Python入门教程

张尧立

2015年11月26日

Python是一种什么语言？

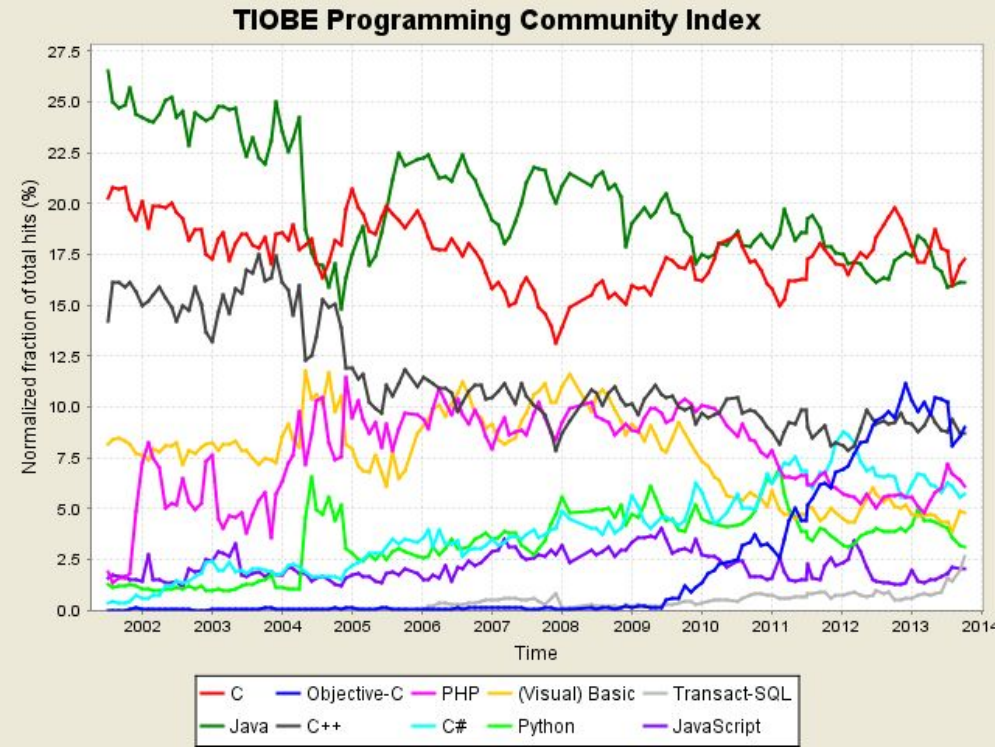
- Python是一种计算机程序设计语言。你可能已经听说过很多种流行的编程语言，比如非常难学的C语言，非常流行的Java语言，适合初学者的Basic语言，适合网页编程的JavaScript语言等等。
- 完成同一个任务，C语言要写1000行代码，Java只需要写100行，而Python可能只要20行。
- 所以Python是一种相当高级的语言。

Python可以做什么

- Python能做的
  - 日常任务
  - 网络游戏的后台
  - 科学计算
- Python不能做的
  - 写操作系统
  - 写3D游戏
  - 写手机应用

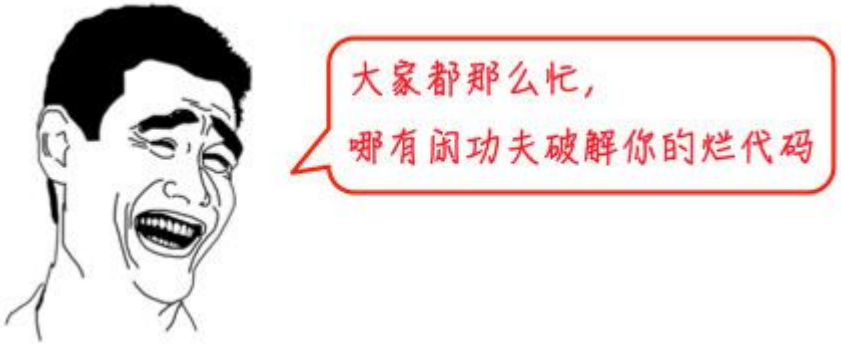
Python的诞生

- “龟叔”Guido van Rossum在1989年圣诞节期间，为了打发无聊的圣诞节而编写的一个编程语言。



Python的优点和缺点

- 优点
  - 简单易懂，
  - 入门容易，
  - 复杂的程序。
- 缺点
  - 运行速度慢，
  - 代码不能加密。



安装Python

- Python是跨平台的，它可以运行在Windows、Mac和各种Linux/Unix系统上。在Windows上写Python程序，放到Linux上也是能够运行的。

