



第3章 Python语言入门与Web信息解析

3.1 Python语言入门

3.2 Python语言进阶

3.3 HTTP解析与Python实现

3.4 HTML解析与Python实现

3.5 Web信息解析

1. 《Python 爬虫开发与项目实战》 范传辉编著 机械工业出版社 2017年11月第一版
2. 《Python带我起飞 入门、进阶、商业实战》 李金洪编著 中国工信出版集团、电子工业出版社 2018年6月第一版 2019年9月第三次印刷



3.1 Python语言入门

3.1.1 Python概述

3.1.2 实例运行

3.1.3 Python语法规则



3.1.1 Python概述

- Python 是一门强大的解释型、面向对象的高级程序设计语言
- 特点：
 - 简单、可移植、易扩展
 - 可用于桌面应用、系统编程、数据库编程、网络编程、Web 开发、图像处理、人工智能、数学应用、文本处理等
 - 资源众多：GitHub 上提供了数以万计的Python 开源项目，供爱好者仔细研究
- Python 已经成为人工智能的第一主流语言
 - 各大主流人工智能框架——TensorFlow、CNTK、Caffe2 等，都支持用户使用Python 语言进行开发。



3.1.1 Python概述

■ Python大事记:

- 1980 年 —— 吉多·范罗苏姆开始构思 Python 的想法
- 1989 年, Guido van Rossum 开发了Python 语言
- 1999 年, 第一个基于Python的 Web 框架(ZOPE) 诞生
- 2000 年, Python 2.0 版本发布, 构成了现在Python 语言框架的基础。
- 2004 年, Python 2.4 版本诞生, Djang(Web 框架)诞生
- 2008 年, Python 3.0 版本诞生, 实现了一次大的跳跃
- 2010 年, 诞生了目前应用最广泛的Python2 . 7 版本。
它是一个承接2 .X 和3.X 特性的过渡版本。



3.1.2 语法规则

■ 语言的分类

1. 从运行角度的分类，Python 属于解释型语言

2. 从形态角度的分类，Python 属于动态语言。

- (1) 动态语言：程序运行时可以改变其结构，可以对变量或函数进行修改。Perl、Ruby 等。
- (2) 静态语言：常用于编译型语言，在使用变量之前必须要定义好数据类型。如：C、C++、C#、Java 等。

3. 语义角度的编程语言分类，Python 属于强定义类型

- (1) 强定义类型语言：严格区分内部的变量类型，类似的语言有C、Java 等。
- (2) 弱定义类型语言：不严格区分内部的变量类型，类似的语言有汇编语言、VB Script、JavaScript 等。

3.1.1 Python概述

■ 常见的集成开发环境

■ 编辑器:

- PyCharm (范例)

- Eclipse+PyDev

- 最简单: 记事本

■ 运行环境

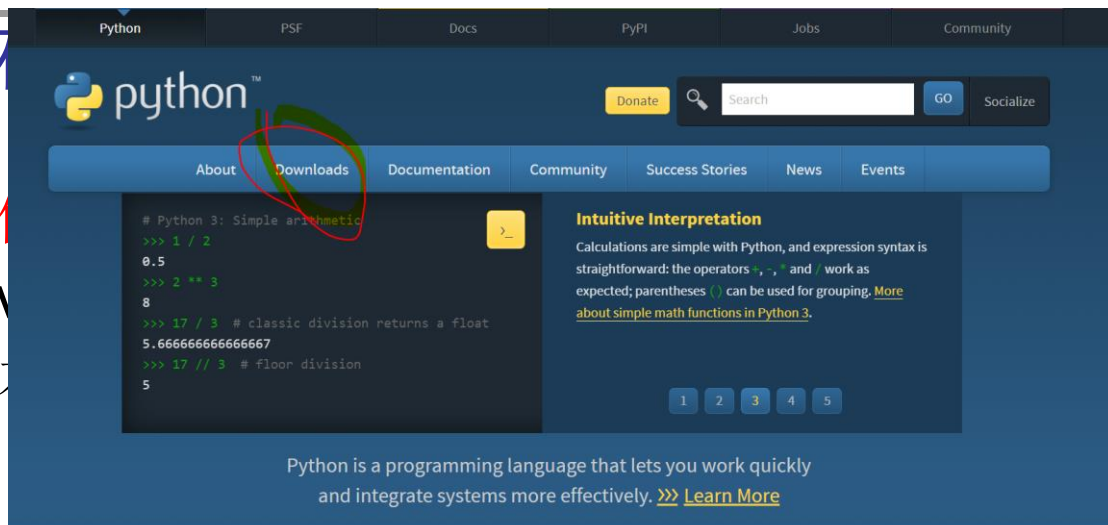
- Windows/Unix上安装Python

- Python的官方网站www.python.org下载最新的3.9或3.8版本 (见助教安装文档)

