Python 程序设计基础

江西财经大学-朱文强

- **1. 教材: 朱文强, 钟元生著，《Python程序设计基础》，清华大学出版社, 2025**

- **2. 开发环境**

  Python版本: **3.11-3.12**

  编辑器: **Geany 1.3, Sublime 3**

  IDE平台: **Anaconda3(64-bit), PyCharm**

第二章

语法基础

1 输入输出函数

任何程序语言都需要通过输入输出功能与用户进行交互和沟通。所谓输入就是通过程序捕获用户通过输入设备(如键盘、鼠标、扫描仪等)输入的信息或数据；而输出则是指程序通过输出设备(如显示器、打印机等)向用户显示程序的运行结果。

在Python语言中，可通过input()函数获取用户的键盘输入信息，使用print()函数打印程序的输出内容。

1.1 input()函数

input()函数：无论用户输入的是数字还是字母内容，该函数都将返回字符串数据。其语法格式为：

inputdata = input(prompt=None)

其中，**prompt表示提示信息**，默认为空，如果不为空，则显示提示信息。调用input()函数后，程序将暂停运行，等待用户输入。用户输入完毕后按回车键，input()函数将**获取用户的输入内容，并将其转化为一个字符串返回，并自动忽略换行符**。

1.1 input()函数

input()函数可以作为独立的语句使用，也可将其返回结果赋值给变量。

```
>>>inputdata = input('请输入数据内容:')
请输入数据内容: 您好！
>>>inputdata
```

注意：在Sublime中要使用input()，需安装SublimeREPL插件，或在cmd中以: python *.py的形式运行程序

1.2 print()函数

print()函数: 用于打印输出相关内容。其语法格式为:

print(value, …, sep=’ ’, end=’\n’, file=sys.stdout, flush=False)

其各参数含义如下。

value: 表示需要输出的内容对象，一次可输出一个或多个对象(其中’…’表示任意多个对象)，当输出多个对象时，对象之间要用逗号’,’分隔开来;

sep: 表示输出内容为多个对象时，使用的对象之间的分隔符，默认为1个空格符;

1.2 print()函数

end: 表示print()函数输出的结尾字符, 默认值为换行符’\n’;

file: 表示输出位置，可将对象内容输出到文件，默认输出位置为sys.stdout, 即显示器屏幕(标准输出);

flush: 是否将缓存中的内容强制刷新输出, 默认为False。

这些参数中, sep和end两个参数使用较多, 需重点掌握。

```
>>>print('Hello', 'world!’)

>>>print('Hello', 'world!', sep='+’)

>>>print('Hello', 'world!', sep='+', end='')
```

2 变量与注释

2.1 变量

Python中变量的概念和其他程序语言的变量概念一致，也和代数方程中的变量概念一致，是用于存储计算结果或表示值的抽象概念。

变量的值通常是可以动态变化的，通过变量名可以访问相应的值。

学习变量要重点关注3部分信息：变量类型、变量名、变量值。

Python语言的变量不需要申明类型，但要求使用前必须赋值，赋值的目的是将值与变量名进行关联。

2.1 变量

变量赋值语句的格式为:

变量名 = 表达式

在对变量进行赋值时，Python解释器首先对表达式求值，然后将结果存储到变量中。

Python中变量的数据类型可动态修改，**可通过type()函数查看变量的数据类型，通过id()方法查看变量的id身份(id身份与变量的存储地址关联)**

2.1 变量

message = ‘hello world!’

print(message)

这里我们添加了一个名为message的变量，每个变量都存储一个值。

在程序中，可以随时修改变量的值，而**Python将始终记录变量的最新值**。

message = ‘hello python world’ # 中国 = ‘China’

print(message) # type(), id()

2.2 变量的命名和使用

使用变量时, 需要遵守一些规则和指南, 否则将引发错误。

1. 变量名只能包含字母、数字和下划线, 且区分大小写字母。变量可以字母或下划线开头, 但不能以数字开头(why?);

2. 变量名不能包含空格, 但可使用下划线来分隔其中的单词;

3. 不能将Python关键词和函数名作为变量名;

4. 使用Python变量前, 必须给它赋值(但不需要先申明), 因为Python变量没有默认值;

2.2 变量的命名和使用

5. 变量名应简短和具有描述性；如student_name比s_n好, name_length比length_of_persons_name好;

6. 慎用小写字母l和大写字母O, 以免和数字1和0混淆;

7. 就目前而言, 应使用小写的变量名;

8. 关键字global可以在函数内将变量申明为全局变量;

9. 可使用id()方法来获取变量、对象的身份标志(identity), 并可据此来判断两个对象或变量是否为同一对象;

