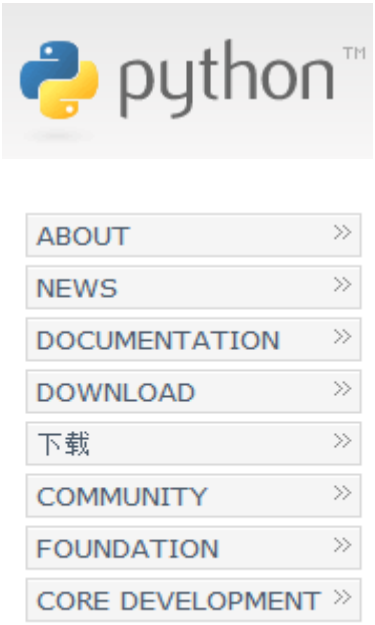
Python 入门

By nincompoop

参考教程：
http://www.cnblogs.com/known/archive/2010/07/31/1789249.html



官方介绍：Python 是一种简单易学，功能强大的编程语言，它有高效率的高层数据结构，简单而有效地实现面向对象编程。Python 简洁的语法和对动态输入的支持，再加上解释性语言的本质，使得它在大多数平台上的许多领域都是一个理想的脚本语言，特别适用于快速的应用程序开发。

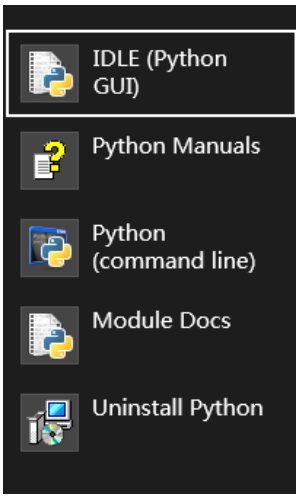
创造者：Guido van Rossum（吉多·范罗苏姆【荷兰】）

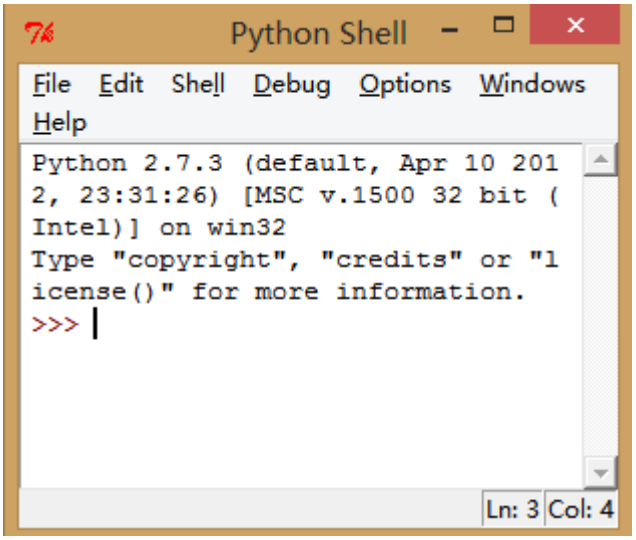
它的特色：简单、易学、免费、开源、高层语言、可移植性、解释性、面向对象、可扩展性、可嵌入性、丰富的库。

官网：http://python.org/

Python 支持命令式程序设计、面向对象程序设计、函数式编程、面向切面编程、泛型编程多种编程范式。与 Scheme、Ruby、Perl、Tcl 等动态语言一样，Python 具备垃圾回收功能，能够自动管理存储器使用。它经常被当作脚本语言用于处理系统管理任务和网络程序编写，然而它也非常适合完成各种高级任务。Python 虚拟机本身几乎可以在所有的作业系统中运行。

版本：3.x 与 2.x （Python 3.0 于 2008 年 12 月 3 日发布，**此版不完全兼容之前的 Python 源代码**。不过，很多新特性后来也被移植到旧的 Python 2.6/2.7 版本。）





