

# python 入门教程大全

- 1.[python 入门教程]将 excel 导入到 sqlite 的方法代码
  - 2.[python 入门教程]Python 爬取微博实例分析
  - 3.[python 入门教程]python 基础常识大全
  - 4.[python 入门教程]Python3 解决中文字符输出乱码的方法
  - 5.[python 入门教程]Python 运行错误详解
  - 6.[python 入门教程]Python 面试基础知识大全
  - 7.[python 入门教程]常用 Python 模版库大全
  - 8.[python 入门教程]在 IIS 下配置 Python 的方法
  - 9.[python 入门教程]Python3 中 Random 的实例教程
  - 10.[python 入门教程]python 中关于单行注释、多行注释以及变量、类型基础知识用法
  - 11.[python 入门教程]Python3 中关于字典和列表以及指定元素排序方法举例说明
  - 12.[python 入门教程]Python3 中的 type 和 object 用法
  - 13.[python 入门教程]python3 中 bs4.FeatureNotFound 提示报错的处理办法
  - 14.[python 入门教程]python 安装 requests 的步骤
  - 15.[python 入门教程]初学习 Python 常见错误
  - 16.[python 入门教程]Python 字符串处理大全
  - 17.[python 入门教程]适合初学者的 python 爬虫开发案例
  - 18.[python 入门教程]Python 爬虫开发入门及开发技巧大全
  - 19.[python 入门教程]Python 语言主要应用领域表
  - 20.[python 入门教程]Python 编程主要可以应用于哪些领域?
  - 21.[python 入门教程]对 Python 开发者的编码建议
  - 22.[python 入门教程]Python 字符串处理技巧大全
  - 23.[python 入门教程]零基础学习 Python 需要学习哪些知识?
  - 24.[python 入门教程]学习 Python 零基础指导路线图
  - 25.[python 入门教程]自学 Python 路线图
  - 26.[python 入门教程]学 Python 有三道坎
  - 27.[python 入门教程]零基础学 python 路线图
-

- 28.[python 入门教程]为什么要学习 Python， 能 Python 来做些什么？
- 29.[python 入门教程]Python 与各种开发语言比较、对比优略
- 30.[python 入门教程]全面认识和了解 Python 编程语言
- 31.[python 入门教程]Python 实现字符串处理功能的函数大全
- 32.[python 入门教程]Python 转义字符大全表
- 33.[python 入门教程]python3 以后用 urllib.request 代替 urllib2

# 将 excel 导入到 sqlite 的方法代码

Python 实现 excel 转 sqlite 的方法，具体如下：

Python 环境的安装配置就不说了，个人喜欢 pydev 的开发环境。

python 解析 excel 需要使用第三方的库，这里选择使用 xlrd

先看excel内容：

	A	B	C	D
1	姓名	年龄	手机	
2	candy	21	1112345	
3	ting	23	879874985	
4				

然后是生成的数据库：

_id	name	age	number
1	candy	21	1112345
2	ting	23	879874985

下面是源代码：

```
#!/usr/bin/python
```

```
# encoding=utf-8
```

```
"""
```

```
Created on 2013-4-2
```

```
@author: ting
```

```
'''
```

```
fromxlrdimportopen_workbook
```

```
importsqlite3
```

```
importtypes
```

```
defread_excel(sheet):
```

```
# 判断有效 sheet
```

```
ifsheet.nrows >0andsheet.ncols >0:
```

```
forrowinrange(1, sheet.nrows):
```

```
row_data=[]
```

```
forcolinrange(sheet.ncols):
```

```
data=sheet.cell(row, col).value
```

```
.....
```

```

# excel 表格内容数据类型转换 float->int, unicode->utf-8

if type(data) is types.UnicodeType: data=data.encode("utf-8")

elif type(data) is types.FloatType: data=int(data)

row_data.append(data)

check_data_length(row_data)

# 检查 row_data 长度

def check_data_length(row_data):

    if len(row_data) != 3:

        insert_sqlite(row_data)

def insert_sqlite(row_data):

    # 打开数据库（不存在时会创建数据库）

    con=sqlite3.connect("test.db")

    cur=con.cursor()

    try:

        cur.execute("create table if not exists contacts(_id integer primary key '\

            "autoincrement,name text,age integer,number integer)")

        # 插入数据不要使用拼接字符串的方式，容易收到 sql 注入攻击

        cur.execute("insert into contacts(name,age,number) values(?,?,?)", row_data)

        con.commit()

    except sqlite3.Error as e:

        print "An error occurred: %s", e.args[0]

    finally:

        cur.close

        con.close

    xls_file="test.xls"

    book=open_workbook(xls_file)

    for sheet in book.sheets():

        read_excel(sheet)

    print "----- Done -----"

```

# Python 爬取微博实例分析

## 引言

利用 Ajax 分析微博并爬取其内容如微博内容，点赞数，转发数，评论数等。

## 分析

打开陈一发微博网站: <https://m.weibo.cn/p/1005051054009064>，并同时打开开发者工具。

我们可以在 Network 中分析可以得出，微博中所有的内容都是名为

“getIndex?containerid=1076031054009064”的响应

The screenshot shows a mobile browser view of a Weibo post by user '陈一发儿'. The post text is: '弦上有春秋是我和@暗杠小发 第三次合作的作品，为了把歌曲意境更完美的勾勒出来，从演唱到歌曲制作上都讨论修改了很多个版本~马上就过年了，今天歌曲全网上线了，就当作给大家的新春礼物吧~#亚洲新歌榜#@陈一发儿《弦上有春秋-陈一发儿》>弦上有春秋-陈一发儿#陈一发儿#'. Below the text are interaction counts: 2784 likes, 4221 comments, and 2.8万 reposts. The network tab on the right shows a request to 'getIndex?containerid=1076031054009064' with a large JSON response. The response contains a 'data' object with 'cardlistInfo' and 'cards' arrays, which contain detailed information about the post and its related content.

经过这其中的 JSON 代码分析后可以找到内容相对应的位置。

This screenshot is an annotated version of the previous one, showing how JSON data maps to the UI. Red boxes highlight specific fields in the JSON response, and red arrows point from these fields to their corresponding elements on the page:

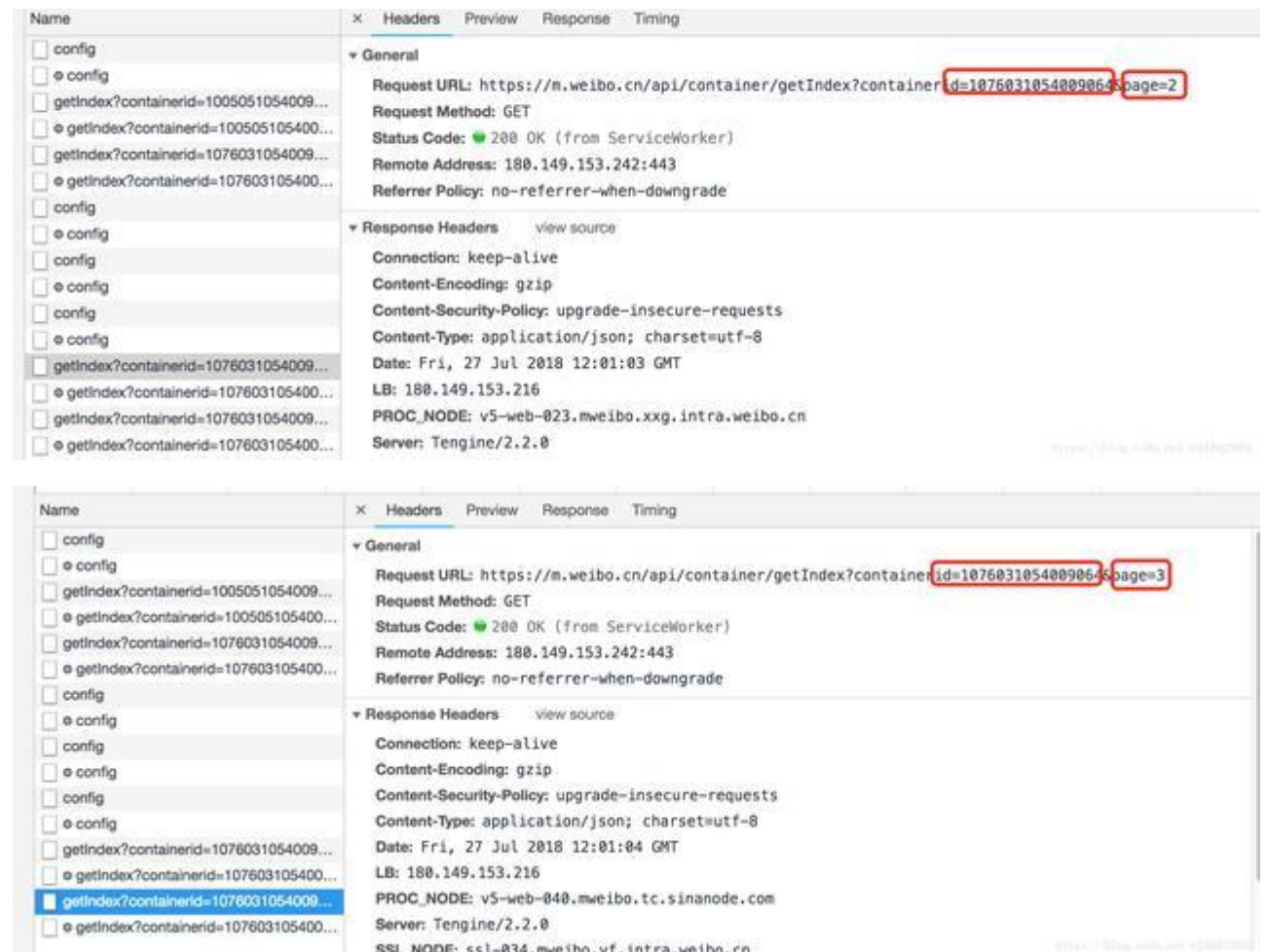
- articleId**: Points to the main text of the post.
- comments.count**: Points to the comment count (4221).
- reposts.count**: Points to the repost count (2.8万).
- source**: Points to the '微博 weibo.com' link.

The JSON response also includes other fields like 'created\_at', 'expire\_time', 'is\_paid', 'mblog\_vip\_type', 'mid', 'more\_info\_type', 'page\_info', and 'text'.

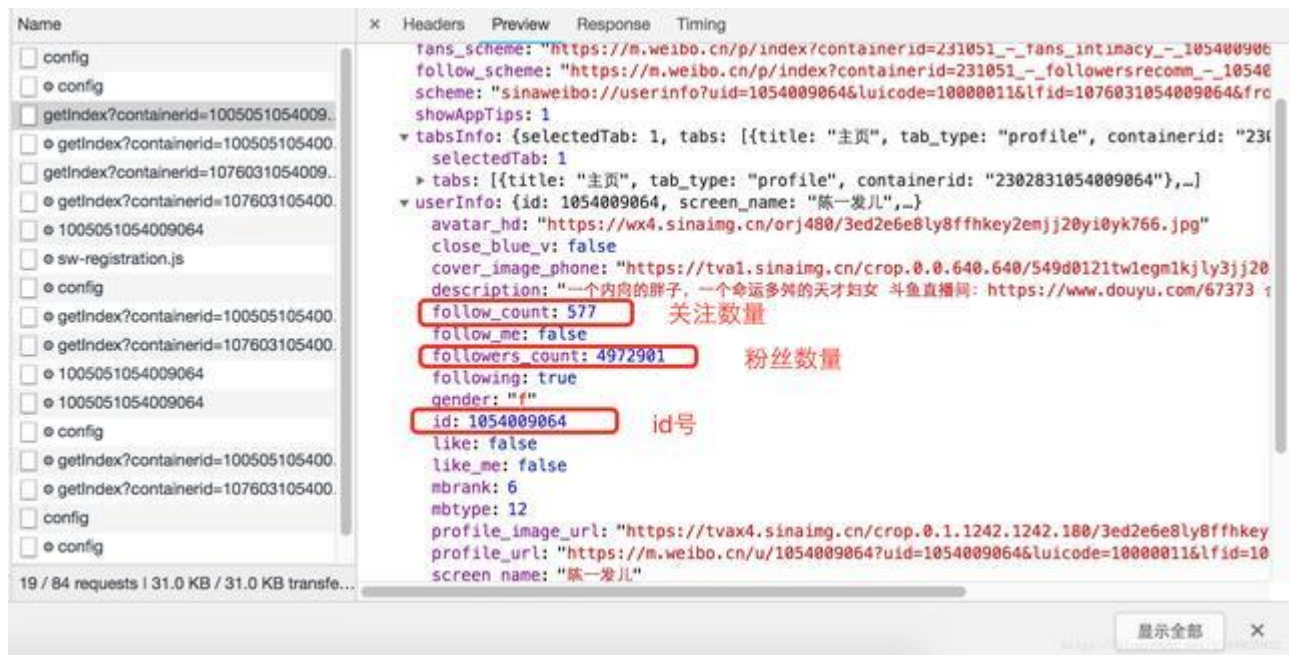
我们可以尝试切换到第一个响应中，查看返回的结果，可以看到代码只要不到 50 行，所以这些数据都是浏览器拿到数据后再进一步渲染出来的。

我们在尝试滚动页面，加载完成后会发现得到了新的响应，这也对应着加载出来的新的微博内容。

通过前后几次 Request URL 代码的不同来看



这其中改变的只有后面的 page，而其中的 containerid 一直都是固定的，组成是由 107603 加上 id 号。



这里还可以得到一个很重要的参数 **total**：微博的总数量，由此我们可以用来估计 page 的数量。

### 代码编写

这里使用 `urlencode()` 方法来将参数转化为 URL 的 GET 请求参数  
`pyQuery` 来解析代码。



```

1 import requests
2 from urllib.parse import urlencode
3 from pyquery import PyQuery as pq
4
5 base_url = 'https://m.weibo.cn/api/container/getIndex?'
6 headers = {
7     'Host': 'm.weibo.cn',
8     'Referer': 'https://m.weibo.cn/u/2830678474',
9     'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, l
10     'X-Requested-With': 'XMLHttpRequest',
11 }
12
13 max_page = 10
14
15 def get_page(page):
16     params = {
17         'containerid': '1076031054009064',
18         'page': page
19     }
20     url = base_url + urlencode(params)
21     print(url)
22     try:
23         response = requests.get(url, headers=headers)
24         if response.status_code == 200:
25             return response.json(), page
26     except requests.ConnectionError as e:
27         print('Error', e.args)
28
29
30 def parse_page(json, page: int):
31     if json:
32         items = json.get('data').get('cards')
33         for index, item in enumerate(items):
34             if page == 1 and index == 1:
35                 continue
36             else:
37                 item = item.get('mblog')
38                 weibo = {}
39                 weibo['id'] = item.get('id')
40                 weibo['text'] = pq(item.get('text')).text()
41                 weibo['attitudes'] = item.get('attitudes_count')
42                 weibo['comments'] = item.get('comments_count')
43                 weibo['reposts'] = item.get('reposts_count')
44                 yield weibo
45
46 if __name__ == '__main__':
47     for page in range(1, max_page + 1):
48         json = get_page(page)
49         results = parse_page(*json)
50         for result in results:

```

代码中，因为是 Ajax 请求得到的内容，所以请求头必须带上 'X-Requested-With': 'XMLHttpRequest'！

到此为止，我们便可以得到喜欢博主的所有微博内容了。





## 文末知识点摘要：Python 解惑之：整数比较

在 Python 中一切都是对象，毫无例外整数也是对象，对象之间比较是否相等可以用 `==`，也可以用 `is`。 `==`和 `is` 操作的区别是：

- `is` 比较的是两个对象的 `id` 值是否相等，也就是比较俩对象是否为同一个实例对象，是否指向同一个内存地址。
- `==`比较的是两个对象的内容是否相等，默认会调用对象的 `__eq__()`方法。

清楚 `is` 和 `==`的区别之后，对此也许你可能会遇到下面的这些困惑，于是就有了这样一篇文章，试图把 Python 中一些隐晦的东西扒出来，希望对你有一定的帮助。我们先来看两段代码：

片段一：

```
>>> a = 256
>>> b = 256
>>> a == b
True
>>>
```

片段二：

```
>>> a = 256
>>> b = 256
>>> a is b
True
>>>
```

在交互式命令行执行上面两段代码，代码片段一中的 `a==b` 返回 `True` 很好理解，因为两个对象的值都是 256，对于片段二， `a is b` 也返回 `True`，这说明 `a` 和 `b` 是指向同一个对象的，可以检查一下他们的 `id` 值是否相等：

```
>>> id(a)
8213296
```

```
>>> id(b)
8213296
>>>
```

结果证明他俩的确是同一个对象，指向的是同一个内存地址。那是不是所有的整数对象只要两个对象的值（内容）相等，它们就是同一个实例对象呢？换句话说，对于整数对象只要 `==` 返回 `True`，`is` 操作也会返回 `True` 吗？带着这个问题来看下面这两段代码：

片段一：

```
>>> a = 257
>>> b = 257
>>> a == b
True
>>>
```

片段二：

```
>>> a = 257
>>> b = 257
>>> a is b
False
>>>
```

对于 `257`，`a is b` 返回的竟然是 `False`，结果可能在你的意料之中，也有可能出乎你的意料，但不管怎么，我们还是要刨根问底，找出问题的真相。

解惑一

出于对性能的考虑，**Python** 内部做了很多的优化工作，对于整数对象，**Python** 把一些频繁使用的整数对象缓存起来，保存到一个叫 `small_ints` 的链表中，在 **Python** 的整个生命周期内，任何需要引用这些整数对象的地方，都不再重新创建新的对象，而是直接引用缓存中的对象。**Python** 把这些可能频繁使用的整数对象规定在范围 `[-5, 256]` 之间的小对象放在 `small_ints` 中，但凡是需要用些小整数时，就从这里面取，不再去临时创建新的对象。因为 `257` 不再小整数范围内，因此尽管 `a` 和 `b` 的值是一样的，但是他们在 **Python** 内部却是以两个独立的对象存在的，各自为政，互不干涉。

---

弄明白第一个问题后，我们继续在 **Python** 交互式命令行中写一个函数，再来看下面这段代码：

片段一：

```
>>> c = 257
>>> def foo():
...     a = 257
...     b = 257
...     print a is b
...     print a is c
...
>>> foo()
True
False
```

呃，什么情况，是的，你没看错，片段一中的这段代码 `a`、`b` 值都是 `257` 的情况下，出现了 `a is b` 返回 `True`，而 `a is c` 返回的 `False`，`a`、`b`、`c` 的值都为 `257`，为什么会出现不同

的结果呢？这对于刚刚好不容易建立起来的认知就被彻底否決了吗，那这段代码中究竟发生了什么？难道解惑一中的结论是错误的吗？

## 解惑二

A Python program is constructed from code blocks. A block is a piece of Python program text that is executed as a unit. The following are blocks: a module, a function body, and a class definition. Each command typed interactively is a block. A script file (a file given as standard input to the interpreter or specified as a command line argument to the interpreter) is a code block. A script command (a command specified on the interpreter command line with the '-c' option) is a code block.

### structure-of-a-program

为了弄清楚这个问题，我们有必要先理解程序代码块的概念。Python 程序由代码块构成，代码块作为程序的一个最小基本单位来执行。一个模块文件、一个函数体、一个类、交互式命令中的单行代码都叫做一个代码块。在上面这段代码中，由两个代码块构成，`c = 257` 作为一个代码块，函数 `foo` 作为另外一个代码块。Python 内部为了将性能进一步的提高，凡是在一个代码块中创建的整数对象，如果存在一个值与其相同的对象于该代码块中了，那么就直接引用，否则创建一个新的对象出来。**Python 出于对性能的考虑，但凡是不可变对象，在同一个代码块中的对象，只有是值相同的对象，就不会重复创建，而是直接引用已经存在的对象。**因此，不仅是整数对象，还有字符串对象也遵循同样的原则。所以 `a is b` 就理所当然的返回 `True` 了，而 `c` 和 `a` 不在同一个代码块中，因此在 Python 内部创建了两个值都是 `257` 的对象。为了验证刚刚的结论，我们可以借用 `dis` 模块从字节码的角度来看看这段代码。

```
1. >>> import dis
2. >>> dis.dis(foo)
3.      2          0 LOAD_CONST          1 (257)
4.      3          3 STORE_FAST        0 (a)
5.
6.      3          6 LOAD_CONST          1 (257)
7.      9          9 STORE_FAST        1 (b)
8.
9.      4         12 LOAD_FAST           0 (a)
10.         15 LOAD_FAST           1 (b)
11.         18 COMPARE_OP         5 (==)
12.         21 PRINT_ITEM
13.         22 PRINT_NEWLINE
14.
15.      5         23 LOAD_FAST           0 (a)
16.         26 LOAD_GLOBAL         0 (c)
17.         29 COMPARE_OP         5 (==)
18.         32 PRINT_ITEM
19.         33 PRINT_NEWLINE
20.         34 LOAD_CONST          0 (None)
21.         37 RETURN_VALUE
```

可以看出两个 `257` 都是从常量池的同一个位置 `co_consts[1]` 获取的。

## 总结

一番长篇大论之后，得出两点结论：1、小整数对象`[-5,256]`是全局解释器范围内被重复使用，永远不会被 GC 回收。2、同一个代码块中的不可变对象，只要值是相等的就不会重复创建新的对象。似乎这些知识点对日常的工作一点忙也帮不上，因为你根本不会用 `is` 来比

较两个整数对象的值是否相等。那为什么还要拿出来讨论呢？嗯，程序员学知识，不应该浅尝辄止，要充分发挥死磕到底的精神。

---

# python 基础常识大全

**0.什么是 Python？使用 Python 有什么好处？**（这个问题是最常见的开头问题，是最基础也是最重要的！）

答案：下面是一些关键点：

- Python 是一种解释型语言。这就是说，与 C 语言和 C 的衍生语言不同，Python 代码在运行之前不需要编译。其他解释型语言还包括 PHP 和 Ruby。

- 
- Python 是动态类型语言，指的是你在声明变量时，不需要说明变量的类型。你可以直接编写类似 `x=111` 和 `x="I'm a string"` 这样的代码，程序不会报错。

- 
- Python 用途非常广泛——网络应用，自动化，科学建模，大数据应用，等等。它也被用作“胶水语言”，帮助其他语言和组件改善运行状况。

- 
- Python 让困难的事情变得容易，因此程序员可以专注于算法和数据结构的设计，而不用处理底层的细节。

- 
- **1.python 中 is 和 == 的区别**（总结了大部分人的面试，这道题出现的概率也很大。）

答案：

- Python 中对象包含的三个基本要素，分别是：`id`(身份标识)、`type`(数据类型)和 `value`(值)。

- 
- `'=='` 比较的是 `value` 值

- 
- `'is'` 比较的是 `id`

-

## 2. Python 是怎样管理内存的？

答案：

Python 的内存管理是由私有 heap 空间管理的。所有的 Python 对象和数据结构都在一个私有 heap 中。程序员没有访问该 heap 的权限，只有解释器才能对它进行操作。

