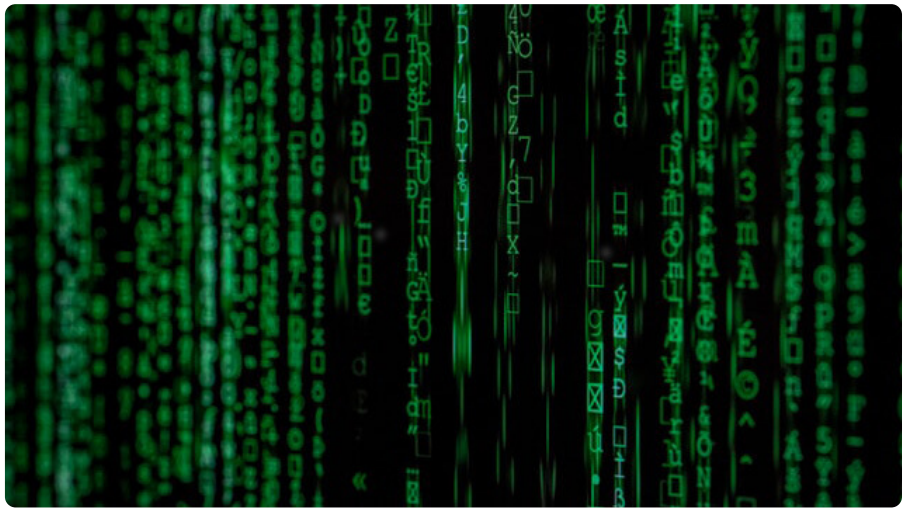
目录

05 数据的名字和种类—变量和类型

更新时间：2019-08-23 11:03:57



> 不经一翻彻骨寒，怎得梅花扑鼻香。
>
> ——宋帆

初探数据种类

在正式开始学习这个小节之前你要明白，现在我们是在学习写程序。那么在写程序之前你要知道程序的作用是什么？

程序的主要作用是处理数据。数据的种类有很多，我们在手机和电脑上看到的那些文字、数字、图片、视频、页面样式等等都是数据。这些数据都是由程序来处理并显示到屏幕上的。

虽然数据的种类形形色色，并且有些看起来比较复杂，但是在编程时它们实际上都是由一些非常基本的数据形式（或经过组合）来表示。这些基本数据形式有哪些呢？比如有常用到的数字和字符，以及其它的诸如数组、字节序列等形式。

以数字和字符为例，为大家介绍下在代码中它们是怎么表示的。

对于数字，数字在代码中的表示形式和平时的电脑输入一样，直接书写即可：

```
123
```

```
3.14159
```

对于字符，和平时的书写稍有不同，Python 代码中表示字符时一定要给字符括上单引号或双引号：

```
'How are you?'
```

这些不同的数据表示（书写）形式，对应着不同的数据种类，而不同的数据种类又具有不同的功能或者作用。

我们将代码中的数据种类称为**数据类型**，也就是数据的类型。

数据类型

代码中的所有数据都是有类型的。

数字所对应的数据类型有**整数型**以及**浮点型**。整数型表示整数数字，比如： `0` ， `-59` ， `100` 。浮点型表示小数数字，如 `-3.5` ， `0.25` ， `0.0` 。

字符所对应的数据类型叫**字符串**，所谓字符串就是一串字符。它里面可以是任意语言的字符，比如 `'哼哼哈嘿'` ， `'Good Good Study'` 。当然字符串里也可以只有一个字符，比如 `'a'` 。

有一种表示「是」或「否」的类型，叫做**布尔型**。它的值叫布尔值，只有 `True` 和 `False` 两种取值。这就好比考试时的判断题，结果只能二选一，要么「是」要么「否」。

