

## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

# PEP8

Python Enhancement Proposal #8, 是Python增强提案 (Python Enhancement Proposals) 中的第8号, 缩写为PEP 8, 它是针对Python代码格式而编订的风格指南。本节将介绍PEP8的部分内容, 例如变量、函数和方法、属性和类、模块和包等关键因素的命名规则, 以及运算符等相关规定, 并强烈建议读者在编写Python程序源代码时, 应该遵循该指南, 可以使项目更利于多人协作, 并且后续的维护工作也将变得更容易。

### 2.1.1 变量

**全局变量**使用英文大写，单词之间加下划线：

`SCHOOL_NAME = 'Tsinghua University'`      `#学校名称`

全局变量一般只在模块内有效，实现方法：使用 `__all__` 机制或添加一个前置下划线。

**私有变量**使用英文小写和一个前导下划线：

`_student_name`

**内置变量**使用英文小写，两个前导下划线和两个后置下划线：

`__maker__`

**一般变量**使用英文小写，单词之间加下划线：

`class_name`

### 2.1.1 变量

变量命名规则：

- 名称第一字符为英文字母或者下划线
- 名称第一字符后可以使用英文字母、下划线和数字
- 名称不能使用python的关键字或保留字符
- 名称区分大小写，单词与单词之间使用下划线连接

关键字与保留字的异同：

- 从字面含义上理解，**保留字**是语言中已经定义过的字，使用者不能再将这些字作为变量名或过程名使用。
- 而**关键字**则指在语言中有特定含义，成为语法中一部分的那些字。**关键字**，一定是**保留字**。

### 2.1.1 变量

Python 3的关键字和保留字，可以从shell命令行中查看，方法如下：

```
>>> import keyword as kw          #导入keyword模块
>>> kw.kwlist                     # 访问变量kwlist显示保留关键字列表
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'i
mport', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'retur
n', 'try', 'while', 'with', 'yield']
```

### 2.1.2 函数和方法

函数名是英文小写，单词之间加下划线，提高可读性。

函数名不能与保留关键字冲突，如果冲突，最好在函数名后面添加一个后置下划线，不要使用缩写或单词拆减，最好的方式是使用近义词代替。

实例方法(Instance method)的第一个参数总是使用self，代表当前对象。

类方法 (Class method) 的第一个参数总是使用cls，表示这个类本身。

### 2.1.3 属性和类

类的命名遵循首字母大写 (CapWords) 的方式，大部分内置的名字都是单个单词（或两个），首字母大写方式只适用于异常名称和内置的常量，模块内部使用的类采用添加前导下划线的方式。

类的属性（方法和变量）命名使用全部小写的方式，可以使用下划线。公有属性不应该有前导下划线，如果公有属性与保留关键字发生冲突，在属性名后添加后置下划线。对于简单的公有数据属性，最好是暴露属性名，不使用复杂的访问属性或修改属性的方法。

如果该类是为了被继承，有不让子类使用的属性，给属性命名时可以给它们加上双前导下划线，不要加后置下划线。

为避免与子类属性命名冲突，在类的一些属性前，前缀两条下划线。比如：类Faa中声明[\\_a](#)，访问时，只能通过[Faa.\\_Faa\\_a](#)，以避免歧义。

### 2.1.4 模块和包

模块命名要使用简短的小写英文的方式，可使用下划线来提高可读性。

包的命名和模块命名类似，但不推荐使用下划线。

模块名对应到文件名，有些模块底层使用C或C++ 书写，并有对应的高层Python模块，C/C++模块名有一前置下划线。



### 2.1.5 规定

下列运算符前后都需使用一个空格：

`= + - < > == >= <== and or not`

下列运算符前后不使用空格：

`* / **` (乘方)

更多PEP8规则，请参考 “Python代码风格指南：PEP8”  
<https://www.cnblogs.com/Yiutto/p/5636193.html>。

## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

### 2.2.1 变量

Python语言是面向对象（Object-oriented）的编程语言，可以说在Python中**一切皆对象**。对象是某类型具体实例中的某一个，每个对象都有身份、类型和值。

- 身份（Identity）与对象都是唯一对应关系，每一个对象的身份产生后就都是独一无二的，并无法改变。对象的ID是对象在内存中获取的一段地址的标识。
- 类型（Type）是决定对象将以哪种数据类型进行存储。
- 值（Value）存储对象的数据，某些情况下可以修改值，某些对象声明值过后就不可以修改了。

