数据类型和变量

廖雪峰
资深软件开发工程师，业余马拉松选手。

数据类型

计算机顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数值。但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。在Python中，能够直接处理的数据类型有以下几种：

整数

Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样，例如：`1`，`100`，`-8080`，`0`，等等。

计算机由于使用二进制，所以，有时候用十六进制表示整数比较方便，十六进制用`0x`前缀和0-9，a-f表示，例如：`0xff00`，`0xa5b4c3d2`，等等。

对于很大的数，例如`10000000000`，很难数清楚0的个数。Python允许在数字中间以`_`分隔，因此，写成`10_000_000_000`和`10000000000`是完全一样的。十六进制数也可以写成`0xa1b2_c3d4`。

浮点数

浮点数也就是小数，之所以称为浮点数，是因为按照科学记数法表示时，一个浮点数的小数点位置是可变的，比如，1.23x10^9和12.3x10^8是完全相等的。浮点数可以用数学写法，如`1.23`，`3.14`，`-9.01`，等等。但是对于很大或很小的浮点数，就必须用科学计数法表示，把10用e替代，1.23x10^9就是`1.23e9`，或者`12.3e8`，0.000012可以写成`1.2e-5`，等等。

整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的（除法难道也是精确的？是的！），而浮点数运算则可能会有四舍五入的误差。

字符串

字符串是以单引号`'`或双引号`"`括起来的任意文本，比如`'abc'`，`"xyz"`等等。请注意，`''`或`""`本身只是一种表示方式，不是字符串的一部分，因此，字符串`'abc'`只有`a`，`b`，`c`这3个字符。如果`'`本身也是一个字符，那就可以用`""`括起来，比如`"I'm OK"`包含的字符是`I`，`'`，`m`，空格，`O`，`K`这6个字符。

如果字符串内部既包含`'`又包含`"`怎么办？可以用转义字符`\`来标识，比如：

```
'I\'m \"OK\"!'
```

表示的字符串内容是：

```
I'm "OK"!
```

转义字符`\`可以转义很多字符，比如`\n`表示换行，`\t`表示制表符，字符`\`本身也要转义，所以`\\`表示的字符就是`\`，可以在Python的交互式命令行用`print()`打印字符串看看：

```
>>> print('I\'m ok.')
I'm ok.
>>> print('I\'m learning\nPython.')
I'm learning
Python.
>>> print('\\\n\\')
\
\
```

如果字符串里面有很多字符都需要转义，就需要加很多`\`，为了简化，Python还允许用`r''`表示`''`内部的字符串默认不转义，可以自己试试：

```
>>> print('\\\t\\')
\       \
```

```
>>> print(r'\\\t\\')
\\\t\\
```

如果字符串内部有很多换行，用 `\n` 写在一行里不好阅读，为了简化，Python允许用 `'''...'''` 的格式表示多行内容，可以自己试试：

```
>>> print('''line1
... line2
... line3''')
line1
line2
```


上面是在交互式命令行内输入，注意在输入多行内容时，提示符由 `>>>` 变为 `...`，提示你可以接着上一行输入，注意 `...` 是提示符，不是代码的一部分：

```
Windows PowerShell                                        - □ x
>>> print('''line1
... line2
... line3''')
line1
line2
line3

>>>
```

当输入完结束符 ``` 和括号) 后，执行该语句并打印结果。

如果写成程序并存为 `.py` 文件，就是：

```
print('''line1
line2
line3''')
```

多行字符串 `'''...'''` 还可以在前面加上 `r` 使用，请自行测试：

```
print(r'''hello,\n
world''')
```

布尔值

布尔值和布尔代数的表示完全一致，一个布尔值只有 `True`、`False` 两种值，要么是 `True`，要么是 `False`，在Python中，可以直接用 `True`、`False` 表示布尔值（请注意大小写），也可以通过布尔运算计算出来：

```
>>> True
True
>>> False
False
>>> 3 > 2
True
>>> 3 > 5
False
```