2.x还是3.x

- Python有两个版本，一个是2.x版，一个是3.x版，这两个版本是不兼容的。
- 请确保你的电脑上安装的Python版本是2.7.x。
- 在科学计算中，Python(x,y)是一个非常优秀的Python集成软件。本教程中的实例均使用Python(x,y)2.7版本完成。

第一个Python程序

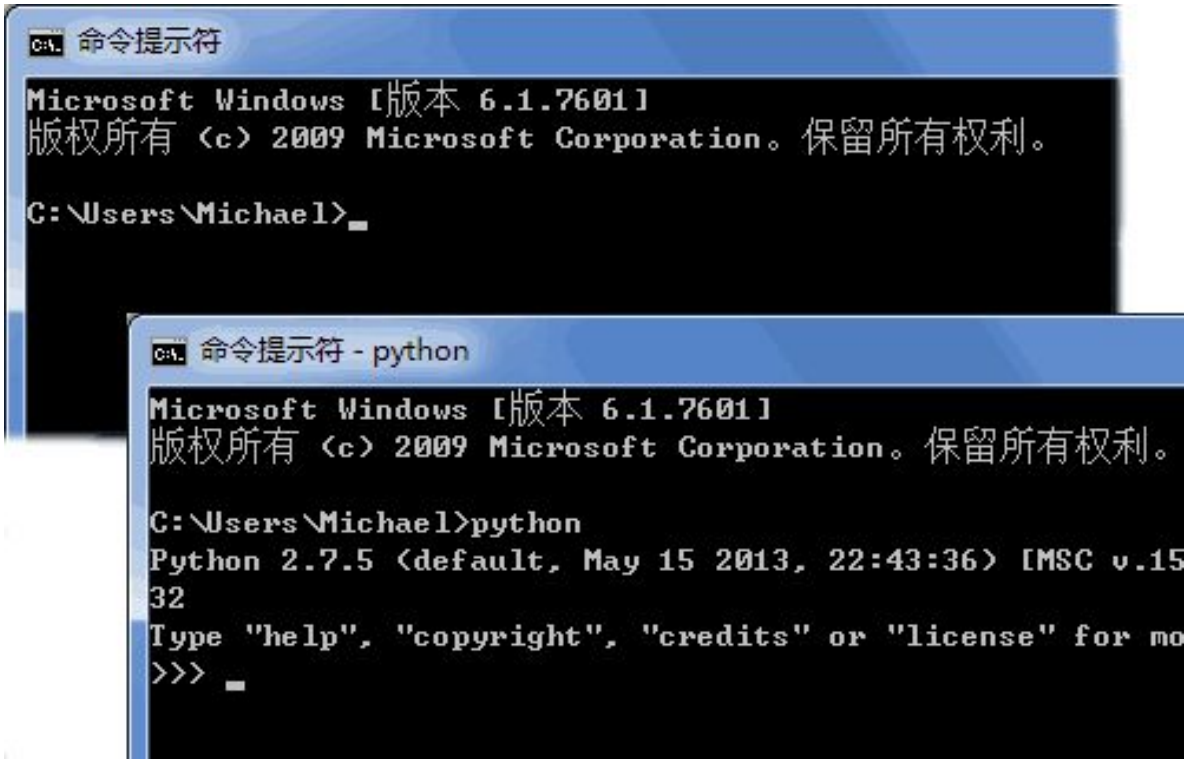
- 在交互式环境的提示符>>>下，直接输入代码，按回车，就可以立刻得到代码执行结果。
- 用exit()退出Python，我们的第一个Python程序完成！唯一的缺憾是没有保存下来，下次运行时还要再输入一遍代码。

