Python 基础

目录

第一章 前言

近年来，随着深度学习算法的逐步成熟，人工智能在相关的应用也在加快落地。随着人工智能的快速发展，国家相继出台一系列政策支持中国人工智能的发展，推动中国人工智能步入新阶段。2017 年 7 月 20 日，国务院印发了《新一代人工智能发展规划》，从国家层面推动中国人工智能的发展。另外，在 2018 年 1 月 16 日，教育部正式将人工智能、物联网、大数据处理正式划入高中新课标，这就意味着现在的学生 16 岁就要开始学习编程了！因此，面向少年的 AI 课程培训变得越来越重要。

为什么选择 Python？Python 的定位是“优雅”、“明确”、“简单”，所以 Python 程序看上去总是简单易懂，初学者学 Python，不但入门容易，而且将来深入下去，可以编写那些非常非常复杂的程序。

本教程的目的是对 Python 做基础介绍，学习 python 语言的基本语法，让学生能够利用 python 进行实践并解决一些问题，对编程有基本的了解。对 Python 的学习是人工智能实践的基础。

第二章 认识 Python

Python 简介

Python 是一种解释型、面向对象、动态数据类型的高级程序设计语言。Python 由 Guido van Rossum 于 1989 年底发明 ,第一个公开发行版发行于 1991 年。像 Perl 语言一样, Python 源代码同样遵循 GPL(GNU General Public License)协议。

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。Python 的设计具有很强的可读性，相比其他语言经常使用英文关键字，其他语言的一些标点符号，它具有比其他语言更有特色语法结构。

1)Python 是一种解释型语言： 这意味着开发过程中没有了编译这个环节。类似于 PHP 和 Perl 语言。

2)Python 是交互式语言： 这意味着，您可以在一个 Python 提示符，直接互动执行写你的程序。

3)Python 是面向对象语言: 这意味着 Python 支持面向对象的风格或代码封装在对象的编程技术。

4)Python 是初学者的语言：Python 对初级程序员而言，是一种伟大的语言，它支持广泛的应用程序开发，从简单的文字处理到 WWW 浏览器再到游戏。

Windows 下环境搭建

1）下载 python 安装包：访问 https://www.python.org/downloads/windows/，在下载列表中选择 Window 平台安装包，包格式为：python-XYZ.msi 文件 ， XYZ 为你要安装的版本号。本文安装 Python2.7.15,如图

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		045fb3440219a1f6923fefdabde63342	17496336	SIG
XZ compressed source tarball	Source release		a80ae3cc478460b922242f43a1b4094d	12642436	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	9ac8c85150147f679f213addd1e7d96e	25193631	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	223b71346316c3ec7a8dc8bff5476d84	23768240	SIG
Windows debug information files	Windows		4c61ef61d4c51d615cbe751480be01f8	25079974	SIG
Windows debug information files for 64-bit binaries	Windows		680bf74bad3700e6b756a84a56720949	25858214	SIG
Windows help file	Windows		297315472777f28368b052be734ba2ee	6252777	SIG
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64	0ffa44a86522f9a37b916b361eebc552	20246528	SIG
Windows x86 MSI installer	Windows		023e49c9fba54914ebc05c4662a93ffe	19304448	SIG

