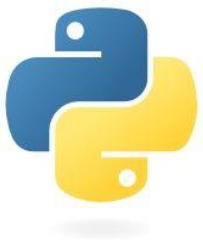
潍坊科技学院
Weifang University of Science and Technology

第2章 Python基础

授课老师：刘国旭

潍坊科技学院



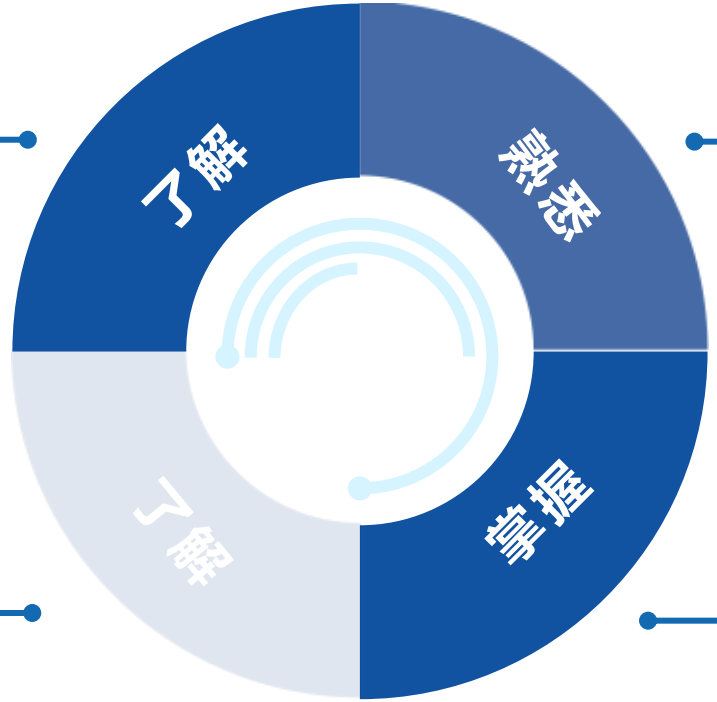
python™

- 代码格式
- 标识符和关键字
- 变量和数据类型
- 数字类型
- 运算符

学习目标

1 了解 代码格式

2 熟悉 标识符、关键字、运算符



3 掌握 变量和数据类型

4 了解 数字类型，数字类型转换

目录页

目录页

目录页

2.1 代码格式

代码格式可提升代码的可读性，与其他语言不同，Python代码的格式是Python语法的组成之一，不符合格式规范的Python代码无法正常运行。

2.1.1 注释

单行注释以“#”开头，用于说明当前行或之后代码的功能。单行注释既可以单独占一行，也可以位于标识的代码之后，与标识的代码共占一行。

```
# 第一个注释
print（“Hello, Python!”）# 第二个注释
```

2.1.1 注释

多行注释是由三对双引号或单引号包裹的语句，主要用于说明函数或类的功能。

```
"""
print(value, ..., sep=' ', end='\n',file=sys.stdout,
flush=False)
"""
```

2.1.2 缩进

Python代码的缩进可以通过Tab键控制，也可使用空格控制。空格是Python3首选的缩进方法，一般使用4个表示一级缩进；Python3不允许混合使用Tab和空格。

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

```
if True:
    print ("True")
else:
    print ("False")
  print ("hello")
```

2.1.3 语句换行

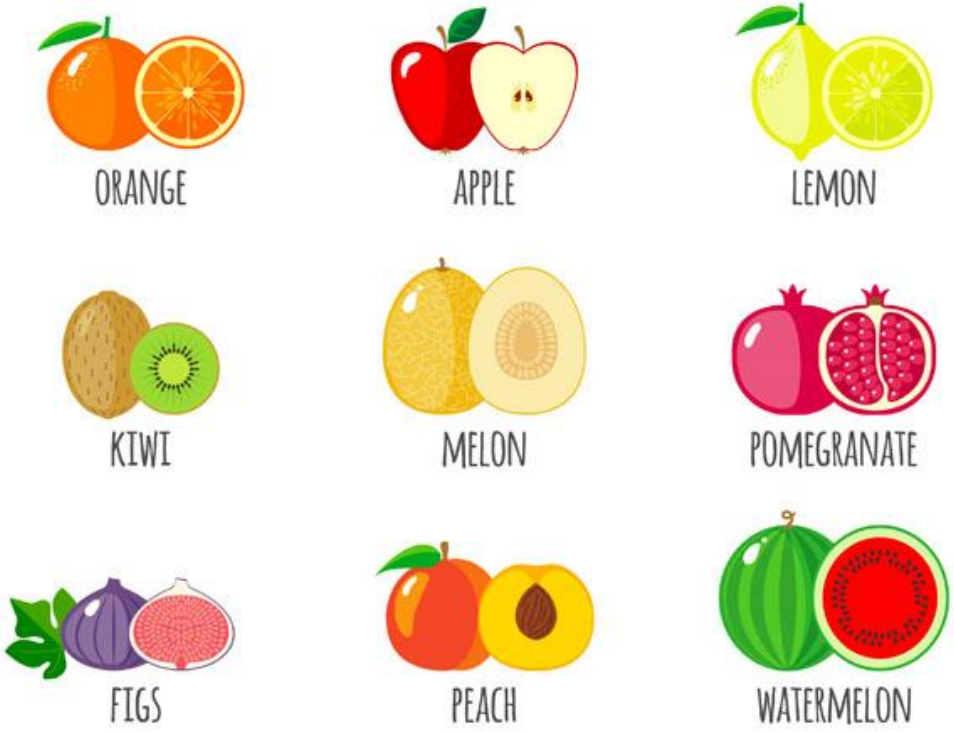
Python官方建议每行代码不超过79个字符，若代码过长应该换行。Python会将圆括号、中括号和大括号中的行进行隐式连接，我们可以根据这个特点实现过长语句的换行显示。

```
string = ("Python是一种面向对象、解释型计算机程序设计语言，"
          "由Guido van Rossum于1989年底发明。"
          "第一个公开发行版发行于1991年，"
          "Python源代码同样遵循GPL(GNU General Public License)协议。")
```

