Python程序设计

第二讲 Python语言基础
常用内置数据类型

张 华

WHU

常用内置数据类型

- 常用内置数据类型
  - int数据类型（任意精度整数）
  - float类型（有限精度浮点数）
  - complex类型（复数）
  - bool数据类型（布尔逻辑值）
  - str数据类型（字符串）
- 常用运算
  - 关系运算和条件表达式
  - 算术运算符和位运算符
  - 混合运算和类型转换
  - 内置标准数学函数

内置的数据类型

Python内置的数据类型

- Python语言中，一切皆为对象，而每个对象都属于某个数据类型
- Python的数据类型包括内置的数据类型、模块中定义的数据类型和用户自定义的类型
- 数值数据类型：int、bool、float、complex
- 序列数据类型：不可变（str、tuple、bytes）和可变（list、bytearray）
- 集合数据类型：set、frozenset
- 字典数据类型：dict
- NoneType、NotImplementedType和EllipsisType

int类型（任意精度整数）

- 整型常量：数字字符串（前面可以带负号-）

数制	前缀	基本数码	示例
十进制（以 10 为基）		0-9	0，1，2，7，999，-12（负数），+12（正数）
十六进制（以 16 为基）	0x(或 0X)	0-9 和 A-F（或 a-f）	0x0，0X1，0x2，0X7，0x3e7
八进制（以 8 为基）	0o(或 0O)	0-7	0o0，0O1，0o2，0O7，0o1747
二进制（以 2 为基）	0b(或 0B)	0-1	0b0，0B1，0b10，0B111，0b1100011

int类型

创建int对象

int(x=0)　　#创建 int 对象（十进制）

int(x, base=10)　　#创建 int 对象，指定进制为 base（2 到 36 之间）

```
>>> int()
0
>>> int(123)
123
>>> int('123',base=8)
83
```

int对象的方法

i.bit_length()：返回 i 的二进制位数，不包括符号

```
>>> i = -10
>>> bin(i)                          #结果： '-0b1010'
>>> i.bit_length(), int.bit_length(i)   #结果： (4, 4)
```

int类型

- 整数的运算
  - 算术运算
  - 位运算
  - 内置函数
  - math模块的数学运算函数

表达式	结果	说明
123	123	整数字面值
+123	123	正号
-123	-123	负号
7 + 4	11	加法
7 – 4	3	减法
7 * 4	28	乘法
7 // 4	1	整除
7 % 4	3	取余
7 ** 4	2401	乘幂
7 // 0	运行时错误	整除，除数不能为 0
3 * 4 - 3	9	*优先级比-优先级高
3 + 4 // 3	4	//优先级比+优先级高
3 - 4 - 2	-3	左结合运算
2 ** 2 ** 3	256	右结合运算
2 ** 1000	107150…376	乘幂
pow(2,10)	1024	乘幂（调用数学模块函数）

先算2**3（右边），再算2**（左边）

float类型（有限精度浮点数）

浮点类型常量

举例	说明
1.23，-24.5，1.0，0.2	带小数点的数字字符串
1.，.2	小数点的前后 0 可以省略
3.14e-10，4E210，4.0e+210	科学计数法(e 或 E 表示底数 10)，如 3.14e-10=3.14*10^{-10}

创建float对象

float(*x*)

float类型

float对象的方法

方法	说明	示例

as_integer_ratio()	转换为分数	1.25.as_integer_ratio() #结果：(5, 4)
		float.as_integer_ratio(1.25) #结果：(5, 4)

hex()	转换为十六进制字符串	12.3.hex() #结果：'0x1.899999999999ap+3' float.hex(12.3) #结果：'0x1.899999999999ap+3'
fromhex(string) 类方法	十六进制字符串转换为浮点数	float.fromhex('0xFF') #结果：255.0 #格式：[sign] ['0x'] integer ['.' fraction] ['p' exponent]
is_integer()	判断是否为 int 类型	3.14.is_integer() #结果：False float.is_integer(2.0) #结果：True

