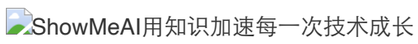
图解 Python 编程(5) | 运算符

韩信子 2021-11-09 1499 0 工具教程 python 编程语言



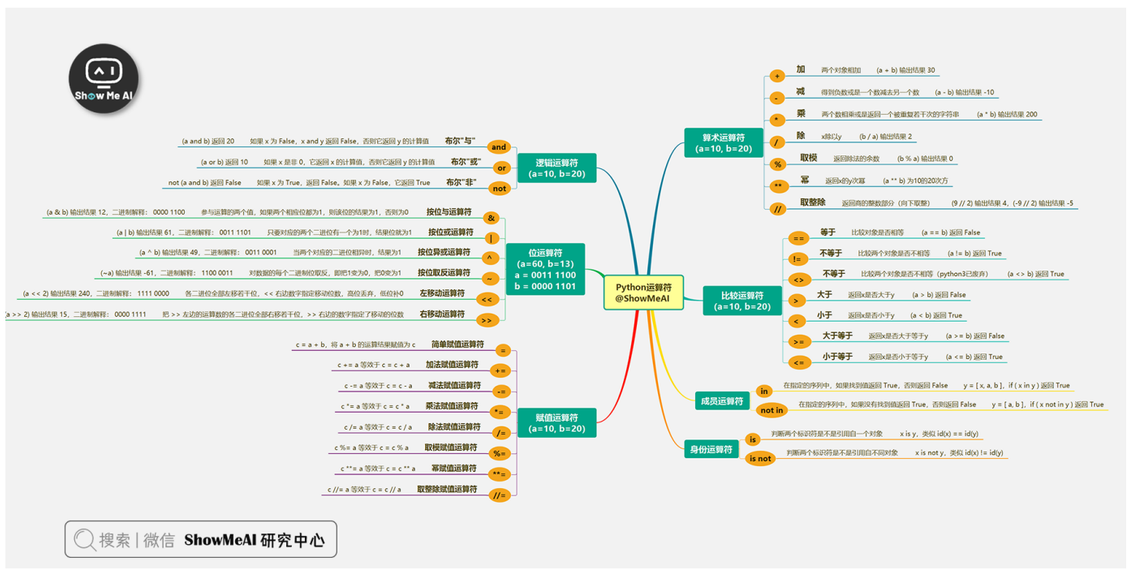
目录

- Python 运算符
- Python算术运算符
- Python比较运算符
- Python赋值运算符
- Python位运算符
- Python逻辑运算符
- Python成员运算符
- Python身份运算符
- Python运算符优先级
- 一键运行所有代码
- 下载Python要点速查表
- 拓展参考资料

作者：韩信子@ShowMeAI
教程地址：**https://www.showmeai.tech/tutorials/56**
本文地址：**https://www.showmeai.tech/article-detail/68**

Python 运算符

运算符用于对变量和值执行操作。举个简单的例子 5 +6 = 11。 例子中，5 和 6 被称为**操作数**，”+” 称为运算符。



@ShowMeAI">

Python 语言支持以下类型的运算符:

- 算术运算符
- 比较（关系）运算符
- 赋值运算符
- 逻辑运算符
- 位运算符
- 成员运算符
- 身份运算符
- 运算符优先级

Python算术运算符

以下假设变量： a=10，b=20：

运算符	描述	实例
+	加 - 两个对象相加	a + b 输出结果 30
-	减 - 得到负数或是一个数减去另一个数	a - b 输出结果 -10
*	乘 - 两个数相乘或是返回一个被重复若干次的字符串	a * b 输出结果 200
/	除 - x除以y	b / a 输出结果 2
%	取模 - 返回除法的余数	b % a 输出结果 0
**	幂 - 返回x的y次幂	a**b 为10的20次方， 输出结果 100000000000000000000
//	取整除 - 返回商的整数部分（向下取整）	`>>> 9//2 4 >>> -9//2 -5`

以下代码演示了Python所有算术运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 30
b = 10
c = 0

c = a + b
print("第1次运算后，c的值为：", c)
```

```
c = a - b
print("第2次运算后，c的值为：", c)

c = a * b
print("第3次运算后，c的值为：", c)

c = a / b
print("第4次运算后，c的值为：", c)

c = a % b
print("第5次运算后，c的值为：", c)

