



Chap2 Python Basic

---

# 第2章 Python基础

---

Department of Computer Science and Technology  
Department of University Basic Computer Teaching

## 2.1

# PYTHON程序 基本构成与风格

## 2.1.1 PYTHON程序基本构成

# 一个小程序

4



*# Filename: prog2-1.py*

*# For loop on a list*

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

prog = int(input("please input the value of prog: "))

for number in num:

    prog = prog \* number

print('The prog is: ', prog)

# 第1行

# 第2行

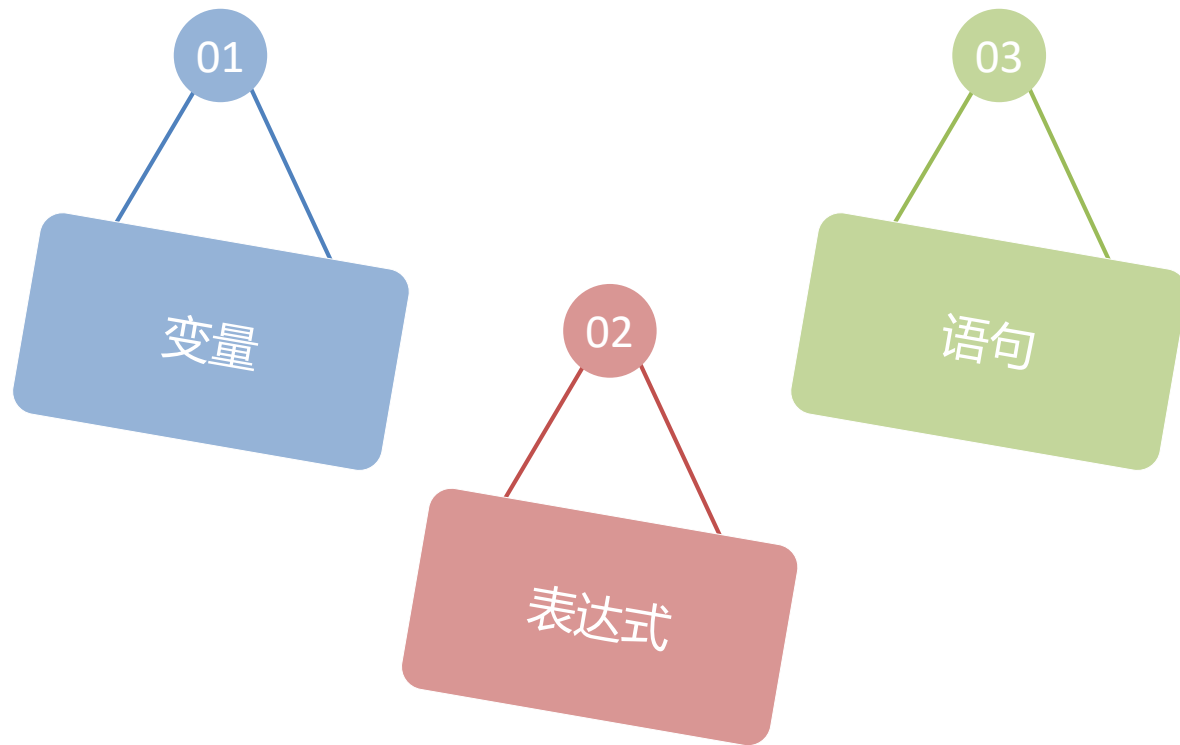
# 第3行

# 第4行

# 第5行

# 第6行

# Python基本构成



# Python输入 : input()函数

S  
ource

```
>>> price = input('input the stock price of Apple: ')
```

```
input the stock price of Apple: 109
```

```
>>> price
```

```
'109'
```

```
>>> type(price)
```

```
<class 'str'>
```

```
>>> price = int(input('input the stock price of Apple: '))
```

```
>>> price = eval(input('input the stock price of Apple: '))
```

input()  
返回的类型  
是字符型

# Python输出：print函数

7

- Python使用print函数实现输出：
  - print(变量)
  - print(字符串)



```
>>> myString = 'Hello, World!'
>>> print(myString)
Hello, World!
```

## 2.1.2 PYTHON程序设计风格



# Python 风格 (一)

9

注释



```
>>> # For loop on a list          # 第1行  
>>> print('The prog is: ', prog) # 第6行
```

## 缩进

01

增加缩进  
表示语句  
块的开始

Python用相  
同的缩进表示  
同级别语句块

02

减少缩进  
表示语句  
块的退出

03

S  
ource

# prog2-1.py

# For loop on a list

# 第1行

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

prog = int(input("please input the value of prog: "))

for number in num:

prog = prog \* number

print("The prog is: ", prog)

