的义务，导致非法来源数据交易的，由有关主管部门责令改正，没收违法所得，处违法所得一倍以上十倍以下罚款，没有违法所得的，处十万元以上一百万元以下罚款，并可以由有关主管部门吊销相关业务许可证或者吊销营业执照；对直接负责的主管人员和其他直接责任人员处一万元以上十万元以下罚款。|未取得许可或者备案，擅自从事本法第三十一条规定业务的，由有关主管部门责令改正或者予以取缔，没收违法所得，处违法所得一倍以上十倍以下罚款；没有违法所得的，处十万元以上一百万元以下罚款；对直接负责的主管人员和其他直接责任人员处一万元以上十万元以下罚款','','0','23cfa446-babd-435c-9733-c84fc4ac7796','','abcd','0','','3','0','100']

tm\_arr[11]=['BF6FB530-722F-4700-9DC0-BACA4B2D467A','3','<p>国家网信部门应当统筹协调有关部门对关键信息基础设施的安全保护采取下列（&nbsp; ）措施。</p>','4','4','对网络安全事件的应急处置与网络功能的恢复等，提供技术支持和协助|促进有关部门、关键信息基础设施的运营者以及有关研究机构、网络安全服务机构等之间的网络安全信息共享|对关键信息基础设施的安全风险进行抽查检测，提出改进措施，必要时可以委托网络安全服务机构对网络存在的安全风险进行检测评估|定期组织关键信息基础设施的运营者进行网络安全应急演练，提高应对网络安全事件的水平和协同配合能力','','0','6f9ccf6a-236e-4505-8673-15d928c94171','','abcd','0','','1','0','100']

tm\_arr[12]=['7AB2F546-B90A-4136-8643-5F3758AE651E','3','<p>以下哪些针对个人信息的操作属于用户权利保障（&nbsp; &nbsp;）。</p>','4','5','个人信息查询|个人信息更正|个人信息删除|用户账户注销|撤回已同意的授权','','0','4cfc4b63-df58-47d7-8625-1afa2bc39897','','abcde','0','','1','0','100']

tm\_arr[13]=['6B2C1E98-CA21-46CE-AE20-48F64BFAC718','3','<p>风险评估报告应包含的内容：</p>','4','4','本组织掌握的重要数据的种类、数量，收集、存储、加工、使用数据的情况|面临的数据安全风险及其应对措施等|发生数据安全事件时，应当按照规定及时告知用户并向有关主管部门报告|发现数据安全缺陷、漏洞等风险时，应当立即采取补救措施','','0','9c451470-a679-4158-b946-210214524f77','','ab','0','','3','0','100']

tm\_arr[14]=['99DDCC5D-6F60-4B80-85B3-C236799DFED8','3','<p>《网络安全法》　中统一采用了“个人信息”的表述，并将个人信息定义为“以电子或者其他方式记录的能够单独或者与其他信息结合识别自然人个人身份的各种信息，个人信息包括（　）。</p>','4','4','姓名|出生日期|身份证件号码|公司人事信息','','0','18460878-d801-4d19-9131-d2b25ca436ac','','abc','0','','1','0','100']

tm\_arr[15]=['7D8A89D6-F3FB-4E73-B7BB-7A8FFE61CD8A','3','以下属于刑法规定的“情节严重”的非法获取、出售或者提供公民个人信息行为是（　　）。','4','5','非法获取、出售或者提供行踪轨迹信息、通信内容、征信信息、财产信息五十条以上的|非法获取、出售或者提供住宿信息、通信记录、健康生理信息、交易信息等其他可能影响人身、财产安全的公民个人信息五百条以上的|非法获取、出售或者提供A项、B项规定以外的公民个人信息五千条以上的|知道或者应当知道他人利用公民个人信息实施犯罪，向其出售或者提供的|违法所得五万元以上的','','0','635a5437-283c-400b-bde7-2c56132ab401','','abcd','','','1','0','100']

tm\_arr[16]=['A1BC8A7A-E768-44A7-8923-EFF5A76B29DA','3','<p>制定数据安全法的目的是：</p>','4','4','保障数据安全|促进数据开发利用|保护公民、组织的合法权益|维护国家主权、安全和发展利益','','0','31360c91-2ebb-4fb2-b82a-ea36a029a999','','abcd','0','','3','0','100']

tm\_arr[17]=['EBD84ABA-88E0-4A03-A2ED-447FCBD269BD','3','用户个人信息保护管理涵盖个人信息哪些环节的相关行为（　　　）。','4','5','收集|存储|使用|销毁|共享、转让、公开披露','','0','4b8ff807-0553-437c-a87c-60f7cdfc2288','','abcde','','','1','0','100']

tm\_arr[18]=['1D837B5C-973F-4ADD-A3EC-B8C5DFCADEC2','3','中国网络安全行业相关的国家部门包括（　）。','4','4','中央网信办|工信部|公安部|国家保密局','','0','382fc169-920a-4a76-bb54-a357337e237c','','abcd','','','1','0','100']

tm\_arr[19]=['13F55292-AA52-4C34-8701-53166B736CE0','3','<p>对数据实行分级分类保护的依据是：</p>','4','4','数据在经济社会发展中的重要程度|一旦遭到篡改、破坏、泄露或者非法获取、非法利用，对国家安全、公共利益或者公民、组织合法权益造成的危害程度|数据的规模|数据的责任人重要程度','','0','1a758a55-4030-4551-957c-b1eb210c5fbe','','ab','0','','3','0','100']

tm\_arr[20]=['3EB2B84A-E4D3-4691-8024-6F6791D5701B','4','指纹、虹膜等生物识别信息属于个人敏感信息。','4','1','','','0','4bc7ab2d-1a1d-4928-aa68-1a99676eaef9','','1','','','1','0','100']

tm\_arr[21]=['61E04BDC-3DD4-4C97-B0CC-488C89B59652','4','为保护个人信息安全，我们应及时撕毁或者涂改快递包裹上的个人信息。','4','1','','','0','375c5692-723e-40d8-9173-0c4a33d6245f','','1','','','1','0','100']

tm\_arr[22]=['5244BB29-D913-4DAD-BFB1-32033DD00B48','4','任何个人和组织不得窃取或者以其他方式获取个人信息，不得非法出售或者非法向他人提供个人信息。','4','1','','','0','08825479-de2d-4b53-810a-892faf33d5ec','','1','','','1','0','100']

tm\_arr[23]=['0494BC73-08D1-41E2-8B47-005A6BF0D477','4','《中华人民共和国个人信息保护法》已于2020年10月1日正式发布。','4','1','','','0','1a0a6a92-6395-43c1-b684-b1d132300abd','','0','','','1','0','100']

tm\_arr[24]=['CE809C67-4925-4831-B130-34FF92F97737','4','为方便亲朋好友、网友随时掌握个人行踪，我们可以任意在公开社交网站上发布火车票、飞机票、护照和日程等信息。','4','1','','','0','bd319a7e-75be-4f4b-b498-f04e8118c267','','0','','','1','0','100']

answers=list(range(size))

for i in range(size):

    answers[i]=(tm\_arr[i][-6])

print(answers)

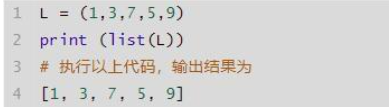
常见方法

len(list):获取列表元素个数。

max(list):获取列表元素最大值。

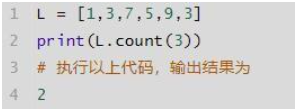
min(list):获取列表元素最小值。

list (seq):元组转换为列表。

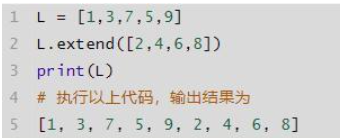


list.append(obj):列表末尾添加新对象。

list.count(obj):统计某个元素在列表中出现的次数。



list.extend(seq):列表末尾一次性追加另一个序列多个元素。



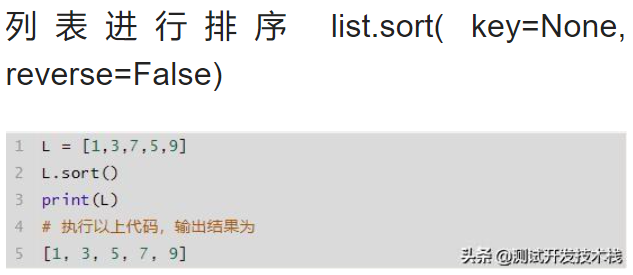
list.index(obj):统计某个元素在列表中出现的次数。



list.insert(obj):统计某个元素在列表中出现的次数。







Round()方法

print(round(1.68,1))

print(round(1.68))

round(1.68,1)：小数点后保留1位

round(1.68)：保留距离小数点最近的整数

结果：



值所在的索引：

print(fam.index(2))

结果：



出现的次数：

print(fam.count(5))

结果：



添加元素也可以使用：

fam.append("tom")

fam.append(100)

print(fam)

print(famtwo)

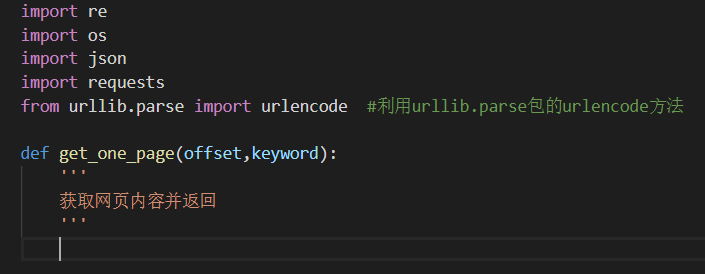
结果：



注释

另单行：#

多行(三个单引号)：’’’



元组

元组是另一个数据类型，类似于List（列表）。元组用"()"标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表。

元组是不允许更新的。而列表是允许更新的。

以上三种类型的区别

o\_list=[1,2,3,4,5,1]

o\_touple=(1,2,3,4) #不可变

o\_list.append(2) #可变

o\_set=set(o\_list) #去重

print(type(o\_list))

print(type(o\_touple))

print(type(o\_set))

print(o\_list)

print(o\_touple)

print(o\_set)

结果：

<class 'list'>

<class 'tuple'>

<class 'set'>

[1, 2, 3, 4, 5, 1, 2]

(1, 2, 3, 4)

{1, 2, 3, 4, 5}

字典

字典(dictionary)是除列表以外python之中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。

两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。字典用"{ }"标识。字典由索引(key)和它对应的值value组成。如：

dict = {}

dict['one'] = "This is one"

dict[2] = "This is two"

或：

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}



例子：

o\_dict={'a':'100','b':'200'}

print(type(o\_dict))

print(o\_dict.keys())

print(o\_dict.values())

结果：

<class 'dict'>

dict\_keys(['a', 'b'])

dict\_values(['100', '200'])











类

在Python中，可以通过class关键字定义自己的类，然后通过自定义的类对象类创建实例对象。

例如，下面创建了一个Student的类，并且实现了这个类的初始化函数”\_\_init\_\_”:

class Student(object):

    count = 0

    books = []

    def \_\_init\_\_(self, name, age):

        self.name = name

        self.age = age

    pass

\*

在Python中，字符串\*代表重复打印多少遍

max\_page=find\_MaxPage()  
print(max\_page\*2)

结果：



时间

time.time()

时间戳单位最适于做日期运算。

格式化日期

localtime = time.localtime(time.time())

print("本地时间为 :",time.strftime("%Y-%m-%d %H:%M:%S",localtime))

data.strftime('%Y-%m-%d')，data为时间变量。

具体时间详见：

https://www.runoob.com/python/att-time-strftime.html

显示中文时间：

data.strftime('%Y{y}%m{m}%d{d}').format(y='年',m='月',d='日')

python 格式化时间含中文报错： 'locale' codec can't encode character '\u5e74'

data.strftime('%Y{y}%m{m}%d{d}').format(y='年',m='月',d='日')

日历

Calendar模块有很广泛的方法用来处理年历和月历，例如打印某月的月历：

import calendar

cal=calendar.month(2016,1) #2016年1月的日历

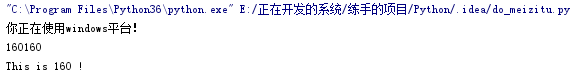
print(cal)



变量打印

max\_page=find\_MaxPage()  
print("This is %s !" %max\_page)

结果：



pass

Python pass是空语句，是为了保持程序结构的完整性。pass 不做任何事情，一般用做占位语句。

流程控制

Python中可以使用if、for和while来实现流程控制。Python中并没有select，取而代之使用if来实现。使用for来枚举列表中的元素。如果希望生成一个由数字组成的列表，则可以使用range(<number>)函数。以下是这些声明的语法示例，注意for这句后有个冒号：

farray=[1,2,3,4,5]

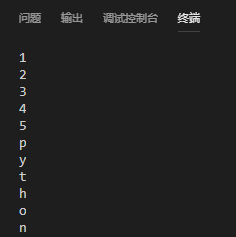
for i in range(len(farray)):

print(farray[i])

for letter in 'python':

print(letter)

结果：



rangelist = range(10)

>>> print rangelist

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

for number in rangelist:

# Check if number is one of

# the numbers in the tuple.

if number in (3, 4, 7, 9):

# "Break" terminates a for without

# executing the "else" clause.

break

else:

# "Continue" starts the next iteration

# of the loop. It's rather useless here,

# as it's the last statement of the loop.

continue

else:

# The "else" clause is optional and is

# executed only if the loop didn't "break".

pass # Do nothing

if rangelist[1] == 2:

print "The second item (lists are 0-based) is 2"

elif rangelist[1] == 3:

print "The second item (lists are 0-based) is 3"

else:

print "Dunno"

while rangelist[1] == 1:

pass

函数

函数通过“def”关键字进行声明。可选参数以集合的方式出现在函数声明中并紧跟着必选参数，可选参数可以在函数声明中被赋予一个默认值。Lambda函数是由一个单独的语句组成的特殊函数，参数通过引用进行传递。

funvar=lambda x:x\*1

print(funvar(100))



相当于(下方这么写是可以的)：

def funcvar(x): return x + 1

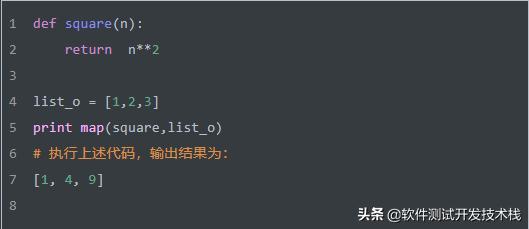
高阶函数

变量可以指向函数。

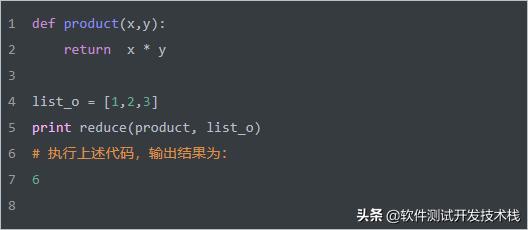


Python函数式编程中常用的高阶函数（内建函数）：

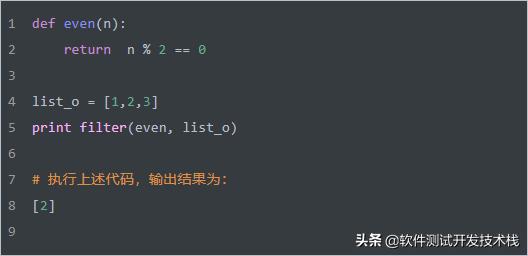
**map()：**它接收一个函数 function 和一个 list，第一个参数 function 以参数序列中的每一个元素调用 function 函数，返回包含每次 function 函数返回值的新列表。Python 2.x 返回列表，Python 3.x 返回迭代器。



**reduce()：**函数会对参数序列中元素进行累积。



**filter()**：函数用于过滤序列，过滤掉不符合条件的元素，返回由符合条件元素组成的新列表。



sorted()函数：可以对list进行排序。

正则表达式

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。re 模块使 Python 语言拥有全部的正则表达式功能，re模块常用方法：

re.match

re.search

re.sub

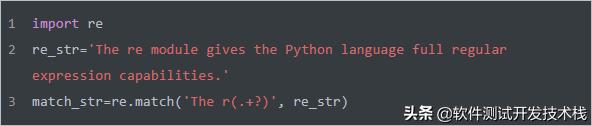
re.compile

re.finditer

re.findall

re.split

re.match从字符串的起始位置匹配，如果起始位置匹配不成功，则match()就返回none。如果匹配成功，则可通过group(num) 或 groups()获取匹配结果。



group() 匹配的整个表达式的字符串，group() 可以一次输入多个组号，在这种情况下它将返回一个包含那些组所对应值的元组。

异常

Python中的异常由 try-except [exceptionname] 块处理，例如：

def some\_function():

try:

# Division by zero raises an exception

10 / 0

except ZeroDivisionError:

print "Oops, invalid."

else:

# Exception didn't occur, we're good.

pass

finally:

# This is executed after the code block is run

# and all exceptions have been handled, even

# if a new exception is raised while handling.

print "We're done with that."

