# 简介

Python：优雅、明确、简单、跨平台

适合领域：Web网站和各种网络服务，系统工具和脚本，作为“胶水”语言将其他语言开发的模块包装起来方便使用。

不适合领域：贴近硬件的代码，移动开发，游戏开发。

实际应用：豆瓣、Youtube…

Pyhton不能加密。

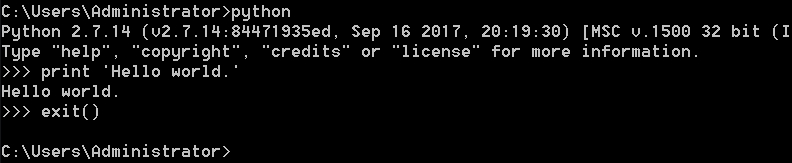
和其它流行语言对比：



2.7版和3.3版本，不兼容；



# 第一个Python程序



解释器解释运行，源代码编译为pyc字节码，字节码会被缓存，提供下次运行速度，根据时间戳和版本判断是否需要重新编译。在PVM虚拟机中运行。

Source（py） -> Byte code（pyc） -> Runtime（PVM）

# Python中数据类型

在Python中，能够直接处理的数据类型有以下几种：

## 一、整数

Python可以处理任意大小的整数，当然包括负整数，在Python程序中，整数的表示方法和数学上的写法一模一样，例如：1，100，-8080，0，等等。

计算机由于使用二进制，所以，有时候用十六进制表示整数比较方便，**十六进制用0x前缀**和0-9，a-f表示，例如：0xff00，0xa5b4c3d2，等等。

## 二、浮点数

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的，比如，1.23x10^9和12.3x10^8是相等的。浮点数可以用数学写法，如1.23，3.14，-9.01，等等。但是**对于很大或很小的浮点数，就必须用科学计数法表示**，把10用e替代，1.23x10^9就是1.23e9，或者12.3e8，0.000012可以写成1.2e-5，等等。

整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。

## 三、字符串

字符串是以''或""括起来的任意文本，比如'abc'，"xyz"等等。请注意，''或""本身只是一种表示方式，不是字符串的一部分，因此，字符串'abc'只有a，b，c这3个字符。

## 四、布尔值

布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、False表示布尔值（请注意大小写），也可以通过布尔运算计算出来。

布尔值可以用and、or和not运算。

1. and运算是与运算，只有所有都为 True，and运算结果才是 True。
2. or运算是或运算，只要其中有一个为 True，or 运算结果就是 True。
3. not运算是非运算，它是一个单目运算符，把 True 变成 False，False 变成 True。

## 五、空值

空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值。

此外，Python还提供了列表、字典等多种数据类型，还允许创建自定义数据类型。

# Python之print语句

print语句可以向屏幕上输出指定的文字。比如输出'hello, world'

print语句也可以跟上多个字符串，用逗号“,”隔开，就可以连成一串输出：

>>> print 'The quick brown fox', 'jumps over', 'the lazy dog'

The quick brown fox jumps over the lazy dog

print会依次打印每个字符串，遇到逗号“,”会输出一个空格。

# Python的注释

程序运行的时候，Python解释器会直接忽略掉注释，所以，有没有注释不影响程序的执行结果，但是影响到别人能不能看懂你的代码。

Python的注释以 # 开头，后面的文字直到行尾都算注释

# 这一行全部都是注释...

print 'hello' # 这也是注释

# 变量

在Python程序中，变量是用一个变量名表示，变量名必须是**大小写英文、数字和下划线（\_）的组合，且不能用数字开头。**

强类型，具有动态性（变量表中的引用没有类型限制），通过引用计数器判断对象是否需要被垃圾回收。

a = “a”

b = “b”

变量在第一次赋值时生成。不赋值将出错。

## 共享引用

多个变量指向同一个对象。

a = “ab”;

b = “ab”;

# id 方法用于获得对象的内存地址

id(“ab”) -> 48479488

id(a) -> 48479488

id (b) -> 48479488

判断两对象在内存中是否为同一对象可使用 **is** 关键字，等同于判断内存地址

**==** 判断字面值是否相同。

0 ~ 255 的值已经被缓存，即可以共享。

短的字符串也会被缓存，过长的不会被缓存（不能共享）。

# 数据类型

## 内置数据类型

检查数据类型方法：type()