### 2.2.1 变量

指向对象的值的名称就是变量，也就是一种标识符，是对内存中的存储位置的命名。

对于不同的对象，有不同的类型，得到的内存地址也不一样，通过对得到的地址进行命名得到变量名称，我们将数据存入变量，为存储的数据设置不同的数据结构。

变量的值是在不断的动态变化的，Python的变量可以不声明直接赋值使用。由于Python采用**动态类型 (Dynamic Type)**，变量可以根据赋值类型决定变量的数据类型。

在Python中，变量使用等号赋值以后会被创建，定义完成后可以直接使用。

### 2.2.2 变量命名规则

Python对编码格式要求严格，对变量命名建议遵守本章2.1.1关于变量命名规则部分。

这里需要说明的是，如果在IDLE或PyCharm中编写源代码使用了Python的关键字或保留字（参见本章2.1.1），会有相应的提示，或以颜色加以区分。

### 2.2.3 数据类型

Python有可以自由的改变变量的数据类型的**动态类型**和变量事先说明的**静态类型**，特定类型是数值数据存入相应的数据类型的变量中，相比下，动态数据类型更加灵活。

变量的数据类型有多种类型，Python3 中有六个标准的数据类型：

- Numbers（数字类型）
- Strings（字符串类型）
- Lists（列表类型）
- Tuples（元组类型）
- Dictionaries（字典类型）
- Sets（集合类型）

### 2.2.3 数据类型

Python内置的数字类型有**整型 (Integers)**、**浮点型 (Floating point numbers)**和**复数 (Complex numbers)**三种，作为可以进行算术运算等的数据类型。

#### 1、整型 (Integers)

整数类型 (int) 简称为整型，表示整数，包括正负的整数，如：0110、-123、123456789。

Python的整型是长整型，能表达的数的范围是无限的，内存足够大，就能表示足够多的数。在使用整型的数还包括其它进制：

0b开始的是二进制 (binary) , 例如0b0110----对应十进制的6

0o开始的是八进制 (octonary) , 例如0o26----对应十进制的22

0x开始的十六进制 (hexadecimal) ,例如0x1F----对应十进制的31

进制之间可以使用函数进行转换，使用时需要注意数值符合进制。

### 举例

- **整型转换为字符型数值**

将数值16转换为2/8/10/16进制的字符串数值  
分别使用`bin()`, `oct()`, `str()`, `hex()`函数完成

```
1 >>> bin(16)
2 '0b10000'
3 >>> oct(16)
4 '020'
5 >>> str(16)
6 '16'
7 >>> hex(16)
8 '0x10'
```

- **字符型数值转换为整型**

将2/8/10/16进制的字符串转换为数值16  
使用`int()`函数完成

```
1 >>> int("0b10000",2)
2 16
3 >>> int("0o020",8)
4 16
5 >>> int("16",10)
6 16
7 >>> int("0x10",16)
8 16
```

- **字符型数值互转**

```
1 >>> bin(int("16")) # 10进制字符型数值转2进制字符型数值
2 '0b10000'
3 >>> oct(int("10",16)) # 16进制字符型数值转8进制字符型数值
4 '020'
5 >>> bin(int("020",8)) # 8进制字符型数值转2进制字符型数值
6 '0b10000'
```



### 2.2.3 数据类型

#### 2、布尔型 (Booleans)

布尔值是整型 (Integers) 的子类，用于逻辑判断真 (True) 或假 (False)，用数值1和0分别代表常量True和False。

在Python语言中，False可以是数值为0、对象为None或者是序列中的空字符串、空列表、空元组。

### 2.2.3 数据类型

#### 3、浮点型 (Float)

浮点型 (Float) 是含有小数的数值，用于实数的表示，由正负号、数字和小数点组成，正号可以省略，如：-3.0、0.13、7.18。Python的浮点型执行IEEE754双精度标准，8个字节一个浮点，范围-1.8308~+1.8308的数均可以表示。

浮点型方法

静态方法**fromhex(s)**: 十六进制浮点数转换为十进制数，例如：

```
x=float.fromhex("0x1F");
```

实例方法**hex()**: 以字符串形式返回十六进制的浮点数；例如：**print(x.hex())**

实例方法**is\_integer()**: 判断是否为小数，小数非零返回False，为零返回True，转换为布尔值。例如：

```
>>> x = 7.0
```

```
>>> x.is_integer()
```

```
True
```