浮点数的运算

- 算术运算
- math模块中浮点数运算的函数

complex类型（复数）

创建complex对象

```
complex(real[, imag])    #创建 complex 对象（虚部可选）
```

complex类型的方法

属性/方法	说明	示例
real	复数的实部	>>> (1+2j).real #结果： 1.0
imag	复数的实部	>>> (1+2j).imag #结果： 2.0
conjugate()	共轭复数	>>> (1+2j).conjugate() #结果： (1-2j)

complex类型

复数的运算

- 算术运算、内置函数、cmath模块中复数运算

```
>>> a = 1 + 2j
>>> b = 1.1 + 2.2j
>>> c = complex(4, 5)
>>> a + b + c      #结果：(6.1+9.2j)
>>> 1J * 1j        #结果：(-1+0j)
>>> sqrt(c)        #NameError: name 'sqrt' is not defined
>>> import cmath
>>> cmath.sqrt(c)  #结果：(2.2806933416652 98+1.096157889501519j)
```

bool数据类型和相关运算符

- bool数据类型包含两个值
  - True（真）或False（假）
- bool对象示例

```
>>> bool(0)        #输出：False
>>> bool(1)        #输出：True
>>> bool("abc")    #输出：True
```

0值（包括0.0）为假
非0值为真

逻辑运算符

运算符	含义	说明	优先级	实例	结果
not	逻辑非	当操作数为 False 时返回 True；当操作数为 True 时返回 False	1	not True not False	False True
and	逻辑与	两个操作数均为 True 时，结果才为 True，否则为 False	2	True and True True and False False and True False and False	True False False False
or	逻辑或	两个操作数中有一个为 True 时，结果即为 True，否则为 False	3	True or True True or False False or True False or False	True True True False

str数据类型（字符串）

- Python中没有独立的字符数据类型，字符即长度为1的字符串。
- Python内置数据类型str，用于字符串处理。
  - str对象的值为字符序列
  - str对象（字符串）是不可变对象

Python字符串字面量

字符串字面量的形式

（1）单引号（' '）。包含在单引号中的字符串，其中可以包含双引号。
（2）双引号（" "）。包含在双引号中的字符串，其中可以包含单引号
（3）三单引号（''' '''）。包含在三单引号中的字符串，可以跨行。
（4）三双引号（""" """）。包含在三双引号中的字符串，可以跨行。

字符串字面量示例

```
>>> 'abc'                  #输出：'abc'
>>> "Hello"                #输出：'Hello'
>>> type("python")         #输出：<class 'str'>
```

字符串编码

- Python 3字符默认为16位Unicode编码
  - 使用内置函数**ord()**可以把字符转换为对应的Unicode码
  - 使用内置函数**chr()**可以把十进制数转换为对应的字符

```
>>> ord('A')        #输出：65
>>> chr(65)         #输出：'A'
>>> ord('张')       #输出：24352
>>> chr(24352)      #输出：'张'
```

转义字符

转义序列	字符	转义序列	字符
\'	单引号	\n	换行（LF）
\"	双引号	\r	回车（CR）
\\	反斜杠	\t	水平制表符（HT）
\a	响铃（BEL）	\v	垂直制表符（VT）
\b	退格（BS）	\ooo	八进制 Unicode 码对应的字符
\f	换页（FF）	\xhh	十六进制 Unicode 码对应的字符

转义字符串示例

```
>>> s = 'a\tb\tc\\td'
>>> s                    #输出：  'a\tb\tc\\td'
>>> print(s)             #输出：  a       b       c\td
```

str对象

创建str类型的对象实例

```
str(object='')      #创建 str 对象，默认为空字符串
```

str对象示例

```
>>> str(123)        #输出：'123'
>>> str(True)       #输出：'True'
>>> str(3.14)       #输出：'3.14'
```

str对象属性和方法

- str对象的方法有两种调用方式：
  - 字符串对象的方法
  - str类方法

- str对象方法示例

```
>>> s='abc'
>>> s.upper()    #字符串对象 s 的方法。输出：'ABC'
>>> str.upper(s)  #str 类方法，字符串 s 作为参数。输出：'ABC'
```