### 数值类型

**int**：

1，-9，999（其大小上限取决于硬件，可认为无上限）：十进制

0b1101：二进制

0o177：八进制

0x9ff：16进制

**运算符**：

+、-、\*、/、%

\*\*：（次方 3 \*\* 2,3的二次方）

//：整除，

**常用函数**：

4 + int(“2”) -> 6

4 + int(“1101”,2) -> 17

同理有flaot()

bin()：将十进制转为二进制

oct()：将十进制转为8进制

hex()：将十进制转为16进制

round()：四舍五入

pow(3,2) -> 9：次方

**工具模块：**

Import math 导入**数学模块**

数轴： \* -4 -3 -2 -1 0 1 2 3 4 \*

math.floor(3.99) -> 3：向下取整（数轴往左取）天花板

math.floor(-3.14) -> -4

math.trunc(3.88) -> 3：截断（数轴往0取）

math.trunc(-3.44) -> -3

math.ceil(3.14) -> 4：向上取整（数轴往右取）地板

math.ceil(-3.24) -> -3

math.pi：PI

math.sqrt()：开方

import decimal 导入高精度模块

decimal.Decimal(‘0.111’) - decimal.Decimal(‘0.111’) = deciaml.Decimal(‘0.0’)：Decimal类构造函数传递浮点数用字符串的形式。

### 序列类型

分为可变和不可变序列。

#### 序列通用操作（以列表为例）

##### 生成序列

list(‘abcd’) -> [‘a’,’b’,’c’,’d’]

list(range(5)) -> [0,1,2,3,4]

##### 判断元素是否包含

a = [1,2,3,4,’a’,’b’]

1 in a -> True

2 not in a -> False

##### 连接序列

a = [1,2]

b = [4,5]

a + b -> [1,2,4,5]：a，b本身没有改变

##### 重复元素

[3] \* 4 -> [3,3,3,3]

##### 访问元素

a = [1,2,3,4,5]

a[0] -> 1

a[-1] -> 3

a[0:2] -> [1,2]

a[:3] - > [1,2,3]

a[3:] -> [4,5]

a[:] -> [1,2,3,4,5]

##### 按步长访问

a = [1,2,3,4,5]

a[::2] -> [1,3,5]

##### 获取序列长度

a = [1,2,3,4,5]

len(a) -> 5

##### 最小/最大/和

a = [1,2,3,4,5]

min(a) -> 1 **全局函数**

max(a) -> 5

sum(a) -> 15

##### 检索元素第一次出现的位置下标

a = [1,2,3,4,5]

a.index(2) -> 1 **方法**

##### 统计元素出现次数

a = [1,2,3,4,5]

a.count(2) -> 1

#### 不可变序列

1. 可包含任意对象的有序集合
2. 通过下标（位置偏移）索引访问
3. 固定长度，异质，可任意嵌套
4. 对象引用数组

不可变体现在不像列表一样可追加(extend)，扩展元素(append)，原位改变。

##### 元组tuple

###### 声明

t=(1,2,3,’1’)

t=1,2,3,’1’：简便写法

t=1,

只有一个元素时末尾“,”要加上如：t = (1,)，否则被当做基本类型处理。

t = tuple(range(2,4)) -> (2,8)

t[1] -> 3

t[0:2] -> (2,3)

###### 常用操作

t1=(1,)

t2=(2,)

t = t1 + t2 -> (1,2)

t = t1 \* 3 -> (1,1,1)

简单赋值/交换

a,b = 1,2 : a -> 1，b -> 2 利用tuple的语法同时给a和b赋值

a,b = b,a 交换a和b的值

##### range

type(range(5)) -> <class ‘range’>

range(1,21,2) 步长值为2

##### 字符串str

可用单引号或双引号

三引号换行将用 \n 自动替代

str = r’c:\ca\ccc\tt.txt’ 字符串前加 r（raw） 表示后续字符串中的 \ 不做转义字符。

str = b’abc123’ 字符串前加b表示用asci字节码表示（字节）（或十六进制）

str=’a’ ‘b’ -> ‘ab’ 空格会被忽略

###### 内置方法

支持序列的通用操作

s = ‘abcd123’