数值判断

数值判断可以链接使用，例如 1<a<3 能够判断变量 a 是否在1和3之间。

列表推导式(List Comprehension)

列表推导式(List Comprehension)提供了一个创建和操作列表的有力工具。列表推导式由一个表达式以及紧跟着这个表达式的for语句构成，for语句还可以跟0个或多个if或for语句。

list1=[1,2,3]

list2=[4,5,6]

print([x\*y for x in list1 for y in list2])

结果：



print(sum(i for i in [3,3,4,4,5,5]))

结果：



print(sum(i for i in [3,3,4,4,5,5] if i==4))

结果：



注意和上面的不同：

print(sum(1 for i in [3,3,4,4,5,5] if i==4))

结果：



print(sum(1 for i in [3,3,4,4,5,5]))

结果：



if \_\_name\_\_ == '\_\_main\_\_'详解

在Python模块当中，通常会写上一行“if \_\_name\_\_==’\_\_main\_\_:”作为程序的入口，但似乎没有这么一行代码，程序也能正常运行。那么这句代码多余吗？为什么？

Python模块通常有两种运行方式，第一种作为脚本直接运行，第二种是被导入（ import ）到其他的 Python 模块中。if \_\_name\_\_==’\_\_main\_\_: 的作用就是控制这两种情况代码执行过程，在if \_\_name\_\_==’\_\_main\_\_: 下的代码只有在第一种情况下才会被执行，而 import 到其他脚本中是不会被执行的。

变量作用域

全局变量在函数之外声明，并且可以不需要任何特殊的声明即能读取，但如果你想要修改全局变量的值，就必须在函数开始之处用global关键字进行声明，否则Python会将此变量按照新的局部变量处理（请注意，如果不注意很容易被坑）。S

number=5  
def myfun():  
 print(number)  
  
def anotherfun():  
 number=100  
  
def setanotherfun():  
 global number  
 number=10  
  
if \_\_name\_\_=='\_\_main\_\_':  
 myfun()  
 anotherfun()  
 myfun()  
 setanotherfun()  
 myfun()

结果：



数据类型转换

转换成int：int(max\_page)

转换为string：str(max\_str)

# 三、数据计算

numpy

arrary

import numpy as np

height=[1.73,1.68,1.71,1.89,1.79]

weight=[65.4,59.2,63.6,88.4,68.7]

np\_height=np.array(height)

np\_weight=np.array(weight)

bmi=np\_weight/np\_height \*\* 2

print(bmi)

结果：



选取数组里的元素

print(bmi[bmi>21])

结果：



import numpy as np

height=[1.73,1.68,1.71,1.89,1.79]

weight=[65.4,59.2,63.6,88.4,68.7]

np\_height=np.array(height)

np\_weight=np.array(weight)

bmi=np\_weight/np\_height \*\* 2

print(bmi)

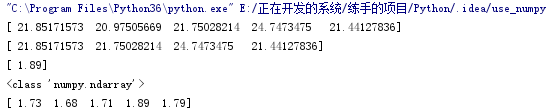
print(bmi[bmi>21])

print(np\_height[np\_height>1.8])

print(type(np\_height))

print(np\_height)

结果：



二维数组numpy

np\_2d=np.array([[1.73,1.68,1.71,1.89,1.79],

[65.4, 59.2, 63.6, 88.4, 68.7]])

print(type(np\_2d))

print(np\_2d.shape)

结果：



表示是一个2行5列的数组。

如果其中一个元素改成了字符型，则该数组的所有元素都为字符型。

np\_2d=np.array([[1.73,1.68,1.71,1.89,1.79],

[65.4, 59.2, 63.6, 88.4, 68.7],

[1.73, 1.68, 1.71, 1.89, 1.71],

[1.73, 1.68, 1.71, 1.89, 1.78],])

print(type(np\_2d))

print(np\_2d.shape)

print(np\_2d[0])

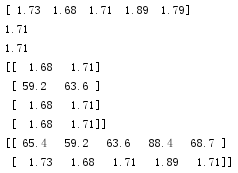
print(np\_2d[0][2])

print(np\_2d[0,2])

print(np\_2d[:,1:3])

print(np\_2d[1:3,:])

结果：



np\_2d[:,1:3]代表行列相交，比如所有行和1:3（列索引为1,2，所有的索引都是从0开始的。）。

第1行的平均值：

print(np.mean(np\_2d[0]))

结果：



第1行的中位数：

print(np.median(np\_2d[0]))

结果：



1.73是数组中5个数字最中间那个。

numpy中还有很多函数，如corrcoef(),std(),sum(),sort()等等。

numpy中一个数组的所有数据都是同一类型，因此查询效率很高。

也可以随机产生5000个数字。

#随机生成  
height=np.round(np.random.normal(1.75,0.20,5000),2)

weight=np.round(np.random.normal(60.32,15,5000),2)

np\_city=np.column\_stack((height,weight))

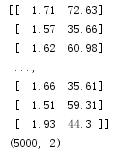
print(np\_city)

print(

np\_city.shape

)

结果：



column\_stack()函数代表着列拼接。

list

lst=['tom',1,2] #list

print(type(lst))

print(lst[-1])

print(len(lst))

classmates = ['Michael', 'Bob', 'Tracy']

for classmate in classmates:

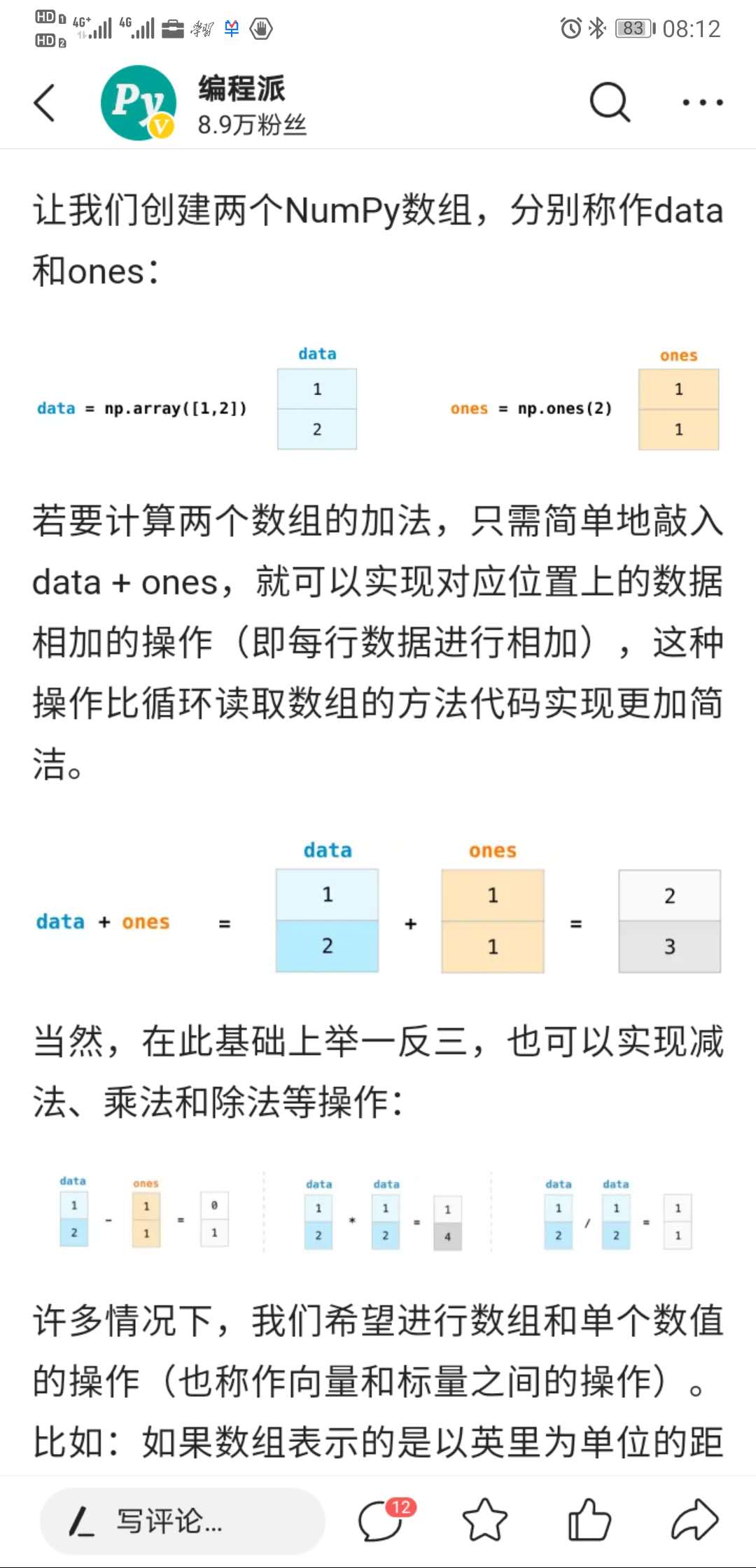
print(classmate)

for i in range(len(classmates)): #必须要加range

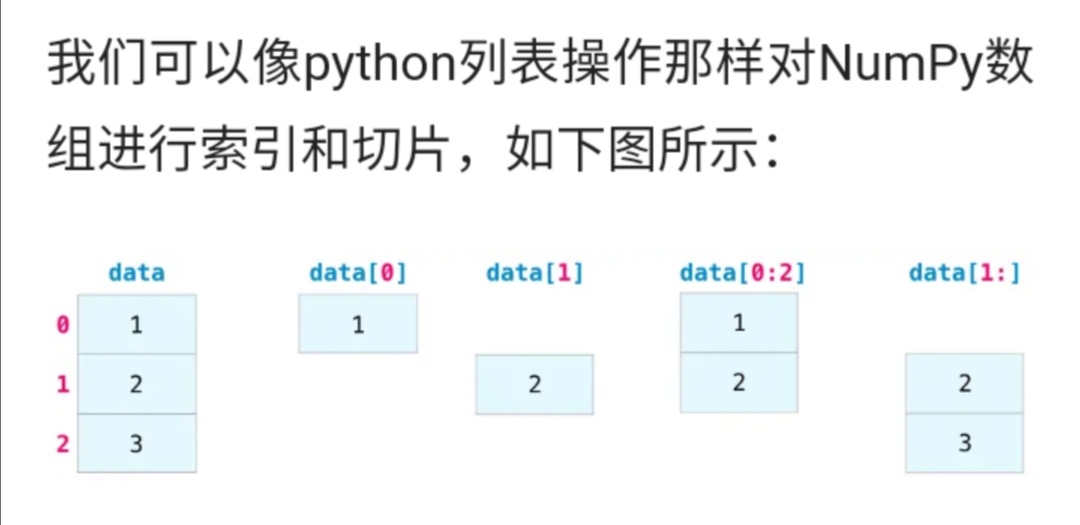
print('当前姓名:%s' % classmates[i])

必了解知识

通常情况下，我们希望NumPy为我们初始化数组的值，为此NumPy提供了诸如ones，zeros和random.random之类的方法。

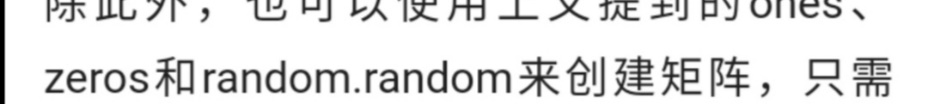


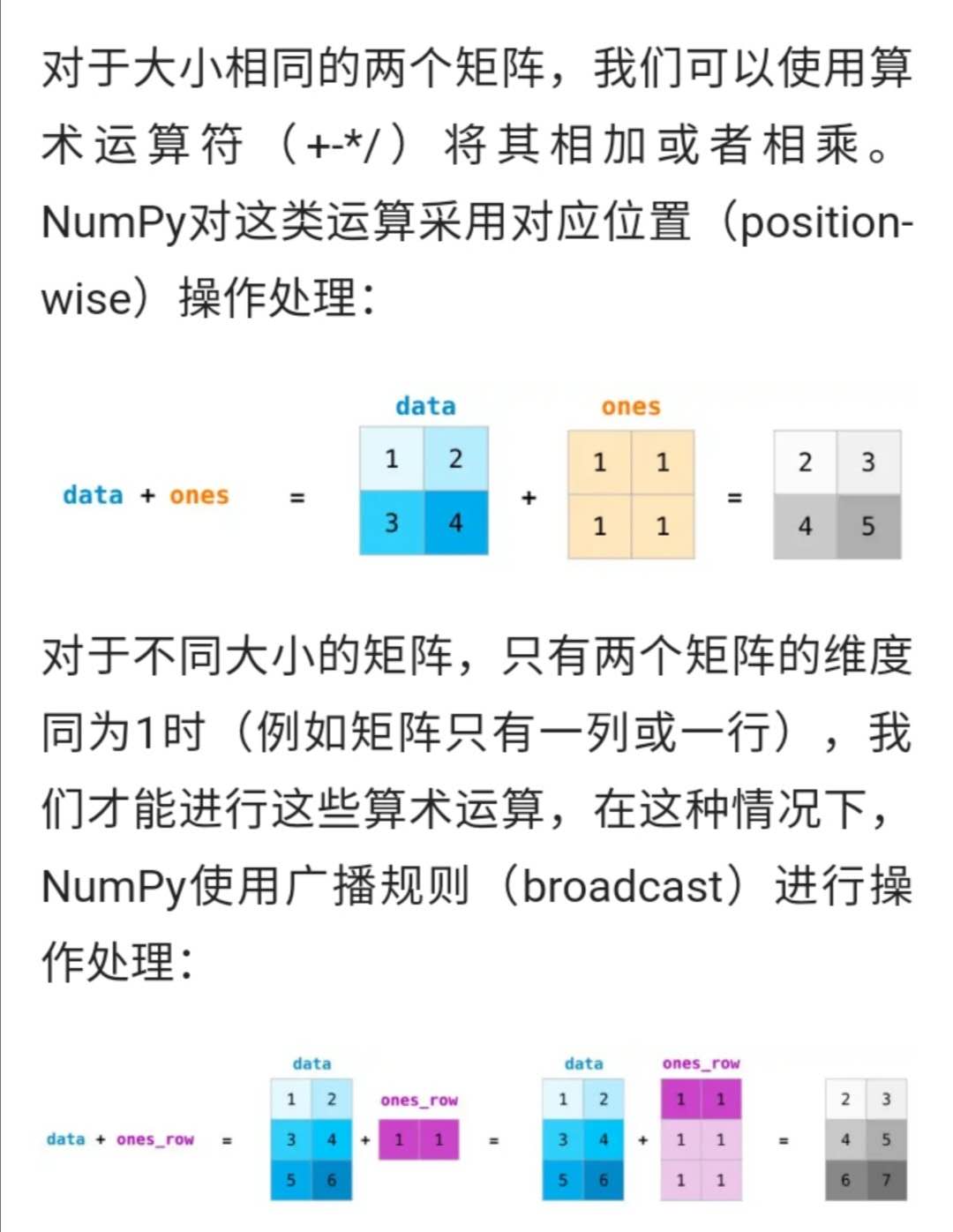
我们可以像python列表操作那样对NumPy数组进行索引和切片：



NumPy为我们带来的便利还有聚合函数，聚合函数可以将数据进行压缩，统计数组中的一些特征值：







scipy

pandas

pandas相当于python中excel：它使用表（也就是 dataframe)，能在数据上做各种变换，但还有其他很多功能。

数据分析的基本过程：<https://zhuanlan.zhihu.com/p/35976466>

① pandas一般解决表格型的数据、二维的。

② pandas是专门为处理表格和混杂数据设计的，而Numpy更适合处理统一数值数据。

③ pandas主要数据结构：Series 和 DataFrame

### 引用

**导入pandas模块并使用别名，以及导入Series模块：**

import pandas as pd

from pandas import Series #相当于using pandas.Series

### 数据结构

Pandas 数据结构就像是低维数据的容器。比如，DataFrame 是 Series 的容器，Series 则是标量的容器。使用这种方式，可以在容器中以字典的形式插入或删除对象。

