



Chap2 Python Basic

第2章 Python基础

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

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

File

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

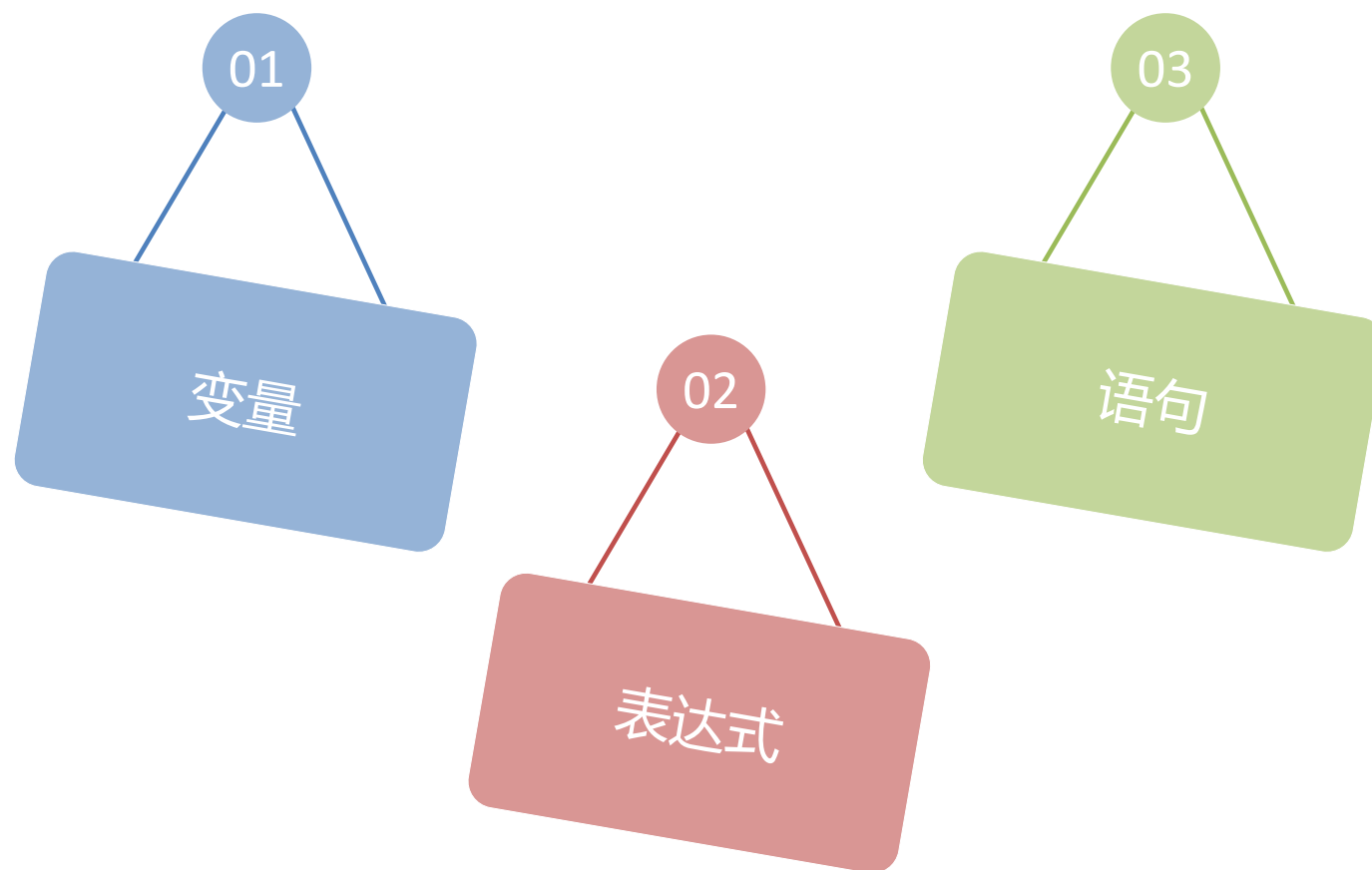
第3行

第4行

第5行

第6行

程序基本要素



Python输入: input()函数

6

 Source

```
>>> 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!
```

注释



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


注释多行



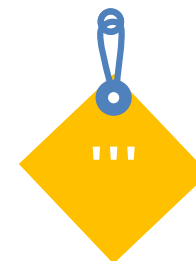
```
>>> '''
```

```
for number in num:
```

```
    prog = prog * number
```