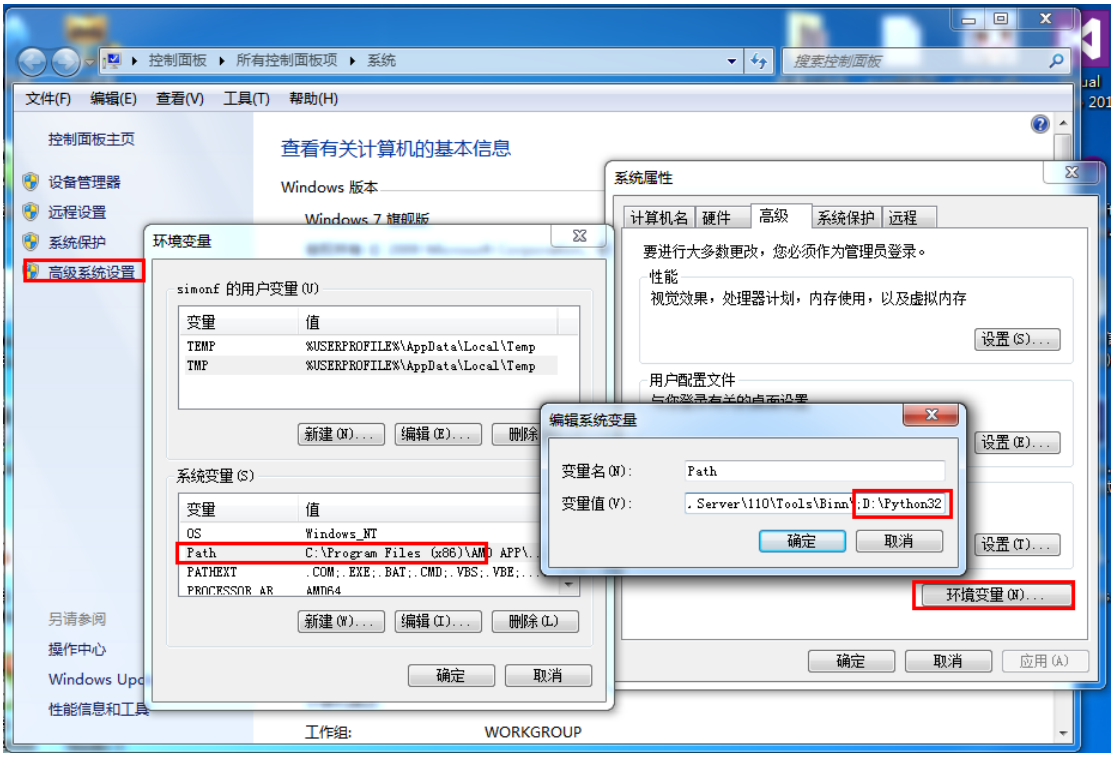
2）安装：下载后，双击下载包，进入 Python 安装向导，安装非常简单，你只需要使用默认的设置一直点击"下一步"直到安装完成即可。

3）环境变量配置：

程序和可执行文件可以在许多目录，而这些路径很可能不在操作系统提供可执行文件的搜索路径中，因此需要告诉系统那些路径下存在可执行文件。path(路径)存储在环境变量中，这是由操作系统维护的一个命名的字符串。这些变量包含可用的命令行解释器和其他程序的信息。

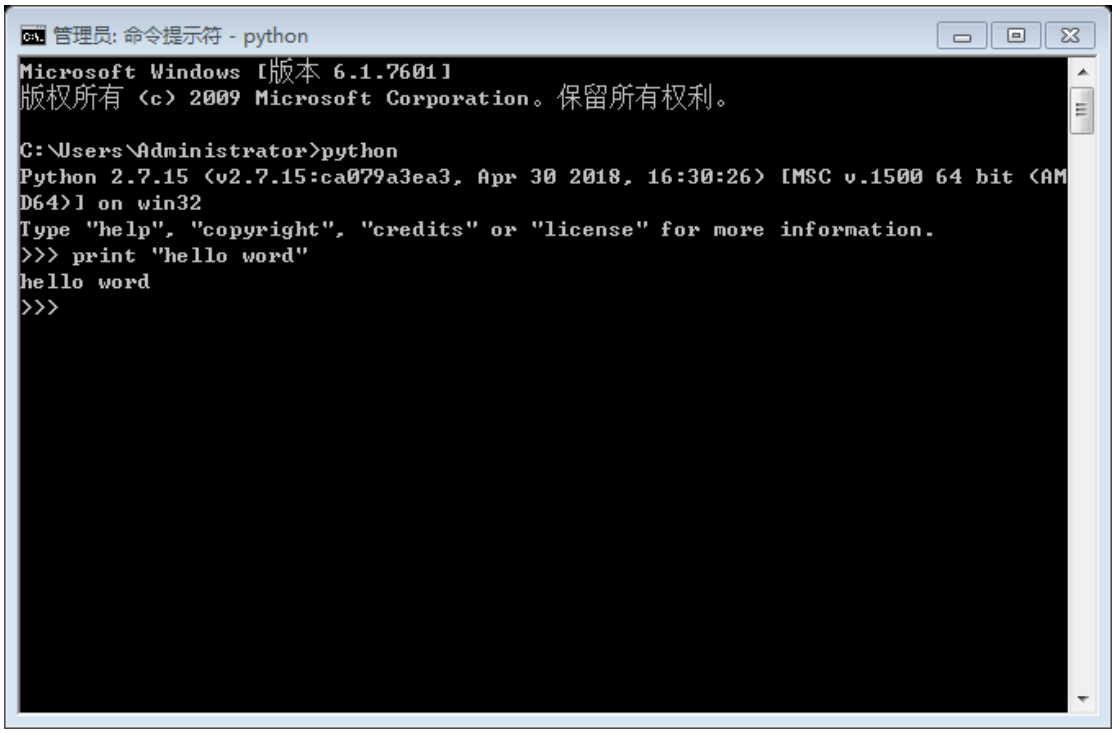
- 右键点击"计算机"，然后点击"属性"
- 然后点击"高级系统设置"

