

PYTHON程序设计

张树兵 编

202300226改

- 第一章1.5:
- Python基础知识

对象Object:

对象是Python程序处理的核心元素。每个对象都有类型，定义了程序能够在这个对象上做的操作。

Python对象就像一个容器，它可以存储各种类型的数据。举个例子，你可以把一个学生的信息放在一个Python对象里面，这个对象里面可以存储学生的名字、地址、电话号码等信息。这样，你可以把学生的一些信息一起存储在一起，而不需要分开存储。

对象Object:

- Python 对象就像一个盒子，里面装着你想要的东西。它可以是数字、字符串、列表、字典等等。下面是一个例子：
- 例如，你有一个字典，里面装着你朋友的信息：

```
friends = {  
    '张三': 20,  
    '李四': 22,  
    '王二': 25  
}
```

- 这个字典就是一个Python对象，它里面装着朋友的姓名和年龄。

对象Object:

- Python 对象是编程的基础，它们类似于实际世界中的实体物品。例如，您可以把一条狗看作一个Python对象，它有它自己的**属性**（比如颜色、体重、品种等）和**方法**（比如叫、吃东西、玩耍等）。因此，Python对象可以是任何我们可以想象到的东西，它们具有自己的属性和方法。

4类对象：

- ❑ int：表示整数。`int`类型的字面量在形式上与通常的整数一样（如-3、5或10 002）。
- ❑ float：表示实数。`float`类型的字面量总是包括一个小数点（如3.0、3.17或-28.72）。
（还可以用科学计数法表示`float`类型的字面量，如字面量1.6E3表示 1.6×10^3 ，也就是1600.0。）你可能很好奇，这个类型为什么不称为`real`。在计算机中，`float`类型的值是以浮点数的形式保存的。所有现代编程语言都使用这种表示方法，它有很多优点。但是，在某些情况下，使用浮点数计算与使用实数计算会有一些微小的差别。我们会在3.4节详加讨论。
- ❑ bool：表示布尔值`True`和`False`。
- ❑ None：这个类型只有一个值。4.1节将详细讨论。

Table 1-2: Common Data Types

Data type	Examples
Integers	-2, -1, 0, 1, 2, 3, 4, 5
Floating-point numbers	-1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.25
Strings	'a', 'aa', 'aaa', 'Hello!', '11 cats'

表达式: Expression

对象和操作符可以组成表达式，每个表达式都相当于某种类型的对象，我们称其为表达式的值。例如，表达式3 + 2表示int类型的对象5，表达式3.0 + 2.0表示float类型的对象5.0。

==操作符用来检验两个表达式的值是否相等，!=操作符用来检验两个表达式的值是否不等。单个=的意义完全不同，我们会在2.1.2节介绍。预先警告一下，如果你在应该使用==的地方使用了=，就会出现一个错误，请随时注意。

符号>>>是shell提示符，表示解释器正等待用户向shell输入某些Python代码。解释器对在提示符处输入的代码进行求值之后，会在提示符所在行的下一行显示代码结果，用户与解释器的交互过程如下所示：

```
>>> 3 + 2
```

```
5
```

```
>>> 3.0 + 2.0
```

```
5.0
```

```
>>> 3 != 2
```

```
True
```

<字面量><操作符><字面量>

表达式: Expression

- Python表达式是一种特殊的编程语句，它可以让计算机执行一些操作，比如计算数字、执行比较或执行一些简单的操作。
- 例如：

`x = 5 + 3`

- 这个python表达式5+3可以让计算机将5和3相加，然后将结果赋值给变量x，即x的值是8.
- 一个更复杂的Python表达式例子是：

`(2 + 3) * 5`，它的意思是将2加上3，然后再将结果乘以5，结果是25。

表达式: Expression

- Python 表达式是一种可以使用变量、常量、运算符和函数组合而成的表达式，它可以用于计算、比较或执行其他操作。它可以表达一系列简单到复杂的操作，尤其是当表达式中包含多个运算符或函数时。
- 例如：

```
import math
```

```
x=2
```

```
y=3
```

```
z=max(2 * x + 3 * y, math.sqrt(x) + pow(y, 3))
```

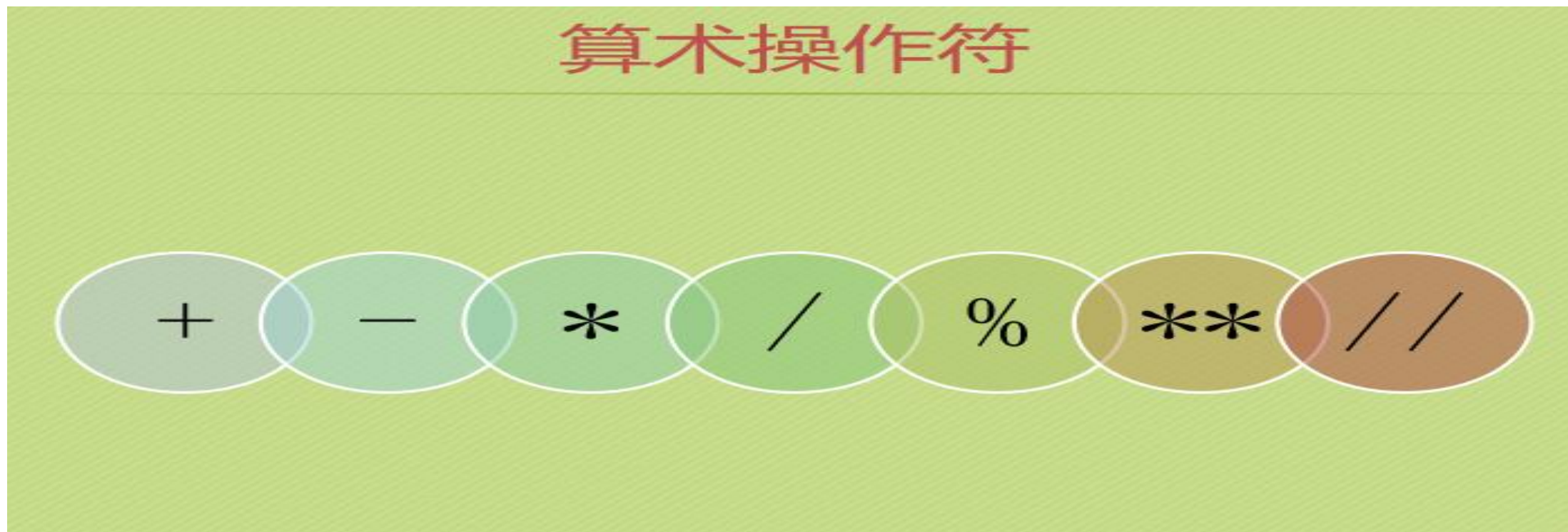
```
print(x,y,z)
```

```
print("x=",x,"y=",y,"z=",z)
```