创建新对象

字符串的运算

- 字符串对象支持的运算操作
  - 关系运算、使用运算符+拼接两个字符串、内置函数、str对象方法等。
- 字符串实际上是字符序列，故支持序列数据类型的基本操作，包括
  - 索引访问、切片操作、连接操作、重复操作、成员关系操作、以及求字符串长度、最大值、最小值等

对象转换为字符串

- 使用内置函数str()可以把数值转换为字符串
  - 用print(123)输出数值时，将自动调用str(123)函数，把123转换为字符串，然后输出
- 另一个内置函数repr()，函数repr()返回一个对象的更精确的字符串表示形式

```
>>> repr('abc123')
"'abc123'"
>>> str('abc123')
'abc123'
```

- 对象转换为字符串示例

```
>>> c=1/3
>>> str(c)          #输出：'0.3333333333333333'
>>> repr(c)         #输出：'0.3333333333333333'
```

字符串的格式化

- 字符串.format(值 1, 值 2,…)
- str.format(格式字符串 1，值 1, 值 2,…)
- format(值, 格式字符串)
- 格式字符串 % (值 1, 值 2,…) #兼容 Python 2 的格式，不建议使用

```
>>> "学生人数{0}，平均成绩{1}".format(15, 81.2)
'学生人数 15，平均成绩 81.2'
>>> str.format("学生人数{0}，平均成绩{1:2.2f}", 15, 81.2)
'学生人数 15，平均成绩 81.20'
>>> format(81.2, "0.5f")      #输出：'81.20000'
>>> "学生人数%4d，平均成绩%2.1f" % (15, 81)
'学生人数  15，平均成绩 81.0'
```

字符串的格式化

字符串示例（string.py）：格式化输出字符串堆积的三角形

```
print("1".center(20))              #1 行 20 个字符，居中对齐
print(format("121", "^20"))        #1 行 20 个字符，居中对齐
print(format("12321", "^20"))      #1 行 20 个字符，居中对齐
print("1".rjust(20,"*"))           #1 行 20 个字符，右对齐，加*号
print(format("121", "*>20"))       #1 行 20 个字符，右对齐，加*号
print(format("12321", "*>20"))     #1 行 20 个字符，右对齐，加*号
```

```
          1
        121
      12321
*******************1
*****************121
***************12321
```

条件表达式

- 条件表达式通常用于选择语句中，用于判断是否满足某种条件
  - 如果表达式的结果为数值类型（0）、空字符串（“”）、空元组（()）、空列表（[]）、空字典（{}），None，则其bool值为False（假）；
  - 否则其bool值为True（真）。
  - 例如：123、"abc"、(1,2)均为True

关系运算和条件表达式

条件表达式示例

```
>>> bool(123),bool("abc"),bool((1,2)),bool([0]),bool(0)
(True, True, True, True, False)
>>> bool(1>2),bool(1>2 or 3>2),bool(1<=2 and 3>2)
(False, True, True)
```

关系运算符

- 关系运算符用于将两个操作数的大小进行比较。若关系成立，则比较的结果为True，否则为False

- 两个相同类型的对象之间的比较

```
>>> 1 > 2                    #输出： False
>>> "ab123" > "ab12"         #输出： True
```

"123">"23" → False
字符串的比较规则是同位次的、从左到右进行比较，最高位的1<2，故错

关系运算符

- 数值类型（包括布尔型，True自动转换为1，False自动转换为0）之间可以进行比较

```
>>> 1 > 1.23        #输出： False
>>> 2>True          #输出： True
>>> 123>"abc"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unorderable types: int() > str()
```