另外，Python 有自带的垃圾回收系统，它回收并释放没有被使用的内存让它们能够被其他程序使用。

## 3. 有哪些工具可以帮助 debug 或做静态分析？

答案：

PyChecker，一个静态分析工具，除了报告源代码中的错误，还能分析出错误的类型和复杂程度。另外，还有 Pylint，用于检验模块是否达到代码标准的工具。

## 4. 你如何管理不同版本的代码？

答案：

一点不属于专业的小技巧——被问到这个问题的时候，你应该要表现得很兴奋，甚至告诉他们你是如何使用 Git（或是其他你最喜欢的工具）追踪自己和女票的书信往来。除了 Git 作为版本控制系统（VCS），你也可以选择 subversion（SVN）。

## 5. 什么是 Python 的命名空间？

答案：

python 使用命名空间记录变量。python 中的命名空间就像是一个 dict，key 是变量的名字，value 是变量的值。

如果你记不住上面这段标准答案，也可以这么回答：

在 Python 中，所有的名字都存在于一个空间中，它们在该空间中存在和被操作——这就是命名空间。每一个变量名字都尤其对应的一个对象，而命名空间可是把他们收纳起来的盒子，当查询变量的时候，会从该盒子里面寻找相应的对象。

## 6. Python 中的 pass 是什么？

答案：

Pass 是一个不可或缺但又毫无作用的语句。pass 就是什么也不做，只是为了防止语法错误，比如：if a>1: pass #我这里先不做任何处理，直接跳过，但是如果不写 pass，就会语法错误。

## 7. 在 Python 中如何拷贝一个对象？

答案：

一般来说可以使用 copy.copy 方法或者 copy.deepcopy 方法，几乎所有的对象都可以被拷贝，一些对象可以更容易的拷贝，Dictionaries 有一个 copy 方法：newdict = olddict.copy 但并不是所有的对象都可以被拷贝。

## 8. Xrange 和 range 的区别是什么？

Xrange 用于返回一个 xrange 对象，而 range 用于返回一个数组。不管那个范围多大，Xrange 都使用同样的内存。

其实在面试中，面试官往往不会出太难的问题，只要掌握好基础，大部分都能过关。

---

# Python3 解决中文字符输出乱码的方法

举例说明：Python3 解决中文字符输出乱码的方法

方法一：

现在用 notepad++，在 UTF-8 格式下编写以下语句：

```
#coding=utf-8  
print"打印中文字符"
```

方法二：

用 encode 和 decode

如：

```
import os.path  
import xlrd,sys  
Filename='./abc.xls'  
if not os.path.isfile(Filename):  
    raise NameError,"%s is not a valid filename"%Filename  
bk=xlrd.open_workbook(Filename)  
shxrange=range(bk.nsheets)  
  
print shxrange  
for x in shxrange:  
    p=bk.sheets()[x].name.encode('utf-8')  
    print p.decode('utf-8')
```

方法三：

在文件开头加上：

```
reload(sys)  
sys.setdefaultencoding('utf8')
```

---



---

# Python 运行错误详解

python 常见的运行时错误原因及解决办法详解:

1.忘记在 if, elif, else, for, while, class, def 语句末尾添加冒号(:), 从而导致: "SyntaxError: invalid syntax"错误

错误发生在如下类似代码里:

```
if spam == 42
print('Hello!')
```

2.使用=号, 而不是==号, 从而导致 "SyntaxError: invalid syntax"错误

"="是赋值语句, 而"=="号是比较两值是否相等的, 这种错误常发生在如下类似代码中:

```
if spam = 42:
print('Hello!')
```

3.使用缩进量错误,从而导致"IndentationError: unexpected indent"、"IndentationError: unindent does not match any outer indentation level"和"IndentationError: expected an indented block"错误

要记住的是缩进只发生在以冒号(:)为结尾的语句之后的行, 而此段语句结束后, 必须恢复到之前的缩进格式, 这种错误常发生在如下代码之中:

```
print('Hello!')
print('Howdy!')
```

---

...and this:

```
if spam == 42:  
    print('Hello!')  
    print('Howdy!')  
...and this:
```

```
if spam == 42:  
    print('Hello!')
```

4.在 `for` 循环语句中忘记调用 `len()` 从而导致“`TypeError: 'list' object cannot be interpreted as an integer`”错误

你想要通过索引来迭代一个 `list` 或者 `string` 的元素，这时就需要调用 `range()` 函数。但是 `range` 函数接受的是只能是数字，比如 `len()` 的返回值，而非一个列表。这种错误常发生在如下代码中：

```
spam = ['cat', 'dog', 'mouse']  
for i in range(spam):  
    print(spam[i])
```

5.尝试修改 `string` 的值，从而导致“`TypeError: 'str' object does not support item assignment`”错误

`string` 是一种不可变的数据类型，该错误常发生在如下代码中：

```
spam = 'I have a pet cat.'  
spam[13] = 'r'  
print(spam)
```

---

6.尝试连接非字符串值与字符串，从而导致“TypeError: Can't convert 'int' object to str implicitly”错误

错误常发生在如下代码中：

```
numEggs = 12  
print('I have ' + numEggs + ' eggs.')
```

而你实际想这样做：

```
numEggs = 12  
print('I have ' + str(numEggs) + ' eggs.')
```

或者：

```
numEggs = 12  
print('I have %s eggs.' % (numEggs))
```

7.在字符串首尾忘记加引号,从而导致“SyntaxError: EOL while scanning string literal”错误

该错误发生在如下代码中：

```
print(Hello!)
```

或者：

```
print('Hello!)
```

或者：

```
myName = 'Al'  
print('My name is ' + myName + . How are you?')
```

8.变量或者函数名拼写错误，从而导致“NameError: name 'fooba' is not defined”错误

-----

该错误发生在如下代码中：

```
foobar = 'Al'  
print('My name is ' + fooba)
```

或者：

```
spam = ruond(4.2)
```

或者：

```
spam = Round(4.2)
```

9.方法名拼写错误，从而导致“AttributeError: 'str' object has no attribute 'lowerr'”错误

该错误发生在如下代码中：

```
spam = 'THIS IS IN LOWERCASE.'  
spam = spam.lowerr()
```

10.引用 list 下标超出范围，从而导致“IndexError: list index out of range”错误

该错误常发生在如下代码中：

```
spam = ['cat', 'dog', 'mouse']  
print(spam[6])
```

11.使用不存在的字典键值，从而导致“KeyError: 'spam'”错误

该错误发生在如下代码中：

```
spam = {'cat': 'Zophie', 'dog': 'Basil', 'mouse': 'Whiskers'}  
print('The name of my pet zebra is ' + spam['zebra'])
```

12.尝试使用 Python 关键字作为变量名，从而导致“SyntaxError: invalid syntax”错误

---

Python 关键字不能用作变量名，该错误发生在如下代码中：

```
class = 'algebra'
```

Python3 的关键字有：and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield

13. 在一个定义新变量中使用增值操作符，从而导致“NameError: name ‘eggs’ is not defined”错误

不要以为变量在创建时就使用 0 或者空字符串作为初始值，就使用自增操作符的一句 `spam += 1` 或 `spam = spam + 1`，而 `spam` 是需要手动显示的指定一个有效的初始值。

该错误发生在如下代码中：

```
spam = 0
spam += 42
eggs += 42
```

14. 在定义局部变量前在函数中使用局部变量(此时有与局部变量同名的全局变量存在), 则会导致“UnboundLocalError: local variable ‘foobar’ referenced before assignment”)

在函数中使用局部变量，而同时又存在同名全局变量时是很复杂的，使用规则是：如果在函数中定义任何东西，如果它只是在函数中使用那它就是局部的，反之就是全局变量。

这意味着你不能在定义它之前把它当全局变量在函数中使用。

该错误发生在如下代码中：

```
someVar = 42
```

-----

```
def myFunction():  
    print(someVar)  
    someVar = 100  
myFunction()
```

15.尝试使用 `range()` 创建整数列表（导致“`TypeError: 'range' object does not support item assignment`”）

有时你想要得到一个有序的整数列表，所以 `range()` 看上去是生成此列表的不错方式。然而，你需要记住 `range()` 返回的是 “range object”，而不是实际的 `list` 值。

该错误发生在如下代码中：

```
spam = range(10)  
spam[4] = -1
```

也许这才是你想做：

```
spam = list(range(10))  
spam[4] = -1
```

注意：在 `Python 2` 中 `spam = range(10)` 是能行的，因为在 `Python 2` 中 `range()` 返回的是 `list` 值，但是在 `Python 3` 中就会产生以上错误

16.使用 `++` 或 `--` 自增自减操作符，从而导致“`SyntaxError: invalid syntax`”）

如果你习惯于例如 `C++` , `Java` , `PHP` 等其他语言，也许你会想要尝试使用 `++` 或者 `--` 自增自减一个变量。在 `Python` 中是没有这样的操作符的。

该错误发生在如下代码中：

```
spam = 1  
-----
```



```
spam++
```

也许这才是你想做的：

```
spam = 1
```

```
spam += 1
```

17. 忘记为方法的第一个参数添加 **self** 参数（导致“`TypeError: myMethod() takes no arguments (1 given)`”）

该错误发生在如下代码中：

```
class Foo():
```

```
def myMethod():
```

```
print('Hello!')
```

```
a = Foo()
```

```
a.myMethod()
```

---

# Python 面试基础知识大全

Python 新手在谋求一份 Python 编程工作前，必须熟知 Python 的基础知识。既有基本的 Python 面试题，也有高阶版试题来指导你准备面试，试题均附有答案。面试题内容包括编码、数据结构、脚本撰写等话题。

## 1: Python 有哪些特点和优点？

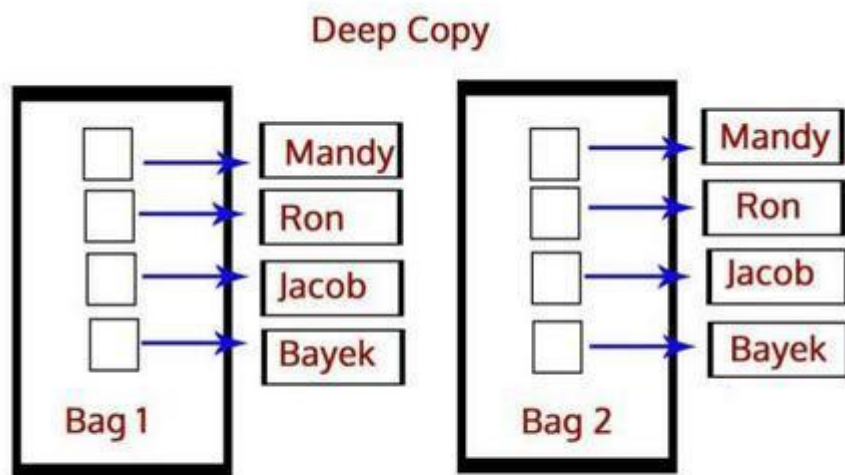
作为一门编程入门语言，Python 主要有以下特点和优点：

- 可解释
- 具有动态特性
- 面向对象
- 简明简单
- 开源
- 具有强大的社区支持

## 2: 深拷贝和浅拷贝之间的区别是什么？（文末放了一份 Python 学习资料给各位哟）

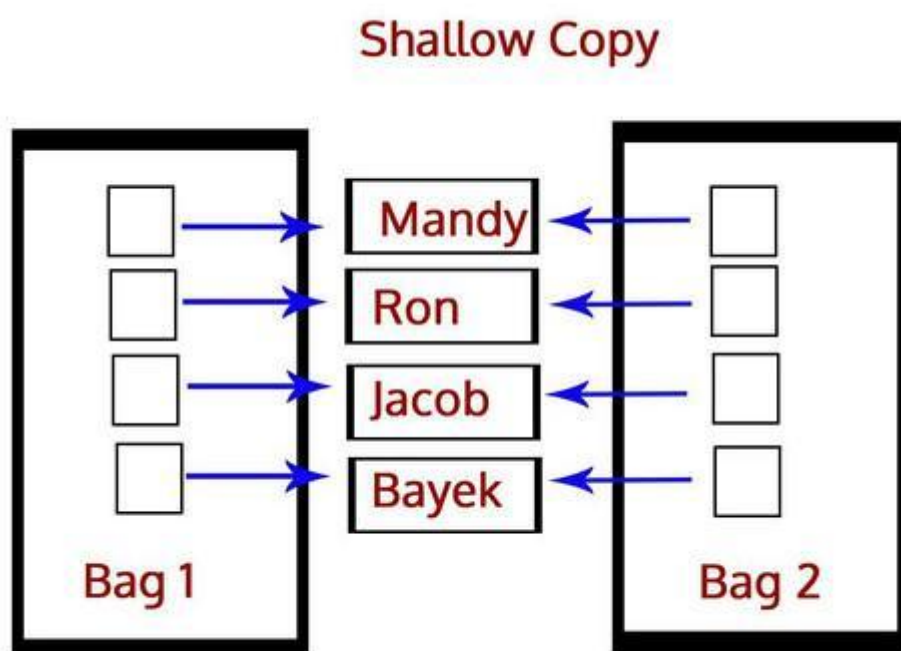
答：深拷贝就是将一个对象拷贝到另一个对象中，这意味着如果你对一个对象的拷贝做出改变时，不会影响原对象。在 Python 中，我们使用函数 `deepcopy()` 执行深拷贝，导入模块 `copy`，如下所示：

```
>>> import copy
>>> b=copy.deepcopy(a)
```



而浅拷贝则是将一个对象的引用拷贝到另一个对象上，所以如果我们在拷贝中改动，会影响到原对象。我们使用函数`copy()`执行浅拷贝，使用如下所示：

```
>>> b=copy.copy(a)
```



### 3. 列表和元组之间的区别是？

答：二者的主要区别是列表是可变的，而元组是不可变的。举个例子，如下所示：

```
>>> mylist=[1,3,3]
>>> mylist[1]=2
>>> mytuple=(1,3,3)
>>> mytuple[1]=2
Traceback (most recent call last):
File "<pyshell#97>", line 1, in <module>
mytuple[1]=2
```

会出现以下报错：

```
TypeError: 'tuple' object does not support item assignment
```

### 4. 解释一下 Python 中的三元运算符

不像C++，我们在Python中没有`?:`，但我们有这个：

```
[on true] if [expression] else [on false]
```

如果表达式为True，就执行[on true]中的语句。否则，就执行[on false]中的语句。

下面是使用它的方法：

```
>>> a,b=2,3
>>> min=a if a<b else b
>>> min
```

```
>>> print("Hi") if a<b else print("Bye")
```

运行结果：

```
Hi
```

## 5. 在 Python 中如何实现多线程？

一个线程就是一个轻量级进程，多线程能让我们一次执行多个线程。我们都知道，Python 是多线程语言，其内置有多线程工具包。

Python中的GIL（全局解释器锁）确保一次执行单个线程。一个线程保存GIL并在将其传递给下一个线程之前执行一些操作，这会让我们产生并行运行的错觉。但实际上，只是线程在CPU上轮流运行。当然，所有的传递会增加程序执行的内存压力。

## 6. 解释一下 Python 中的继承

当一个类继承自另一个类，它就被称为一个子类/派生类，继承自父类/基类/超类。它会继承/获取所有类成员（属性和方法）。

继承能让我们重新使用代码，也能更容易的创建和维护应用。Python 支持如下种类的继承：

## 7. 什么是 Flask？

Flask是Python编写的一款轻量级Web应用框架。其WSGI工具箱采用 Werkzeug，模板引擎使用 Jinja2。Flask使用 BSD 授权。其中两个环境依赖是Werkzeug和jinja2，这意味着它不需要依赖外部库。正因如此，我们将其称为轻量级框架。

Flask会话使用签名cookie让用户查看和修改会话内容。它会记录从一个请求到另一个请求的信息。不过，要想修改会话，用户必须有密钥Flask.secret\_key。

## 8. 在 Python 中是如何管理内存的？

Python 有一个私有堆空间来保存所有的对象和数据结构。作为开发者，我们无法访问它，是解释器在管理它。但是有了核心 API 后，我们可以访问一些工具。Python 内存管理器控制内存分配。

另外，内置垃圾回收器会回收使用所有的未使用内存，所以使其适用于堆空间。

## 9. 解释 Python 中的 help()和 dir()函数

Help()函数是一个内置函数，用于查看函数或模块用途的详细说明：

```
>>> import copy
>>> help(copy.copy)
```

运行结果为：

```
Help on function copy in module copy:

copy(x)

Shallow copy operation on arbitrary Python objects.

See the module's __doc__ string for more info.
```

Dir()函数也是Python内置函数，dir() 函数不带参数时，返回当前范围内的变量、方法和定义的类型列表；带参数时，返回参数的属性、方法列表。

以下实例展示了 dir 的使用方法：

```
>>> dir(copy.copy)
```

运行结果为：

```
['__annotations__', '__call__', '__class__', '__closure__',
 '__code__', '__defaults__', '__delattr__', '__dict__',
 '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
 '__get__', '__getattr__', '__globals__', '__gt__',
 '__hash__', '__init__', '__init_subclass__',
 '__kwdefaults__', '__le__', '__lt__', '__module__',
 '__name__', '__ne__', '__new__', '__qualname__',
 '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
 '__sizeof__', '__str__', '__subclasshook__']
```

#### 10. 当退出 Python 时，是否释放全部内存？

答案是 No。循环引用其它对象或引用自全局命名空间的对象的模块，在 Python 退出时并非完全释放。

另外，也不会释放 C 库保留的内存部分。

#### 11. 什么是猴子补丁？

在运行期间动态修改一个类或模块。

```
>>> class A:
    def func(self):
        print("Hi")
>>> def monkey(self):
    print "Hi, monkey"
>>> m.A.func = monkey
>>> a = m.A()
>>> a.func()
```

运行结果为:

```
Hi, Monkey
```

## 12. Python 中的字典是什么?

字典是 C++ 和 Java 等编程语言中所没有的东西, 它具有键值对。

```
>>> roots={25:5,16:4,9:3,4:2,1:1}
>>> type(roots)
<class 'dict'>
>>> roots[9]
```

运行结果为:

```
3
```

字典是不可变的, 我们也能用一个推导式来创建它。

```
>>> roots={x**2:x for x in range(5,0,-1)}
>>> roots
```

运行结果:

```
{25: 5, 16: 4, 9: 3, 4: 2, 1: 1}
```

## 13. 请解释使用 \*args 和 \*\*kwargs 的含义

当我们不知道向函数传递多少参数时, 比如我们向传递一个列表或元组, 我们就使用 \*args。

```
>>> def func(*args):
    for i in args:
        print(i)
>>> func(3,2,1,4,7)
```



运行结果为：

```
3
2
1
4
7
```

在我们不知道该传递多少关键字参数时，使用\*\*kwargs来收集关键字参数。

```
>>> def func(**kwargs):
    for i in kwargs:
        print(i,kwargs[i])
>>> func(a=1,b=2,c=7)
```

运行结果为：

```
a.1
b.2
c.7
```

**14.** 请写一个 **Python** 逻辑，计算一个文件中的大写字母数量

运行结果：

```
26
```

**15.** 什么是负索引？

我们先创建这样一个列表：

```
>>> mylist=[0,1,2,3,4,5,6,7,8]
```

负索引和正索引不同，它是从右边开始检索。

```
>>> mylist[-3]
```

运行结果：

```
6
```

它也能用于列表中的切片：

```
>>> mylist[-6:-1]
```

结果：

```
[3, 4, 5, 6, 7]
```

#### 16. 如何以就地操作方式打乱一个列表的元素？

为了达到这个目的，我们从random模块中导入shuffle()函数。

```
>>> from random import shuffle
>>> shuffle(mylist)
>>> mylist
```

运行结果：

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

#### 17. 解释 Python 中的 join()和 split()函数

Join()能让我们将指定字符添加至字符串中。

```
>>> ','.join('12345')
```

运行结果：

```
'1,2,3,4,5'
```

Split()能让我们用指定字符分割字符串。

```
>>> '1,2,3,4,5'.split(',')
```

运行结果：

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

#### 18. Python 区分大小写吗？

如果能区分像 myname 和 Myname 这样的标识符，那么它就是区分大小写的。也就是说它很在乎大写和小写。我们可以用 Python 试一试：

```
>>> myname='Ayushi'
>>> Myname
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
```

运行结果:

```
Myname
NameError: name 'Myname' is not defined
```

### 19. Python 中的标识符长度能有多长?

在 Python 中, 标识符可以是任意长度。此外, 我们在命名标识符时还必须遵守以下规则:

1. 只能以下划线或者 A-Z/a-z 中的字母开头
2. 其余部分可以使用 A-Z/a-z/0-9
3. 区分大小写
4. 关键字不能作为标识符, Python中共有如下关键字:

and	def	False	import	not	True
as	del	finally	in	or	try
assert	elif	for	is	pass	while
break	else	from	lambda	print	with
class	except	global	None	raise	yield
continue	exec	if	nonlocal	return	

### 20. 怎么移除一个字符串中的前导空格?

字符串中的前导空格就是出现在字符串中第一个非空格字符前的空格。我们使用方法 `lstrip()` 可以将它从字符串中移除。

```
>>> '  Ayushi '.lstrip()
```

结果:

```
'Ayushi  '
```

可以看到, 该字符串既有前导字符, 也有后缀字符, 调用 `lstrip()` 去除了前导空格。如果我们想除后缀空格, 就用 `rstrip()` 方法。

```
>>> '  Ayushi '.rstrip()
```

结果:

```
'  Ayushi'
```

## 21. 怎样将字符串转换为小写?

我们使用 `lower()` 方法。

```
>>> 'AyuShi'.lower()
```

结果:

```
'ayushi'
```

使用 `upper()` 方法可以将其转换为大写。

```
>>> 'AyuShi'.upper()
```

结果:

```
'AYUSHI'
```

另外, 使用 `isupper()` 和 `islower()` 方法检查字符串是否全为大写或小写。

```
>>> 'AyuShi'.isupper()
```

```
False
```

```
>>> 'AYUSHI'.isupper()
```

```
True
```

```
>>> 'ayushi'.islower()
```

```
True
```

```
>>> '@yu$hi'.islower()
```

```
True
```

```
>>> '@YU$HI'.isupper()
```

```
True
```

那么，像@和\$这样的字符既满足大写也满足小写。

istitle()能告诉我们一个字符串是否为标题格式。

```
>>> 'The Corpse Bride'.istitle()
True
```

## 22. Python 中的 pass 语句是什么？

在用 Python 写代码时，有时可能还没想好函数怎么写，只写了函数声明，但为了保证语法正确，必须输入一些东西，在这种情况下，我们会使用 pass 语句。

```
>>> def func(*args):
        pass
>>>
```

同样，break语句能让我们跳出循环。

```
>>> for i in range(7):
        if i==3: break
print(i)
```

结果：

```
0
1
2
```

最后，continue语句能让我们跳到下个循环。

```
>>> for i in range(7):
        if i==3: continue
print(i)
```

结果:

```
0
1
2
4
5
6
```

### Q 23. Python中的闭包是什么?

当一个嵌套函数在其外部区域引用了一个值时，该嵌套函数就是一个闭包。其意义就是会记录这个值。

```
>>> def A(x):
    def B():
        print(x)
    return B
>>> A(7)()
```

