Python语言

第二周 Python基础知识（1）

陈世峰 岭南师范学院信息工程学院
chenshifeng@lingnan.edu.cn/13234075348

Python书写规范

Python的书写规范

1. Python语句
2. 缩进
3. 注释

Python的语句

1. 通常一行书写一条语句

如果一行内写多条语句，要求使用分号分隔

建议每行只写一条语句，并且语句结束时不写分号。

2. 如果一条语句过长

在语句的外部加上一对圆括号来实现

使用“\”（反斜线）来实现分行书写功能。

代码块与缩进

- 缩进指在代码行前面添加空格或Tab，这样做可以使程序更有层次、更有结构感，从而使程序更易读。
- 缩进是Python语言中表明程序框架的唯一手段。
- 在Python程序中，缩进不是任意的。平级的语句行（代码块）的缩进必须相同。
- 在Python中，一般1个缩进 = 4个空格

注释

- 注释用于说明程序或语句的功能
- Python的注释分类单行注释和多行注释两种
  - 单行注释以“#”开头，可以是独立的1行，也可以附在语句的后部。
  - 多行注释可以使用三个引号（英文的单引号或双引号）作为开始和结束符号。

例如：

```
""" 多行注释开始
下面的代码根据变量 x 的值计算 y
注意代码中使用缩进表示代码块
多行注释结束
"""

x=5
if x > 100:
    y = x *5 - 1                #单行注释：x>100 是执行该语句
else:
    y = 0                       #x<=100 时执行该语句
print(y)                        #输出 y
```

注释的作用

你应该在你的程序中尽可能多地使用有用的注释，它们的作用包括：

- 解释假设
- 说明重要的决定
- 解释重要的细节
- 说明你想要解决的问题
- 说明你想要在程序中克服的问题，等等。

有一句非常有用的话叫做：**代码会告诉你怎么做，注释会告诉你为何如此。**

标识符和关键字

标识符

用户定义的、由程序使用的符号都就是标识符。

- 标识符由字母、数字和下划线“_”组成，且不能以数字开头标识符区分大小写，没有长度限制
- 区分大小写，没有长度限制
- 不能使用计算机语言中预留有特殊作用的关键字。
- 命名尽量符合见名知意的原则

Python中合法的标识符

myVar、_Variable、姓名

Python中非法标识符

2Var、vari#able、finally、stu@lnnu、my name

关键字

Python保留某些单词用做特殊用途，这些单词被称为**关键字**，也叫保留字。用户定义的标识符（变量名、方法名等）不能与关键字相同。

and	as	assert	break	class	continue
def	del	elif	else	except	False
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
None	pass	raise	return	True	try
while	with	yield			

Python的数据类型

常见数据类型

数字

字符串

布尔值

空值

列表

元组

字典

集合

数字型

常见的数字主要有为三种类型：

- 整数（Integers）：有关整数的例子即 2或者100，它们都是一个整数，即没有小数点，也没有分数的表示形式。
- 浮点数（Floats，也称为实数）：有关浮点数的例子是 3.23 或 52.3E-4。其中，E 表示 10 的幂。在这里，52.3E-4 表示 52.3 * 10-4。
- 复数（Complex）：由实部和虚部组合在一起构成的数，例如3+4j、3.1+4.1j，其中加号左边的数为实部，加号右边的为虚部，用后缀j表示

整数

- **十进制整数**，如，0、-1、9、123
- **十六进制整数**，需要16个数字0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f来表示整数，必须以0x开头，如0x10、0Xfa、0xabcdef
- **八进制整数**，只需要8个数字0、1、2、3、4、5、6、7来表示整数，必须以0o开头，如0O35、0o11
- **二进制整数**，只需要2个数字0、1来表示整数，必须以0b开头如，0B101、0b100

整数类型的数据对象不受数据位数的限制，只受可用内存大小的限制。

浮点数

浮点数又称小数

例如：1.0、1.、0.12、.123、12.345、 52.3E-4 、1.8e-5等。

其中，E 表示 10 的幂。在这里，52.3E-4 表示 52.3 * 10-4。

浮点数用64位存储，表达数据的范围为:

-1.7E+308～1.7E+308，提供大约15位的数据精度。

复数

- 数学上，可以用$a + bi$表示的复数。a和b是实数，i是虚数单位。虚数单位是二次方程式 $x^2+1=0$ 的一个解, 所以虚数单位同样可以表示 $i = \sqrt{-1}$
- 复数$a + bi$中，a称为复数的实部，b称为复数的虚部。
- python语言中，用a + bj 来表示复数，a,b分别为实部和虚部

```
>>> a = 3+4j
>>> a.real        #查看复数实部
3.0
>>> a.imag        #查看复数虚部
4.0
>>> a.conjugate() #返回共轭复数
(3-4j)
```