目录页

2.2.1 标识符

现实生活中，人们常用一些名称来标记事物。例如，每种水果都有一个名称来标识。



2.2.1 标识符

若希望在程序中表示一些事物，开发人员需要自定义一些符号和名称，这些符号和名称叫做标识符。 Python中的标识符需要遵守一定的规则。

命名规则

- 标示符由字母、下划线和数字组成，且数字不能开头。
- Python中的标识符是区分大小写的。例如，andy和Andy是不同的标识符。
- Python中的标识符不能使用关键字 。

2.2.1 标识符

为了规范命名标识符，关于标识符的命名提以下建议：

- 见名知意。
- 常量名使用大写的单个单词或由下画线连接的多个单词。
- 模块名、函数名使用小写的单个单词或由下画线连接的多个单词。
- 类名使用大写字母开头的单个或多个单词。

2.2.2 关键字

关键字是Python已经使用的、不允许开发人员重复定义的标识符。Python3中一共有35个关键字，每个关键字都有不同的作用。

关键字

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

2.2.2 关键字

在Jupyter单元格中执行“help(”关键字“)”可查看关键字的声明。

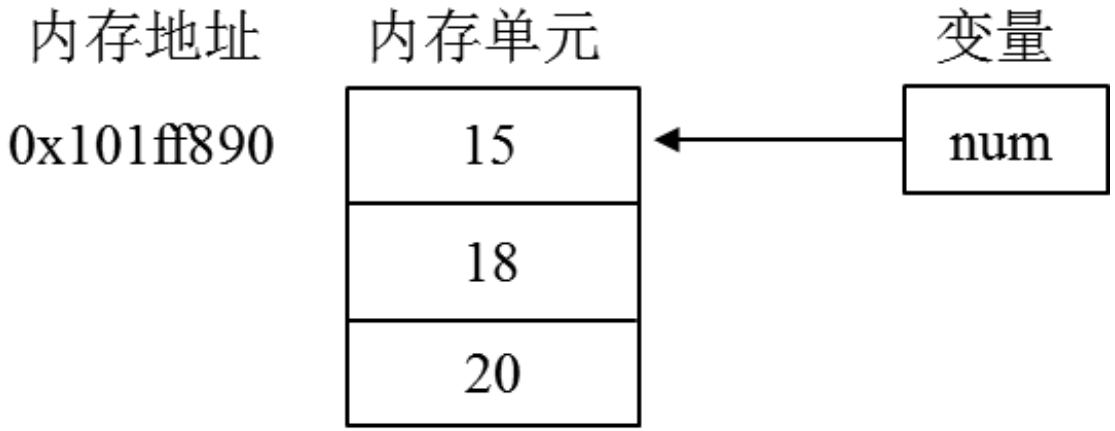
```
In [1]: help("import")
```

```
The "import" statement
**********************

   import_stmt     ::= "import" module ["as" identifier] ("," module ["as" identifier])*
                     | "from" relative_module "import" identifier ["as" identifier]
                     ("," identifier ["as" identifier])*
                     | "from" relative_module "import" "(" identifier ["as" identifier]
                     ("," identifier ["as" identifier])* [","] ")"
                     | "from" module "import" "*"
   module          ::= (identifier ".")* identifier
   relative_module ::= "."* module | "."+

The basic import statement (no "from" clause) is executed in two
steps:

1. find a module, loading and initializing it if necessary

2. define a name or names in the local namespace for the scope
   where the "import" statement occurs.
```