### Series

Series 就如同列表一样，一系列数据，每个数据对应一个索引值。Series 就是“竖起来”的list。

s=Series([1,4,'w','a'])

print(s[1])

区别于列表的是，**Series 可以自定义索引**:

s = pd.Series([9, 'zheng', 'beijing', 128, 'usa', 990], index=[1,2,3,'e','f','g'])

print(s)

打印:

1          9

2      zheng

3    beijing

e        128

f        usa

g        990

dtype: object

根据索引找出值:

print(s['f'])

#Series可以看成是一个定长的有序字典

di={'apple':15,'double':1.52,'pen':'1.02'}

s=pd.Series(di)

di=[1,2,3,4]

s=pd.Series(di,index=['a','b','c','d'])

di={'apple':[15,1.52,20,'1.02']}

s=pd.Series(di)

print(s)

其他都正常，最后一个打印出来其实是：



### IO工具

pandas的I/O API是一组read函数，比如pandas.read\_csv()函数。这类函数可以返回pandas对象。

### 使用pandas读取excel

pd.read\_excel(io, sheet\_name=0, header=0, names=None, index\_col=None,usecols=None, squeeze=False,dtype=None, engine=None,converters=None, true\_values=None, false\_values=None,skiprows=None,nrows=None, na\_values=None, parse\_dates=False,date\_parser=None, thousands=None, comment=None, skipfooter=0,convert\_float=True, \*\*kwds)

pandas读取Excel后返回**DataFrame**，接下来我们就pd.read\_excel()的常用参数进行详细解析。

io，Excel的存储路径：建议使用英文路径以及英文命名方式。

sheet\_name，要读取的工作表名称

可以是整型数字、列表名或SheetN，也可以是上述三种组成的列表。

整型数字：目标sheet所在的位置，以0为起始，比如sheet\_name = 1代表第2个工作表。

1. data = pd.read\_excel(io, sheet\_name = 1)
2. data.head()
3. data = pd.read\_excel(io, sheet\_name = '英超射手榜')
4. data.head()
5. data = pd.read\_excel(io, sheet\_name = 'Sheet5')
6. data.head()

SheetN：代表第N个sheet，S要大写，注意与整型数字的区别。

其他的参数：使用pandas读取excel

<https://blog.csdn.net/weixin_38546295/article/details/83537558>

### 结果导回excel

data.to\_csv( my\_new\_file.csv , index=None)

index=None 表示将会以数据本来的样子写入。如果没有写 index=None，你会多出一个第一列，内容是 1，2，3，...，一直到最后一行。

### dataframe

.shape #有多少行，多少列

.dtypes #查看每列的数据类型

.head() #打印出前5行，以确保数据运行正常

.loc

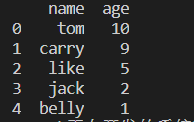
基础操作：

data={'name':['tom','carry','like','jack','belly'],'age':[10,9,5,2,1]}

df=pd.DataFrame(data)

print(df)

打印结果：

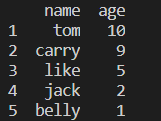


data=[['tom',10],['carry',9],['like',5],['jack',2],['belly',1]]

df=pd.DataFrame(data,columns=['name','age'],index=[1,2,3,4,5])

print(df)

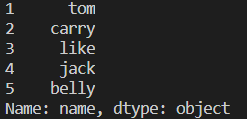
打印结果：



print(df['name'])

print(df.name)

打印结果：



print(df.loc[1:3,'name']) #打印1、2、3行name的值

print(df[df['name']=='carry']) #打印'name'=='carry'的行

print(df[df['age']>9]) #打印大于9岁的行

处理空值

**import** pandas **as** pd

df **=** pd**.**read\_csv('music.csv')

**print**(df**.**isnull())

如果我想知道哪列存在空值，可以使用 df.isnull().any()

**import** pandas **as** pd

df **=** pd**.**read\_csv('music.csv')

**print**(df**.**isnull()**.**any())

处理空值，Pandas 库提供很多方式。最简单的办法就是删除空值的行。



除此之外，还可以使用取其他数值的平均值，使用出现频率高的值进行填充缺失值。

**import** pandas **as** pd

*# 将值填充为 0*

pd**.**fillna(0)

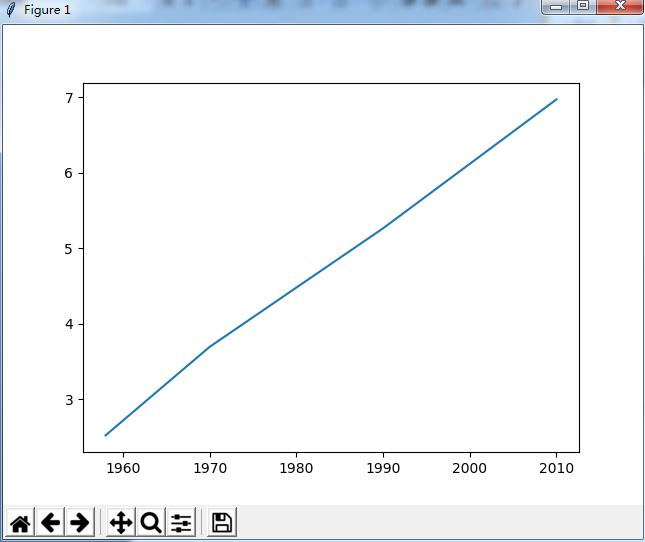
# 四、数据可视化

## matplotlib

使用时需要安装一下这个包。

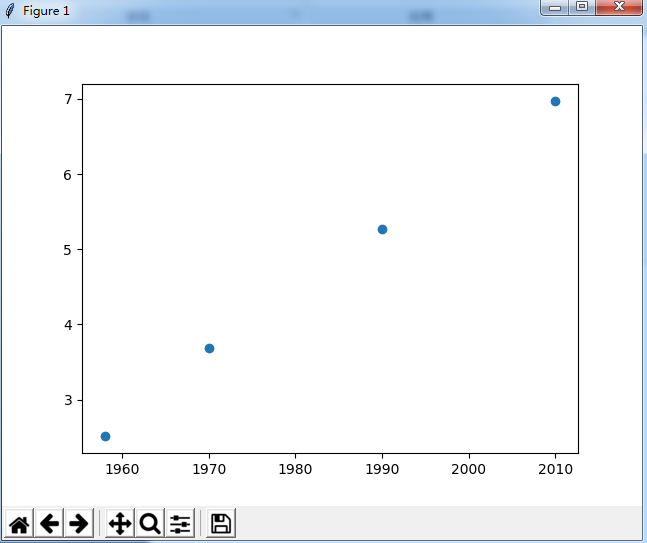
import matplotlib.pyplot as plt  
year=[1958,1970,1990,2010]  
pop=[2.519,3.692,5.263,6.972]  
plt.plot(year,pop)  
plt.show()

结果：



import matplotlib.pyplot as plt  
year=[1958,1970,1990,2010]  
pop=[2.519,3.692,5.263,6.972]  
'''  
plt.plot(year,pop)  
plt.show()  
'''  
plt.scatter(year,pop)  
plt.show()

结果：



‘’’三个引号代表多行注释。

图表还有很多，慢慢去研究。

# 五、爬虫

## scrapy

<https://www.jianshu.com/p/cecb29c04cd2>

<https://blog.csdn.net/ck784101777/article/details/104468780?utm_source=app>

https://c.cicdi.com/zkzs/web/newAppStore/#/BasicLayout/AppStore/AppList

Python开发的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据。Scrapy用途广泛，可以用于数据挖掘、监测和自动化测试。

创建项目：scrapy startproject xxx

进入项目：cd xxx #进入某个文件夹下

创建爬虫：scrapy genspider xxx（爬虫名） xxx.com （爬取域） 注：不能创建和项目同名的爬虫。

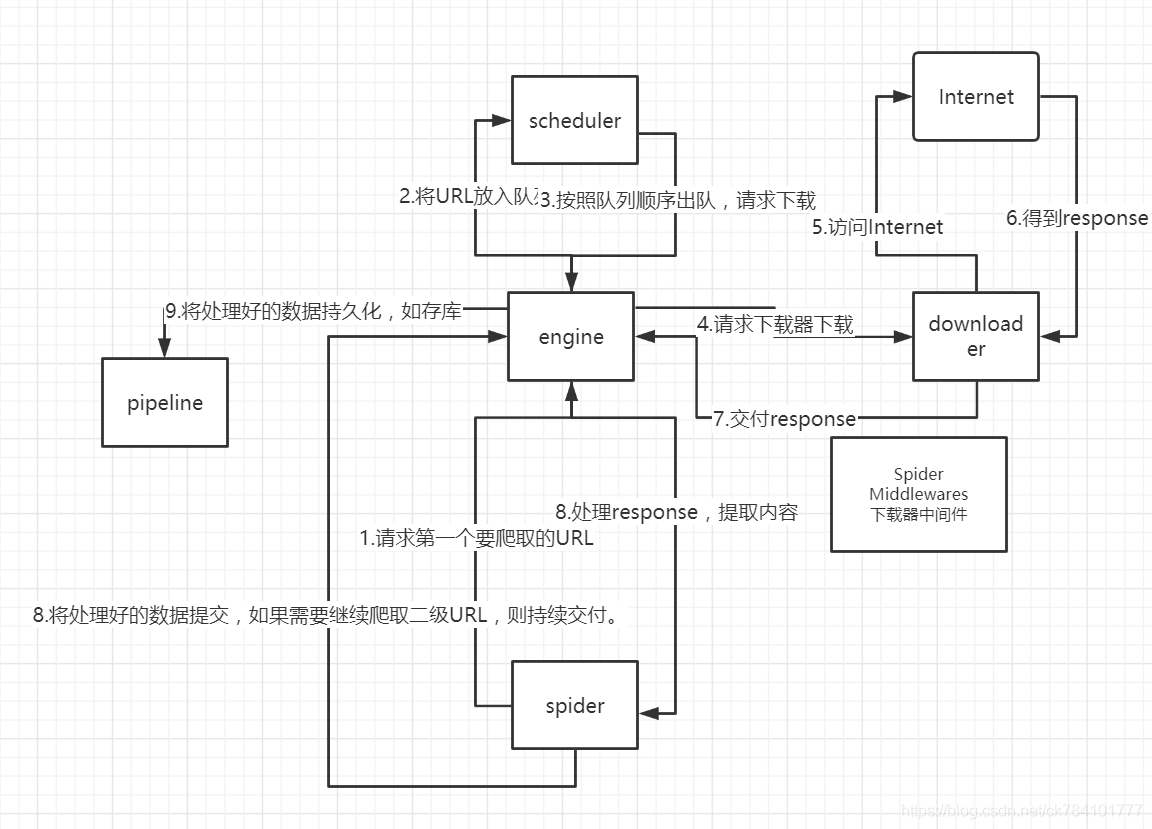
生成文件：scrapy crawl xxx -o xxx.json (生成某种类型的文件)

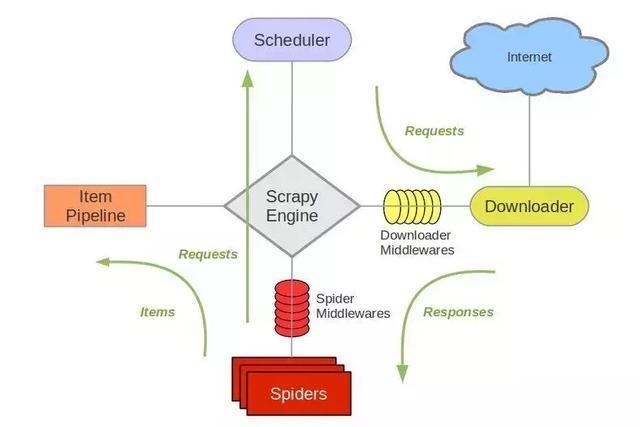
运行爬虫：scrapy crawl XXX

列出所有爬虫：scrapy list

获得配置信息：scrapy settings [options]

Scrapy框架主要由五大组件组成，它们分别是调度器(Scheduler)、下载器(Downloader)、爬虫（Spider）和实体管道(Item Pipeline)、Scrapy引擎(Scrapy Engine)。





创建后目录大致页如下

|-ProjectName              #项目文件夹

   |-ProjectName           #项目目录

      |-items.py               #定义数据结构

      |-middlewares.py    #中间件，用来存放各种中间的文件

      |-pipelines.py          #数据处理，用来将items的模型存储到本地磁盘中

      |-settings.py            #全局配置

      |-spiders

          |-\_\_init\_\_.py       #爬虫文件

          |-baidu.py

   |-scrapy.cfg               #项目基本配置文件

spdier项目的配置文件，打开文件settings.py

BOT\_NAME：项目名

USER\_AGENT：默认是注释的，这个东西非常重要，如果不写很容易被判断为电脑，简单点洗一个Mozilla/5.0即可

ROBOTSTXT\_OBEY：是否遵循机器人协议，默认是true，需要改为false，否则很多东西爬不了。

流程描述

  1、爬虫将需要发送的请求提交给引擎。 ​  
    2、引擎把从爬虫获取到的请求传递给调度器。 ​  
    3、调度器接收引擎发来的请求，并进行整理、排列、入队生成Request交还给引擎。 ​  
    4、引擎拿到从调度器中传递过来的Request，通过MIDDLEWARE进行层层过滤后发送给下载器。 ​  
    5、下载器去互联网中下载并获取Response数据，通过MIDDLEWARE层层过滤后将Response数据返回给引擎。 ​  
    6、引擎获取到下载器返回的Response数据传递给爬虫。 ​  
    7、爬虫获取到Response数据，通过parse()方法提取Items数据和Requests并返回给引擎。 ​  
    8、引擎接收到Items和Requests，将Items传递给Piplines进行分析过滤、存储，把Requests继续传给调度器，直到调度器不存在任何Requests了，整个程序停止，对于失败的URL也会进行重新下载。

### Scrapy五大基本构成:

Scrapy框架主要由五大组件组成，它们分别是调度器(Scheduler)、下载器(Downloader)、爬虫（Spider）和实体管道(Item Pipeline)、Scrapy引擎(Scrapy Engine)。下面我们分别介绍各个组件的作用。

**(1)、调度器(Scheduler):**

调度器，说白了把它假设成为一个URL（抓取网页的网址或者说是链接）的优先队列，由它来决定下一个要抓取的网址是 什么，同时去除重复的网址（不做无用功）。用户可以自己的需求定制调度器。