布尔值可以用 `and`、`or` 和 `not` 运算。

`and` 运算是与运算，只有所有都为 `True`，`and` 运算结果才是 `True`：

```
>>> True and True
True
>>> True and False
False
>>> False and False
False
>>> 5 > 3 and 3 > 1
True
```

`or` 运算是或运算，只要其中有一个为 `True`，`or` 运算结果就是 `True`：

```
>>> True or True
True
```

```
>>> True or False
True
>>> False or False
False
>>> 5 > 3 or 1 > 3
True
```

`not` 运算是非运算，它是一个单目运算符，把 `True` 变成 `False`， `False` 变成 `True`：

```
>>> not True
False
>>> not False
True
>>> not 1 > 2
True
```

布尔值经常用在条件判断中，比如：

```
if age >= 18:
    print('adult')
else:
    print('teenager')
```

空值

空值是Python里一个特殊的值，用 `None` 表示。 `None` 不能理解为 `0` ，因为 `0` 是有意义的，而 `None` 是一个特殊的空值。

此外，Python还提供了列表、字典等多种数据类型，还允许创建自定义数据类型，我们后面会继续讲到。

变量

变量的概念基本上和初中代数的方程变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。

变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和 `_` 的组合，且不能用数字开头，比如：

```
a = 1
```

变量 `a` 是一个整数。

```
t_007 = 'T007'
```

变量 `t_007` 是一个字符串。

```
Answer = True
```

变量 `Answer` 是一个布尔值 `True` 。

在Python中，等号 `=` 是赋值语句，可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量，例如：

```
a = 123 # a是整数
print(a)
a = 'ABC' # a变为字符串
print(a)
```

这种变量本身类型不固定的语言称之为***动态语言***，与之对应的是***静态语言***。静态语言在定义变量时必须指定变量类型，如果赋值的时候类型不匹配，就会报错。例如Java是静态语言，赋值语句如下（// 表示注释）：

```
int a = 123; // a是整数类型变量
a = "ABC"; // 错误：不能把字符串赋给整型变量
```

和静态语言相比，动态语言更灵活，就是这个原因。

请不要把赋值语句的等号等同于数学的等号。比如下面的代码：

```
x = 10
x = x + 2
```

如果从数学上理解 `x = x + 2` 那无论如何是不成立的，在程序中，赋值语句先计算右侧的表达式 `x + 2`，得到结果 `12`，再赋给变量 `x`。由于 `x` 之前的值是 `10`，重新赋值后，`x` 的值变成 `12`。

最后，理解变量在计算机内存中的表示也非常重要。当我们写：

```
a = 'ABC'
```

时，Python解释器干了两件事情：

1. 在内存中创建了一个 `'ABC'` 的字符串；
2. 在内存中创建了一个名为 `a` 的变量，并把它指向 `'ABC'`。

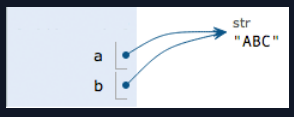
也可以把一个变量 `a` 赋值给另一个变量 `b`，这个操作实际上是把变量 `b` 指向变量 `a` 所指向的数据，例如下面的代码：

```
a = 'ABC'
b = a
a = 'XYZ'
print(b)
```

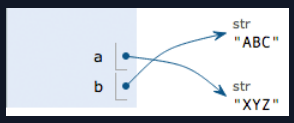
最后一行打印出变量 `b` 的内容到底是 `'ABC'` 呢还是 `'XYZ'`？如果从数学意义上理解，就会错误地得出 `b` 和 `a` 相同，也应该是 `'XYZ'`，但实际上 `b` 的值是 `'ABC'`，让我们一行一行地执行代码，就可以看到到底发生了什么事：

执行 `a = 'ABC'`，解释器创建了字符串 `'ABC'` 和变量 `a`，并把 `a` 指向 `'ABC'`：

执行 `b = a`，解释器创建了变量 `b`，并把 `b` 指向 `a` 指向的字符串 `'ABC'`：



执行 `a = 'XYZ'`，解释器创建了字符串'XYZ'，并把 `a` 的指向改为 `'XYZ'`，但 `b` 并没有更改：



所以，最后打印变量 `b` 的结果自然是 `'ABC'` 了。

常量

所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量：

```
PI = 3.14159265359
```

