

第二章 基本数据类型



基本概念





数据、对象、变量

数据：数字、文字、图形图像和声音等

对象：Python用对象表示数据

对象三要素：

identity ——通常是对象在内存中的位置

type ——对象类型

value ——对象的值

变量：对象的引用，存储的是对象的地址。因此变量本身是没有类型，只有对象才有类型。引用的名称就是我们经常说的变量名，是一种标识符。

例如：对象数字“1”：

```
>>> id(1),type(1),1
(4521265952, <class 'int'>, 1)
>>> a = 1
>>> id(a),type(a),a
(4521265952, <class 'int'>, 1)
```

例如：对象字符串“abc”：

```
>>> id("abc"),type("abc"),"abc"
(140250279564912, <class 'str'>, 'abc')
>>> s = "abc"
>>> id(s),type(s),s
(140250279564912, <class 'str'>, 'abc')
```





对象驻留机制 (Object Interning) (*)

对象驻留机制是一种在编程中用于优化内存使用和对对象创建的技术。

基本思想：将某些类型的对象，尤其是那些不可变的对象（整数、字符串、元组等），在内存中保留一份，以便在需要时重复使用，而不是每次需要时都创建新的对象。

例：Python的一个称为“小整数池”（Small Integer Pool），它预先创建了一个包含[-5, 256]范围内整数的对象池。当程序需要这些范围内的整数时，Python会重用池中的对象，而不是创建新的对象。这种机制可以显著减少内存使用和垃圾收集的开销。

■ 对象驻留机制的实现细节可能因Python的版本和解释器的不同而有所变化。





标识符

■ **标识符**：用来标识某个实体的一个符号（可以是变量的名称、函数名、类名、方法名、包名等等），在不同的应用环境下有不同的含义。

■ Python语言标识符：

- 由**字母**、**数字**和**下划线**构成的字符序列。
- 必须以字母或下划线开头，不能以数字开头。
- 不能是关键字（保留字）。
- 可以为任意长度，中间不能出现空格。

➤ **注意**：标识符对大小写敏感，**total**和**Total**是两个不同的名字

合法标识符如：my_test _123

非法标识符，如：3x ab&





标识符约定命名规则

■ “见名知义”，规范大小写的使用方式

1. 大多数为小写字母开头

- ◆ 变量名、函数名、类名
- ◆ 标识符由多个单词构成，则首字母小写，其后单词的首字母大写，其余字母小写。如 `getAge`。

2. 类名首字母大写

3. 常量名全部字母大写

此规则非强制性





保留字

- **保留字**，也称为**关键字**，指被编程语言内部定义并保留使用的标识符。
- 程序员编写程序不能定义与保留字相同的标识符。
- 每种程序设计语言都有一套保留字，保留字一般用来构成程序整体框架、表达关键值和具有结构性的复杂语义等。
- 掌握一门编程语言首先要熟记其所对应的保留字。

注：内置的函数名一旦被用于作为某个变量名，虽然合法但是原有的函数功能就会丧失。





保留字

■ Python 3.x保留字列表

```
>>> 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']
```



主要数据类型





主要数据类型

■ 数字类型

- 整数 int
- 浮点数 float
- 复数 complex
- 布尔值 bool

不可变类型：一旦创建，内容不能被修改。

对不可变对象的任何“修改”操作，实际上是创建一个新对象。

■ 字符串 str

■ 元组 tuple

■ 列表 list

■ 集合 set

■ 字典 dict

可变类型：内容可以在原地修改，无需创建新对象

（对象的id不变）

字面量（Literals）：直接在代码中明确表示固定值的语法形式，字面量的类型由其书写格式决定，Python 会自动识别其数据类型。



数字类型





整型 (int)

Python的整数类型在逻辑行为上与数学整数高度一致，支持任意大小和精确运算，满足大多数数学场景的需求。

进制	前缀	数码	示例
十进制	无	0--9	123、-8、0
二进制	0b 或 0B	0、1	0b101
八进制	0o 或 0O	0--7	0o17
十六进制	0x 或 0X	0--9、a--f或A--F	0x1FF

bin(5)

'0b101'

oct(11)

'0o13'

hex(29)

'0x1d'

转换以后的数据类型是str

int('101',2)

5

int('13',8)

11

int('1d',16)

29

转换以后的数据类型是int





浮点数(float)

- 在计算机科学中，Python的浮点数（float）与数学中的实数（Real Numbers）在概念上有显著差异：

维度	数学实数	Python浮点数
连续性	无限连续，任意精度	离散近似，有限精度（如64位双精度）
范围	无界（ $-\infty$ 到 $+\infty$ ）	有界（约 $\pm 1.8e308$ 到 $\pm 5e-324$ ）
精度	无限精确	约15-17位有效十进制数字
特殊值	无	inf（无穷大）、-inf、NaN（非数）
运算规则	严格符合数学公理	存在舍入误差和精度损失