目录页

2.3.1 变量

程序在运行期间用到的数据会被保存在计算机的内存单元中，为了方便存取内存单元中的数据，Python使用标识符来标识不同的内存单元，如此，标识符与数据建立了联系。



2.3.1 变量

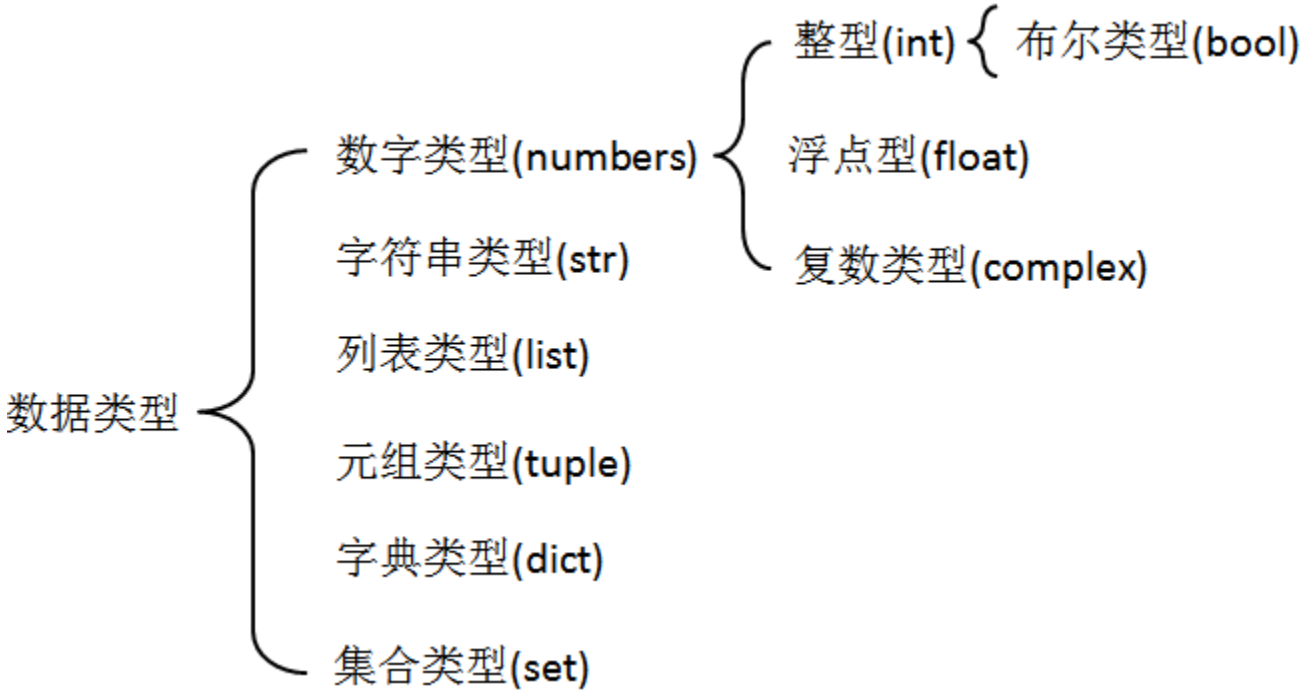
标识内存单元的标识符又称为变量名，Python通过赋值运算符“=”将内存单元中存储的数值与变量名建立联系，即定义变量，具体语法格式如下：变量 = 值

将内存单元中存储的数据100与变量名data建立联系

data = 100

2.3.2 数据类型

根据数据存储形式的不同，数据类型分为基础的数字类型和比较复杂的组合类型，其中数字类型又分为整型、浮点型、布尔类型和复数类型；组合类型分为字符串、列表、元组、字典等。



2.3.2 数据类型

Python内置的数字类型有整型（int）、浮点型（float）、复数类型（complex）和布尔类型(bool)，其中int、float和complex分别对应数学中的整数、小数和复数；bool类型比较特殊，它是int的子类，只有True和False两种取值。

数字类型示例

- 整型： 0　101　-239　False　True
- 浮点型： 3.1415　4.2E-10　-2.334E-9
- 复数类型： 3.12+1.2.3j　-1.23-93j
- 布尔类型： True　False

2.3.2 数据类型

字符串是一个由单引号、双引号或者三引号包裹的、有序的字符集合。

- 使用单引号包含： 'Python123￥'
- 使用双引号包含： "Python4*&%"
- 使用三引号包含： '''Python s1 ~(())'''

字符串示例

2.3.2 数据类型

列表是多个元素的集合，它可以保存任意数量、任意类型的元素，且可以被修改。Python中使用“[]”创建列表，列表中的元素以逗号分隔，示例如下：

- [1, 2, 'hello']

2.3.2 数据类型

元组与列表的作用相似，它可以保存任意数量与类型的元素，但不可以被修改。Python中使用“()”创建元组，元组中的元素以逗号分隔，示例如下：

- (1, 2, 'hello'])