使用文本编辑器

- 在Python的交互式命令行写程序，好处是一下就能得到结果，坏处是没法保存，下次还想运行的时候，还得再敲一遍。
- 对于初学者，推荐使用Python(x,y)自带的IDLE。绝对不能用Word和Windows自带的记事本。

注意！

- 请注意区分命令行模式和Python交互模式：



直接运行py文件

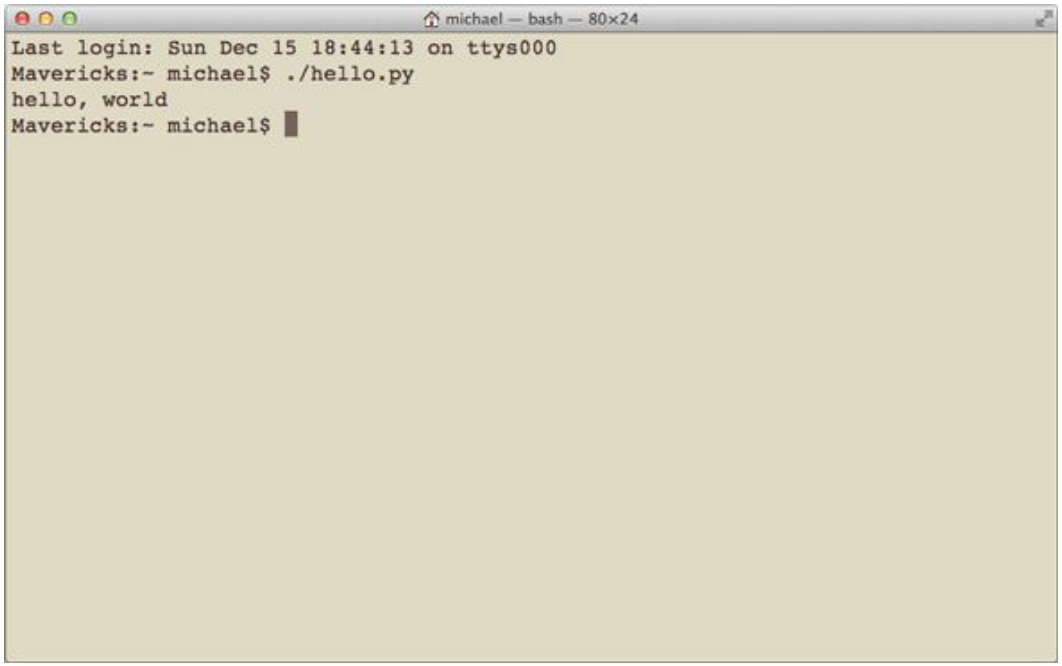
- 能不能像.exe文件那样直接运行.py文件呢？在Windows上是不行的，但是，在Mac和Linux上是可以的，方法是在.py文件的第一行加上：

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

- 然后，通过命令：

```
$ chmod a+x hello.py
```

- 就可以直接运行hello.py了，比如在Mac下运行：



输入和输出

- 输出
  - 用print加上字符串，就可以向屏幕上输出指定的文字。
- 输入
  - Python提供了一个raw_input，可以让用户输入字符串，并存放到一个变量里。

什么是变量？

- 设正方形的边长为a，则正方形的面积为a x a。把边长a看做一个变量，我们就可以根据a的值计算正方形的面积，比如：
  - 若a=2，则面积为a x a = 2 x 2 = 4；
  - 若a=3.5，则面积为a x a = 3.5 x 3.5 = 12.25。

Python基础

- Python的语法比较简单，采用缩进方式，写出来的代码就像下面的样子：

```
# print absolute value of an integer:
a = 100
if a >= 0:
    print a
else:
    print -a
```

数据类型

- 整数
- 浮点数
- 字符串
- 布尔值
- 空值

静态语言和动态语言

- 在Python中，等号=是赋值语句，可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量，例如：
  - a = 123 # a是整数print a
  - a = 'ABC' # a变为字符串print a
- 静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如C++是静态语言。
  - int a = 123; // a是整数类型变量
  - a = "ABC"; // 错误：不能把字符串赋给整型变量