- 字面量：

- 小数形式：1.23, 3.14, -9.01
- 科学记数法：1.23x10⁹就是1.23e9，0.000012可以写成1.2e-5。

“e”的前后都不能空，“e”的后面要整数

- float()：将某些类型数据转换成浮点型





浮点数误差

在Python中，浮点型是基于IEEE 754标准的双精度进行表示，具体的信息可以通过 `sys.float_info` 查看，包括如数据范围、有效位数、取舍方式等。

```
import sys  
sys.float_info
```

```
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)
```

特别注意：

- 与整数不同，存在范围限制，超出会产生溢出错误。
- 不是所有的实数在计算机里都可以精确表达，会存在误差。

思考题：

- 为什么会存在误差？
- 在python中 `0.1+0.2` 的结果是0.3吗？
- 浮点误差会带来什么危害？
- 需要高精度计算怎么办？





复数 (complex)

■ 在数学、工程、物理学等多个领域有一种重要的数据就是复数，python也提供了复数类型，由实部 (real) 和虚部 (imaginary) 两部分组成，虚部用j表示。

■ 字面量：2+3j、8j

■ complex()：创建一个复数。complex(1,2) → (1+2j)

成员描述符：实部real，虚部imag

(1+2j).real → 1.0 (float)

(1+2j).imag → 2.0 (float)





布尔类型 (bool)

布尔类型只有两个值：**True** 和 **False** (请注意大小写)

Python3 中，bool 是 int 的子类，True 和 False 可以和数字进行算术运算，True 等价于1、False等价于0

(后续会介绍与之相关的关系运算和逻辑运算)





算术表达式

如何实现将数学等计算转换为python算术表达式

例： $\cos\left(\frac{a(x+1)+b}{2}\right)$ ，a等于2，x等于5，b等于3

- 算术运算符
- 内置函数
- 标准库中的模块，如math
- 第三方库，如numpy





算术运算符

运算符	示例	结果	描述	优先级
**	2 ** 3	8	幂运算，求幂次方	1
+、-	-5	-5	正、负号，一般正号省略不写。属于一元运算符	2
*	7 * 6	42	乘号，实现乘法运算	3
/	9 / 2	4.5	除号，实现除法运算	3
//	9 // 2	4	地板除，实现两个数相除后，取不大于商的最大整数作为结果	3
%	10 % 3	1	取模，求余数	3
+	5 + 3	8	加号，实现加法运算	4
-	10 - 4	6	减号，实现减法运算	4

不同类型数据混合运算时，遵循bool→int→float→complex的方向自动转换，这样使结果不受精度损失。





常用内置函数

函数名	描述	示例
abs(x)	返回数字x的绝对值	abs(-5) 返回 5
divmod(x, y)	返回x除以y的商和余数，结果为一个元组(商，余数)	divmod(10, 3) 返回 (3, 1)
pow(base, exp, mod=None)	返回base的exp次幂，如果给定mod，则返回base的exp次幂后对mod取余的结果。	pow(2, 3) 返回 8 , pow(2, 3, 3) 返回 2
round(number, ndigits=None)	将number四舍五入到ndigits位小数。	round(3.141, 2) 返回 3.14
max(x₁, x₂, x₃...x_n)	返回x ₁ , x ₂ , x ₃ ...x _n 中的最大值	max(2, 1, 5, 6, 9) 返回9
min(x₁, x₂, x₃...x_n)	返回x ₁ , x ₂ , x ₃ ...x _n 中的最小值	min(2, 1, 5, 6, 9) 返回1





数学模块(math)

math模块：是Python标准库中的一个模块，它提供了大量数学函数，涵盖了基本的算术运算、幂和对数函数、三角函数、数论函数、数学常数（如 π 和 e ）等。

模块使用步骤：

1. 导入模块
2. 调用模块内函数

导入格式：`import math`

调用语句：`math.sqrt()`



字符串





字符串

Python语言中，字符串是用**引号**括起来的一个或多个字符。

■ 重要性：字符串操作广泛应用于数据处理、文件读写和网络通信等场景。

■ 特性：

- 不可变性：无法原地修改对象内容。
- 灵活的操作：处理文本的常见需求（分割、替换、格式化等）。
- Unicode 支持：便于处理多语言文本。





字符串操作

- 创建字符串
 - 字面量创建
 - `str()`转换
- 编码字符转换
 - `chr()`
 - `ord()`
- 拼接: `+`
- 重复: `*`
- 判断子串
- 索引和切片