■三种类型存在一种逐渐“扩展”的关系：

整数 -> 浮点数 -> 复数

（整数是浮点数特例，浮点数是复数特例）

■不同数字类型之间可以进行混合运算，运算后生成结果为最宽类型

123 + 4.0 = 127.0（整数 + 浮点数 = 浮点数）

■数字类型之间相互运算所生成的结果是“更宽”的类型，基本规则是：

整数之间运算，如果数学意义上的结果是小数，结果是浮点数；

整数之间运算，如果数学意义上的结果是整数，结果是整数；

整数和浮点数混合运算，输出结果是浮点数；

整数或浮点数与复数运算，输出结果是复数。

字符串

- 用**单引号、双引号或三引号**括起来的符号系列称为字符串

单引号、双引号、三单引号、三双引号可以互相嵌套，用来表示复杂字符串

' abc' 、 ' 123' 、 '中国大连' 、 "辽宁沈阳"、'''Tom said, "Let's go"'''

- 空串表示为''或 ""

三引号'''或"""表示的字符串可以换行，支持排版较为复杂的字符串；三引号还可以在程序中表示较长的注释。例如：

```
>>> s = """abc
... 1234567890
... xyz"""
>>> s
  'abc\n1234567890\nxyz'
```

转义符

如果你希望生成一串包含单引号（'）的字符串，你应该如何指定这串字符串？

例如：你不能指定 'What's your name? '

因为这会使 Python 对于何处是字符串的开始、何处又是结束而感到困惑。所以，你必须指定这个单引号不代表这串字符串的结尾。这可以通过 **转义字符（Escape Sequence）** 来实现。Python中通过 \ 来表示一个转义字符。

你可以将字符串指定为 'What\'s your name?'

也常用于计算机中的不可见字符。不可见字符是指不能显示图形仅仅是表示某一控制功能的代码，如ASCII码中的换行、制表符、铃声等。

转义字符以“\”开头，后跟字符或数字。

常见的转义符

符号	含义	符号	含义描述
\	续行符	\n	换行
\\	反斜杠符号	\t	横向制表符
\'	单引号	\r	回车
\"	双引号	\f	换页
\a	响铃	\ooo	八进制
\b	退格	\xhh	十六进制
\0	空	\other	其它的字符

布尔值

布尔值通常用来判断条件是否成立。

Python包含两个布尔值，包含**True（逻辑真）**和**False（逻辑假）**。布尔值区分大小写，也就是说true和TRUE不能等同于True。

空值

Python有一个特殊的空值常量**None**。与0和空字符串（“”）不同，None表示什么都没有。None与任何其他的数据类型比较永远都返回False。

- 列表类型(list)
- 元组类型(tuple)
- 字典类型(dict)
- 集合类型（set）

在后续章节中分别介绍

查看数据类型

我们在程序中引入**type()**函数，该函数可以输出参数的数据类型，例如在交互模式中输入以下命令可以得到各个常量的数据类型：

```
>>> type(100)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type("Hello")
<class 'str'>
>>> type(True)
<class 'bool’>
>>> type(3+4j)
<class 'complex'>
```

常量

- **常量是内存中用于保存固定值的单元，在程序中常量的值不能发生改变。**

  我们可以先来看一看 常量 的例子，比如5和1.23 这样的数字常量，或者是如 “这是一串文本” 或 “This is a string” 这样的字符串常量。之所以称这些数据为常量，是因为我们使用的就是它 字面意义上（Literal） 的值或是内容。不管在哪种应用场景中，数字 2 总是表示它本身的意义而不可能有其他的含义，所以它就是一个常量，因为它的值不能被改变。

- Python并没有命名常量，也就是说不能像C语言那样给常量起一个名字。
- Python常量包括**数字**、**字符串**、**布尔值**和**空值**等。