Hello World 程序

下面是一个在标准输设备上输出 Hello World 的简单程序，这种程序通常作为开始学习编程语言时的第一个程序：

- **适用于 Python 3.0 以上版本以及 Python 2.6、Python 2.7**

```
print("Hello, world!")
```

- **适用于 Python 2.6 以下版本以及 Python 2.6、Python 2.7**

```
print "Hello, world!"
```

```
Python 2.7.3 (default, Apr 1(
32
Type "copyright", "credits" (
>>> print("hello world:)")
hello world:)
>>>
```

将这行程序码保存为 myhello.py。然后在 Linux 终端机下输入 `python myhello.py`，或者在 Windows 命令编辑字符下输入 `myhello.py` 运行。Python 也可以单步直译运行。运行 Python 解释器进入交互式命令行的环境，你可以在提示符号>>>旁输入 print("Hello, world!")，按 Enter 键输出结果：

- **适用于 Python 3.0 以上版本以及 Python 2.6、Python 2.7**

```
>>> print("Hello, world!")
Hello, world!
```

- **适用于 Python 2.6 以下版本以及 Python 2.6、Python 2.7**

```
>>> print "Hello, world!"
Hello, world!
```

注意，低于 3.0 版本的 Python，"Hello, world!"周围不需要括号。Python 3.x 与 Python 2.x 的 print 语法是不一样的。

Python 语法

Python 的设计目标之一是让代码具备高度的可阅读性。它设计时尽量使用其它语言经常使用的标点符号和英文单字，让代码看起来整洁美观。它不像其他的静态语言如 C、Pascal 那样需要重复书写声明语句，也不像它们的语法那样经常有特殊情况和惊喜。

缩进（告别{}!）

Python 开发者有意让违反了缩进规则的程序不能通过编译，以此来强制程序员养成良好的编程习惯。并且 Python 语言利用缩进表示语句块的开始和退出（Off-side 规则），而非使用花括号或者某种关键字。增加缩进表示语句块的开始，而减少缩进则表示语句块的退出。缩进成为了语法的一部分。例如 `if` 语句：

```
if age < 21:
    print("你不能買酒。")
    print("不過你能買口香糖。")
print("這句話處於if語句塊的外面。")
```

根据 PEP 的规定，必须使用 **4 个空格**来表示每级缩进。*使用 Tab 字符和其它数目的空格虽然都可以编译通过，但不符合编码规范。*支持 Tab 字符和其它数目的空格仅仅是为兼容很旧的的 Python 程序和某些有问题的编辑程序。

语句和控制流

- `if` 语句，当条件成立时运行语句块。经常与 `else`, `elif`(相当于 `else if`) 配合使用。
- `for` 语句，遍历列表、字符串、字典、集合等迭代器，依次处理迭代器中的每个元素。
- `while` 语句，当条件为真时，循环运行语句块。
- `try` 语句。与 `except`,`finally` 配合使用处理在程序运行中出现的异常情况。
- `class` 语句。用于定义类型。
- `def` 语句。用于定义函数和类型的方法。
- `pass` 语句。表示此行为空，不运行任何操作。
- `assert` 语句。用于程序调适阶段时测试运行条件是否满足。
- `with` 语句。Python2.6 以后定义的语法，在一个场景中运行语句块。比如，运行语句块前加密，然后在语句块运行退出后解密。
- `yield` 语句。在迭代器函数内使用，用于返回一个元素。自从 Python 2.5 版本以后。这个语句变成一个运算符。
- `raise` 语句。制造一个错误。
- `import` 语句。导入一个模块或包。
- `from import` 语句。从包导入模块或从模块导入某个对象。
- `import as` 语句。将导入的对象赋值给一个变量。
- `in` 语句。判断一个对象是否在一个字符串/列表/元组里。