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

```
'''
```



2.1.2 PYTHON程序设计风格

Python 风格 (一)

11

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

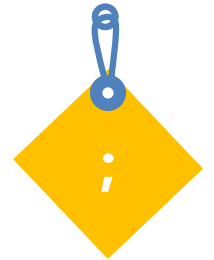
print('The prog is: ', prog)

Python 风格 (二)

13



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



一行多语句



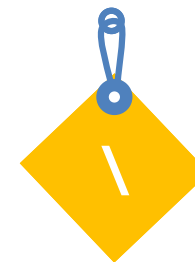
```
>>> x = 'Today'  
>>> y = 'is'  
>>> z = 'Thursday'  
>>> print(x, y, z)  
Today is Thursday
```

续行



```
>>> # long sentence
```

```
>>> if signal == 'red' and car == 'moving':  
    car = 'stop'  
    elif signal == 'green' and car == 'stop':  
        car = 'moving'
```



Python 风格 (三)

15

续行



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

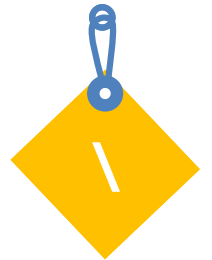


续行

- 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 luck!")
```



2.2

PYTHON 语法基础

2.2.1 变量



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

- Python是面向对象编程语言
 - 实例、函数、方法、类都是对象
 - 唯一的身份标识，由id()函数得到
 - 对象的三个属性：身份、类型、值
 - type()查看对象类型

OOP(Object-oriented programming)

标识符

- 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用作一般的变量名。

- 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
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', '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']
```

Python 3.8

关键字

- while 和 if在IDLE中显示为橙色，它们均是关键字



```
# prog2-2.py
```

```
i = 0
```

```
while i < 20:
```

```
    if i % 2 == 0:
```

```
        print(i)
```

```
    i = i + 1
```


函数名

- 尽量避免使用函数名做变量名
- 函数名如果定义为变量名, 则该函数会被改写, 失去原来函数的功能。



```
>>> str = "hi"
```

```
>>> str
```

```
'hi'
```

```
>>> str(123)
```

Traceback (most recent call last):

File "<pyshell#3>", line 1, in <module>

str(123)

TypeError: 'str' object is not callable

变量名

- 变量命名要见名识义
- 命名方式
 - 单个变量/单词/单词缩写
 - stuName驼峰式
 - stu_name下划线式



```
>>> nums = 5
>>> salary = 3450.7
>>> tax = salary * nums * 0.15
>>> class_no = 1
```

变量的使用

Python属于动态类型语言，对象在运行时绑定类型

变量赋值后使用

不需要显示声明变量

第一次对变量名赋值时由值自动确定变量类型

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'
```

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

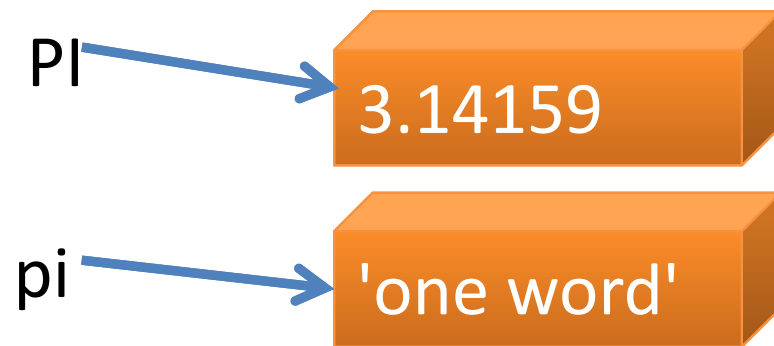