- 选择"系统变量"窗口下面的"Path",双击即可！
- 然后在"Path"行 ,添加 python 安装路径即可(我的 D:\Python32) ,所以在后面，添加该路径即可。 ps：记住，路径直接用分号"；"隔开！
- 最后设置成功以后，在 cmd 命令行，输入命令"python"，就可以有相关显示。



4）运行 python

配置好 python 环境，就可以执行 python 命令了。让我们先打出 hello word。点击 window 开始->附件->命令提示符。进入 Python 命令，如图：



Linux 下环境搭建

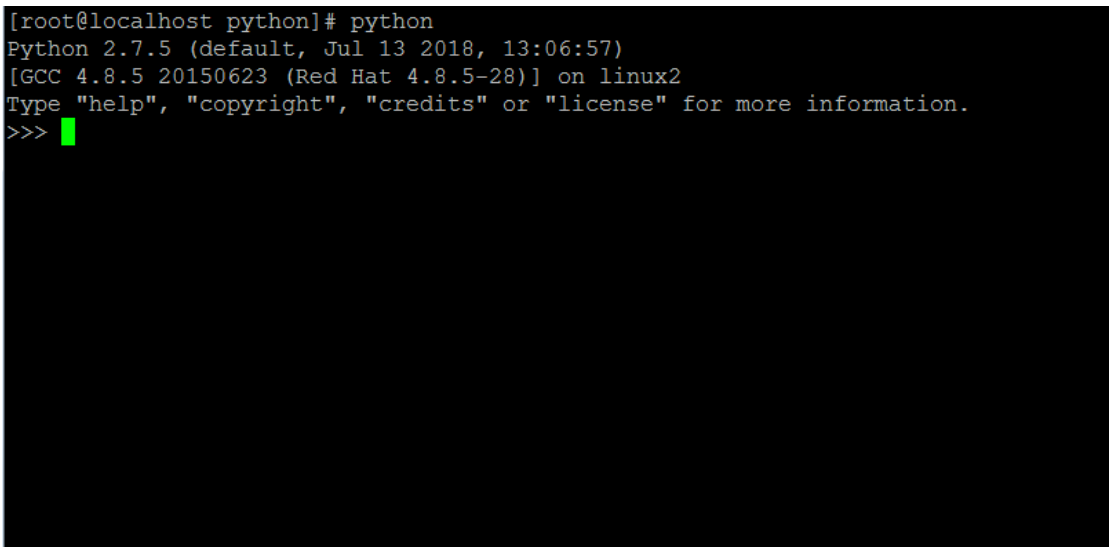
Linux 下一般已经安装了 Python 环境。如果没有安装 ,可用 Linux 自带的安装工具来安装。

不同 Linux 系统的安装工具略有差异。具体安装命令如下：

Centos: yum install python2

Ubuntu: apt-get install python2.7

安装完之后，在终端输入 python，即可看到如下界面：



Python 的版本

Python 的版本众多，不同版本间可能存在兼容性问题。一般使用的版本包括 Python2 和 Python3，这两个版本的代码不能混用。前面例子中安装的是 Python2。如果要安装 Python3，可选择相应的安装包（Windows）或相应的安装命令（Linux）。不同版本的 Python 可以共存在同一个系统中，用来执行不同版本的代码。