# 修改变量 a 、b 、c
a = 2
b = 3
c = a**b
print("第6次运算后，c的值为：", c)

a = 10
b = 5
c = a//b
print("第7次运算后，c的值为：", c)
```

以上代码输出结果：

```
第1次运算后，c的值为： 40
第2次运算后，c的值为： 20
第3次运算后，c的值为： 300
第4次运算后，c的值为： 3.0
第5次运算后，c的值为： 0
第6次运算后，c的值为： 8
第7次运算后，c的值为： 2
```

Python比较运算符

以下假设变量 a 为 10，变量 b 为 20：

运算符	描述	实例
==	等于 - 比较对象是否相等	(a == b) 返回 False。
!=	不等于 - 比较两个对象是否不相等	(a != b) 返回 true.
<>	不等于 - 比较两个对象是否不相等。**python3 已废弃**。	(a <> b) 返回 true。 这个运算符类似 != 。
>	大于 - 返回x是否大于y	(a > b) 返回 False。
<	小于 - 返回x是否小于y。所有比较运算符返回1表示真， 返回0表示假。这分别与特殊的变量True和False等价。	(a < b) 返回 true。
>=	大于等于 - 返回x是否大于等于y。	(a >= b) 返回 False。
<=	小于等于 - 返回x是否小于等于y。	(a <= b) 返回 true。

以下代码演示了Python所有比较运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 30
b = 10
c = 0

if  a == b :
   print("a 等于 b")
else:
   print("a 不等于 b")

if  a != b :
   print("a 不等于 b")
else:
   print("a 等于 b")

if  a < b :
   print("a 小于 b" )
else:
   print("a 大于等于 b")

if  a > b :
   print("a 大于 b")
else:
   print("a 小于等于 b")

# 修改变量 a 和 b 的值
a = 5
b = 20
if  a <= b :
   print("a 小于等于 b")
else:
   print("a 大于  b")

if  b >= a :
   print("b 大于等于 a")
else:
   print("b 小于 a")
```

以上实例输出结果：

```
a 不等于 b
a 不等于 b
a 大于等于 b
a 大于 b
a 小于等于 b
b 大于等于 a
```

Python赋值运算符

以下假设变量 a 为 10，变量 b 为 20：

运算符	描述	实例
=	简单的赋值运算符	c = a + b 将 a + b 的运算结果赋值为 c
+=	加法赋值运算符	c += a 等效于 c = c + a
-=	减法赋值运算符	c -= a 等效于 c = c - a
*=	乘法赋值运算符	c *= a 等效于 c = c * a
/=	除法赋值运算符	c /= a 等效于 c = c / a
%=	取模赋值运算符	c %= a 等效于 c = c % a
**=	幂赋值运算符	c **= a 等效于 c = c ** a
//=	取整除赋值运算符	c //= a 等效于 c = c // a

以下代码演示了Python所有赋值运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 30
b = 10
c = 0

c = a + b
print("第1次运算后，c的值为：", c)

c += a
print("第2次运算后，c的值为：", c )

c *= a
print("第3次运算后，c的值为：", c )

c /= a
print("第4次运算后，c的值为：", c )

c = 2
c %= a
print("第5次运算后，c的值为：", c)

c **= a
print("第6次运算后，c的值为：", c)

c //= a
print("第7次运算后，c的值为：", c)
```

以上代码输出结果：

```
第1次运算后，c的值为： 40
第2次运算后，c的值为： 70
第3次运算后，c的值为： 2100
第4次运算后，c的值为： 70.0
第5次运算后，c的值为： 2
第6次运算后，c的值为： 1073741824
第7次运算后，c的值为： 35791394
```

Python位运算符

按位运算符是把数字看作二进制来进行计算的。Python中的按位运算法则如下：

下表中变量 a 为 60，b 为 13，二进制格式如下：