关系和测试运算符

运算符	表达式	含义	实例	结果
==	x == y	x 等于 y	"ABCDEF" == "ABCD"	False
!=	x != y	x 不等于 y	"ABCD" != "abcd"	True
>	x > y	x 大于 y	"ABC" > "ABD"	False
>=	x >= y	x 大于等于 y	123 >= 23	True
<	x < y	x 小于 y	"ABC" < "上海"	True
<=	x <= y	x 小于等于 y	"123" <= "23"	True
is	x is y	x 和 y 是同一个对象	x=y=1; x is y x=1; y=2; x is y	True False
is not	x is not y	x 和 y 不是同一个对象	x=1; y=2; x is not y	True
in	x in y	x 是 y 的成员（y 是容器，如元组）	1 in (1, 2, 3) "A" in "ABCDEF"	True True
not in	x not in y	x 不是 y 的成员（y 是容器，如元组）	1 not in (1, 2, 3)	False

算术运算符

运算符	含义	说明	优先级	实例	结果
+	一元+	操作数的值	1	+n	8
-	一元-	操作数的反数	1	-n	-8
*	乘法	操作数的积	2	n*n*2	128
/	除法	第二个操作数除第一个操作数	2	10 / n	1.25
//	整数除法	两个整数相除，结果为整数	2	10 // n	1
%	模数	第二个操作数除第一个操作数后的余数	2	10 % n	2
+	加法	两个操作数之和	3	10 + n	18
-	减法	从第一个操作数中减去第二个操作数	3	n - 10	-2

位运算符

运算符	用法	含义	优先级	实例	结果
~	~op	按位求补	1	~0x1	-0x2
<<	op1<<op2	将 op1 左移 op2 位	2	0xf0 << 4	0xf00
>>	op1>>op2	将 op1 右移 op2 位	2	0xf0 >> 4	0xf
&	op1&op2	按位逻辑与	3	0xff00 & 0xf0f0	0xf000
^	op1^op2	按位逻辑异或	4	0xff00 ^ 0xf0f0	0xff0
|	op1|op2	按位逻辑或	5	0xff00 | 0xf0f0	0xfff0

混合运算和类型转换

混合运算和隐式转换

- int、float和complex对象可以混合运算
  - 如果表达式中包含**complex**对象，则其他对象自动转换（隐式转换）为**complex**对象，结果为**complex**对象
  - 在没有**complex**对象的表达式中如果包含**float**对象，则其他对象自动转换（隐式转换）为**float**对象，结果为**float**对象
- 示例

```
>>> f = 123 + 1.23
>>> f                    #输出： 124.23
>>> type(f)              #输出：<class 'float'>
>>> 123 + True           #True 转换为 1。输出:124
>>> 123 + False          # False 转换为 0。输出： 123
```

混合运算和类型转换

- 显式转换（强制转换）
  - 使用target-type(value)将表达式强制转换为所需的数据类型。
  - 示例

```
>>> int(1.23)              #输出：1
>>> float(10)              #输出：10.0
>>> bool("abc")            #输出：True
>>> float("123xyz")
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    float("123xyz")
ValueError: could not convert string to float: '123xyz'
```

数制转换函数

函数	说明	示例
bin(number)	数值转换为二进制字符串	bin(100) #结果：'0b1100100'
hex(number)	数值转换为十六进制字符串	hex(100) #结果：'0x64'
oct(number)	数值转换为八进制字符串	oct(100) #结果：'0o144'

内置标准数学函数

函数	含义	实例	结果
abs(x)	数值 x 的绝对值。如果 x 为复数，则返回 x 的模	abs(-1.2) abs(1-2j)	1.2 2.23606797749979
divmod(a,b)	返回 a 除以 b 的商和余数	divmod(5,3)	(1, 2)
pow(x, y[, z])	返回 x 的 y 次幂（x**y）。如果指定 z，则为：pow(x, y) % z	pow(2,10) pow(2,10,10)	1024 4
round(number[, ndigits])	四舍五入取整。如果指定 ndigits，则保留 ndigits 小数	round(3.14159) round(3.14159,4)	3 3.1416
sum(iterable[, start])	求和	sum((1, 2, 3)) sum((1, 2, 3), 44)	6 50