## 缩进



```
# prog2-1.py
```

```
# For loop on a list
```

# 第1行

```
num = [1, 2, 3, 4, 5]
```

```
prog = int(input("please input the value of prog: "))
```

```
for number in num:
```

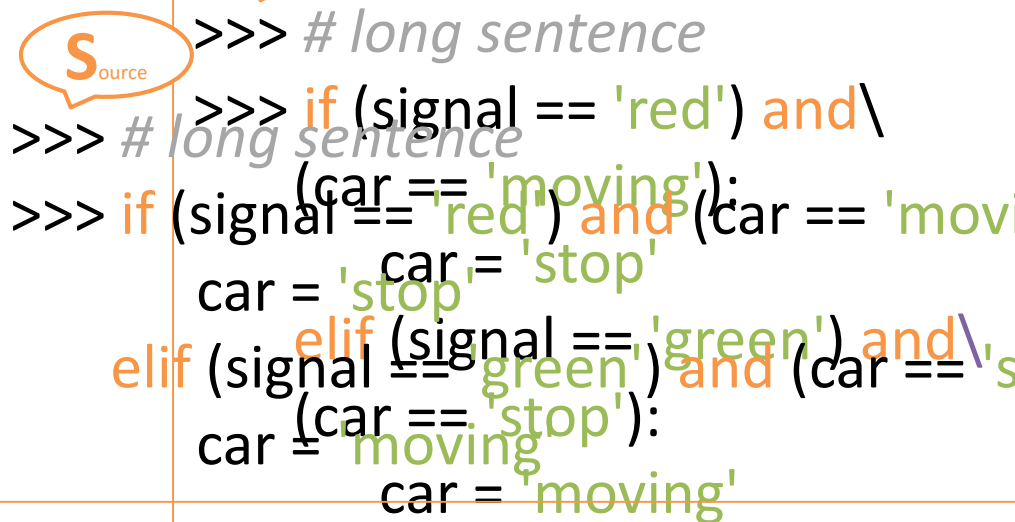
```
    prog = prog * number
```



```
print('The prog is: ', prog)
```

# Python 风格 (三)

12

续行



```
>>> # long sentence
>>> if (signal == 'red') and \
    (car == 'moving'):
>>> if (signal == 'red') and (car == 'moving'):
    car = 'stop'
    elif (signal == 'green') and \
    (car == 'stop'):
    car = 'moving'
```



## 续行

- 无需续行符可直接换行的两种情况：
  - 小括号、中括号、花括号的内部可以多行书写
  - 三引号包括下的字符串也可以跨行书写



```
>>> # triple quotes
>>> print("hi everybody,
welcome to python's MOOC course.
Here we can learn something about
python. Good lucky!")
```



# Python 风格 ( 四 )

14



```
>>> x = 'Today' ; y = 'is' ; z = 'Thursday' ; print(x, y, z)  
Today is Thursday
```



一行多语句



```
>>> x = 'Today'  
>>> y = 'is'  
>>> z = 'Thursday'  
>>> print(x, y, z)  
Today is Thursday
```

## 2.2

# PYTHON

## 语法基础

# 变量



```
>>> # variable  
>>> PI = 3.14159  
>>> pi = 'circumference ratio'  
>>> print(PI)  
3.14159  
>>> print(pi)  
circumference ratio
```



- 标识符是指Python语言中允许作为变量名或其他对象名称的有效符号
  - 首字符是字母或下划线
  - 其余可以是字母、下划线、数字
  - 大小写敏感(PI和pi是不同的标识符)



```
>>> # Identifier
```

```
>>> PI = 3.14159
```

```
>>> pi = 'circumference ratio'
```

```
>>> print(PI)
```

```
3.14159
```

```
>>> print(pi)
```

```
circumference ratio
```

# 特殊意义标识符



一个下划线或两个下划线开头的标识符对解释器来讲是有特殊意义，避免使用这种形式的标识符，也不能用作变量名。

- 关键字是Python语言的关键组成部分，不可随便作为其他对象的标识符


- 在一门语言中关键字是基本固定的集合
- 在 IDE 中常以不同颜色字体出现

```
>>> import keyword  
>>> print(keyword.kwlist)
```

False	None	True	and	as	assert	break	class	continue
def	del	elif	else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal	not	or	pass
raise	return	try	while	with	yield			

# 关键字

- while 和 if在IDLE中显示为橙色，它们均是关键字



```
>>> i = 0
>>> while i < 20:
    if i % 2 == 0:
        print(i)
    i = i + 1
```

# 变量名

- 变量命名要见名识义

- num表示人数
- salary表示薪酬
- tax表示税额



```
>>> num = 5
```

```
>>> salary = 3450.7
```

```
>>> tax = salary * 5 * 0.15
```