2.3.2数据类型

集合与列表和元组类似，也可以保存任意数量、任意类型的元素，不同的是，集合使用“{}”创建，集合中的元素无序且唯一。示例如下：

- {'apple', 'orange', 1}

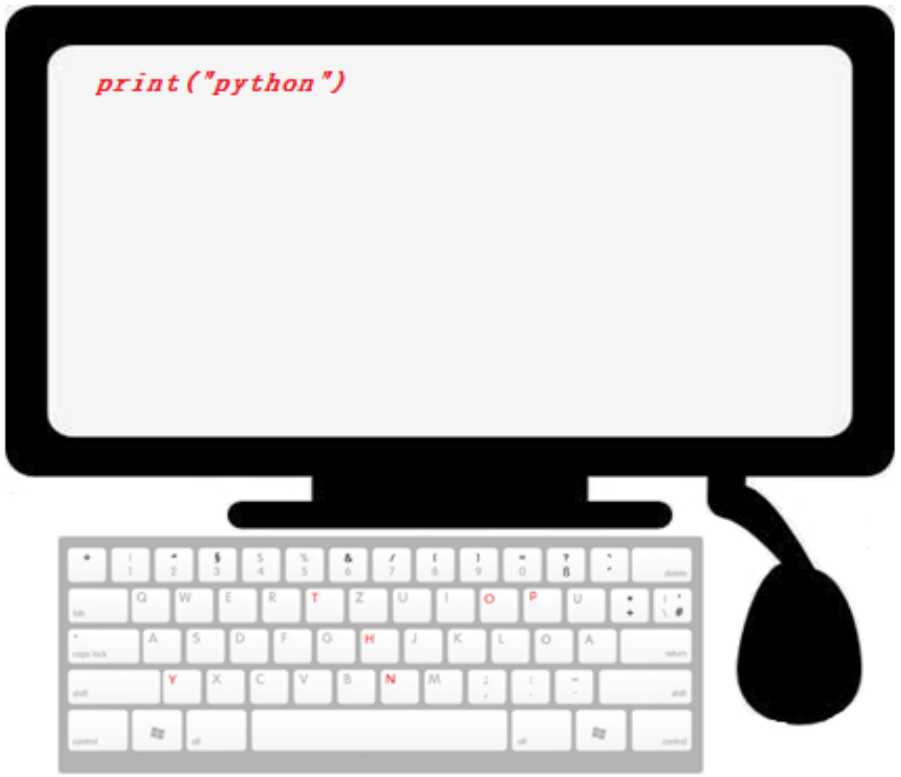
2.3.2 数据类型

字典中的元素是“键（Key）：值（Value）”形式的键值对，键不能重复。Python中使用“{}”创建字典，字典中的各元素以逗号分隔，示例如下：

- {"name": "zhangsan", "age": 18}

2.3.3 变量的输入与输出

程序要实现人机交互功能，需能从输入设备接收用户输入的数据，也需要向显示设备输出数据。



2.3.3 变量的输入与输出

input()函数用于接收用户键盘输入的数据，返回一个字符串类型的数据，其语法格式如下所示：

```
input([prompt])
```

prompt表示函数的参数，用于设置接收用户输入时的提示信息。

2.3.3 变量的输入与输出

print()函数用于向控制台中输出数据，它可以输出任何类型的数据，其语法格式如下所示：

```
print(*objects, sep=' ', end='\n', file=sys.stdout)
```

- objects：表示输出的对象。输出多个对象时，对象之间需要用分隔符分隔。
- sep：用于设定分隔符，默认使用空格作为分隔。
- end：用于设定输出以什么结尾，默认值为换行符\n。
- file：表示数据输出的文件对象。

目录页

2.4.1 打印购物小票

购物小票又称购物收据，是指消费者购买商品时由商场或其它商业机构给用户留存的销售凭据。购物小票中一般会包含用户购买的商品名称、数量、单价以及总金额等信息。本实例要求编写代码，实现打印购物小票的功能。


单号：DH20140923000l
时间：2014-09-23 08:56:14

名称	数量	单价	金额
金士顿U盘8G	1	40.00	40.00
胜创16GTF卡	1	50.00	50.00
读卡器	1	8.00	8.00
网线2米	1	5.00	5.00

总数:4　　总额:103.00
折后总额:103.00
实收:103.00　找零:0.00
收银:管理员

2.4.2 打印蚂蚁森林植树证书

蚂蚁森林是支付宝客户端发起“碳账户”的一款公益活动：用户通过步行、地铁出行、在线消费等行为，可在蚂蚁森林中获取能量，当能量到达一定数值后，用户可以在支付宝中申请一颗虚拟的树，申请成功后会收到支付宝发放的一张植树证书。植树证书中包含申请日期、树苗编号等信息。本实例要求编写代码，实现打印植树证书信息的功能。

目录页

2.5.1 整型

整数类型（int）简称整型，它用于表示整数。整型常用的计数方式有4种，分别是二进制（以“0B”或“0b”开头）、八进制（以数字“0o”或“0O”开头）、十进制和十六进制（以“0x”或“0X”开头）。

示例

- 0b101 # 二进制
- 0o5 # 八进制
- 5 # 十进制
- 0x5 # 十六进制

2.5.1 整型

为了方便使用各进制的数据，Python中内置了用于转换数据进制的函数：bin()、oct()、int()、hex()，关于这些函数的功能说明如下。

函数	说明
bin(x)	将 x 转换为二进制数据
oct(x)	将 x 转换为八进制数据
int(x)	将 x 转换为十进制数据
hex(x)	将 x 转换为十六进制数据

2.5.2 浮点型

浮点型（float）用于表示实数，由整数和小数部分（可以是0）组成例如，3.14、0.9等。较大或较小的浮点数可以使用科学计算法表示。

科学计数法会把一个数表示成a与10的n次幂相乘的形式，数学中科学计数法的格式为：

$$a \times 10^n \quad (1 \le |a| < 10,\ n \in N)$$

Python程序中使用字母e或E代表底数10。

示例

- -3.14e2 # 即-314
- 3.14e-3 # 即0.00314

2.5.2 浮点型

Python中的浮点型每个浮点型数据占8个字节（即64位），且遵守IEEE标准。Python中浮点型的取值范围为-1.8e308~1.8e308，若超出这个范围，Python会将值视为无穷大（inf）或无穷小（-inf）。