■ 安装:

- 助教文档

- 网上: 菜鸟教程<https://www.runoob.com/python/python-install.html>



■ Python 的运行方式

1. 命令行方式——最基础方式。
 - 可以在命令行里一句一句地输入

命令提示符 - python

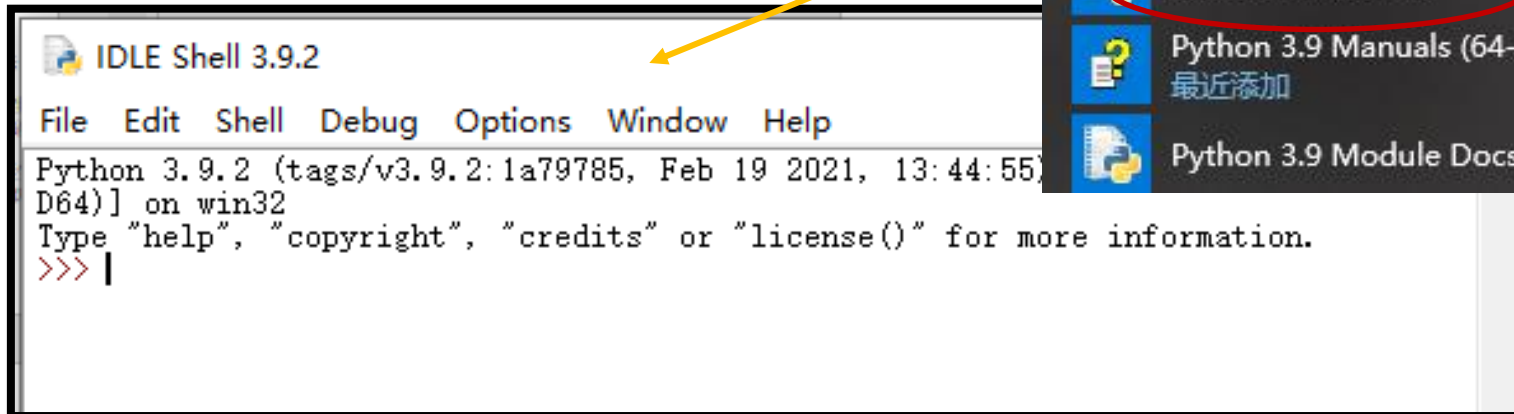
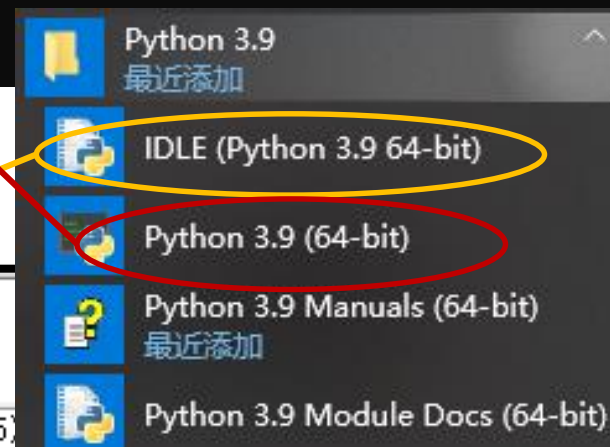
Microsoft Windows [版本 10.0.19041.
(c) 2020 Microsoft Corporation. 保留

```
C:\Users\fcx>python
Python 3.8.2 (tags/v3.8.2:7b3ab59,
Type "help", "copyright", "credits"
>>> _
```

Python 3.9 (64-bit)

```
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

2. 图形交互式





3.1 Python语言入门

3.1.1 Python概述

3.1.2 实例运行

3.1.3 Python语法规则

3.1.2 实例运行

实例1：一个简单的抓取程序

功能：获得电子工业出版社的首页内容

程序环境：Python (3.x) + requests模块

- 安装requests模块（操作系统环境下）

```
Microsoft Windows [版本 10.0.18362.720]  
(c) 2019 Microsoft Corporation。保留所有权利。
```