结果:

```
7
```

### 24. 解释一下 Python 中的//, %和 \*\* 运算符

//运算符执行地板除法（向下取整除），它会返回整除结果的整数部分。

```
>>> 7//2
3
```

这里整除后会返回3.5。

同样地，**执行取幂运算**。a b会返回a的b次方。

```
>>> 2**10
1024
```



最后，%执行取模运算，返回除法的余数。

```
>>> 13%7
6
>>> 3.5%1.5
0.5
```

## 25. 在 Python 中有多少种运算符？解释一下算数运算符。

在 Python 中，我们有 7 种运算符：算术运算符、关系运算符、赋值运算符、逻辑运算符、位运算符、成员运算符、身份运算符。

我们有 7 个算术运算符，能让我们对数值进行算术运算：

1.加号 (+)，将两个值相加

```
>>> 7+8
15
```

2.减号 (-)，将第一个值减去第二个值

```
>>> 7-8
-1
```

3.乘号 (\*)，将两个值相乘

```
>>> 7*8
56
```

4.除号 (/)，用第二个值除以第一个值

```
>>> 7/8
0.875
>>> 1/1
1.0
```

5.向下取整除、取模和取幂运算，参见上个问题。

## 26. 解释一下 Python 中的关系运算符

关系运算符用于比较两个值。

1.小于号 (<)，如果左边的值较小，则返回True。

```
>>> 'hi'<'Hi'
False
```

2.大于号 (>) , 如果左边的值较大, 则返回True。

```
>>> 1.1+2.2>3.3
True
```

3.小于等于号 (<=) , 如果左边的值小于或等于右边的值, 则返回True。

```
>>> 3.0<=3
True
```

4.大于等于号 (>=) , 如果左边的值大于或等于右边的值, 则返回True。

```
>>> True>=False
True
```

5. 等于号 (==) , 如果符号两边的值相等, 则返回True。

```
>>> {1,3,2,2}=={1,2,3}
True
```

6. 不等于号 (!=) , 如果符号两边的值不相等, 则返回True。

```
>>> True!=0.1
True
>>> False!=0.1
True
```

## 27. 解释一下 Python 中的赋值运算符

这在 Python 面试中是个重要的面试问题。

我们将所有的算术运算符和赋值符号放在一起展示:

```
>>> a=7
>>> a+=1
>>> a
8

>>> a-=1
>>> a
7

>>> a*=2
>>> a
14

>>> a/=2
>>> a
7.0

>>> a**=2
>>> a
49
```

```
>>> a//=3
>>> a
16.0

>>> a%=4
>>> a
0.0
```

## 28. 解释一下 Python 中的逻辑运算符

Python 中有 3 个逻辑运算符：and，or，not。

```
>>> False and True
False

>>> 7<7 or True
True

>>> not 2==2
False
```

## 29. 解释一下 Python 中的成员运算符

通过成员运算符'in'和'not in'，我们可以确认一个值是否是另一个值的成员。

```
>>> 'me' in 'disappointment'
True

>>> 'us' not in 'disappointment'
True
```

### 30. 解释一下 Python 中的身份运算符

这也是一个在 Python 面试中常问的问题。

通过身份运算符'is'和'is not'，我们可以确认两个值是否相同。

```
>>> 10 is '10'
False

>>> True is not False
True
```

### 31. 讲讲 Python 中的位运算符

该运算符按二进制位对值进行操作。

1. 与 (&) ， 按位与运算符：参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0

```
>>> 0b110 & 0b010
2
```

2. 或 (|) ， 按位或运算符：只要对应的二个二进位有一个为1时，结果位就为1。

```
>>> 3|2
3
```

4. 取反 (~) ， 按位取反运算符：对数据的每个二进制位取反,即把1变为0,把0变为1

```
>>> ~2
-3
```

5. 左位移 (<<) ， 运算数的各二进位全部左移若干位，由 << 右边的数字指定了移动的位数，高位丢弃，低位补0

```
>>> 1<<2
4
```

6.右位移 (>>)，把">>"左边的运算数的各二进制位全部右移若干位，>> 右边的数字指定了移动的位数

```
>>> 4>>2
1
```

### 32. 在 Python 中如何使用多进制数字？

我们在 Python 中，除十进制外还可以使用二进制、八进制和十六进制。

二进制数字由 0 和 1 组成，我们使用 0b 或 0B 前缀表示二进制数。

```
>>> int(0b1010)
10
```

2.使用bin()函数将一个数字转换为它的二进制形式。

```
>>> bin(0xf)
'0b1111'
```

3.八进制数由数字 0-7 组成，用前缀 0o 或 0O 表示 8 进制数。

```
>>> oct(8)
'0o10'
```

4.十六进数由数字 0-15 组成，用前缀 0x 或者 0X 表示 16 进制数。

```
>>> hex(16)
'0x10'

>>> hex(15)
'0xf'
```

### 33. 怎样获取字典中所有键的列表？

使用 keys() 获取字典中的所有键

```
>>> mydict={'a':1,'b':2,'c':3,'e':5}
>>> mydict.keys()
dict_keys(['a', 'b', 'c', 'e'])
```

### 34. 为何不建议以下划线作为标识符的开头

因为 Python 并没有私有变量的概念，所以约定速成以下划线为开头来声明一个变量为私有。所以如果你不想让变量私有，就不要使用下划线开头。

### Q 35. 怎样声明多个变量并赋值?

一共有两种方式:

```
>>> a,b,c=3,4,5 #This assigns 3, 4, and 5 to a, b, and c respectively
>>> a=b=c=3 #This assigns 3 to a, b, and c
```

### 36. 元组的解封装是什么?

首先我们来看解封装:

```
>>> mytuple=3,4,5
>>> mytuple
(3, 4, 5)
```

这将 3, 4, 5 封装到元组 mytuple 中。

现在我们将这些值解封装到变量 x, y, z 中:

```
>>> x,y,z=mytuple
>>> x+y+z
```

得到结果12。

# 常用 Python 模版库大全

## 核心模块

- 1.1. 介绍
- 1.2. `__builtin__` 模块
- 1.3. `exceptions` 模块
- 1.4. `os` 模块
- 1.5. `os.path` 模块
- 1.6. `stat` 模块
- 1.7. `string` 模块
- 1.8. `re` 模块
- 1.9. `math` 模块
- 1.10. `cmath` 模块
- 1.11. `operator` 模块
- 1.12. `copy` 模块
- 1.13. `sys` 模块
- 1.14. `atexit` 模块
- 1.15. `time` 模块
- 1.16. `types` 模块
- 1.17. `gc` 模块

## 更多标准模块

- 2.1. 概览
- 2.2. `fileinput` 模块
- 2.3. `shutil` 模块
- 2.4. `tempfile` 模块
- 2.5. `StringIO` 模块
- 2.6. `cStringIO` 模块
- 2.7. `mmap` 模块
- 2.8. `UserDict` 模块
- 2.9. `UserList` 模块
- 2.10. `UserString` 模块
- 2.11. `traceback` 模块
- 2.12. `errno` 模块
- 2.13. `getopt` 模块
- 2.14. `getpass` 模块
- 2.15. `glob` 模块
- 2.16. `fnmatch` 模块
- 2.17. `random` 模块
- 2.18. `whrandom` 模块
- 2.19. `md5` 模块

2.20. sha 模块

2.21. crypt 模块

2.22. rotor 模块

2.23. zlib 模块

2.24. code 模块

线程和进程

3.1. 概览

3.2. threading 模块

3.3. Queue 模块

3.4. thread 模块

3.5. commands 模块

3.6. pipes 模块

3.7. popen2 模块

3.8. signal 模块

数据表示

4.1. 概览

4.2. array 模块

4.3. struct 模块

4.4. xdrlib 模块

4.5. marshal 模块

4.6. pickle 模块

4.7. cPickle 模块

4.8. copy\_reg 模块

4.9. pprint 模块

4.10. repr 模块

4.11. base64 模块

4.12. binhex 模块

4.13. quopri 模块

4.14. uu 模块

4.15. binascii 模块

文件格式

5.1. 概览

5.2. xmllib 模块

5.3. xml.parsers.expat 模块

5.4. sgmlib 模块

5.5. htmlib 模块

5.6. htmlentitydefs 模块

5.7. formatter 模块

5.8. ConfigParser 模块

5.9. netrc 模块

5.10. shlex 模块

5.11. zipfile 模块

5.12. gzip 模块

邮件和新闻消息处理



- 6.1. 概览
- 6.2. rfc822 模块
- 6.3. mimetools 模块
- 6.4. MimeWriter 模块
- 6.5. mailbox 模块
- 6.6. mailcap 模块
- 6.7. mimetypes 模块
- 6.8. packmail 模块
- 6.9. mimify 模块
- 6.10. multifile 模块

#### 网络协议

- 7.1. 概览
- 7.2. socket 模块
- 7.3. select 模块
- 7.4. asyncore 模块
- 7.5. asynchat 模块
- 7.6. urllib 模块
- 7.7. urlparse 模块
- 7.8. cookie 模块
- 7.9. robotparser 模块
- 7.10. ftplib 模块
- 7.11. gopherlib 模块
- 7.12. httplib 模块
- 7.13. poplib 模块
- 7.14. imaplib 模块
- 7.15. smtplib 模块
- 7.16. telnetlib 模块
- 7.17. nntplib 模块
- 7.18. SocketServer 模块
- 7.19. BaseHTTPServer 模块
- 7.20. SimpleHTTPServer 模块
- 7.21. CGIHTTPServer 模块
- 7.22. cgi 模块
- 7.23. webbrowser 模块

#### 国际化

- 8.1. locale 模块
- 8.2. unicodedata 模块
- 8.3. ucnhash 模块

#### 多媒体相关模块

- 9.1. 概览
- 9.2. imghdr 模块
- 9.3. sndhdr 模块
- 9.4. whatsound 模块
- 9.5. aifc 模块

- 9.6. sunau 模块
- 9.7. sunaudio 模块
- 9.8. wave 模块
- 9.9. audiodev 模块
- 9.10. winsound 模块

#### 数据储存

- 10.1. 概览
- 10.2. anydbm 模块
- 10.3. whichdb 模块
- 10.4. shelve 模块
- 10.5. dbhash 模块
- 10.6. dbm 模块
- 10.7. dumbdbm 模块
- 10.8. gdbm 模块

#### 工具和实用程序

- 11.1. dis 模块
- 11.2. pdb 模块
- 11.3. bdb 模块
- 11.4. profile 模块
- 11.5. pstats 模块
- 11.6. tabnanny 模块

#### 其他模块

- 12.1. 概览
- 12.2. fcntl 模块
- 12.3. pwd 模块
- 12.4. grp 模块
- 12.5. nis 模块
- 12.6. curses 模块
- 12.7. termios 模块
- 12.8. tty 模块
- 12.9. resource 模块
- 12.10. syslog 模块
- 12.11. msvcrt 模块
- 12.12. nt 模块
- 12.13. \_winreg 模块
- 12.14. posix 模块

#### 执行支持模块

- 13.1. dospath 模块
- 13.2. macpath 模块
- 13.3. ntpath 模块
- 13.4. posixpath 模块
- 13.5. strop 模块
- 13.6. imp 模块
- 13.7. new 模块

13.8. pre 模块  
13.9. sre 模块  
13.10. py\_compile 模块  
13.11. compileall 模块  
13.12. ihooks 模块  
13.13. linecache 模块  
13.14. macurl2path 模块  
13.15. nturl2path 模块  
13.16. tokenize 模块  
13.17. keyword 模块  
13.18. parser 模块  
13.19. symbol 模块  
13.20. token 模块  
其他模块  
14.1. 概览  
14.2. pycldr 模块  
14.3. filecmp 模块  
14.4. cmd 模块  
14.5. rexec 模块  
14.6. Bastion 模块  
14.7. readline 模块  
14.8. rlcompleter 模块  
14.9. statvfs 模块  
14.10. calendar 模块  
14.11. sched 模块  
14.12. statcache 模块  
14.13. grep 模块  
14.14. dircache 模块  
14.15. dircmp 模块  
14.16. cmp 模块  
14.17. cmpcache 模块  
14.18. util 模块  
14.19. soundex 模块  
14.20. timing 模块  
14.21. posixfile 模块  
14.22. bisect 模块  
14.23. knee 模块  
14.24. tzparse 模块  
14.25. regex 模块  
14.26. reesub 模块  
14.27. reconvert 模块  
14.28. regex\_syntax 模块  
14.29. find 模块

# 在 IIS 下配置 Python 的方法

让 IIS 支持 Python:

环境 IIS6 + Python2.5, 其他版本都差不多。

新建 Web 服务扩展, py, 要求的文件填写 C:\Python25\python.exe %s %s。网站>属性>主目录配置>映射>添加。扩展名: .py, 可执行文件填写 C:\Python25\python.exe %s %s。确定, 怎么出错? 改成"C:\Python25\python.exe" %s %s, 哈哈。可以了, 测试一下。

```
1print('Status: 200 OK')
2print('Content-Type: text/html')
3print('')
4print('
```

## Hello, Python

```
')

```

浏览器访问一下, 正常则成功了。

# Python3 中 Random 的实例教程

Python3 中 Random 的实例教程

1.random.random():

会随机生成 0-1 之间的小数

2.random.uniform(min,max):

会随机生成 min - max 之间的小数，其中 min 和 max 的位置可以互换而不会报错：

3.random.randint(min,max):

随机生成 min - max 之间的整数，如果 min > max 会报错：

4.random.choice(元祖/列表/range()/字符串):

会从给定的元祖/列表/range()/字符串 中随机挑选出一个元素：（由于该操作不会对给定对象中的元素进行修改，所以对象类型可以是不可变类型，例如元祖和字符串）：

5.random.randrange(min,max,tap\_num):

会在 min - max 之间随机产生一个数，其中以 tap\_num 作为选取数字的间隔：（这样可以选取某一范围内的奇数和偶数）：

6.random.sample(元祖/列表/字符串/range,num):

会从给定对象的所有元素中随机选取 num 个元素：

7.random.shuffle(list (可变变量)):

shuffle: “洗牌”:

会对给定参数对象的所有元素的位置进行随机变动，就像洗牌一样:

由于 **shuffle** 会改变对象的值，所以对象的类型只能是可变类型，像元祖和字符串类型的变量就不能对其进行 **shuffle** 操作，否则会报错:

-----

# python 中关于单行注释、多行注释以及变量、类型基础知识用法

python 中关于单行注释、多行注释以及变量、类型使用方法

## 1、注释

单行注释，使用#，#号后面的都是注释，例如

```
#我是单行注释
```

```
print("Hello Python world")
```

多行注释：开始和结束用三个单引号扩起来

```
'''
```

```
我是多行注释
```

```
我是多行注释
```

```
我是多行注释
```

```
'''print("Hello Python world")
```

多行注释：开始和结束用三个双引号扩起来

```
"""
```

```
我是多行注释
```

```
我是多行注释
```

```
我是多行注释
```

```
"""
```

```
print("Hello Python world")
```

注意：单引号和双引号混合注释是不可以的哟，只能是开始结束三个单引号或者三个双引号，而不能是开始（结束）三个单引号+结束（开始）三个双引号。

## 2、变量

变量是用来存储【值】的。python 中的变量【不】需要声明类型

```
#声明变量 message
```

```
message="我是一条信息"
```

```
print(message)
```

```
#第二次默认重新赋值
```

```
message="我是重新赋值的信息"
```

```
print(message)
```

打印结果：

我是一条信息

我是重新赋值的信息

### 3、数据类型

虽然 `python` 声明变量时，不需要直接定义类型，但实际数据还是分类型处理的，下面就重要数据类型进行整理。

字符串类型

字符串就是一系列字符串。在 `python` 中用引号括起来的都是字符串，括号可以是单引号，或者双引号。

```
#双引号
```

```
strA="This is a string"
```



#单引号

```
strB='This is also a string'
```

#双引号内可包含单引号

```
strC="This is also a 'string'"
```

#单引号内也可以包含双引号

```
strD='This is also a "string"'
```

数字

python 支持四种数字类型:

int (有符号整型)

long (长整型)

float (浮点类型)

complex (复数)

长整型一般数字后面紧跟大写 L 来区分，尽量不要使用小写的 l

复数是有实数部分和虚数部分构成，可以用 `a+bj` 或者 `complex(a,b)` 表示，复数的实数 `a` 和虚数 `b` 都是浮点型

## 列表

List（列表）是 python 中使用最频繁的数据类型，用[]表述,是有序集合，后面会有专题介绍。

#我是列表

```
list=[521,'mark']
```

## 元组

类似于列表，用()标书，不能进行二次赋值，可以看做是一个只读列表

#我是元组

```
tuple=(521,'mark')
```

## 字典

通过键来取值，是无序的

#我是字典

dictionary={'name':'mark','age':18}

---

# Python3 中关于字典和列表以及指定元素排序方法

## 举例说明

中文源码网在本教程中举例说明：**Python3** 中关于字典和列表以及指定元素排序方法

### 1.字典排序

按 value 排序 d1 = {"name":"python","bank":"icbc","country":china}

# reverse 是否倒序,x[1]代表 value,x[0]为 key

d1 = sorted(d1.items(),lambda x: x[1],reverse=True)

按 key 排序 d1 = {"name":"python","bank":"icbc","country":china}

d1 = sorted(d1.items())列表排序 不去重排序 l = [3,2,4,5]

l = sorted(l) # 升序

# l = sorted(l,reverse=True) # 降序

去重排序 l = [3,2,3,5,1]

l = set(l) # 使用 set 集合去重

l = list(l) #

l = sorted(l) # 排序字典列表排序

# 以 age 升序排序

L = [

    {"name":"python","age":12},

    {"name":"ghj","age":10},

    {"name":"java","age":17}

]

L = sorted(L,key=lambda x: x["age"])

### 2.根据指定列表的元素顺序进行排序

用途：对于取出 MySQL 与 MongoDB 的数据时，我们常常需要对数据进行位置更换顺序，此时此种排序就能很好的解决我们的问题，对于下面列子中的 sortList 的数据我们可以使用 Redis 进行存储

# 指定列表，假设存储的是 curList 中的 ID 列表

sortList = ["4","3","5","2","1"]

# 当前列表

curList = [{"id":"1","province":"河南"}, {"id":"2","province":"河北"}, {"id":"3","湖南"}, {"id":"4","province":"湖北"}, {"id":"5","province":"江西"}]

# 根据指定列表中的 ID 顺序，对当前列表进行排序

```
curList = sorted(curList,key = lambda item:sortList.index(item["id"]))
print(curList)
# [{ "id": "4", "province": "湖北"}, { "id": "3", "province": "湖南"}, { "id": "5", "province": "江西"}, { "id": "2", "province": "河北"}, { "id": "1", "province": "河南"}]
```

# Python3 中的 type 和 object 用法

中文源码网举例说明：Python 中的 type 和 object 的用法

type 用法举例：

```
a = 1
b = "abc"
print("type a: {}".format(type(a)))
print("type int: {}".format(type(int)))
print("type b: {}".format(type(b)))
print("type str: {}".format(type(str)))
```

打印结果如下：

```
type a:
type int:
type b:
type str:
```

在 python 中是一切皆对象的，类其实也是对象，首先 type 生成了这个对象，又生成了 1 这个对象，type --> int --> 1

同样，type 生成了这个对象,又生成了"abc"这个对象，type --> str--> "abc"，即 type --> 生成类对象 -->对象

object 所有类的最顶层基类是 object

```
print("int 的基类是: {}".format(int.__bases__))
print("str 的基类是: {}".format(str.__bases__))
```

打印结果如下：

```
int 的基类是: (,)
str 的基类是: (,)
```

和的基类都是 即：object 是最顶层的基类

type 与 object 的关系(type 的基类是 object,object 是 type 生成的,object 的基类为空)

```
print("type 的基类是: {}".format(type.__bases__))
print("type object: {}".format(type(object)))
print("object 的基类是: {}".format(object.__bases__))
```

打印结果如下：

```
type 的基类是: (,)
type object:
object 的基类是: ()
```

# Python3 中的 type 和 object 用法

中文源码网举例说明：Python 中的 type 和 object 的用法

type 用法举例：

```
a = 1
b = "abc"
print("type a: {}".format(type(a)))
print("type int: {}".format(type(int)))
print("type b: {}".format(type(b)))
print("type str: {}".format(type(str)))
```

打印结果如下：

```
type a:
type int:
type b:
type str:
```

在 python 中是一切皆对象的，类其实也是对象，首先 type 生成了这个对象，又生成了 1 这个对象，type --> int --> 1

同样，type 生成了这个对象,又生成了"abc"这个对象，type --> str--> "abc"，即 type --> 生成类对象 -->对象

object 所有类的最顶层基类是 object

```
print("int 的基类是: {}".format(int.__bases__))
print("str 的基类是: {}".format(str.__bases__))
```

打印结果如下：

```
int 的基类是: (,)
str 的基类是: (,)
```

和的基类都是 即：object 是最顶层的基类

type 与 object 的关系(type 的基类是 object,object 是 type 生成的,object 的基类为空)

```
print("type 的基类是: {}".format(type.__bases__))
print("type object: {}".format(type(object)))
print("object 的基类是: {}".format(object.__bases__))
```

打印结果如下：

```
type 的基类是: (,)
type object:
object 的基类是: ()
```

# python3 中 bs4.FeatureNotFound 提示报

## 错的处理办法

一般 pip 和 pip2 对应的是 python2.x, pip3 对应的是 python3.x 的版本, python2 和 python3 的模块是独立的, 不能混用, 混用会出问题。所以命令行通过 python3 的 pip: pip3 安装解析器:

操作步骤: 执行 cmd 进入 python 的 pip3.exe 文件夹后, 执行: pip3 install lxml  
安装即可。



# python 安装 requests 的步骤

python 代码中在引入 requests 时，提示下面的错误代码：

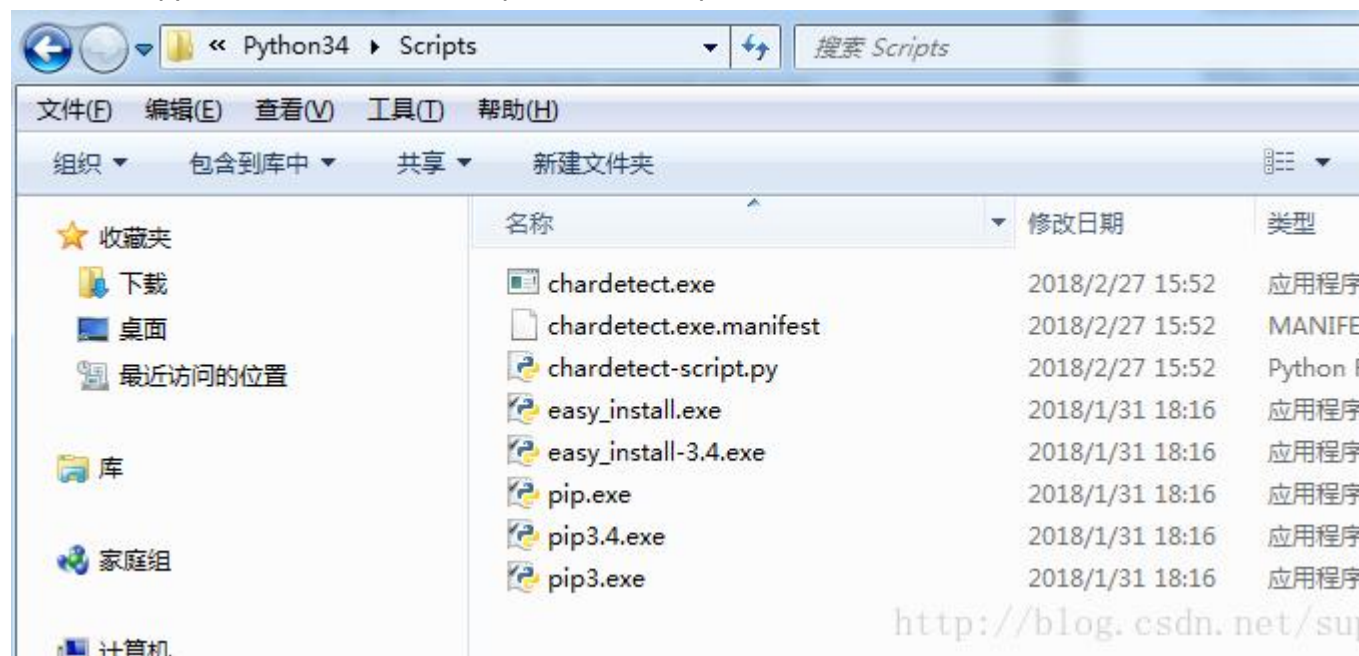
```
import requests
```

```
ModuleNotFoundError: No module named 'requests'
```