# 变量的使用

不同于C语言  
不需要显式声明变量

变量类型不需要专门声明

第一次对变量名赋值时自动声明该变量

# 变量的使用

- 变量必须创建和赋值后使用
  - a没有赋值，所以无法直接使用
  - 变量b和c均有赋值
  - 通过赋值号 “=” 右边的表达式的结果确定左边变量的类型。

# 变量的使用



```
>>> a
```

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

a

NameError: name 'a' is not defined

```
>>> b = 3.14
```

```
>>> b
```

```
3.14
```

```
>>> c = 'Circle'
```

```
>>> c
```

```
'Circle'
```


a没有赋值，所以无法直接使用；变量b和c均有赋值；通过赋值号“=”右边的表达式的结果确定左边变量的类型。



# 赋值

- 变量第一次赋值，同时获得类型和“值”

- Python是动态的强类型语言
- 不需要显式声明，根据“值”确定类型
- 以“引用”的方式实现赋值



*S*<sub>ource</sub>

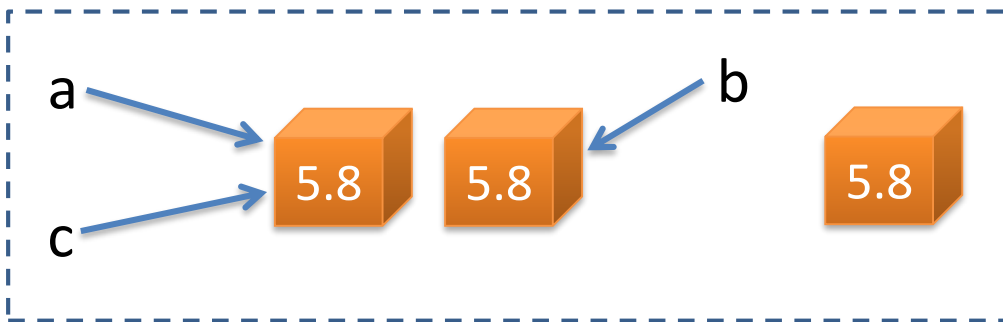
```
>>> # Identifier
>>> PI = 3.14159
>>> pi = 'one word'
>>> print(PI)
3.14159
>>> print(pi)
one word
```

# 变量的管理

## 动态类型

- 引用计数

- $a = c$  , 指向了相同的数据对象
- b和a创建的是不同的5.8对象
- 单独`id(5.8)`是创建的全新的对象



## 动态类型

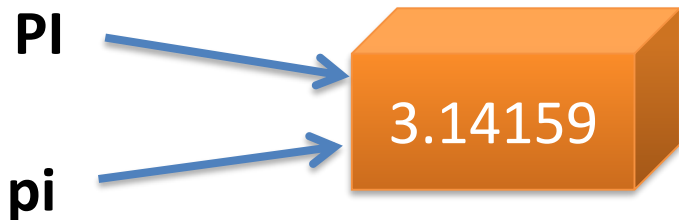



```
>>> a = 5.8
>>> id(a)
1709988157600
>>> b = 5.8
>>> id(b)
1709988157648
```




```
>>> id(5.8)
1709988157696
>>> c = a
>>> id(c)
1709988157600
>>> c = 3
```

# 变量的管理



  
>>> p = 3  
>>> q = 3  
>>> p is q  
True

  
>>> PI = 3.14159  
>>> pi = 3.14159  
>>> PI is pi  
False  
>>> pi = PI  
>>> print(PI)  
3.14159  
>>> pi is PI  
True

图中的形式  
用哪个语句  
可以表示

is运算符的  
基础是id()  
函数

## 2.2.2 表达式和赋值表达式

# 表达式

- 用运算符连接各种类型数据的式子就是表达式

## 算术运算符

乘方	**
正负号	+ -
乘除	* /
整除	//
取余	%
加减	+ -

## 位运算符

取反	~
与	&
或	
异或	^
左移	<<
右移	>>

## 比较运算符

小于	<
大于	>
小于等于	<=
大于等于	>=
等于	==
不等于	!=

## 逻辑运算符

非	not
与	and
或	or

- 运算符有优先级顺序
- 表达式必须有运算结果

**S**ource

```
>>> # expression
```

```
>>> PI = 3.14159
```

```
>>> r = 2
```

```
>>> c_circ = 2 * PI * r
```

```
>>> print("The circle's circum is", c_circ)
```