**(2)、下载器(Downloader):**

下载器，是所有组件中负担最大的，它用于高速地下载网络上的资源。Scrapy的下载器代码不会太复杂，但效率高，主要的原因是Scrapy下载器是建立在twisted这个高效的异步模型上的(其实整个框架都在建立在这个模型上的)。

**(3)、 爬虫（Spider）:**

爬虫，是用户最关心的部份。用户定制自己的爬虫(通过定制正则表达式等语法)，用于从特定的网页中提取自己需要的信息，即所谓的实体(Item)。 用户也可以从中提取出链接,让Scrapy继续抓取下一个页面。

**(4)、 实体管道(Item Pipeline):**

实体管道，用于处理爬虫(spider)提取的实体。主要的功能是持久化实体、验证实体的有效性、清除不需要的信息。

**(5)、Scrapy引擎(Scrapy Engine):**

Scrapy引擎是整个框架的核心.它用来控制调试器、下载器、爬虫。实际上，引擎相当于计算机的CPU,它控制着整个流程。

### spdier项目的配置文件

打开文件settings.py：

BOT\_NAME：项目名

USER\_AGENT：默认是注释的，这个东西非常重要，如果不写很容易被判断为电脑，简单点洗一个Mozilla/5.0即可

ROBOTSTXT\_OBEY：是否遵循机器人协议，默认是true，需要改为false，否则很多东西爬不了

CONCURRENT\_REQUESTS：最大并发数，很好理解，就是同时允许开启多少个爬虫线程

DOWNLOAD\_DELAY：下载延迟时间，单位是秒，控制爬虫爬取的频率，根据你的项目调整，不要太快也不要太慢，默认是3秒，即爬一个停3秒，设置为1秒性价比较高，如果要爬取的文件较多，写零点几秒也行

COOKIES\_ENABLED：是否保存COOKIES，默认关闭，开机可以记录爬取过程中的COKIE，非常好用的一个参数

DEFAULT\_REQUEST\_HEADERS：默认请求头，上面写了一个USER\_AGENT，其实这个东西就是放在请求头里面的，这个东西可以根据你爬取的内容做相应设置。

ITEM\_PIPELINES：项目管道，300为优先级，越低越爬取的优先度越高。

### 爬取

到这里我们尝试用scrapy做一下爬取，打开spider.py下的baidu.py(取决于你scrapy genspider 爬虫名域名时输入的爬虫名）

输入一下代码，我们使用xpath提取百度首页的标题title

import scrapy

class BaiduSpider(scrapy.Spider):

name = 'baidu'

allowed\_domains = ['www.baidu.com']

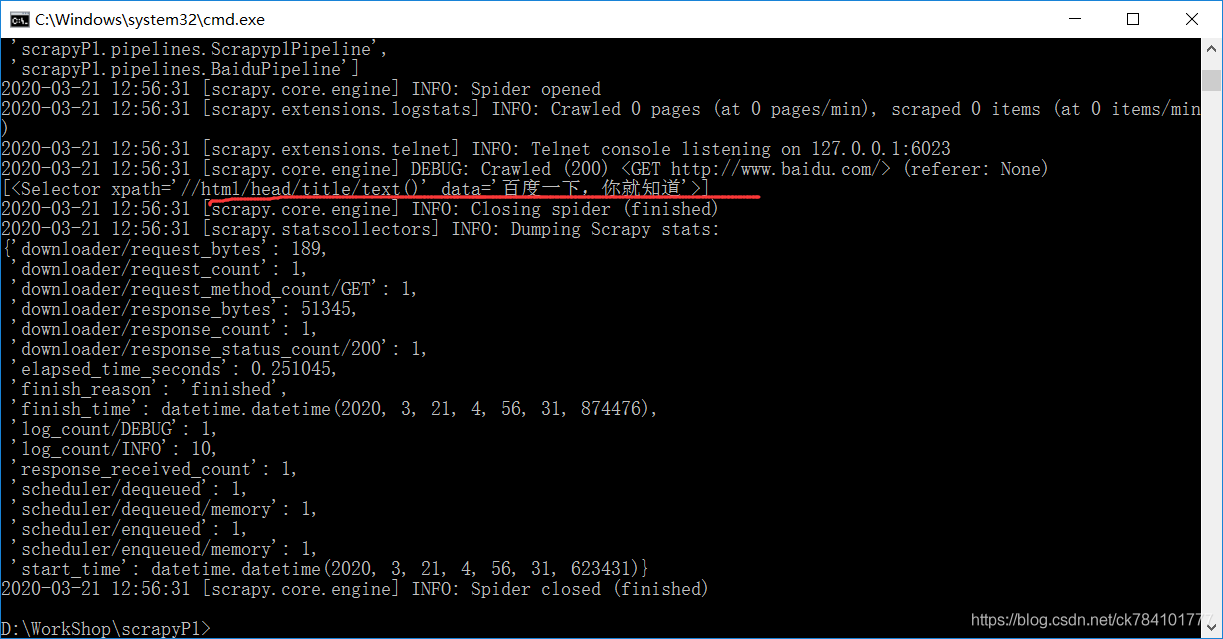
start\_urls = ['http://www.baidu.com/']

def parse(self, response):

tile=response.xpath('//html/head/title/text()')

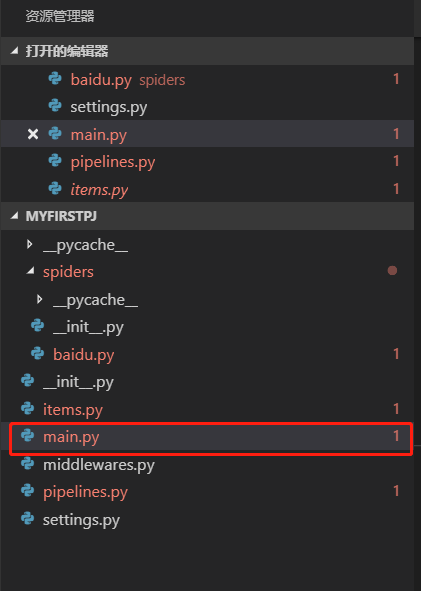
print(tile)

打开一个终端cmd，输入scrapy crawl baidu(爬虫名），就可以看到一大堆输出信息，而其中就包括我们要的内容：



### 爬虫入口

使用终端运行太麻烦了，而且不能提取数据，我们一个写一个run文件作为程序的入口,split是必须写的，目的是把字符串转为列表形式，第一个参数是scrapy,第二个crawl,第三个baidu。



from scrapy import cmdline

cmdline.execute('scrapy crawl baidu'.split())

### 注意事项

项目和爬虫名不能一样，否则会有提示错误如：scrapy No module named 'ancient\_poetry'

### 例子:vgirls

<https://zhuanlan.zhihu.com/p/80368091>



### scrapy框架中多个spider,tiems,pipelines的使用及运行方法

<https://www.cnblogs.com/nmsghgnv/p/12369656.html>

## urllib

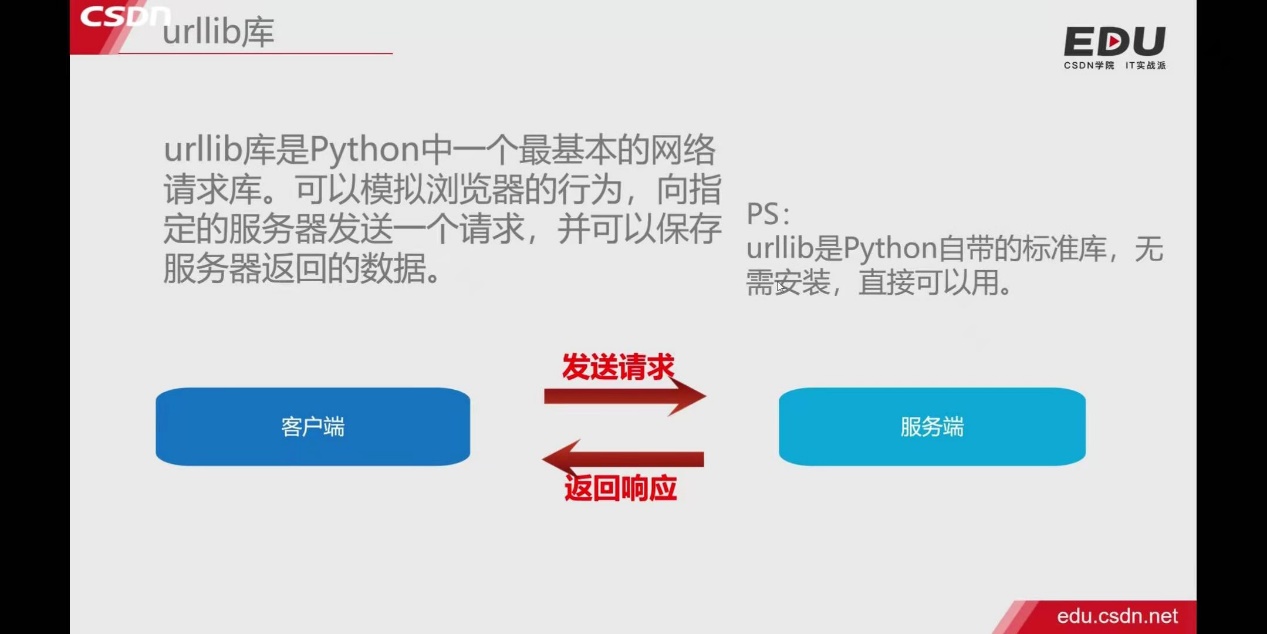
python3中将urllib和urllib2合并，且自带了，不需要安装。而urllib又拆分成urllib.request、urllib.parse、urllib.error,其中urllib2.urlopen()变urllib.request.urlopen()，urllib2.Request()变urllib.request.Request()。

简单例子

import urllib.request  
def getHtml(url):  
 page=urllib.request.urlopen(url)  
 html=page.read()  
 return html  
  
html=getHtml("http://www.baidu.com")  
print(html)

其中def getHtml(url):指的是声明一个名称为getHtml的函数，参数为url,所以才有后面的html=getHtml("http://www.baidu.com")。

def getHtml(url):  
 page=urllib.request.urlopen(url)  
 html=page.read()  
 return html