```
>>> type(b)
<class 'float'>
>>> type(c)
<class 'str'>
3.14
```

使用type()函数查看
变量b和c的类型

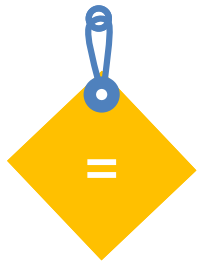
变量的使用

30



S_{ource}

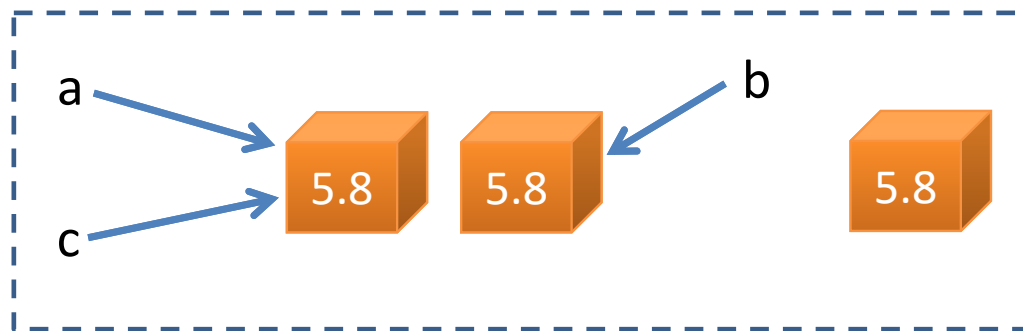
```
>>> # Identifier
>>> PI = 3.14159
>>> pi = 'one word'
>>> print(PI)
3.14159
>>> print(pi)
one word
```



动态类型

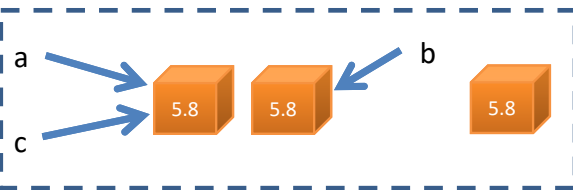
- 引用计数

- 每个对象在内存中申请开辟一块空间保存
- 该对象在内存中的地址称为“引用”



a和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

引用计数



S_{ource}

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

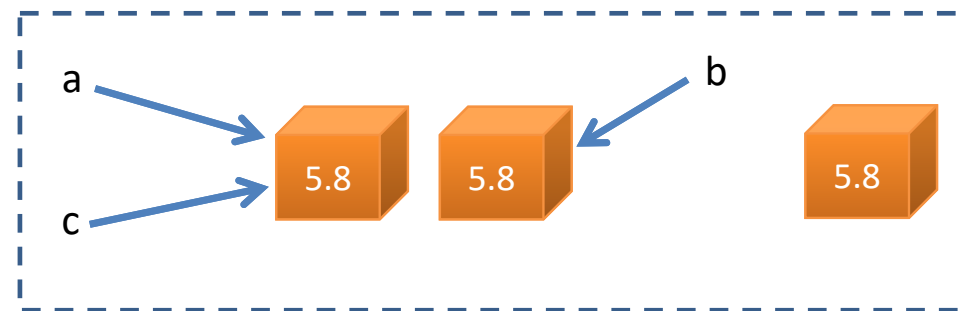
S_{ource}

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


引用计数

- 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



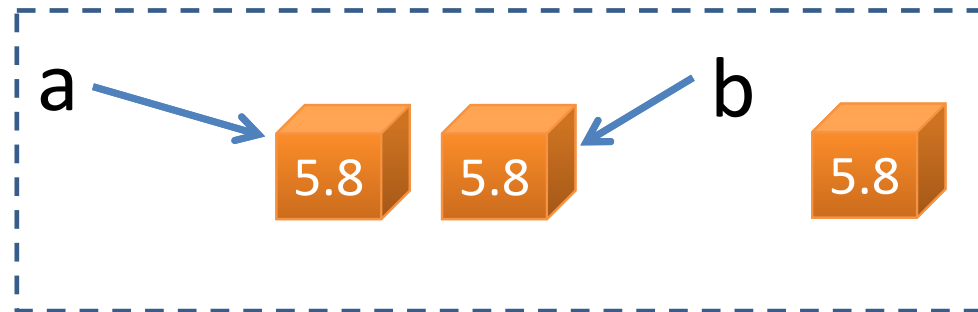
引用计数

S_{ource}

```
>>> c = 567.8
```

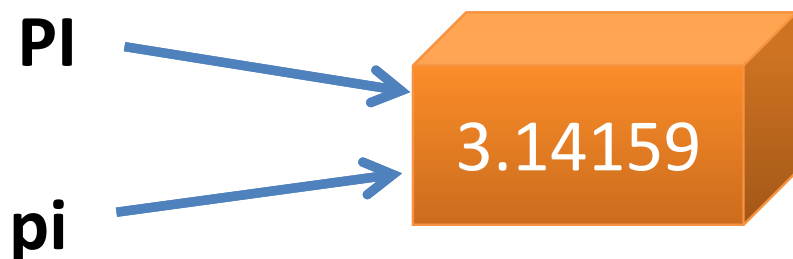
```
>>> id(c)
```