表达式

Python 的表达式写法与 C/C++类似。只是在某些写法有所差别。

- 主要的算术运算符**与 C/C++类似**。`+, -, *, /, //, **, ~, %`分别表示加法或者取正、减法或者取负、乘法、除法、整除、乘方、取补、取模。`>>, <<`表示右移和左移。`&, |, ^`表示二进制的 `AND, OR, XOR` 运算。`>, <, ==, !=, <=, >=`用于比较两个表达式的值，分别表示大于、小于、等于、不等于、小于等于、大于等于。在这些运算符里面，`~, |, ^, &, <<, >>`必须应用于整数。

- Python 使用 **`and, or, not` 表示逻辑运算**。

- **`is, is not` 用于比较两个变量是否是同一个对象。`in, not in` 用于判断一个对象是否属于另外一个对象。**

- Python 支持“列表推导式”（list comprehension），比如计算 0-9 的平方和:

```
>>> sum(x * x for x in range(10))
285
>>>
```

- Python 使用 `lambda` 表示匿名函数。匿名函数体只**能是表达式**。比如：

```
>>> add=lambda x, y : x + y
>>> add(3,2)
5
```

- Python 使用 `y if cond else x` 表示条件表达式。意思是当 `cond` 为真时，表达式的值为 `y`，否则表达式的值为 `x`。相当于 C++和 Java 里的 `cond?y:x`。

```
>>> y= 3 if 5>6 else 2
>>> y
2
```

- Python 区分列表(list)和元组(tuple)两种类型。list 的写法是`[1,2,3]`，而 tuple 的写法是`(1,2,3)`。可以改变 list 中的元素，而不能改变 tuple。在某些情况下，tuple 的括号可以省略。tuple 对于赋值语句有特殊的处理。因此，可以同时赋值给多个变量，比如：

```
>>> x, y=1, 2 #同时给x,y赋值，最终结果：x=1, y=2
```

特别地，可以使用以下这种形式来交换两个变量的值：

```
>>> x, y=y, x #最终结果：y=1, x=2
```

- Python 使用**'(单引号)和"(双引号)**来表示字符串。与 Perl、Unix Shell 语言或者 Ruby、Groovy 等语言不一样，**两种符号作用相同**。一般地，如果字符串中出现了双引号，就使用单引号来表示字符串;反之则使用双引号。如果都没有出现，就依个人喜好选择。出现在字符串中的\(反斜杠)被解释为特殊字符，比如`\n` 表示换行符。**表达式前加 `r` 指示 Python 不解释字符串中出现的\。这种写法通常用于编写正则表达式或者 Windows 文件路径。**

- Python 支持列表切割(list slices)，可以取得完整列表的一部分。支持切割操作的类型有 `str, bytes, list, tuple` 等。它的语法是`...[left:right]`或者`...[left:right:stride]`。假定 `nums` 变量的值是`[1, 3, 5, 7, 8, 13, 20]`，那么下面几个语句为真：

```
>>> nums = [1,2,3,4,5,6,]
>>> nums
[1, 2, 3, 4, 5, 6]
>>> nums[:3]
[1, 2, 3]
>>> nums = [0,1,2,3,4,5,6,]
>>> nums
[0, 1, 2, 3, 4, 5, 6]
>>> nums[:3]
[0, 1, 2]
>>> nums[2:5]
[2, 3, 4]
>>> nums[1:]
[1, 2, 3, 4, 5, 6]
>>> nums[:]
[0, 1, 2, 3, 4, 5, 6]
>>> nums[1:5:2]
[1, 3]
>>> nums[-1:]
[6]
>>> nums[::-1]
[6, 5, 4, 3, 2, 1, 0]
```

- `nums[2:5] == [5, 7, 8]` 从下标为 2 的元素切割到下标为 5 的元素，但不包含下标为 5 的元素。
- `nums[1:] == [3, 5, 7, 8, 13, 20]` 切割到最后一个元素。
- `nums[:-3] == [1, 3, 5, 7]` 从最开始的元素一直切割到倒数第 3 个元素。
- `nums[:] == [1, 3, 5, 7, 8, 13, 20]` 返回所有元素。改变新的列表不会影响到 nums。
- `nums[1:5:2] == [3, 7]` 从下标为 1 的元素切割到下标为 5 的元素但不包含下标为 5 的元素，且步长为 2