10. 在交互式环境中, '_'表示最近未命名的变量内容。

2.3 Python的常用关键字

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

提示：Python的关键字可通过keyword模块的kwlist属性查看

2.4 注释

通常来讲, 一个好的、可读性强的程序都会包含必要的代码注释。代码中的注释主要是写给程序员看的, 以便维护和交流使用。**程序执行时，Python解释器会自动忽略注释部分**。

Python中的**注释主要分为单行注释和多行注释两种**。

单行注释以'#'开头, 表示本行代码'#'之后的内容为注释。

多行注释使用3个单引号或3个双引号对来包含, 可对多条语句进行注释, 通常用于模块、类及函数的使用说明之中。

注意：应尽量确保我们编写的程序代码，即便没有注释帮助的情况下, 也简洁易懂。

3 数据类型

Python语言不需要事先指定变量的数据类型, 程序会自动根据变量的值来确定变量的类型。

Python数据类型定义为一个值的集合以及定义在这个值集上的一组运算操作。

Python中基本的**数据类型主要有整型(int)、浮点型(float)、布尔型(bool)和字符串类型(str)**。

3.1 整型

整型用int表示, 通常用于表示整数, 可以用多种进制表达, **默认为十进制**。

>>>a = 50 *2; b = a -10;

其中, 16、8和2进制数分别用以下方式表示:

>>>0xAF **(不区分大小写)** # 0X为16进制

>>>0o16 **(‘o’为字母)** # 0o为8进制

>>>0B10110 # 0B为2进制

>>>0X18 + 0O14

>>>100 + 0x16

3.1 整型

书写很大的整数时，**可使用下划线将其中的数字分组**，使其更清晰易读。Python打印时，不会打印其下划线。

```
>>> uni_age = 14_000_000_000   #3.6以上的大数表示方式
>>> print(uni_age)
14000000000
```

注意：在使用运算符进行算式运算时，应在运算符前后空一个字符，以使程序更加简洁美观

3.1 整型

除了对整数的运算支持外, Python还提供了**int()函数用于将浮点数或整数字符串转化为整数**。 int()函数的格式如下。

int(x, base=y)

函数作用: 将y进制的字符串x转化成十进制整数, 默认为10, 表示十进制, base的取值范围为[2, 36]。

```
>>> int("1001", 2)   # 9
>>> int("1001", 4)   # 65
>>> int("1001", 8)   # 513
>>> int("1001", base=32)   # 32769
```

3.1 整型

当base参数取0时, 程序将自动根据字符串表达的数字含义确定其对应的进制。

```
>>> int('0b11', base=0)  # 3

>>> int('0o11', base=0)   # 9

>>> int('0x11', base=0)    # 17

>>> int(56.2)      # 56
```

3.2 浮点型

Python将带小数点的数字统称为浮点数, 其有两种表示形式: 小数形式和科学计数法。科学计数法使用大写字母E或小写字母e表示10的指数, 后面只能跟1个整数, 不能是小数。同样也可对它们进行’+’,’-’, ‘*’,’/’运算。

```
>>>96e-4        # 0.0096

>>>96e4         # 960000.0

>>>0.1 + 0.1    # 0.2          >>> 0.2 + 0.2    # 0.4
```

3.2 浮点型

```
>>>2 * 0.1    # 0.2

>>>2 * 0.2    # 0.4

>>>0.2 + 0.1   # 0.30000000000000004

>>>0.3 * 1     # 0.2 + 0.1 == 0.3   False
```

注意: 由于机器精度及程序设计机制问题, 浮点数运算可能存在一定的误差, 要尽可能避免浮点数之间进行相等性判断。

3.2 浮点型

对应于整型的int()函数, Python也提供了**float()函数用于将一个整数或数字字符串转换成浮点数**。

```
>>>float(3.5)   #  3.5

>>>float(3)    #  3.0

>>>float('3.5’)     # 3.5
```

提示: 如果使用input()函数来获得用户输入的整数或浮点数字符串, 则可以用int()或float()函数对其直接转换, 以提高程序编写效率。

3.2.1 复数

在Python中, 复数的实数和虚数部分都是浮点类型, 其中虚数部分通过后缀j或J来表示。**对于复数, 可以使用.real和.imag来分别获得它的实数和虚数部分**。

```
>>> a = 12.3 + 4j

>>> a.real    # 12.3

>>> a.imag    # 4.0
```

注意: Python有自动管理内存的功能, 会跟踪所有的值, 并自动删除不再使用或近期引用次数为0的值, 释放其内存空间。

3.3 布尔型

布尔型(bool)是用来表示逻辑’是’或’非’的一种数据类型, 它只有2个值, True和False。

在Python中，使用bool(x)函数用于将x转换成布尔型。**标准值False、None、各种类型(包括浮点数、整数和复数等)的数值0、空序列以及空映射都被视为假，其他各种值都视为真**。