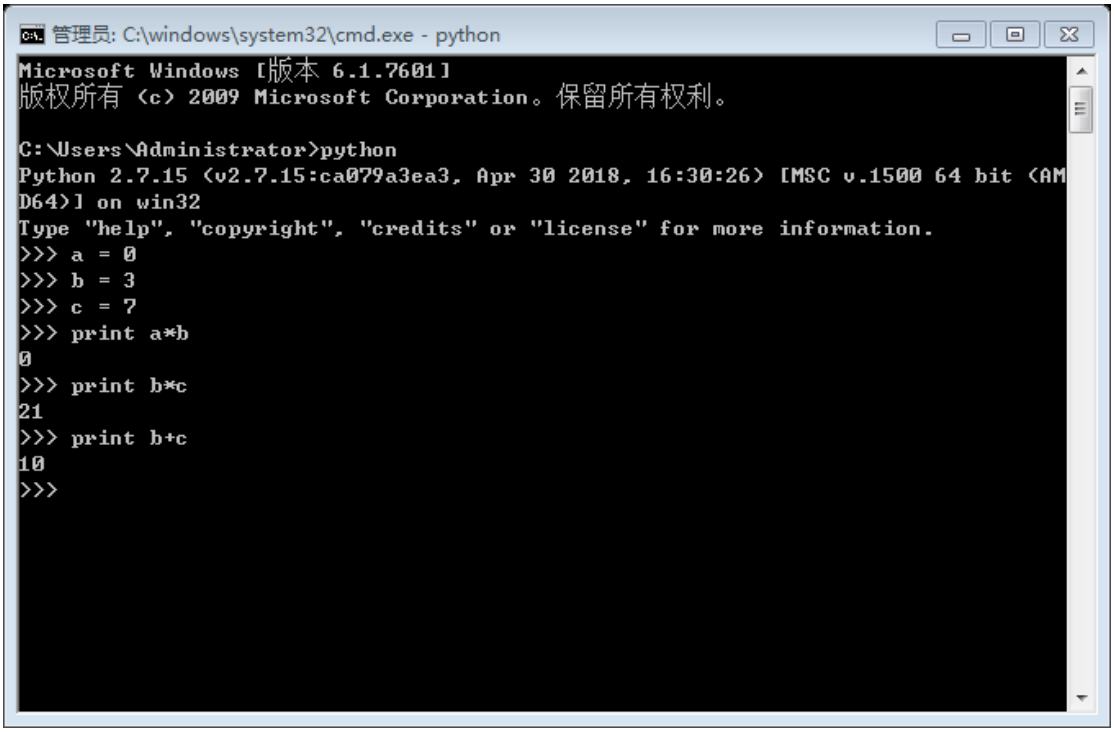
第三章 基础语法

在上一节中，我们对 python 有了简单的了解，并在自己的电脑上配置了 python 的开发环境，下面就可以开始 python 的学习了。 在本小节中，我们将学习对 python 的基本语法，通过该学习，我们将能够运用 python 处理一些简单问题，掌握基本的编程思想。因此，本小节分为 如何运行程序 、 python 的基本语法、Python 变量和运算符、python 循环语句、Python 函数。

3.1 如何运行程序

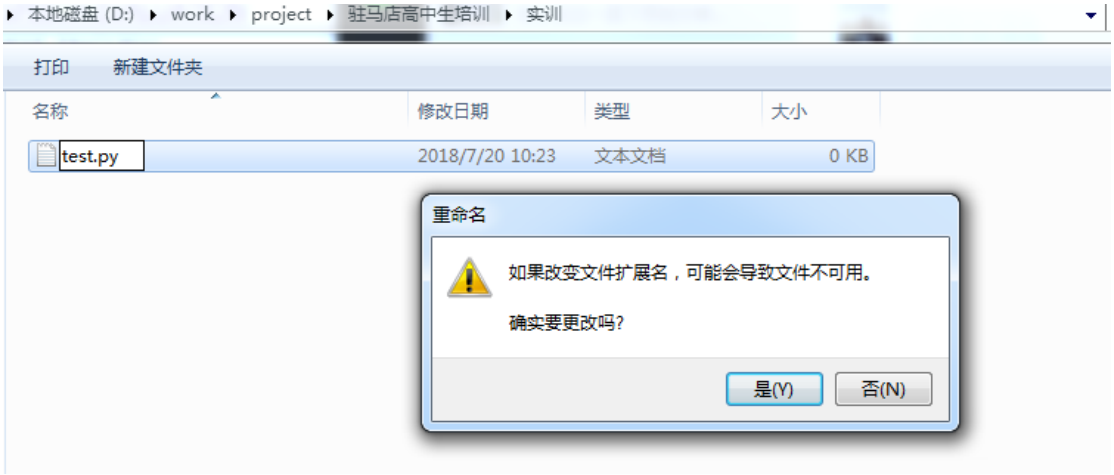
执行 python 程序，可以通过两种方式执行交互式编程和脚本式编程。

交互式编程：交互式编程不需要创建脚本文件，是通过 Python 解释器的交互模式进来编写代码。在 window 中，可以通过命令终端进行交互式编程，打开 windos 命令终端，输入如下命令,如图：