另外还有一种很特别的类型：**None 型**，表示什么都没有，它就一个取值 `None` 。

> 说明：为了不增加大家的记忆负担，这里只介绍这五种基本数据类型，后续的我们慢慢掌握。

考大家一个问题，在代码中 `1000` 和 `'1000'` 是相同的东西吗？答案是不同，一个是数字，一个是字符串，数据类型不同。

数值运算

对于整数型和浮点型，因为它们都被用来表示数值，理所应当这二者可以做数值运算，也就是加减乘除等操作。

我们进入 Python 解释器交互模式中，输入代码试验一下这些数值运算：

- 加法

```
33+725
```

```
>>> 33+725
758
```

- 减法

```
12-24
```

```
-12
```

- 乘法

```
8*12.5
```

```
>>> 8*12.5
100.0
```

- 除法

```
1/3
```

```
>>> 1/3
0.3333333333333333
```

- 除余

```
10%3
```

```
>>> 10%3
1
```

可以看到，数值的加（ + ）、减（ - ）、乘（ * ）、除（ / ）、除余（ % ）都可以被计算。这些操作也是多种程序语言所通用的，除此之外 Python 还内置了次方运算（ ** ）和整除（ // ）：

- 次方

```
2**3
```

```
>>> 2**3
8
```

- 整除

```
9//2
```

这恐怕是 Python 的最简单的用法了——当作计算器！

> 说明：通常我们为了美观，会在上面的运算符号的左右各加上一个空格，如 `12 - 24`，`2 ** 3`。
>
> 之后的代码示例中我们会添加空格。

比较运算

整数型和浮点型除了数值运算外，还可以做比较运算，也就是比较两个数值的大小。比较的结果是布尔值。如：

```
2 > 3
```

```
>>> 2 > 3
False
```

```
2 <= 3
```

```
>>> 2 <= 3
True
```

```
2 == 3
```

```
>>> 2 == 3
False
```

比较运算的运算符可以是大于（`>`），小于（`<`），大于等于（`>=`），小于等于（`<=`），等于（`==`），不等于（`!=`）。其写法与数学中的比较运算很相似，但不同的是「等于」和「不等于」，尤其注意「等于」是用两个等号 `==` 表示。

变量和赋值

刚才我们学习了数值运算，那我们现在来算算一周有多少秒，一年有多少秒。

首先我们不难得出一天有 `60 * 60 * 24` 秒。我们可以暂时把这个结果用某种方式记录下来，以便后续使用。用什么方式记录呢？我们可以使用变量。

器。

创建变量的动作我们称之为**定义变量**。如下是定义变量的方法：

```
seconds_per_day = 60 * 60 * 24
```

在这里我们起了个名字 `seconds_per_day`，并且通过符号 `=` 把 `60 * 60 * 24` 的计算结果给了它。`seconds_per_day` 这个名字就是我们所定义的变量，它的值（也就是其背后的数据）是 `60 * 60 * 24` 的实际运算结果。也就是说我们将一天的秒数 `60 * 60 * 24` 保存在了变量 `seconds_per_day` 中。

等号（ `=` ）在代码中是**赋值**的意思，表示将 `=` 右边的值赋予 `=` 左边的变量。注意赋值用等号 `=` 表示，而「等于」用 `==` （连续两个等号）表示。

执行刚才的代码后，紧接着输入 `seconds_per_day` 可以看到这个变量的值：

```
>>> seconds_per_day
86400
```

回到「一周有多少秒」的问题上去。我们有了表示一天的秒数的 `seconds_per_day` 变量，那我们的程序就可以这样写下去：

```
seconds_per_day * 7
```

```
>>> seconds_per_day * 7
604800
```

一天的秒数乘以七（天），最终结果是 `604800`，没有任何问题。

刚才的完整连贯代码是：

```
seconds_per_day = 60 * 60 * 24
seconds_per_day * 7
```

变量的好处

你可能会说「一周的秒数，直接计算 `60 * 60 * 24 * 7` 不就好了，也用不着使用变量」？是的，有时确实可以不使用变量。但使用变量有一个好处，那就是可以暂存一个中间结果，方便之后去重复利用它。

比如我们现在还想要再算一下「一年有多少秒」，因为前面已经算好了一天的秒数 `seconds_per_day`，所以可以直接拿来利用：

```
>>> seconds_per_day * 365
31536000
```

除此之外变量的好处还有，你可以通过妥当的变量名字来改善程序的可读性（阅读的容易程度）。比如我们在代码里写下 `60 * 60 * 24`，别人（包括未来的你自己）在阅读时很难一下子理解这串运算表示什么。但是如果这样写呢： `seconds_per_day = 60 * 60 * 24` 。噢，原来是指一天的秒数。