其中，max,pow是内置函数，sqrt是math库的开平方根函数，x和y是变量，2、3是常量，*和+是运算符。

```
>>> %Run '表达式new.py'
2 3 28.414213562373096
x= 2 y= 3 z= 28.414213562373096
>>>
```

算术运算符：



Python算术运算符包括：

加法 (+)，减法 (-)，乘法 (*)，除法 (/)，取模 (%)，
幂运算 (**)，取整除 (//)。

基本操作符:

int类型和float类型支持的操作符如图2-1所示。

$i + j$: i 和 j 的和。如果 i 和 j 都是int类型, 结果也是int类型; 如果其中任意一个是float类型, 那么结果就是float类型。

$i - j$: 表示 i 减 j 。如果 i 和 j 都是int类型, 结果也是int类型; 如果其中任意一个是float类型, 那么结果就是float类型。

$i * j$: i 和 j 的积。如果 i 和 j 都是int类型, 结果也是int类型; 如果其中任意一个是float类型, 那么结果就是float类型。

$i // j$: 表示整数除法。例如, $6 // 2$ 的值是3, int类型。 $6 // 4$ 的值是1, int类型。值为1的原因是整数除法只返回商, 不返回余数。如果 $j == 0$, 会发生一个错误。

i / j : 表示 i 除以 j 。在Python 3中, $/$ 操作符执行的是浮点数除法。例如, $6 / 4$ 的值是1.5。如果 $j == 0$, 会发生一个错误。(在Python 2中, 当 i 和 j 都是int类型时, $/$ 操作符和 $//$ 操作符一样, 返回int类型的结果。如果 i 或 j 中任意一个是float类型, 那么 $/$ 操作符和Python 3中的 $/$ 操作符一样, 返回float类型的结果。)

$i \% j$: 表示int i 除以int j 的余数。通常读作 $i \bmod j$, 是 $i \text{ modulo } j$ 的缩写。

$i ** j$: 表示 i 的 j 次方。如果 i 和 j 都是int类型, 结果也是int类型。如果其中任意一个是float类型, 那么结果就是float类型。

比较运算符包括: $==$ (等于)、 $!=$ (不等于)、 $>$ (大于)、 $>=$ (大于等于)、 $<$ (小于) 和 $<=$ (小于等于)。

算术运算符：

- Python算术运算符是用来做数学计算的。例如：
- 我们可以用加法运算符 (+) 来把两个数字相加，比如： $2+3=5$
- 也可以用减法运算符 (-) 来把两个数字相减，比如： $5-3=2$
- 还可以用乘法运算符 (*) 来把两个数字相乘，比如： $2*3=6$
- 最后，还可以用除法运算符 (/) 来把两个数字相除，比如： $12/4=3$
- 例子：
- 现在，假设你要计算 $(2 + 3) * 5$
 - 首先，用加法运算符 (+) 把2和3相加，得到5；
- 然后，用乘法运算符 (*) 把5和5相乘，得到25；
- 最后，你就得到了答案： $(2 + 3) * 5 = 25$

算术运算符：

- 例如，如果你有5个苹果，3个橘子，可以使用加号（+）来计算苹果和橘子的总数： $5 + 3 = 8$.
- 你也可以使用减号（-）来计算一个数字减去另一个数字的结果： $8 - 3 = 5$.
- 你还可以使用乘号（*）来计算两个数字的乘积： $3 * 5 = 15$. 或者，你可以使用除号（/）来计算一个数字除以另一个数字的结果： $15 / 5 = 3$.
- 你也可以用取余（%）来计算一个数字除以另一个数字的余数： $7 \% 3 = 1$. **两个数可以整除时，余数是多少？**

算术运算符：//

- Python整除规则就是数学中的除法运算，
- 它的运算结果是商的整数部分。
- 比如： $8 \div 3 = 2$ ，其中8是被除数，3是除数，2就是商的整数部分。

算术运算符：//

- Python整除规则就是数学中的除法运算，
- 它的运算结果是商的整数部分。
- 比如： $8 \div 3 = 2$ ，其中8是被除数，3是除数，2就是商的整数部分。

算术运算符：%

- Python求余规则就是把两个数相除，得到的余数就是求余的结果。
- 例如：
- 如果我们想把17除以5，那么 $17 \div 5 = 3 \dots 2$ ，这里的2就是17除以5的余数。

Table 1-1: Math Operators from Highest to Lowest Precedence

Operator	Operation	Example	Evaluates to . . .
**	Exponent	2 ** 3	8
%	Modulus/remainder	22 % 8	6
//	Integer division/floored quotient	22 // 8	2
/	Division	22 / 8	2.75
*	Multiplication	3 * 5	15
-	Subtraction	5 - 2	3
+	Addition	2 + 2	4

比较运算符:

Python 的比较操作符

>

左边大于右边

>=

左边大于等于右边

<

左边小于右边

<=

左边小于等于右边

==

左边等于右边

!=

左边不等于右边

Comparison Operators

Comparison operators, also called *relational operators*, compare two values and evaluate down to a single Boolean value. Table 2-1 lists the comparison operators.