- $2*PI*r$  是表达式
- 运算结果赋值给变量 `c_circ`

# 赋值 增量赋值

## 增量赋值 操作符

+=	-=	*=	/=	%=	**=
<<=	>>=	&=	^=	=	

- $m /= 5$

即  $m = m / 5$



>>> *# Augmented assignment*

>>>  $m = 18$

>>>  $m /= 5$

>>>  $m$

3.6



# 赋值 链式赋值

•  $b = a = a + 1$  相当于

如下2条语句：

```
>>> a = a + 1
```

```
>>> b = a
```



```
>>> # Chained assignment
```

```
>>> a = 1
```

```
>>> b = a = a + 1
```

```
>>> b
```

```
2
```

```
>>> a
```

```
2
```

## 赋值 多重赋值

- 等号左右两边都以元组的方式出现
- 相当于：  
>>> (PI, r) = (3.14159, 3)



```
>>> # Multiple assignment
>>> PI, r = 3.1415, 3
>>> PI
3.1415
>>> r
3
>>> temp = 3.1415, 3
>>> PI, r = temp
```

- 完整执行一个任务的一行逻辑代码
  - 赋值语句完成了赋值
  - `print()`函数调用语句完成了输出



```
>>> # statement  
>>> PI = 3.14159  
>>> print(PI)
```

# 语句和表达式

## 语句

**完成一个任务**

如，打印一份文件



## 表达式

**任务中的一个具体组成部分**

如，这份文件的  
具体内容



## 2.3

# PYTHON 数据类型

- 必须有明确的数据类型，程序才能分配给常量、变量精确的存储大小，才能进行精确或高效率的运算

1	0	0	1	0	0	1	1
1	0	0	1	0	0	1	1

+

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---



1	0	0	1	0	0	1	1
---	---	---	---	---	---	---	---

+

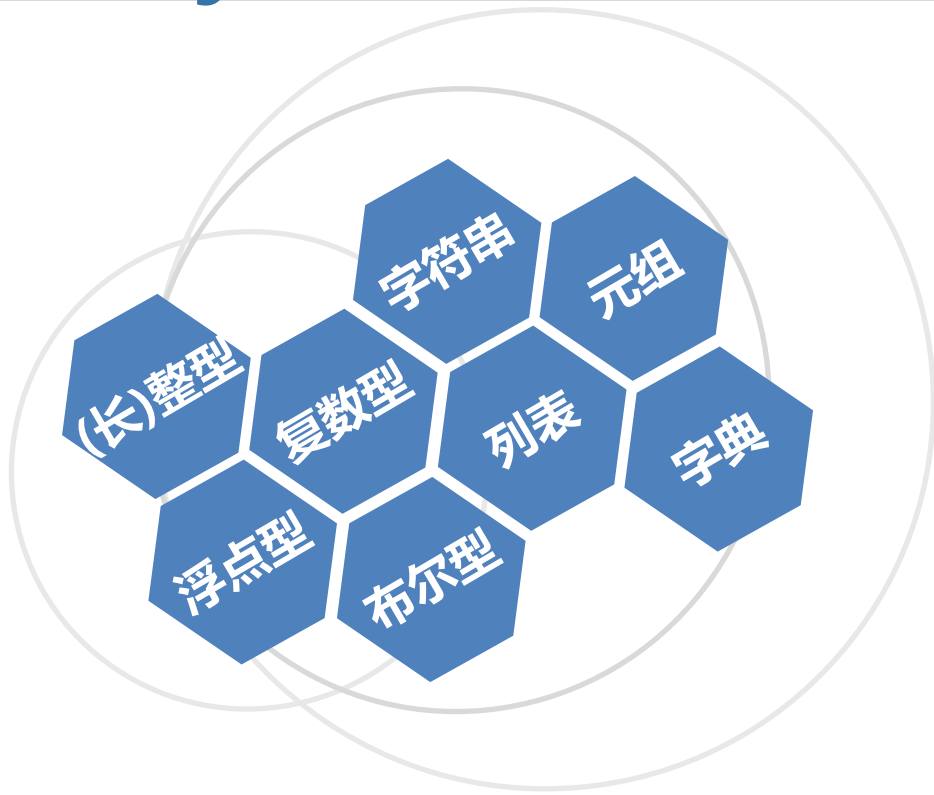
0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