用赋值更新变量

前面内容中的变量是在定义的时候被赋值的，其实变量被定义后也可以反复给这个变量赋予新的值，这样变量中的数据就被更新了。如：

```
>>> day = 1
>>> day
1
>>> day = 2
>>> day
2
>>> day = 3
>>> day
3
```

变量和数据类型的关系

变量用来保存数据，而数据类型用来指明数据的种类。

刚才我们使用了 `seconds_per_day = 60 * 60 * 24` 语句来定义变量 `seconds_per_day`，并将它赋值为 `60 * 60 * 24` 。因为变量 `seconds_per_day` 中保存的是个整数型的值，所以我们说 `seconds_per_day` 是个整数型（的）变量。

总结

数据类型

这个章节中我们提到的 Python 基础数据类型有：

类型	表示	取值示例
整数型	整数	`-59` ， `100`
浮点型	小数	`-3.5` ， `0.01`
字符串	文本	`'哼哼哈嘿'` ， `'Good Good Study'`
布尔型	是与非	`True` ， `False`

None 型	什么都没有	None

Python 中的数据类型不止这些，之后会渐渐涉及，表格中的这些类型也会在之后被应用到。

数值运算

数值运算的符号有：

符号	含义	示例
+	加法	1 + 1
-	减法	2 - 3
*	乘法	4 * 5
/	除法	6 / 7
%	取余	8 % 9
**	次方	2 ** 3 （2 的 3 次方）
//	整除	5 // 4

数值比较

数值比较的符号有：

符号	含义
>	大于
<	小于
>=	大于等于
<=	小于等于
==	等于
!=	不等于

上面的内容看起来罗列了很多，但其实不会带来记忆负担。数值运算和数值比较与数学上的概念和符号大致相同，略有区别而已。

变量和赋值

我们通过以下形式来定义变量和赋值：

```
变量名 = 数据值
```

> 多语言比较：
>
> 「多语言比较」这部分内容，是为让大家了解本章节所介绍的语言基本特性在其它语言中是如何表达的。大家可以了解体会它们之间的相识之处。

整数型根据长度的不同分为：byte（1 字节）、short（2 字节）、int（4 字节）、long（8 字节），浮点型分为 float（4 字节）、double（8 字节）。其它语言也有一些类似。C/C++ 中的整数型有「有无符号」之分（如 unsigned int 表示无符号的 int 型，也就是说这只能表示 0 和正数，不能表示负数）。

Java 定义变量并初始化：

```
int yearDays = 365
```

C/C++ 定义变量并初始化：

```
int yearDays = 365
```

把 C 和 C++ 合并称为 C/C++，是因为 C++ 基本上是 C 的强大很多的超集，虽然 C++ 严格来说不是 100% 兼容 C，但几乎是兼容的。

Go 语言定义变量并初始化：

```
var yearDays int = 365
```

Go 语言中的变量定义需要加上关键字 var，且数据类型（这里是 int ）放在变量名后面。或者采用另一种写法：

```
yearDays := 365
```

这种写法不但可以省略关键字 var 还可以省略数据类型，数据类型可直接由编译器推导出来。

以上语言在变量定义后，都可通过下述语句再次赋值：

```
yearDays = 366
```

← 04 如何运行 Python 代码？

06 一串数据怎么存—列表和字符串 →

精选留言 5

欢迎在这里发表留言，作者筛选后可公开显示

K21vin

Python的None和JavaScript的Null是同一个概念吗？

0 回复 2020-01-16

1 回复 2019-10-29

老韩Linux

可以发一个完整的Python学习路线吗？ 可以开发完整的项目，从前端到后台。

0 回复 2019-10-23

广东靓仔

请问现在更新完了吗?

0 回复 2019-10-06

DongHj 回复 **广东靓仔**

同学你好，现在专栏还没有更新完成哦

回复 2019-10-10 13:59:25

Just90

要是可以和 Javascript 比较下就好了，因为 Javascrpt 和 python 同样作为解释型语言，对比之后记忆更深，并且使用 JavaScript 的人也非常多

1 回复 2019-09-29

千学不如一看，千看不如一练