Table 2-1: Comparison Operators

Operator	Meaning
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

比较运算符:

- Python比较运算符就是用来比较两个值是否相等或者大小关系的运算符。比如:
- 相等运算符: **==**, 用来比较两个值是否相等。例如: `2 == 3`, 表示把2和3进行比较, 2不等于3, 所以结果是False。
- 不等运算符: **!=**, 用来比较两个值是否不等。例如: `2 != 3`, 表示把2和3进行比较, 2不等于3, 所以结果是True。
- 大于运算符: **>**, 用来比较两个值的大小, 左边的值是否大于右边的值。例如: `2 > 3`, 表示把2和3进行比较, 2比3小, 所以结果是False。
- 小于运算符: **<**, 用来比较两个值的大小, 左边的值是否小于右边的值。例如: `2 < 3`, 表示把2和3进行比较, 2比3小, 所以结果是True。

比较运算符:

• 例如:

x = 5

y = 6

如果 x 比 y 大, 则输出 True:
print(x > y) # False

如果 x 比 y 小, 则输出 True:
print(x < y) # True

```
1 x = 5
2 y = 6
3 # 如果 x 比 y 大, 则输出 True:
4 print(x > y) # False
5 # 如果 x 比 y 小, 则输出 True:
6 print(x < y) # True
7
8 print("apple"<"apples")
```

Shell ×

```
>>> %Run '比较运算符new.py'
```

```
False
True
True
```

```
>>>
```

比较运算符：字符串比较

- Python 字符串比较规则就是比较两个字符串中的每一个字符，从左至右依次
- 比较，如果某一位字符较大，那么字符串就较大；如果某一位字符相等，
- 那么继续比较下一位字符；如果比较到最后还是相等，那么两个字符串就是相等的。
- 举个例子：
- 比较两个字符串：**"ABCDE"** 和 **"ABCDF"**
- 首先比较第一个字符："A" 和 "A"，相等，继续比较第二个字符："B" 和 "B"，相等，继续比较第三个字符："C" 和 "C"，相等，继续比较第四个字符："D" 和 "D"，相等，继续比较第五个字符："E" 和 "F"，因为E先F后，所以 **"ABCDE" 小于 "ABCDF"**

比较运算符：字符串比较

- Python字符串比较规则就像比较两个单词，把字符串看作是一个一个的字母。当比较两个字符串时，它会从左到右，一个字母一个字母地比较。
- 比如，比较 "apple" 和 "apples"，首先Python会比较 "a" 和 "a"，它们相同，所以然后就比较 "p" 和 "p"，它们也相同，所以继续比较 "" 和 "s"，由于 "" 比 "s" 小，所以比 "apple" 大，因此推导出：
"apple" < "apples" ,得 True

比较运算符：字符串比较 ord函数

- Python 字符 ascii 码是一种在计算机中表示字符的代码。
 - Python ord 函数可以用来比较字符串。例如，你可以比较两个字符串中的字符，看看哪个字符的ASCII编码更大。
 - 举个例子，比较 "a" 和 "b"，可以使用ord函数看看谁大：
-
- `>>> ord('a')`
 - `97`
 - `>>> ord('b')`
 - `98`
 - 可以看出，"b" 的ASCII编码比"a"大，因此"b"比"a"要大。

比较运算符：字符串比较 ord函数

- Python ord函数可以用来比较字符串。它返回一个字符串中每个字符的Unicode编码值。例如：
- 字符串"apple"
- ord("a") 返回 97
- ord("p") 返回 112
- ord("l") 返回 108
- ord("e") 返回 101
- 所以我们可以比较两个字符串，比较它们中每个字符的Unicode编码值。
- 例如：
- 字符串"apple" 和 字符串"cat"
- **ord("a") < ord("c") 所以 "apple" 比 "cat" 小。**

```
8 print("apple"<"apples")
9
10 print(ord('a'))
11 print(ord('b'))
12 print('a'<'b')
13 print("apple" < "cat")
```

Shell >

>>> %Run '比较运算符new.py'

True

97

98

True

True

ASCII可显示字符

二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形	二进制	十进制	十六进制	图形
0010 0000	32	20	(空格) (sp)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	\$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r
0011 0011	51	33	3	0101 0011	83	53	S	0111 0011	115	73	s
0011 0100	52	34	4	0101 0100	84	54	T	0111 0100	116	74	t
0011 0101	53	35	5	0101 0101	85	55	U	0111 0101	117	75	u
0011 0110	54	36	6	0101 0110	86	56	V	0111 0110	118	76	v
0011 0111	55	37	7	0101 0111	87	57	W	0111 0111	119	77	w
0011 1000	56	38	8	0101 1000	88	58	X	0111 1000	120	78	x

```
>>> 42 == 42
```

```
True
```

```
>>> 42 == 99
```

```
False
```

```
>>> 2 != 3
```

```
True
```

```
>>> 2 != 2
```

```
False
```

```
>>> 'hello' == 'hello'
True
>>> 'hello' == 'Hello'
False
>>> 'dog' != 'cat'
True
>>> True == True
True
>>> True != False
True
>>> 42 == 42.0
True
>>> 42 == '42'
False
```

```
>>> 42 < 100
```

```
True
```

```
>>> 42 > 100
```

```
False
```

```
>>> 42 < 42
```

```
False
```

```
>>> eggCount = 42
```

```
>>> eggCount <= 42
```

```
True
```

```
>>> myAge = 29
```

```
>>> myAge >= 10
```

```
True
```

THE DIFFERENCE BETWEEN THE == AND = OPERATORS

You might have noticed that the == operator (equal to) has two equal signs, while the = operator (assignment) has just one equal sign. It's easy to confuse these two operators with each other. Just remember these points:

- The == operator (equal to) asks whether two values are the same as each other.
- The = operator (assignment) puts the value on the right into the variable on the left.

To help remember which is which, notice that the == operator (equal to) consists of two characters, just like the != operator (not equal to) consists of two characters.

逻辑运算符：

- Python 逻辑运算符用于比较两个或多个值，并返回 True 或 False 的结果。
- 以下是 Python 中常用的逻辑运算符：
- **与 (and)** ： 如果两个值都为True，结果才是True。
- 例如：
- $(3 > 2)$ and $(3 < 5)$ 的结果为True
- **或 (or)** ： 如果其中一个值为True，结果就是True。
- 例如：
- $(3 > 2)$ or $(3 < 1)$ 的结果为True
- **非 (not)** ： 如果值为True，结果为False。
- 例如：
- not $(3 > 2)$ 的结果为False

逻辑运算符：

- Python 逻辑运算符是用来比较真假的。它可以用来判断两个或多个事件是否同时发生或者其中一个发生。 **与 (and) , 或 (or) , 非 (not)**
- 例如：
 - 如果你想买一个新的书包，你需要先攒够钱，而且你必须有足够的时间去买，那么，你就可以使用逻辑运算符来比较这两个条件：
 - **你有足够的钱 and 你有足够的时间**
 - 如果都为真，那么你就可以买新的书包，否则你就不能买。