赋值语句的等号

- 请不要把赋值语句的等号等同于数学的等号。比如下面的代码：
  - x = 10
  - x = x + 2
- 如果从数学上理解x = x + 2那无论如何是不成立的，在程序中，赋值语句先计算右侧的表达式x + 2，得到结果12，再赋给变量x。由于x之前的值是10，重新赋值后，x的值变成12。

常量

- 所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量：
  - PI = 3.14159265359

字符编码

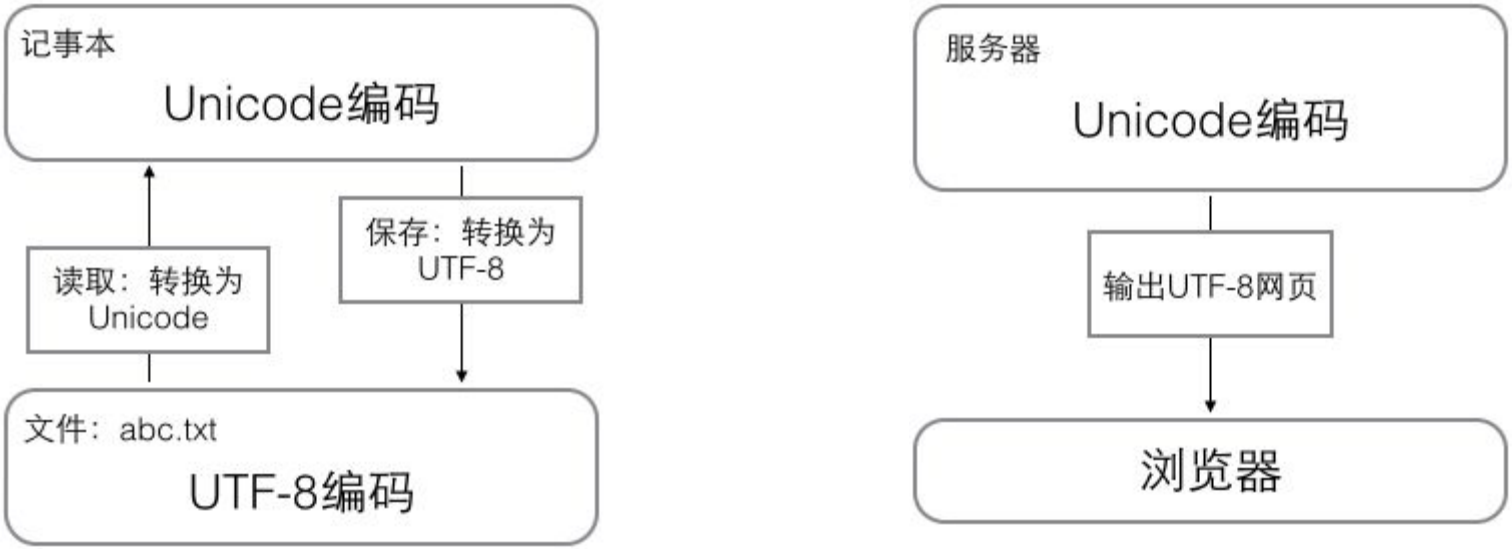
- 字符串也是一种数据类型，但是，字符串比较特殊的是还有一个编码问题。
- 由于计算机是美国人发明的，因此，最早只有127个字母被编码到计算机里，也就是大小写英文字母、数字和一些符号，这个编码表被称为ASCII编码，比如大写字母A的编码是65，小写字母z的编码是122。
- 但是要处理中文显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。

字符编码（续）

字符	ASCII	Unicode	UTF-8
A	01000001	00000000 01000001	01000001
中	x	01001110 00101101	11100100 10111000 10101101

- Unicode应运而生。Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题了。
- 本着节约的精神，又出现了把Unicode编码转化为“可变长编码”的UTF-8编码。