2.5.3 复数类型

复数由实部和虚部构成，它的一般形式为：real+imagj，其中real为实部，imag为虚部，j为虚部单位。示例如下：

- complex_one = 1 + 2j　　# 实部为1，虚部为2
- complex_two = 2j　　# 实部为0，虚部为2

通过real和imag属性可以获取复数的实部部分和虚部部分。

2.5.4 布尔类型

布尔类型（bool）是一种特殊的整型，其值True对应整数1，False对应整数0。

布尔值为False的数据

- None。
- False。
- 任何数字类型的0，如0、0.0、0j。
- 任何空序列，如 ‘’ “” 、()、[]。
- 空字典，如{}。

2.5.5 数字类型转换

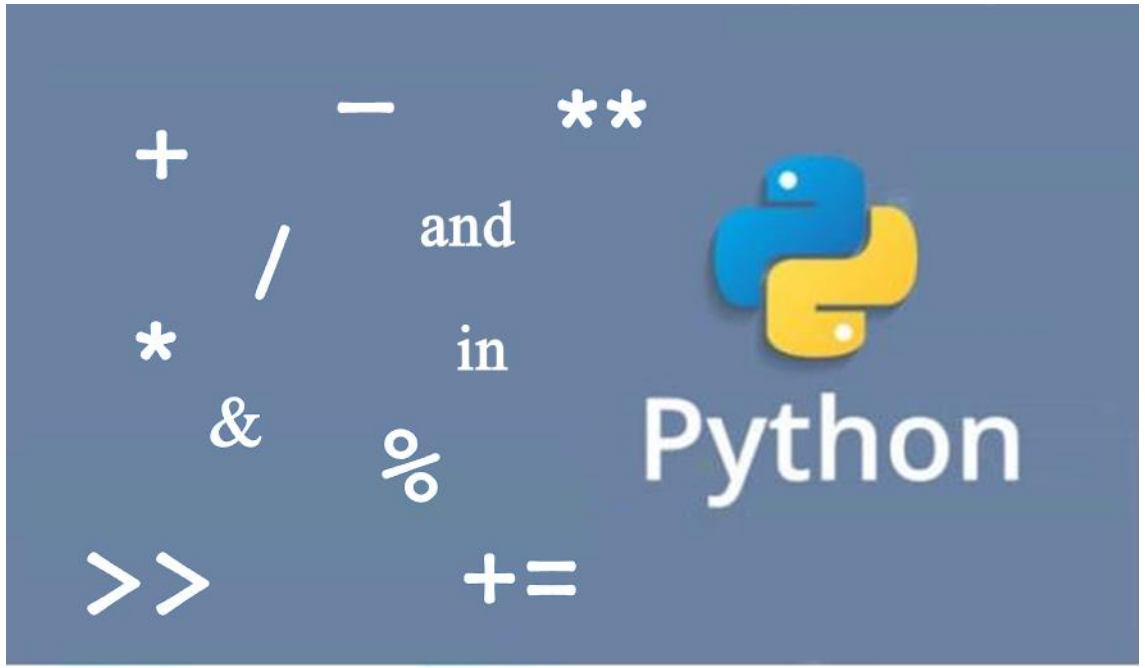
Python内置了一系列可实现强制类型转换的函数，使用这些函数可以将目标数据转换为指定的类型。数字类型间进行转换的函数有int()、float()、complex()。需要注意的是浮点型数据转换为整型数据后只保留整数部分。

函数	说明
int(x[,base])	将 x 转换为一个整型数据
float(x)	将 x 转换为一个浮点型数据
complex(x)	将 x 转换为复数类型

目录页

2.6 运算符

Python运算符是一种特殊的符号，主要用于实现数值之间的运算。根据操作数数量的不同，运算符可分为单目运算符、双目运算符；根据运算符的功能，运算符可分为算术运算符、赋值运算符、比较运算符、逻辑运算符和成员运算符。



2.6.1 算术运算符

Python中的算术运算符包括+、-、*、/、//、%和**。以操作数a = 2，b = 8为例对算术运算符进行使用说明。

运算符	功能说明	示例
+	加：使两个操作数相加，获取操作数的和	a + b，结果为 10
-	减：使两个操作数相减，获取操作数的差	a –b ，结果为-6
*	乘：使两个操作数相乘，获取操作数的积	a * b，结果为 16
/	除：使两个操作数相除，获取操作数的商（除数不能为 0）	a / b，结果为 0.25
//	整除：使两个操作数相除，获取商的整数部分	a // b，结果为 0
%	取余：使两个操作数相除，获取余数	a % b，结果为 2
**	幂：使两个操作数进行幂运算，获取 a 的 b 次幂	a ** b，结果为 256

2.6.1 算术运算符

Python中的算术运算符既支持对相同类型的数值进行运算，也支持对不同类型的数值进行混合运算。在混合运算时，Python会强制将数值的类型进行临时类型转换，这些转换遵循如下原则：