逻辑运算符： 闰年条件

- Python 闰年的条件是：满足其中一个：
- 一年要能被4整除，但是不能被100整除；
- 要是能被400整除，那么也是闰年。代码如下：

获取年份

```
year = int(input("请输入一个年份： "))
```

判断是否是闰年

```
if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
```

```
    print(year,"是闰年！ ")
```

```
else:
```

```
    print(year,"不是闰年！ ")
```

```
1 # 获取年份
2 year = int(input("请输入一个年份: "))
3
4 # 判断是否是闰年
5 if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
6     print(year, "是闰年!")
7 else:
8     print(year, "不是闰年!")
```

Shell ×

```
>>> %Run '闰年new.py'
```

```
请输入一个年份: 2000
2000 是闰年!
```

```
>>> %Run '闰年new.py'
```

```
请输入一个年份: 2023
2023 不是闰年!
```

```
>>> |
```


逻辑运算符:

x = 5

y = 6

print(x and y == 6) # returns True

print(x or y == 6) # returns True

print(not x == 6) # returns True

Bool类型操作符:

bool类型上的基本操作符为and、or和not。

- ❑ `a and b`: 当a和b都为True时, 值为True, 否则为False。
- ❑ `a or b`: 当a和b至少有一个为True时, 值为True, 否则为False。
- ❑ `not a`: 如果a为False, 值为True; 如果a为True, 值为False。

Table 2-2: The and Operator's Truth Table

Expression	Evaluates to . . .
True and True	True
True and False	False
False and True	False
False and False	False

Table 2-3: The or Operator's Truth Table

Expression	Evaluates to . . .
True or True	True
True or False	True
False or True	True
False or False	False

Table 2-4: The not Operator's Truth Table

Expression		Evaluates to . . .
not	True	False
not	False	True

```
>>> (4 < 5) and (5 < 6)
```

```
True
```

```
>>> (4 < 5) and (9 < 6)
```

```
False
```

```
>>> (1 == 2) or (2 == 2)
```

```
True
```


$(4 < 5) \text{ and } (5 < 6)$



$\text{True and } (5 < 6)$



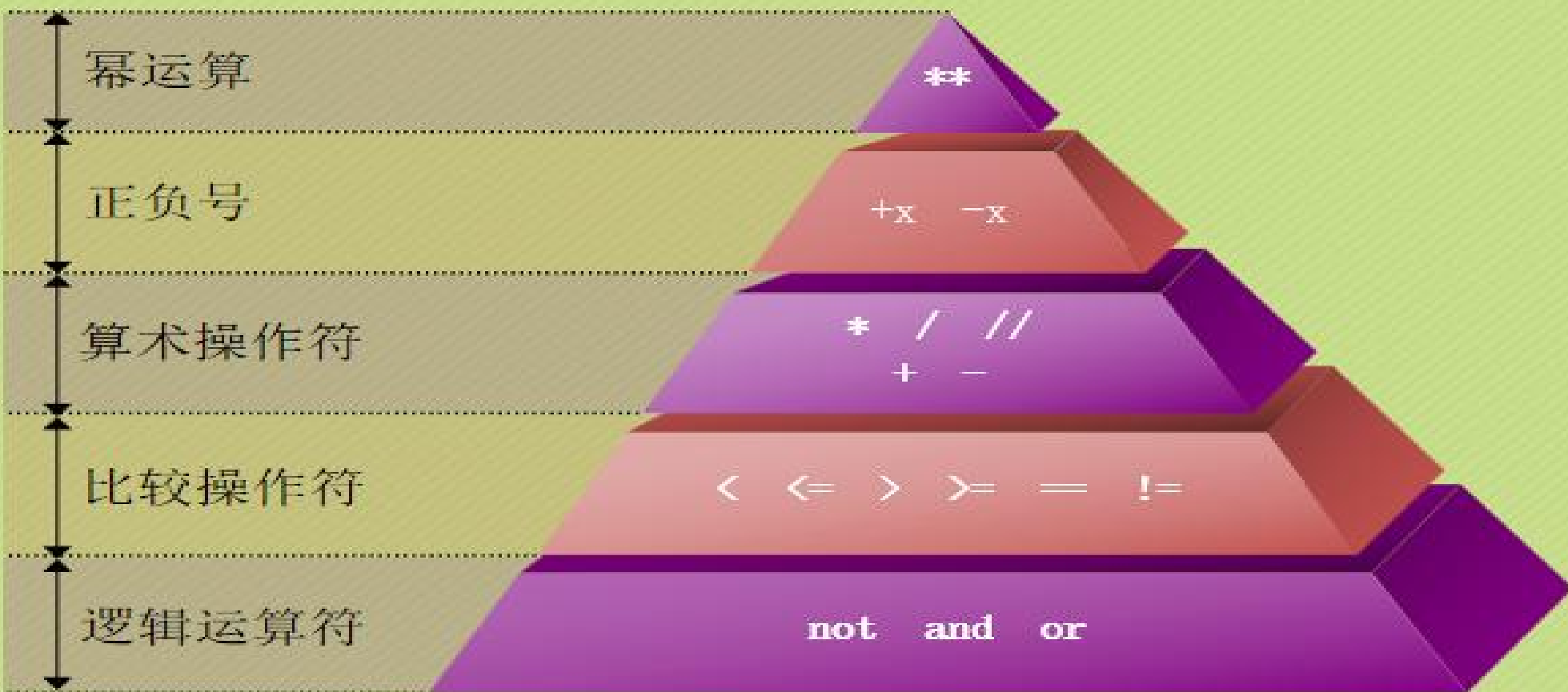
True and True



True

优先级：

又是优先级问题



$$\boxed{(5 - 1)} * ((7 + 1) / (3 - 1))$$

$$4 * \boxed{((7 + 1))} / (3 - 1)$$

$$4 * (8) / \boxed{(3 - 1)}$$

$$4 * (8) / (2)$$

$$\boxed{4 * 4.0}$$

$$16.0$$

变量与赋值：

变量将名称与对象关联起来。看下面的代码：

```
pi = 3  
radius = 11  
area = pi * (radius**2)  
radius = 14
```

如果程序接着执行radius = 14

这段代码先将名称pi和radius绑定到两个int类型的对象。^①然后将名称area绑定到第三个int类型的对象，如图2-2中左图所示。

变量绑定对象:

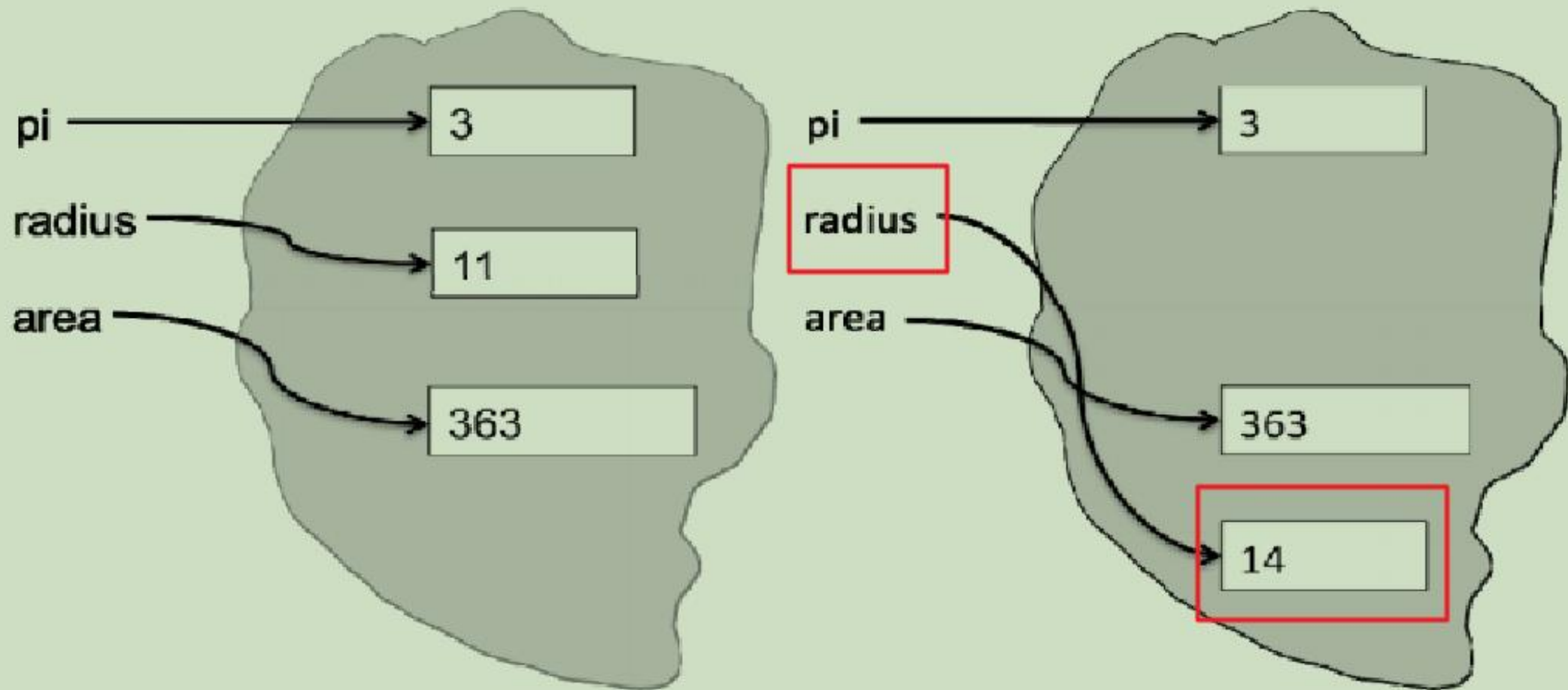
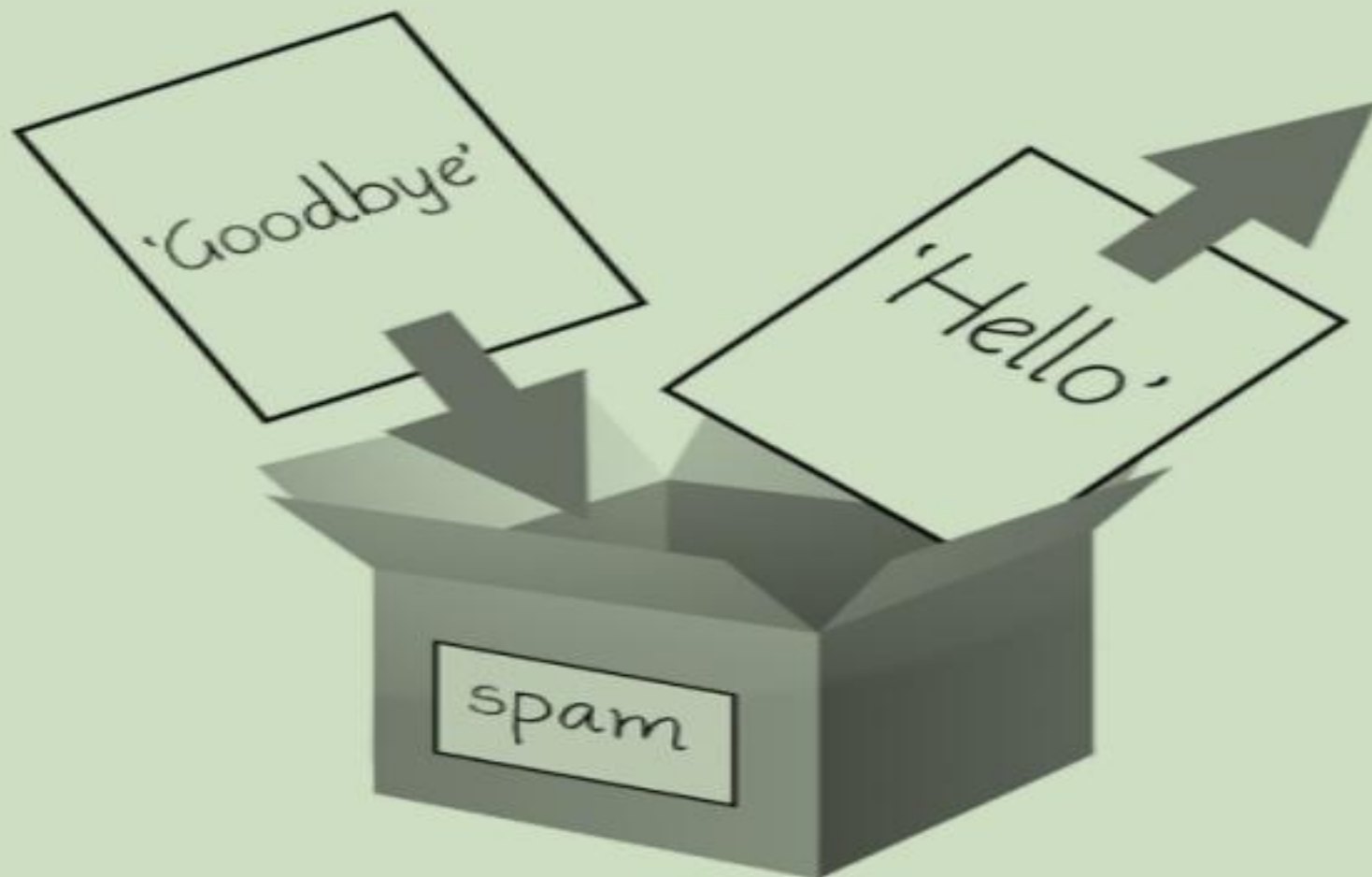