布尔数值可以隐式转换为整数类型使用，默认布尔值True等价于整数1, 布尔值False等价于整数0。

3.3 布尔型

```
>>>bool(1)            #  True
>>>False == bool(’’)    #  True
>>>bool([]) == bool('')        #  True
>>>a = [1, 2, 3]
>>>bool(a)    #  True
>>>True == 1       #  True
>>>True == 2     #  False
>>>True + False + 1   #  2
```

3.3.1 一些常用的数值运算函数

bin(x): 返回整数x的2进制形式;

oct(x): 返回整数x对应的8进制形式;

hex(x): 返回整数x对应的16进制形式;

abs(x): 计算x的绝对值;

round(x, b): 将x圆整到最接近的整数, 参数b指定小数位.

3.3.1 一些常用的数值运算函数

```
import math

math.floor(32.9)   # 向下取整

math.ceil(32.5)    # 向上取整

math.sqrt(9)       # 开平方

round(3.1415, 2)
```

3.4 字符串

字符串就是一系列的字符，属于不可变序列。在Python中，字符串的表示方式有3种。

普通字符串：用一对相同引号(可以是单引号，也可以是双引号)括起的都是字符串；

原始字符串：在普通字符串前加字符r，字符串中的特殊字符不需要转义，按字符串的本来面目呈现；

长字符串：使用3单引号或3双引号包括的字符，将保留字符串的原有排版格式。

Python不支持字符类型，单字符也被视为一个字符串。

3.4 字符串

```
>>>message1 = ‘I said, “python is a wonderful language”’
>>>a = "abc\ndef“
>>>b = r"abc\ndef"
>>>c = """abc
...     def"""
>>>print(a); print(b); print(c)
abc
def
abc\ndef
abc
    def
```

3.4 字符串

>>> msg = b’byte text’ # 将字符串以字节形式存放

>>> msg = “中国” # 非英文字母, 不支持直接加b

注意：在Python3中，默认字符串都是Unicode字符串(加b前缀的除外)。

3.4.1 字符串引号使用规则

使用字符串时, **第一对左单引号和右单引号之间的内容视为字符串, 其余的文本视为普通代码**; 同理, 第一对左双引号和右双引号之间的内容视为字符串, 其余的文本视为普通代码。

message = ‘I’m a person’ (错误)

message = ”I said ” my name is Jony” ” (错误)

message = “I’m a person” (正确)

message = ‘I said ” my name is Jony” ’ (正确)

3.4.1 字符串引号使用规则

要避免单引号、双引号混用所产生的错误, 还可以使用**三引号表示的长字符串(两个三单引号或两个三双引号括起)**。

```
message = """This is a very long string. It continue here.

And it's not over yet. "Hello, world!"

Still here.

"""

print(message)   # 不转义时, 单引和双引号要用在一行内
```

3.4.2 转义字符

对于一些特殊的，难以输入的字符，如换行符、退格符等，Python使用转义字符’\’来辅助实现。常见的转义字符如下表所示。

字符表示	Unicode 编码	功能说明
\t	\u0009	水平制表符
\n	\u00a	换行(分段)
\r	\u00d	回车(下方向键)
\”	\u0022	双引号
\’	\u0027	单引号
\\	\u005c	反斜杠
\b	\u0008	退格符

3.4.2 转义字符

```
>>>print(“Languages:\n\tpython\n\tC\n\tJavaScript”)
>>>print(‘let\’s go’)
>>>print(“\”Hello world\”, she said”)
>>>print(“C:\\npython”)
>>>print(“Hello, \
world”)  # 反斜杠和换行符将被转义, 即被忽略
>>> print(‘\u9009’)    # 选
```

3.4.2 转义字符(用于运算)

```
>>>1 + 2 + \
4 + 5
>>>endings = ['st', 'nd', 'rd'] + 17 * ['th'] \
        + ['st', 'nd', 'rd'] + 7 * ['th'] \
        + ['st']
>>>print(endings)
```

注意: 转义符’\’如果用于运算符中, 后面不能有空字符。

3.4.3 字符串运算符

Python支持使用运算符对字符串进行操作，常见的字符串运算符如下表所示。

操作符	描述
+	字符串拼接
*	重复输出字符串
[]	通过索引获取字符串中字符，从左往右以0开始，从右往左以-1开始。
[a : b]	截取字符串中的一部分,包含 a 不包含 b。
in	成员运算符--如果字符串中包含给定的字符串返回True
not in	成员运算符--如果字符串中不包含给定的字符串返回True
%	格式化字符串