变量

即内存变量，用于在程序中临时保存数据。与常量不同的是变量的值可以动态变化。

变量用标识符来命名，变量名区分大小写。Python定义变量的格式：

varName = value

varName是变量名字，value是变量的值，这个过程叫做为变量赋值，“=”被称为赋值运算符，即把“=”后面的值传递给前面的变量名。

计算机语言中的赋值是一个重要的概念，x=8，含义是将8赋予变量x；x=x+1，赋值运算的含义是将x加1之后的值再送给x，x的值是9

Python变量具有类型，变量的类型由所赋的值来决定。

Python定义了一个变量，并且该变量存储了数据，那么变量的数据类型就已经确定了，系统会自动识别变量的数据类型。

```
x=8             #x是整形数据
x="Hello"       #则x是一个字符串类型
x=None          #x是一个空类型
```

Python属于**强类型编程语言**，Python解释器会根据赋值或运算来自动推断变量类型。Python还是一种**动态类型语言**，变量的类型也是可以随时变化的。

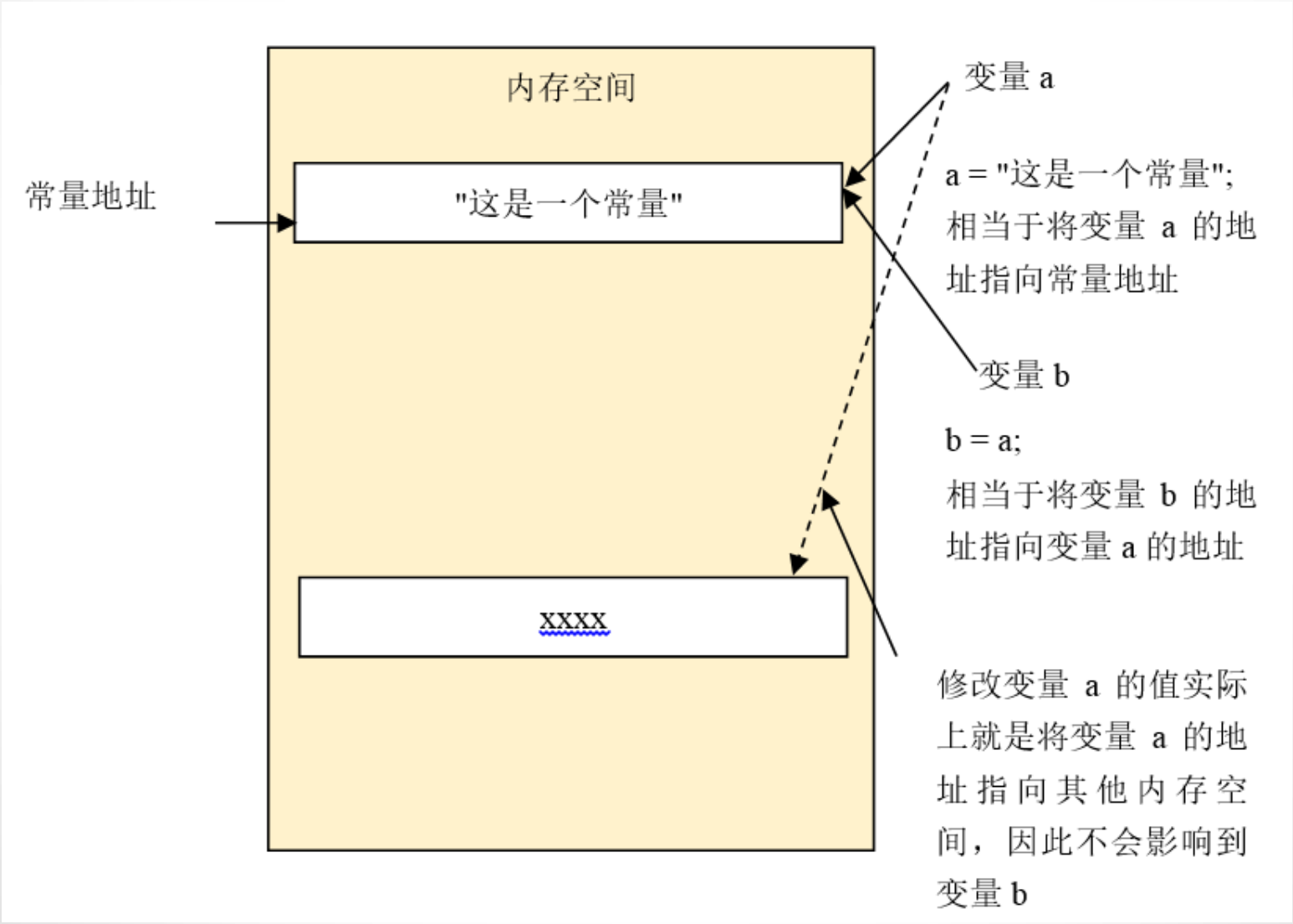
```
>>> x = 3
>>> type(x)
<class 'int'>
>>> x = 'Hello world.'
>>> type(x)
<class 'str'>
>>> x = [1,2,3]
>>> type(x)
<class 'list'>
```

```
>>> x=True
>>> type(x)
<class 'bool'>
>>> x=3+4j
>>> type(x)
<class 'complex'>
>>> x=None
>>> type(x)
<class 'NoneType'>
```