```
a = 0011 1100

b = 0000 1101

-----------------

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a  = 1100 0011
```

运算符	描述	实例
&	按位与运算符：参与运算的两个值, 如果两个相应位都为1,则该位的结果为1,否则为0	(a & b) 输出结果 12 ， 二进制解释： 0000 1100
|	按位或运算符： 只要对应的二个二进位有一个为1时，结果位就为1	(a | b) 输出结果 61 ， 二进制解释： 0011 1101

	只要对应的二个二进位有一个为1时，结果位就为1。	0011 1101
^	按位异或运算符：当两对应的二进位相异时，结果为1	(a ^ b) 输出结果 49 ， 二进制解释：0011 0001
~	按位取反运算符：对数据的每个二进制位取反,即把1变为0,把0变为1 。**~x** 类似于 **-x-1**	(~a) 输出结果 -61 ， 二进制解释：1100 0011，在一个有符号二进制数的补码形式。
<<	左移动运算符：运算数的各二进位全部左移若干位，由 **<<** 右边的数字指定了移动的位数，高位丢弃，低位补0。	a << 2 输出结果 240 ， 二进制解释：1111 0000
>>	右移动运算符：把">>"左边的运算数的各二进位全部右移若干位，**>>** 右边的数字指定了移动的位数	a >> 2 输出结果 15 ， 二进制解释：0000 1111

以下代码演示了Python所有位运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 60            # 60 = 0011 1100
b = 13            # 13 = 0000 1101
c = 0

c = a & b;        # 12 = 0000 1100
print("第1次运算后，c的值为：", c)

c = a | b;        # 61 = 0011 1101
print("第2次运算后，c的值为：", c)

c = a ^ b;        # 49 = 0011 0001
print("第3次运算后，c的值为：", c)

c = ~a;           # -61 = 1100 0011
print("第4次运算后，c的值为：", c)

c = a << 2;       # 240 = 1111 0000
print("第5次运算后，c的值为：", c)

c = a >> 2;       # 15 = 0000 1111
print("第6次运算后，c的值为：", c)
```

以上代码输出结果：

```
第1次运算后，c的值为： 12
第2次运算后，c的值为： 61
第3次运算后，c的值为： 49
第4次运算后，c的值为： -61
第5次运算后，c的值为： 240
第6次运算后，c的值为： 15
```

Python逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

运算符	逻辑表达式	描述	实例
and	x and y	布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	(a and b) 返回 20。
or	x or y	布尔"或" - 如果 x 是非 0，它返回 x 的计算值，否则它返回 y 的计算值。	(a or b) 返回 10。
not	not x	布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。	not(a and b) 返回 False

以下代码演示了Python所有逻辑运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 10
b = 20

if  a and b :
   print("1.变量 a 和 b 都为 true")
else:
   print("1.变量 a 和 b 有一个不为 true")

if  a or b :
   print("2.变量 a 和 b 都为 true，或其中一个变量为 true")
else:
   print("2.变量 a 和 b 都不为 true")

# 修改变量 a 的值
a = 0
if  a and b :
   print("3.变量 a 和 b 都为 true")
else:
   print("3.变量 a 和 b 有一个不为 true")

if  a or b :
   print("4.变量 a 和 b 都为 true，或其中一个变量为 true")
else:
   print("4.变量 a 和 b 都不为 true")

if not( a and b ):
   print("5.变量 a 和 b 都为 false，或其中一个变量为 false")
else:
   print("5.变量 a 和 b 都为 true")
```

以上代码输出结果：

```
1.变量 a 和 b 都为 true
2.变量 a 和 b 都为 true，或其中一个变量为 true
3.变量 a 和 b 有一个不为 true
4.变量 a 和 b 都为 true，或其中一个变量为 true
5.变量 a 和 b 都为 false，或其中一个变量为 false
```

Python成员运算符

除了以上的一些运算符之外，Python 还支持成员运算符，测试实例中包含了一系列的成员，包括字符串，列表或元组。

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中 , 如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True， 否则返回 False。	x 不在 y 序列中 , 如果 x 不在 y 序列中返回 True。

以下代码演示了Python所有成员运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 10
b = 20
list = [1, 2, 3, 4, 5 ];

if ( a in list ):
   print("1.变量 a 在给定的列表中 list 中")
else:
   print("1.变量 a 不在给定的列表中 list 中")

if ( b not in list ):
   print("2.变量 b 不在给定的列表中 list 中")
else:
   print("2.变量 b 在给定的列表中 list 中")

# 修改变量 a 的值
a = 2
if ( a in list ):
   print("3.变量 a 在给定的列表中 list 中")
else:
   print("3.变量 a 不在给定的列表中 list 中")
```

以上代码输出结果：

```
1.变量 a 不在给定的列表中 list 中
2.变量 b 不在给定的列表中 list 中
3.变量 a 在给定的列表中 list 中
```

Python身份运算符

身份运算符用于比较两个对象的存储单元

运算符	描述	实例
is	is 是判断两个标识符是不是引用自一个对象	**x is y**, 类似 **id(x) == id(y)** , 如果引用的是同一个对象则返回 True，否则返回 False
is not	is not 是判断两个标识符是不是引用自不同对象	**x is not y** ， 类似 **id(x) != id(y)**。 如果引用的不是同一个对象则返回结果 True， 否则返回 False。