函数

Python 的函数支持递归、默认参数值、可变参数，但不支持函数重载。为了增强代码的可读性，可以在函数后书写“文档字符串”(Documentation Strings，或者简称 docstrings)，用于解释函数的作用、参数的类型与意义、返回值类型与取值范围等。可以使用内置函数 `help()` 打印出函数的使用帮助。比如：

```
>>> def randint(a, b):
...     "Return random integer in range [a, b], including both end points."
...
>>> help(randint)
Help on function randint in module __main__:

randint(a, b)
    Return random integer in range [a, b], including both end points.
```

对象的方法

对象的方法是指绑定到对象的函数。调用对象方法的语法是 `instance.method(arguments)`。它等价于调用 `Class.method(instance, arguments)`。当定义对象方法时，必须显式地定义第一个参数，一般该参数名都使用 `self`，用于访问对象的内部数据。这里的 `self` 相当于 C++, Java 里面的 `this` 变量，但是我们还可以使用任何其它合法的参数名，比如 `this` 和 `mine` 等，**`self` 与 C++,Java 里面的 `this` 不完全一样**，它可以被看作是一个习惯性的用法，我们传入任何其它的合法名称都行，比如：

```
class Fish:
    def eat(self, food):
        if food is not None:
            self.hungry=False

class User:
    def __init__(myself, name):
        myself.name = name

#构造Fish的实例:
f=Fish()
#以下两种调用形式是等价的:
Fish.eat(f, "earthworm")
f.eat("earthworm")

u = User('username')

u.name
```

Python 认识一些以“__”开始并以“__”结束的特殊方法名，它们用于实现运算符重载和实现多种特殊功能。

类型

Python 采用***动态类型系统***。在编译的时候，Python 不会检查对象是否拥有被调用的方法或者属性，而是直至运行时，才做出检查。所以操作对象时可能会抛出异常。不过，虽然 Python 采用动态类型系统，它同时也是强类型的。Python 禁止没有明确定义的操作，比如数字加字符串。

与其它面向对象语言一样，Python 允许程序员定义类型。构造一个对象只需要像函数一样调用类型即可，比如，对于前面定义的 `Fish` 类型，使用 `Fish()`。类型本身也是特殊类型 `type` 的对象(`type` 类型本身也是 `type` 对象)，这种特殊的设计允许对类型进行反射编程。

Python 内置丰富的数据类型。与 Java、C++相比，这些数据类型有效地减少代码的长度。下面这个列表简要地描述了 Python 内置数据类型(适用于 Python 3.x)：

类型	描述	例子

`str`	一个由字符组成的不可更改的有串行。在 Python 3.x 里，字符串由 Unicode 字符组成。	`'Wikipedia'`
		`"Wikipedia"`
		`"""Spanning multiple lines"""`

`bytes`	一个由字节组成的不可更改的有串行。	`b'Some ASCII'` `b"Some ASCII"`
`list`	可以包含多种类型的可改变的有串行	`[4.0, 'string', True]`
`tuple`	可以包含多种类型的不可改变的有串行	`(4.0, 'string', True)`
`set`,`frozenset`	与数学中集合的概念类似。无序的、每个元素唯一。	`{4.0, 'string', True}` `frozenset([4.0, 'string', True])`
`dict`	一个可改变的由键值对组成的无串行。	`{'key1': 1.0, 3: False}`
`int`	精度不限的整数	`42`
`float`	浮点数。精度与系统相关。	`3.1415927`
`complex`	复数	`3+2.7j`
`bool`	逻辑值。只有两个值：真、假	`True` `False`