原因如下：未安装 requests

解决办法：安装 requests 的步骤如下

可以切换到 python 的安装目录找到 script 文件夹 example:



进入 cmd 窗口切换到上面的目录直接运营下面两个命令中的一个

1. > Path\pip install requests
2. > Path\easy\_install.exe requests

需要注意的是：Scripts 安装 python 时的具体路径别搞错了。

下图中 1 的路径是错误的，所以提示：pip 不是外部或内部命令。

路径 2 是正确的，进入到 pip.exe 所在文件夹为当前路径，然后在输入命令安装即可：pip install requests

（本教程为中文源码网 [www.zwyuanma.com](http://www.zwyuanma.com) 小编亲自安装体验）

```
管理员: C:\Windows\system32\cmd.exe
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>cd C:\Users\Administrator\AppData\Local\Programs\Python

C:\Users\Administrator\AppData\Local\Programs\Python>pip install requests
'pip' 不是内部或外部命令，也不是可运行的程序
或批处理文件。 ①

C:\Users\Administrator\AppData\Local\Programs\Python>cd C:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\Scripts ②

C:\Users\Administrator\AppData\Local\Programs\Python\Python37-32\Scripts>pip install requests
Collecting requests
  Downloading https://files.pythonhosted.org/packages/65/47/7e02164a2a3db50ed6d8a6ab1d6d60b69c4c3fdf57a284257925dfc12bda/requests-2.19.1-py2.py3-none-any.whl (91kB)
    44% |#####| 40kB 113kB/s eta 0:00:
    55% |#####| 51kB 127kB/s eta 0:
    66% |#####| 61kB 152kB/s et
    77% |#####| 71kB 143kB/s
    89% |#####| 81kB 163
    100% |#####| 92kB
173kB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading https://files.pythonhosted.org/packages/16/1f/50d729c104b21c1042aa51560da6141d1cab476ba7015d92b2111c8db841/certifi-2018.8.13-py2.py3-none-any.whl (146kB)
    41% |#####| 61kB 370kB/s eta 0:00:0
    48% |#####| 71kB 393kB/s eta 0:00
    55% |#####| 81kB 417kB/s eta 0:
    62% |#####| 92kB 465kB/s eta
    69% |#####| 102kB 471kB/s
    76% |#####| 112kB 478kB/
    83% |#####| 122kB 324k
    90% |#####| 133kB 31
    97% |#####| 143kB
    100% |#####| 153k
B 494kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl (133kB)
    38% |#####| 51kB 1.3MB/s eta 0:00:01
    46% |#####| 61kB 1.5MB/s eta 0:00:
    53% |#####| 71kB 1.4MB/s eta 0:
    61% |#####| 81kB 1.6MB/s eta
    69% |#####| 92kB 1.5MB/s e
    76% |#####| 102kB 1.4MB/
    84% |#####| 112kB 1.4MB/
```

# 初学习 Python 常见错误

## 1. 忘记写冒号

在 if、elif、else、for、while、class、def 语句后面忘记添加 ":"

```
if spam == 42
    print('Hello!')
```

导致: SyntaxError: invalid syntax

## 2. 误用 “=” 做等值比较

“=” 是赋值操作，而判断两个值是否相等是 “==”

```
if spam = 42:
    print('Hello!')
```

导致: SyntaxError: invalid syntax

## 3. 使用错误的缩进

Python 用缩进区分代码块

常见的错误用法:

导致: IndentationError: unexpected indent.

同一个代码块中的每行代码都必须保持一致的缩进量

```
if spam == 42:
    print('Hello!')
print('Howdy!')
```

导致: IndentationError: unindent does not match any outer indentation level.

代码块结束之后缩进恢复到原来的位置

```
if spam == 42:
    print('Hello!')
```

导致: IndentationError: expected an indented block

“:” 后面要使用缩进

## 4. 变量没有定义

```
if spam == 42:  
    print('Hello!')
```

导致: `NameError: name 'spam' is not defined`

## 5. 获取列表元素索引位置忘记调用 `len` 方法

通过索引位置获取元素的时候

忘记使用 `len` 函数获取列表的长度。

```
spam = ['cat', 'dog', 'mouse']  
for i in range(spam):  
    print(spam[i])
```

导致: `TypeError: range() integer end argument expected, got list.`

正确的做法是:

```
spam = ['cat', 'dog', 'mouse']  
for i in range(len(spam)):  
    print(spam[i])
```

当然, 更 Pythonic 的写法是用 `enumerate`

```
spam = ['cat', 'dog', 'mouse']  
for i, item in enumerate(spam):  
    print(i, item)
```

## 6. 修改字符串

字符串一个序列对象, 支持用索引获取元素

但它和列表对象不同, 字符串是不可变对象, 不支持修改。

```
spam = 'I have a pet cat.'  
spam[13] = 'r'  
print(spam)
```

导致: `TypeError: 'str' object does not support item assignment`

正确地做法应该是:

```
spam = 'I have a pet cat.'  
spam = spam[:13] + 'r' + spam[14:]  
print(spam)
```

## 7. 字符串与非字符串连接

```
num_eggs = 12  
print('I have ' + num_eggs + ' eggs.')
```

导致: `TypeError: cannot concatenate 'str' and 'int' objects`

字符串与非字符串连接时

必须把非字符串对象强制转换为字符串类型

```
num_eggs = 12  
print('I have ' + str(num_eggs) + ' eggs.')
```

或者使用字符串的格式化形式

```
num_eggs = 12  
print('I have %s eggs.' % (num_eggs))
```

## 8. 使用错误的索引位置

```
spam = ['cat', 'dog', 'mouse']  
print(spam[3])
```

导致: `IndexError: list index out of range`

列表对象的索引是从 0 开始的

第 3 个元素应该是使用 `spam[2]` 访问

## 9. 字典中使用不存在的键

```
spam = {'cat': 'Zophie', 'dog': 'Basil', 'mouse': 'Whiskers'}  
print('The name of my pet zebra is ' + spam['zebra'])
```

在字典对象中访问 `key` 可以使用 `[]`,

但是如果该 `key` 不存在

就会导致: `KeyError: 'zebra'`

正确的方式应该使用 `get` 方法:



```
spam = {'cat': 'Zophie', 'dog': 'Basil', 'mouse': 'Whiskers'}  
print('The name of my pet zebra is ' + spam.get('zebra'))
```

key 不存在时，get 默认返回 None

## 10. 用关键字做变量名

```
class = 'algebra'
```

导致: `SyntaxError: invalid syntax`

在 Python 中不允许使用关键字作为变量名。

Python3 一共有 33 个关键字。

```
>>> import keyword  
>>> print(keyword.kwlist)  
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',  
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',  
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'r
```

## 11. 函数中局部变量赋值前被使用

```
someVar = 42  
  
def myFunction():  
    print(someVar)  
    someVar = 100  
  
myFunction()
```

导致: `UnboundLocalError: local variable 'someVar' referenced before assignment`

当函数中有一个与全局作用域中同名的变量时

它会按照 LEGB 的顺序查找该变量，如果在函数内部的局部作用域中也定义了一个同名的变量，那么就不再到外部作用域查找了。

因此，在 myFunction 函数中 someVar 被定义了，所以 print(someVar) 就不再外面查找了，但是 print 的时候该变量还没赋值，所以出现了 UnboundLocalError

## 12. 使用自增 “++” 自减 “--”

```
spam = 0
spam++
```

哈哈，Python 中没有自增自减操作符

如果你是从 C、Java 转过来的话，你可要注意了。

你可以使用 “+=” 来替代 “++”

```
spam = 0
spam += 1
```

## 13. 错误地调用类中的方法

```
class Foo:
    def method1():
        print('m1')
    def method2(self):
        print("m2")

a = Foo()
a.method1()
```

导致：TypeError: method1() takes 0 positional arguments but 1 was given

method1 是 Foo 类的一个成员方法

该方法不接受任何参数，调用 a.method1() 相当于调用 Foo.method1(a)，但 method1 不接受任何参数，所以报错了。

正确的调用方式应该是 Foo.method1()

**注意：**

以上代码都是基于 Python3 的，在 Python2 中即使是同样的代码出现的错误也不尽一样，尤其是最后一个例子。

# Python 字符串处理大全

在 Python 中字符串连接有多种方式，这里简单做个总结，应该会比较全面的了，方便以后查阅。

加号连接

第一种，通过+号的形式：

```
>>> a, b = 'hello', ' world'
>>> a + b
'hello world'
```

逗号连接

第二种，通过,逗号的形式：

```
>>> a, b = 'hello', ' world'
>>> print(a, b)
hello world
```

但是，使用逗号形式要注意一点，就是只能用于 print 打印，赋值操作会生成元组：

```
>>> a, b
('hello', ' world')
```

Python 之禅注：实际上，这不算是字符串连接的一种方式，因为'hello', ' world'会当作一个元组存在，通过解包（unpacking）的方式赋值给变量 a 和 b.

直接连接

第三种，直接连接中间有无空格均可：

```
print('hello' ' world')
print('hello"world')
```

Python 之禅注：这算是 Python 里面的一种语法糖，连续的字符串会自动拼接成一个字符串。在内存中不会存在两个字符串对象。（代码可左右滑动）

```
>>> def x():
```



```
... a = 'a"b'
```

```
...
```

```
>>> dis.dis(x)
```

```
2 0 LOAD_CONST 1 ('ab')
```

```
3 STORE_FAST 0 (a)
```

```
6 LOAD_CONST 0 (None)
```

```
9 RETURN_VALUE
```

最全面的 Python 字符串拼接总结（带注释版）

百分号 %

第四种，使用 % 操作符。

在 Python 2.6 以前，% 操作符是唯一一种格式化字符串的方法，它也可以用于连接字符串。

```
print('%s %s' % ('hello', 'world'))
```

format 函数

第五种，使用 format 方法。

format 方法是 Python 2.6 中出现的一种代替 % 操作符的字符串格式化方法，同样可以用来连接字符串。

```
print('{}{}'.format('hello', ' world'))
```

最全面的 Python 字符串拼接总结（带注释版）

join 函数

第六种，使用 join 内置方法。

字符串有一个内置方法 join，其参数是一个序列类型，例如数组或者元组等。

```
print('-'.join(['aa', 'bb', 'cc']))
```

f-string

第七种，使用 f-string 方式。

Python 3.6 中引入了 Formatted String Literals（字面量格式化字符串），简称 f-string，f-string 是 % 操作符和 format 方法的进化版，使用 f-string 连接字符串的方法和使用 %操作符、format 方法类似。

```
>>> aa, bb = 'hello', 'world'
>>> f'{aa} {bb}'
'hello world'
星号 *
```

第八种，使用\*操作符。

```
>>> aa = 'hello '
>>> aa * 3
'hello hello hello '
Python 之禅注：*操作符其实是一种操作符重载操作，对应的魔术方法是 __mul__
```

```
>>> a = [1]
>>> a*2
[1, 1]
>>> a.__mul__(3)
[1, 1, 1]
小结
```

连接少量字符串时，推荐使用+号操作符。

如果对性能有较高要求，并且 python 版本在 3.6 以上，推荐使用 f-string。例如，如下情况 f-string 可读性比+号要好很多：

```
a = f'姓名: {name} 年龄: {age} 性别: {gender}'

b = '姓名: ' + name + '年龄: ' + age + '性别: ' + gender
```

连接大量字符串时，推荐使用 join 和 f-string 方式，选择时依然取决于你使用的 Python 版本以及对可读性的要求。

曾经做过一个测试，python3.6 中，数据量不大的情况下 +操作甚至比 join 操作还快。

# 适合初学者的 python 爬虫开发案例

## 一、论述

这几个案例以前是给一些想进入 Python 行业的朋友写的，看到大家都比较满意，所以就再次拿了出来，如果你已经开始学 python，对爬虫没有头绪，不妨看看这几个案例，文末更多分享！

## 二、环境准备

Python 3

requests 库、lxml 库、beautifulsoup4 库

pip install XX XX XX 一并安装。

```
C:\Users\K>pip install requests lxml beautifulsoup4
Collecting requests
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
    100% |#####| 92kB 168kB/s
Collecting lxml
  Downloading lxml-4.2.1-cp36-cp36m-win_amd64.whl (3.6MB)
    100% |#####| 3.6MB 19kB/s
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.6.0-py3-none-any.whl (86kB)
    100% |#####| 92kB 15kB/s
Collecting certifi<2017.4.17 (from requests)
  Downloading certifi-2018.1.18-py2.py3-none-any.whl (151kB)
    100% |#####| 153kB 35kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)
    100% |#####| 143kB 28kB/s
Collecting urllib3<1.23,>=1.21.1 (from requests)
  Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
    100% |#####| 133kB 35kB/s
Collecting idna<2.7,>=2.5 (from requests)
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
    100% |#####| 61kB 72kB/s
Installing collected packages: certifi, chardet, urllib3, idna, requests, lxml, beautifulsoup4
Successfully installed beautifulsoup4-4.6.0 certifi-2018.1.18 chardet-3.0.4 idna-2.6 lxml-4.2.1 requests-2.18.4 urllib3-1.22
You are using pip version 9.0.1, however version 9.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

## 三、Python 爬虫小案例

### 1、获取本机的公网 IP 地址

利用 python 的 requests 库+公网上查 IP 的接口，自动获取 IP 地址

```
import requests
r = requests.get("http://2017.ip138.com/ic.asp")
r.encoding = r.apparent_encoding #使用requests的字符编码智能分析，
print(r.text)
# 你还可以使用正则匹配re模块提取出IP
import re
print(re.findall("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",r.text))
```

## 2、利用百度的查找接口，Python 编写 url 采集工具

需要用到 requests 库、BeautifulSoup 库，观察百度搜索结构的 URL 链接规律，绕过百度搜索引擎的反爬虫机制的方法为在程序中设置 User-Agent 请求头。

例如第一页的url链接参数pn=0

第二页的url链接参数pn=10

依次类推。这里，我们使用css选择器路径提取数据。

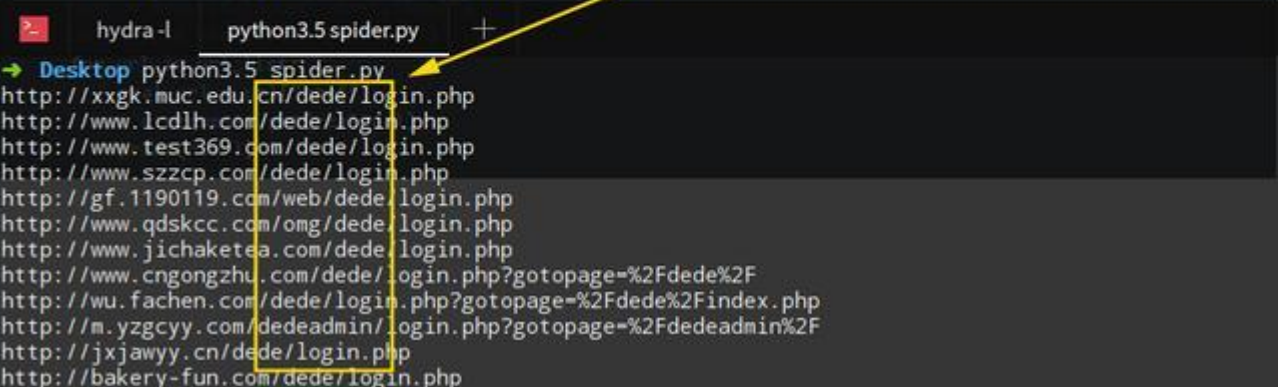
Python 源代码：

```
import requests
from bs4 import BeautifulSoup
# 设置User-Agent头，绕过百度搜索引擎的反爬虫机制
headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0'}
# 注意观察百度搜索结构的URL链接规律，例如第一页pn=0，第二页pn=10... 依次类推
for i in range(0,100,10):
    bd_search = "https://www.baidu.com/s?wd=inurl:/dede/login.php?pn=%s" % str(i)
    r = requests.get(bd_search,headers=headers)
    soup = BeautifulSoup(r.text,"lxml")
    # 下面的select使用了css选择器路径提取数据
    url_list = soup.select(".t > a")
    for url in url_list:
        real_url = url["href"]
        r = requests.get(real_url)
        print(r.url)
```

Python 语言编写好程序后，利用关键词 inurl:/dede/login.php 来批量提取某网 cms 的后台地址：

```
import requests
from bs4 import BeautifulSoup

headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0'}
for i in range(0,100,10):
    bd_search = "https://www.baidu.com/s?wd=inurl:/dede/login.php?pn=%s" % str(i)
    r = requests.get(bd_search,headers=headers)
```



## 3、利用 Python 打造搜狗壁纸自动下载爬虫

搜狗壁纸的地址是 json 格式，所以用 json 库解析这组数据，爬虫程序存放图片的磁盘路径改成欲存图片的路径就可以了。

```
import requests
import json
#下载图片
url = "http://pic.sogou.com/pics/channel/getAllRecomPicByTag.jsp?category=
%E5%A3%81%E7%BA%B8&tag=%E6%B8%B8%E6%88%8F&start=0&len=15&width=1366&height=768"
r = requests.get(url)
data = json.loads(r.text)
for i in data["all_items"]:
    img_url = i["pic_url"]
    # 下面这行里面的路径改成你自己想要存放图片的目录路径即可
    with open("/home/evilk0/Desktop/img/%s" % img_url[-10:]+".jpg","wb") as f:
        r2 = requests.get(img_url)
        f.write(r2.content)
    print("下载完毕:",img_url)
```

效果图：

#### 4、Python 自动填写问卷调查



```
import requests
import random

url = "https://www.wjx.cn/joinnew/processjq.ashx?submitttype=1&curID=21581199&t=1521463484600&starttime=2018%2F3%2F19%2020%3A44%3A30&rn=990598061.78751211"
data = {
    "submitdata" : "1%s}2%s}3%s}4%s}5%s}6%s}7%s}8%s}9%s}10%s"
}
header = {
    "User-Agent" : "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)",
    "Cookie": ".ASPXANONYMOUS=iBuvxgz20wEkAAAAZGY4MDE1MjctNWU4Ni00MDUwLjgwYjQtMjFhMmZhMDE2MTA3h_bb3gNw4XRPsyh-qPh4XWlmfJ41; spiderregkey=baidu.com%c2%a7%e7%9b%b4%e8%be%be%c2%a71; UM_distinctid=1623e28d4df22d-08d0140291e4d5-102c1709-100200-1623e28d4e1141; _umdata=535523100CBE37C329C8A3EEEEEE289B573446F594297CC3BB3C355F09187F5ADC C492EBB07A9CC65CD43AD3E795C914CD57017EE3799E92F0E2762C963EF0912; WjxUser=UserName=17750277425&Type=1; LastCheckUpdateDate=1; LastCheckDesign=1; DeleteQCookie=1; _cnzz_CV4478442=%E7%94%A8%E6%88%B7%E7%89%88%E6%9C%AC%7C%E5%85%8D%E8%B4%B9%E7%89%88%7C1521461468568; jac21581199=78751211; CNZZDATA4478442=cnzz_eid%3D878068609-1521456533-https%253A%252F%252Fwww.baidu.com%252F%26ntime%3D1521461319; Hm_lvt_21be24c80829bd7a683b2c536fcf520b=1521461287,1521463471; Hm_lpvt_21be24c80829bd7a683b2c536fcf520b=1521463471",
}

for i in range(0,500):
    choice = (
        random.randint(1, 2),
        random.randint(1, 4),
        random.randint(1, 3),
        random.randint(1, 4),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
    )
    data["submitdata"] = data["submitdata"] % choice
    r = requests.post(url=url,headers=header,data=data)
    print(r.text)
    data["submitdata"] = "1%s}2%s}3%s}4%s}5%s}6%s}7%s}8%s}9%s}10%s"
```

与一般网页一样，多次提交数据会要输入验证码，这就是反爬机制。

```
→ Desktop python3.5 ../PycharmProjects/month3/ctf/auto_submit.py
2.2.2.2 利用菜
10 千 /wjsx/join/complete.aspx?q=21581199&JoinID=101421626218&jidx=292
2.2.2.2
10 千 /wjsx/join/complete.aspx?q=21581199&JoinID=101421626130&jidx=293
2.2.2.2
10 千 /wjsx/join/complete.aspx?q=21581199&JoinID=101421626124&jidx=294
2.2.2.2
10 千 /wjsx/join/complete.aspx?q=21581199&JoinID=101421626318&jidx=295
2.2.2.2
7 千 您输入的验证码有误，请重新输入！
2.2.2.2
7 千 您输入的验证码有误，请重新输入！
2.2.2.2
7 千 您输入的验证码有误，请重新输入！
```

如图：



10. 你经常要比预计的玩更长时间的游戏吗？ \*

☐ 经常 ☐ 有时 ☐ 很少

6 JAVG 提交

试填

那么如何绕过验证码的反爬措施？利用 X-Forwarded-For 伪造 IP 地址访问即可，Python 代码如下：