‘3’+str(4) -> ‘34’

s1 = s.replace(‘a’,’b’) -> ‘bbcd123’

s2=s.replace(‘b’,’f’,1) -> ‘afcd123只替换前一个

s2.capitalize() 单词第一个字母大写

su = s2.upper() 变为大写

sl = s2.lower() 变为小写

s2.startWith(‘a’)

s2.endWidth(‘a’)

s2.isnumeric() 是否为数字

s2.isalpha() 是否为字母

l = s2.split(‘a’) 拆分，无参时将整体作为**列表**的一个元素

‘:’.join([‘a’,’b’,’c’]) -> ‘a:b:c’ 连接

>>> a = 'a'

>>> b = 'b'

>>> c = 12

>>> 'A:{0},B:{1},C:{2},DA:{0}'.format(a,b,c) 字符串拼接，参数顺序的情况下 {} 中的下标可取消

'A:a,B:b,C:12,DA:a'

>>> 'A:{0},B:{1},C:{2},DA:{0},{depo}'.format(a,b,c,**depo='depo1**') depo变量要在参数中指定

'A:a,B:b,C:12,DA:depo1'

>>> '{0:10}={1:8}'.format('10个字符','22.33')

'10个字符 =22.33 ' 默认字符左对齐，数字右对齐

>>> '{0:>10}={1:<8}'.format('10个字符','22.33')

' 10个字符=22.33 ' 指定对齐方式

print('{:f},{:.2f},{:06.2f}'.format(3.1415926,3.1415926,3.1415926))

3.141593,3.14,003.14

**:06.2f** 总位数为6，浮点数形式，小数点后显示2位，位数不足前面补0

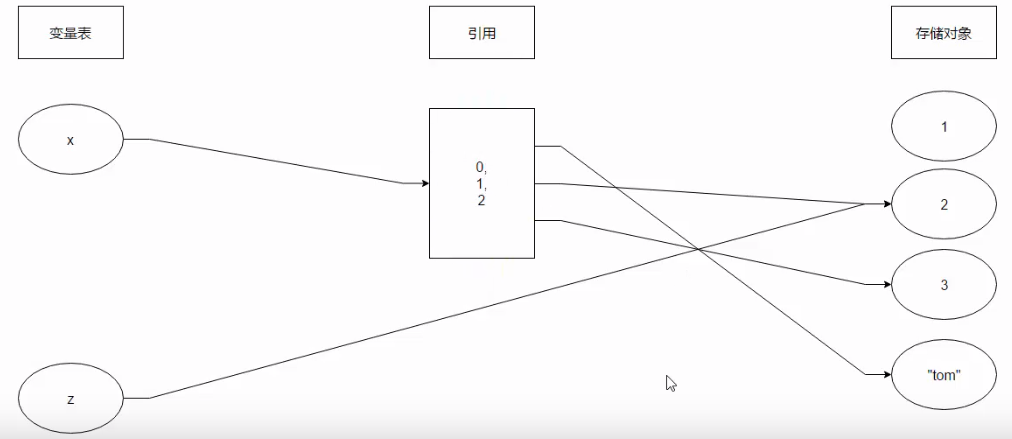
print('{:X},{:o},{:b}'.format(32,32,32))

20,40,100000 指定进制显示

#### 可变序列

1. 包含**任意对象**的**有序**序列
2. 通过下标索引（位置偏移）访问
3. 可变长度，异质，可任意嵌套
4. 支持原位修改
5. 对象引用数组

##### 列表list



###### 声明

scores = [23,4,24,56.3]

scores[0] -> 23 ：下标索引访问，从0开始

scores[-1] -> 56.3

scores[0:2] -> [23,4]：范围访问：含头不含尾

scores.append(78)：追加到最后

scores.append([2,33,’a’])：任意嵌套

scores[0] = 3：原位改变

###### 常用操作

a = [1,5,4,2]

排序

数值型：a.sort() -> [1,2,4,5]

自定义：c = [‘aa’,’vv’,’ff’]

c.sort(key=lambda x:x[-1])

函数：sorted(c,key=lambad n:n[1])

a = list(range(1,11)) -> [1,2,3,4,5,6,7,8,9,10]