```
1709988157696
```



变量的管理

35



Source

```
>>> 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

变量的管理

Source

```
>>> x = 257
>>> y = 257
>>> id(x)
2389205691344
>>> id(y)
2389205691152
```

Source

```
>>> x = 3
>>> y = 3
>>> id(x)
1693606176
>>> id(y)
1693606176
>>> z = x
>>> id(z)
1693606176
```

小整数的默认范围：
[-5, 256]
模块内有特殊处理

2.2.2 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 | >> |

关系运算符

| | |
|------|----|
| 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

Source

```
>>> # 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`

- 赋值表达式 (海象表达式)

```
>>> (score := 78) >= 80
```

```
False
```

Python 3.8
开始支持

2.2.3 语句和赋值语句

- 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 * 2 * 2)
```

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
```

语句和表达式

语句

完成一个独立任务

如，打印一份文件



表达式

**任务中的一个具体
组成部分或语句**

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，程序才能给对象分配存储空间，从而进行运算

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 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 |
|---|---|---|---|---|---|---|---|

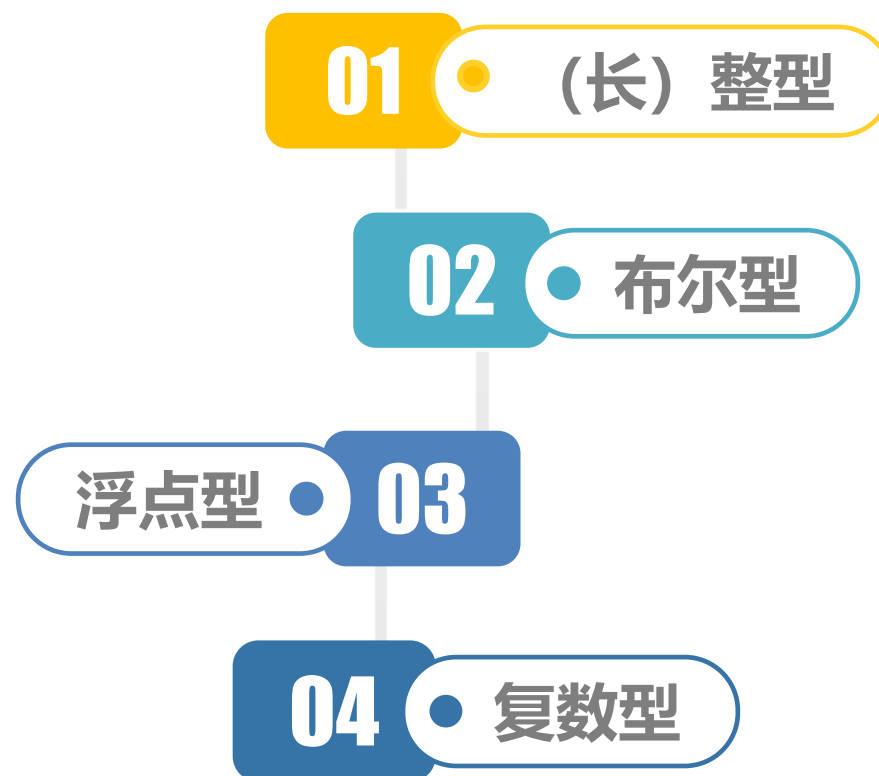


Python数据类型


49



2.3.1 基本类型




- 整型和长整型并不严格区分



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

- 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'>
```

- $j=\sqrt{-1}$, 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'>

2.3.2 序列类型

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!
```

2.4

PYTHON 基本运算

Python基本运算

59



2.4.1 算术运算

算术运算

- 算术运算符的优先级:

(1)**

(2)+ - (正负号)

(3)*、 /、 //、 %

(4)+ -



```
>>> # 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
```

- Python可以处理很大的数




```
>>> # arithmetic
```

```
>>> 3333333333333333333333333333 * 123456789  
41152262999999999999999958847737
```


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

算术运算中的取余

- $x \% y$
 - x整除y的余数
 - x和y可以是浮点数

 *Source*