除了各种数据类型，Python 语言还用类型来表示函数、模块、类型本身、对象的方法、编译后的 Python 代码、运行时信息等等。因此，Python 具备很强的动态性。

开发环境

适用于 Python 的集成开发环境（IDE）软件，除了标准二进制发布包所附的 IDLE 之外，还有许多其他选择。其中有些软件设计有语法着色、语法检查、运行调试、

自动补全、智能感知等便利功能。由于 Python 的跨平台出身，这些软件往往也具备各种操作系统的版本或一定的移植性。

而很多并非集成开发环境软件的文本编辑器，也对 Python 有不同程度的支持，并且加上专门为 Python 设计的编辑器插件也会有很高的可用性。

专门为 Python 设计的 IDE 软件：

- Eric：基于 PyQt 的自由软件，功能强大。支持自动补全、智能感知、自动语法检查、工程管理、svn/cvs 集成、自动单元测试等功能。调试功能与 Visual Studio 和 Eclipse 类似。目前同时有两个版本。Eric4 支持 Python2.x，Eric5 支持 Python3.x。使用前需要先安装相应的 PyQt 版本。
- IDLE：Python“标准”IDE。一般随 Python 而安装，支持较少的编辑功能。调试功能也比较弱。
- Komodo 和 Komodo Edit：后者是前者的免费精简版。也可以用于 PHP，Ruby，Javascript，Perl，Web 和云开发。
- PyCharm：由 JetBrains 打造，该公司的 Java IDE 软件 IntelliJ 拥有海量的用户；PyCharm 具备一般 IDE 的功能，比如， 调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制等等，同时另外，PyCharm 还提供了一些很好的功能用于 Django 开发，同时支持 Google App Engine，更酷的是，PyCharm 支持 IronPython。PyCharm 是商业软件，目前已经到 2.5 版本。
- PythonWin：包含在 pywin32 内的编辑器，仅适用于 Windows。
- SPE（Stani's Python Editor）：功能较多的免费软件，依赖 wxPython。
- Ulipad：功能较全的免费软件，依赖 wxPython。
- WingIDE：可能是功能最全的 IDE，但不是免费软件。
- PyScripter：功能较全的开源 IDE，使用 Delphi 开发。

通用 IDE / 文本编辑器：

- eclipse + pydev 插件，目前对 Python 3.X 只支持到 3.0
- emacs + 插件
- NetBeans + 插件
- SlickEdit
- TextMate
- Python Tools for Visual Studio
- Vim + 插件
- Sublime Text + 插件
- EditPlus
- UltraEdit
- PSPad

官网（英文）：http://www.djangoproject.com/
自学材料（中文）：http://djangobook.py3k.cn/2.0/

Django 是一个开放源代码的 Web 应用框架，由 Python 写成。采用了 MVC 的软件设计模式，即模型 M，视图 V 和控制器 C。它最初是被开发来用于管理劳伦斯出版集团旗下的一些以新闻内容为主的网站的。并于 2005 年 7 月在 BSD 许可证下发布。这套框架是以比利时的吉普赛爵士吉他手 Django Reinhardt 来命名的。

Django 的主要目标是使得开发复杂的、数据库驱动的网站变得简单。Django 注重组件的重用性和“可插拔性”，敏捷开发和 DRY 法则（Don't Repeat Yourself）。在 Django 中 Python 被普遍使用，甚至包括配置文件和数据模型。

Django 可以运行在启用了 mod python 的 Apache 2 上，或是任何 WSGI 兼容的 Web 服务器。 Django 也有启动 FastCGI 服务的能力，因此能够应用于任何支持 FastCGI 的机器上。

下列数据库引擎被 Django 官方支持：

- PostgreSQL
- MySQL
- SQLite
- Oracle

Microsoft SQL Server 的适配器正在开发中，处于试验阶段。（注：SQL Server 的支持在 1.0 版本中已经被完全去除）

Django1.0 已经可以利用 Jython 运行在任何 J2EE 服务器。