注： `id()` 函数用于获取对象内存地址。

以下代码演示了 Python 所有身份运算符的操作（代码可以在在线 Python 3 环境中运行）：

```
a = 20
b = 20

if ( a is b ):
   print("1.a 和 b 有相同的标识")
else:
   print("1.a 和 b 没有相同的标识")

if ( a is not b ):
   print("2.a 和 b 没有相同的标识")
else:
   print("2.a 和 b 有相同的标识")

# 修改变量 b 的值
b = 30
if ( a is b ):
   print("3.a 和 b 有相同的标识")
else:
   print("3.a 和 b 没有相同的标识")

if ( a is not b ):
   print("4.a 和 b 没有相同的标识")
```

```
23. else:
24.     print("4.a 和 b 有相同的标识")
```

以上实例输出结果：

```
1. 1.a 和 b 有相同的标识
2. 2.a 和 b 有相同的标识
3. 3.a 和 b 没有相同的标识
4. 4.a 和 b 没有相同的标识
```

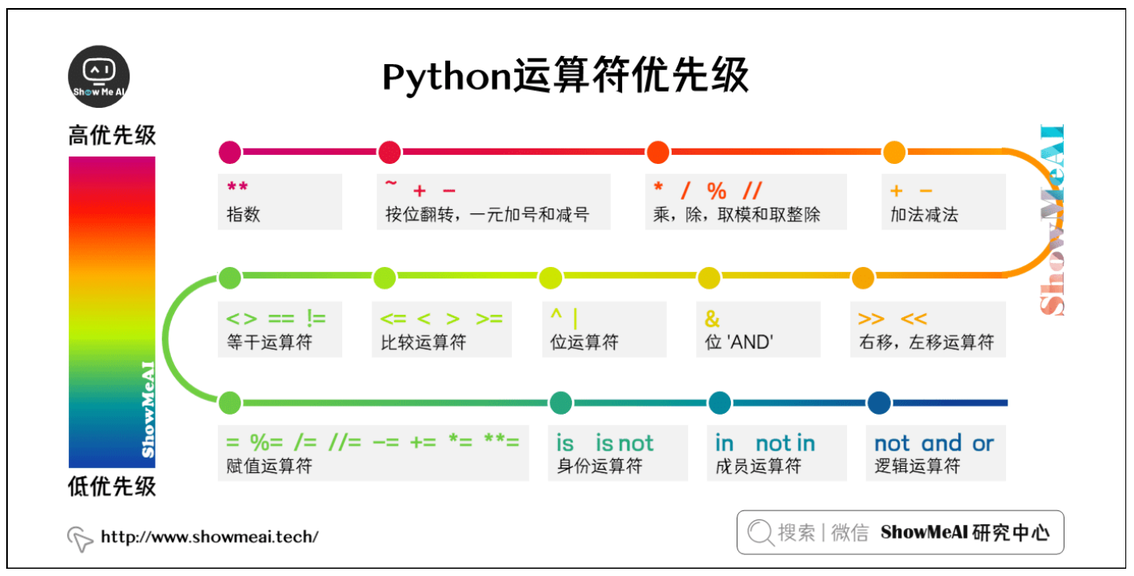
> *is 与 == 区别：*
> *is 用于判断两个变量引用对象是否为同一个(同一块内存空间)， == 用于判断引用变量的值是否相等。*

```
1. >>> a = [1, 2, 3]
2. >>> b = a
3. >>> b is a
4. True
5. >>> b == a
6. True
7. >>> b = a[:]
8. >>> b is a
9. False
10. >>> b == a
11. True
```

Python运算符优先级

以下表格列出了从最高到最低优先级的所有运算符：

运算符	描述
**	指数 (最高优先级)
~ + -	按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@)
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 ‘AND’
^ |	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += = *=	赋值运算符
is is not	身份运算符
in not in	成员运算符
not and or	逻辑运算符



以下代码演示了 Python 所有运算符优先级的操作（代码可以在在线 Python 3 环境中运行）：

```
1. a = 20
2. b = 10
3. c = 15
4. d = 5
5. e = 0
6.
7. e = (a + b) * c / d       #( 30 * 15 ) / 5
8. print("(a + b) * c / d 运算结果为：", e)
9.
10. e = ((a + b) * c) / d     # (30 * 15 ) / 5
11. print("((a + b) * c) / d 运算结果为：", e)
12.
13. e = (a + b) * (c / d);    # (30) * (15/5)
14. print("(a + b) * (c / d) 运算结果为：", e)
15.
16. e = a + (b * c) / d;      #  20 + (150/5)
17. print("a + (b * c) / d 运算结果为：", e)
```