但事实上 `PI` 仍然是一个变量，Python根本没有任何机制保证 `PI` 不会被改变，所以，用全部大写的变量名表示常量只是一个习惯上的用法，如果你一定要改变变量 `PI` 的值，也没人能拦住你。

最后解释一下整数的除法为什么也是精确的。在Python中，有两种除法，一种除法是 `/`：

```
>>> 10 / 3
3.3333333333333335
```

`/` 除法计算结果是浮点数，即使是两个整数恰好整除，结果也是浮点数：

```
>>> 9 / 3
3.0
```

还有一种除法是 `//`，称为地板除，两个整数的除法仍然是整数：

```
>>> 10 // 3
3
```

你没有看错，整数的地板除 `//` 永远是整数，即使除不尽。要做结果为浮点数的除法，使用 `/` 就可以。

因为 `//` 除法只取结果的整数部分，所以Python还提供一个余数运算，可以得到两个整数相除的余数：

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

无论整数做 `//` 除法还是取余数，结果永远是整数，所以，整数运算的结果永远是精确的。

练习

请打印出以下变量的值：

```
n = 123
f = 456.789
s1 = 'Hello, world'
s2 = 'Hello, \'Adam\''
s3 = r'Hello, "Bart"'
s4 = r'''Hello,
Bob!'''

print(???)
```

小结

Python支持多种数据类型，在计算机内部，可以把任何数据都看成一个“对象”，而变量就是在程序中用来指向这些数据对象的，对变量赋值就是把数据和变量给关联起来。

对变量赋值 `x = y` 是把变量 `x` 指向真正的对象，该对象是变量 `y` 所指向的。随后对变量 `y` 的赋值 ***不影响*** 变量 `x` 的指向。

注意：Python的整数没有大小限制，而某些语言的整数根据其存储长度是有大小限制的，例如Java对32位整数的范围限制在 `-2147483648` - `2147483647` 。

Python的浮点数也没有大小限制，但是超出一定范围就直接表示为 `inf` （无限大）。

« Python基础 字符串和编码 »

Comments

Comments loaded. To post a comment, please Sign In

. @ 2026/1/1 15:18:34

n = 123 f = 456.789 s1 = 'Hello, world' s2 = 'Hello, 'Adam'' s3 = r'Hello, "Bart"' s4 = r'''Hello, Bob!'''

print(n , f, s1 , s2, s3, s4, sep='\n')

123 456.789 Hello, world Hello, 'Adam' Hello, "Bart" Hello, Bob!

仰望星空 @ 2025/12/30 09:01:46

n = 123 f = 456.789 s1 = 'Hello,world' s2 = 'Hello,'Adam'' s3 = r'hello,"Bart"' s4 = r'''Hello, Bob!'''
print(n) print(f) print(s1) print(s2) print(s3) print(s4)

DragonCarbon @ 2025/12/29 03:07:45

n=123

> *f=456.789 s1='hello,world' s2='hello,'adam'' s3=r'hello,"bart"' s4=r'''hello, ... bob!'''*
> *print(n,f,s1,s2,s3,s4) 123 456.789 hello,world hello,'adam' hello,"bart" hello, bob!*

张肥肥 @ 2025/12/28 03:39:08

> *print(r'''hello,\n ... world''') hello,\n world*

张肥肥 @ 2025/12/28 03:56:51

> *print(n) 123 print(f) 456.789 print(s1) hello,world print(s2) hello,'Adam' print(s3) hello,"Bart" print(s4) hello, Bob!*

breeze @ 2025/12/24 10:07:50

> *n= 123 f= 456.789 s1= 'Hello, world' s2= 'Hellow, 'Adam'' S3= r'Hellow, "Bart"' s4= r'''Hello, ... Bob!'''*
>
> *print(n) 123 print(f) 456.789 print(s1) Hello, world print(s2) Hellow, 'Adam' print(s3) Traceback (most recent call last): File "<python-input-42>", line 1, in <module> print(s3) ^^ NameError: name 's3' is not defined. Did you mean: 'S3'? print(S3) Hellow, "Bart" print(s4) Hello, Bob!*

breeze @ 2025/12/24 04:09:39

打卡第二天： age=76

> *if age>= 18: ... print('adult') ... else: ... print('teenager') ...*
> *adult*