图2-2 将变量绑定到对象

```
>>> spam = 'Hello'  
>>> spam  
'Hello'  
>>> spam = 'Goodbye'  
>>> spam  
'Goodbye'
```

Just like the box in Figure 1-2, the `spam` variable in this example stores 'Hello' until you replace the string with 'Goodbye'.



重点:

在Python中，变量仅是名称，没有其他意义。请牢记这一点，这非常重要。赋值语句将=左边的名称与=右边的表达式所表示的对象关联起来，也请牢记这一点。一个对象可以有一个或多个名称与之关联，也可以不关联任何名称。

变量例子： 计算苹果价格

```
1 score = 100 #定义了一个变量,这个变量的名字叫做score,它里面存储了一个数值 100
2 high = 180 #单位是cm
3
4 applePrice = 3.5 #苹果的价格 元/斤  3.5
5 weight = 7.5 #购买的苹果的重量 斤  7.5
6
7 money = applePrice * weight #如果money=xxxx是第一次的话,那么就表示定义了一个变量  26.25
8
9 money = money  10 #如果 money=xxxx不是第一次出现,那么就不是定义变量,而是给这个已经存在的变量赋上新的值  16.25
```



标示符的规则

标示符由字母、下划线和数字组成，且数字不能开头

fromNo12

from#12

my_Boolean

my-Boolean

Obj2

2ndObj

myInt

test1

Mike2jack

My_tExt

test!32

haha(da)tt

int

jack_rose

jack&rose

GUI

注意：

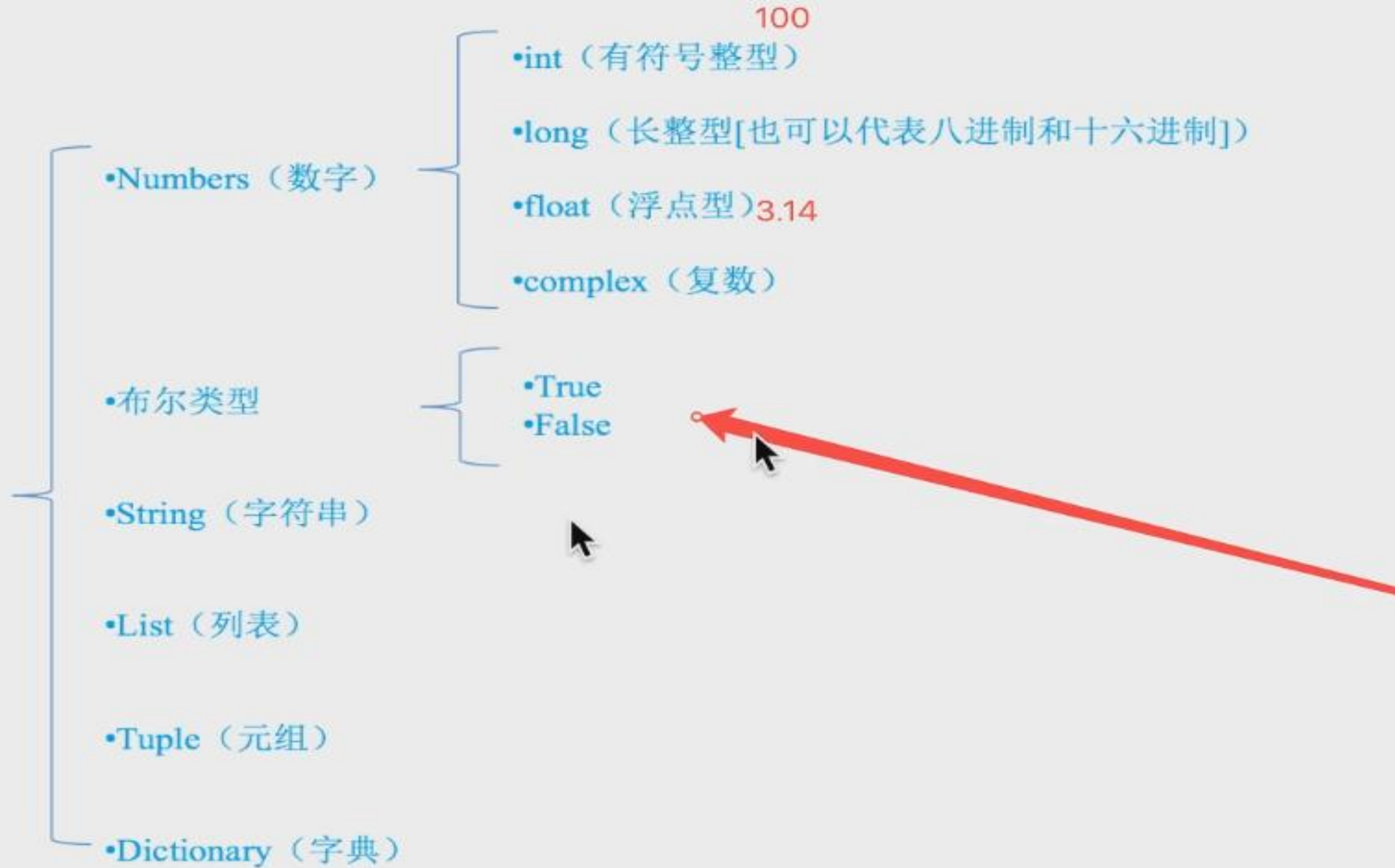
1 大小写是不同的

Cauc与**cauc**

2 不能和关键字重名

Table 1-3: Valid and Invalid Variable Names

Valid variable names	Invalid variable names
current_balance	current-balance (hyphens are not allowed)
currentBalance	current balance (spaces are not allowed)
account4	4account (can't begin with a number)
_42	42 (can't begin with a number)
TOTAL_SUM	TOTAL_\$UM (special characters like \$ are not allowed)
hello	'hello' (special characters like ' are not allowed)



