浙江理工大学
ZHEJIANG SCI-TECH UNIVERSITY

2. 变量

《Python程序设计》 / 教师：罗志一

School of Computer Science and Technology
计算机科学与技术学院

概览

- 变量
- 基本数据类型
- 表达式
- 语句

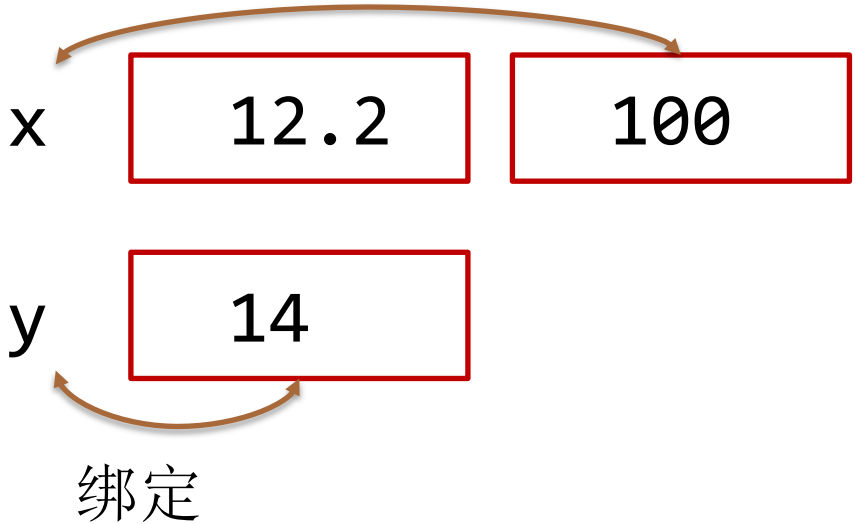
变量（名称）

- 变量是内存中的一个命名位置，程序员可以在其中存储数据，然后使用变量的“名称”检索数据。
- 程序员可以自行选择变量的名称（参见标识符的定义原则）
- 可以在后续语句中更改变量的内容

名称绑定

- 名称绑定是名称与对象（值）之间的关联。因此，在Python中，我们将一个“名称”绑定（或附着）到一个“对象”上。
- 将名称绑定到对象的常见方式是使用赋值操作符（=）。