### 2.2.3 数据类型

#### 4、复数型 (Complex)

复数类型 (Complex) 由实数和虚数组成，用于复数的表示，虚数部分需加上**j或J**，如：-1j、0j，1.0j。Python的复数类型是其他语言一般没有的。

```
>>> z=2+4j
```

```
>>> z
```

```
(2+4j)
```

```
>>> type(z)
```

```
<class 'complex' >
```

```
>>> z.real
```

```
2.0
```

```
>>> z.imag
```

```
4.0
```

```
>>> z.conjugate()
```

```
(2-4j)
```

```
>>> abs(z)
```

```
4.47213595499958
```

### 2.2.3 数据类型

#### 5、字符串类型 (Strings)

字符串 (Strings)，用于**Unicode字符序列**，使用一对单引号、双引号和使用三对单引号或者双引号引起来的字符就是字符串，如'hello world'、"20180520"、'''hello'''、"""happy!"""。

严格地说，在 Python 中的字符串是一种对象类型，使用 **str** 表示，通常单引号"或者双引号""包裹起来。

字符串和前面讲过的数字一样，都是对象的类型，或者说都是值。

- 如果不想让反斜杠发生转义，可以在字符串前面加个**r表示原始字符串**，例如：  
`print(r"\t\nabc")`---输出  
`\t\nabc`
- 加号+是字符串的连接符，例如`print("Hello"+"World")` ---输出  
`Hello World`
- 星号\*表示复制当前的字符串，紧跟的数字为复制的次数，例如：  
`print("Hello"*2)` ---输出  
`HelloHello`

### 2.2.4 type() 函数

type()函数是内建的用来查看变量类型的函数，调用它可以简单的查看数据类型，基本用法：

type(对象)

对象即为需要查看类型的对象或数据，通过返回值返回相应的类型，例：

```
>>> type(1)      #查看数值1的数据类型
```

```
<class 'int'>     #返回结果
```

```
>>> type("int")  #查看" int" 的数据类型
```

```
<class 'str'>     #返回结果
```

### 2.2.5 数据类型的转换

转换为整型int类型:

`int(x [,base])`

`int()`函数将x转换为一个整数，x为字符串或数字，base进制数，默认为十进制。

```
>>> int(100.1)      #浮点转整数
```

```
100                 #返回结果
```

```
>>> int('01010101', 2)  #二进制转换整数
```

```
85                  #返回结果
```

### 2.2.5 数据类型的转换

转换为浮点型float类型：

`float(x)`

`float()`函数将x转换为一个浮点数，x为字符串或数字，没有参数的时候默认返回0.0。

```
>>> float()           #空值转换
```

```
0.0                   #返回结果
```

```
>>> float(1)          #整数转浮点
```

```
1.0                   #返回结果
```

```
>>> float('120')      #字符转浮点
```

```
120.0                 #返回结果
```

### 2.2.5 数据类型的转换

转换为字符串str类型：

`str(x)`

`str()` 函数将对象转化为适于人阅读的形式，`x`为对象，返回值为对象的string类型。

```
>>> x = "今天是晴天"  #定义x
```

```
>>> str(x)             #对x进行转换
```

```
'今天是晴天'          #返回结果
```



### 2.2.5 数据类型的转换

转换为布尔值布尔类型：

`bool(x)`

`bool()` 函数用于把给定参数转换为布尔类型，返回值为True或者False，在没有参数的情况下默认返回 False。

```
>>> bool()          #空置转布尔类型
False               #返回结果
>>> bool(0)         #整数0转布尔值
False               #返回结果
>>> bool(1)         #整数1转布尔值
True                #返回结果
>>> bool(100)       #整数100转布尔值
True                #返回结果
```

```
>>> bool("")        #空串转布
尔值
False               #返回结果
>>> bool("0")       #字符串"0"
转布尔值
True                #返回结果
>>> bool("a")       #字符串"a"
转布尔值
True                #返回结果
```

### 2.2.5 数据类型的转换

Python中常用的数据类型：整数(int)、字符串(str)、布尔值(bool)、列表(list)、元组(tuple)、字典(dict)、浮点数(float)、复数(complex)、可变集合(set)之间可以按规则互相转化。

```
>>> str(3.14)
'3.14'
>>> float('3.14')
3.14
>>> str(2020)
'2020'
>>> int('2020')
2020
```

```
>>> str(False)
'False'
>>> int(False)
0
>>> bool(1)
True
```

```
>>> aList=list(range(1,6))
>>> aList
[1, 2, 3, 4, 5]
>>> tuple(aList)
(1, 2, 3, 4, 5)
```