脚本式编程：通过脚本参数调用解释器开始执行脚本，直到脚本执行完毕。当脚本执行完成后，解释器不再有效。下面写一个简单的 Python 脚本程序。

1）创建一个扩展名为.py 的文件。如图，创建 test.py

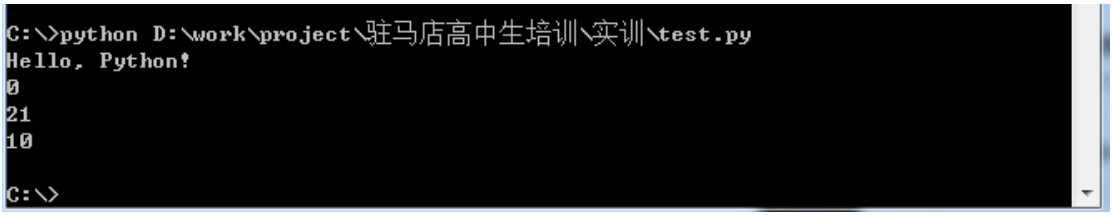


2）编写 python 程序：

```
print "Hello, Python!";

a = 0

b = 3

c = 7

print a*b

print b*c

print c+b
```

3）执行程序：在命令终端窗口输入

```
python test.py
```



3.2 基本语法

语言格式

什么是语言格式呢？就像英语一样，有固定的语法和符号表示。对于程序语言也一样，有自己的一套语言体系格式，即我们如何计算机进行

python 最具特色的就是用缩进来写模块。即通过缩进来表示逻辑结构。缩进的空白数量是可变的，但是所有代码块语句必须包含相同的缩进空白数量，这个必须严格执行。如下所示：

```
if True:
      print "True"
else:
   print "False"
```

另外，Python 可以使用引号(')、双引号(")、三引号(''' 或 """) 来表示字符串，引号的开始与结束必须的相同类型的。其中三引号可以由多行组成，编写多行文本的快捷语法。

```
word = 'word'
sentence = "这是一个句子。"
paragraph = """这是一个段落。
包含了多个语句"""
```

Python 注释

程序加注释对程序设计者本身是一个标记 ,在大型程序中 ,能及时有效的进行维护/修改。对程序阅读者来说 , 是一个解释 , 能让读者通彻的了解程序和设计者的思路。注释的原则是有助于对程序的阅读理解 , 在该加的地方都加了 , 注释不宜太多也不能太少 , 注释语言必须准确、易懂、简洁。

在 python 中注释分为单行注释和多行注释。

python 中单行注释采用 # 开头，如下

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-

# 文件名：test.py



# 第一个注释

print "Hello, Python!";  # 第二个注释
```

python 中多行注释使用三个单引号(''')或三个双引号(""")。

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-

# 文件名：test.py




'''
```

```
这是多行注释，使用单引号。

这是多行注释，使用单引号。

这是多行注释，使用单引号。

'''



"""

这是多行注释，使用双引号。

这是多行注释，使用双引号。

这是多行注释，使用双引号。

"""
```

程序输入输出

程序为什么需要输入输出？程序的输入输出相当于是和人类的交互 ,你告诉程序某些数值，叫做程序的输入；程序执行过程或执行结束，反馈给你结果叫做输出。

在 python 中输入用 input 等，输出用 print 等.

```
h1 = input()

print h1
```

print 默认输出是换行的，如果要实现不换行需要在变量末尾加上逗号 ，

变量和运算符

变量

变量存储在内存中的值。这就意味着在创建变量时会在内存中开辟一个空间。基于变量的数据类型，解释器会分配指定内存，并决定什么数据可以被存储在内存中。因此，变量可以指定不同的数据类型，这些变量可以存储整数，小数或字符。

Python 中的变量赋值不需要类型声明。每个变量在内存中创建，都包括变量的标识，名称和数据这些信息。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。等号（=）用来给变量赋值。等号（=）运算符左边是一个变量名,等号（=）运算符右边是存储在变量中的值