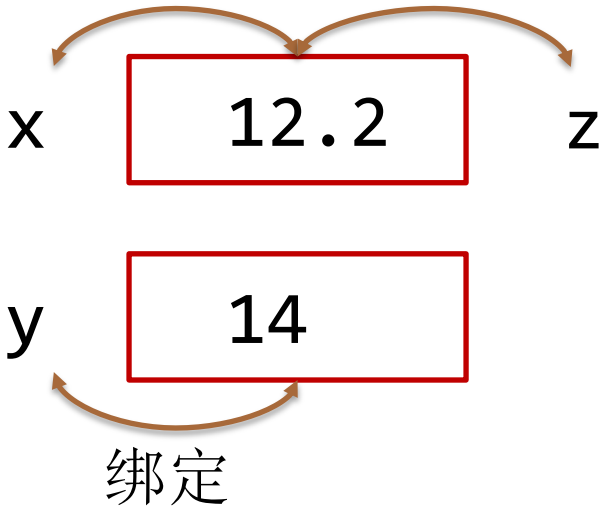
```
x = 12.2
y = 14
x = 100
```



名称绑定

- 当我们在=运算符的右侧使用一个名称时会发生什么？
- 例如，z = x 表示将名称 z 绑定到与名称 x 绑定的对象。

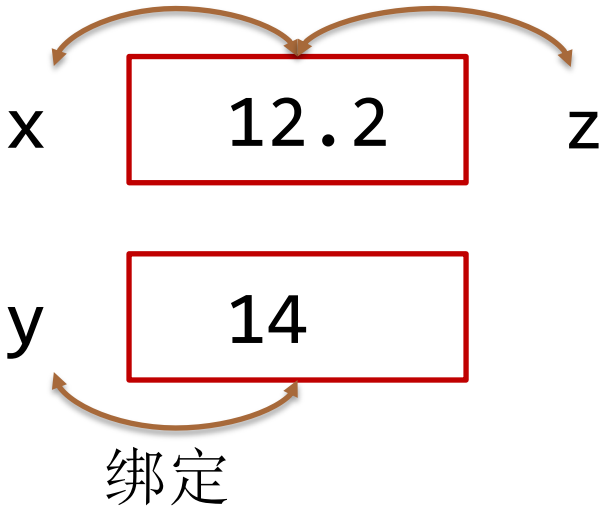
```
x = 12.2
y = 14
z = x
```



名称绑定

- 当你在Python解释器中输入一个名称（并回车）时，它会返回绑定到该名称的对象。
- 试试看吧！

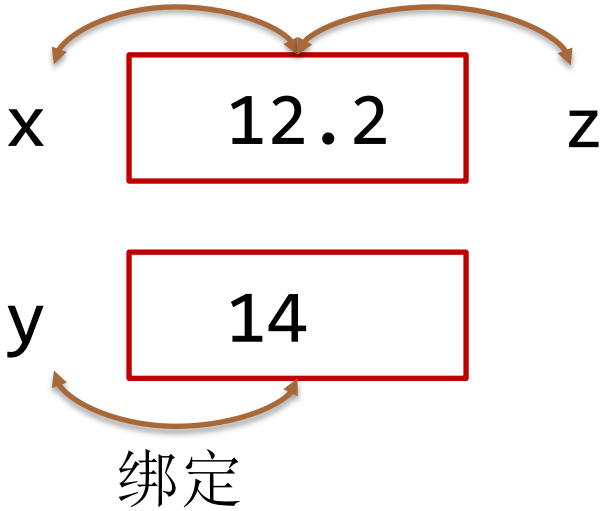
```
x = 12.2
y = 14
z = x
```



名称绑定

- Python创建了一个名称 z 并将其绑定到对象 12.2，该对象是名称 x 绑定的对象。请注意，Python并没有创建或复制对象 12.2，它只是为该对象引入了一个新的名称。因此，名称 x 和名称 z 都绑定到对象 12.2。
- 请记住，一个名称只能绑定到一个对象，而不能绑定到一个名称。

```
x = 12.2
y = 14
z = x
```



多变量赋值（绑定）

```
>>> x,y=4,8
>>> print(x,y)
    4 8
```

例 交换a,b值

```
a = int(input())
b = int(input())
print(a,b)
a,b = b,a    #a和b交换
print(a,b)
```

内置转换函数

```
>>>int(3.6)
    3
>>>float(3)
    3.0
```

赋值和“+”、“-”、“*”和“\”组合

```
>>>i=2
>>>i*=3
>>>i
6
>>>j=5
>>>j*=3+1    # j=j*(3+1)
>>>j
20
```

对象三要素

- id,对象存储的位置
- type,对象类型
- value,对象的值

对象的可变性

- 可变对象：对象的值可变，修改它的值对象的id不变
- 不可变对象：对象的值不可变，修改它的值对象的id要变，即创建另一个对象

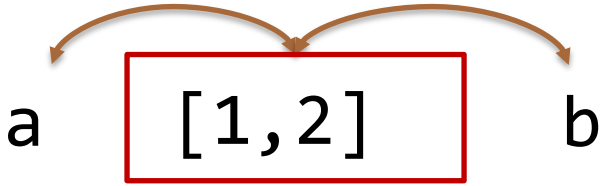
```
>>> a=5;lst=[3,9,78]
>>> print(id(a),id(lst))
    1549134400 55838560
>>> a=8;lst[1]=45
>>> print(id(a),id(lst))
    1549134448 55838560
```