1 0 0 1 1 0 1 1



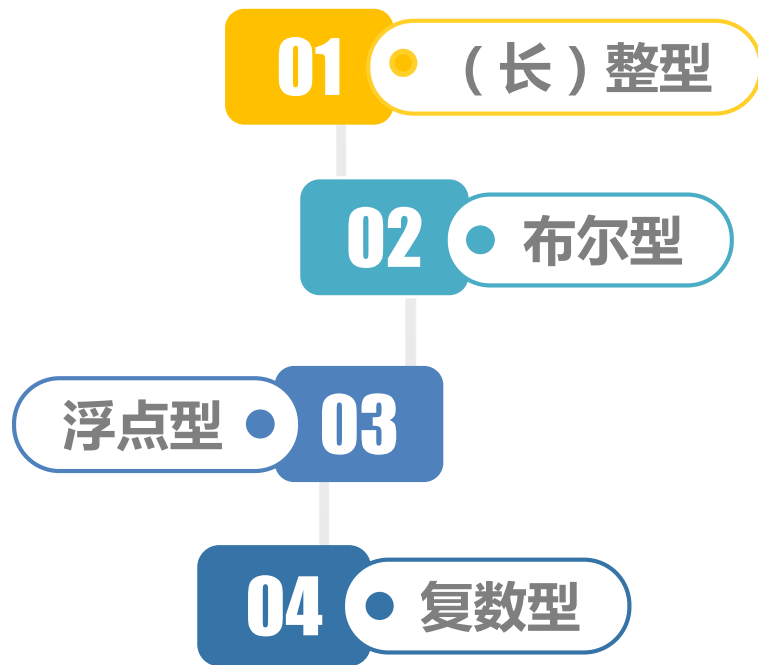
# Python数据类型

39



## 2.3.1 基本类型






- 整型和长整型并不严格区分
- Python 2支持整型值后加  
“L” 即为长整型



```
>>> # integer  
>>> type(3)  
<class 'int'>
```

# 布尔型

- 整型的子类
- 仅有2个值：True、False
- 本质上是用整型的1、0分别存储的

 `>>> # boolean`  
`>>> x = True`  
`>>> type(x)`  
`<class 'bool'>`  
`>>> int(x)`  
`1`  
`>>> y = False`  
`>>> int(y)`  
`0`

- 即数学中的实数
- 可以类似科学计数法表示



```
>>> # float
```

```
>>> 3.22
```

```
3.22
```

```
>>> 9.8e3
```

```
9800.0
```

```
>>> -4.78e-2
```


```
-0.0478
```


```
>>> type(-4.78e-2)
```

```
<class 'float'>
```

# 复数型

- $j = \sqrt{-1}$  , 则  $j$  是虚数
- 实数+虚数 就是复数
- 虚数部分必须有 $j$

  
>>> *# complex*  
>>> 2.4+5.6j  
(2.4+5.6j)  
>>> type(2.4+5.6j)  
<class 'complex'>

  
>>> *# complex*  
>>> 3j  
3j  
>>> type(3j)  
<class 'complex'>  
>>> 5+0j  
(5+0j)  
>>> type(5+0j)  
<class 'complex'>

- 复数可以分离实数部分和虚数部分
  - 复数.real
  - 复数.imag
- 复数的共轭
  - 复数.conjugate()



```
>>> # complex
>>> x = 2.4+5.6j
>>> x.imag
5.6
>>> x.real
2.4
>>> x.conjugate()
(2.4-5.6j)
```

## 2.3.2 序列类型

# 序列类型

序列类型是一种容器  
通过索引访问成员

01

● 字符串

单引号、双引号、三引号内的都是  
字符串，不可变类型

列表 ●

02

强大的类型，用方括号 [] 界别，  
可变类型

03

● 元组

与列表相似，用小括号 () 界  
别，不可变类型

range对象 ●

04

用range()函数生成一个不可  
变的数字序列，不可变类型



# 字符串的表示

- 单引号
- 双引号
- 三引号



```
>>> myString = 'Hello World!'
```

```
>>> print(myString)
```

```
Hello World!
```

```
>>> myString = "Hello World!"
```

```
>>> print(myString)
```

```
Hello World!
```

```
>>> myString = """Hello World!"""
```

```
>>> print(myString)
```

```
Hello World!
```

# 字符串中字符的访问

- 利用索引值访问单个字符外
- 利用切片操作进行多个字符的访问



```
>>> myString = 'Hello World!'
```

```
>>> myString1[1]
```

```
'e'
```

```
>>> myString1[1:4]
```

```
'ell'
```

- 可以存储不同类型的数据对象
- 列表中的元素值是可变的



```
>>> aList = [1, 'Maths', 88]
>>> aList
[1, 'Maths', 88]
>>> aList[2] = 90
>>> aList
[1, 'Maths', 90]
```

- 可以存储不同类型的数据对象
- 元组中的元素值是**不可**变的



```
>>> aTuple = (1, 'Maths', 88)
```

```
>>> aTuple  
(1, 'Maths', 88)
```

```
>>> aTuple[2] = 90
```


Traceback (most recent call last):

```
File "<pyshell#2>", line 1, in <module>  
    aTuple[2] = 90
```

TypeError: 'tuple' object does not support item assignment

# range对象

- 使用range()函数生成一个不可变的数字序列
- range()函数常常用在for循环中



```
>>> list(range(1, 11))  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
>>> for i in range(1, 5):  
        print(i ** 2)  
  
1  
4  
9  
16
```