字符编码（续）



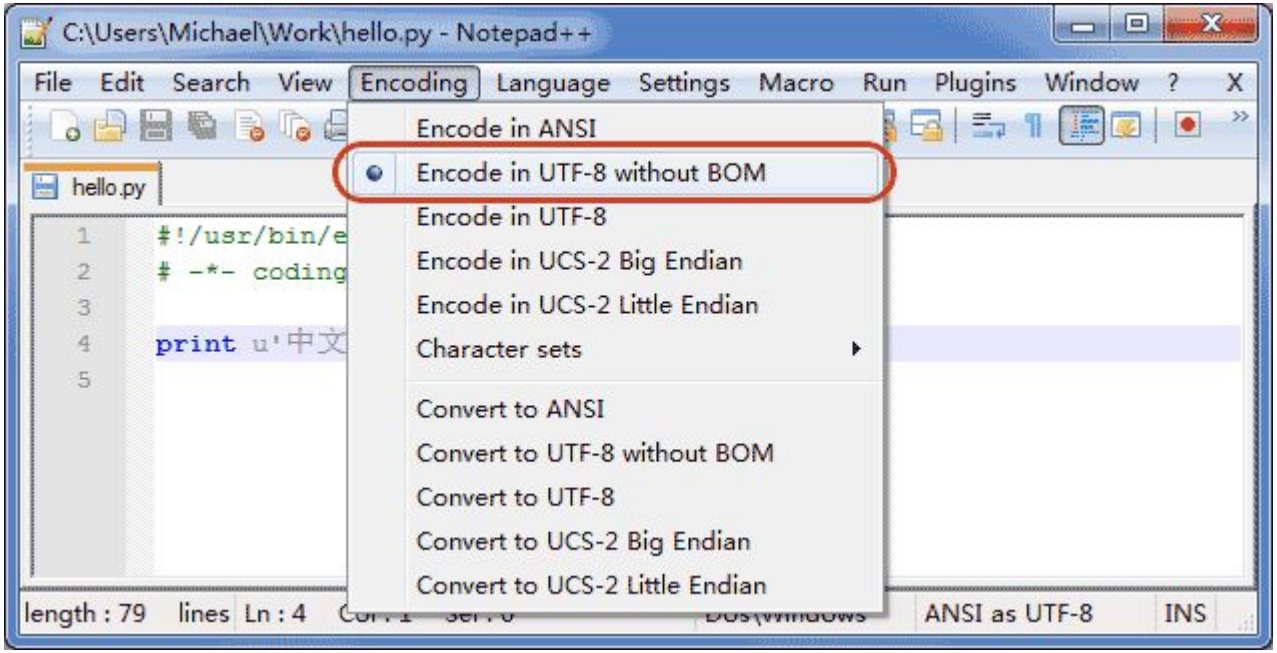
- 所以你看到很多网页的源码上会有类似<meta charset="UTF-8" />的信息，表示该网页正是用的UTF-8编码。

Python的字符串

- 由于Python源代码也是一个文本文件，所以，当你的源代码中包含中文的时候，在保存源代码时，就需要务必指定保存为UTF-8编码。当Python解释器读取源代码时，为了让它按UTF-8编码读取，我们通常在文件开头写上这两行：
  - #!/usr/bin/env python
  - # -*- coding: utf-8 -*-
- 第一行注释是为了告诉Linux/OS X系统，这是一个Python可执行程序，Windows系统会忽略这个注释；
- 第二行注释是为了告诉Python解释器，按照UTF-8编码读取源代码，否则，你在源代码中写的中文输出可能会有乱码。