▵→奥霍斯德尔萨拉多 @ 2025/12/14 02:27:48

打卡

```
>>> n=123
>>> f=456.789
>>> s1='Hello,World'
>>> s2='Hello,\'Adam\''
>>> S3=r'Hello,"Bart"'
>>> s4=r'''Hello,
... Bob!'''
>>> Print(n)
Traceback (most recent call last):
  File "<python-input-15>", line 1, in <module>
    Print(n)
    ^^^^^
NameError: name 'Print' is not defined. Did you mean: 'print'?
>>> print(n)
```

Read More ▼

▵→奥霍斯德尔萨拉多 @ 2025/12/14 03:02:49

有点搞不太清, 像a2和a3里遇到'''，前面加不加r似乎关系不大?

```
>>> a1='''Hello,
... World'''
>>> print(a1)
Hello,
World
>>> a2='''Hello,/n
... World!'''
>>> print(a2)
Hello,/n
World!
>>> a3=r'''Hello,//
... World!'''
>>> print(a3)
```

```
Hello,//
World!
```

Carry行 @ 2025/12/16 03:01:24

r'"里面不会转义，你写的a2，a3里面/要变成\才是转义

Delete @ 2025/10/6 01:08:14

请观察下面的代码，写出它们分别的输出结果：

1

print("C:\new\test\data")

输出：C:\new\test\data

2

print(r"C:\new\test\data")

输出：C:\new\test\data

3

Read More ▼

大汪汪 @ 2025/10/18 03:30:14

1的答案不对

国王KING专心工作 @ 2025/10/25 02:53:35

1的运行结果是 C: ew est\data

吃西瓜不吐籽 @ 2025/10/26 11:36:20

只有3对吧123\456\789 \会吞一个字母

青水王 @ 2025/10/31 03:11:45

除了1都对

▵→奥霍斯德尔萨拉多 @ 2025/12/14 02:14:47

1不对吧：

```
>>> print("C:\new\test\data")
<python-input-0>:1: SyntaxWarning: invalid escape sequence '\d'
C:
ew      est\data
>>> print(r"C:\new\test\data")
C:\new\test\data
>>> print("Hello\nWorld")
Hello
World
>>> print(r"Hello\nWorld")
Hello\nWorld
```

Chaser @ 2025/12/7 09:46:41

print(len('%12d-%%%.2f' % (1, 3.1415926)))

Delete @ 2025/10/6 01:50:31

请写出以下代码的输出结果，并解释原因： import re

1

print("A\tB" == "A B")

2

print(len("\t") == 4 or len("\t") == 1)

3

s = r"\t\n\" print(len(s) == 6 and "\n" in s)

4

Read More ▼

躺着就好。。。 @ 2025/11/20 04:09:47

第三题 s赋值的时候少了引号吧

pure @ 2025/11/28 23:17:29

第三行 s = r"\t\n\"，这个是语法错误无法给s赋值，最后一个\把"双引号，转义掉了，结构不够完整，syntax error :unterminated string literal,字符串字面量没有终止少了"

Tiger paw @ 2025/11/23 06:01:35

s4 = r'''Hello, Bob!'''这里的r有没有是不是都一样?

汪洋浩渺 @ 2025/11/23 22:27:26

s4 这个字符串 中的r 是的 这个r的意思是代码原始字符串 如果字符串中包含转移符\ 不会进行转义直接输出原始字符串 如 s2 = 'Hello, 'Adam'' 前面不加r 输出的是 Hello, 'Adam' 如果前面加上r s2 = r'Hello, \'Adam\'' 输出的是 Hello, \'Adam\'

多云的季节 @ 2025/11/13 21:39:28

print(r'\\t\')，t后面加2个斜杠可以，为啥加3个斜杠就报错SyntaxError: unterminated string literal (detected at line 1)