### 2.2.5 数据类型的判断

**isinstance(x, aType)**函数可以用来测试变量x的是否属于数据类型aType。

```
>>> x=3.14
```

```
>>> y=5
```

```
>>> z='abc'
```

```
>>> f=True
```

```
>>> isinstance(x,float)
```

```
True
```

```
>>> isinstance(x,int)
```

```
False
```

```
>>> isinstance(y,int)
```

```
True
```

```
>>> isinstance(z,bool)
```

```
False
```

```
>>> isinstance(z,str)
```

```
True
```

```
>>> isinstance(f,bool)
```

```
True
```

## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

### 2.3.1 算术运算符

算术运算符主要是用于数字类型的数据基本运算，Python支持直接进行计算，也就是可以将python shell当计算器来使用。

运算符	说明	表达式	结果
+	加：把数据相加	10 + 24	34
-	减：把数据相减	34 - 10	10
*	乘：把数据相乘	34*10	340
/	除：把数据相除	34/10	3.4
%	取模：除法运算求余数	34 % 10	4
**	幂：返回 x 的 y 次幂	2**4	16
//	取整除：返回商整数部分	34 // 10	3

\*可以返回重复若干次的字符串

### 2.3.2 比较运算符

比较运算符用于判断同类型的对象是否相等，比较运算的结果是布尔值True或False，比较时因数据类型不同比较的依据不同。**复数不可以比较大小，但可以比较是否相等。**

在Python中比较的值相同时也不一定是同一个对象。

运算符	说明	表达式	结果
==	等于：判断是否相等	1 == 1	True
!=	不等于：判断是否不相等	1 != 1	False
>	大于：判断是否大于	1 > 2	False
<	小于：判断是否小于	1 < 2	True
>=	大于等于：判断是否大于等于	1 >= 2	False
<=	小于等于：判断是否小于等于	1 <= 2	True

### 2.3.3 逻辑运算符

逻辑运算符为and（与）、or（或）、not（非）用于逻辑运算判断表达式的True或者False，通常与流程控制一起使用

运算符	表达式	x	y	结果	说明
and	x and y	True	True	True	表达式一边有 False 就会返回 False，当两边都是 True 时返回 True。
		True	False	False	
		False	True	False	
		False	False	False	
or	x or y	True	True	True	表达式一边 True 就会返回 True，当两边都是 False 时返回 False。
		True	False	True	
		False	True	True	
		False	False	False	
not	not x	True	/	False	表达式取反，返回值与原值相反。
		False	/	True	

### 2.3.4 赋值、复合赋值运算符

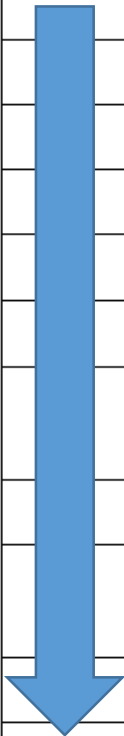
复合赋值运算符时将一个变量参与运算的运算结果赋值给该变量，即a参加了该运算，运算完成后结果赋值给a

运算符	说明	表达式	等效表达式
=	直接赋值	$x = y + z$	$x = z + y$
+=	加法赋值	$x += y$	$x = x + y$
-=	减法赋值	$x -= y$	$x = x - y$
*=	乘法赋值	$x *= y$	$x = x * y$
/=	除法赋值	$x /= y$	$x = x / y$
%=	取模赋值	$x \% = y$	$x = x \% y$
**=	幂赋值	$x ** = y$	$x = x ** y$
//=	整除赋值	$x //= y$	$x = x // y$



### 2.3.5 运算符优先级

由数值、变量、运算符组合的表达式和数学上相同，是有运算符优先级的，**优先级高的运算符先进行运算，同级运算符，自左向右运算，遵从小括号优先原则。等号的同级运算时例外，一般都是自右向左进行运算。**



优先级	类别	运算符	说明
最高	算术运算符	**	指数，幂
高	位运算符	+X,-X,~X	正取反，负取反，按位取反
	算术运算符	*,/,%,//	乘，除，取模，取整
	算术运算符	+,-	加，减
	位运算符	>>,<<	右移，左移运算符
	位运算符	&	按位与，集合并
	位运算符	^	按位异或，集合对称差
	位运算符		按位或，集合并
	比较运算符	<=,<,>,>=	小于等于，小于，大于，大于等于
	比较运算符	==,!=	等于，不等于
	赋值运算符	=,%=,/=,//=, -=,+=,*=,**=	赋值运算
	逻辑运算符	not	逻辑“非”
	逻辑运算符	and	逻辑“与”
低	逻辑运算符	or	逻辑“或”

## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

### 2.4.1 用常量和变量

常量，Python中在程序运行时不会被更改的量称之为常量，一旦初始化后就不能修改的固定值。Python中定义常量需要用对象的方法来创建。

现在有直径为68cm的下水道井盖，需要求面积，其中 $\pi$ 直接使用数学库中的pi，pi即为Python中的常量。

实验实例如下：

```
>>> from math import *    #引入数学库
>>> pi*(68/2)**2          #计算
3631.681107549801         #计算结果
>>> int(pi*(68/2)**2)      #嵌套转换为int类型
3631                       #返回取整的结果
```

### 2.4.1 用常量和变量

#### 变量的使用

Python中变量不需要声明，使用等号直接赋值，值的数据类型为动态类型，也可以使用等号为多个变量赋值。

为a、b、c赋值为“Python编程”，“3.8”，“2020”，然后输出“2020Python编程3.8”然后计算b和c的和，在输出a的内容。

实验实例如下：

```
>>> a , b , c = 'Python编程' , 3.8, 2020  #定义变量和赋值
>>> print(str(c) + a + str(b))             #打印
2020Python编程3.8                         #打印结果
>>> b+c                                     #计算b+c
2023.8                                     #计算结果
>>> a                                       #输出a的内容
'Python编程'                              #输出
```

### 2.4.2 用运算符和表达式

由于Python shell可以直接当计算器使用，输入表达式后可以直接计算出结果，也可以使用变量。

下面计算二的三次方加上三乘五除以十再加上二加一的结果，先使用直接计算，再使用变量。

实验实例如下：

```
>>> 1 + 2 + 3*5/10 + 2**3
```

```
12.5
```

```
>>> a = 1 + 2 + 3*5/10 + 2**3
```

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

```
12.5
```

#输入表达式

#返回计算结果

#给变量a赋值的表达式

#输出变量

#返回计算结果

### 2.4.3 type()函数的使用

type()函数是Python内置的函数用于返回数据类型，当我们要对一个变量赋值时，先要确定变量的数据类型，就会使用到type()函数。

下面将对pi和一些变量进行type()函数的使用实验。

```
>>> from math import * #导入数学库
>>> type(pi)           #查询pi的数据类型
<class 'float'>        #返回为float类型
>>> a = 1               #定义变量a并赋值
>>> b = "python"        #定义变量b并赋值
>>> c = 2.5             #定义变量c并赋值
>>> type(a)             #查询a的数据类型
<class 'int'>          #返回int类型
>>> type(b)             #查询b的数据类型
<class 'str'>          #返回str类型
>>> type(c)             #查询c的数据类型
<class 'float'>        #返回float类型
```

### 2.4.4 help()函数的使用

help() 函数是Python内置用于查看函数或模块用途的详细说明文档的帮助函数。在Python语言中有很多的函数，一般在定义函数时会加上说明文档，说明函数的功能以及使用方法。

下面我们通过查看print()函数、input()函数和一些数据类型来进行help()函数的使用实验（部分文档内容进行了删减）。

实验实例如下：

```
>>> help(print)      #查询print()函数的帮助
Help on built-in function print in module builtins:
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
```

略

### 2.4.4 help()函数的使用

下面我们通过查看print()函数、input()函数和一些数据类型来进行help()函数的使用实验（部分文档内容进行了删减）。

实验实例如下：

```
>>> help(input)      #查询input()函数的帮助
Help on built-in function input in module builtins:
input(prompt=None, /)
```

略

```
>>> help("int")      #查询int的使用说明
Help on class int in module builtins:
class int(object)
```

略

```
>>> help("float")    #查询float的使用说明
Help on class float in module builtins:
class float(object)
```

略



## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

本章主要对Python的代码风格、变量、数据类型、运算符进行了简单讲解，都是学习Python语言的基础知识，希望大家在学习时多加理解，对代码风格也要多加记忆和练习，对Python的变量和运算符要经常使用，加深印象，为后面更好的学习Python做准备。

## 第二章 基本语法

2.1 PEP8 风格指南

2.2 变量与数据类型

2.3 表达式

2.4 实验

2.5 小结

习题

## 习题:

1.在python中, float的数据类型是如何表达的 (实例)

\_\_\_\_\_。

2.Int类型的数据转换为布尔值类型的结果有\_\_\_\_\_和

\_\_\_\_\_。

3.要查询变量的类型可以用\_\_\_\_\_。

4.运算符中优先级最高的是\_\_\_\_\_。

5.Python中的数据类型分为\_\_\_\_\_个大类, bool是哪一个  
大类中的\_\_\_\_\_。

感谢聆听