数字、字符串是不可变对象，列表是可变对象

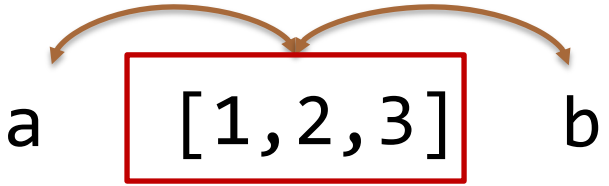
修改一个对象

- 修改可变对象：直接原地修改（不改变对象的id）

```
a = [1, 2]
b = a
```




```
a.append(3)
```



用名称检索其绑定的对象

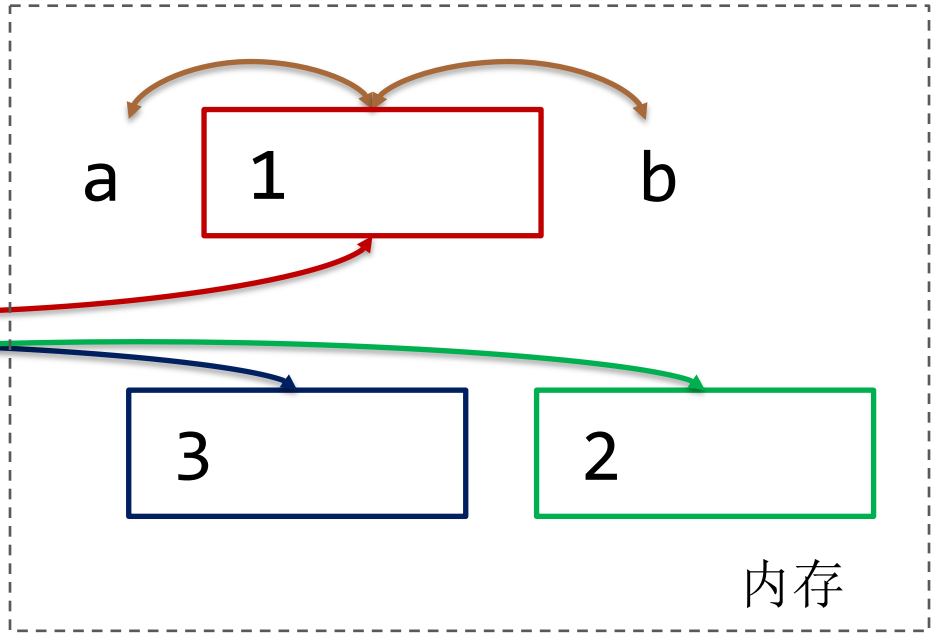
修改一个对象

```
Python 3.8.5 (default, Sep  4 2020, 02:22:02)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> a = [1, 2]
>>> b = a
>>> id(a)
140432277687744
>>> id(b)
140432277687744
>>> a.append(3)
>>> id(a)
140432277687744
>>> id(b)
140432277687744
```

修改一个对象

- 修改不可变对象（例如：整数和浮点数）：重新绑定

```
a = 1
b = a
a = a + 2
```



用名称检索其绑定的对象

修改一个对象

```
Python 3.8.5 (default, Sep  4 2020, 02:22:02)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>> a = 1
>>> b = a
>>> id(a)
4302481760
>>> id(b)
4302481760
>>> a = a + 2
>>> id(a)
4302481824
```

基本数据类型

	Type	Mutable	Examples
Numbers	bool	No	True, False
	int	No	13, 256,1024
	float	No	1.21, 3.14, 2e-7
	complex	No	5+9j
String	str	No	“hello”, ‘Jack’
Bytes	bytes	No	b‘ab\xff’
List	list	Yes	[‘Winken’, ’Blinken’, ’Nod’]
Tuple	tuple	No	(2, 4, 8)
Dictionary	dict	Yes	{“name”: “Jack”, “age”: 18}
Set	set	Yes	Set([3, 5, 7])

数字类型

- 整数
- 浮点数
- 复数

整数

```
>>>66
   66
>>>-175
   -175
>>>07
   SyntaxError: invalid token
>>>0
   0
```

二进制、八进制、十六进制

- 0b 或 0B 代表二进制
- 0o 或 0O 代表八进制
- 0x 或 0X 代表十六进制

```
>>>0b10
    2
>>>0o10
   8
>>>0x10
   16
```

整数可以表示很大的数

```
>>>google=10**50
>>>google
100000000000000000000000000000000000000000000000000
```

浮点数

- 浮点数也就是小数

  1.23，3.14，-9.01

- 科学计数法：

  1.23x109就是1.23e9，

  0.000012可以写成1.2e-5。

- "e"的前后都不能空，"e"的后面要整数。

浮点数运算

- 浮点数运算有误差

```
>>>2.1-2.0==0.1
False
>>>3.8/0.7
5.428571428571429
```

浮点数的整除还是浮点数

```
>>>3.8//0.7
5.0
>>>3.8%0.7
0.30000000000000004
```

复数

- 所谓复数，就是由实部（real）和虚部（imaginary）两部分组成的数，虚部用j表示。

```
2+3j
8j
(7+1j)*1j
```

- real方法取实部，imag方法取虚部，complex() 函数用于创建一个值为 real + imag * j 的复数。

```
>>> complex(1,2)
(1+2j)
>>> x = complex(1,2)
>>> x.real
1.0
>>> x.imag
2.0
```

数学库（math）

- math库是一个数学库，包含了很多的数学常数和数学函数
- 要使用math库，先要用"import math"语句引入math库

```
>>> import math
>>> math.pi
3.141592653589793
>>> math.sqrt(9)
3.0
```

标识符

- 标识符：给变量命名的名称
- 标识符是由字符、数字和下划线组成的组合，其中字符包括小写字母（a-z）、大写字母（A-Z）、数字（0-9）和下划线（_）
- 标识符不能以数字开头。如果标识符以数字开头，将会产生语法错误。
- 特殊符号如！、@、#、$、%等在标识符中不允许出现
- Python标识符不能只包含数字
- 标识符的长度没有限制
- 标识符名称区分大小写