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-



counter = 100 #  赋值整型变量

miles = 1000.0 #  浮点型

name = "John" #  字符串



print counter

print miles

print name
```

运算符

什么是 python 的运算符？举个简单的例子 4 +5 = 9 。例子中 ,4 和 5 被称为操作数，"+" 称为运算符。python 中运算符包括 算术运算符、比较（关系）运算符、赋值运算符、逻辑运算符等。

算术运算符存在 加、减、乘、除、取模、幂、取整，具体如下

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x 除以 y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回 x 的 y 次幂	a**b 为 10 的 20 次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分（**向下取整**）	9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-



a = 21

b = 10

c = 0
```

```
c = a + b

print "1 - c 的值为：", c



c = a - b

print "2 - c 的值为：", c



c = a * b

print "3 - c 的值为：", c



c = a / b

print "4 - c 的值为：", c



c = a % b

print "5 - c 的值为：", c



# 修改变量 a 、b 、c

a = 2

b = 3

c = a**b

print "6 - c 的值为：", c
```

```
a = 10

b = 5

c = a//b

print "7 - c 的值为：", c
```

比较运算符就是对变量进行比较，如下

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等	(a <> b) 返回 true。这个运算符类似 != 。
>	大于 - 返回 x 是否大于 y	(a > b) 返回 False。
<	小于 - 返回 x 是否小于 y。所有比较运算符返回 1 表示真，返回 0 表示假。这分别与特殊的变量 True 和 False 等价。	(a < b) 返回 true。
>=	大于等于 - 返回 x 是否大于等于 y。	(a >= b) 返回 False。
<=	小于等于 - 返回 x 是否小于等于 y。	(a <= b) 返回 true。

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-



a = 21
```

```
b = 10

c = 0



if ( a == b ):

    print "1 - a  等于  b"

else:

    print "1 - a  不等于  b"



if ( a != b ):

    print "2 - a  不等于  b"

else:

    print "2 - a  等于  b"



if ( a <> b ):

    print "3 - a  不等于  b"

else:

    print "3 - a  等于  b"



if ( a < b ):

    print "4 - a  小于  b"

else:

    print "4 - a  大于等于  b"
```

```
if ( a > b ):
   print "5 - a  大于  b"
else:
   print "5 - a  小于等于  b"

# 修改变量 a 和 b 的值
a = 5
b = 20
if ( a <= b ):
   print "6 - a  小于等于  b"
else:
   print "6 - a  大于   b"

if ( b >= a ):
   print "7 - b  大于等于  a"
else:
   print "7 - b  小于  a"
```

赋值运算符

运算符	描述	实例

=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-



a = 21

b = 10

c = 0



c = a + b

print "1 - c 的值为：", c



c += a
```

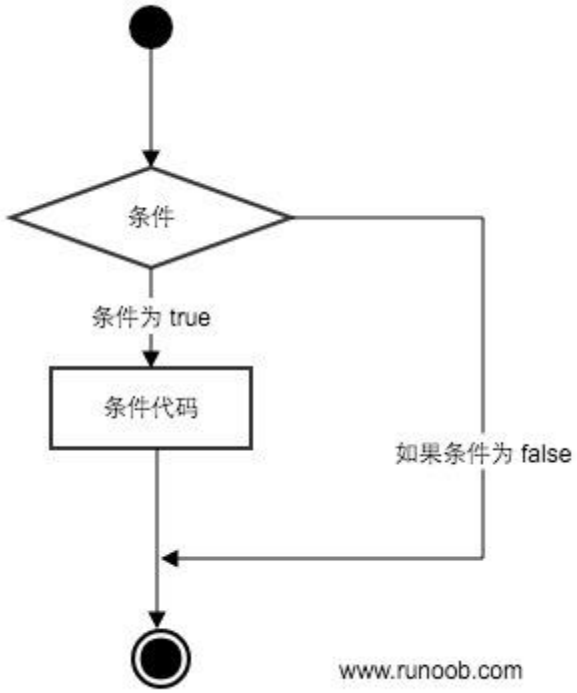
```
print "2 - c  的值为 ：", c



c *= a

print "3 - c  的值为 ：", c



c /= a

print "4 - c  的值为 ：", c



c = 2

c %= a

print "5 - c  的值为 ：", c



c **= a

print "6 - c  的值为 ：", c



c //= a

print "7 - c  的值为 ：", c
```

条件语句和循环语句

条件语句

Python 条件语句可以理解为人类的选择行为，当天下雨时，你外出就会打伞。Python 中

是通过一条或多条语句的执行结果(True 或者 False)来决定执行的代码块。可以通过下图来简单了解条件语句的执行过程:



Python 编程中条件语句通过 if 关键词来实现，if 语句用于控制程序的执行，基本形式为：

```
if 判断条件：
    执行语句……
else：
    执行语句……
```

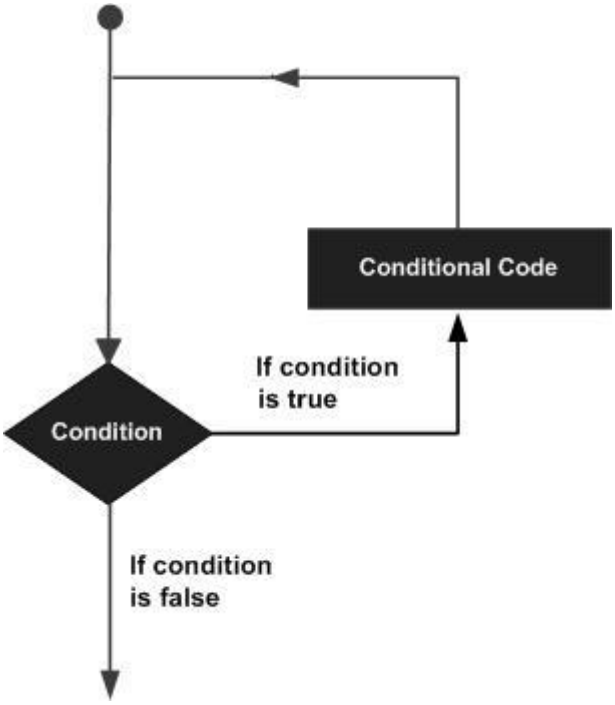
具体实例

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
```

```
# 例 1：if 基本用法



flag = False

name = 'luren'

if name == 'python':              # 判断变量否为'python'

    flag = True               # 条件成立时设置标志为真

    print 'welcome boss'      # 并输出欢迎信息

else:

    print name                    # 条件不成立时输出变量名称
```

循环语句

编程语言提供了各种控制结构，允许更复杂的执行路径。其中循环语句就是为了重新执行某些相同的命令而设计，即循环语句允许我们执行一个语句或语句组多次，下面是在大多数编程语言中的循环语句的一般形式。



在 python 中可以通过两个关键词实现程序的循环：for 和 while. 本文只介绍 for 在 python 中是如何实现循环的。具体 for 的形式如下;

```
for iterating_var in sequence:
    statements(s)
```

具体实例;

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

for num in range(10,20):  # 迭代 10 到 20 之间的数字
    print num, '是一个质数'
```

函数

什么是函数？举个例子，如我们生活中的家用电器，买来就可以直接通过命令控制实现某些功能，洗衣机放上衣服点击开始，就开始洗衣服了。其实程序中的函数就类似我们生活中的家用电器，他们可按照某种逻辑实现特定功能，我们写程序时直接拿过来用就行了。

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。函数能提高应用的模块性，和代码的重复利用率。

你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 def 关键词开头，后接函数标识符名称和圆括号()。
- 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- return [表达式] 结束函数，选择性地返回一个值给调用方。不带表达式的 return 相当于返回 None。

具体形式如下：

```
def functionname( parameters ):
    "函数_文档字符串"
    function_suite
    return [expression]
```

实例：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
```

```
# 定义函数
def printme( str ):
    "打印任何传入的字符串"
    print str;
    return;

# 调用函数
printme("我要调用用户自定义函数!");
printme("再次调用同一函数");

我要调用用户自定义函数!
再次调用同一函数
```

第四章 实践练习

本章我们尝试利用上面学习到的 python 知识进行问题求解。我们将练习三个小问题：数字累加、斐波那契数列、水仙花数；之后，我们做一个稍微复杂的问题，用计算机模拟随机现象。

（一） 数字累加

问题描述：需要通过编写 python 程序，求解出 1+2+3…+N 的和。

问题分析：本问题需要使用 for 循环 和 赋值运算符。

Python 编程如下：

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-

'''

'''

sum=0

a=1

N=1000

for n in range(a,N):

    sum += n

print sum
```

（二） 斐波那契数列

问题描述：斐波那契数列（Fibonacci sequence），又称黄金分割数列，指的是这样一个数列：0、1、1、2、3、5、8、13、21、34、……。

在数学上，费波那契数列是以递归的方法来定义：

```
F0 = 0        (n=0)

F1 = 1       (n=1)

Fn = F[n-1]+ F[n-2](n=>2)
```