```
C:\Users\fcx>pip install requests
```

```
C:\Users\fcx>pip install requests  
Collecting requests  
  Downloading https://files.pythonhosted.org/packages/29/c1/24814557f1d22c56d  
/requests-2.25.1-py2.py3-none-any.whl (61kB)  
    100% |#####| 6  
Collecting chardet<5,>=3.0.2 (from requests)  
  Downloading https://files.pythonhosted.org/packages/19/c7/fa589626997dd07bd  
/chardet-4.0.0-py2.py3-none-any.whl (178kB)  
    100% |#####| 1  
Collecting idna<3,>=2.5 (from requests)  
  Downloading https://files.pythonhosted.org/packages/a2/38/928ddce2273eaa564  
/idna-2.10-py2.py3-none-any.whl (58kB)  
    100% |#####| 6  
Collecting urllib3<1.27,>=1.21.1 (from requests)
```

first_get.py

import requests #引入requests模块

def get_content(url): #定义函数

 resp=requests.get(url)

 return resp.text

url="https://www.phei.com.cn/" #定义URL值，目标网站地址

content=get_content(url) #调用函数值返回值

print("前500个字符为：", content[0:500])

content_len=len(content)

print("内容的长度为：", content_len)

if content_len>=40*1024: #判断内容长度是否大于40K

 print("内容长度大于等于40KB.")

else:

 print("内容长度小于等于40KB.")

请输入书名、作者、ISBN

所有分类 ▾

返回(B)	Alt+向左箭头
前进(F)	Alt+向右箭头
重新加载(R)	Ctrl+R
另存为(A)...	Ctrl+S
打印(P)...	Ctrl+P
投射(C)...	
翻成中文(简体)(T)	
查看网页源代码(V)	Ctrl+U

```
2 <html ng-app="topDivApp" ng-controller="topDivCtrl">
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
6 <title>电子工业出版社有限公司</title>
7 <meta name="keywords" content="" />
8 <meta name="description" content="" />
9 <link rel="shortcut icon" href="#" />
10 <link href="/templates/stylesheets/global.css" rel="stylesheet" />
11 <link href="/templates/stylesheets/web_base1.css" rel="stylesheet" />
12 <link href="/templates/stylesheets/web_base.css" rel="stylesheet" />
13 <script src="/templates/javascript/jquery-3.2.1.min.js" type="text/javascript"></script>
14 <script src="/templates/javascript/ajax.js" type="text/javascript"></script>
15 <script src="/templates/javascript/page_script.js" type="text/javascript"></script>
16 <script src="/templates/javascript/angular/angular.min.js" type="text/javascript"></script>
17 <script language="javascript" type="text/javascript">
18 function createXMLHttpRequest() {
```

File Edit Format Run Options Window Help

Run Module	F5
Run... Customized	Shift+F5
Check Module	Alt+X
Python Shell	

import request

def get_content

resp=request

IDLE Shell 3.9.2

File Edit Shell Debug Options Window Help

url=

cont

prin

cont

prin

if c

else

Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: C:\Users\fcx\Desktop\first_get.py =====

前500个字符为: i>><!DOCTYPE HTML>

<html ng-app="topDivApp" ng-controller="topDivCtrl">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>çI|å-|å·%ä,|å|°ç||çQ%æ||é||å|~å|, </title>

<meta name="keywords" content="" />

<meta name="description" content="" />

<link rel="shortcut icon" href="#" />

<link href="/templates/stylesheets/global.css" rel="stylesheet" />

<link href="/templates/stylesheets/web_base1.css" rel=

内容的长度为: 54811

内容长度大于等于40KB.

>>>

命令提示符

Microsoft Windows [版本 10.0.19041.804]

(c) 2020 Microsoft Corporation. 保留所有权利。

C:\Users\fcx>python first_get.py

python: can't open file 'first_get.py': [Errno 2] No such file or directory

C:\Users\fcx>_

■ 注意程序目录！

C:\Users\fcx>python C:\Users\fcx\Desktop\first_get.py

前500个字符为：<!DOCTYPE HTML>

<html ng-app="topDivApp" ng-controller="topDivCtrl">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<title>çä·%ä, äç(æääååå</title>

<meta name="keywords" content="" />

<meta name="description" content="" />

<link rel="shortcut icon" href="#" />

<link href="/templates/stylesheets/global.css" rel="stylesheet" />

<link href="/templates/stylesheets/web_base1.css" rel=

内容的长度为： 54811

内容长度大于等于40KB.

C:\Users\fcx>

■ 解释程序特点：一行行执行

[illegible]

没有>>>提示符
是Print执行结果

没有>>>提示符
是Print执行结果

没有>>>提示符
是Print执行结果

引入模块模块和包：可以统称模块