以上实例输出结果：

```
1. (a + b) * c / d 运算结果为： 90.0
```

```
2. ((a + b) * c) / d 运算结果为： 90.0
3. (a + b) * (c / d) 运算结果为： 90.0
4. a + (b * c) / d 运算结果为： 50.0
```

一键运行所有代码

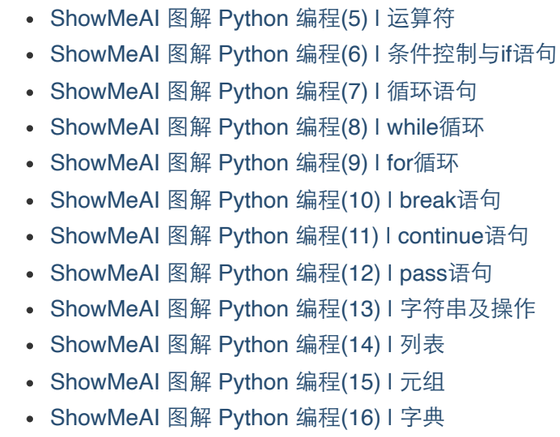
图解Python编程系列 配套的所有代码，可前往ShowMeAI 官方 **GitHub**，下载后即可在本地 Python 环境中运行。能访问 Google 的宝宝也可以直接借助 Google Colab一键运行与交互学习！

下载Python要点速查表

Awesome cheatsheets | **ShowMeAI速查表大全** 系列包含『编程语言』『AI技能知识』『数据科学工具库』『AI垂直领域工具库』四个板块，追平到工具库当前最新版本，并跑通了所有代码。点击 官网 或 **GitHub** 获取~


ShowMeAI速查表大全

👇 **Python 速查表（部分）**



拓展参考资料

- Python教程 - Python3文档
- Python教程 - 廖雪峰的官方网站

ShowMeAI图解Python编程系列推荐（要点速查版）

- ShowMeAI 图解 Python 编程(1) | 介绍
- ShowMeAI 图解 Python 编程(2) | 安装与环境配置
- ShowMeAI 图解 Python 编程(3) | 基础语法
- ShowMeAI 图解 Python 编程(4) | 基础数据类型
- ShowMeAI 图解 Python 编程(5) | 运算符
- ShowMeAI 图解 Python 编程(6) | 条件控制与if语句
- ShowMeAI 图解 Python 编程(7) | 循环语句
- ShowMeAI 图解 Python 编程(8) | while循环
- ShowMeAI 图解 Python 编程(9) | for循环
- ShowMeAI 图解 Python 编程(10) | break语句
- ShowMeAI 图解 Python 编程(11) | continue语句
- ShowMeAI 图解 Python 编程(12) | pass语句
- ShowMeAI 图解 Python 编程(13) | 字符串及操作
- ShowMeAI 图解 Python 编程(14) | 列表
- ShowMeAI 图解 Python 编程(15) | 元组
- ShowMeAI 图解 Python 编程(16) | 字典
- ShowMeAI 图解 Python 编程(17) | 集合
- ShowMeAI 图解 Python 编程(18) | 函数
- ShowMeAI 图解 Python 编程(19) | 迭代器与生成器
- ShowMeAI 图解 Python 编程(20) | 数据结构
- ShowMeAI 图解 Python 编程(21) | 模块
- ShowMeAI 图解 Python 编程(22) | 文件读写
- ShowMeAI 图解 Python 编程(23) | 文件与目录操作
- ShowMeAI 图解 Python 编程(24) | 错误与异常处理
- ShowMeAI 图解 Python 编程(25) | 面向对象编程
- ShowMeAI 图解 Python 编程(26) | 命名空间与作用域
- ShowMeAI 图解 Python 编程(27) | 时间和日期

ShowMeAI系列教程精选推荐

- 大厂技术实现：推荐与广告计算解决方案
- 大厂技术实现：计算机视觉解决方案
- 大厂技术实现：自然语言处理行业解决方案
- 图解Python编程：从入门到精通系列教程
- 图解数据分析：从入门到精通系列教程
- 图解AI数学基础：从入门到精通系列教程
- 图解大数据技术：从入门到精通系列教程
- 图解机器学习算法：从入门到精通系列教程
- 机器学习实战：手把手教你玩转机器学习系列
- 深度学习教程：吴恩达专项课程 · 全套笔记解读
- 自然语言处理教程：斯坦福CS224n课程 · 课程带学与全套笔记解读
- 深度学习与计算机视觉教程：斯坦福CS231n · 全套笔记解读