```
>>> # arithmetic
>>> 5 // 2
2
>>> 5 % 2
1
>>> 4 // 2
2
>>> 4 % 2
0
>>> 7.5 // 2.0
3.0
>>> 7.5 % 2.0
1.5
>>> -7.5 // 2.0
-4.0
>>> -7.5 % 2.0
0.5
```


2.4.2 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

| | |
|----|----|
| 取反 | ~ |
| 与 | & |
| 或 | |
| 异或 | ^ |
| 左移 | << |
| 右移 | >> |

Source

>>> ~1

-2

>>> 16 << 2

64

>>> 16 >> 2

4

>>> 65 & 15

1

>>> 65 | 15

79


>>> 65 ^ 15

78

2.4.3 关系运算

- 数值的比较：按值比大小
- 字符串的比较：按ASCII码值大小

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

 *Source*

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

关系运算

69



```
>>> # 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
```

2.4.4 逻辑运算

- 逻辑运算符优先级:

(1)not

(2)and

(3)or



```
>>> # logical
```

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

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

```
True
```

```
>>> not x < 5.0
```

```
False
```

```
>>> not x is y
```

```
True
```

A speech bubble icon containing the word "Source" in orange text.

```
>>> # 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

- 算术运算符 > 位运算符 > 关系运算符 > 逻辑运算符

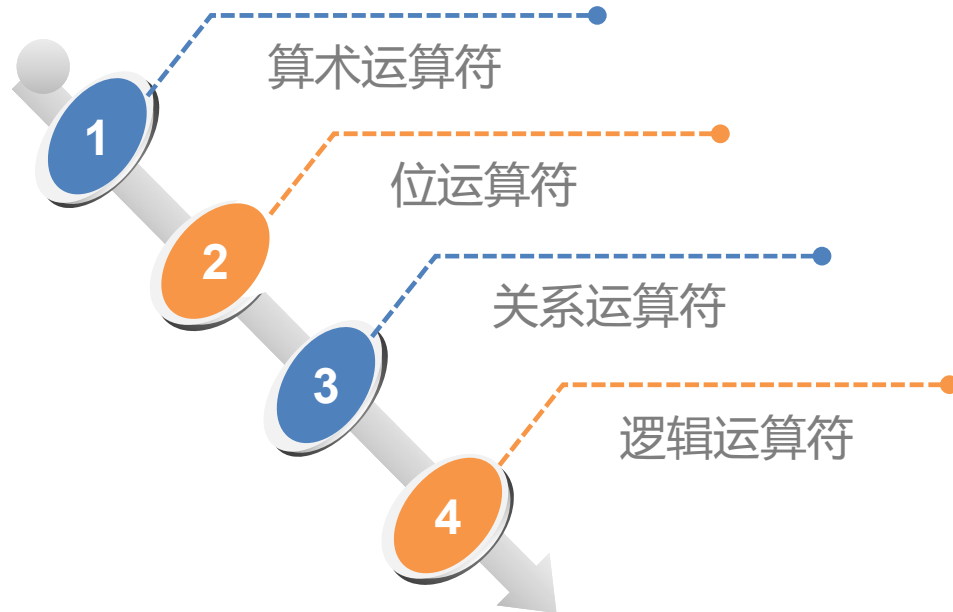
- 算术运算符的优先级



– ** > + - (正负号) > * / // % > + -

- 逻辑运算符的优先级

– not > and > or



S_{ource}

```
>>> # mix
>>> 3 < 2 and 2 < 1 or 5 > 4
True
>>> x, y, z = 1, 2, 3
>>> 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

78



- 函数可以看成类似于数学中的函数
- 实现某个功能并获得某种反馈的一段代码
 - 绝对值函数`abs(x)`
 - 类型函数`type(x)`
 - “四舍五入”函数`round(x)`

```
>>> dir(__builtins__)  
或>>> dir(builtins) # 先导入包
```

内建函数

80

| 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()/hex()</code> |
| <code>round()</code> | <code>int()</code> | <code>float()</code> |
| <code>divmod()</code> | <code>pow()</code> | <code>ceil()</code> |
| <code>floor()</code> | <code>ord()</code> | <code>chr()</code> |

实用函数

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

内建函数

82

Source