```

import requests
import random

url = "https://www.wjx.cn/joinnew/processjq.ashx?submitttype=
1&curID=21581199&t=1521463484600&starttime=2018%2F3%2F19%2020%3A44%3A30%rn=990598061.78751211"
data = {
    "submitdata" : "1%s}2%s}3%s}4%s}5%s}6%s}7%s}8%s}9%s}10%s"
}
header = {
    "User-Agent" : "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)",
    "Cookie": ".ASPXANONYMOUS=
iBuvxgz20wEkaAAAZGY4MDE1MjctNWU4Ni00MDUwLTgwYjQ0MjFhMmZhMDE2MTA3h_bb3gNw4XRPsyh-qPh4XW1mfJ41;
spiderregkey=baidu.com%2a7%e7%9b%b4%e8%be%be%2a71;
UM_distinctid=1623e28d4df22d-08d0140291e4d5-102c1709-100200-1623e28d4e1141;
_umdata=535523100CBE37C329C8A3EEEE289B573446F594297CC3BB3C355F09187F5ADCC492EBB07A9CC
6SCD43AD3E795C914CD57017EE3799E92F0E2762C963EF0912; WjxUser=UserName=17750277425&Type=1;
LastCheckUpdateDate=1; LastCheckDesign=1; DeleteQCookie=1;
_cnzz_CV4478442=E7%94%A8%E6%88%B7%E7%89%88%E6%9C%AC%7C%E5%85%8D%E8%B4%B9%E7%89%88%7C1521461468568;
_jac21581199=78751211;
CNZZDATA4478442=cnzz_eid%3D878068609-1521456533-https%253A%252F%252Fwww.baidu.com%252F%26ntime%3D1521461319;
Hm_lvt_21be24c80829bd7a683b2c536fcf520b=1521461287,1521463471;
Hm_lpv21be24c80829bd7a683b2c536fcf520b=1521463471",
    "X-Forwarded-For" : "%s"
}

for i in range(0,500):
    choice = (
        random.randint(1, 2),
        random.randint(1, 4),
        random.randint(1, 3),
        random.randint(1, 4),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
        random.randint(1, 3),
    )
    data["submitdata"] = data["submitdata"] % choice
    header["X-Forwarded-For"] = (str(random.randint(1,255))+"."+(str(random.randint(1,255))+"."+(
    str(random.randint(1,255))+"."+(str(random.randint(1,255))
    r = requests.post(url = url,headers=header,data=data)
    print(header["X-Forwarded-For"],r.text)
    data["submitdata"] = "1%s}2%s}3%s}4%s}5%s}6%s}7%s}8%s}9%s}10%s"
    header["X-Forwarded-For"] = "%s"

```

效果：

```

→ Desktop python3.5 ../PycharmProjects/month3/ctf/auto_submit.py
33.39.29.205 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695012&jidx=111
55.97.214.145 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420694940&jidx=112
143.208.147.175 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420694920&jidx=113
116.117.168.166 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420694956&jidx=114
31.207.204.235 10T /wjx/join/complete.aspx?q=21581199&JoinID=10142069512&jidx=115
11.2.90.152 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695128&jidx=116
X-Forwarded-For 11.2.90.152 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695116
伪造的IP 11.2.90.152 10T /wjx/join/complete.aspx?q=21581199&JoinID=10142069515
143.02.231.77 10T /wjx/join/complete.aspx?q=21581199&JoinID=10142069516
147.205.127.251 10T /wjx/join/complete.aspx?q=21581199&JoinID=1014206950
71.74.195.192 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695048
43.103.120.65 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695098&jidx=124
232.98.109.172 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695253&jidx=125
213.52.231.98 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695262&jidx=126
188.156.98.32 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695235&jidx=127
134.203.242.51 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695281&jidx=128
65.247.231.223 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695270&jidx=129
197.192.218.49 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695275&jidx=130
233.141.28.136 10T /wjx/join/complete.aspx?q=21581199&JoinID=101420695175&jidx=131

```

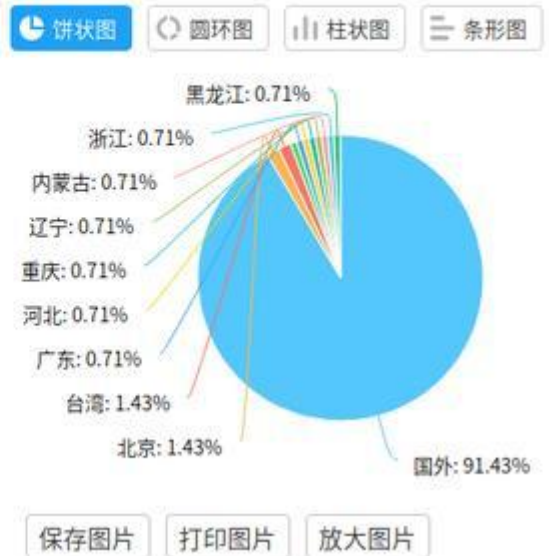
有效的  
问卷调  
查数



来源渠道分析	时间段分析	地理位置分析
--------	-------	--------

各省份填写分布: [查看地图分布情况](#)

省份	数量	百分比	统计	详情
国外	128	91.43%		
北京	2	1.43%		
台湾	2	1.43%		
福建	1	0.71%		
广东	1	0.71%		
河北	1	0.71%		
重庆	1	0.71%		
辽宁	1	0.71%		
内蒙古	1	0.71%		
浙江	1	0.71%		



大学生网络游戏调查问卷[复制]

您好, 这是一份关于大学生玩网络游戏的调查问卷, 希望能占用你两分钟时间帮忙填一下, 非常感谢!

1. 性别: \*

☐ 男 ☐ 女

2. 你的年级: \*

☐ 大一 ☐ 大二 ☐ 大三 ☐ 大四

3. 你经常玩网游吗? \*

☐ 经常 ☐ 有时 ☐ 很少

4. 网络游戏是一种良好的放松方式, 是对智力的一种挑战, 能学习知识。你同意这种观点吗? \*

☐ 完全同意 ☐ 基本同意 ☐ 部分同意 ☐ 不同意

## 5、获取西刺代理上的 IP, 验证这些代理被封禁掉的可能性与延迟时间

可以把 Python 爬取的代理 IP 添加到 proxychain 里面, 就可以进行一般的渗透任务了。这里直接调用了 linux 的系统命令 `ping -c 1 " + ip.string + " | awk 'NR==2{print}' -`, 在

Windows 中运行此程序需要修改倒数第三行 `os.popen` 里的命令，修改为 Windows 能够执行的就可以了。

```
from bs4 import BeautifulSoup
import requests
import os

url = "http://www.xicidaili.com/nn/1"
headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.62 Safari/537.36'}
r = requests.get(url=url, headers=headers)
soup = BeautifulSoup(r.text, "lxml")
server_address = soup.select(".odd > td:nth-of-type(4)")
ip_list = soup.select(".odd > td:nth-of-type(2)")
ports = soup.select(".odd > td:nth-of-type(3)")
for server, ip in zip(server_address, ip_list):
    if len(server.contents) != 1:
        print(server.a.string.ljust(8), ip.string.ljust(20), end='')
    else:
        print("未知".ljust(8), ip.string.ljust(20), end='')
    delay_time = os.popen("ping -c 1 " + ip.string + " | awk 'NR==2{print}' -")
    delay_time = delay_time.read().split("time=")[-1].strip("\r\n")
    print("time = " + delay_time)
```

爬取到的数据如图：



```
root@PC: Desktop
python3.5 ../PycharmProjects/month3/ctf/get_proxies.py
湖北武汉 58.19.81.71 time = 32.8 ms
江苏扬州 114.230.41.99 time = 28.7 ms
福建莆田 110.85.89.145 time = 26.7 ms
江苏常州 114.228.80.38 time = 55.3 ms
河南郑州 122.114.31.177 time =
山东济南 124.128.39.138 time =
北京 42.96.168.79 time = 66.2 ms
湖北武汉 113.57.96.141 time = 51.7 ms
北京 111.155.116.225 time = 51.6 ms
江苏常州 218.93.172.102 time = 32.3 ms
安徽淮北 117.69.231.238 time =
江西 182.86.189.55 time = 35.2 ms
江苏常州 180.116.45.218 time = 32.3 ms
广西南宁 110.73.55.118 time = 45.4 ms
河南 123.101.141.51 time = 79.7 ms
广东广州市番禺区 59.42.42.104 time = 34.5 ms
上海 116.231.37.119 time = 37.4 ms
湖北武汉 113.57.35.147 time = 43.6 ms
江苏苏州 49.70.32.39 time = 39.1 ms
```

# Python 爬虫开发入门及开发技巧大全

## 1、基本抓取网页

get 方法

```
import urllib2
url = "http://www.baidu.com"
response = urllib2.urlopen(url)
print response.read()
```

post 方法

```
import urllib
import urllib2
url = "http://abcde.com"
form = {'name': 'abc', 'password': '1234'}
form_data = urllib.urlencode(form)
request = urllib2.Request(url, form_data)
response = urllib2.urlopen(request)
print response.read()
```

## 2. 使用代理服务器

这在某些情况下比较有用，

比如 IP 被封了，或者比如 IP 访问的次数受到限制等等。

```
import urllib2
proxy_support = urllib2.ProxyHandler({'http': 'http://XX.XX.XX.XX:XXXX'})
opener = urllib2.build_opener(proxy_support, urllib2.HTTPHandler)
urllib2.install_opener(opener)
content = urllib2.urlopen('http://XXXX').read()
```

## 3. Cookies 处理

```
import urllib2, cookielib
cookie_support = urllib2.HTTPCookieProcessor(cockielib.CookieJar())
opener = urllib2.build_opener(cookie_support, urllib2.HTTPHandler)
urllib2.install_opener(opener)
content = urllib2.urlopen('http://XXXX').read()
```

是的没错，如果想同时用代理和 cookie，

那就加入 proxy\_support 然后 opener 改为 ， 如下：

```
opener=urllib2.build_opener(proxy_support, cookie_support, urllib2.HTTPHandl
```

## 4. 伪装成浏览器访问

某些网站反感爬虫的到访，于是对爬虫一律拒绝请求。

这时候我们需要伪装成浏览器，

这可以通过修改 http 包中的 header 来实现：

```
headers = {  
    'User-Agent': 'Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.6  
}  
req = urllib2.Request(  
    url = 'http://secure.verycd.com/signin/*/http://www.verycd.com/',  
    data = postdata,  
    headers = headers  
)
```

## 5、页面解析

对于页面解析最强大的当然是正则表达式，

这个对于不同网站不同的使用者都不一样，就不用过多的说明。



语法	说明	表达式实例	完整匹配的字符串
字符			
一般字符	匹配自身	abc	abc
.	匹配任意除换行符“\n”外的字符。 在DOTALL模式中也能匹配换行符。	a.c	abc
\	转义字符, 使后一个字符改变原来的意思。 如果字符串中有字符*需要匹配, 可以使用\*或者字符集[*]。	a\.c a\\c	a.c a\c
[...]	字符集 ( 字符类 )。对应的位置可以是字符集中任意字符。 字符集中的字符可以逐个列出, 也可以给出范围, 如[abc]或[a-c]。第一个字符如果是^则表示取反, 如[^abc]表示不是abc的其他字符。 所有的特殊字符在字符集中都失去其原有的特殊含义。在字符集中如果要使用]、-或^, 可以在前面加上反斜杠, 或把]、-放在第一个字符, 把^放在非第一个字符。	a[bcd]e	abe ace ade
预定义字符集 ( 可以写在字符集[...]中 )			
\d	数字: [0-9]	a\d.c	a1c
\D	非数字: [^\d]	a\D.c	abc
\s	空白字符: [<空格>\t\r\n\f\v]	a\s.c	a c
\S	非空白字符: [^\s]	a\S.c	abc
\w	单词字符: [A-Za-z0-9_]	a\w.c	abc
\W	非单词字符: [^\w]	a\W.c	a c
数量词 ( 用在字符或(...)之后 )			
*	匹配前一个字符0或无限次。	abc*	ab abccc
+	匹配前一个字符1次或无限次。	abc+	abc abccc
?	匹配前一个字符0次或1次。	abc?	ab abc
{m}	匹配前一个字符m次。	ab{2}c	abbc
{m,n}	匹配前一个字符m至n次。 m和n可以省略: 若省略m, 则匹配0至n次; 若省略n, 则匹配m至无限次。	ab{1,2}c	abc abbc
*? +? ?? {m,n}?	使 * + ? {m,n}变成非贪婪模式。	示例将在下文中介绍。	
边界匹配 ( 不消耗待匹配字符串中的字符 )			
^	匹配字符串开头。 在多行模式中匹配每一行的开头。	^abc	abc
\$	匹配字符串末尾。 在多行模式中匹配每一行的末尾。	abc\$	abc
\A	仅匹配字符串开头。	\Aabc	abc
\Z	仅匹配字符串末尾。	abc\Z	abc
\b	匹配\w和\W之间。	a\b!bc	a!bc
\B	[^\b]	a\Bbc	abc
逻辑、分组			
	代表左右表达式任意匹配一个。 它总是先尝试匹配左边的表达式, 一旦成功匹配则跳过匹配右边的表达式。 如果 没有被包括在()中, 则它的范围是整个正则表达式。	abc def	abc def
(...)	被括起来的表达式将作为分组, 从表达式左边开始每遇到一个分组的左括号'(', 编号+1。 另外, 分组表达式作为一个整体, 可以后接数量词。表达式中的 仅在该组中有效。	(abc){2} a(123 456)c	abccabc a456c
(?P<name>...)	分组, 除了原有的编号外再指定一个额外的别名。	(?P<id>abc){2}	abccabc 1abc1

其次就是解析库了，常用的有两个 lxml 和 BeautifulSoup。

对于这两个库，我的评价是，

都是 HTML/XML 的处理库，Beautifulsoup 纯 python 实现，效率低，

但是功能实用，比如能用通过结果搜索获得某个 HTML 节点的源码；

lxml 语言编码，高效，支持 Xpath。

## 6. 验证码的处理

**碰到验证码咋办？**

这里分两种情况处理：

google 那种验证码，没办法。

**简单的验证码：**字符个数有限，只使用了简单的平移或旋转加噪音而没有扭曲的，

这种还是有可能可以处理的，一般思路是旋转的转回来，噪音去掉，

然后划分单个字符，划分好了以后再通过特征提取的方法(例如 PCA)降维并生成特征库，

然后把验证码和特征库进行比较。

这个比较复杂，这里就不展开了，

具体做法请弄本相关教科书好好研究一下。

## 7. gzip/deflate 支持

现在的网页普遍支持 gzip 压缩，这往往可以解决大量传输时间，

以 VeryCD 的主页为例，未压缩版本 247K，压缩了以后 45K，为原来的 1/5。

这就意味着抓取速度会快 5 倍。

**然而 python 的 urllib/urllib2 默认都不支持压缩**

要返回压缩格式，必须在 request 的 header 里面写明 'accept-encoding'，

然后读取 response 后更要检查 header 查看是否有 'content-encoding' 一项来判断是否需要解码，很繁琐琐碎。

**如何让 urllib2 自动支持 gzip, deflate 呢？**

其实可以继承 BaseHandler 类，

然后 build\_opener 的方式来处理：

```

import urllib2
from gzip import GzipFile
from StringIO import StringIO
class ContentEncodingProcessor(urllib2.BaseHandler):
    """A handler to add gzip capabilities to urllib2 requests"""
    # add headers to requests
    def http_request(self, req):
        req.add_header("Accept-Encoding", "gzip, deflate")
        return req
    # decode
    def http_response(self, req, resp):
        old_resp = resp
        # gzip
        if resp.headers.get("content-encoding") == "gzip":
            gz = GzipFile(
                fileobj=StringIO(resp.read()),
                mode="r"
            )
            resp = urllib2.addinfourl(gz, old_resp.headers, old_resp.url, old_resp.code)
            resp.msg = old_resp.msg
        # deflate
        if resp.headers.get("content-encoding") == "deflate":
            gz = StringIO( deflate(resp.read()) )
            resp = urllib2.addinfourl(gz, old_resp.headers, old_resp.url, old_resp.code) # 'class to add info() and
            resp.msg = old_resp.msg
        return resp
    # deflate support
    import zlib
    def deflate(data): # zlib only provides the zlib compress format, not the deflate format;
        try: # so on top of all there's this workaround:
            return zlib.decompress(data, -zlib.MAX_WBITS)
        except zlib.error:
            return zlib.decompress(data)
    然后就简单了,
    encoding_support = ContentEncodingProcessor
    opener = urllib2.build_opener( encoding_support, urllib2.HTTPHandler )
    #直接用opener打开网页, 如果服务器支持gzip/defalte则自动解压缩
    content = opener.open(url).read()

```

## 8、多线程并发抓取

单线程太慢的话，就需要多线程了，

这里给个简单的线程池模板 这个程序只是简单地打印了 1-10，

但是可以看出是并发的。

虽然说 **Python** 的多线程很鸡肋

但是对于爬虫这种网络频繁型，

还是能一定程度提高效率的。

```

from threading import Thread
from Queue import Queue
from time import sleep
# q是任务队列
# NUM是并发线程总数
# JOBS是有多少任务
q = Queue()
NUM = 2
JOBS = 10
#具体的处理函数，负责处理单个任务
def do_somthing_using(arguments):
    print arguments
#这个是工作进程，负责不断从队列取数据并处理
def working():
    while True:
        arguments = q.get()
        do_somthing_using(arguments)
        sleep(1)
        q.task_done()
#fork NUM个线程等待队列
for i in range(NUM):
    t = Thread(target=working)
    t.setDaemon(True)
    t.start()
#把JOBS排入队列
for i in range(JOBS):
    q.put(i)
#等待所有JOBS完成
q.join()

```

## 9. 总结

阅读 Python 编写的代码感觉像在阅读英语一样，这让使用者可以专注于解决问题而不是去搞明白语言本身。

Python 虽然是基于 C 语言编写，但是摒弃了 C 中复杂的指针，使其变得简单易学。

并且作为开源软件，Python 允许对代码进行阅读，拷贝甚至改进。

这些性能成就了 Python 的高效率，有“人生苦短，我用 Python”之说，是一种十分精彩又强大的语言。

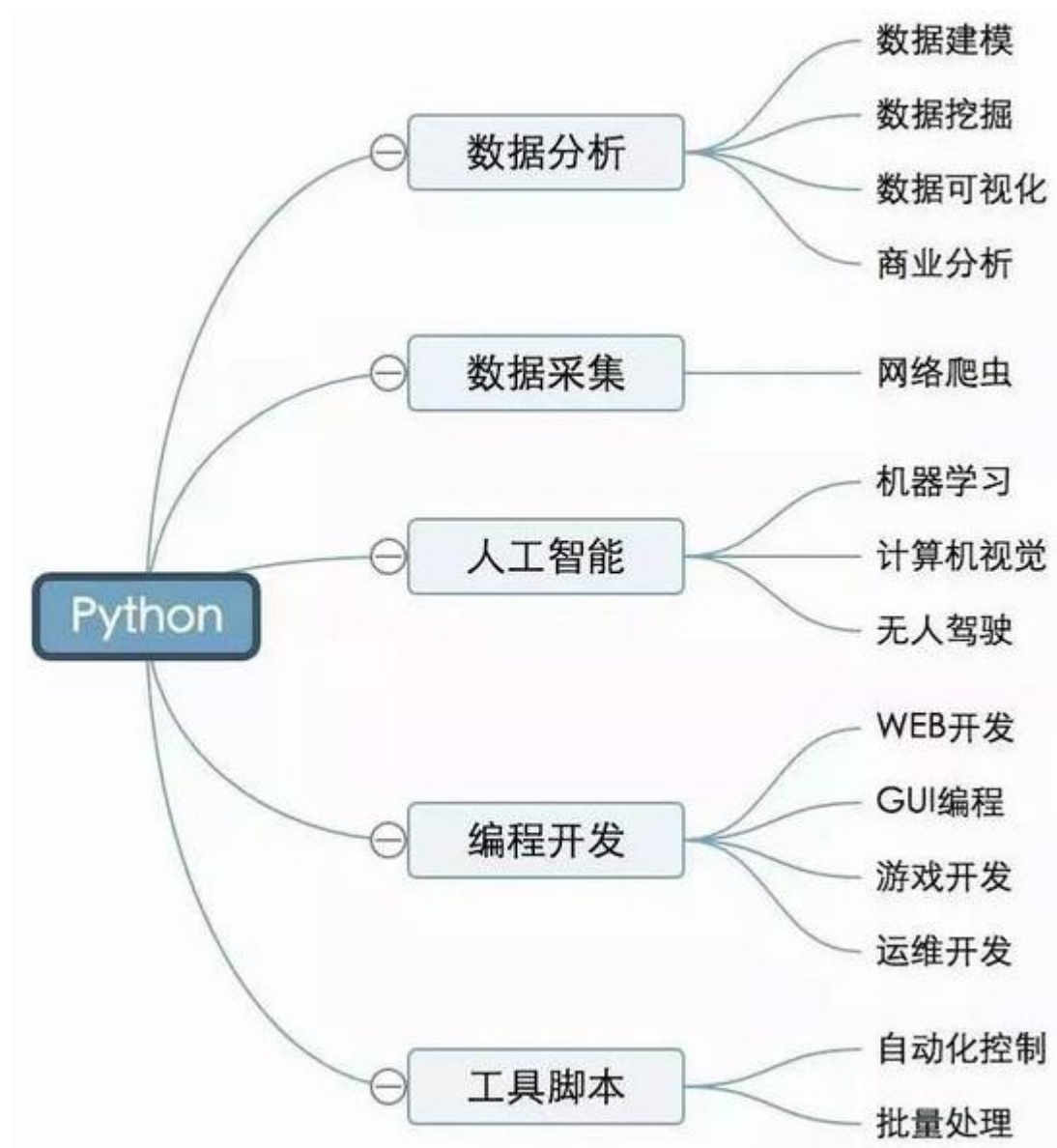
总而言之，开始学 Python 一定要注意这 4 点：

- . 代码规范，这本身就是一个非常好的习惯，如果开始不养好好的代码规划，以后会很痛苦。
- . 多动手，少看书，很多人学 Python 就一味的看书，这不是学数学物理，你看例题可能就会了，学习 Python 主要是学习编程思想。



- . 勤练习，学完新的知识点，一定要记得如何去应用，不然学完就会忘，学我们这行主要都是实际操作。
- . 学习要有效率，如果自己都觉得效率非常低，那就停不停，找一下原因，去问问过来人这是为什么。

## Python 语言主要应用领域表



# Python 编程主要可以应用于哪些领域?

在人工智能时代的今天，当然不懂 Python 语言，你就是“文盲”！现在你肯定在问 Why？

众所周知，中国人工智能行业正处于一个创新发展的时期，对人才的需求也在急剧增长。

如今 Python 语言的学习已经上升到了国家战略的层面上。

国家相关教育部门对于“人工智能的普及”格外重视，不仅将 Python 语言列入到小学、中学和高中等传统教育体系中，并借此为未来国家和社会发展奠定了人工智能的人才培养基础，逐步由底层向高层推动“全民学 Python”，从而进一步实现人工智能技术的推动和社会人才结构的更迭。

## Python 的重要程度

根据 TIOBE 最新排名，Python 已超越 C#，与 Java,C,C++ 一起成为全球前 4 大最流行语言。

如果你有留意互联网信息，随手打开一个招聘网站，你会发现知名的互联网企业都在招聘 Python 人才，如饿了么、小米、360、腾讯、阿里巴巴、美团和知乎等等，这是一个十分重要的信号。

一线互联网企业早就在储备人工智能的人才，为即将到来的人工智能时代做准备。

Python 岗位年薪至少在 10-20w 之间，而且除了北、上、广、深这样的一线城市以外，一些二、三线城市 Python 工程师的待遇正在与一线城市持平。未来，无论你身处何地，都能享受到人工智能、Python 带来的“市场红利”。

学会了 Python，那么首先要恭喜你，月薪 15k 是你最低的保障。同时你的人生之路又将拓宽，不仅仅是 Python 分析师，Python 开发工程师也会有你的一席之地，这样一个全面型人才，HR 又怎么会放过你！

在这样的大背景和大环境下，各大企业都在加大研发投入力度、招募高端人才，同时，为了抢占先机还通过收购等方式吸收人工智能优秀中小企业来提升整体竞争力。更重要的是，Python 凭借着积极开放的开源技术平台，正在构建着围绕自有体系的生态环境。

Python 到底能做什么？

Python 里通常可以用很少的时间实现。Python 作为一门编程语言，这门语言的魅力和影响力已经远超 C#、C++ 等编程语言前辈，被程序员誉为“美好的”编程语言。Python 基本上可以说是全能的，系统运维、图形处理、数学处理、文本处理、数据库编程、网络编程、web 编程、多媒体应用、pymo 引擎、黑客编程、爬虫编写、机器学习、人工智能等等，Python 应用无处不在。同时掌握一门

Python 语言有什么优点？

简单易学、高层语言、免费开源、可移植性强、丰富的库、面向对象、可扩展性、可嵌入型、规范的代码等.....

未来，无论你身处何地，都能享受到全国人工智能 Python“一盘棋”带来的“市场发展红利”。顺应潮流的风向标，Python 人工智能工程师发展前途广阔。

Python 是解释语言，程序写起来非常方便，写程序方便对做机器学习的人很重要。

Python 的开发生态成熟，有很多现有库可以用。相比而言，Lua 虽然也是解释语言，甚至有 LuaJIT 这种神器加持，但其本身很难做到 Python 这样。

Python 效率超高，解释语言的发展已经大大超过许多人的想象。毫无疑问使用 Python 语言的企业将会越来越多，Python 程序猿的人才缺口也将越来越大，认准时机，把握机遇。

基于这个市场需求背景，上海交通大学致立于打造史上专业、权威的数据分析师专业高端全站智能工程师课程，通过线上线下结合的方式，让您在业务数据分析、计算机编程、机器学习算法上有着一个全面的提升，从基础的数据分析理论方法到必备的数据分析算法，再到现在流行的数据可视化技术及基于 Python 的数据分析语言，直至时下热门的大数据分析技术。

掌握全栈数据分析能力，使你成为在互联网、零售、金融、等行业专门从事数据的采集、清洗、处理、分析并能制作业务报告、提供决策的新型复合型数据分析人才。

Python 项目国家在大力支持发展，企业也有很大的人才缺口，更是人工智能时代不可少少的语言，不懂可真的有可能被这个人工智能时代所抛弃!如果你不想成为未来的“文盲”或“失业者”，就不妨考虑提前投入到人工智能+Python 发展的大浪潮中吧!人工智能时代正是 Python 发展的好时机，希望大家都能抓住机遇，在 Python 领域闯出自己的一片天!

# 对 Python 开发者的编码建议

## 1、编码规范

每种编程语言、每个框架都有自己的一套编码规范和编码最佳实践方式，例如 PEP8 是 Python 语言的编码规范，作为 Python 开发者，每个人至少要将 PEP8 读上三五遍，熟记于心并运用在项目中。

## 2、文档注释

程序员最讨厌的两件事情：一，别人的项目没有文档和注释，二，自己的项目要写文档和注释。

经常看到一些项目连起码的 README 文件都没有，这跟 demo 没什么区别。如果你的项目没有文档，一旦有新人进来接手也是一脸懵逼，还显得非常业余。所以，一个正经的项目至少要说明项目的是做什么的，用了哪些东西，程序怎么跑起来等等。

简单的代码自然无需写注释，良好的代码自身就是文档和注释，但是如果有较复杂的代码逻辑就必须用注释来加以说明你当时写这段代码是如何思考的，否则隔两个月自己都不知道写的什么烂代码，还在骂这是哪个傻逼写的。所以，我们写注释应该在该写的时候才写，写多了或者写少了都不合适。还有一种情况就是过期的注释，需求变更代码更新后，发现注释还是旧的。

虽然 Python 是一门优雅简洁的语言，但是同样离不开注释的帮助，毕竟，代码是写给人看的。

## 3、避免重复代码

软件设计原则里面有个 DRY 原则，就是在一个项目里面不要写重复的代码，如果同样的逻辑出现超过两次，那么你就考虑将它进行封装成为一个函数或者公有方法。过多重复的代码到后面将导致很多潜在的问题，一处有问题，导致处处都有改动。

## 4、测试覆盖率

测试驱动开发（TDD）是目前主流的开发模式，但是我们往往因为项目进度或者懒惰，而将单元测试忽略了。没有单元测试的代码就像一座危楼，你永远都不知道里面有哪些坑，因为一旦改了一处代码，你无法预知对整个系统有什么影响，而单元测试是对代码质量的一种保障，测试覆盖率越高，潜在问题越少。

## 5、安全意识

安全问题没发生前都是小事，一旦发生将是不可估量的大事情，几年前 CSDN 这么大一个网站竟然还闹出将密码用明文存储的笑话，可见程序员的安全意思有多低，密码至少要做 HASH 存储并加盐处理。去年时候大疆的程序员将服务器的 key 上传到 github 也是典型的安全意思缺乏。什么东西该同步到代码库什么东西不该放要区分清楚。常见的 web 安全也要清楚，像 SQL 注入，CSRF、XSS 攻击的基本原理以及如何预防等等。

## 6、设计与架构

如果设计架构错了，代码写的再好也白搭，好比建房子设计错了，房子在漂亮也华而不实，可能隐藏重大安全隐患。合理的设计与架构在于根据业务做合理的取舍，遵循逐步演进原则，切忌无脑跟风参考 BAT 规模量级来设计你的系统，好的系统都是演变过来的。抛开业务谈技术都是耍流氓。

## 7、学习交流分享

分享你的所得，和比你厉害的人多交流。

# Python 字符串处理技巧大全

## 1.字符串的连接和合并

连接和合并

- 

相加 //两个字符串可以很方便的通过 '+' 连接起来