```
import requests      #引入requests模块
```

注释：
#：单行注释
三个单引号或三个双引号：多行

```
def get_content(url):    #定义函数
```

```
    resp=requests.get(url)  
    return resp.text
```

定义函数

```
url="https://www.phei.com.cn/"    #定义URL值，目标网站地址  
content=get_content(url)           #调用函数值返回值  
print("前500个字符为：", content[0:500])  
content_len=len(content)  
print("内容的长度为：", content_len)  
if content_len>=40*1024:            #判断内容长度是否大于40K  
    print("内容长度是否大于等于40KB.")  
else:  
    print("内容长度是否小于等于40KB.")
```

“代码块”，即同一个代码块中的语句具有相同的缩进格式

流程控制



3.1 Python语言入门

3.1.1 Python概述

3.1.2 实例运行

3.1.3 Python语法规则



3.1.3 Python语法规则

■ 语句的基本规则：变量、语句、代码块

1. 变量的命名

- 可以使用字符数字和下画线的组合，首字母可以是字母或者下画线。

2. 语句区分

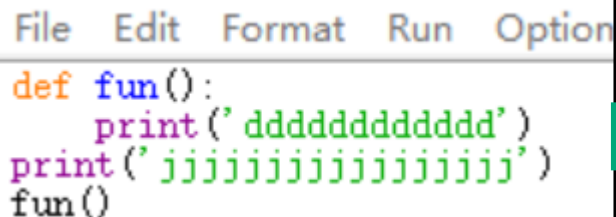
- 在Python 解释器中，源代码是一行一行被解释执行的，行与行之间通过换行符号来区分。
- 默认是一行一条语句，如果在一行中放有多条语句可以通过“；”来区分。例如：

```
A=6          #一行就是一条语句
```

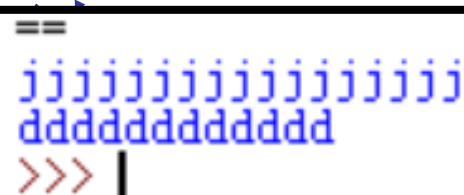
```
B=8;cc=10    #这种属于两条语句，虽然在一行，但是使用分号分开了
```

3.1.3

3. 代码块



```
File Edit Format Run Option
def fun():
    print('dddddddddd')
print('jjjjjjjjjjjjjjjjjj')
fun()
```



```
==
jjjjjjjjjjjjjjjjjj
dddddddddd
>>> |
```

- 通过缩进来表述“代码块”，即同一个代码块中的语句具有相同的缩进格式。缩进可以是使用空格

```
def fun():          #定义一个函数名字叫作 fun
    print("hello fun") #该语句被缩进，表示 fun 函数的函数体，只有执行函数 fun 时才被执行
print("hello main")  #该语句没被缩进，直接执行
fun()                #该语句没被缩进，会执行其函数体内的语句
```

- 程序运行时，会先执行第三行代码，输出“hello main”
- 接着执行第四行代码，调用函数fun，输出“hello fun”



3.1.3 Python语法规则

4.注释

- “#”用来代表注释。它的生效范围是“行”，在一行代码中#之后的内容将不被Python解释器处理。

```
#print("hello main")           #该语句前面有#号，表示为注释语句，不会被执行
```

- 也可以使用三个单引号('')或者三个双引号('"')将代码段变为字符串，程序同样不会执行。

```
'''                               #开始
def fun():
    print("hello fun")
print("hello main")
fun()
'''                               #结束
```



3.1.3 Python语法规则

5.使用Python 的“帮助”

- 使用**help** 命令来获取帮助信息。它可以查找关于Python的基础函数、类型、常用库等信息

```
help (print)           #显示 print 函数的帮助信息
```

```
Help on built-in function print in module builtins:
```

```
print(...)
```

```
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

```
Prints the values to a stream, or to sys.stdout by default.
```

```
Optional keyword arguments:
```

```
file: a file-like object (stream); defaults to the current sys.stdout.
```

```
sep:  string inserted between values, default a space.
```

```
end:  string appended after the last value, default a newline.
```

```
flush: whether to forcibly flush the stream.
```

6.代码文件的结构

这三个概念实际上都是模块，
只不过是个体和集合的区别

■ 模块

- 是.py文件，里面定义了一些函数和变量，需要的时候就可以导入这些模块
- 仍具有独立功能的代码文件

■ 包

- 在模块之上的概念，放置模块的文件夹
- 为了方便管理而将文件进行打包。

```
package_a
├── __init__.py
├── module_a1.py
└── module_a2.py
```