3.4.3 字符串运算符

使用’+’号来拼接字符串, 使用’*’号来重复复制字符串。

```
>>>first_name = ‘ada’

>>>last_name = ‘lovelace’

>>>full_name = first_name + ‘ ’ + last_name

>>>full_name =  “Hello, ” + full_name.title()

>>>a = "he" * 5

>>>print(a)      #  hehehehehe

>>>print(r'Hello, \nworld!’)   #  Hello, \nworld!
```

3.4.4 字符及字符串转换函数

Python还提供了str()函数, 用于将整数、浮点数、序列等非字符串值转化为对应的字符串。

```
>>>age =  23

>>>message = ‘Happy ’ + str(age) + ‘rd Birthday’

>>>print(message)

>>>str([1, 2, 3])     # '[1, 2, 3]'
```

注意: print()在使用逗号作为参数时, 会对其中的表达式进行计算并自动转化为字符串, 这时无需使用str()函数。

3.4.4 字符及字符串转换函数

chr(x): 返回unicode编码x对应的单个字符;

ord(x): 返回单个字符x对应的unicode编码;

```
>>>chr(97)        # ’a’
>>>ord(’a’)       # 97
```

注意：在Python中，字符串以Unicode进行编码，因此，中文字符和英文字符都算是一个字符；而Python解释器则采用UTF-8来存储和传输所有字符信息。

3.4.5 字符串格式化输出

Python支持字符串的格式化输出，其基本用法是将一个值插入到有字符串格式符的模板中。

```
>>>print('我的名字是%s, 年龄是%d' % ('金文', 22))
# 我的名字是金文, 年龄是22
```

注意：从3.6开始，Python社区更加推荐使用format()方法进行格式化输出，其使用更加灵活和方便。

3.4.5 字符串格式化输出

常用的字符串格式化符号如下表所示。

符 号	说 明
%c	格式化字符及其ASCII码
%s	格式化字符串
%d	格式化十进制整数
%o	格式化无符号八进制数
%x	格式化无符号十六进制数
%X	格式化无符号十六进制数（大写）
%f	格式化定点数，可指定小数点后的精度
%e	用科学计数法格式化定点数
%E	作用同%e，用科学计数法格式化定点数
%g	根据值的大小决定使用%f或者%e
%G	作用同%g，根据值的大小决定使用%F或者%E

3.4.5 字符串格式化输出

```
>>>print('我的名字是%s' % 'Python’) # 我的名字是Python
>>>print('我的年龄是%d' % 20)  # 我的年龄是20
>>>print('我的年龄是%o' % 20)   #  我的年龄是24
>>>print('我的年龄是%x' % 20)  #  我的年龄是14
>>>print('我的身高是%f' % 175.8)   #我的身高是175.800000
>>>print('我的身高是%10.2f' % 175.8)
# 我的身高是    175.80
```

3.4.5 字符串格式化输出

```
>>>print('我的身高是%g' % 175.8)  # 我的身高是175.8

>>>print('我的身高是%e' % 175.8)

#  我的身高是1.758000e+02
```

4 运算符

不同的数据类型可执行的操作不同, Python提供了一些常见的运算符用于执行一些基本运算, 如算术运算符、关系运算符、逻辑运算符、位运算符、赋值运算符、成员运算符等。

4.1 算术运算符

算术运算符用于执行加、减、乘、除、取余等基本数学运算, 其中**加、减、乘运算符还可以用于部分序列之中**。常见的算术运算符如下表所示。

运算符	描述	示例（a=10,b=20）
+	将两个对象相加	a + b 输出结果 30
-	负号或者是一个数减去另一个数	a - b 输出结果 -10
*	两个数相乘或者返回一个被重复若干次的字符串	a * b 输出结果 200
/	除以	b / a 输出结果 2.0
%	取模, 返回除法的余数	b % a 输出结果 0
**	返回x的y次幂	a**b 为10的20次方
//	取整，返回商的整数部分（向下取整）	>>> 9//2 # 4 >>> -9//2 # -5

4.1 算术运算符

可对整数执行’+’,’-’, ‘*’,’/’运算, 并可通过圆括号来修改运算次序。

>>>2 + 3

>>>3 % 2　(取模运算)

>>>2 * 3

>>>3 / 2　　# 1.5

>>>2 // 3　(整除运算, 向下取整)

>>>3 ** 2　等价于pow(3, 2) **注意: -3 ** 2与(-3) ** 2的区别**

>>>(2 + 3) * 4

4.2 关系运算符

关系运算符用于比较两个对象之间的大小, 运算结果为True(真)或False(假)。常见的关系运算符如下表所示。

运算符	描述	示例（a=10,b=20）
==	判断两个对象是否相等	(a == b) 返回 False
!=	判断两个对象是否不相等	(a != b) 返回 true
>	大于	(a > b) 返回 False
<	小于	(a < b) 返回 True
>=	大于等于	(a >= b) 返回 False
<=	小于等于	(a <= b) 返回 True