薛盼 @ 2025/11/16 02:08:52

报错的原因是\' 。 文中 (Python还允许用r''表示''内部的字符串默认不转义) 这句话不准确，其中唯一保留的转义就是引号的转义（但也仅限于编译，实际其实是不会转义的），所以\' 被解释为："这是一个普通的单引号字符"，因为判定为没用引号收回，进而报错。 print(r'\'') 这样就不会报错。

The Forgotten One @ 2025/11/17 10:44:08

研究好半天终于搞明白了-.- age = input('enter your age:') if int(age) >= 18: print('adult') else: print('teenager')

1234 @ 2025/11/18 22:23:57

还是静态语言好，把c学一遍再来看python会容易理解一些

炬火 @ 2025/11/1 06:56:42

打卡第二天

头发不要乱了 @ 2025/10/27 04:34:06

Hello, 'Adam' Hello, "Bart"

Hello, Bob!

吃西瓜不吐籽 @ 2025/10/26 11:38:29

有没有讨论群刚学习带带一起学习

吃西瓜不吐籽 @ 2025/10/26 11:37:29

n=123 f=456.789 s1='hello world' s2='hello,\'Adam\'' s3=r'hello,"Bart"' s4=r'''hello, Bob! ''' print(n,f,s1,s2,s3,s4,sep='\n') 123 456.789 hello world hello,'Adam' hello,"Bart" hello, Bob!

318197375 @ 2025/10/12 23:42:43

注: 原文: 整数运算永远是精确的（除法难道也是精确的？是的！） 这里廖老师说的 整数除法也是精确的 是指结果是整数的运算(//除) 如果你把结果取值为小数(/除), 那么这里不能确保一定精确!

大汪汪 @ 2025/9/23 04:55:44

我有个问题请教：''' 可以自动换行输入，直到用另一个''' 来结束，那么''' 算是转义符吗？如果它算是转义符，r' 是取消转义功能的，那么取消 ''' 的转义为什么不应该是4个引号呢？(r' + ''') 如：print(r''''abc？我试了回车后'''仍然有效，r'没起到作用，在下一行让我继续输入字符。请高手解答，谢谢！

Moriya Kaeru @ 2025/9/25 05:01:42

同问

2333 @ 2025/9/25 08:30:32

'''应该不算转义符吧，按我的理解是 ‘，“，'''都属于引号的一种，都当成引号使用 所以试试用'''来替换r'的引号，应该会同时有r'的无视转义符效果以及'''的换行效果

```
print(r'''a\\
b\nc''')
```

得到结果 a\ b\nc 所以确实是这样，'''是特殊的'

大汪汪 @ 2025/10/6 00:56:28

谢谢，但如果我要的输出结果是 '''abc''' (abc前后各3个引号)怎么办，r' 就没法用了?

虺 @ 2025/10/9 21:43:16

我的理解里'''不仅仅只有自动换行输入的作用，本身也可以当成单引号或双引号来使用，因此不能算是转义符。python中转义符是以反斜杠()开头的字符，而'''是字符串的边界，告诉python从这里开始是一个字符串，直到遇到另一个'''为止，'''不是转义序列，他依旧是字符串的界定符，r'只影响反斜杠转义序列。

```
print('\'\'\'abc\'\'\'')
print('''''abc''''')
```

的输出结果都可以是'''abc'''。不过我也是自学刚看到这，不算高手，希望可以帮到你

虺 @ 2025/10/9 22:02:19

复查发现前面()里的()没打出来，后面应该是用双引号包裹里面的'''abc'''，做一下修改

```
print("'''abc'''")
```

AADM @ 2025/10/9 21:48:17

Python允许在数字中间以_分隔符以增强可读性，如下所示：1_000_000和1000000是一样的

浮点数支持科学计数法表示法，例如：1.23e4表示12300.0

r''表示''内部的字符串默认不转义，如print(r'\n')输出\n

#如果字符串内部有很多换行，用\n会很不方便，这时可以用''' print('''line1 line2 line3''') print('line1\nline2\nline3')

python是动态语言，变量在使用前必须赋值，变量赋值以后该变量才会被创建。 静态语言在定义变量时必须指定变量类型。