- 

```
str1='Hello'
str2='World'
new_str=str1+str2
print(new_str)
>>HelloWorld
```

- 

- 合并//用 join 方法

## 2.字符串的切片和相乘

- 

相乘//比如写代码的时候要分隔符，用 python 很容易实现

- 

```
line=' '*30
print(line)
>>
```

- 

切片

-

```

str='Monday is a busy day'
print(str[0:7])//表示取第一个到第七个的字符串
>>Monday

print(str[-3:])
>>day//表示取从倒数第三个字符开始到结尾的字符串

print(str[:])//复制字符串
>>Monday is a busy day

```

- 

### 3.字符串的分割

- 

普通的分割，用 `split`

- 

`split` 只能做非常简单的分割，而且不支持多个分隔

- 

```

phone='400-800-800-1234'
print(phone.split('-'))
>>['400', '800', '800', '1234']

```

- 

复杂的分割

- 


`r` 表示不转义,分隔符可以是;或者,或者空格后面跟 0 个或多个额外的空格，然后按照这个模式去分割

- 

```

line='hello world; python, I ,like, it'
import re
print(re.split(r'[;,s]\s*',line))
>>['hello world', 'python', 'I ', 'like', 'it!']

```

 菜鸟学python

### 4.字符串的开头和结尾的处理

比方我们要查一个文件的名字是以什么开头或者什么结尾

```

filename='trace.h'
print(filename.endswith('h'))
>>True
print(filename.startswith('trace'))

```



>>True

## 5.字符串的查找和匹配

- 一般查找
- 我们可以很方便的在长的字符串里面查找子字符串，会返回子字符串所在位置的索引，若找不到返回-1
- 复杂的匹配

```
mydate='11/27/2016'
import re
if re.match(r'\d+/\d+/\d+',mydate):
    print('ok,match')
else:
    print('ko,not match')
>>ok,match
```

- 6.字符串的替换

普通的替换//用 replace 就可以

```
text='Python is an easy to learn, powerful programming language.'
print(text.replace('learn','study'))
>>Python is an easy to study, powerful progr
```

复杂的替换//若要处理复杂的或者多个的替换，需要用到 re 模块的 sub 函数

## 7.字符串中去除一些字符

- 去除空格//对文本处理的时候比如从文件中读取一行，然后需要去除每一行的两侧的空格，table 或者是换行符
- 
- line=' Congratulations, you guessed it. '
- print(line.strip)

- 

>>Congratulations, you guessed it.

- 

注意:字符串内部的空格不能去掉,若要去掉需要用 re 模块

- 

复杂的文本清理,可以利用 str.translate,

- 

先构建一个转换表, table 是一个翻译表,表示把't'o'转成大写的'T' 'O',

- 

然后在 old\_str 里面去掉'12345',然后剩下的字符串再经过 table 翻译