4.2 关系运算符

```
>>>print(12>=8)          # True

>>>print(12<=8)          # False

>>>print(12>=8>=7)       # True 两个表达式都成了才返回True

>>>print(6<=8>=7)        # True

>>>print('abc'<'abe')     # True

>>>print('abc'!='abe')     # True
```

注意：利用关系运算符比较大小，**首先要保证操作对象之间是可比较大小的; 关系运算符可以连用**

4.2 关系运算符

```
>>>print('123'<'abe’)    #  True

>>>print('ABC'<'abe’)   #  True
```

注意：字符串比较大小时, 是从左到右依次比较每个字符Unicode编码来得到结果; 数字字母的Unicode编码<大写英文字母的Unicode编码<小写英文字母的Unicode编码。

4.3 逻辑运算符

逻辑运算符用于判断多个条件是否满足某一要求。 Python提供了3种逻辑运算符:**and(逻辑与), 二元运算符; or(逻辑或), 二元运算符; not(逻辑非), 一元运算符**。

逻辑非的运算结果一定为True或False, 而逻辑与和逻辑或的结果则与具体表达式结果有关。

4.3 逻辑运算符

Python逻辑运算符如下表所示。

运算符	逻辑表达式	描述	示例 (a=10,b=20)
and	x and y	逻辑与, 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	逻辑或, 如果x是非0, 它返回x的值, 否则它返回y的计算值	(a or b) 返回 10
not	not x	逻辑非, 如果x为True, 返回False。如果x为False, 它返回True	not(a and b) 返回 False

4.3 逻辑运算符

```
>>>print(not 5)       #   False
>>>print(not 0)       #   True
>>>print(not ’a’)     #   False
>>>print(not ’’)      #   True
>>>print(not None)    #  True
```

4.3 逻辑运算符

逻辑运算符**and和or也称为短路操作符，具有惰性求值的特点**，表达式从左到右解析，**一旦结果确定就停止运算**。当计算表达式**exp1 and exp2**时，**先计算exp1的值，当exp1的值为True或非空值时，才计算并输出exp2的值**。

```
>>>print(4>3 and 8<9)      # True
>>>print(4>3 and 8)        # 8
>>>print(4<3 and 8)        # False
>>>print(2 and 8)          # 8
>>>print(0 and 5)          # 0
```

4.3 逻辑运算符

计算表达式**exp1 or exp2**时, **先计算exp1的值, 当exp1为True或非空值时, 直接输出exp1的值**, 不再计算exp2的值; **当exp1的值为False或空值时, 才计算并输出exp2的值**。

```
>>>print(4>3 or 8<9)      #  True
>>>print(5>3 or 8)        #  True
>>>print(5<3 or 8)        #  8
>>>print(0 or 8)          #  8
>>>print(5 or 2)          #  5
```

4.3 逻辑运算符

注意：在设计逻辑运算符表达式时，如果能大概预测不同条件失败的概率，并根据and\or短路检测特性组织先后顺序，则可以提高程序运行效率。

4.4 位运算符(进阶)

位(bit)是计算机中表示信息的最小单位，**位运算符实现对数据的位进行操作**。

Python的位运算符有：按位与(&)、按位或(|)、按位异或(^)、按位取反(~)、左位移(<<)、右位移(>>)。

4.4 位运算符(进阶)