- 整型与浮点型进行混合运算时，将整型转化为浮点型。
- 其他类型与复数运算时，将其他类型转换为复数类型。

2.6.2赋值运算符

- 赋值运算符的作用是将一个表达式或对象赋值给一个左值。左值是指一个能位于赋值运算符左边的表达式，它通常是一个可修改的变量，不能是一个常量。
- 例如将整数3赋值给变量num：num=3。
- 赋值运算符允许同时为多个变量赋值

```
x = y = z = 1   # 变量x、y、z均赋值为1
a, b = 1, 2     # 变量a赋值为1，变量b赋值为2
```

2.6.2 赋值运算符

Python中的算术运算符可以与赋值运算符组成复合赋值运算符，赋值运算符同时具备运算和赋值两项功能。以变量num为例， Python复合赋值运算符的功能说明及示例如下表所示：

运算符	功能说明	示例
+=	变量增加指定数值，结果赋值原变量	num+=2 等价于 num = num+2
-=	变量减去指定数值，结果赋值原变量	num-=2 等价于 num = num－2
*=	变量乘以指定数值，结果赋值原变量	num*=2 等价于 num = num* 2
/=	变量除以指定数值，结果赋值原变量	num/=2 等价于 num = num/2
//=	变量整除指定数值，结果赋值原变量	num//=2 等价于 num = num//2
%=	变量进行取余，结果赋值给原变量	num%=2 等价于 num = num%2
**=	变量执行乘方运算，结果赋值原变量	num**=2等价于num = num**2

2.6.2 赋值运算符

Python3.8中新增了一个赋值运算符——海象运算符“:=”，该运算符用于在表达式内部为变量赋值，因形似海象的眼睛和长牙而得此命名。

```
num_one = 1
# 使用海象运算符为num_two赋值
result = num_one + (num_two:=2)
print(result)
```

2.6.3 比较运算符

比较运算符也叫关系运算符，用于比较两个数值，判断它们之间的关系。Python中的比较运算符包括==、!=、>、<、>=、<=，它们通常用于布尔测试，测试的结果只能是True或False。以变量x=2，y=3为例，具体如下：

运算符	功能说明	示例
==	比较两个数的值是否相等，如果相等返回 True	x==y，返回 False
!=	比较两个数的值是否相等，如果不相等返回 True	x!=y，返回 True
>	比较左数是否大于右操作数，如果大于返回 True	x>y，返回 False
<	比较左数是否小于右操作数，如果小于返回 True	x<y，返回 True
>=	比较左数是否大于等于右操作数，如果大于等于返回 True	x>=y，返回 False
<=	比较左数是否小于等于右操作数，如果小于等于返回 True	x<=y，返回 True

2.6.4 逻辑运算符

Python中分别使用“or”，“and”，“not”这三个关键字作为逻辑运算符，其中or与and为双目运算符，not为单目运算符。以x=10，y=20为例，具体如下：

运算符	逻辑表达式	功能说明	示例
and	x and y	若两个操作数的布尔值均为True，则结果为 y	x and y 的结果为 y
or	x or y	若两个操作数的布尔值均为True，则结果为 x	x or y 的结果为 y
not	not x	若操作数 x 的布尔值为 True，则结果为 False	not x 的结果为 False

2.6.5 成员运算符

成员运算符in和not in用于测试给定数据是否存在于序列（如列表、字符串）中，关于它们的介绍如下：

- in：如果指定元素在序列中返回True，否则返回False。
- not in：如果指定元素不在序列中返回True，否则返回False。

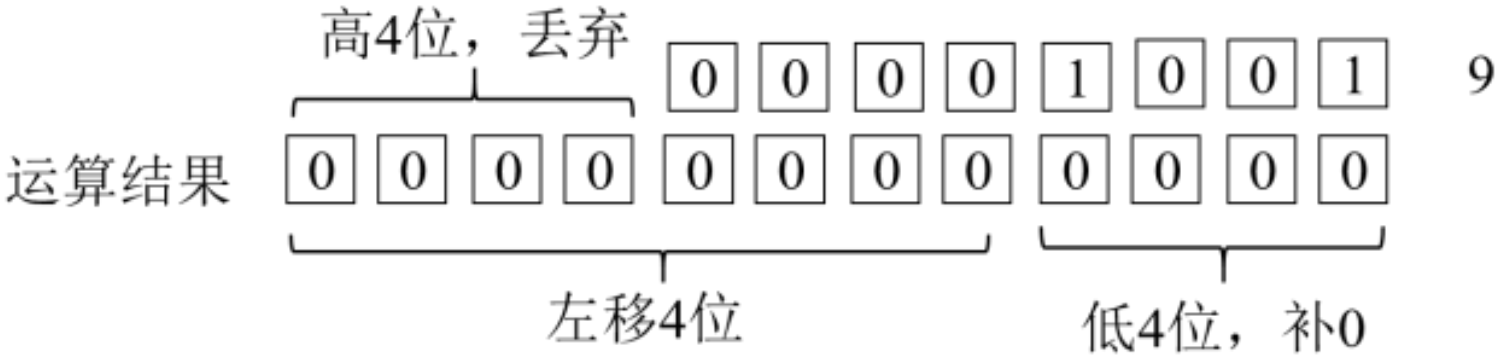
2.6.6 位运算符

位运算符用于按二进制位进行逻辑运算，操作数必须为整数。下面介绍位运算符的功能，并以a=2，b=3为例进行演示，具体如下：

运算符	功能说明	示例
<<	操作数按位左移	a<<b，结果为 16
>>	操作数按位右移	a>>b，结果为 0
&	左操作数与右操作数执行按位与运算	a&b，结果为 2
|	左操作数与右操作数执行按位或运算	a|b，结果为 3
^	左操作数与右操作数执行按位异或运算	a^b，结果为 1
~	操作数按位取反	~a，结果为-3