改变指定范围的值

a[i:j] = t，t必须可迭代，当len(t) > j – i 时，t将插入，当len(t) < j-i 时替换并将长度不够的位置删除，len(t) = 1时将前j-i替换为t[0]

a[:3] = [11,22,33] -> [11,22,33,4,5,6,7,8,9,10]

a[0:3] = [11,22] -> [11,22,4,5,6,7,8,9,10]

a[:3] = [11] -> [11,11,11,4,5,6,7,8,9,10]

a[i:j:k] = t，加步长时替换的新值的数目要匹配

a[:6:2] = [0,0,0] -> [0,2,0,4,0,6,7,8,9,10]

a[:6:2] = [1] X错误

根据索引删除

del a[0]：删除列表a的第一个元素

del a[::2]

del a[3:6]

a[:2] = [] 等同于 del a[:2]

方法删除

a.remove(1) 删除第一个匹配的值

清空序列

a.clear()

a = [1,2,3]

追加序列

a.append(78) -> [1,2,3,78]：追加到最后

a.append([1,2]) -> [1,2,3,[1,2]]

扩展元素

a.extend([1,2]) -> [1,2,3,1,2]

插入元素

a.insert(0,99) -> [99,1,2,3]

a[0:0] = [99] -> [99,1,2,3]

弹出并删除值

a.pop() -> 3 [1,2]

a.pop(1) -> 2 [1,3]

反转序列

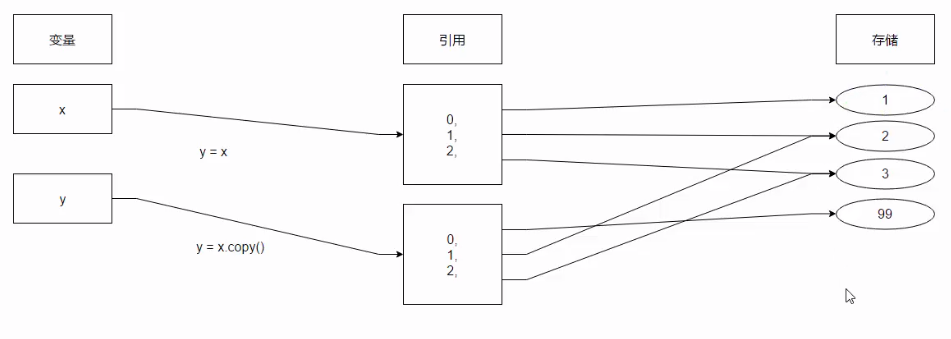
a.reverse() [3,2,1] ：影响实际序列

复制序列

l = a l is a -> True共享引用

l = a[:] l is a -> False复制序列

l = a.copy() l is a -> False复制序列



### 集合类型

### 映射类型

#### 可变映射

##### 字典表 dict

1. Key – vlue存储结构
2. 包含任意对象的**无序**集合
3. 可变长度，异质，可任意嵌套
4. 对象引用表 hash table

键只能是不可变的类型。

emplo = {'name':'tom','age':23,'salary':333.55} –> {'name': 'tom', 'age': 23, 'salary': 333.55}

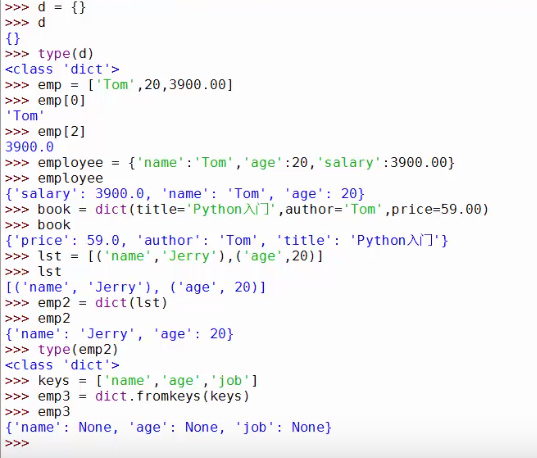
books = dict(title='Python',author='Tom',price=59.00) -> {'title': 'Python', 'author': 'Tom', 'price': 59.0}

>>> keys = ['a','b','c']

>>> emp3 = dict.fromkeys(keys)

>>> emp3

{'a': None, 'b': None, 'c': None}



###### 元素访问



book[‘key’] 元素不存在时会抛异常

book.get(‘key’) 元素不存在时返回None

book.get(‘key’,’not exist’) 元素不存在时返回’not exist’

book.keys() –> 视图（dict\_keys） 得到所有的键。

list(book.keys()) -> 转换为list

>>> for val in books.values(): 得到字典表直接遍历

... print(val)

...