Python位运算符及其功能如下表所示。

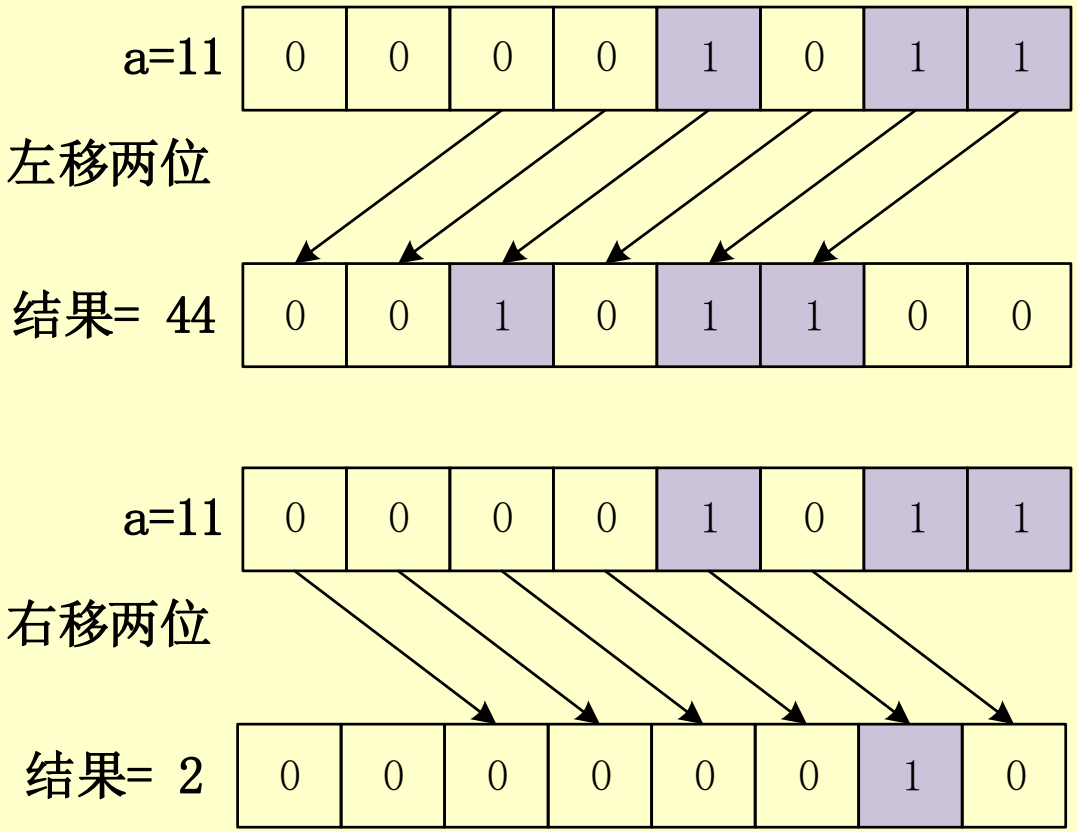
运算符	描述	示例（a=60,b=13）
&	位与运算符，参与运算的两个值，如果两个相应位都为1,则该位的结果为1，否则为0	(a & b) 输出结果 12，二进制解释: 0000 1100
|	位或运算符，只要对应的二个二进位有一个为1时，结果位就为1	(a | b) 输出结果61，二进制解释: 0011 1101
^	按位异或运算符，当两对应的二进位相异时，结果为1	(a ^ b) 输出结果49，二进制解释: 0011 0001
~	按位取反运算符，对数据的每个二进制位取反，即把1变为0,把0变为1 。~x 类似于 -x-1	(~a) 输出结果 -61，二进制解释: 1100 0011，在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

4.4 位运算符(进阶)

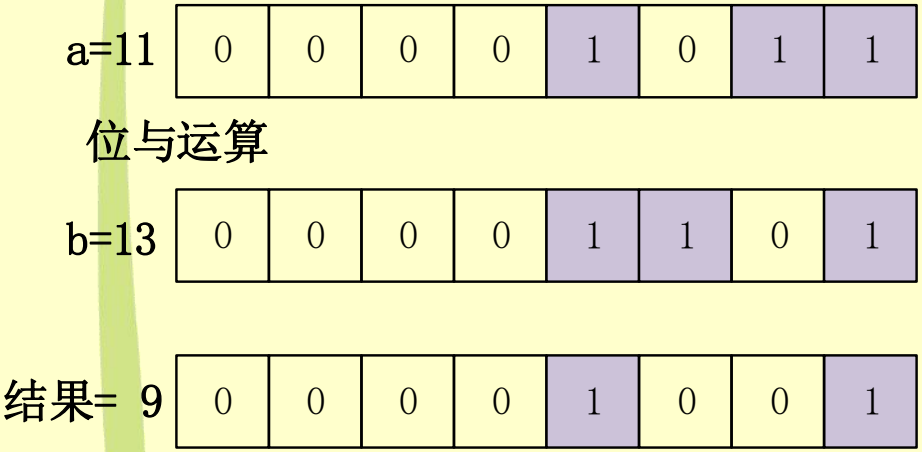
```
>>>print(~5)      # -6
>>>print(11&13)    # 9
>>>print(11|13)    # 15
>>>print(11^13)    # 6
>>>print(11<<2)    # 44
>>>print(11>>2)    # 2
```