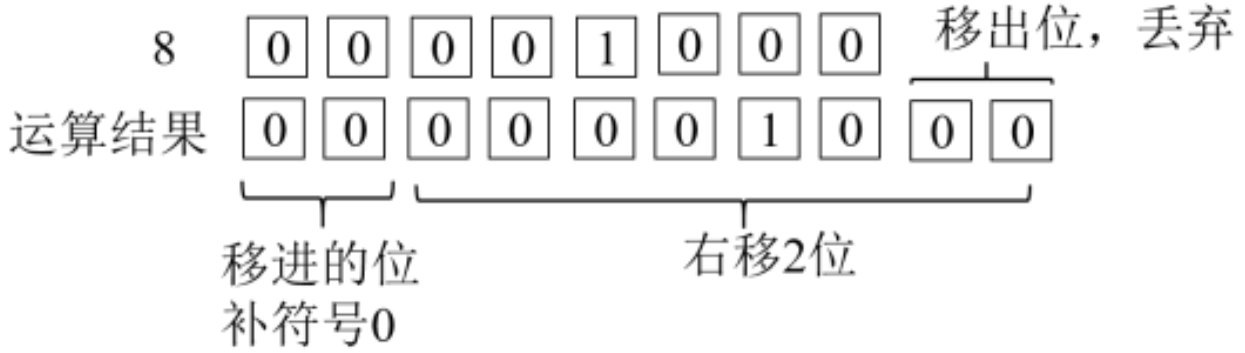
2.6.6 位运算符

按位左移(<<)是指将二进制形式操作数的所有位全部左移n位，高位丢弃，低位补0。以十进制9为例，9转为二进制后是00001001，将转换后的二进制数左移4位。



2.6.6 位运算符

按位右移(>>)是指将二进制形式操作数的所有位全部右移n位，低位丢弃，高位补0。以十进制8为例，8转换为二进制后是00001000，将转换后的二进制数右移2位。



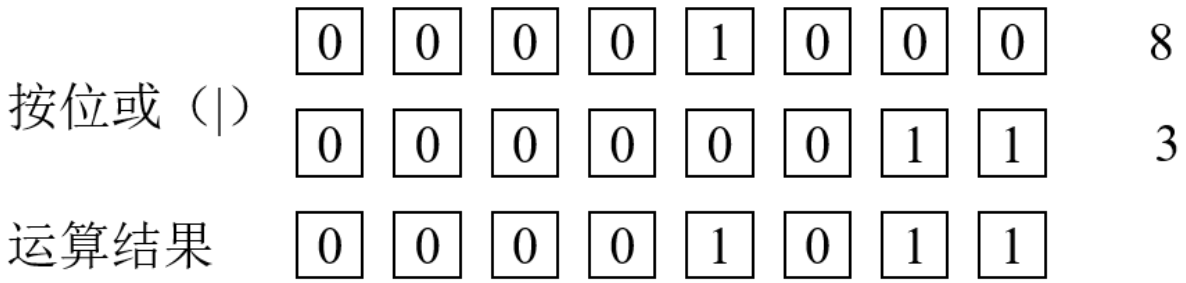
2.6.6 位运算符

按位与(&)是指将参与运算的两个操作数对应的二进制位进行“与”操作。当对应的两个二进制位均为1时，结果位就为1，否则为0。以十进制9和3为例，9和3转换为二进制后分别是00001001和00000011。

	0	0	0	0	1	0	0	1	9
按位与（&）	0	0	0	0	0	0	1	1	3
运算结果	0	0	0	0	0	0	0	1	

2.6.6 位运算符

按位或(|)是指将参与运算的两个操作数对应的二进制位进行“或”操作。若对应的两个二进制位有一个为1时，结果位就为1。若参与运算的数值为负数，参与运算的两个数均以补码出现。以十进制8和3为例，8和3转换为二进制后分别是00001000和00000011。



2.6.6 位运算符

按位异或(^)是指将参与运算的两个操作数对应的二进制位进行“异或”操作。当对应的两个二进制位中有一个为1，另一个为0时，结果位为1，否则结果位为0。以十进制8和4为例，8和4转换为二进制后分别是00001000和00000100。

	0	0	0	0	1	0	0	0	8
按位或（^）	0	0	0	0	0	1	0	0	4
运算结果	0	0	0	0	1	1	0	0	

2.6.6 位运算符

按位取反(~)是指将二进制的每一位进行取反，0取反为1，1取反为0。按位取反操作首先会获取这个数的补码，然后对补码进行取反，最后将取反结果转换为原码，例如，对9按位取反的计算过程如下：