常用字符串方法

方法	功能	示例
<code>split()/join()</code>	分割与合并	<code>",".join(["a", "b"]) → "a,b"</code>
<code>strip()</code>	去除两端空白或指定字符	<code>" text ".strip() → "text"</code>
<code>upper()/lower()</code>	大小写转换	<code>"Hello".lower() → "hello"</code>
<code>startswith()/endswith()</code>	判断开头/结尾	<code>s.startswith("Py")</code>
<code>find()/index()</code>	查找子串位置 (find未找到返回-1)	<code>s.find("th")</code>
<code>replace()</code>	替换子串	<code>s.replace("old", "new")</code>
<code>isdigit()/isalpha()</code>	判断是否为数字/字母	<code>"123".isdigit() → True</code>

课后任务：

理解方法和函数的区别

探索学习字符串的其他方法。





特殊字符串

- **转义字符：** 用于表示那些难以直接输入的字符或具有特殊意义的字符。

转义字符	功能
<code>\n</code>	换行符，用于将光标移动到下一行。
<code>\t</code>	制表符，用于在文本中插入一个水平制表位。
<code>\r</code>	回车符。用于将光标移动到当前行的开头，而不会移动到下一行
<code>\\</code>	反斜杠本身，因为反斜杠在字符串中有特殊意义，所以需要用两个反斜杠来表示一个实际的反斜杠。
<code>\'</code>	单引号，用于在字符串中包含单引号字符，避免与字符串的定界符冲突。
<code>\"</code>	双引号，同理，用于在字符串中包含双引号字符。

- **r字符串（原始字符串）：**

以字母r或R开头，不需要进行转义处理，所有字符都将被视为字面量。

- **f字符串（格式字符串）：**

以字母f或F开头，它允许在字符串中嵌入用花括号{}包围的简单变量、表达式，函数调用等，方便实现字符串格式化。





内置函数：input()

功能：用于从标准输入（通常是键盘）读取用户输入实现从键盘输入一个**字符串**。

返回值：输入的内容会作为字符串（str）返回

语法格式：变量 = input(prompt=None) 蓝色部分为可选。

```
>>> a=input()
```

```
9
```

```
>>> type(a)
```

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

注：看似整数类型，但实则是字符串类型，若想获得整数类型，可用int()函数进行转换：a=int(input())





小练习

功能要求：

1. 设置输入提示语
2. 在一行输入中获得多个值

```
>>>m,n=input( "请输入多个值： ").split()
```

请输入多个值： 3 5

```
>>>m
```

```
'3'
```

```
>>>n
```

```
'5'
```

练习：输入形如1/3的分数，获取分子和分母的值

```
numerator, denominator = input().split('/')  
numerator = int(numerator)  
denominator = int(denominator)
```

```
numerator, denominator = map(int, input().split('/'))
```





内置函数：print()

功能： 用于将信息输出到标准输出（通常是控制台或屏幕终端）的内置函数。

该函数可以接受多个参数，并可以通过设置各种参数来控制输出的格式。

语法格式： `print(value1, value2, ..., sep=' ', end='\n')`

```
>>> print(100)
```

```
100
```

```
>>> print("Hello World!")
```

```
Hello World!
```

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

```
>>> print(a)
```

```
1
```





小练习

问题：

1. 如何设置多个输出值之间的分隔

```
print(a,b,c,sep=',')
```

2. 如何设置输出的结束符

```
print(a,end='')
```

练习：尝试写出以下代码输出结果

```
1 a = "apple"
2 b = "banana"
3 c = "orange"
4 print("第一组输出：")
5 print(a)
6 print(b)
7 print(c)
8 print("第二组输出：")
9 print(a,end='')
10 print(b,end='')
11 print(c,end='')
12 print()
13 print("第三组输出：")
14 print(a,b,c)
15 print("第四组输出：")
16 print(a,b,c,sep=',')
```

第一组输出：

apple
banana
orange

第二组输出：

applebananaorange

第三组输出：

apple banana orange

第四组输出：

apple,banana,orange





输出格式控制

```
1 #需要输出的数据
2 name = "Alice"
3 age = 25
4 height = 180.5
5
6 #方法一：使用%
7 print("我的名字是 %s, 年龄是 %d 岁, 身高是 %.1f cm." % (name, age, height))
8
9 #方法二：使用format()方法
10 print("我的名字是 {:s}, 年龄是 {:d} 岁, 身高是 {:.1f} cm.".format(name, age, height))
11
12 #方法三：使用f-string
13 print(f"我的名字是 {name}, 年龄是 {age} 岁, 身高是 {height:.1f} cm.")
```

我的名字是 Alice, 年龄是 25 岁, 身高是 180.5 cm。
我的名字是 Alice, 年龄是 25 岁, 身高是 180.5 cm。
我的名字是 Alice, 年龄是 25 岁, 身高是 180.5 cm。

课后任务：

探索f字符串的格式控制