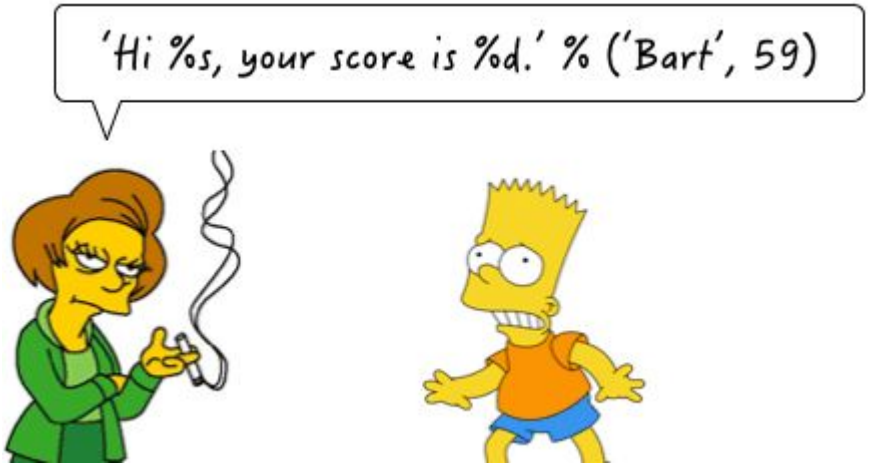
备注

- 对于有的编辑器，申明了UTF-8编码并不意味着你的.py文件就是UTF-8编码的，必须并且要确保编辑器正在使用UTF-8 without BOM编码：



格式化

- 最后一个常见的问题是如何输出格式化的字符串。我们经常会输出类似'亲爱的xxx你好！你xx月的话费是xx，余额是xx'之类的字符串，而xxx的内容都是根据变量变化的，所以，需要一种简便的格式化字符串的方式。



- 在Python中，采用的格式化方式和C语言是一致的，用%实现，举例如下：
- >>> 'Hello, %s' % 'world"

Hello, world'

- >>> 'Hi, %s, you have $%d.' % ('Michael', 1000000)

'Hi, Michael, you have $1000000.'

- 常见的占位符有：

%d	整数
%f	浮点数
%s	字符串
%x	十六进制整数

- 其中，格式化整数和浮点数还可以指定是否补0和整数与小数的位数：
- >>> '%2d-%02d' % (3, 1)

' 3-01'

- >>> '%.2f' % 3.1415926

'3.14'

- 如果你不太确定应该用什么，%s永远起作用，它会把任何数据类型转换为字符串：
- >>> 'Age: %s. Gender: %s' % (25, True)
- 'Age: 25. Gender: True'

list类型

- Python内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。
- 比如，列出班里所有同学的名字，就可以用一个list表示：
- >>> classmates = ['Michael', 'Bob', 'Tracy']
- >>> classmates

['Michael', 'Bob', 'Tracy']

- 用索引来访问list中每一个位置的元素，记得索引是从0开始的。

条件判断

- 计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。
- 比如，输入用户年龄，根据年龄打印不同的内容，在Python程序中，用if语句实现：

```
age = 20
if age >= 18:
    print 'your age is', age
    print 'adult'
```

- 根据Python的缩进规则，如果if语句判断是True，就把缩进的两行print语句执行了，否则，什么也不做。
- 也可以给if添加一个else语句，意思是，如果if判断是False，不要执行if的内容，去把else执行了。
- elif是else if的缩写，完全可以有多个elif，所以if语句的完整形式就是：

```
if <条件判断1>:
    <执行1>
elif <条件判断2>:
    <执行2>
elif <条件判断3>:
    <执行3>
else:
    <执行4>
```

if语句

- if语句执行有个特点，它是从上往下判断，如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else，所以，请测试并解释为什么下面的程序打印的是teenager：

```
age = 20
if age >= 6:
    print 'teenager'
elif age >= 18:
    print 'adult'
else:
    print 'kid'
```

if语句（续）

- if判断条件还可以简写，比如写：

```
if x:
    print 'True'
```

- 只要x是非零数值、非空字符串、非空list等，就判断为True，否则为False。

循环

- Python的循环有两种，一种是for...in循环，依次把list或tuple中的每个元素迭代出来。

```
names = ['Michael', 'Bob', 'Tracy']
for name in names:
   print name
```

- 如果要计算1-100的整数之和，从1写到100有点困难，幸好Python提供一个range()函数，可以生成一个整数序列，比如range(5)生成的序列是从0开始小于5的整数：
- >>> range(5)
- [0, 1, 2, 3, 4]
- range(101)就可以生成0-100的整数序列，计算如下：

```
sum = 0
for x in range(101):
    sum = sum + x
print sum
```

循环（续）

- 第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环。比如我们要计算100以内所有奇数之和，可以用while循环实现：

```
sum = 0
n = 99
while n > 0:
    sum = sum + n
    n = n - 2
print sum
```