01 因为9是正数，计算机中正数的原码=反码=补码，所以9的补码为00001001

02 对正数9的补码00001001进行取反操作，取反后结果为补码11110110

03 将补码00001001转换为原码时，符号位不变，其他位取反，然后+1得到原码，最终结果为10001010，即-10

2.6.7 运算符优先级

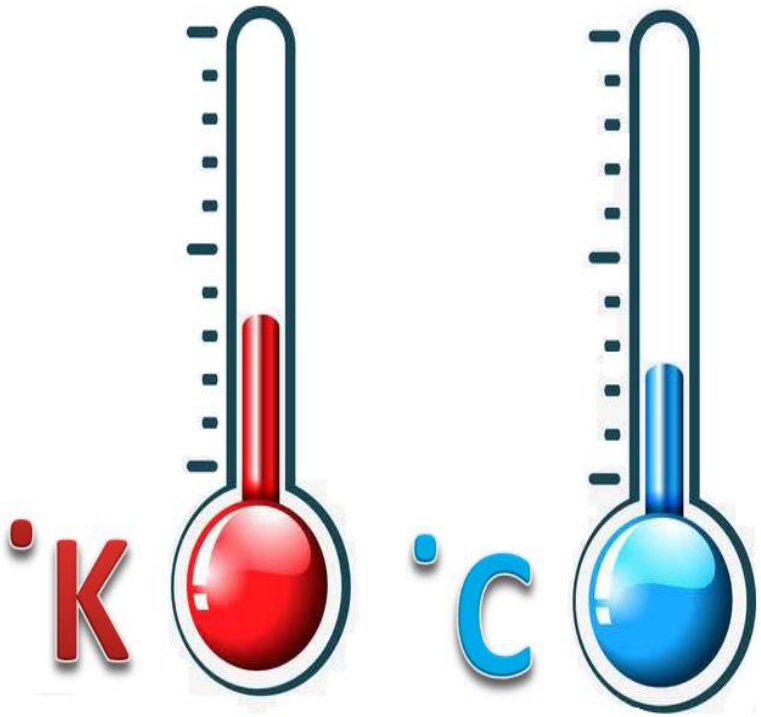
Python支持使用多个不同的运算符连接简单表达式，实现相对复杂的功能，为了避免含有多个运算符的表达式出现歧义，Python为每种运算符都设定了优先级。

Python中运算符的优先级从高到低如下：

运算符	描述
(expression…)	加圆括号的表达式
**	幂（最高优先级）
*、/、%、//	乘、除、取模、整除
+、-	加法、减法
>>、<<	按位右移、按位左移
&	按位与
^、|	按位异或、按位或
==、!=>=、>、<=、<	比较运算符
in、not in	成员运算符
not、and、or	逻辑运算符
=	赋值运算符

目录页

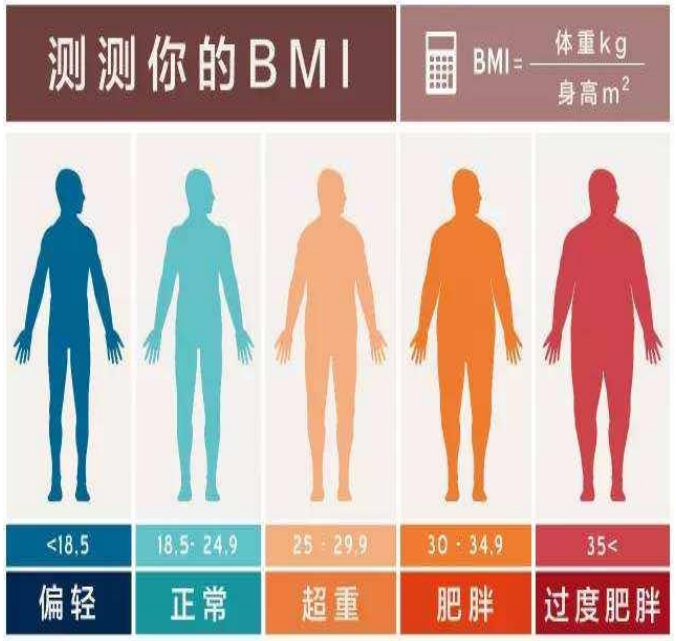
2.7.1 绝对温标



绝对温标又称开氏温标、热力学温标，是热力学和统计物理中的重要参数之一，也是国际单位制七个基本物理量之一。绝对温标的单位为开尔文（简称开，符号为K），绝对温标的零度对应我们日常使用的摄氏温度（单位为摄氏度，简称度，符号为℃）的-273.15℃。

本实例要求编写代码，实现将用户输入的摄氏温度转换为移绝对温标标识的开氏温度的功能。

2.7.2 身体质量指数



BMI指数即身体健康指数，它与人的体重和身高相关，是目前国际常用的衡量人体胖瘦程度以及是否健康的一个标准。已知BMI值的计算公式如下：

体质指数（BMI）= 体重（kg）÷身高^2（m）

本实例要求编写代码，实现根据用户输入的身高体重计算BMI指数的功能。

目录页

2.8本章小结

本章主要介绍了Python基础知识，包括代码格式、标识符和关键字、变量和数据类型、数字类型以及运算符。本章比较简单易学，希望大家在初学Python时，结合实训案例对该部分内容多加练习，为后期深入学习Python打好基础。