关键字

- 什么是关键字

python一些具有特殊功能的标示符，这就是所谓的关键字

关键字，是python已经使用的了，所以不允许开发者自己定义和关键字相同的名字的标示符

- 查看关键字:

and	as	assert	break	class	continue	def	del
elif	else	except	exec	finally	for	from	global
if	in	import	is	lambda	not	or	pass
print	raise	return	try	while	with	yield	


```
1 import keyword
2 print(keyword.kwlist)
3 |
```

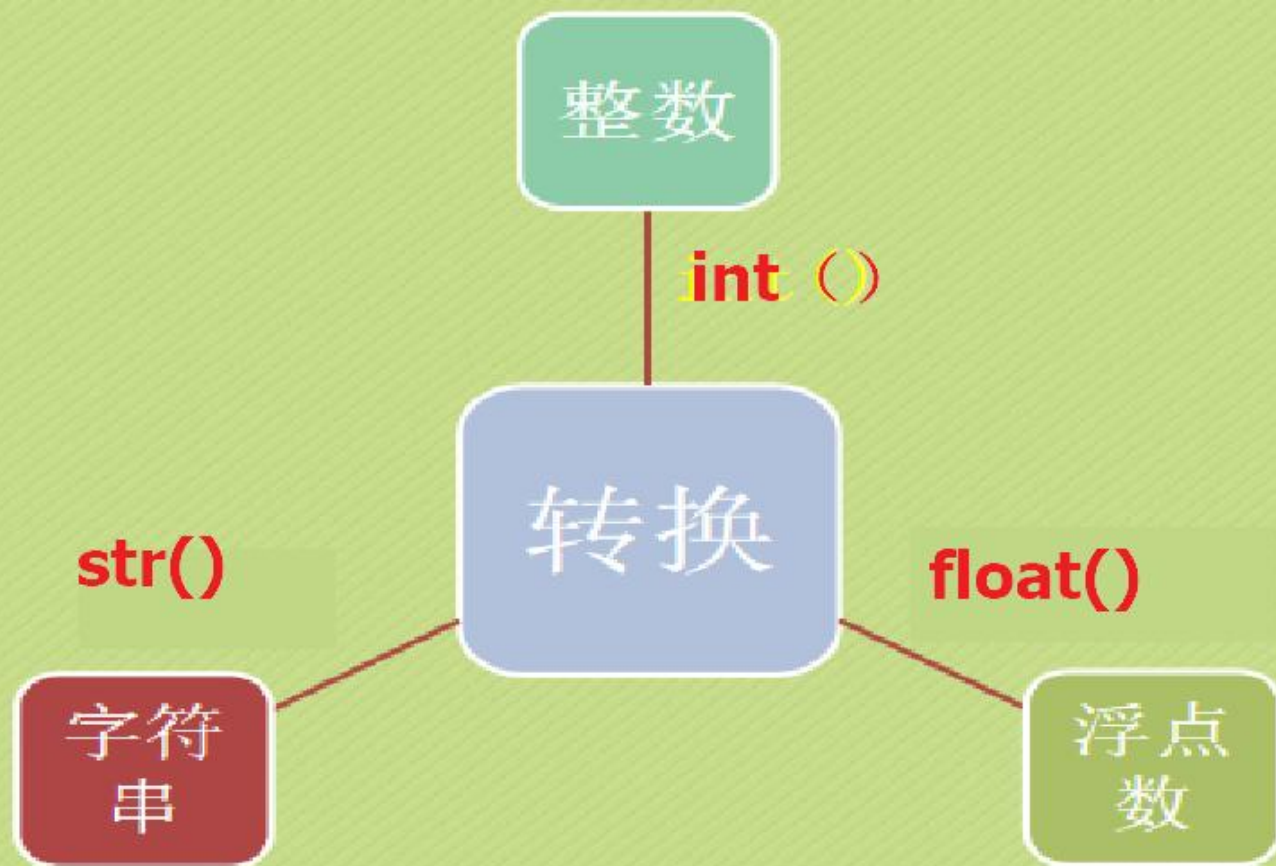
```
>>> %Run my1_keyword.py
```

```
['False', 'None', 'True', 'and', 'as', 'assert',
, 'async', 'await', 'break', 'class', 'continue',
, 'def', 'del', 'elif', 'else', 'except', 'fin
ally', 'for', 'from', 'global', 'if', 'import',
'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'wit
h', 'yield']
```

```
>>>
```

类型转换:

类型转换



The str(), int(), and float() Functions

If you want to concatenate an integer such as 29 with a string to pass to `print()`, you'll need to get the value `'29'`, which is the string form of 29. The `str()` function can be passed an integer value and will evaluate to a string value version of the integer, as follows:

```
>>> str(29)
```

```
'29'
```

```
>>> print('I am ' + str(29) + ' years old.')
```

```
I am 29 years old.
```

```
>>> str(0)
'0'
>>> str(-3.14)
'-3.14'
>>> int('42')
42
>>> int('-99')
-99
>>> int(1.25)
1
>>> int(1.99)
1
>>> float('3.14')
3.14
>>> float(10)
10.0
```

Input函数：输入手段

Python 3中有一个`input`函数，可以直接接受用户输入。^①它可以使用一个字符串作为参数，显示在shell中作为提示信息，然后等待用户输入，用户输入以回车键结束。用户输入的行信息被看作一个字符串，并成为这个函数的返回值。

看下面的代码：

```
>>> name = input('Enter your name: ')\nEnter your name: George Washington
```

```
>>> n = input('Enter an int: ')\nEnter an int: 3\n>>> print(type(n))\n<type 'str'>
```


Print: 输出

1. 输出字符串和数字

```
>>> print("runoob") # 输出字符串
```

```
runoob
```

```
>>> print(100) # 输出数字
```

```
100
```

```
>>> str = 'runoob'
```