再议raw_input

- 最后看一个有问题的条件判断。很多同学会用raw_input()读取用户的输入，这样可以自己输入，程序运行得更有意思：

```
birth = raw_input('birth: ')
if birth < 2000
    print '00前'
else:
    print '00后'
```

- 输入1982，结果却显示00后，这么简单的判断Python也能搞错？

函数

- 我们知道圆的面积计算公式为：

S = π r2

- 当我们知道半径r的值时，就可以根据公式计算出面积。假设我们需要计算3个不同大小的圆的面积：

```
r1 = 12.34
r2 = 9.08
r3 = 73.1
s1 = 3.14 * r1 * r1
s2 = 3.14 * r2 * r2
s3 = 3.14 * r3 * r3
```

- 当代码出现有规律的重复的时候，你就需要当心了，每次写3.14 * x * x不仅很麻烦，而且，如果要把3.14改成3.14159265359的时候，得全部替换。
- 有了函数，我们就不再每次写s = 3.14 * x * x，而是写成更有意义的函数调用s = area_of_circle(x)，而函数area_of_circle本身只需要写一次，就可以多次调用。

抽象

- 抽象是数学中非常常见的概念。举个例子：
- 计算数列的和，比如：1 + 2 + 3 + ... + 100，写起来十分不方便，于是数学家发明了求和符号Σ，可以把1 + 2 + 3 + ... + 100记作：

$$\sum_{n=1}^{100} n$$

调用函数

- Python内置了很多有用的函数，我们可以直接调用。
- 要调用一个函数，需要知道函数的名称和参数，比如求绝对值的函数abs，只有一个参数。可以直接从Python的官方网站查看文档：
- http://docs.python.org/2/library/functions.html#abs
- 也可以在交互式命令行通过help(abs)查看abs函数的帮助信息。

调用函数（续）

```
>>> abs(100)
100
>>> abs(-20)
20
>>> abs(12.34)
12.34
```

- 调用函数的时候，如果传入的参数数量不对，会报TypeError的错误，并且Python会明确地告诉你

数据类型转换

- Python内置的常用函数还包括数据类型转换函数，比如int()函数可以把其他数据类型转换为整数：

```
>>> int('123')
123
>>> int(12.34)
12
>>> float('12.34')
12.34
>>> str(1.23)
'1.23'
>>> unicode(100)
u'100'
>>> bool(1)
True
>>> bool('')
False
```

定义函数

- 在Python中，定义一个函数要使用def语句，依次写出函数名、括号、括号中的参数和冒号:，然后，在缩进块中编写函数体，函数的返回值用return语句返回。
- 自定义一个求绝对值的my_abs函数为例：

```
def my_abs(x):
    if x >= 0:
        return x
    else:
        return -x
```

Return

- 请注意，函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。因此，函数内部通过条件判断和循环可以实现非常复杂的逻辑。
- 如果没有return语句，函数执行完毕后也会返回结果，只是结果为None。
- return None可以简写为return。

空函数

- 如果想定义一个什么事也不做的空函数，可以用pass语句：

```
def nop():
    pass
```

- pass语句什么都不做，那有什么用？实际上pass可以用来作为占位符，比如现在还没想好怎么写函数的代码，就可以先放一个pass，让代码能运行起来。
- pass还可以用在其他语句里，比如：

```
if age >= 18:
    pass
```

- 缺少了pass，代码运行就会有语法错误。

返回多个值

- 函数可以返回多个值吗？答案是肯定的。
- 比如在游戏中经常需要从一个点移动到另一个点，给出坐标、位移和角度，就可以计算出新的新的坐标：

```
import math
def move(x, y, step, angle=0):
    nx = x + step * math.cos(angle)
    ny = y - step * math.sin(angle)
    return nx, ny
```

几个常用的库

- import math
- import numpy as np
- inport matplotlib.pyplot as plt

实例——numpy解线性方程组

```
import numpy as np
import matplotlib.pyplot as plt
a = np.array([[3,-1,0,0,0],[-1,2,-1,0,0],[0,-1,2,-1,0],[0,0,-1,2,-1],[0,0,0,-1,3]])
b = np.array([200,0,0,0,1000])
b = b.reshape(-1,1)
T = np.linalg.solve(a,b)
```