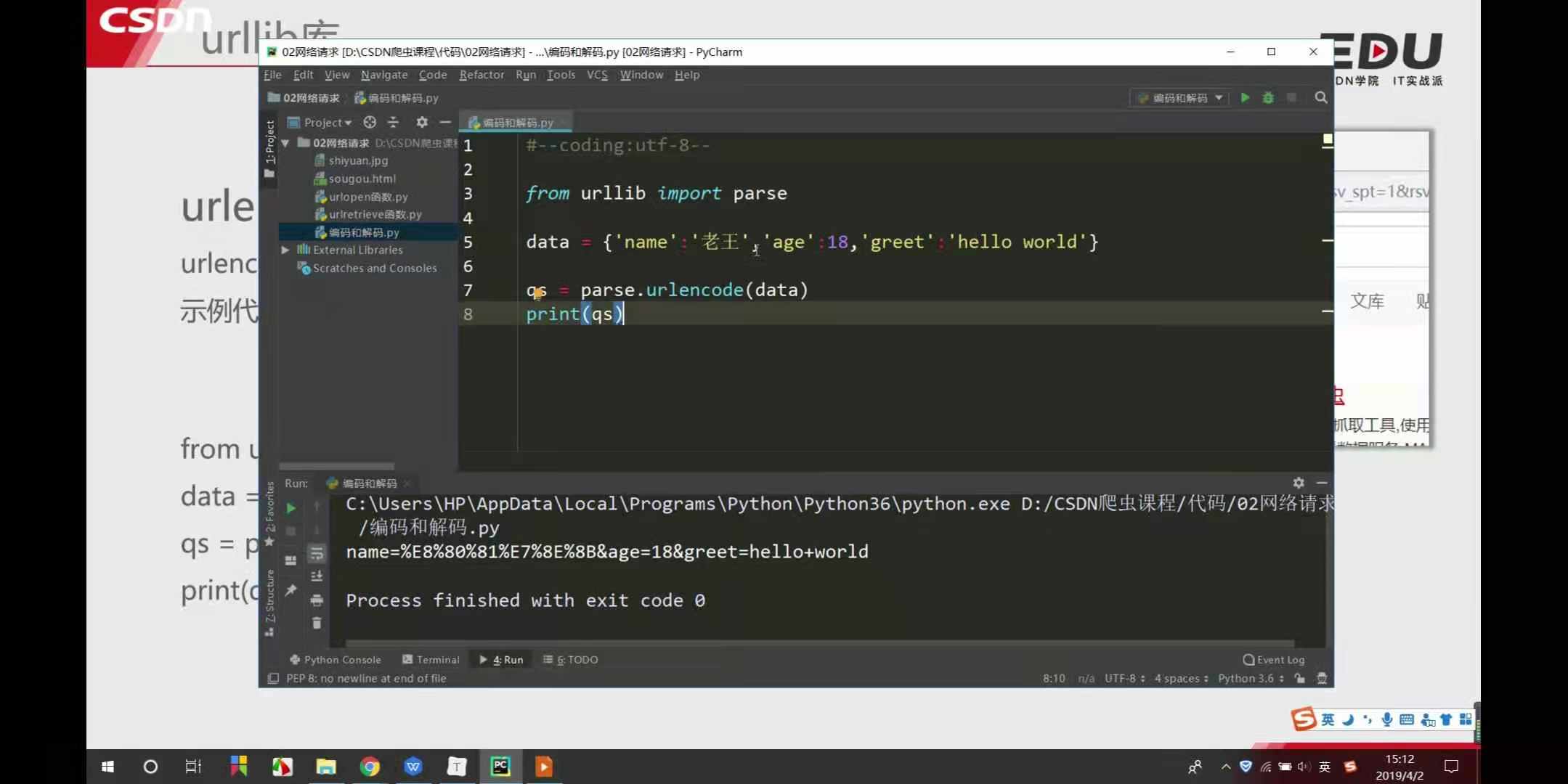


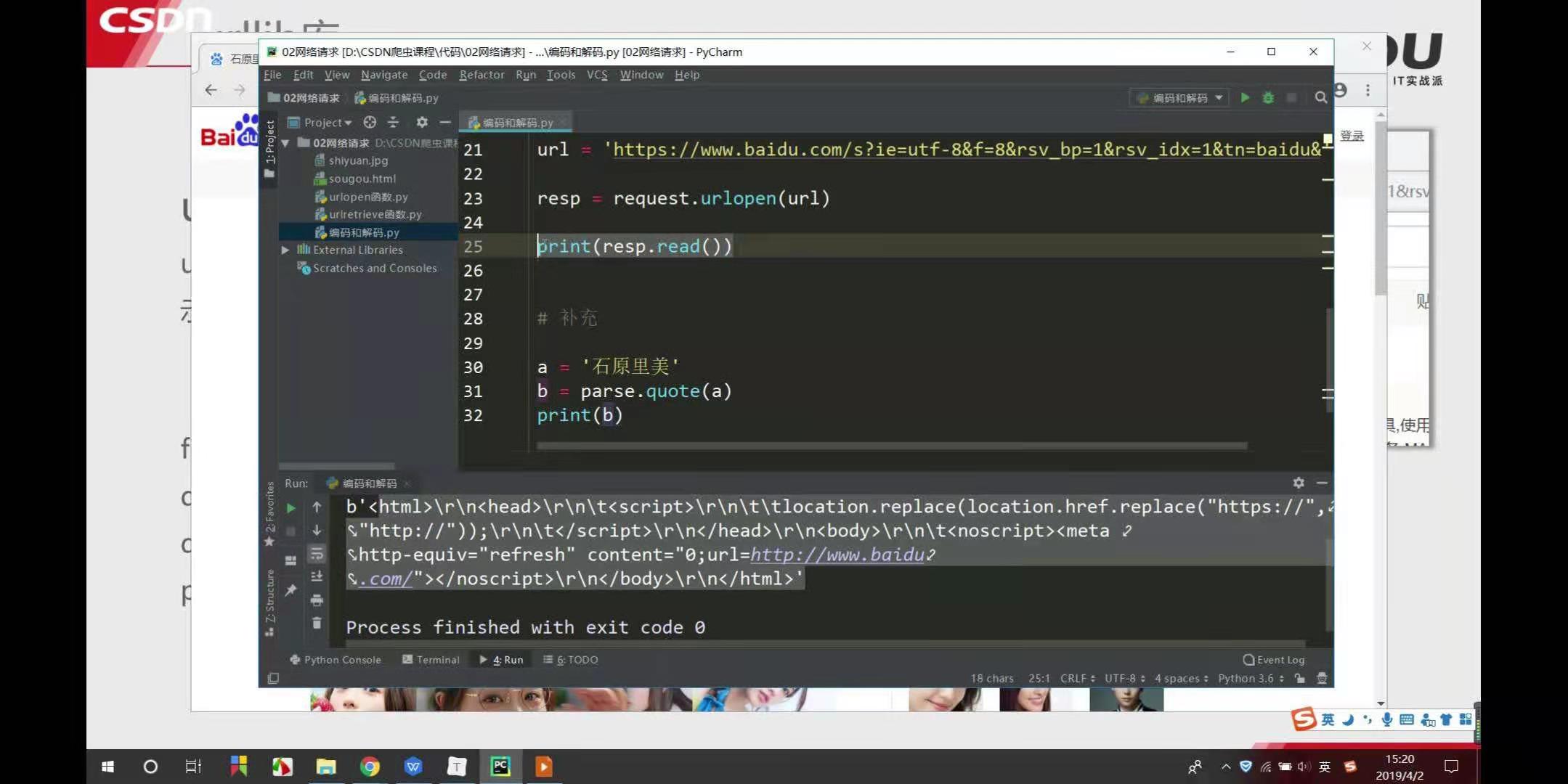








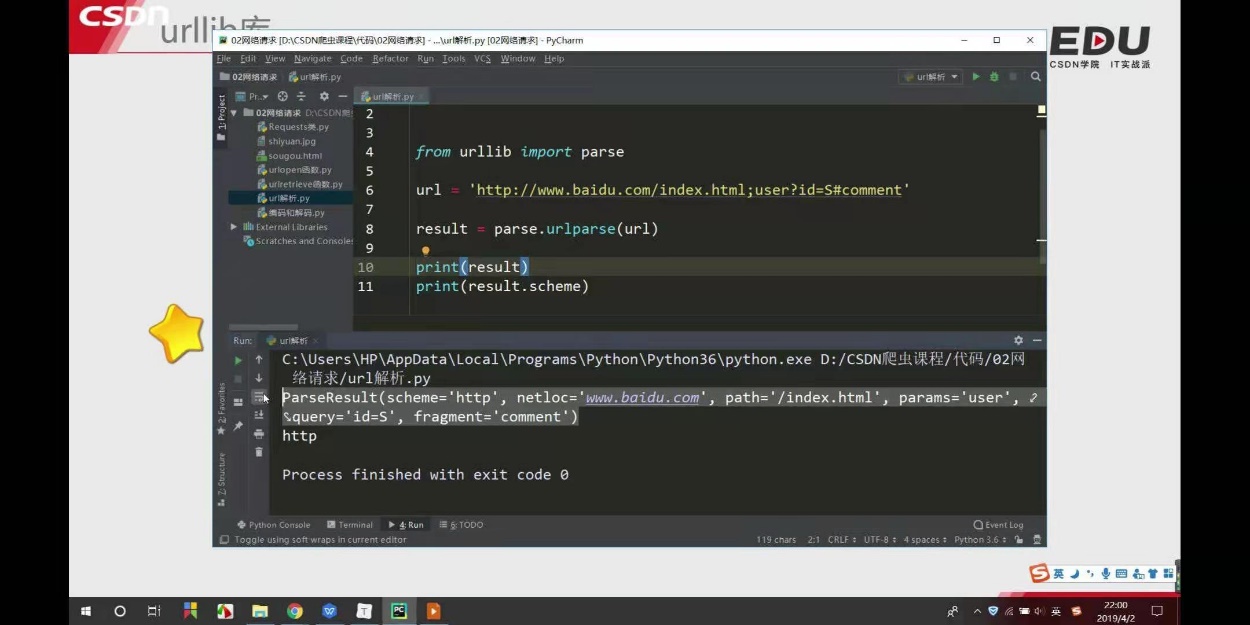




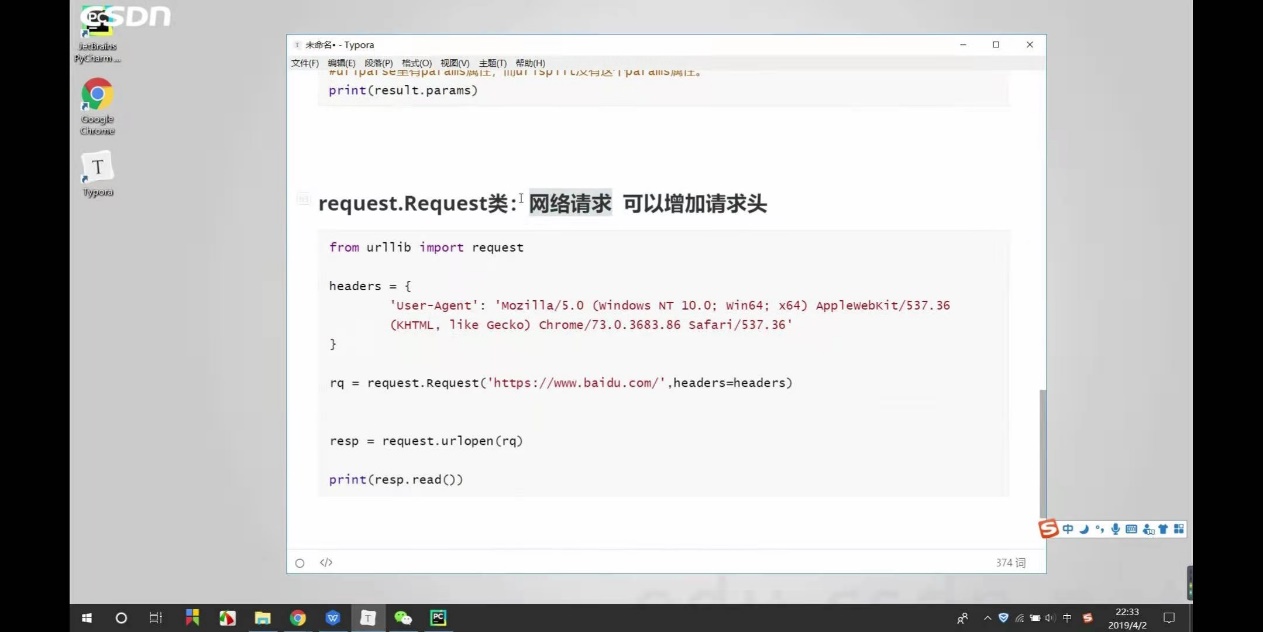
urlencode



parse和split



Request



## 第三方库

可以使用第三方的库来实现爬虫功能，下面介绍两个库：

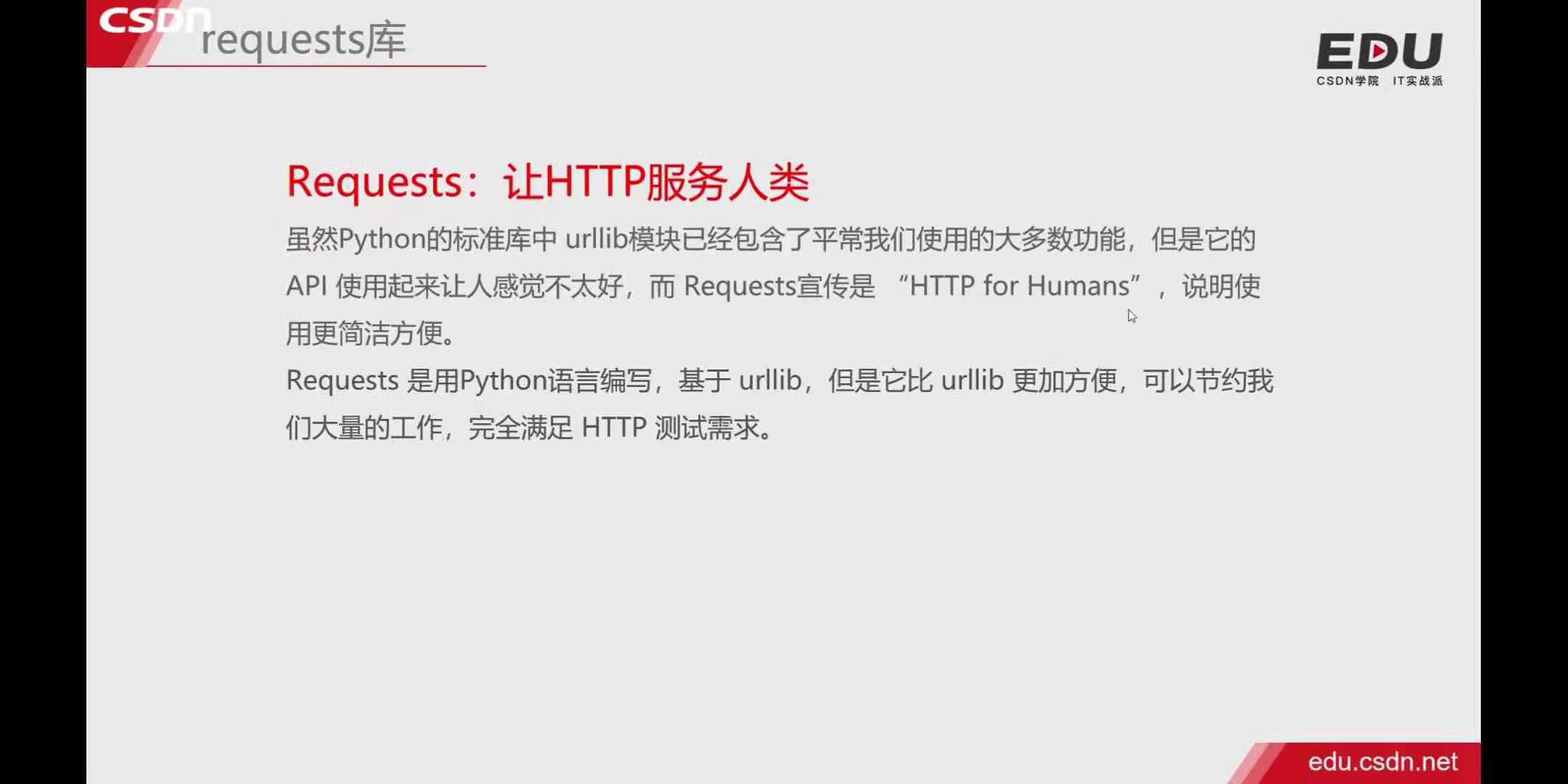
requests:解析url ；

beautifulsoup4:不会正则表达式童鞋的福音，可以容易的提取到html文件中各种标签及其属性。

安装方法：

pip3 install requests

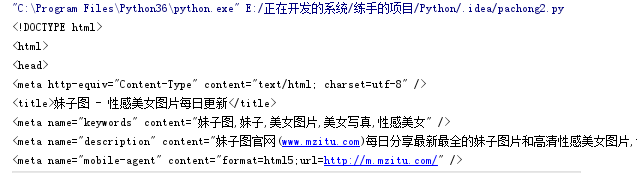
pip3 install beautifulsoup4



使用方法：

import requests  
url="http://mzitu.com"  
#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问  
header = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36'}  
#用get方法打开url并发送headers  
html=requests.get(url,headers=header)  
print(html.text)

得到的结果比较规整：



如果需要进一步解析，可以：

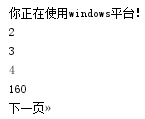
import requests  
import bs4  
url="http://www.mzitu.com/26685"  
#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问  
header = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36'}  
#用get方法打开url并发送headers  
html=requests.get(url,headers=header)  
soup=bs4.BeautifulSoup(html.text,'html.parser')

或者：

import requests  
from bs4 import BeautifulSoup  
url="http://www.mzitu.com/26685"  
#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问  
header = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36'}  
#用get方法打开url并发送headers  
html=requests.get(url,headers=header)  
soup=BeautifulSoup(html.text,'html.parser')

获取标签：

def find\_MaxPage():  
 all\_url="http://www.mzitu.com"  
 start\_html = requests.get(all\_url, headers=header) #header在main中被定义过  
 soup=BeautifulSoup(start\_html.text,'html.parser')  
 page=soup.find\_all('a',class\_='page-numbers')  
 for page\_index in page:  
 print(page\_index.text)

****

如果不用text，取到的是整个标签：

def find\_MaxPage():  
 all\_url="http://www.mzitu.com"  
 start\_html = requests.get(all\_url, headers=header) #header在main中被定义过  
 soup=BeautifulSoup(start\_html.text,'html.parser')  
 page=soup.find\_all('a',class\_='page-numbers')  
 max\_page=page[-2];  
 print(max\_page)

****

requests库的一些用法可见<http://www.jianshu.com/p/e1f8b690b951>。

## requests

### 访问后台

url="https://c.cicdi.com/zkzs/webtest/smartDesign/wx/back/operation/templateManage/list"

header={

'appId': '10910001',

'appKey': 'SlY7X3w5',

'agentId': '10910039',

'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36',

}

html=requests.get(url,headers=header,params={'modelId':'0a6d6a3358af44fbbb11ae8377b55976'})

data=json.loads(html.text)

print(data['retCode']['status'])

<https://www.cnblogs.com/yuany66/p/10785589.html>

### Post

## 爬取meizitu上美图的完整代码(防盗链，多线程)

import requests

from bs4 import BeautifulSoup

import os

import sys

from multiprocessing import Pool #多线程

'''

#安卓

reload(sys)

sys.setdefaultencoding('utf-8')

'''

def find\_MaxPage():

all\_url="http://www.mzitu.com"

start\_html = requests.get(all\_url, headers=header) #header在main中被定义过

soup=BeautifulSoup(start\_html.text,'html.parser')

page=soup.find\_all('a',class\_='page-numbers')

max\_page=page[-2].text;

return max\_page

def funcvar(x): return x + 1

def download\_Pic(href,header,title,path):

single\_html=requests.get(href,headers=header)

soup=BeautifulSoup(single\_html.text,'html.parser')

div\_content=soup.find('div',class\_='content')

div\_pagenavi=soup.find('div',class\_='pagenavi')

all\_navipages=div\_pagenavi.find\_all('span')

max\_navipage=all\_navipages[-2].text

single\_path=path+title.strip().replace('?','')

if (os.path.exists(single\_path)):

print("[" + title + "]已存在！")

return

print("开始扒取[" + title + "]，共" + max\_navipage + "张......")