■ 库：

- 具有相关功能模块的集合，Python的一大特色之一
- 即具有强大的标准库、第三方库以及自定义模块。
- 标准库：
 - 就是下载安装的python里那些自带的模块，有一些模块是看不到的比如像sys模块
- 第三方库：
 - 就是由其他的第三方机构，发布的具有特定功能的模块。
- 自定义模块：
 - 用户自己可以自行编写模块，然后使用。



Python 编码规范(Google)

<https://www.runoob.com/w3cnote/google-python-styleguide.html>

■ 分号

- 不要在行尾加分号, 也不要加分号将两条命令放在同一行。

■ 行长度

- 每行不超过**80**个字符, 以下情况除外:
 - 长的导入模块语句
 - 注释里的URL
 - 不要使用反斜杠连接行。
- 可以在表达式外围增加一对额外的圆括号代表一行

```
推荐: foo_bar(self, width, height, color='black', design=None, x='foo',  
              emphasis=None, highlight=0)
```

```
if (width == 0 and height == 0 and  
    color == 'red' and emphasis == 'strong'):
```

如果一个文本字符串在一行放不下, 可以使用圆括号来实现隐式行连接:

```
x = ('这是一个非常长非常长非常长非常长 '  
     '非常长非常长非常长非常长非常长非常长的字符串')
```

在注释中, 如果必要, 将长的URL放在一行上。

```
Yes: # See details at  
     # http://www.example.com/us/developer/documentation/api/content/v2.0/csv_file_name_extension_full_specification.h  
tml
```

```
No:  # See details at  
     # http://www.example.com/us/developer/documentation/api/content/\  
     # v2.0/csv_file_name_extension_full_specification.html
```

3.1.3 Python语法规则

■ 空行

- 顶级定义之间空两行, 比如函数或者类定义.
- 方法定义, 类定义与第一个方法之间, 都应该空一行.

■ 空格

- 按照标准的排版规范来使用标点两边的空格
- 括号内不要有空格!
- 用4个空格来缩进代码
- 绝对不要用**tab**, 也不要**tab**和空格混用.

```
No: spam( ham[ 1 ], { eggs: 2 }, [ ] )
```



3.1.3 Python语法规则

- 不要在逗号, 分号, 冒号前面加空格, 但应该在它们后面加(除了在行尾).

Yes: `if x == 4:`

`print x, y`

`x, y = y, x`



No: `if x == 4 :`

`print x , y`

`x , y = y , x`



- 参数列表, 索引或切片的左括号前不应加空格.

Yes: `spam(1)`

no: `spam (1)`

Yes: `dict['key'] = list[index]`

No: `dict ['key'] = list [index]`

3.1.3 Python语法规则

- 在二元操作符两边都加上一个空格
 - 比如赋值(=), 比较(==, <, >, !=, <>, <=, >=, in, not in, is, is not), 布尔(and, or, not).
 - 算术操作符两边的空格该如何使用, 需要好好判断. 不过两侧务必要保持一致.
- 当 '=' 用于指示关键字参数或默认参数值时, 不要在其两侧使用空格.

Yes: `x == 1`

No: `x<1`

Yes: `def complex(real, imag=0.0): return magic(r=real, i=imag)`

No: `def complex(real, imag = 0.0): return magic(r = real, i = imag)`

3.1.3 Python语法规则

- 不要用空格来垂直对齐多行间的标记, 因为这会成为维护的负担(适用于:, #, =等):

Yes:

```
foo = 1000 # 注释  
long_name = 2 # 注释不需要对齐
```

```
dictionary = {  
    "foo": 1,  
    "long_name": 2,  
}
```

No:

```
foo      = 1000 # 注释  
long_name = 2   # 注释不需要对齐  
  
dictionary = {  
    "foo"      : 1,  
    "long_name": 2,  
}
```

3.1.3 Python语法规则

■ 导入格式

- 每个导入应该独占一行
- 导入总应该放在文件顶部, 位于模块注释和文档字符串之后, 模块全局变量和常量之前.
- 导入应该按照从最通用到最不通用的顺序分组:
 - 标准库导入
 - 第三方库导入
 - 应用程序指定导入
- 每种分组中, 应该根据每个模块的完整包路径按字典序排序, 忽略大小写.

```
Yes: import os  
import sys
```

```
No: import os, sys
```