Python 编程如下：

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-

'''

题目：斐波那契数列。



程序分析：斐波那契数列（Fibonacci sequence），又称黄金分割数列，指的是这样一个

数列：0、1、1、2、3、5、8、13、21、34、……。



在数学上，费波那契数列是以递归的方法来定义：

'''

def fib(n):

    a,b = 1,1

    for i in range(n-1):

        a,b = b,a+b

    return a



# 输出了第 10 个斐波那契数列

print fib(10)
```

（三） 水仙花数

问题描述：打印出所有的"水仙花数"，所谓"水仙花数"是指一个三位数，其各位数字立方和等于该数本身。例如：153 是一个"水仙花数"，因为 153=1 的三次方 + 5 的三次方 + 3 的

三次方。

程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。

Python 编程如下：

```
#!/usr/bin/python

# -*- coding: UTF-8 -*-

'''

题目：打印出所有的"水仙花数"，所谓"水仙花数"是指一个三位数，其各位数字立方和等

于该数本身。例如：153 是一个"水仙花数"，因为 153=1 的三次方 + 5 的三次方 + 3 的三

次方。

程序分析：利用 for 循环控制 100-999 个数，每个数分解出个位，十位，百位。

'''



for n in range(100,1000):

    i = n / 100

    j = n / 10 % 10

    k = n % 10

    if n == i ** 3 + j ** 3 + k ** 3:

        print n
```

(四) 蒙特卡洛方法模拟

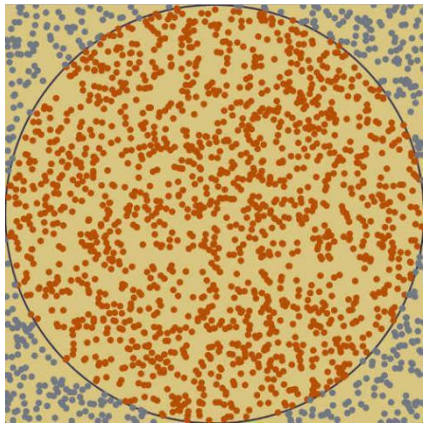
问题描述： 蒙特卡洛算法是以概率和统计的理论、方法为基础的一种计算方法，将所求解的问题同一定的概率模型相联系；用电子计算机实现统计模拟和抽样，以获得问题的近似

解，故又称统计模拟法或统计实验法。

蒙特卡洛算法：蒙特卡洛是美国摩纳哥的一个城市，以赌博闻名于世。蒙特卡洛算法借用这一城市的名称是为了象征性的表明该方法的概率统计特点。蒙特卡洛算法作为一种计算方法 ,是由 S.M.乌拉姆和 J.冯诺依曼在 20 世纪 40 年代中叶为研制核武器的需要而提出的。蒙特卡洛方法的基本思想虽然早已被人提出 ,例如在古典概率中的著名法国数学家布丰利用投针求圆周率 Pi 值，却很少被使用。直到电子计算机出现后，使得人们可以通过电子计算机来模拟巨大数目的随机试验过程，使得蒙特卡洛方法得到广泛地应用。

用蒙特卡洛投点法计算 Pi 的值在一个边长为 a 的正方形内一均匀概率随机投点 ,该点落在此正方形的内切圆中的概率即为内切圆与正方形的面积比值，即：

$$\frac{Area\ of\ Circle}{Area\ of\ Square} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$



那么，对于此问题，我们如何用计算机来模型呢？

```
from random import random

from time import perf_counter

'''
```

```
思路即是随机生成点，落在正方形内。计算正方形内的圆内落点与正方形内落点之比，
近似为面积之比，随机数越随机，数量越大越准确。
'''
def calPI(N = 100):
    hits = 0
    start = perf_counter()
    for i in range(1, N*N+1):
        x, y = random(), random()
        dist = pow(x ** 2 + y ** 2, 0.5)
        if dist <= 1.0:
            hits += 1
    pi = (hits * 4) / (N * N)
    use_time = perf_counter() - start
    return pi, use_time

PI, use_time = calPI(1000)
print('\nuse Monte Carlo method to calculate PI: {}'.format(PI))
print('use time: {} s'.format(use_time))
```

参考资料

1. Python 简易教程：http://www.runoob.com/python/python-tutorial.html
2. W3C 教室：https://www.w3cschool.cn/python/
3. 慕课教程：https://www.imooc.com/learn/177