- 

```
import string
instr = 'to'
outstr = 'TO'
table = string.maketrans(instr,outstr)
old_str='Hello world ,welcome to use Python. 123456'
new_str=old_str.translate(table,'12345')
print(new_str)
>>Hello wOrld ,welcOme TO use PyThOn. 6
```

-

# 零基础学习 Python 需要学习哪些知识？

## Python 基础与高级编程

- 1、Linux 环境搭建与 python 安装
- 2、Python 语法基础
- 3、Python 字符串解析
- 4、Python 时间和日历
- 5、Python 文件操作
- 6、Python 面向对象
- 7、设计模
- 8、异常处理
- 9、模块
- 10、Python 高级编程

## Linux 基础与高级编程

- 1、Linux 基本命令
- 2、Linux 系统编程
- 3、Linux 网络编程
- 4、正则表达式
- 5、shell 脚本编程
- 6、实战案例

其中正则表达式特别重要，希望大家学习的时候，多花点时间和精力在正则表达式上面，这是应用领域最广泛的一个模块。

## 数据结构

- 1、时间复杂度
- 2、数据列表
- 3、树
- 4、哈希
- 5、图
- 6、队列
- 7、堆栈

这些都是计算机的基础知识，不管是在 java、c 或是 PHP，这些模块都会涉及上面所说的知识。

## 数据库开发

- 1、MySQL 开发
- 2、MongoDB 开发
- 3、Redis 开发
- 4、数据库调优和部署

## 前端与移动开发

- 1、html
- 2、css
- 3、ps 基础应用
- 4、JavaScript
- 5、jquery
- 6、移动端框架和库
- 7、前端自动化、前端性能优化

有可能大家会觉得奇怪，怎么 python 还涉及到前端的知识，现在全栈程序员已经成为企业稀缺的人才，特别受重视，所以如果你也能成为一个全栈工程师，那么你将会是一个获得老板喜爱的 python 工程师。

## 爬虫开发

- 1、网络爬虫
- 2、爬虫原理与数据抓取
- 3、scrapy 框架
- 4、爬虫分布式集群
- 5、实战 AI 项目数据采集

爬虫是 python 的核心知识，现在 python 应用得最多的就是爬虫，比如说百度谷歌等的搜索引擎。

## 人工智能入门

- 1、Tensorflow
- 2、全连接网络
- 3、卷积神经网络
- 4、实战图片识别

作为人工智能完美搭档的 `python`，肯定还是要学习点人工智能知识。

# 学习 Python 零基础指导路线图

## 第一个疑问：学习 Python 难吗？

是不是越低级的程序越难学，越高级的程序越简单？

表面上来说，是的。

但是，在非常高的抽象计算中，高级的 Python 程序设计也是非常难学的，所以，高级程序语言不等于简单。

但是，对于初学者和完成普通任务，Python 语言是非常简单易用的。

## 第二个疑问：我 0 基础学 Python 可以吗？

首先我个人赞成把 Python 作为入门语言：

**1、语法简单明了。**第一门语言，其实就是语法+Flow control，而 Python 的语法简单，代码可读性高，容易入门。

**2、Python 的哲学是「做一件事情应该只有一种最好的方法」**，对于初学者规范自己的学习有很大的帮助，同时也帮助初学者能够读懂其他人的代码(相比 Perl 的代码简直没法看)

**3、养成良好的习惯。**Python 对于代码的要求严谨，特别是缩进(Indentation)，对于初学者养成良好的代码习惯很有帮助。

**4、Python 的语法设计非常优秀**，思想也比较现代，可以更快的理解现代编程语言的一些思想。

**5、Python 仍然是传统基于 Class 的 OO(对比 Javascript 基于 prototype 的 OO)**，和 Java，C#，Ruby 一样，比较大众。从 Python 去学 Design Pattern 也是比较合适的。

**6、Python 的内置数据结构清晰好用**，同时 Library 比较多，优秀的代码很多(相比 PHP 就有许多烂的代码，误导新人)。

**7、Python 免费的书很多(英文)，可以找到许多资料啃。同时(国外)社区比较集中，有问题可以向高手问。**

**8、Python 在其他领域，比如科学计算等等有广泛的运用，对于学一门语言作为工具来说，Python 很合适。**

### **第三个疑问：如何学 Python？**

如果你选择了自学，我想给你提几点建议：

1、找浅显易懂，例程比较好的教程，从头到尾看下去。不要看很多本，专注于一本。把里面的例程都手打一遍，搞懂为什么。

2、去找实际项目练手。最好是要有真实的项目做。可以找几个同学一起做个网站之类。注意，真实项目不一定非要是商业项目。

3、最好能找到一个已经会 python 的人。问他一点学习规划的建议，然后在遇到卡壳的地方找他指点。这样会事半功倍。

4、另外，除了学习编程语言，也兼顾补一点计算机基础，和英语。

5、不但要学写代码，还要学会看代码，更要会调试代码。读懂你自己程序的报错信息。再去找些 github 上的程序，读懂别人的代码。

6、学会查文档，用好搜索引擎和开发者社区。

**当然，如果你是 0 基础，周围也没有大神带领，自己也学不进去，我劝你还是放弃吧，或者就找个培训机构花点钱学习。**

开发是枯燥的，前期学起来比较痛苦，熬出头了就是一片晴天！

# 自学 Python 路线图

## Python 语言优点

Python,是一种面向对象的解释型计算机程序设计言语,具有丰厚和强壮的库,Python 已经成为继 JAVA, C++之后的的第三大言语。特色:简略易学、免费开源、高层言语、可移植性强、面向对象、可扩展性、可嵌入型、丰厚的库、标准的代码等。

Python--挨近无所不能的编程言语

Python 除了很少的工作不能做之外,其他基本上都涉及到,体系运维、图形处理、数学处理、文本处理、数据库编程、网络编程、web 编程、多媒体使用、pymo 引擎、黑客编程、爬虫编写、机器学习、人工智能等等。

Python 的使用特别广,我国现在的人才缺口超越 100 万,国内:豆瓣、搜狐、金山、通讯、隆重、网易、百度、阿里、土豆、新浪等,国外:谷歌、NASA、YouTube、Facebook、红帽等企业都在广泛使用,尤其是 Linux 运维、web 开发、大数据、人工智能等等。

关于 Python 的学习,先来一张学习线路图:







看到了大家的反馈，我会在每个阶段下面补充一下书籍资料。谢谢各位对我的赞同！

## 第一阶段：Python 核心编程





可解决的现实问题：  
能够熟练使用 Python 技术完成针对小问题的程序编写以及小游戏程序的开发。  
市场价值：

具备最基本的编程思维, 掌握基础的 Python 编程技术, 能够完成较小程序的开发,尚达不到企业的用人标准。

推荐书籍:

《Python 快速编程入门》



以 Window 平台、系统全面的讲解了 Python3 的基础知识,

第 1 章主要是带领大家认识 Python;

第 2 章主要针对 Python 的基础语法进行讲解;

第 3 章主要介绍的是 Python 中的常用语句;

第 4~5 章主要介绍了字符串、列表、元组、字典等类型;

第 6~7 章讲解了函数的基础和高级知识。

第 8 章讲解了 Python 中的文件操作;

第 9 章讲解了 Python 中异常的相关知识;

第 10 章讲解了 Python 中的模块;

第 11~12 章侧重讲解了面向对象编程思想;

第 13 章围绕着面向对象的编程思想, 开发了一个飞机大战的小游戏。

配套视频

Python 入门教程完整版 (懂中文就能学会)

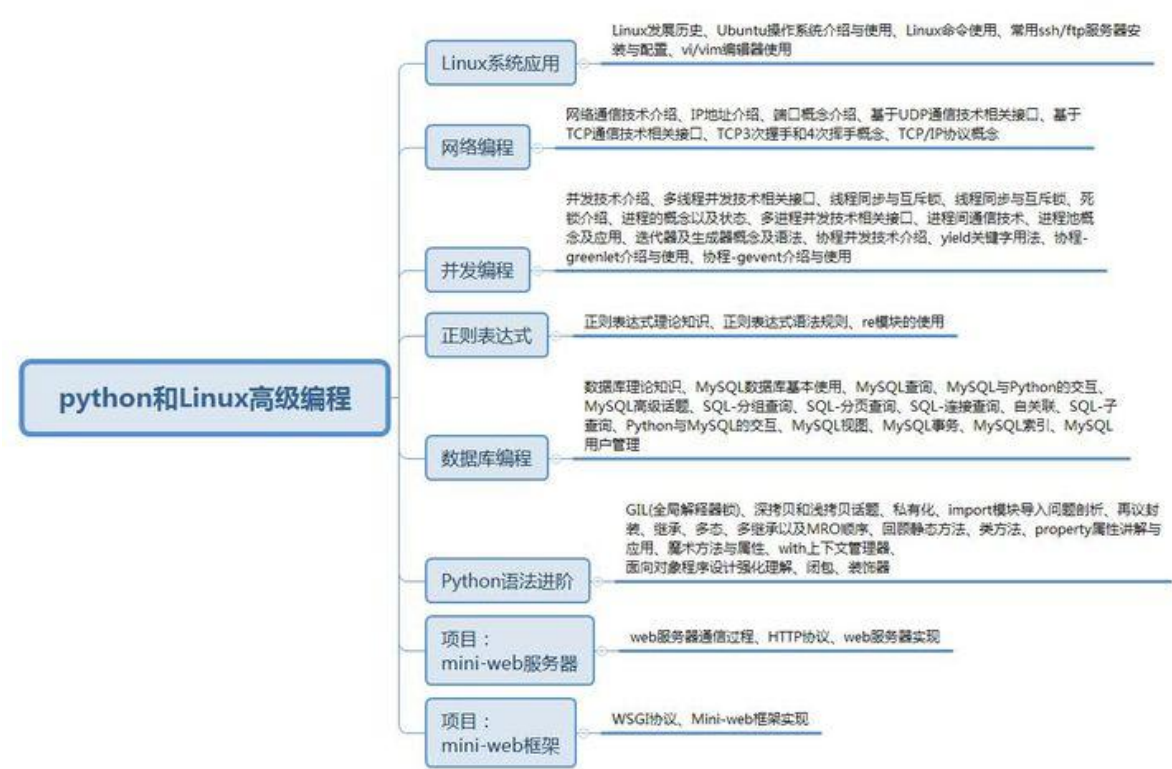
资料链

接: <https://pan.baidu.com/s/1gFpFL3ZmpD7ZpF4Syaupdw>

密码: ko9l

## 第二阶段: python 和 Linux 高级编程





可解决的现实问题：  
能够使用面向对象的程序设计方法，基于 Linux 操作系统进行高并发量的网络程序开发。

市场价值：  
熟练掌握 Python 技术和常见网络协议，可满足企业开发的初级需求，根据市场反馈数据看，薪资普遍在 6000-8000 元/月。

推荐书籍

《Linux 编程基础》  




class="content\_image" width="123">

本书分为 11 个章节，首先介绍了 Linux 的背景、开发环境、网络配置与远程操作及管理；其次讲解了 Linux 系统操作中的基本命令以及基础开发中使用的工具；

然后讲解了 shell 编程的基本语法、Linux 系统中用户、用户组以及 Linux 中的文件系统和操作；

之后对 Linux 程序开发中涉及的进程、信号、线程、网络编程等重难点知识进行了讲解；

最后讲解了 Linux 系统中高并发服务器的几种模型。

本书中的每个章节都以理论与案例结合的模式，在理论知识后通过切实可行的案例帮助学生在学的同时，实践、巩固所学知识。

配套视频

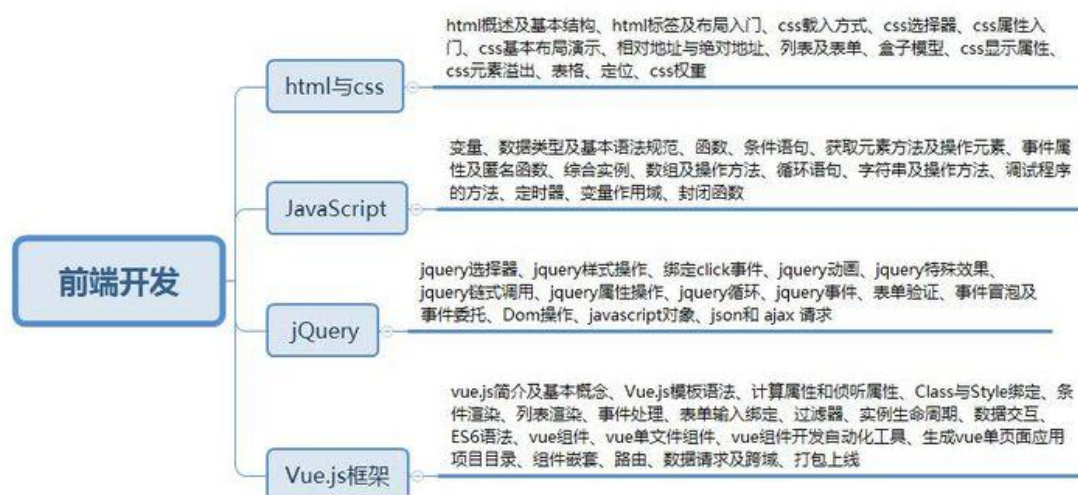
linux 从入门到精通

链接: <https://pan.baidu.com/s/1nvAQ-0QECVDueNjYAK245w>

密码: emem

## 第三阶段：前端开发





可解决的现实问题：

- 1、开发常见 Web 网站的前端页面和移动端 H5 页面；
- 2、跟后台进行数据通信；
- 3、掌握当前市场流行的前后端分离的开发模式中所用的前端框架。

市场价值：

前端作为 web 网站开发的半壁江山，全栈工程师必备技能，8000-12000 月。

推荐书籍

《HTML+CSS+JavaScript 网页制作案例教程》



全书共 10 章，结合 HTML、CSS 和 JavaScript 的基础知识及应用，提供了 29 个精选案例，以及 1 个综合实训项目。

第 1 到 3 章主要讲解 HTML 与 CSS 的基础知识，包括 Web 基本概念、HTML 与 CSS 简介、Dreamweaver 工具的使用、HTML 文本与图像标记、CSS 选择器、CSS 文本样式属性、CSS 的继承性和优先级。

第 4 到 7 章分别讲解了盒子模型、列表与超链接、表单、元素的浮动与定位。

第 8 到 9 章主要讲解 JavaScript 编程基础与事件处理。第 10 章为实训项目，带领读者开发一个包含结构、样式和行为的网页。

配套视频

vue.js 入门到实战开发

链接: <https://pan.baidu.com/s/1mhJLvJi> 密码: d8co

## 第四阶段：Web 开发







可解决的现实问题：

- 1、高并发全功能的 web 网站开发；
- 2、提供数据响应速度灵活运用缓存；
- 3、根据实际问题设计出相应数据库表。

市场价值：

web 全栈工程师，独立开发前端和后端业务，10000-20000。

配套视频

6 节课入门 Flask 框架 web 开发

链接：<https://pan.baidu.com/s/1htLsTSg> 密码：fbwr

## 第五阶段：爬虫开发





可解决的现实问题：

- 1、能够实现定向抓取互联网上的海量信息；
- 2、能够运用爬虫框架包括实现分布式爬虫；
- 3、能够根据具体需求，实现定制的爬虫框架。

市场价值：

具备互联网数据采集的爬虫开发能力，市场薪资普遍在 11000-25000

推荐书籍

《用 Python 写网络爬虫》



class="content\_image"



width="350">

《用 Python 写网络爬虫》作为使用 Python 来爬取网络数据的杰出指南，讲解了从静态页面爬取数据的方法以及使用缓存来管理服务器负载的方法。

此外，本书还介绍了如何使用 AJAXURL 和 Firebug 扩展来爬取数据，以及有关爬取技术的更多真相，比如使用浏览器渲染、管理 cookie、通过提交表单从受验证码保护的复杂网站中抽取数据等。

本书使用 Scrapy 创建了一个高级网络爬虫，并对一些真实的网站进行了爬取。

配套视频

6 节课掌握 Python 爬虫视频

链接: <https://pan.baidu.com/s/1eTT4AYm> 密码: 6d75

## 第六阶段：人工智能



data-original="https://pic3.zhimg.com/v2-a03d78b0e200b9f7095adc16db12c80a\_r.jpg">



可解决的现实问题：

- 1、从数据支持到策略开发；
- 2、实现自动交易策略；
- 3、深度学习模型的训练过程；
- 4、图像识别、检测任务。

市场价值：

具备可对数据进行初步分析和挖掘，进行机器学习建模或深度学习训练，根据市场反馈数据看，薪资普遍在 15000-30000。

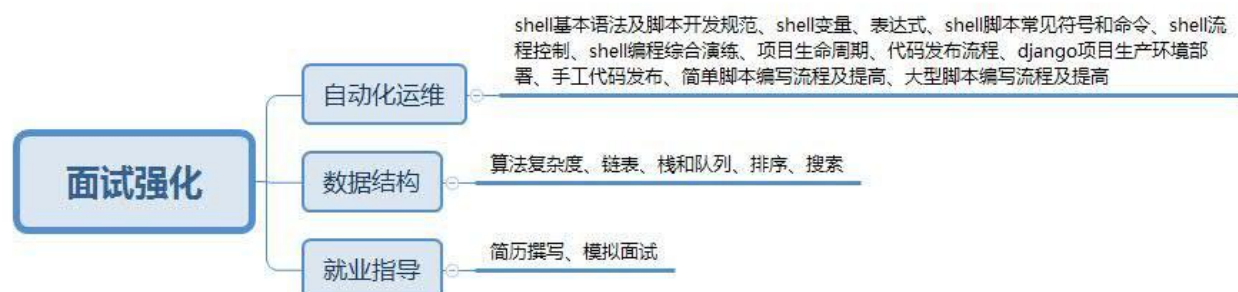
配套视频

6 节课机器学习入门

链接：<https://pan.baidu.com/s/1htUM116> 密码：3nrz

## 第七阶段：面试强化





可解决的现实问题：

- 1、项目环境自动化部署；



2、项目代码自动化发布；

3、项目生命周期理解。

复习和巩固所学知识，进行专题面试强化。

# 学 Python 有三道坎

为什么要学习 Python 做数据分析？在我看来，大概有 3 大理由。

**广度：**各行各业都有自己的商业场景，每一个行业都需要使用数据来辅助决策。面对现在人人谈大数据的情境，数据分析是一个你不得不学（却不知上哪学的）的技能。

**精度：**Python 是一门编程语言。也许从前的你完全依靠 excel 的默认设置生成图表，从不思考为什么做一张数据图，而使用编程工具的你必须从图表长宽开始思考每一步成形的理由，从而更精地理解数据。

**高效：**传统的数据工作涵盖大量的重复不动脑操作，比如把日表合成周表，比如批量删除某个字段，比如批量删除空值。这些工作通过鼠标点击软件没有办法编成工作流，但却可以通过 Python 程序编写自动化，省去大量时间。

于是你不甘落后...决定加入学习 Python 的浪潮中，和“别人”一争高下。

一个月后.....

Python 入门并不困难，那么“你”和“别人”的区别在哪？就四个字：**学习方法**。

如果你向各种前辈请教学习 python 的方法，那么听到的一定是这三种回答：

## 【1】

**学习是反人类的，自己不动手永远学不会**

买了一本 Python 入门书，翻看了 20 页；收集了 20G Python 练习资料，保存在硬盘里；看了一些视频跟着写了会，突然就开始看剧了，导致放弃。这些情况是初学者很容易碰到的，学习是反人类的，Python 学习更是一项前期很枯燥的行为。

学 Python 的第一道坎：一旦决定学习，耐得住寂寞，坚持动手，才可能学会！

## 【2】

**学不进去的时候**

**找到一个小目标才能获得学习成就感**

Python 语言基础很枯燥，所以需要一些目标刺激。这些目标不能像“我要学会 Python 去做数据挖掘工作”一样抽象，而得像“我想试试能否自动打开文件夹里所有的表格，并整理成结构化的数据”一样具体，像是做数学应用题一样，只有实战项目能让学习者快速记住 Python 的用法和各种坑，比如一个简单的分析案例（来自数据分析师体验课）：

学 Python 的第二道坎：不能为学习而学习，要随时找理由去使用它！

## 【3】

**能坚持自学是好事，**

**找到老师帮你答疑解惑，更会事半功倍**

对于成年学习者来说,编程领域的知识可能是完全零散的,但面临的情况却可能是就业、是学业或者工作急需。那么像学拼音那样从基础学习 Python 并不适合。老师的作用就是免去小伙伴们大量没头没脑的搜寻时间、快速了解建立必须知识体系,有阶段的完成实战项目,启发初学者对应用场景的理解,从而降低入门到放弃的几率。

**学 Python 的第三道坎: 你需要一个愿意答疑的“好老师”。**

所以,如果你想学 Python 数据编程,我强烈建议你从一次“要动手、有目标、有答疑”的 7 天《Python 数据分析师训练营》开始学习。

# 零基础学 python 路线图

人生苦短，我用 Python，为啥这么说，因为我们有个金句："学完 python，便可上天"。

废话不多说，相信很多人都听过之前的 Python 进入小学课本、Python 进入浙江省高考等新闻，有这么多头衔加持的 Python 究竟魅力在哪？

为啥说 2018 年，python 是大家最想学的语言？又为什么大家都对 python 如此关注？下面咱们来剖析一下。

同样，有很多人都会抱有这样的疑问：

转行学习 Python，完全 0 基础能否学会呢？

Python 的难度到底有多大？

...

就如我刚开始学 Python 的时候也会出现这些疑问，下面咱们来分析一下。

**第一个疑问：学习 python 难吗？**

是不是越低级的程序越难学，越高级的程序越简单？

表面上来说，是的。

但是，在非常高的抽象计算中，高级的 Python 程序设计也是非常难学的，所以，高级程序语言不等于简单。

但是，对于初学者和完成普通任务，Python 语言是非常简单易用的。

**第二个疑问：我零基础学 python 可以吗？**

首先我个人赞成把 Python 作为入门语言：

1、语法简单明了。第一门语言，其实就是语法+Flow control，而 Python 的语法简单，代码可读性高，容易入门。

2、Python 的哲学是「做一件事情应该只有一种最好的方法」，对于初学者规范自己的学习有很大的帮助，同时也帮助初学者能够读懂其他人的代码。

3、养成良好的习惯。Python 对于代码的要求严谨，特别是缩进(Indentation)，对于初学者养成良好的代码习惯很有帮助。

4、Python 的语法设计非常优秀，思想也比较现代，可以更快的理解现代编程语言的一些思想。

5、Python 仍然是传统基于 Class 的 OO(对比 Javascript 基于 prototype 的 OO)，和 Java, Ruby 一样，比较大众。从 Python 去学 Design Pattern 也是比较合适的。

6、Python 的内置数据结构清晰好用，同时 Library 比较多，优秀的代码很多(相比 PHP 就有许多烂的代码，误导新人)。

7、Python 免费的书很多，可以找到许多资料啃。同时(国外)社区比较集中，有问题可以向高手问。

8、Python 在其他领域，比如科学计算等等有广泛的运用，对于学一门语言作为工具来说，Python 很合适。

### 第三个疑问：如何学习 python?

如果你选择了自学，我想给你提几点建议：

1、找浅显易懂，例程比较好的教程，从头到尾看下去。不要看很多本，专注于一本。把里面的例程都手打一遍，搞懂为什么。

2、去找实际项目练手。最好是要有真实的项目做。可以找几个同学一起做个网站之类。注意，真实项目不一定非要是商业项目。

3、最好能找到一个已经会 python 的人。问他一点学习规划的建议，然后在遇到卡壳的地方找他指点。这样会事半功倍。

4、另外，除了学习编程语言，也兼顾补一点计算机基础，和英语。

5、不但要学写代码，还要学会看代码，更要会调试代码。读懂你自己程序的报错信息。

6、学会查文档，用好搜索引擎和开发者社区。

当然，如果你是零基础，周围也没有大神带领，自己也学不进去，我劝你还是放弃吧，或者就找个培训机构花点钱学习，北邮在线就是你不错的选择。

开发是枯燥的，前期学起来比较痛苦，熬出头了就是一片晴天！

# 为什么要学习 Python， 能 Python 来做些什么？

说起编程语言，Python 也许不是使用最广的，但一定是现在被谈论最多的。随着近年大数据、人工智能的兴起，Python 越来越多的出现在人们的视野中。

那么人们在谈论 Python 的时候究竟在谈论什么？Python 的实际应用场景有哪些？这里给大家简单做一个介绍：

## Web 应用开发

在因大数据、人工智能为人所熟知之前，Python 就已经在 Web 开发领域被广泛使用，产生了 Django、Flask、Tornado 等 Web 开发框架。得益于其简洁的语法和动态语言特性，Python 的开发效率很高，因而深受创业团队的青睐。

一些将 Python 作为主要开发语言的知名互联网企业/产品：

豆瓣

知乎

果壳网

Instagram

Quora

Dropbox

Reddit

由于后台服务器的通用性，除了狭义的网站之外，很多 App 和游戏的服务器端也同样用 Python 实现。

## 自动化运维

在 Web 开发领域，Python 只是众多语言选择之一；但在自动化运维领域，Python 则是必备技能。灵活的功能和丰富的类库使其成为运维工程师的首选语言。大量自动化运维工具和平台或以 Python 开发，或提供 Python 的配置接口。单从 Linux 内置 Python 这一点来看也足见其在服务器和运维领域的地位。

因此很多公司虽然核心业务不是使用 Python，但在管理系统、运维等方面也大量使用。比如 Facebook 工程师维护了上千个 Python 项目，包括基础设施管理、广告 API 等。

## 网络爬虫

也叫网络蜘蛛，是指从互联网采集数据的程序脚本。对于很多数据相关公司来说，爬虫和反爬虫技术都是其赖以生存的重要保障。尽管很多语言都可以编写爬虫，但灵活的 Python 无疑也是当前的首选。基于 Python 的爬虫框架 Scrapy 也很受欢迎。

这个地球上最大的“爬虫”公司 -- Google 一直力推 Python，不仅在公司内部大量使用 Python，也为开发社区做了巨大贡献。就连 Python 之父 Guido van Rossum 也曾曾在 Google 工作七年。

## 数据分析

当通过爬虫获取了海量数据之后，需要对数据进行清洗、去重、存储、展示、分析，在这方面 Python 有许多优秀的类库：NumPy、Pandas、Matplotlib 可以让你的数据分析工作事半功倍。

## 科学计算

虽然 Matlab 在科学计算领域有着不可取代的地位，但 Python 作为一门通用的编程语言，可以带来更广泛的应用和更丰富的类库。NumPy、SciPy、BioPython、SunPy 等类库在生物信息、地理信息、数学、物理、化学、建筑等领域发挥着重要作用。

而大名鼎鼎的 NASA 也早已把 Python 作为主要开发语言。

## 人工智能

Python 在人工智能大范畴领域内的数据挖掘、机器学习、神经网络、深度学习等方面都是主流的编程语言，得到广泛的支持和应用。

机器学习：Scikit-learn

自然语言处理：NLTK

深度学习：Keras、Google 的 TensorFlow、Facebook 的 PyTorch、Amazon 的 MxNet

这些已经占据业内主流的工具要么是用 Python 开发，要么也提供了 Python 版本。Python 无疑已成为 AI 领域的必修语言。

## 胶水语言

Python 简洁、灵活、通用，几乎可以在各种场景与各种平台、设备、语言进行连接，因此被称为胶水语言。有人把它比作小巧而又多功能的瑞士军刀。除了上面提到的，在其他领域也常常见到 Python 的身影：

金融：大量金融分析和量化交易工具使用 Python 作为的开发脚本语言

游戏：一些引擎使用 Python 作为开发脚本，比较有名的游戏有《文明》系列、网易的《阴阳师》

桌面应用：虽然不算主流，但 PyQt、wxPython、Tkinter 等 GUI 库也足以应付一般的桌面程序

在各家公司里，Python 还常被用来做快速原型开发，以便更快验证产品概念。而众多极客也把 Python 作为实现自己天马行空想法的神兵利器。在知乎上就有一个问题：

有着如此广泛的应用，再加上简单易懂的语法，使得 Python 成为一门既适合初学，又值得深入的语言。即使不是程序员，能用 Python 写上一小段程序，调用几个接口，也能极大提升工作效率。所以连 C++ 大牛 Bruce Eckel 也要感叹：

Life is short, you need Python

人生苦短，你需要 Python

由此来看，Python 越来越热也是情理之中的事情。



# Python 与各种开发语言比较、对比优略

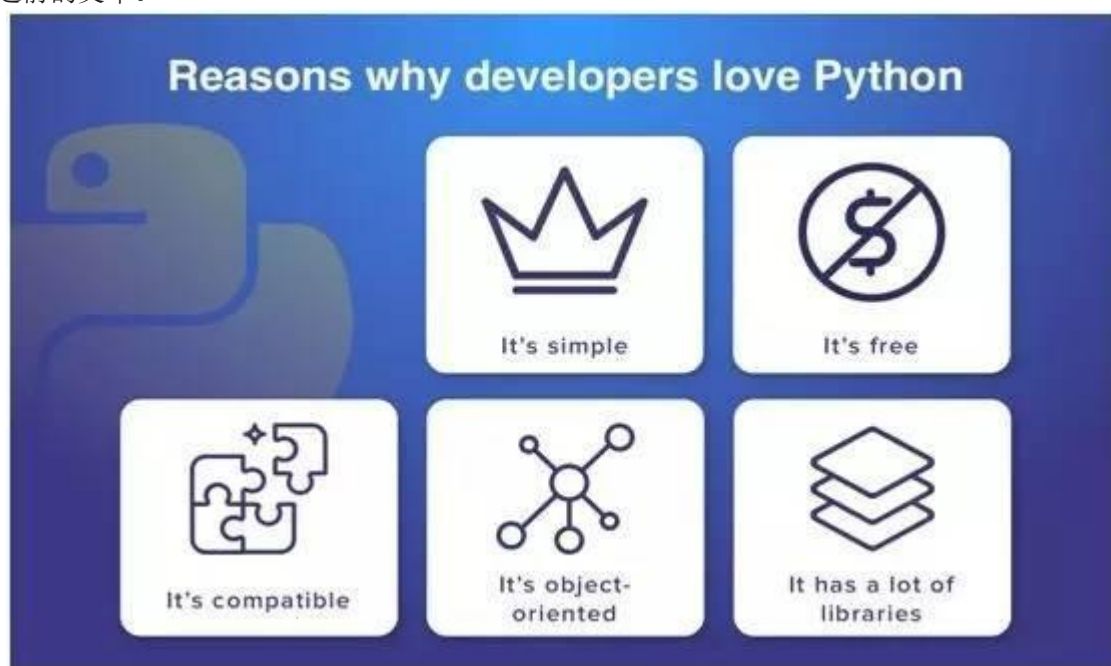
选择要学习的技术和选择要上的大学一样重要，如果选错了，你将来不仅得不到自己喜欢的高薪工作，反而会弄得一堆麻烦。如果你打开了这篇文章，说明你已经考虑选择 Python 开发作为你以后的职业了。在这篇文章里，我们会详细找出 Python 和其他语言相比的优势。我们会指出 Python 与 Java,Ruby,PHP 和 C# 的差异，帮你了解你所需要的技术。但在开始之前，我们先看下 Python 能做什么，谁使用它，为什么使用它

为什么用 Python，它能用在哪儿，能做什么呢？

Python 在约 40 年前出现以来，已经有数以千计基于这项技术的网站和软件项目，Python 因其独有的特点从众多开发语言中脱颖而出，深受世界各地的开发者喜爱。

Python 有什么优点呢？

下面，我们列举了 Python 最显著的一面。所有这些优点使它非常流行，也正因于此，众多跨国公司选择了 Python 作为他们的编程语言，关于这些公司的更多信息，您可以阅读我们之前的文章。



简单

我们可以说 Python 是简约的语言，非常易于读写，遇到问题时，程序员可以把更多的注意力放在问题本身上，而不用花费太多精力在程序语言、语法上。

免费

Python 是免费开源的。这意味着程序员不用花钱，就可以共享、复制和交换它，这也帮助 Python 形成了强壮的社区，使用它更加完善，技术发展更快。专业人士可以在社区和初学者分享他们的知识和经验。

找到你可以使用的开源库会得到什么好处？削减一半的项目支出！

兼容性

Python 兼容众多平台，所以开发者不会遇到使用其他语言时常会遇到的困扰

面向对象

Python 既支持面向过程，也支持面向对象编程。在面向过程编程中，程序员复用代码，在面向对象编程中，使用基于数据和函数的对象 尽管面向对象的程序语言通常十分复杂，PYTHON 却设法保持简洁。

库

Python 社区创造了一大堆各种各样的 Python 库。在他们的帮助下,你可以管理文档,执行单元测试、数据库、web 浏览器、电子邮件、密码学、图形用户界面和更多的东西。所有东西包括在标准库,然而,除了它,还有很多其他的库。

Python 语言的用途



多年来，Python 在各种流行编程语言中一直排名靠前。它几乎可以适用任何开发，它旨在提高程序员的开发效率而不在于他们编的代码。Python 适用于网站、桌面应用开发，自动化脚本，复杂计算系统，科学计算，生命支持管理系统，物联网，游戏，机器人，自然语言处理等很多方面。而且，既使对于那些从没有开发经验的人来讲，Python 的代码也是简洁易懂的。，由于 Python 程序代码简单，所以和与其他程序语言相比，后期的程序维护更容易，更舒心。从商业角度来看，需要的成本降低，程序员的效率提高

谁使用 Python

Python 开发人员社区不断壮大，支持库持续增多，使 Python 成为世界上功能最丰富的编程语言之一，可以适用于任何项目开发。但我们仍要指出，Python 在科学领域非常流行，特别是在数据挖掘和机器学习等方面。为了全面理解 Python 相对于其他语言的优势，我们将把 Python 和最流行的 WEB 技术做下比较，先从 PHP 开始。

Python 和 PHP

从开发的角度来看，PHP 是面向 WEB 的语言。PHP 应用程序更像是一组单独的脚本，甚至只是一个单独入口。而 Python 是多用途语言，也可以用于 WEB 开发，基于 Python 的 WEB 应用是加载到内存的完整应用，有自己的内容声明，保存所有的查询和请求。选择 Python 还是 PHP 进行 web 应用开发需要注意以下几点：

Python vs PHP		
		
Popularity	Very popular	Very popular
Frameworks	A lot of frameworks	A few frameworks
Learning	Easy to learn	Harder to learn

## Python 和 PHP web 开发的比较

### 通用性

当今时代，趋势和流行非常重要，一些客户和产品负责人只想使用最流行最热门的技术来开发他们的项目，造成了技术精湛的开发人员却完全没有客户和工作的情况出现。所以在开始学习任何东西前，要确保你要学的技术在未来 1、5 或 10 年内仍保持流行，不会被遗忘。

不过，你大可不必担心，因为 PHP 和 Python 都属于世界上最流行的编程语言。PHP 用于构建 Wikipedia, Yahoo, WordPress, Friendster, MailChimp, Flickr 等许多“巨头”，但不要以为 Python 没什么用，它也用来构建了 YouTube, Instagram, 桌面版的 Dropbox, Reddit, Bitbucket, Quora, Spotify, Pinterest, Facebook 的内部服务，以及 PayPal 系统的一部分

### 框架

当你选择一项技术的时候，工具的多样性也是极其重要的，它会使你的工作简单、方便。如果一个技术为不同的任务提供了多种工具，程序员就不必每件事都要从头开始了。PHP 最流行的框架是 Laravel, Symfony, CodeIgniter, Yii 1 and 2, Phalcon 和其他一些，这些工具能帮你创建功能强大而整洁的应用程序，而 Python 却没有这么多引以为豪的框架，最好用的是 Django 和 Flask，但是，我们可以向你保证，随着 Python 社区的成长，这种情况会很快改变。

### 学习

这通常是学生第一个想要知道的问题，学习教育的过程越容易，意味着开始工作和赚钱越快。

Python 当然是这类的赢家，它的语法容易，简单易学，而 PHP 却不是这样。掌握 PHP 需要花费很多的时间和努力。Python 允许你犯些小错但不会破坏代码，给新手一些信心继续学习。从新手的角度来看，想学一些更容易、更灵活的技术，而 Python 正是这样的技术。

你可以用 **Python** 创建安全的应用程序，但使用 **PHP** 却需要额外的工具。不过不要忘记，**PHP** 是专门用于 **WEB** 开发的，也的确用在这方面较多。

**Java** 和 **Python** 哪个更好？

一些开发人员声称 **Python** 比 **Java** 更有效率。但这应该先弄清 **Python** 和 **Java** 之间的区别是什么？

Python vs Java		
	 python	 Java
Learning	Easy to learn	Harder to learn
Cross-platform apps	✗	✓
Compatibility with operating systems	✓	✓
Network based apps	✗	✓

**Java** 和 **Python** 的区别

**Java** 是一种严格的类型语言，这意味着必须显式声明变量名。相比之下,动态类型的 **Python** 则不需要声明变量。在编程语言上有许多关于动态和静态类型的争论，但有一点应该注意：**Python** 是一种语法简单的功能强大的语言，能够通过编写脚本就提供优秀的解决方案，并能够快捷地部署在各个领域。

**Java** 可以创建跨平台的应用程序，而 **Python** 几乎兼容当前所有操作系统。对新手来讲，**Python** 比 **Java** 更容易上手，而且代码易读性强，但是如果你想你的代码可以在任何地方都能执行的话，那么还是选择 **Java** 吧。不过 **Java** 的可移植性也是有代价的，使用 **Java** 你需要购买更大的机器，消耗更多的内存，并且程序更加难以开发。

**Java** 比 **Python** 更复杂，没有技术背景的人学起来并非易事。

**Python** 与 **c#**

现在再来和看下 **c#**。它们的技术差异很大，但都适用于 **web** 开发。

Python vs C#		
	python	C#
Simplicity	✓	✗
Script writing	In any environment	Only in IDE
Libraries	A lot of libraries	Few libraries
Performance	Low	High

### Python 对 c#的比较

简而言之，**Python** 原本就被设计的类似用英语表达一样，只要你使用合适的变量名称，许多表达式就很容易读懂。另外，由于 **Python** 语法简单，没有像句法括号和大量的修饰词，各种类 **C** 的构造和不同的初始化变量，所以 **Python** 写的代码易读易学。

同时，**C#**从 **C++**和 **Java** 遗传了很多原始表达的类 **C** 语法，更重要的是，**C#**语法使我们必须遵循一定的规则来编写自己的方法或是继承类，伴随而来的是另一大串修饰词，还有一点不能忘记的是要把代码段放在括号内。而 **Python** 只用 **SHIFTS** 键就可以让代码看上去很整洁。

至于编写脚本方面，值得一提的是 **Python** 的脚本是真正的脚本，能够被解释器执行。你可以用任何编辑器打开它，修改后就可以立即运行。这在手边没有 **IDE** 或编译器的时候优势十分明显。而且使用 **Python** 更容易编写跨平台的脚本，甚至都不需要重新编译。

但是，我们要指出 **Python** 的一个不足，就是需要在机器上安装脚本解释器，至少要在一个包或可执行文件里打包上一个解释器，从而使脚本的大小从几 **KB** 增加到十几 **M**，不过对于现代计算机来说，十几兆的空间需求基本可以忽略不计，所以这也不是什么大问题。

而 **C#**需要 **IDE** 来编写程序，**C#**的一个好处是，当你编写基于 **WINDOWS** 平台的脚本时，它有强大的各类 **WINDOWS** 系统组件支持。例如，注册表、**WMI**、网络等内置工具。**C#**可以使用 **WINFORMS**，你需要的时候可以很容易地创建图形化接口。

没法说 **Python** 和 **C#**哪个更好，**Python** 比 **C#**更容易学，开源库更多，但 **C#**的标准库比 **Python** 的更好，其性能更高。

### Ruby 和 Python 的区别

就第一语言而言，**Ruby** 和 **Python** 是最受欢迎的。**Ruby** 是非常流行的构建网站技术，其中最著名的是 **Twitter**(早期版本),**Basecamp**,**Github**,**Airbnb**,**Slideshare**,**Groupon**。

Python vs Ruby		
	python	
Approach to a problem	One solution	A lot of solutions
Community	Large	Large
Syntax	Very simple	More complex

## Ruby 和 Python

Ruby 和 Python 都是面向对象的语言,都是动态和灵活的。这些技术的主要区别在于他们解决问题的方式。Ruby 提供了不同的方法而 Python 通常只有一个。这个事实既是优势也劣势。

最流行的 Ruby 框架是 Ruby on rails。它和 Django 非常类似,因为这两个框架都是为了解决相同的任务。如果我们比较这些技术的社区,我们会发现他们几乎是一样的,然而,形成这些团体的人是不同的。Python 在数据科学和数学方面很受欢迎,所以在这里你可以找到很多的学者和教授。

记住,当你开发 web 应用程序时,可以用 RoR 实现,也可以用 Django,两者都是快速高效的。如果开发偏重于大量计算和数据处理的应用,应该选择 Python。

任何技术成功的秘密在于围绕它所构建的社区,不同人群协同工作来确定其未来的发展演变,Python 的支持者是世界上规模最大和构成最多样化的一个团队,不仅有数以千计的个人开发者,而且还有诸如谷歌,Yandex,Dropbox,Mozilla,微软(在 Visual Studio 中使用),英特尔等许多巨头公司,他们和其他许多公司一起,已经用 Python 创建了世界上最大的和最受欢迎的项目。

# 全面认识和了解 Python 编程语言

Python 编程语言拥有诸多用于网络应用开发、图形用户界面、数据分析、数据可视化等工作的框架和特性。Python 可能不是网络应用开发的理想选择，但是正被很多机构广泛用于评估大型数据集（dataset）、数据可视化、进行数据分析或制作原型。在数据科学领域，Python 编程语言正不断获得用户的亲睐，而作为网络开发语言，Python 显得有点过时了。本篇博文，就是要对这两种截然不同的 Python 使用方式，进行详细的对比，并且帮助大家明白一点：如果要利用 Python 做数据科学工作，并没有必要了解它用于网络开发的部分。

python:web development vs data science

面向数据科学的 Python

从顶级金融机构到最小的大数据创业公司，各行各业、各种规模的机构都在使用 Python 编程语言支撑业务运作。Python 作为数据科学编程语言，不仅受顶级大数据公司欢迎，还有众多技术创业企业拥泵。它还位列 2015 推荐学习的前 10 种编程语言。

世上只有两种编程语言：一种是总是被人骂的，一种是从来没人用的。--Bjarne Stroustrup

Python 属于前一种，而且日益被用于数学计算、机器学习和多种数据科学应用。除了性能依赖性强和底层的业务外，它能够做其他任何事情。利用 Python 编程语言的最好选择，就是做数据分析和统计计算。学习面向网络开发的 Python，需要程序员掌握像 Django 这样的多种网络框架协助建设网站；但是学习面向数据科学的 Python，则要求数据科学家学习如何使用正则表达式和科学计算库，并掌握数据可视化的概念。由于目的、方向不同，那些不了解 Python 网络开发的程序员，能很轻松地走上利用 Python 编程语言做数据科学工作的道路。

Python 是一个有着 23 年历史的强大动态编程语言，语言表现力很强。程序员编码完成后，不需要编译器即可运行程序。面向网络开发的 Python 支持多种编程范式，包括结构化编程（structured programming）、函数式编程（functional programming）和面向对象编程（object-oriented programming, OOP）。Python 代码可以很容易地嵌入到许多拥有编程接口的网络应用中。但是，Python 更是

开发学术研究和科学计算程序的绝佳选择，这些程序要求运行快速、数学计算精确。

而面向网络编程的 **Python**，则要求程序员学习多种网络开发框架，这个学习难度比较大，因为现有 **Python** 网络开发框架的文档不太容易理解。当然，不可否认的是，要想利用 **Python** 开发一个动态网站或网络应用，学习网络框架是必需的。

## Python 网络开发框架

目前，**Python** 社区已经有多种免费的网络应用开发框架，比如：

### Django

**Django** 是帮助完美主义者按时完成工作的 **Python** 网络开发框架（译者注：原文是 **Django is the python web development framework for perfectionists with deadlines**。这也是 **Django** 官网上对该框架的描述）。使用 **Django** 进行网络开发，最适合的场景是开发那些依靠数据库驱动，同时也具备类似自动化后台管理界面和模板系统等炫酷功能的应用。对于不需要太多功能的网络开发项目来说，**Django** 可能是大材小用，主要是它的文件系统容易让人搞混，而且文件目录结构要求严格。使用 **Django** 进行 **Python** 网络开发的公司有纽约时报、**Instagram** 和 **Pinterest**（译者注：**Pinterest** 联合创始人 **Paul Sciarra** 在 **Quora** 上的回答提到了使用 **Django**，**Quora** 地址）。

### Flask

**Flask** 是针对初学者的框架，它简单，轻量，初学者很快就可以上手开发单页网络应用。这个框架并不支持验证，没有数据抽象层和其他许多框架所包括的组件。它不是一个全栈开发框架，也只用于小型网站的开发。（译者注：其实 **Pinterest** 也使用了 **Flask**，只是没用在整站开发上，而是用来开发 **API**，具体见链接。）

### CherryPy

**CherryPy** 框架强调要符合 **Python** 语言规范，做到程序员像进行面向对象编程一样开发网络应用。它还是诸如 **TurboGears** 和 **Web2py** 等流行全栈框架的基础模板引擎。



还有很多其他框架，包括 **Pyramid**、**Bottle** 和 **Pylons** 等，但是无论 **Python** 开发者使用哪一种框架，他/她都要花精力仔细地研究教程和文档。

为什么使用 **Python** 进行网络开发不现实？

**Python** 作为网络开发语言，很可能是一个不太现实的选择：

面向网络开发的 **Python** 需要非标准化、昂贵的主机服务，尤其是程序员使用流行的 **Python** 网络框架开发网站时。由于利用 **PHP** 进行网络编程如此的便捷，大部分的用户没有兴趣在 **Python** 上投入太多的精力。

面向网络开发的 **Python** 与诸如 **PHP**、**Java** 或 **Ruby on Rails** 等语言不同，不是一个经常需要的技能。但是面向数据科学的 **Python** 却越来越受欢迎，而且由于它更多地被用于机器学习和其他数据科学程序，**Python** 更是招聘数据科学家的公司所最看重的技能。

面向网络开发的 **Python** 已经经历了较长的发展，但是它的学习曲线并没有像 **PHP** 这样的网络编程语言那么高。

为什么将 **Python** 用于数据科学是最好的选择？

**Python** 编程是驱动大数据、金融、统计和数字运算的核心科技，而它的语法却像英语一样易懂。近来，由于拥有多个针对机器学习、自然语言处理、数据可视化、数据探索、数据分析和数据挖掘的插件，丰富的 **Python** 数据科学生态体系得到了较大的发展，甚至有将数据科学社区 **Python** 化的趋势。今天，面向数据科学的 **Python** 已经具备了清洗、转换和处理大数据的所有工具。对于数据分析师岗位来说，掌握 **Python** 也是最受欢迎的技能。一名具备 **Python** 编程能力的数据科学家，可以在纽约挣到平均年薪 14 万美元的工资。

为什么数据科学家喜欢使用 **Python** 语言？

数据科学家喜欢那些能够快速输出原型，帮助他们轻松地记录下自己的想法和模型的编程环境。他们喜欢通过分析巨量的数据集，得出结论，完成工作。而 **Python** 编程语言则是开发数据科学应用的多面手，因为它能帮助数据科学家，以最短最优的时间进行编码、调试、运行并获取结果，从而高效地完成工作。

一名技术娴熟的企业数据科学家的真正价值，在于利用多种数据可视化手段，向公司的不同利益相关者有效地传递数据模式和预测。否则，数据科学工作就是一场零和游戏。**Python** 以其优良特性，符合高强度科学计算的几乎所有方面要求，这使得它成为在不同的数据科学应用之间进行编程的绝佳选择，原因很简单：开

发人员仅用一种语言就可以完成开发和分析工作。面向数据科学的 **Python** 将企业业务的不同部分连接在一起，提供了一个数据分享和处理的直接媒介。

**Python** 遵循统一的设计哲学，注重可用性、可读性，对于数据科学的学习曲线也较低。

**Python** 有很高的可扩展性，且与 **Matlab**、**Stata** 等语言相比，运行更加快速。另外，**Python** 生态系统中还在涌现出更多的数据视觉化库，以及炫酷的应用编程结构，目的是使用图形更好地展现数据分析的结果。**Python** 社区有着诸如 **Sci-Kit learn**、**NumPy**、**Pandas**、**Statsmodel** 和 **SciPy** 等许多优秀的数据分析库。这些库的数量还在不断增长。

数据分析与 **Python** 编程语言十分契合。如果你决定要通过 **Python** 语言学习数据科学，那么你应该考虑的下一个问题，就是 **Python** 库中有哪些是可以完成大部分的数据分析工作？接下来，我们给大家介绍全球的企业数据科学家都在使用的 **Python** 数据分析库。

### **NumPy**

**Numpy** 是使用 **Python** 开发的高级（**high level**）工具的基础。这个库不能用于高级数据分析，但是深入理解 **Numpy** 中面向数组的计算，可以帮助数据科学家有效使用 **Pandas** 库。

### **SciPy**

**SciPy** 主要用于科学计算，拥有许多不同的模块，可用于特殊函数、图像处理、插值法（**interpolation**）、线性代数、常微分方程（**ODE**）求解器以及其他多种用途。这个库还可以与 **NumPy** 数组一起使用，实现许多高效的数学运算。

### **Pandas**

**Pandas** 是用于数据再加工最好的库，因为它使得处理遗失的数据、自动数据对齐（**data alignment**）变得更加简单，它还支持处理从不同的数据源收集而来的索引数据。

**SciKit** 这个流行的机器学习库拥有多种回归、分类和聚类算法，还支持 **gradient boosting**、向量机、朴素贝叶斯模型和逻辑回归。这个库还被设计成能够与 **NumPy** 和 **SciPy** 进行交互。

### **Matplotlib**

这是一个二维绘图库，有着交互性很强的特性，生成的图表可以放大、推移，并且能够用于发行刊物印刷出版。而且，还支持多平台的交互环境。

**Matplotlib**、**NumPy** 和 **SciPy** 是科学计算的基础。还有许多其他的 **Python** 库，诸如用于网络挖掘的 **Pattern**，用于自然语言处理的 **NLTK**，用于深度学习的 **Theano**，用于爬取网络的 **Scrappy**，**IPython**，**Statsmodels**，**Mlpy** 等。对于初学 **Python** 数据科学的人，他们需要很好地掌握上面提到的优秀数据分析库。

# Python 实现字符串处理功能的函数大全

## 一、拆分含有多种分隔符的字符串

### 1.如何拆分含有多种分隔符的字符串

问题： 我们要把某个字符串依据分隔符号拆分不同的字段，该字符串包含多种不同的分隔符，例如：