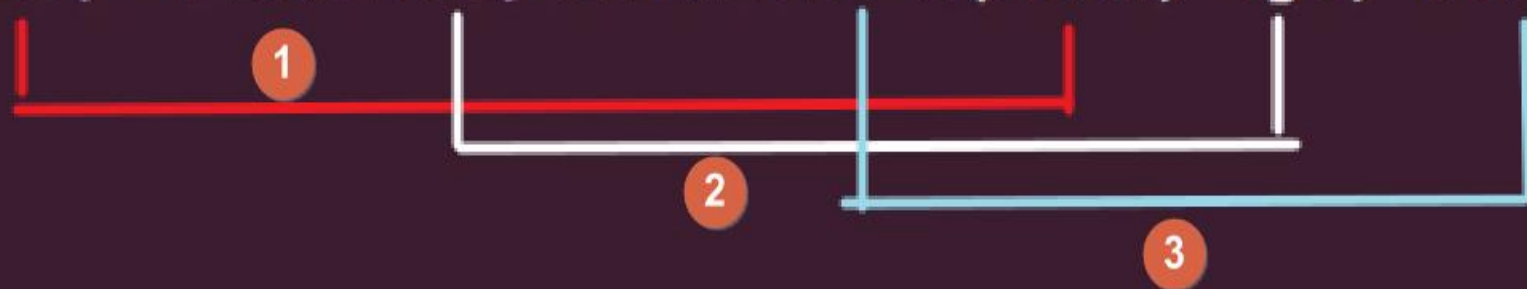
```
>>> print(str) # 输出变量
```

Print格式化输出:

```
1 print("hello world")
2
3 age = 18
4 print("age变量里的值是%d"%age)
5
6 name = "东哥"
7 print("名字是:%s"%name) I
```

Print格式化输出:

```
1 name = "laowang"  
2 age = 20  
3 addr = "山东....."  
4  
5 print("姓名是:%s, 年龄是:%d, 地址是:%s"%(name, age, addr))
```



三个变量，按先后顺序，一一对应，
输出信息

执行细节:

```
⑥ print('What is your age?') # ask for their age
   myAge = input()
   print('You will be ' + str(int(myAge) + 1) + ' in a year.')
```

The diagram illustrates the execution of the print statement for an input age of 4. It shows a sequence of seven steps, each with a curved arrow pointing from the previous step to the next, indicating the order of evaluation from left to right. The steps are:

- 1. `print('You will be ' + str(int(myAge) + 1) + ' in a year.')`
- 2. `print('You will be ' + str(int('4') + 1) + ' in a year.')`
- 3. `print('You will be ' + str(4 + 1) + ' in a year.')`
- 4. `print('You will be ' + str(5) + ' in a year.')`
- 5. `print('You will be ' + '5' + ' in a year.')`
- 6. `print('You will be 5' + ' in a year.')`
- 7. `print('You will be 5 in a year.')`

一些常犯错误：

```
>>> '42' + 3
❶ Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    '42' + 3
❷ TypeError: Can't convert 'int' object to str implicitly
>>>
```

The error message ❷ appears because Python couldn't understand your instruction. The traceback part ❶ of the error message shows the specific instruction and line number that Python had trouble with. If you're not sure what to make of a particular error message, search for it online. Enter “**TypeError: Can't convert 'int' object to str implicitly**” (including the quotes) into your favorite search engine, and you should see tons of links explaining what the error message means and what causes it, as shown in Figure 0-2.



"TypeError: Can't convert 'int' object to str implicitly"



Web

Shopping

Images



News

Videos

More ▾

Search tools

About 2,100 results (0.31 seconds)

python - TypeError: Can't convert 'int' object to str implicitly ...  

[stackoverflow.com/.../typeerror-cant-convert-int-object-to-str-implicitly ▾](#)

Nov 30, 2012 - You cannot concatenate a string with an int. You would need to convert your int to string using str function, or use formatting to format your output.

TypeError: Can't convert 'int' object to str implicitly error python  

[stackoverflow.com/.../typeerror-cant-convert-int-object-to-str-implicitly-... ▾](#)

Sep 22, 2013 - As the error message say, you can't add int object to str object. >>> 'str' + 2 Traceback (most recent call last): File "<stdin>", line 1, in <module> ...

```
>>> 5 +
```

```
File "<stdin>", line 1
```

```
5 +  
  ^
```

```
SyntaxError: invalid syntax
```

```
>>> 42 + 5 + * 2
```

```
File "<stdin>", line 1
```

```
42 + 5 + * 2  
          ^
```

```
SyntaxError: invalid syntax
```

```
>>> 'Alice' * 'Bob'
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#32>", line 1, in <module>
```

```
    'Alice' * 'Bob'
```

```
TypeError: can't multiply sequence by non-int of type 'str'
```

```
>>> 'Alice' * 5.0
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#33>", line 1, in <module>
```

```
    'Alice' * 5.0
```

```
TypeError: can't multiply sequence by non-int of type 'float'
```

```
>>> print('I am ' + 29 + ' years old.')
```

```
Traceback (most recent call last):
```

```
File "<pyshell#6>", line 1, in <module>
```

```
    print('I am ' + 29 + ' years old.')
```

```
TypeError: can only concatenate str (not "int") to str
```

章节练习1:

- 摄氏度转换为华氏度:
- 公式: 华氏度 = 摄氏度 $\times 9/5 + 32$
- 流程图:
- 输入摄氏温度 (C) :C,input ,float
- 计算华氏温度 (F) : $F=C\times 9/5+32$
- 输出华氏温度 (F) :print

30度摄氏度 = 86华氏度

```
1 # 获取用户输入摄氏温度
2 c = float(input('请输入摄氏温度: '))
3
4 # 计算华氏温度
5 f = c * 9/5 + 32
6
7 # 输出结果
8 print(c,"---",f)
9 print('%.1f摄氏度等于%.1f华氏度' % (c, f))
```

```
1 # 获取用户输入摄氏温度
2 c = float(input('请输入摄氏温度: '))
3
4 # 计算华氏温度
5 f = c * 9/5 + 32
6
7 # 输出结果
8 print(c, "---", f)
9 print('%.1f摄氏度等于%.1f华氏度' % (c, f))
```

Shell ×

```
>>> %Run '摄氏转华氏温度new.py'
```

请输入摄氏温度: 36

36.0 --- 96.8

36.0摄氏度等于96.8华氏度

```
>>>
```


章节练习2:

- 华氏温度 转换 摄氏温度。
- 公式：摄氏温度 = (华氏温度 - 32)*9 / 5
- 流程图：
- 输入华氏温度 (F) :F,input ,float
- 计算摄氏温度 (C) : $C = (F - 32) * 9 / 5$
- 输出摄氏温度 (C) :print
- **同学们自己完成吧**

问题? ? ?