## 2.3.3 字典

# 映射类型 字典

- 用大括号 {} 界别
- 字典每一个元素键值对
  - 不可变对象作为键
  - 值可以是任意类型

字典类型是一种容器  
通过键访问值



```
>>> # dictionary
>>> d={'sine':'sin','cosine':'cos','PI':3.14159}
>>> d['sine']
'sin'
```

## 2.4

---

# PYTHON

## 基本运算





## 2.4.1 算术运算

- 算术运算符的优先级： $** > + - (\text{正负号}) > * / > // > \% > + -$

S  
source

```
>>> # arithmetic
```

```
>>> x = 1
```

```
>>> y = 2
```

```
>>> z = 3
```

```
>>> result1 = x + 3/y - z % 2
```

```
>>> result2 = (x + y**z*4)//5
```

```
>>> print(circum, result1, result2)
```

```
18.84954 1.5 6
```

S  
source

```
>>> # arithmetic
```

```
>>> pi = 3.14159
```

```
>>> r = 3
```

```
>>> circum = 2 * pi * r
```

# 算术运算中的除法

- 除法有2种运算符
  - `"/"` 和 `"//"`



```
>>> # arithmetic
```

```
>>> 3 / 4
```

```
0.75
```

```
>>> 4 / 2
```

```
2.0
```

```
>>> 5 // 2
```

```
2
```

```
>>> 3 // 4
```

```
0
```

```
>>> -6 // 4
```

```
-2
```

# 位运算

- 只适用于整数，位运算就是按整数的二进制位进行的运算



```
>>> ~1
```

```
-2
```

```
>>> 16 << 2
```

```
64
```

```
>>> 16 >> 2
```

```
4
```

```
>>> 64 & 15
```

```
0
```

```
>>> 64 | 15
```

```
79
```

```
>>> 64 ^ 14
```

```
78
```

# 比较运算

- 数值的比较：按值比大小
- 字符串的比较：按ASCII码值大小



```
>>> # compare  
>>> 2 == 2  
True  
>>> 2.46 <= 8.33  
True  
>>> 'abc' == 'xyz'  
False  
>>> 'abc' > 'xyz'  
False  
>>> 'abc' < 'xyz'  
True
```



```
>>> # compare
```

```
>>> 3 < 4 < 7 # same as ( 3 < 4 ) and ( 4 < 7 )
```

```
True
```

```
>>> 4 > 3 == 3 # same as ( 4 > 3 ) and ( 3 == 3 )
```

```
True
```

```
>>> 4 < 3 < 5 != 2 < 7
```

```
False
```

- 逻辑运算符优先级：

- not、and、or



```
>>> # logical
```

```
>>> x, y = 3.1415926536, -1024
```

```
>>> x < 5.0
```

```
True
```

```
>>> not (x < 5.0)
```

```
False
```

```
>>> not (x is y)
```

```
True
```



- 逻辑运算符优先级：

- not、and、or



```
>>> # logical
```

```
>>> x, y = 3.1415926536, -1024
```

```
>>> (x < 5.0) or (y > 2.718281828)
```

```
True
```

```
>>> (x < 5.0) and (y > 2.718281828)
```

```
False
```

```
>>> 3 < 4 < 7
```

```
True
```

## 2.4.5 优先级

# Python运算符

- 算术运算符 > 位运算符 > 比较运算符 > 逻辑运算符

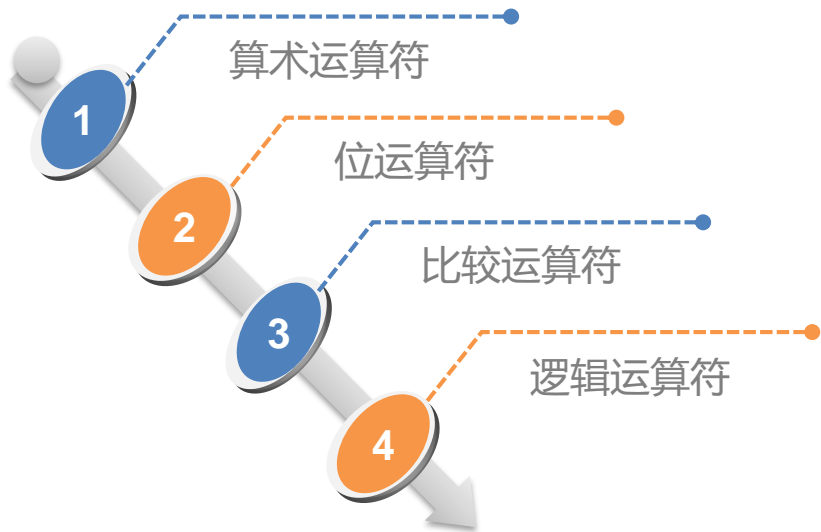
- 算术运算符的优先级



– \*\* > + - (正负号) > \* / > // > % > + -

- 逻辑运算符的优先级

– not > and > or



```
>>> # mix
```

```
>>> 3 < 2 and 2 < 1 or 5 > 4
```

```
True
```

```
>>> x + 3/y -z % 2 > 2
```

```
False
```

```
>>> 3-2 << 1
```

```
2
```

```
>>> 3-2 << 1 < 3
```