变量与地址的关系

- 在C语言中，编译器会为变量分配一个空间，当变量改变值时，改变的是这块空间中保存的值。在程序运行中，变量的地址就不能再发生改变了。
- Python不同，它的变量与C语言中的指针相似，当变量赋值时，解释器为数值开辟一块空间，而变量则指向这块空间，当变量改变值时，改变的并不是这块空间中保存的值，而是改变了变量指向的空间，使变量指向另一空间。

可以使用id()函数输出变量的地址，语法如下：id(变量名)


内存空间
常量地址
"这是一个常量"
xxxx
变量 a
a = "这是一个常量";
相当于将变量 a 的地址指向常量地址
变量 b
b = a;
相当于将变量 b 的地址指向变量 a 的地址
修改变量 a 的值实际上就是将变量 a 的地址指向其他内存空间，因此不会影响到变量 b

❖Python具有**自动内存管理功能**，对于没有任何变量指向的值，Python自动将其删除。Python会跟踪所有的值，并自动删除不再有变量指向的值。因此，Python程序员一般情况下不需要太多考虑内存管理的问题。

❖尽管如此，显式使用del命令删除不需要的值或显式关闭不再需要访问的资源，仍是一个好的习惯，同时也是一个优秀程序员的基本素养之一。

❖del x

常量与变量的数据类型转换

转换为数字

可以将字符串常量或变量转换为数字。

（1）使用**int()**函数将数字或字符串转换为整数（10进制表示），语法如下：

int(x [,base])

参数x是待转换的数字或字符串；参数base为可选参数，为参数x的进制，默认为10进制。

注意：

① x可以是数字或字符串，但是base被赋值后x只能是字符串。

② x作为字符串时必须是base类型，也就是说x变成数字时必须能用base进制表示。

例题:

int(3.14)

int(100,2)

int('23',16)

int('Pythontab’,8)

int(9.99)

3

出错，base被赋值后函数只接收字符串

35

出错，Pythontab不是8进制数

9

（2）使用float()函数将字符串或数字转换为浮点数，语法如下：

```
float(x)
```

参数x是待转换的字符串或数字。

（3）使用complex()函数将字符串或数字转换为复数，语法如下：

```
complex(re[, im])
```

re实部， im为虚部。re可以是整数、浮点数或字符串，im可以是整数或浮点数但不能为字符串。

（4）使用eval()函数计算字符串中的有效Python表达式，并返回结果，语法如下：

```
eval(str)
```

参数str是待计算的Python表达式字符串。

```
x = 4
y = "1+2"
z= ' 5 '
m= ' c '
c=6
```

```
print(float(x))
print(float(y))
print(float(z))
print(float(m))
print(eval(y))
print(eval(z))
print(eval(m))
print(complex(x))
print(complex(y))
print(complex(m))
```

运行结果为

4.0
出错
5.0
出错
3
5
6
4+0J
出错
出错

转换为字符

可以将数字常量或变量转换为字符串。

（1）使用**str()**函数将数值转换为字符串，语法如下：

str(x) 参数x是待转换的数值。　　str(65)　　#'65'

（2）使用repr()函数将对象转换为字符串显示，语法如下：

repr(obj) 参数obj是待转换的对象。　　repr('string') #"'string'"

（3）使用chr()函数将一个整数转换为对应ASCII的字符，语法如下：

chr(整数)　　chr(65)　　#'A'

（4）使用ord()函数将一个字符转换为对应的ASCII，语法如下：

ord(字符)　　ord('a’)　　#97

（5）使用hex()函数将一个整数转换为一个十六进制字符串，语法如下：

hex(整数)　　hex(8) # '0x8'

（6）使用oct()函数将一个整数转换为一个八进制字符串，语法如下：

oct(整数)　　oct(8) # '0o10'

（7）使用bin()函数将一个整数转换为一个二进制字符串，语法如下：

bin(整数)　　bin(8) # '0b1000'

让编程改变世界

Change the world by program