```
>>> abs(-4.73)
4.73
>>> int(4.5)
4
>>> int('123')
123
>>> int('123', 8)
83
>>> oct(83)
'0o123'
```

Source

```
>>> round(5.79)
6
>>> round(3.22)
3
>>> round(3.5)
4
>>> round(4.5)
4
```



```
>>> type(3)
```

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

```
>>> type([3, 2, '111'])
```

```
<class 'list'>
```

```
>>> help(abs)
```

Help on built-in function abs in module builtins:

```
abs(x, /)
```

Return the absolute value of the argument

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

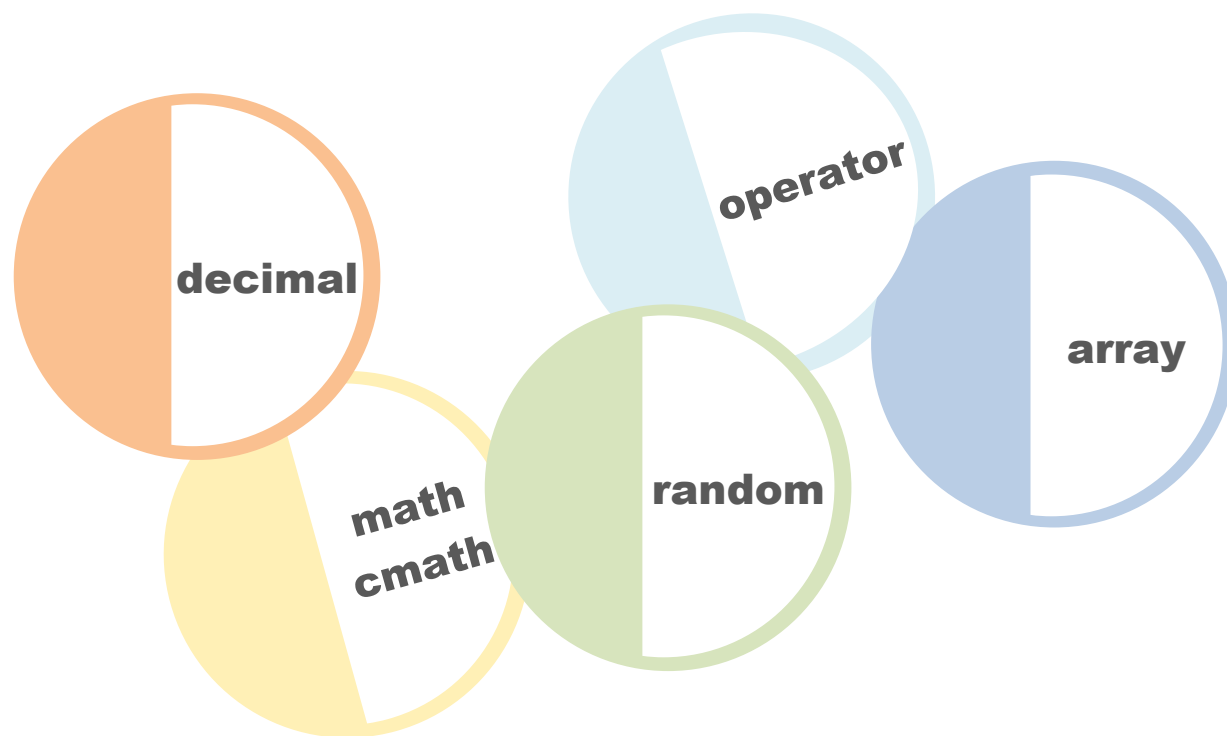
B

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?

math模块就是包含函数定义的math.py文件

Source

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

```
>>> floor(3.5)
```

Traceback (most recent call last):

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

floor(3.5)

NameError: name 'floor' is not defined

Source

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

```
>>> import math
```

```
>>> math.floor(3.5)
```

```
3
```

```
>>> math.floor(-4.3)
```

```
-5
```



- 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`


```
>>> import math
```

```
>>> math.pi
```

```
3.141592653589793
```

- 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 math, os, operator`
`>>> 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 A.C.c1  
>>> A.C.c1.foo(123)
```

```
>>> from A.C.c1 import foo  
>>> foo(123)
```

```
A/  
  __init__.py  
  b.py  
  C/  
    __init__.py  
    c1.py  
    c2.py  
  D/  
    __init__.py  
    d1.py  
    d2.py  
  ...
```

- 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