```
s="ab;cd|efg|hi,jkl|mn\topq;rst,uvw\txyz"
```

其中;,|,\t 都是分隔符号，如何处理？

方法一： 连续使用 `str.split()`方法，每次处理一种分隔符号

```
s="ab;cd|efg|hi,jkl|mn\topq;rst,uvw\txyz"
```

```
defmySplit(s,ds):
```

```
    res=[s]
```

```
    fordinds:
```

```
        t=[]
```

```
        map(lambdax: t.extend(x.split(d)), res)
```

```
    res=t
```

```
    returnres
```

```
printmySplit(s,'|,\t')
```

输出：

```
['ab','cd','efg','hi','jkl','mn','opq','rst','uvw','xyz']
```

方法二： 使用正则表达式的 `re.split()` 方法，一次性拆分字符串

```
import re
```

```
s="ab;cd|efg|hi,jkl|mn\topq;rst,uvw\txyz"
```

```
print re.split(r'[;|,\t]+',s)
```

输出：

```
['ab','cd','efg','hi','jkl','mn','opq','rst','uvw','xyz']
```

## 二、调整字符串中文本格式

### 1. 如何判断字符串 `a` 是否以字符串 `b` 开头或结尾

问题：某文件系统目录下有一系列文件：`a.py, quicksort.c, stack.cpp, b.sh`，编写程序给其中所有 `.sh` 文件和 `.py` 文件加上用户可执行权限？

解决方案：使用字符串中的 `str.startswith()` 和 `end.startswith()` 方法（注意：多个匹配时参数使用元组）

```
In [1]: import os
```

```
# 列出当前目录以.sh 和以.py 结尾的文件
```

```
In [2]: [name for name in os.listdir('.') if name.endswith((''.py', '.sh'))]
```

```
Out[2]: ['b.sh', 'a.py']
```

```
In [3]:importstat
```

```
# 查看 a.py 文件权限
```

```
In [4]: os.stat('a.py').st_mode
```

```
Out[4]:33204
```

```
# 把文件权限转换成 8 进制，即为平常看到的权限
```

```
In [5]:oct(os.stat('a.py').st_mode)
```

```
Out[5]:?'
```

```
# 更改文件权限，添加一个可执行权限
```

```
In [6]: os.chmod('a.py',os.stat('a.py').st_mode | stat.S_IXUSR)
```

```
In [7]: ll
```

```
total0
```

```
-rwxrw-r--1yangyang05 月 914:48a.py*
```

```
-rw-rw-r--1yangyang05 月 914:48b.sh
```

```
-rw-rw-r--1yangyang05 月 914:48quicksort.c
```

-rw-rw-r--1yangyang05 月 9 14:48stack.cpp

## 2.如何对字符串中文本的格式进行调整

问题： 某软件的 log 文件，其中日期格式为“yyyy-mm-dd”：

2017-05-0809:12:48status half-configured passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48status installed passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48status half-configured passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48status installed passwd:amd641:4.2-3.1ubuntu5.2

2017-05-0809:12:48startup packages configure

09:12:48startup packages configure

我们想把其中日期改为美国日期的格式"mm/dd/yyyy",2017-05-08 ==>

05/08/2017 ,应如何处理？

解决方案： 使用正则表达式 `re.sub()` 方法做字符串替换，利用正则表达式的捕获组捕获每个部分内容，在字符串中调整各个组的捕获顺序。

```
In [1]:importre
```

```
In [2]: log=open('/var/log/dpkg.log').read()
```

# `(\d{4})` 匹配到 4 个数字为一个捕获组，其顺序为 1。故后面替换用 `\1` 放到最后，`r` 是为了防止字符串被转义

```
In [3]:printre.sub('(\d{4})-(\d{2})-(\d{2})',r'\2/\3/\1', log)
```

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status half-configured passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status installed passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48startup packages configure

# 也可以为每个捕获组起个名称，而不使用默认顺序来处理

```
In [5]: printre.sub('(P\d{4})-(P\d{2})-(P\d{2})',r'\g/\g/\g', log)
```

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status unpacked passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status half-configured passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48status installed passwd:amd641:4.2-3.1ubuntu5.2

05/08/201709:12:48startup packages configure

### 三、字符串拼接

#### 1.如何将多个小字符串拼接成一个大的字符串

问题：在程序中我们将各个参数按次序收集到列表中： ["<0112>","<32>","<1024x768>","<60>"],要把各个参数拼接成数据报进行发送  
"<0112><32><1024x768><60>"

解决方案：

方法一：迭代列表，连续使用“+”操作依次拼接每一个字符串

```
In [1]: pl=["<0112>","<32>","<1024x768>","<60>"]
```



```
In [2]: s=""
```

# 这种方法会产生许多临时结果，会造成资源的浪费

```
In [3]: for pin in pl:
```

```
...:     s=s+p
```

```
...:     print s
```

```
...:
```

```
<0112>
```

```
<0112><32>
```

```
<0112><32><1024x768>
```

```
<0112><32><1024x768><60>
```

```
In [4]: s
```

```
Out[4]: '<0112><32><1024x768><60>'
```

方法二：使用 `str.join()` 方法，更加快速的拼接列表中所有字符串

```
In [5]: ".join(pl)
```

```
Out[5]: '<0112><32><1024x768><60>'
```

有个列表 `l = ['abc', 123, 45, 'xyz']`，如何让 123 和 45 以字符串的方式拼接

```
In [6]: l=['abc',123,45,'xyz']
```

# 使用生成器表达式，开销比列表表达式小

```
In [7]: (str(x) for x in l)
```

```
...:
```

```
Out[7]:<generatorobjectat0x7fe3cadedf550>
```

```
In [8]: ".join(str(x)forxinl)
```

```
Out[8]:'abc12345xyz'
```

#### 四、字符串居中对齐

##### 1.如何对字符串进行左、右、居中对齐

问题： 某个字典存储了一系列属性值

```
{
```

```
"loDist":100.0,
```

```
"smartCull":0.04,
```

```
"farclip":477
```

```
}
```

在程序中想以工整的格式进行输出，如何处理？

解决方案：

方法一： 使用字符串的 `str.ljust()`,`str.rjust()`,`str.center()`进行,右,居中对齐

方法二： 使用 `format` 方法，传递类似'`<20`','`>20`','`^20`'参数完成同样任务

```
In [1]: s='abc'
```

```
In [2]: s.ljust(20)
```

```
Out[2]:'abc          '
```

```
In [3]: s.ljust(20,'=')
```

```
Out[3]: 'abc====='
```

```
In [4]: s.center(20)
```

```
Out[4]: '  abc  '
```

```
In [5]: format(s,'<20')
```

```
Out[5]: 'abc      '
```

```
In [6]: d={
```

```
....: "loDist":100.0,
```

```
....: "smartCull":0.04,
```

```
....: "farclip":477
```

```
....: }
```

```
In [7]: d.keys()
```

```
Out[7]: ['loDist','smartCull','farclip']
```

```
In [8]: w=max(map(len,d.keys()))
```

In [9]:forkind:

...: printk.ljust(w),':',d[k]

...:

loDist :100.0

smartCull :0.04

farclip :477

## 2.去掉不需要的字符串

问题:

1.过滤掉用户输入中前后多余的空白字符: ' nick@gmail.com '

2.过滤某 windows 下编辑文本中的'\r': 'hello world\r\n'

3.去掉文本中的 unicode 组合符号(音调):u'zǒu'

解决方案:

方法一: 字符串 strip(),lstrip(),rstrip()方法去掉字符串两端字符

方法二: 删除单个固定位置的字符, 可以使用切片+拼接的方式

方法三: 字符串的 replace 方法或正则表达式 re.sub()方法删除任意位置字符

方法四: 字符串 translate()方法,可以同时删除多种不同字符

In [1]: s=' abc 123 '

In [2]: s.strip()

Out[2]:'abc 123'

In [3]: s.lstrip()

Out[3]:'abc 123 '

```
In [4]: s='-----ab+++++'
```

```
In [5]: s.strip('-+')
```

```
Out[5]: 'ab'
```

```
In [6]: s='abc:123'
```

```
In [7]: s[:3]+s[4:]
```

```
Out[7]: 'abc123'
```

```
In [8]: s='\tabc\t123\txyz'
```

```
# 去除\t
```

```
In [9]: s.replace('\t','')
```

```
Out[9]: 'abc123xyz'
```

```
In [10]: s='\tabc\t123\txyz\ropq\r'
```

```
In [11]:importre
```

```
# 去除\t\r
```

```
In [12]: re.sub('[\t\r]','',s)
```

```
Out[12]:'abc123xyzopq'
```

```
In [13]: s='abc\refg\n\2342\t'
```

```
# 去除\t\r\n
```

```
In [14]: s.translate(None,'\t\r\n')
```

```
Out[14]:'abcefg\x9c2'
```

```
In [15]: u=u'zǒu'
```

```
In [16]: u
```

```
Out[16]: u'z\u01d2u'
```

```
In [17]:printu.translate({0x01d2:None})
```

**Zu**

# Python 转义字符大全表

Python 转义符用反斜杠(\)转义字符。如下表：

转义字符	描述
\(在行尾时)	续行符
\\	反斜杠符号
\'	单引号
\"	双引号
\a	响铃
\b	退格 (Backspace)
\e	转义
\000	空
\n	换行
\v	纵向制表符
\t	横向制表符
\r	回车
\f	换页
\oyy	八进制数 yy 代表的字符，例如：\o12 代表换行
\xyy	十进制数 yy 代表的字符，例如：\x0a 代表换行
\other	其它的字符以普通格式输出

# python3 以后用 urllib.request 代替

## urllib2

今天试了一下一段简单的代码：

```
import urllib2
```

```
response = urllib2.urlopen("http://www.anxin66.com")  
print response.read()
```

运行后报错：

Traceback (most recent call last):

File "D:/PycharmProjects/network\_test/scrapy.py", line 1, in

import urllib2

ImportError: No module named 'urllib2'

自己电脑里装的是 python 3.7 里面，在 3 以后用 urllib.request 代替 urllib2，所以改成这样：

```
import urllib.request
```

```
response = urllib.request.urlopen("http://www.anxin66.com")  
print(response.read())
```

于是就可以看到输出结果啦！

---