Python

Tom

59.0

>>>

>>> for (k,v) in books.items():

... print('{} -> {}'.format(k,v))

...

title -> Python

author -> Tom

price -> 59.0

>>>

###### 常用操作

course = { … }

c = course.copy() 复制

c.clear() 清空

c[‘key’] = ‘newValue’ 修改

c.update(c1) 更新并合并

del c[‘key’] 删除

c.pop(‘key’) 弹出（删除并返回值），找不到抛异常

c.pop(‘key’,None) 弹出（删除并返回值），找不到返回None

c.popitem(‘key’) 弹出整个键值对

### 程序单元类型

### 其他类型

类型对象

空对象 None

布尔 bool：True、False

True本质为 1（等且只等于），False本质为0（等且只等于）。

True + 4 -> 5

False – 3 -> -3

注意：

bool(3) -> True

bool(-3) -> True

bool(0) -> False

bool(‘s’) -> True

bool([]) -> False

bool(‘’) -> False

真值测试：数字0或0.0，None，空对象（[]，’’），空的映射返回False

### 文件

open(‘路径’,模式’’,encoding=’编码’)

路径：

r’D:\GitRepositories\Pictures\BlogSystem’ 或

’D:\\GitRepositories\\Pictures\\BlogSystem’

模式：

‘r’：读

‘w’：写

‘rw’：读写

‘a’：追加

二进制 ‘\*b’：r，w，rw或a后加b。

f.read() 读取全部内容

f.close() 关闭

f.seek(0) 移动读取指正到指定位置

#### 文本文件读/写

f.read(4) 读取4个字符（字节）

f.readlines() 读取所有行到列表

f.readline() 读取下一行

for line in file 直接迭代遍历

cs.write(‘啊打发\naa’)

cs.flush() 同步到文件

cs.close() 同步并关闭文件

cs.readlines([‘a’,’b’]) 一次写入多个元素

name = [‘a’,’b’,’c’]

n = [n + ‘\n’ for n in name] -> [‘a\n’,’b\n’,’c\n’] 从已有类别中创建新列表

**上下文代码体**，上下文对象 f，最后f会自动关闭（无需显示调用f.close()）。

with open(‘’) as f:

for line in f:

print(line)

# 语句与语法

## 语句

### 赋值

x = 1

name, age = 'tom', 20

等同于：(name, age) = ('tom', 20) 本质为元组赋值

[name, age] = ['tom', 20] 本质为列表赋值

a,b,c,d=’abcd’ -> a=’a’ b=’b’ c=’c’ d=’d’ 前后数量要一致

a,\*b=’a**bcd**’ -> a=’a’ b=’bcd’

a,\*b,c=’a**bc**d’ -> a=’a’ b=’bc’ c=’d’

多目标赋值：a = b = 5

b += 10 -> b = b + 10

交换值：a,b = b,a

1. 赋值创建对象引用
2. 名称创建与首次赋值
3. 名称引用前必须赋值
4. 某些操作会执行隐式赋值

### 函数调用

import math

math.pow(x,2)

### 条件判断

if x > 3 :

print(‘大于3’)

### 迭代

for i in range(1,4):

print(i)

## 输入输出

s = input() 接收输入，接收为str型

input(‘提示’) 先输出提示信息

print(‘=’ \* 20) 输出20次 =

print(n,b,v,sep=’ - ’) 用-分隔n,b,v并输出

print(1, 2, 3, 4, **sep**='\n', **end**=' end') 用换行分隔，最后输出end

print('f={:08,.4f}'.format(math.pi \* 10000)) -> f=31,415.9265

### 函数

### 模块与命名空间

### 类

### 异常处理

## 语法

强制缩进，四个空格。

PEP8标准

猜数字