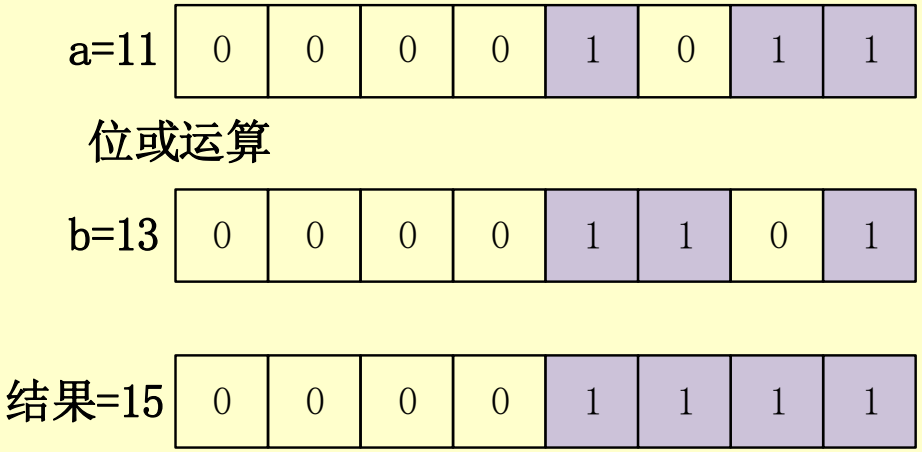
4.4 位运算符(进阶)

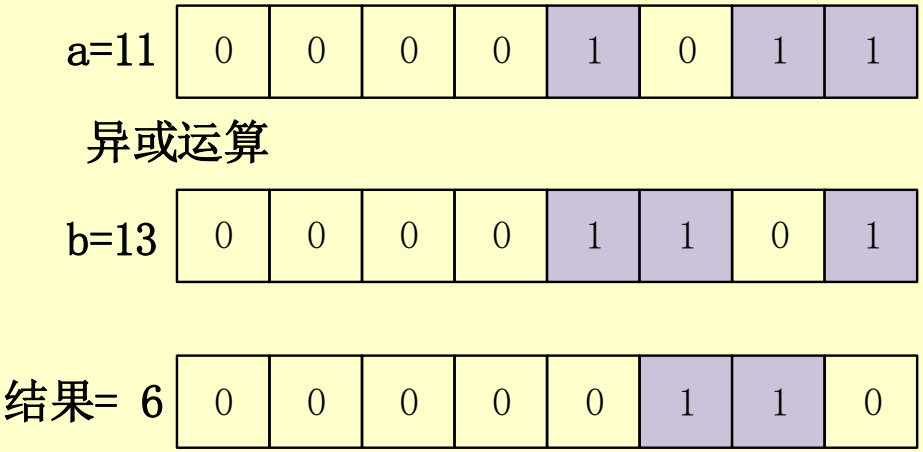
位运算符是对操作数据按其二进制形式逐位进行运算，**参与位运算的操作数必须为整数**。



4.4 位运算符(进阶)


a=11
0 0 0 0 1 0 1 1
位与运算
b=13
0 0 0 0 1 1 0 1
结果= 9
0 0 0 0 1 0 0 1


a=11
0 0 0 0 1 0 1 1
位或运算
b=13
0 0 0 0 1 1 0 1
结果=15
0 0 0 0 1 1 1 1


a=11
0 0 0 0 1 0 1 1
异或运算
b=13
0 0 0 0 1 1 0 1
结果= 6
0 0 0 0 0 1 1 0

4.5 复合赋值运算符

Python支持算术运算符、位运算符和赋值运算符的联合使用，形成复合赋值运算符，等价于**先执行算术运算符或位运算符，然后对结果重新赋值**。

4.5 复合赋值运算符

Python的复合赋值运算符如下表所示。

运算符	名称	例子	等价于
+=	加法赋值运算符	c += a	c = c + a
-=	减法赋值运算符	c -= a	c = c - a
*=	乘法赋值运算符	c *= a	c = c * a
/=	除法赋值运算符	c /= a	c = c / a
%=	取模赋值运算符	c %= a	c = c % a
**=	幂赋值运算符	c **= a	c = c ** a
//=	取整除赋值运算符	c //= a	c = c // a

4.6 成员运算符

成员运算符用于判断对象是否在指定的序列之中。Python的成员运算符如下表所示。

运算符	描述	说明
in	如果在指定的序列中找到相应对象，返回True，否则返回False	x in y，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相关对象，返回 True，否则返回False	x not in y，如果x不在y序列中返回True，否则返回False

```
>>>print('is' in 'This is a python lesson.')      # True

>>>print('eaz' not in 'This is a python lesson.')      # True
```

4.7 身份运算符

身份运算符用于判断对象是否为同一对象，比较两个对象的内存地址是否一致。Python的身份运算符如下表所示。

运算符	描述	说明
is	判断两个对象是否引用自同一个对象	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 是判断两个对象是否引用自不同对象	x is not y, 类似 id(a) != id(b)。如果引用的不是同一个对象则返回结果True, 否则返回False

```
>>>a = 5000; b = 5000; c = a

>>>b is a        #  False

>>>c is a        #  True
```