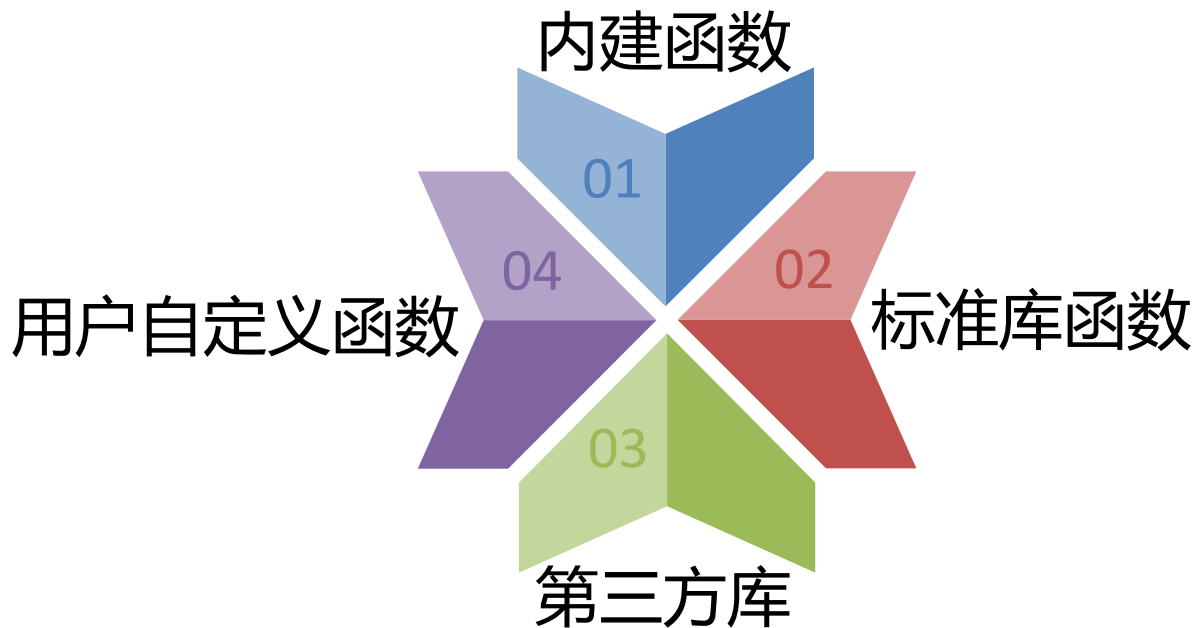
```
True
```

## 2.5

# PYTHON的 函数、模块和包

## 2.5.1 函数

# Python中的函数



- 函数可以看成类似于数学中的函数
- 完成一个特定功能的一段代码
  - 绝对值函数`abs(x)`
  - 类型函数`type(x)`
  - 四舍五入函数`round(x)`



```
>>> dir(__builtins__)
```

## 内建函数

73

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	<u>input()</u>	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

- 内建函数
  - `str()` 和 `type()`等适用于所有标准类型

## 数值型内建函数

<code>abs()</code>	<code>bool()</code>	<code>oct()</code>
<code>round()</code>	<code>int()</code>	<code>hex()</code>
<code>divmod()</code>	<code>ord()</code>	<code>pow()</code>
<code>float()</code>	<code>chr()</code>	<code>complex()</code>

## 实用函数

<code>dir()</code>	<code>input()</code>
<code>help()</code>	<code>open()</code>
<code>len()</code>	<code>range()</code>

S<sub>ource</sub>

```
>>> # round-off int
```

```
>>> int(35.4)
```

```
35
```

```
>>> int(35.5)
```

```
35
```

```
>>> int(35.8)
```

```
35
```

```
>>> type(int(35.8))
```

```
<class 'int'>
```

S<sub>ource</sub>

```
>>> # ord
```

```
>>> ord('3')
```

```
51
```

```
>>> ord('a')
```

```
97
```

```
>>> ord('\n')
```

```
10
```

```
>>> type(ord('A'))
```

```
<class 'int'>
```

