

► **INFO**

本章重点介绍数据：我们研究的技术使我们可以表示和操作许多不同领域的信息。由于互联网的爆炸式增长，所有人都可以在网上免费获取大量结构化信息，并且计算也可以应用于范围广泛的不同问题。有效使用内置数据类型和用户定义的数据类型是数据处理型应用（data processing applications）的基础。

Python 中的每个值都有一个类（class）来确定它的类型。拥有相同类的值，行为也相同。例如，整数 1 和 2 都是 `int` 类的实例，我们就可以使用相似的方法进行处理。例如，它们都可以取反或与另一个整数相加。内置的 `type` 函数允许我们检查任何值的类。

py

py

<https://composingprograms.netlify.app/2/1>

```
>>> type(1.5)
<class 'float'>
>>> type(1+1j)
<class 'complex'>
```

浮点数：“Float”这个名字来源于 Python 和许多其他编程语言中对实数的表示方式：就是“具有浮动的小数点”的值。虽然关于数字如何表示的细节不是本文的主题，但了解 `int` 和 `float` 对象之间的一些区别是很重要的。特别是，`int` 对象可以精确地表示整数，没有任何近似处理或大小限制。另一方面，`float` 对象可以表示很大范围内的小数，但并不是所有的数字都能被精确表示，它有最小值和最大值之分。因此，`float` 值应被视为真实值的近似值，它们只能保证有限的精度，组合 `float` 值可能会导致近似误差；如果不进行近似处理，下面两个表达式的计算结果均为 7。

译者注：不同于其他的编程语言，Python3 中的 `int` 值是无界的，也就是说它可以存储任意大小的数，具体可以查看 [PEP 237](#)。

py

```
>>> 7 / 3 * 3
7.0
>>> 1 / 3 * 7 * 3
6.999999999999999
```

尽管上式是 `int` 值的组合，但一个 `int` 值除以另一个 `int` 值，却会得到一个 `float` 值：一个截断的有限近似值，相当于两个整数相除的实际比值。

py

```
>>> type(1/3)
<class 'float'>
>>> 1/3
0.3333333333333333
```

当我们使用等式测试时，这种近似就会出现問題。

py

```
>>> 1/3 == 0.333333333333333312345 # 请注意浮点数近似
True
```

`int` 和 `float` 类之间的细微差别对编写程序有着广泛的影响，因此，这也是程序员必须熟记的细节。幸运的是，原始数据类型的数量很少，这无疑减少了精通一门编程语言所需的记忆量。此外，这些细节在许多编程语言中都是一致的，它们会由 [IEEE 754 浮点标准](#) 等社区指南强制要求执行。

非数值类型 (Non-numeric types)：值可以表示许多其他类型的数据，比如声音、图像、位置、网址、网络连接等等。它们中间的少数可以用原始数据类型表示，例如用于值 `True` 和 `False` 的 `bool` 类，其他大多数值的类型必须由程序员使用我们将在本章中学习到的组合和抽象方法来定义。

下面几节将介绍更多 Python 的原始数据类型，并将重点介绍它们在创建数据抽象方面所起到的作用。那些对更多细节感兴趣的人，可以查看在线书籍 Dive Into Python 3 中的一个关于 [原始数据类型](#) 的章节，它给出了所有 Python 的原始数据类型以及如何操作它们的实用概述，包括大量的使用示例和实用技巧。

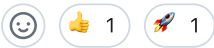
[在 GitHub 上编辑此页面](#)

最后更新于: 2023/10/28 12:46:12

上一页
[1.7 递归函数](#)

下一页
[2.2 数据抽象](#)

2 个表情



2 条评论 – 由 *giscus* 提供支持

[最早](#) [最新](#)