4.7 身份运算符

```
>>>a = [1, 2, 3]

>>>b = a; c = a.copy(); d = c

>>>b is a       # True

>>>c is a       #  False

>>>d is c       #  True
```

注意：Python语句结尾加上分号，不会影响语句的执行。同时执行多条语句时可用。

4.8 运算符优先级

不同的运算符拥有不同的优先级, 当表达式中包含多种运算符时, 会先**按照运算符的优先级依次执行运算, 对于优先级相同的运算符则从左至右依次进行运算**。

优先级越高的运算执行越早, 在实际应用中, 如果不清楚执行顺序, 可通过加括号的方式来改变运算符的执行顺序。

Python运算符的优先级顺序如下表所示。

4.8 运算符优先级

运算符	描述	运算符	描述
()	小括号, 最高优先级	^, |	异或, 或运算符
**	幂运算	<=, <, >, >=	比较运算符
~, +, -	按位翻转, 正号, 负号	<>, ==, !=	等于运算符
* , /, %, //	乘, 除, 取模和取整除	=, %=, /=, //=, -=, +=, *=, **=	赋值运算符
+, -	加法减法	is, is not	身份运算符
>> , <<	右移, 左移运算符	in, not in	成员运算符
&	位 'AND'	not, and, or	逻辑运算符, 最低优先级

4.8 运算符优先级

```
>>>a = 20; b = 10; c = 15; d = 5

>>>print('(a+b)*c/d的结果为:', (a+b)*c/d)

#   (a+b)*c/d的结果为: 90.0

>>>print('((a+b)*c)/d的结果为:', ((a+b)*c)/d)

#   ((a+b)*c)/d的结果为: 90.0

>>>print('(a+b)*(c/d)的结果为:', (a+b)*(c/d))

#   (a+b)*(c/d)的结果为: 90.0

>>>print('a+(b*c)/d的结果为:', a+(b*c)/d)

#  a+(b*c)/d的结果为: 50.0
```

5 理解traceback

message = ‘Hello, python world’

print(mesage)

当程序无法正常运行时，**解释器会提供一个traceback。它是一条记录，指出解释器尝试运行代码时：** 1. 在什么地方(哪一行)陷入了困境; 2. 它还指出发现的是什么样的错误。

Python解释器不会对代码做拼写检查，但要求变量名的拼写一致。

6 Python编程规范

1、**PEP8(Python Enhancement Proposal)是最好的Python编码规范**, 需要经常查看;

2、**PEP8建议每级缩进四个空格**, 这即提高了程序的可读性, 也留下了足够的多级缩进空间;

3、**缩进是使用的是空格缩进**, 而不是使用制表符缩进;

4、**每行代码的长度建议不要超过80个字符**;

5、**在诸如==、 >=和<=等比较运算符两边各添加一个空格,** 以提高代码的可读性;

6、**尽量使用空行来分隔程序中不同功能的代码块**

6.1 一些原则

1、尽可能选择简单可行的解决方案;

2、项目涉及复杂的代码时, 一定要为这些代码编写有益的注释;

3、在不散失效率的情况下, 尽可能选择常规的解决方案;

4、不要企图编写完美无缺的代码; 先编写行之有效的代码, 再决定对其做进一步改进。

import this

7 Python之禅

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

justice delayed is justice denied

What is rational is real; and what is real is rational.

Answers

```
'python' 不是内部或外部命令，也不是可运行的程序
或批处理文件。


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

(program exited with code: 9009)

请按任意键继续. . .
```

Answers

Spyder无法正常启动的几个解决方式:

1、如果当前用户名为中文名, 那需要新建一个全英文的系统用户, 使用新用户名登录后, 重新安装Anaconda;

2、卸载原有的Anaconda后, 将Anaconda安装在系统盘(c盘);

3、尝试安装32位的Anaconda;

4、尝试卸载Pycharm、Python标准开发包等工具;

5、如果最近安装了pyqt5库, 建议删除试试, 或重装低版本, 并安装Visual Studio社区版。

Answers

```
>>>a = "Thisis"

>>>b = "Thisis"

>>>a is b    # True

>>>b = "This is"

>>>a = "This is"

>>>a is b    # False

>>>a = "中"

>>>b = "中"
```

Answers

```
>>>a is b     #  False

>>>a = 200

>>>b = 200

>>>a is b    # True

>>>a = 300

>>>b = 300

>>>a is b     #  False
```

练习

01 编写教材及课件中的程序和练习

02 熟悉Python语言中的各种运算符及其优先级关系

03 认真阅读Python之禅, 体会其中的含义

谢 谢