# 其他方式定义的函数

A

**标准库函数**：需要先导入模块再使用函数，每个库有相关的一些函数如math库中的sqrt()函数

B

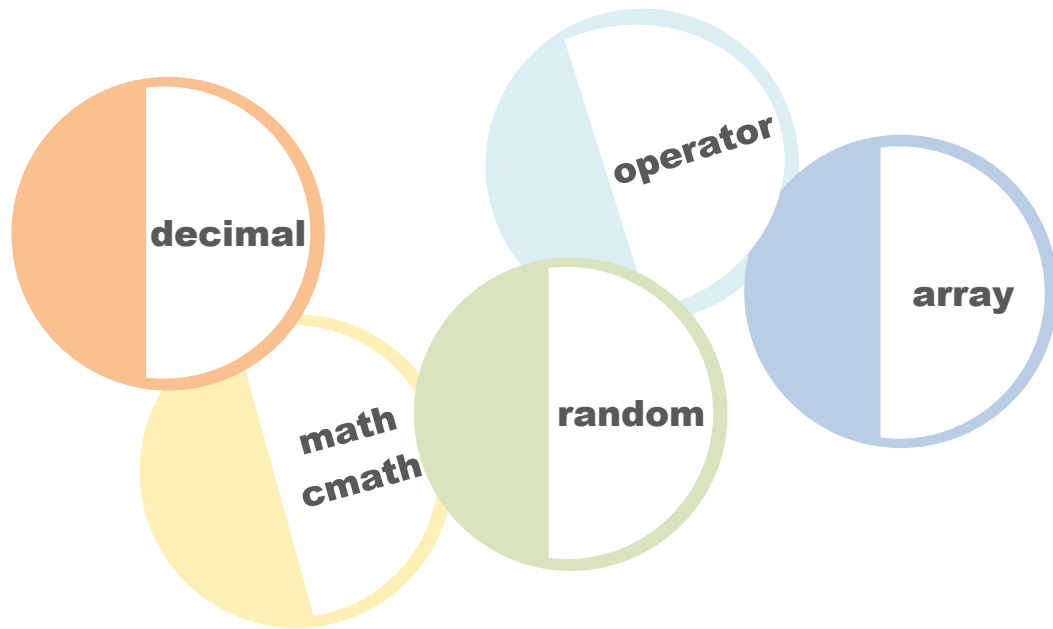
**第三方库函数**：数量非常惊人，这也是Python重要的特征和优势，例如著名的科学计算包SciPy中就包含了很多用于科学计算的函数

C

**用户自定义函数**：有固定的定义、调用和参数传递方式等

# 库 ( library )

- 库是一组具有相关功能的模块的集合
- Python的一大特色就是具有强大的标准库、以及第三方库、以及自定义模块



数值型相关标准库

## 2.5.2 模块

- 非内建函数如何使用？

Source

```
>>> # round-off floor
```

```
>>> floor(5.4)
```

```
Traceback (most recent call last):
```

```
File "<pyshell#0>", line 1, in <module>
```

```
    floor(5.4)
```

```
NameError: name 'floor' is not defined
```

Source

```
>>> # round-off floor
```

```
>>> from math import *
```

```
>>> floor(-35.4)
```

```
-36
```

```
>>> floor(-35.5)
```

```
-36
```

```
>>> floor(-35.8)
```

```
-36
```

## 模块（二）



```
>>> import math
>>> math.pi
3.141592653589793
```

- 一个完整的Python文件即是一个模块
  - 文件：物理上的组织方式 `math.py`
  - 模块：逻辑上的组织方式 `math`
- Python通常用 “`import 模块`” 的方式将现成模块中的函数、类等重用到其他代码块中
  - `math.pi`的值可以直接使用，不需要自行定义



- 导入多个模块
- 模块里导入指定的模块属性，  
也就是把指定名称导入到当前  
作用域

```
>>>import ModuleName
```

```
>>>import ModuleName1, ModuleName2, ...
```

```
>>>from Module1 import ModuleElement
```



```
>>> import math
```

```
>>> math.log(math.e)
```

```
1.0
```

```
>>> math.sqrt(9)
```

```
3.0
```

```
>>> from math import floor
```

```
>>> floor(5.4)
```

```
5
```

## 2.5.3 包

## 包 ( package )

- 一个有层次的文件目录结构
- 定义了一个由模块和子包组成的 Python 应用程序执行环境

```
>>> import AAA.CCC.c1  
>>> AAA.CCC.c1.func1(123)
```

```
>>> from AAA.CCC.c1 import func1  
>>> func1(123)
```

```
AAA/  
    __init__.py  
    bbb.py  
    CCC/  
        __init__.py  
        c1.py  
        c2.py  
    DDD/  
        __init__.py  
        d1.py  
    EEE/  
    ...
```

- **Python程序基本构成与风格**
- **Python语法基础**
- **Python数据类型**
- **Python基本运算**
- **Python中的模块和函数**