3.1.3 Python语法规则

■ 命名约定

- 所谓"内部(**Internal**)"表示仅模块内可用, 或者, 在类内是保护或私有的.
- 用单下划线(**_**)开头表示模块变量或函数是**protected**的(使用 **import * from** 时不会包含).
- 用双下划线(**__**)开头的实例变量或方法表示类内私有.
- 将相关的类和顶级函数放在同一个模块里. 不像**Java**, 没必要限制一个类一个模块.
- 类名使用大写字母开头的单词(如**CapWords**, 即**Pascal**风格),
- 模块名应该用小写加下划线的方式(如**lower_with_under.py**)

Python之父Guido推荐的规范

Type	Public	Internal
Modules	lower_with_under	_lower_with_under
Packages	lower_with_under	
Classes	CapWords	_CapWords
Exceptions	CapWords	
Functions	lower_with_under()	_lower_with_under()
Global/Class Constants	CAPS_WITH_UNDER	_CAPS_WITH_UNDER
Global/Class Variables	lower_with_under	_lower_with_under
Instance Variables	lower_with_under	_lower_with_under (protected) or __lower_with_under (private)
Method Names	lower_with_under()	_lower_with_under() (protected) or __lower_with_under() (private)
Function/Method Parameters	lower_with_under	
Local Variables	lower_with_under	



3.1.3 Python语法规则

■ main函数

- Python没有一个特殊的函数作为脚本的入口点
- 但是最佳的做法是将入口点函数命名为`main()`，一般主功能在`main()`函数中。
- `main()`是可导入的。为避免导入时`main()`中的功能被执行，Python设系统内置变量“`__name__`”，判断一个模块是被导入还是独立运行（3.2.3例题讲解）

```
def main():  
    ...  
  
if __name__ == '__main__':  
    main()
```



3.1.3 Python语法规则

实例2：输出当前系统信息

- 创建一个代码文件用作模块，并命名为“`getenv.py`”
- 在模块中实现一个函数，令其打印系统信息。
- python内置了很多模块，但是这些模块功能仅限于通用的技术层面。
- 实际开发中每个公司都有各自的业务，有必要把自己业务的功能整合起来，这时就需要自定义模块了。

platform模块,提供很多方法获取操作系统的信息
如: platform.platform() 获取操作系统名称及版本号

代码 3-1: getenv.py

```
import platform
```

```
import sys
```

```
import os
```

```
def showENV():
```

```
    s = platform.platform()
```

```
    print("当前系统: ", s)
```

```
    p = sys.path
```

```
    print("当前安装路径: ", p)
```

```
    op = os.getcwd()
```

```
    print("当前代码路径: ", op)
```

```
    print("Python 版本信息: ", sys.version_info)
```

sys模块, 提供有关Python运行环境的变量和函数
如: sys.path返回模块的搜索路径

os模块, 提供与操作系统交互的接口。
如: os.getcwd()获取当前路径

#函数

#获取系统信息

#获取安装路径

#获取当前代码路径

`__name__` 是系统内置变量
代表所在模块名字，也即所在文件名

如果是主模块运行，调用函数showENV
代码运行后得到如下输出：

```
if __name__ == '__main__':  
    showENV()
```

#进行模块的单元测试

当前系统: Windows-7-6.1.7601-SP1

当前安装路径: ['', 'C:\\local\\Anaconda3\\lib\\site-packages\\pymysql-0.7.11-py3.5.egg', 'C:\\local\\Anaconda3\\python35.zip', 'C:\\local\\Anaconda3\\DLLs', 'C:\\local\\Anaconda3\\lib', 'C:\\local\\Anaconda3', 'c:\\local\\anaconda3\\lib\\site-packages\\setuptools-23.0.0-py3.5.egg', 'C:\\local\\Anaconda3\\lib\\site-packages', 'C:\\local\\Anaconda3\\lib\\site-packages\\Sphinx-1.4.1-py3.5.egg', 'C:\\local\\Anaconda3\\lib\\site-packages\\win32', 'C:\\local\\Anaconda3\\lib\\site-packages\\win32\\lib', 'C:\\local\\Anaconda3\\lib\\site-packages\\Pythonwin', 'C:\\local\\Anaconda3\\lib\\site-packages\\IPython\\extensions', 'C:\\Users\\Administrator\\.ipython']

当前代码路径: D:\\python2

Python 版本信息: sys.version_info(major=3, minor=5, micro=2, releaselevel='final', serial=0)