# 创建文件夹

os.makedirs(single\_path)

os.chdir(single\_path)

for i in range(1, int(max\_navipage) + 1):

t\_url = href + '/' + str(i)

t\_html = requests.get(t\_url, headers=header)

soup = BeautifulSoup(t\_html.text, 'html.parser')

main\_image = soup.find('div', class\_='main-image')

img = main\_image.find('img', alt=title)

pic\_html = requests.get(img['src'], headers=header)

file\_name = img['src'].split(r'/')[-1]

f = open(file\_name, 'wb')

f.write(pic\_html.content)

f.close()

print("[" + title + "]已完成！")

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

if(os.name=='nt'):

print('你正在使用windows平台！')

else:

print('你正在使用linux平台！')

#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问

header = {"Accept": "text/html,application/xhtml+xml,application/xml;",

"Accept-Encoding": "gzip",

"Accept-Language": "zh-CN,zh;q=0.8",

"Referer": "http://www.mzitu.com/",

"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36"

}

#保存地址

path="D:/mzitu/"

# 找寻最大页数

max\_page=find\_MaxPage()

# http请求头

same\_url = "http://www.mzitu.com/page/"

pool=Pool(5)

for i in range(1,int(max\_page)+1):

page\_url = same\_url + str(i)

page\_html = requests.get(page\_url, headers=header)

soup = BeautifulSoup(page\_html.text, 'html.parser')

all\_postlist = soup.find('div', class\_='postlist')

all\_a = all\_postlist.find\_all('a', target='\_blank')

for a in all\_a:

title = a.get\_text()

if (title != ''):

#print('正在下载[%s]数据......' % title)

href=a['href']

pool.apply\_async(download\_Pic,args=(href,header,title,path))

pool.close()

pool.join()

print('操作完毕！')

#start\_html = requests.get(same\_url, headers=header)

## 防盗链

作为普通的网民来说，一般不需要知道也不用关心什么是盗链，不过如果你是网站的开发者或维护者，就不得不重视盗链的问题了。如果你刚刚开发完一个没有防盗链的带有文件下载功能的网站，挂上internet，然后上传几个时下非常热门的软件或电影并在网站内公布下载地址，让MSN上的所有好友都来体验一下你的杰作。

不用多久就会发现网速出奇地变慢，从web服务器的访问日志里会发现疯狂的访问请求正从四面八方涌过来，web服务器为了迎接这批访客而没有时间处理正常的页面，这种状况可能会一直持续好几个周时间。

网站资源被盗链简单来说就是别人不是从你的网站通过下载资源，有哪些方法可以防止盗链呢：

方法1：判断引用地址

这个方法是最早及最常见的方法。所谓判断引用地址，就是判断浏览器请求时HTTP头的Referer字段的值，这个值在asp.net里面可以用 Request.UrlReferrer属性取得。几个例子来说，在正常情况下当用户在浏览 http://uushare.com/abc.html 时点击一个链接去到 http://uushare.com/jacky.mp3 文件时，浏览器在发出请求jacky.mp3 资源时还会附带当刻浏览器所处的页面地址（即http://uushare.com/abc.html），所以当你的网站程序接收到下载 jacky.mp3 资源请求的时候，先判断http的referer字段的值，如果是从自己的域名（uushare.com）过来的，则可以认为是合法的连接请求，否则就返回一个错误的提示信息。上例中的meizitu就是通过在request的head中添加"Referer": <http://www.mzitu.com/>来解决的，如：

header = {"Accept": "text/html,application/xhtml+xml,application/xml;",  
 "Accept-Encoding": "gzip",  
 "Accept-Language": "zh-CN,zh;q=0.8",  
 "Referer": "http://www.mzitu.com/",  
 "User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36"  
 }

这种方法通常用于图片、 mp3这种容易被人用html“嵌入”到其他网站的资源，使用这种方法可以防止你的图片直接出现在别人的网页里（或者防止mp3直接被其他网站嵌入到 flash播放器里），不过访客使用下载工具还是可以轻松下载，因为现在的下载工具一般会自动用你的域名构造一个引用地址，所以如果想再进一步防范的话，可以使用一个对应表限制每个资源的引用地址，例如将 jacky.mp3 的引用地址限制为 http://uushare.com/abc.htmlid=12345，这样下载工具就不太可能构造一个“正确”的引用地址了。

方法2：使用登录验证

这个方法常见于论坛、社区。当访客请求网站上的一个资源时，先判断此请求是否通过登录验证（在asp.net里常用session或form验证来记录登录状态），如果尚未登录则返回一个错误提示信息。使用这个方法还可以进一步判断登录的用户的权限是否足够，以实现带“权限”的下载。

方法3：使用cookie

其实这种方法原理上跟方法2差不多。就是在显示“下载”链接的页面里产生一个动态值的cookie，然后在处理资源下载请求时先判断cookie里有没有正确的cookie，如果没有则返回错误提示信息。至于这个动态值如何产生，只要能逆向判断动态值是否合法的都可以，例如将当前的时间去除秒数取哈希值（也叫散列值）。如果网页程序是asp.net则更简单，可以往Session里随便存一个字符串或数字，然后在处理下载请求时先检查Session 里是否存在这个字符串或数字。使用这个方法的缺点跟方法2一样。

方法4：使用POST下载

客户端浏览器请求资源都是使用HTTP的GET方法的，其实使用POST方法也可以往客户端返回数据。所以可以将下载链接换成一个表单（Form）和一个按钮(Submit)，将待下载的文件的名称或id放到表单的一个隐藏文本框（Input）里，当用户点击提交按钮时，服务程序先判断请求是否为 POST方式，如果是则读取目标资源的二进制数据并写入响应对象（在asp.net里是respone.BinaryWrite方法）。

当然还有其他的方法，就不一一介绍了，感兴趣可看：<http://blog.csdn.net/linshichen/article/details/51679074>，这里不再详述。

<http://blog.csdn.net/baidu_35085676/article/details/68958267>的后面一段也举了防盗图的解决办法。

## XPath

在 XML 文档中查找信息的语言。

<https://www.runoob.com/xpath/xpath-tutorial.html>

在 XPath 中，有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档（根）节点。XML 文档是被作为节点树来对待的。树的根被称为文档节点或者根节点。

请看下面这个 XML 文档：

<?xml version="1.0" encoding="UTF-8"?>  
  
<bookstore>  
  <book>  
    <title lang="en">Harry Potter</title>  
    <author>J K. Rowling</author>  
    <year>2005</year>  
    <price>29.99</price>  
  </book>  
</bookstore>

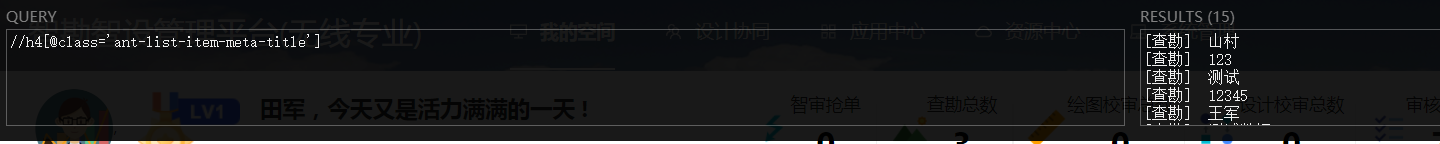
上面的XML文档中的节点例子：

<bookstore> (文档节点)  
  
<author>J K. Rowling</author> (元素节点)  
  
lang="en" (属性节点)

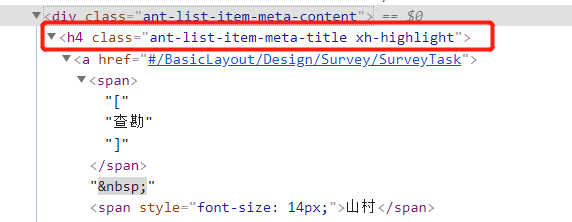
@\* 匹配任何属性节点。

//title[@\*] 选取所有带有属性的 title 元素。

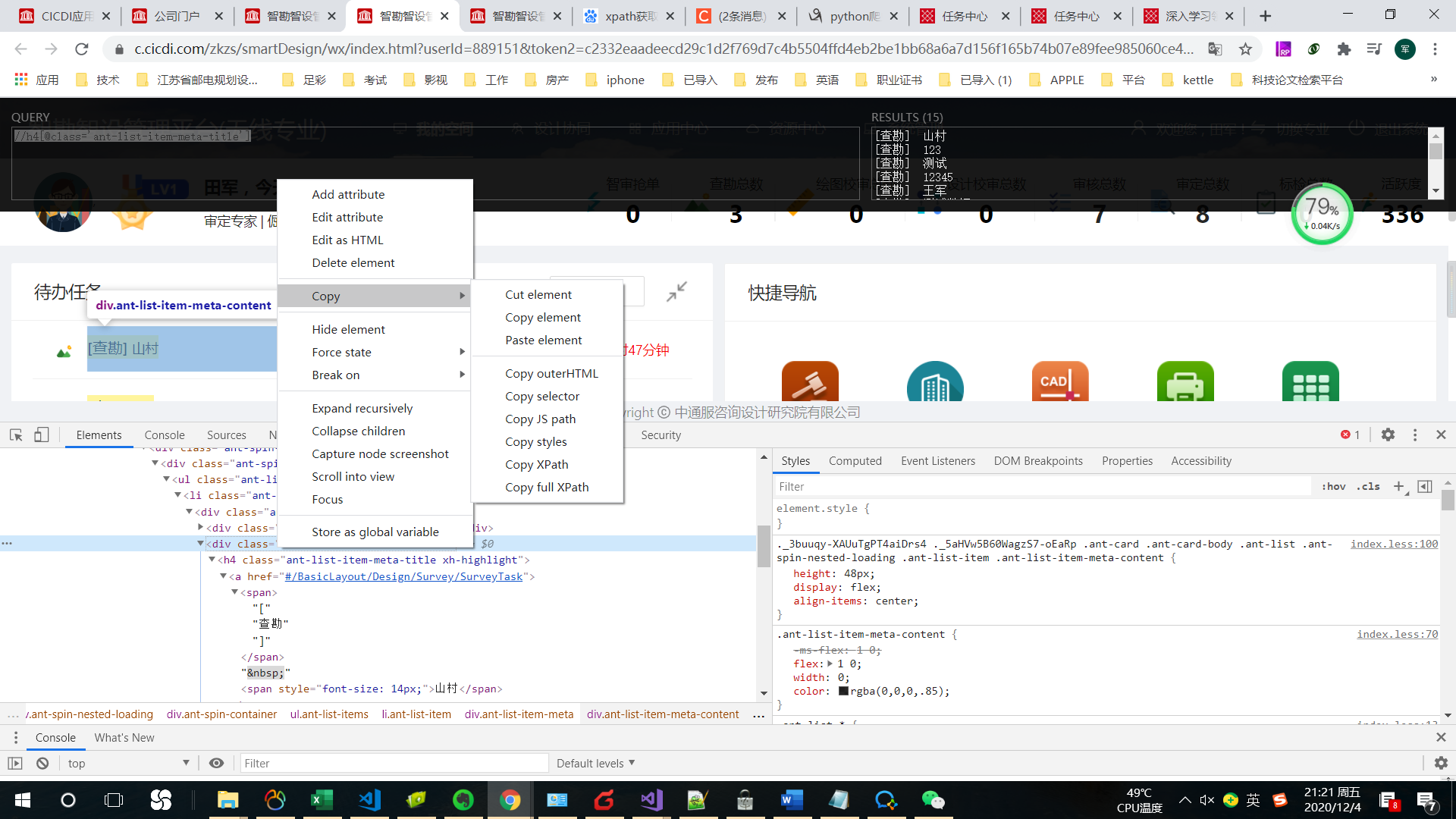
个人感觉：//搜索匹配元素，@匹配属性



//h4[@class='ant-list-item-meta-title']



如何从chrome中获取XPath:



