**Python学习经验**

**序列：**

**索引**同样可以是**负数**，在那样的情况下，位置是从序列尾开始计算的。因此，shoplist[-1]表示序列 的最后一个元素而shoplist[-2]抓取序列的倒数第二个项目。

**切片操作符**中的第一个数(冒号之前)表示切片开始的位置，第二个数(冒号之后)表示切片到哪里结 束。如果不指定第一个数，Python就从序列首开始。如果没有指定第二个数，则Python会停止在序列尾。 注意，返回的序列从**开始位置 开始** ，刚好在 **结束 位置之前结束**。即开始位置是包含在序列切片中的，而 结束位置被排斥在切片外

切片： list[m:n] m:n 表示元素 m...n-1。如果切片从链表第一个元素开始，我们可以省略第一个数字;如果切片到链表最后一个元素处结尾，我们可以省略第二个数字

索引同样可以是负数，在那样的情况下，位置是从序列尾开始计算的。因此，shoplist[-1]表示序列 的最后一个元素而shoplist[-2]抓取序列的倒数第二个项目

可以用负数做切片。负数用在从序列尾开始计算的位置。例如，shoplist[:-1]会返回除了最后一个 项目外包含所有项目的序列切片。

使用物理行尾的**反斜杠**来表示逻辑行在下一物理行继续

Option-configure IDLE-Keys： 查看IDLE快捷操作

Control+p 获取之前输入的命令

which pip pip2 pip3 显示python版本路径

sudo pip3 install \*\* 安装到python 3

安装homebrew进行文件管理

**字符串：**

使用单引号(')

你可以用单引号指示字符串，就如同'Quote me on this'这样。所有的空白，即空格和制表符 都照原样保留。

使用双引号(")

在双引号中的字符串与单引号中的字符串的使用完全相同，例如"What's your name?"。可以在双引号中使用单引号。

双引号可以包含单引号，单引号也可以包含双引号

有时字符串跨好几行，我们需要使用反斜杠或者括号，这样解释器就知道第一行的表达式不完整。

使用三引号('''或""") 利用三引号，你可以指示一个多行的字符串。你可以在三引号中自由的使用单引号和双引号。

转义符：用\'来指示单引号——注意这个反斜杠。可以把字符串表示 为'What\'s your name?'。类似地，要在双引号字符串中使用双引号本身的时候，也可以借助于转义符。另外，你可以用转义符\\来指示反斜杠本身。在一个字符串中，行末的单独一个反斜杠表示字符串在下一行继续，而不是开始一个新的行。

自然字符串：指示某些不需要如转义符那样的特别处理的字符串，通过给字符串加上前缀r或R来指定。其中的转义符失效直接输出。例如r"Newlines are indicated by \n"。

加法和乘法可以用于字符串，但减法和除法不行

**Print语句**

Print(,end=’’)

print语句可以使用跟着%符号的项目元组的字符串。这些字符串具备定制的功能。定制让输出满足某种特定的格式。定制可以是%s表示字符串或%d表示整数。元组必须按照相同的顺序来对应这些定制。

在大多数时候，你可以只使用%s定制，而让Python来提你处理剩余的事情。这种方法对数同样奏效。然而，你可能希望使用正确的定制，从而可以避免多一层的检验程序是否正确。

在第二个print语句中，我们使用了一个定制，后面跟着%符号后的单个项目——没有圆括号。这只在字符串中只有一个定制的时候有效。

例：

print ('%s is %d years old' % (name, age))

print ('Why is %s playing with that python?' % name)

if ‘ ’ in XX 判断XX中是否含有某字符

from 文件名 import \* 可以达到跟直接在IDLE中运行python文件相同的效果

python的当前目录是指上一个执行完或目前正在执行的文件所在的目录

有一些在python解释器上可以使用的代码格式，在IDLE文件上使用可能会没有输出

因为这些代码属于中间变量，在解释器里面写相当于单步调试，中间值都能看到，在编辑器里面的话中间值是看不到的

对于可分的变量，可以对几个变量同时赋值等于该变量，但变量的数目必须严格等于该变量含有的可拆分变量的数目，如：

For word,pron in entries

Ph1,ph2,ph3 = pron

当使用IDLE运行过一次文件后，python的当前目录会自动变为该文件所在的目录

**yield关键字**

yield 是一个类似 return 的关键字，只是这个函数返回的是个生成器

当你调用这个函数的时候，函数内部的代码并不立马执行 ，这个函数只是返回一个生成器对象

当你使用for进行迭代的时候，函数中的代码才会执行

第一次迭代中你的函数会执行，从开始到达 yield 关键字，然后返回 yield 后的值作为第一次迭代的返回值. 然后，每次执行这个函数都会继续执行你在函数内部定义的那个循环的下一次，再返回那个值，直到没有可以返回的。

在IDLE中重新运行一个程序，即可将原来的程序终止并开始新的程序

if \_\_name\_\_ == '\_\_main\_\_'的意思是：当.py文件被直接运行时，if \_\_name\_\_ == '\_\_main\_\_'之下的代码块将被运行；当.py文件以模块形式被导入时，if \_\_name\_\_ == '\_\_main\_\_'之下的代码块不被运行。

用open函数write方法写文件内容时，必须在最后写f.close()方法，否则文件无内容

Open函数读取上一级文件目录open('..')，上上一级open('../..')

Readlines()方法连续两次使用时会出现问题，所以最好在第一次使用时把读取到的数据存储在一个链表当中

**Scrapy**

Scrapy提供了一种简便的方式来查看表达式是否正确、是否真的起作用。另起一个命令行窗口，在其中输入scrapy shell“url地址”，记得加上双引号

直接在scrapy shell中输入response.xpath()命令，就可以执行该命令，测试自己写的xpath表达式

解决403问题：在命令上加上-s USER\_AGENT='Mozilla/5.0'，即scrapy shell -s USER\_AGENT='Mozilla/5.0' url

[scrapy.dupefilters] DEBUG: Filtered duplicate request: <GET https://movie.douban.com/subject/25882296/comments?start=20&limit=20&sort=new\_score&status=P&percent\_type=> - no more duplicates will be shown (see DUPEFILTER\_DEBUG to show all duplicates)

表示出现了重复的url，scrapy自动停止工作

出现这个问题的原因是因为没考虑好翻页元素对应的位置，从第二页开始把首页而不是后页的url传送到了next\_url变量里，造成重复

**Xpath**

**Xpath节点**

共有七种类型的节点：元素、属性、文本、命名空间、处理指令、注释以及文档（根）节点

树的根被称为文档节点或根节点

节点关系：父、子、同胞、先辈、后代

先辈包括父、父的父等

后代包括子、子的子等

**Xpath路径表达式**

表达式 描述

nodename 选取此节点的所有子节点。

/ 从根节点选取。

// 从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。

. 选取当前节点。

.. 选取当前节点的父节点。

@ 选取属性。

路径表达式 结果

bookstore 选取 bookstore 元素的所有子节点。

/bookstore 选取根元素 bookstore。

注释：假如路径起始于正斜杠( / )，则此路径始终代表到某元素的绝对路径！

bookstore/book 选取属于 bookstore 的子元素的所有 book 元素。

//book 选取所有 book 子元素，而不管它们在文档中的位置。

bookstore//book 选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。

//@lang 选取名为 lang 的所有属性。

**谓语（Predicates）**

谓语用来查找某个特定的节点或者包含某个指定的值的节点。

谓语被嵌在方括号中。

**选取未知节点**

XPath 通配符可用来选取未知的 XML 元素。

通配符 描述

\* 匹配任何元素节点。

@\* 匹配任何属性节点。

node() 匹配任何类型的节点。

**选取若干路径**

通过在路径表达式中使用“|”运算符，可以选取若干个路径。