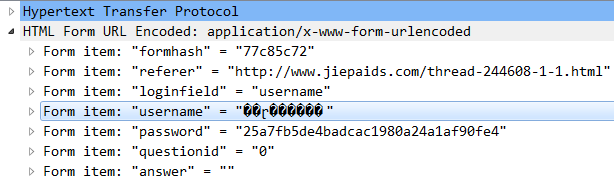
Copy XPath，但不如自己写的好用。

FireFox:Try XPath

## 抓包

有时，有的网站并不是简单地就可以抓取，而是需要登录后才能操作，如何获取登录时需要的参数呢，这时我们就需要使用抓包工具来进行了。

抓包工具有很多，小到最常用的web调试工具firebug，还有fiddler，大到通用的强大的抓包工具wireshark。我们使用的是wireshark：



应该可以看出参数了。

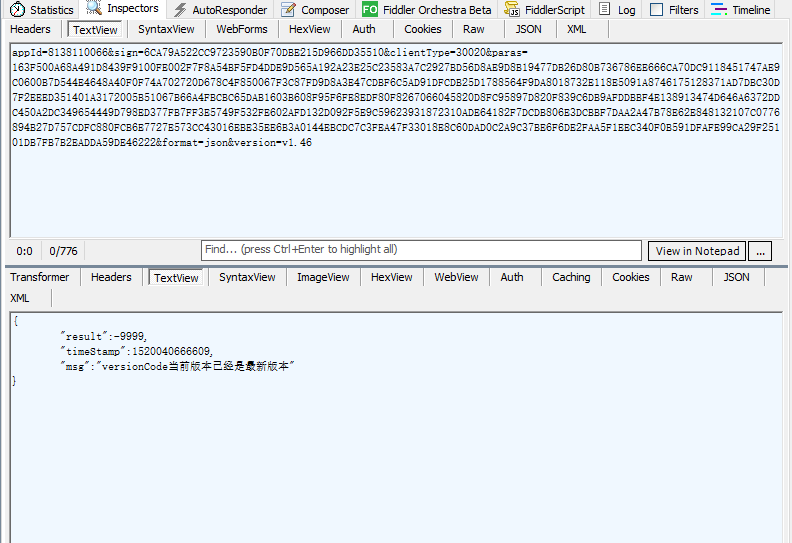
我用基金的试了一下：



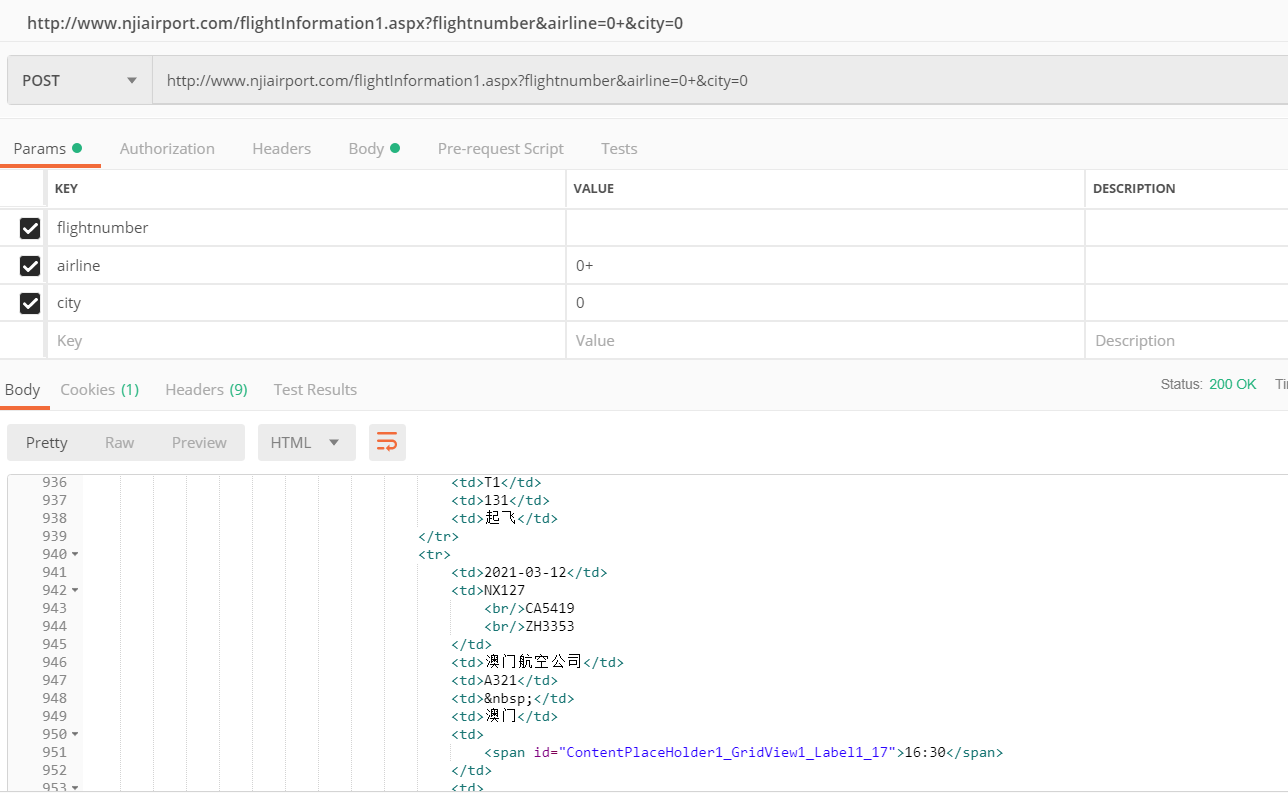
“帮助”-“wiki”：可以打开Wireshark wiki。

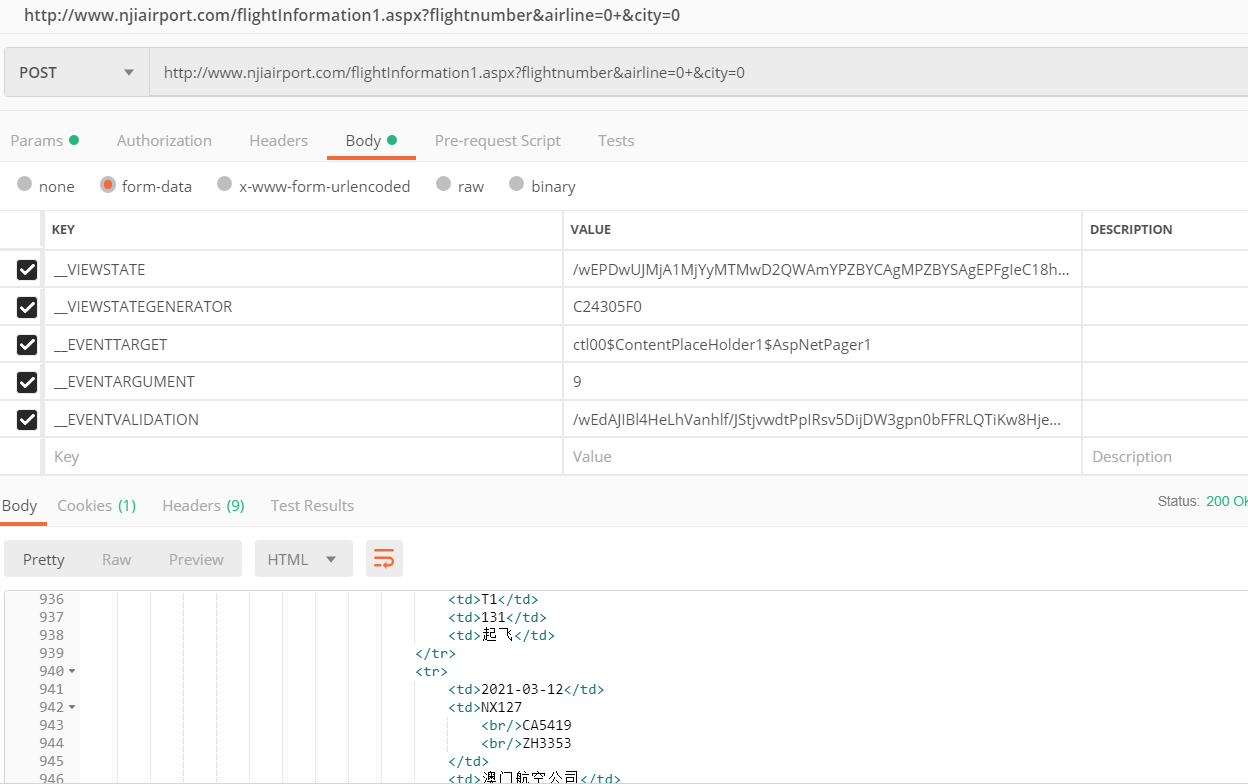
## Fiddle

选中请求，在右边的Inspectors窗口中可以看到这个请求的请求报文和响应报文信息。

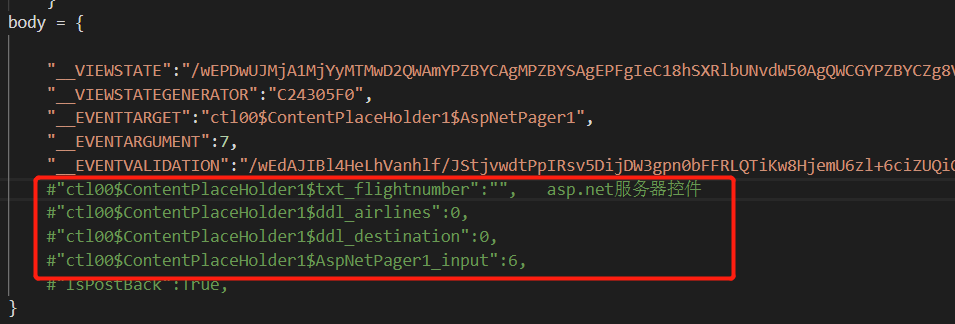


## Asp.net服务器控件





ASP.NET服务器控件有IspostBack属性，如果为true，加上下图的参数书，就会报ArgumentException]: Invalid postback or callback argument.错误。



# 六、运维

## 获取操作系统信息

import os

import sys

if(os.name=='nt'):  
 print('你正在使用windows平台！')  
else:  
 print('你正在使用linux平台！')

## os

### os.path.expanduser

在linux下面，一般如果你自己使用系统的时候，是可以用~来代表＂/home/你的名字/＂这个路径的．但是python是不认识~这个符号的，如果你写路径的时候直接写＂~/balabala＂，程序是跑不动的．所以如果你要用~，你就应该用这个os.path.expanduser把~展开．

path=os.path.expanduser('~')

linux系统path为"/home/newt"

windows系统path为"C:\Users\newt"

其中newt为我的用户名

### 系统盘

os.path.join(os.path.expanduser("~"), 'Desktop')

# 七、HTTP访问和接口测试

## get api

url="http://58.240.86.44:23588/smartDesign/"

page\_html = requests.get(url+"api/dic/list")

print(page\_html.text)

或：

#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问

header = {

"Content-Type":"application/json;charset=UTF-8",

"X-Requested-With" : "XMLHttpRequest",

"Connection" : "keep-alive",

"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36"

}

url="http://58.240.86.44:23588/smartDesign/"

#print(print\_diclist(url,"api/dic/list"))

page\_html = requests.get(url+"api/dic/list",headers = header)

print(page\_html.text)

最终整合成：

def get\_apiget(url,route,params):

#设置headers，网站会根据这个判断你的浏览器及操作系统，很多网站没有此信息将拒绝你访问

header = {

"Content-Type":"application/json;charset=UTF-8",

"X-Requested-With" : "XMLHttpRequest",

"Connection" : "keep-alive",

"User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 UBrowser/6.1.2107.204 Safari/537.36"

}

page\_html = requests.get(url+route,params=params,headers = header)

print(page\_html.text)

## get请求

1. 没有请求参数类型

response = requests.get(url='')

print(response.text)

1. 有请求参数的类型（键值对形式表示参数）

response = requests.get(url='',params={'key1':'value1','key2':'value2'})

print(response.text)

（3）有请求头（键值对形式表示请求头）

response = requests.get(url='',headers={'key1':'value1'})

print(response.text)

## post请求

1. 请求正文是application/x-www-form-urlencoded

requests.post(url='',data={'key1':'value1','key2':'value2'},headers={'Content-Type':'application/x-www-form-urlencoded'})

1. 请求正文是multipart/form-data

requests.post(url='',data={'key1':'value1','key2':'value2'},headers={'Content-Type':'multipart/form-data'})

1. 请求正文是raw

传入xml格式文本:

requests.post(url='',data='<?xml ?>',headers={'Content-Type':'text/xml'})

传入json格式文本(这种方式在chrome中显示为Request Payload):

requests.post(url='',data=json.dumps({'key1':'value1','key2':'value2'}),headers={'Content-Type':'application/json'})

或者：

requests.post(url='',json={{'key1':'value1','key2':'value2'}},headers={'Content-Type':'application/json'})

1. 请求正文是binary

requests.post(url='',files={'file':open('test.xls','rb')},headers={'Content-Type':'binary'})

例子：

#写入时间

def WriteTime():

url="https://m.cicdi.com/WebTraining/Syllabus/Timing\_Ajax"

headers = {

"Content-Type": "application/x-www-form-urlencoded; charset=UTF-8",

"user-agent": "Mozilla / 5.0(Windows NT 10.0;Win64;x64)AppleWebKit / 537.36(KHTML, likeGecko) Chrome /84.0.4147.89Safari / 537.36Edg / 84.0.522.40",

"cookie": "\_\_jsluid\_s=5695b36e9f5226b162c99d2084d6181f; LoginInfo=userName=889151&userTrueName=%e7%94%b0%e5%86%9b&ROLES=&STAFFING=6689507&CODE=889151&login=true; ASP.NET\_SessionId=bir22hyasgu2lqtur3ql2si1"

}

body = {

"p\_CourseId":"01e12c44-e35e-4fd9-8144-b887015d84fd",

"p\_PlayId":"4e18b6a9-c002-4477-8709-85c333c0922a",

"p\_Time":-50, #当前学习的秒，每次都增加

"p\_Progress":100

}

response = requests.post(url=url,headers=headers,data=body)

print(response)

print(response.content)

Request Payload的例子:

import requests,sys,os,time,re

import json

#写入时间

def WriteTime():

url="https://www.ccspx.com.cn/video/play/refresh"

headers = {

"Content-Type": "application/json;charset=UTF-8",

"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.75 Safari/537.36",

"cookie": "JSESSIONID=48c58abc-76ec-4cef-b0b3-4019f7d6336b; Hm\_lvt\_04f08a1fdecae8accd6731f7066a8d95=1615862677,1615897986; Hm\_lpvt\_04f08a1fdecae8accd6731f7066a8d95=1615900086",

"Referer": "https://www.ccspx.com.cn/course/detail?classId=792830462560264192&catalogType=1",

"Host": "www.ccspx.com.cn"

}

# payloadData数据

payloadData = {

"isFinish":"0",

"isSeeked":"true",

"lastTime":301.689302,

"userId":"718500329791969563",

"videoId":"d56bf9659d0d4781815734dc86f7db84",

"watchNum":3,

"pxStudyRecord": {

"chapterDetailId": "792830872511537152",

"classId": "792830462560264192",

"instId": "710556396281659392",

"kId": "792828133253537792",

"keyName": "study\_log",

"logId": "821485528665960448",

"stageDetailId": "null",

"studyDuration": 0, #具体的时间

"studyEndTime": 1615900937129,

"studyStartTime": 1615899013406,

"tenantId": "703992481523433472",

"userId": "718500329791969563",

"videoTime": 301.689302

}

}

dumpJsonData = json.dumps(payloadData)

response = requests.post(url=url,data=dumpJsonData,headers=headers)

response.encoding="utf-8"

print(response)

print(response.content.decode("utf-8")) #转换为中文字符

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

WriteTime()

可随时补充新的写法。

# 八、json

## 基本

import json

# Python 字典类型转换为 JSON 对象

data = {

'no' : 1,

'name' : 'Runoob',

'url' : 'http://www.runoob.com'

}

json\_str=json.dumps(data)

print ("json对象："+json\_str)

data1=json.loads(json\_str)

print ("name:"+data1["name"])

print ("url:"+data1["url"])

# module 'json' has no attribute 'dumps'

因为同名，不能将文件命名为json.py，这会冲突。

## 转换json.dumps()

dumps是将dict转化成str格式，loads是将str转化成dict格式。

备注：文件路径前面加上 r 是为了避免转义

1 import json

2

3 dict = {'a': 'wo', 'b': 'zai', 'c': 'zhe', 'd': 'li'}

4 string = json.dumps(dict)

5 print(dict)

6 print(string)

7 print(type(dict))

8 print(type(string))

9

10 with open(r'C:\Users\zy\Documents\GitHub\python3\searchTest\json.json','w')as f:

11 f.write(string)

结果：

{'a': 'wo', 'b': 'zai', 'c': 'zhe', 'd': 'li'}

2 {"a": "wo", "b": "zai", "c": "zhe", "d": "li"}

3 <class 'dict'>

4 <class 'str'>

例子：

import json

# Python 字典类型转换为 JSON 对象

data = {

'no' : 1,

'name' : 'Runoob',

'url' : 'http://www.runoob.com'

}

json\_str=json.dumps(data) #序列化

print ("json对象："+json\_str)

data1=json.loads(json\_str)

print ("name:"+data1["name"])

print ("url:"+data1["url"])

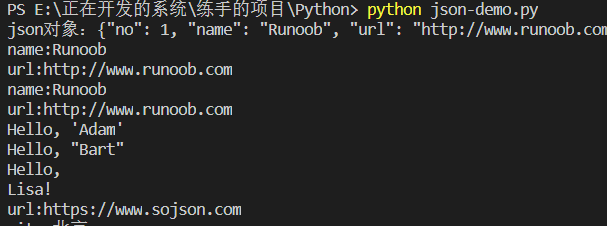
datastr = '{"no" : "1","name" : "Runoob","url" : "http://www.runoob.com"}'

data2=json.loads(datastr)

print ("name:"+data2["name"])

print ("url:"+data2["url"])

先通过dumps转换为字符串，然后json.loadsz再处理字符串，结果：



json.dump()用于将dict类型的数据转成str，并写入到json文件中：

1 dict = {'a': 'wo', 'b': 'zai', 'c': 'zhe', 'd': 'li'}

2 json.dump(dict,open(r'C:\Users\zy\Documents\GitHub\python3\searchTest\json.json','w'))

## 读取json

with open('./test.json','r')as fp:

data2=json.load(fp)

print("url:"+data2["url"])

print("city:"+data2["address"]["city"])

## 读取文本文件中的json并检查

1、str转换为dict是json.loads;

2、数据格式如下图：



import json,os,sys

def DoCheck(path):

list\_chongfu=[]

list\_item=[]

#读取txt

with open(path,'rb') as f:

#解析json

data=json.loads(f.read())

#fieldConfigs name查重

sub\_data=data["fieldConfigs"] #list

for item in sub\_data:

if item["name"] not in list\_item:

list\_item.append(item["name"])

else:

list\_chongfu.append(item["name"])

#打印重复数据

for item\_chongfu in list\_chongfu:

print(item\_chongfu)

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

#DoCheck(os.path.join(os.path.expanduser("~"), 'Desktop')) #没有权限

DoCheck(r'./任务集字典.txt')

## python文件open()函数参数：r和rb比较

# 读取文本文件时，不要使用rb模式打开文件，而应该仅使用r模式。

# 使用r：

# 是推荐使用的打开文本文件的模式。因为使用此模式打开文本文件时，python默认为我们做了一些处理，比如：假设在windows下，将本来应该读入的换行符\r\n处理成\n,方便我们处理。（值得一提的是，当你将\n写入文件时，python也会默认将其替换成\r\n，如果你是win系统的话）

# 补充：其实是启用了通用换行符支持（UNS），它默认开启。

# 使用rb：

# 则python不会对文本文件预处理了，你读入的\r\n依然是\r\n.

# 九、语音朗读

## pyttsx3

import pyttsx3

engine=pyttsx3.init()

engine.say('智勘智设系统介绍')

print('这是一段话')

engine.runAndWait()

先pip install pyttsx3

# 十、多线程



## 基本

from multiprocessing import Pool #多线程

if \_\_name\_\_=='\_\_main\_\_':  
 if(os.name=='nt'):  
 print('你正在使用windows平台！')  
 else:  
 print('你正在使用linux平台！')

pool=Pool(5)

pool.apply\_async(download\_Pic,args=(href,header,title,path))

pool.close()  
pool.join()  
print('操作完毕！')

多线程threading

正常无线程：

import time

import threading

def coding(x):

for x in range(x):

print("正在第%d个画画……" % x)

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

coding(10)

简单多线程：

import time

import threading

def coding(x):

for x in range(x):

print("正在第%d个代码……" % x)

time.sleep(1)

def painting(x):

for x in range(x):

print("正在第%d个画画……" % x)

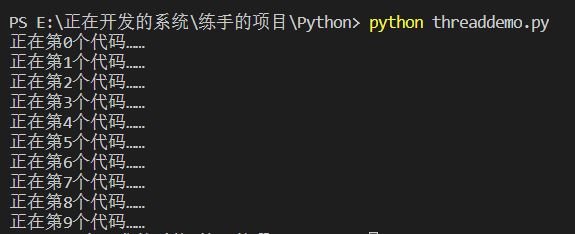
time.sleep(1)

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

t1=threading.Thread(target=coding,args=(10,))

t1.start()

上图中args=(10,)，一方面是参数一定要是可迭代对象，元祖+()，另一方面一定要有个逗号。也可以是一个字典：eat\_thread = threading.Thread(target=eat, kwargs={"name": "爆米花", "number": 1})#此处一定要用kwargs，不能用args



import time

import threading

def coding(x):

for x in range(x):

print("正在第%d个代码……" % x)

time.sleep(1)

def painting(num,name):

for x in range(num):

print("%s正在第%d个画画……" % (name,x))

time.sleep(1)

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

t1=threading.Thread(target=coding,args=(10,))

t2=threading.Thread(target=painting,kwargs={"num":10,"name":"keywords"}) #此处一定要用kwargs，不能用args

t1.start()

time.sleep(5)

t2.start()

结果：



线程之间执行是无序的

主线程会等待所有的子线程执行结束再结束

线程之间共享全局变量

线程之间共享全局变量数据出现错误问题

## 守护线程

<https://www.cnblogs.com/cnkai/p/7504980.html>

设置为守护线程，主线程结束以后，子线程还没有来得及执行，整个程序就退出了。

## 协程

协程（微线程）是比线程更轻量化的存在，像一个进程可以拥有多个线程一样，一个线程也可以拥有多个协程。最重要的是，协程不是被操作系统内核所管理，而完全是由程序所控制。

# 十一、Chrome插件

XPath

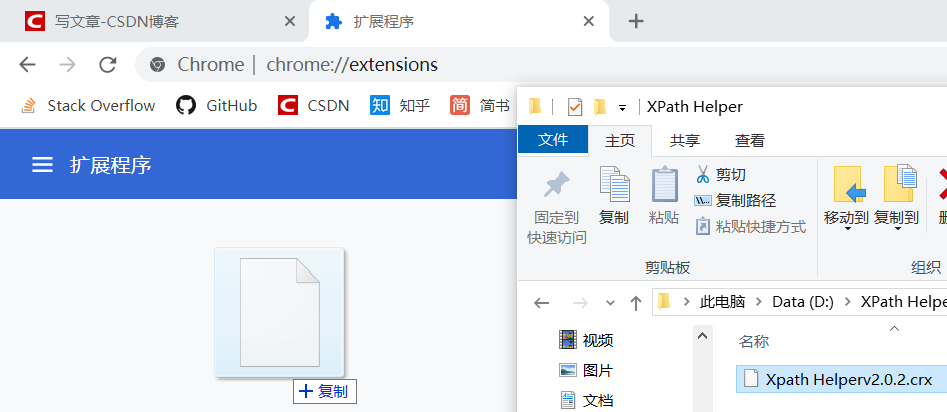
<https://blog.csdn.net/weixin_45961774/article/details/104534166?utm_medium=distribute.pc_relevant_download.none-task-blog-searchfrombaidu-6.nonecase&depth_1-utm_source=distribute.pc_relevant_download.none-task-blog-searchfrombaidu-6.nonecas>

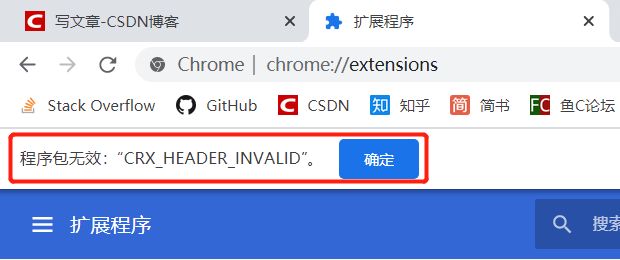
##### XPath Helper 2.0.2版本

链接：<https://pan.baidu.com/s/1pYVld0Kc6NaaYdgpfcQ3rg>  
提取码：tayt

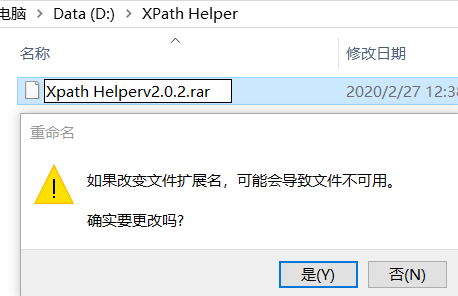
**下载完之后，是一个CRX文件：**  

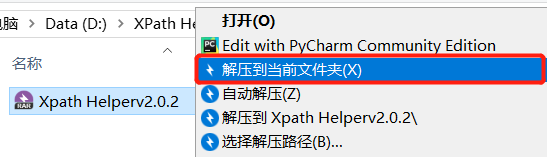
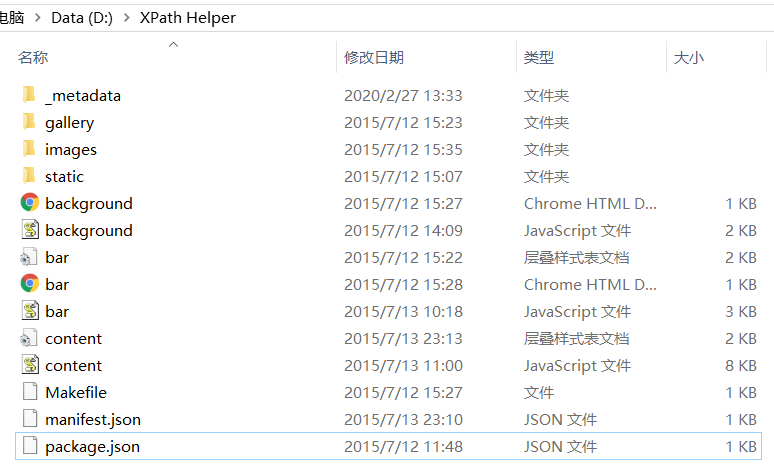

**接下来我们打开Chrome浏览器的扩展程序：**  

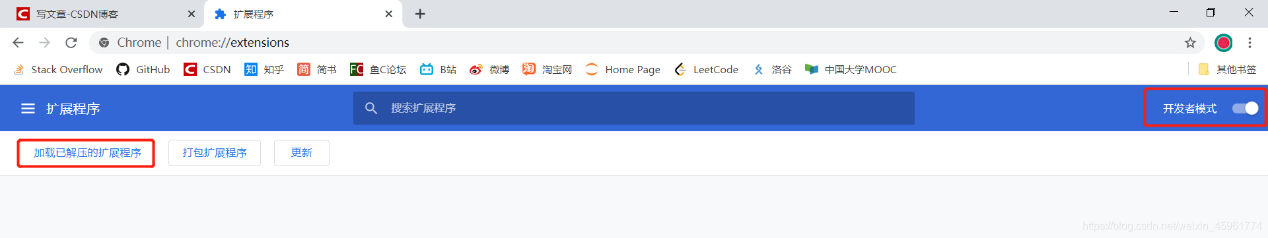

**然后把CRX文件拖动进去：**  


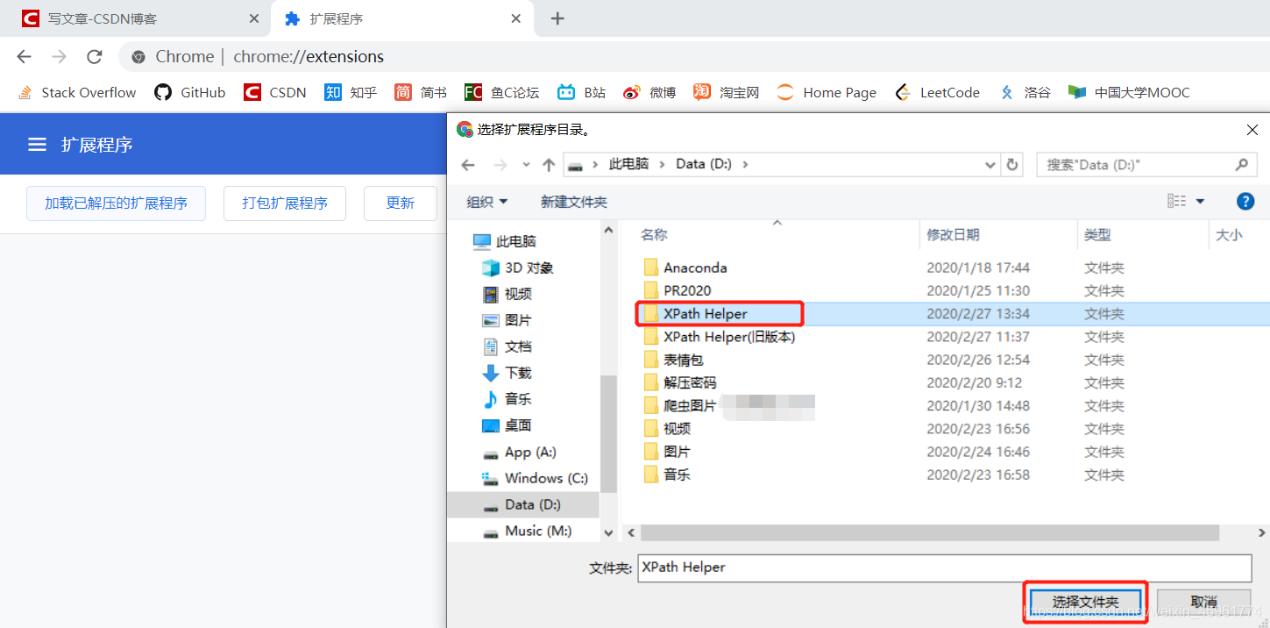
**然而出现了程序包无效：“CRX\_HEADER\_INVALID”。的情况：**  


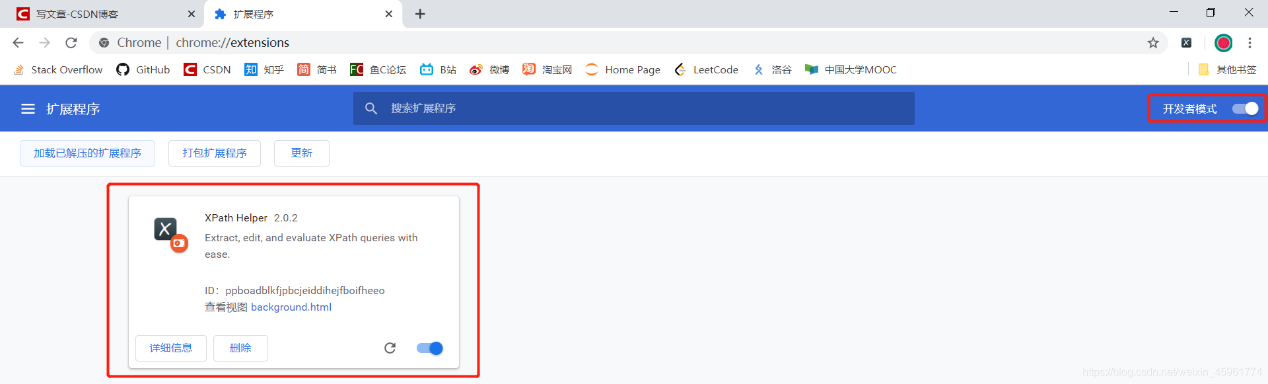
# 解决方法：

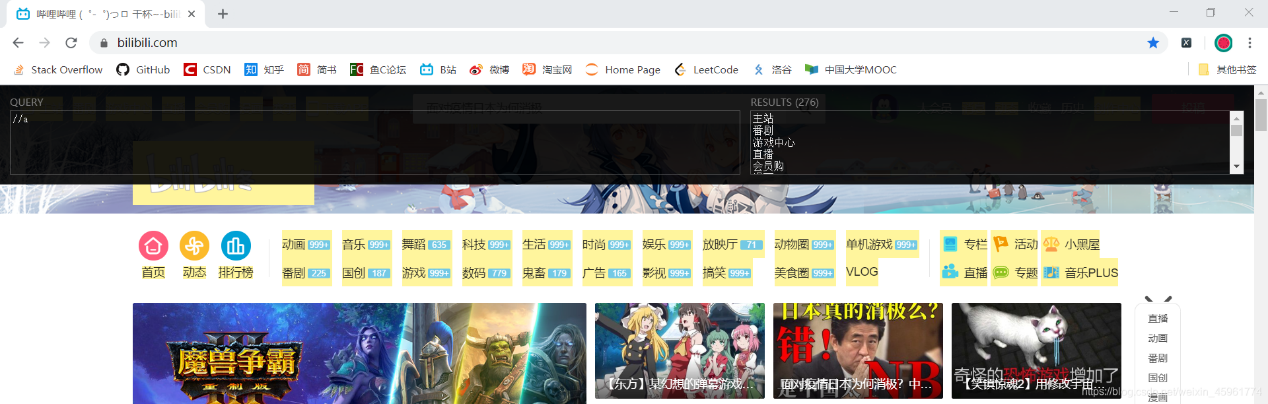
**我们把CRX文件的后缀改成.rar（或者.zip）：**  
  


**然后再解压：**  
  
**解压完把刚才的压缩包删掉：**  


**然后再打开Chrome浏览器的扩展程序，打开右边的开发者模式，接下来点击加载已解压的扩展程序：**  


**选择解压出来的文件所在的文件夹：**  


**然后 XPath Helper 就安装成功了，接下来关闭开发者模式：**  


**接下来重新启动浏览器，按下快捷键 ctrl + shift + x 打开 XPath Helper：**  
  
**然后就可以正常使用 XPath Helper 了。**

# 十二、桌面应用

tkinter

<https://blog.csdn.net/Lawson_Li/article/details/99850686>

<https://blog.csdn.net/weixin_33739541/article/details/93964322?utm_medium=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-6.control&dist_request_id=8005f13c-a345-4a64-9701-07f81ea4d2b4&depth_1-utm_source=distribute.pc_relevant.none-task-blog-BlogCommendFromMachineLearnPai2-6.control>

import tkinter as tk

import threading

window=tk.Tk()

WIDTH=310

HEIGHT=395

screentWidth=window.winfo\_screenwidth() #获取显示区域宽度

screentHeight=window.winfo\_screenheight() #获取显示区域高度

ico\_path='network.ico'

left=(screentWidth-WIDTH)/2

top=(screentHeight-HEIGHT)/2

window.title('微博爬虫')

window.iconbitmap(ico\_path)

#window.geometry('500x100')

window.geometry("%dx%d+%d+%d" % (WIDTH, HEIGHT, left, top))

#加入主窗口销毁事件

#window.protocol('WM\_DELETE\_WINDOW',window\_quit)

window.mainloop()

# 十三、分词

jieba

pip install jieba

jieba分词的三种模式:精确模式、全模式、搜索引擎模式

# 十四、NLP自然语义

tkinter

<https://blog.csdn.net/Lawson_Li/article/details/99850686>

# 十五、常见问题

# Python中解决UnicodeDecodeError: 'gbk' codec can't decode byte 0x86

pycharm中打开文件需要使用open，代码如下

import os

file = open("exercise01.py")

print(file.read())

此时会报错：UnicodeDecodeError: ‘gbk’ codec can’t decode byte 0x86……

笔者的处理方式是，在open函数中加入文件读写模式的说明，其中二进制文件需要用“rb”读写，故将第二行代码改为file = open(“exercise01.py”, “rb”)即可读出文件。