关键字：保留字

- 不允许将保留字用作变量名/标识符

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

标识符举例

- **Python有效标识符示例**
  - abc123, abc_de, _abc, ABC, abc
- **Python无效标识符示例**
  - 123abc, abc@, 123, for

标识符

- 测试Python标识符的有效性
  - 使用 **str.isidentifier()** 函数来检查一个标识符的有效性，但此方法不考虑保留字。因此，我们可以将此函数**keyword.iskeyword()**一起使用来检查名称是否有效。

```
print("abc".isidentifier())
print("123abc".isidentifier())
print("_abc".isidentifier())
print("for".isidentifier())
```

标识符

- 测试Python标识符的有效性
  - 使用 **str.isidentifier()** 函数来检查一个标识符的有效性，但此方法不考虑保留字。因此，我们可以将此函数**keyword.iskeyword()**一起使用来检查名称是否有效。

```
import keyword
def is_valid_identifier(x):
    Return x.isidentifier() and not keyword.iskeyword(x)
print(is_valid_identifier("for"))
```

表达式

- 表达式是变量和运算符的组合，根据运算符优先级进行求值。
- Python表达式只包含标识符、字面值和运算符。
- 字面常量 (**literals**)
  - 字面值是某些内置类型的常量值的表示法。
  - 字符串字面值、字节字面值、整数、浮点数、复数
  - 例如：‘hello’, b‘spam’, 100, 3.14, 1+3j

字面常量可变还是不可变？

表达式

- 表达式是变量和运算符的组合，根据运算符优先级进行求值。
- Python表达式只包含标识符、字面值和运算符。
- 字面常量 (**literals**)
  - 字面值是某些内置类型的常量值的表示法。
  - 字符串字面值、字节字面值、整数、浮点数、复数
  - 例如：‘hello’, b‘spam’, 100, 3.14, 1+3j

不可变！

运算符

- 运算符是特殊符号，用于对一个或多个操作数（值）执行特定的操作，并返回结果。
- Python有多种运算符，可以对数据执行数学、逻辑和布尔运算。
- 算术运算符
  - +, -, *, /, // floor division, % modulus, ** exponent

运算符

- // floor division
  - 对商向下取整
  - 5//2, 9//4, -7//3, 3.0//2, 3.3//3
- % modulus
  - 取模： a = a%b + b*a//b 对商向下取整
  - 模 a%b 一定是个非负数
  - 8%2, 7%2, -7%3

运算符

- 关系运算符
  - 关系运算符在两个值之间进行比较。根据比较的结果，它返回布尔值True或False。

```
>>> 1<3<5                          #等价于1<3 and 3<5
True
>>> 3<5>2
True
>>> 1>6<8
False
>>> import math                    #sqrt是math 模块下的函数，导入math模块
>>> 1>6<math.sqrt(9)
False
```

运算符

- 关系运算符
  - 关系运算符在两个值之间进行比较。根据比较的结果，它返回布尔值True或False。
- 赋值运算符
  - =, +=, -=, *=, /=, %=, //=, **=

运算符

- 逻辑运算符/布尔运算符用于检验条件是否为真。Python有三个逻辑运算符。所有的逻辑运算符根据使用的条件返回布尔值True或False。
  - 逻辑与 **and**: 当两个操作数都为True时返回True
  - 逻辑或 **or**: 当其中一个操作数为True时返回True
  - 逻辑非 **not**: 当操作数为False时返回True

运算符

- 成员运算符
  - Python的成员运算符用于检查对象在序列（如字符串、列表、元组）中的成员关系。它检查给定的值或变量是否存在于给定的序列中。如果存在，则返回True，否则返回False
  - **in, not in**
- 身份运算符
  - 用于检验两个变量是否具有相同的内存地址
  - **is, is not**

运算符

- 位运算符
  - 在Python中，位运算符用于对整数执行位运算操作。要执行位运算，首先需要将整数值转换为二进制（0和1）值。
  - 位运算符逐位操作值，因此被称为“位运算”。它始终以十进制格式返回结果。Python共有6个位运算符，如下所示。
  - & 位与, | 位或, ^ 位异或, ~ 位取反, << 按位左移, >>按位右移
  - 例如，& 在将整数转换为二进制值后对整数值执行逻辑与操作，并将结果以十进制值返回。

运算符优先级和结合性

- 在Python中，运算符的优先级和结合性在解析表达式时起着至关重要的作用。我们必须了解该运算符的优先级（优先顺序）以及它们将如何计算得出一个单一的值。
- 运算符优先级用于在表达式中确定首先执行哪个操作。

Precedence level	Operator	Meaning
1 (Highest)	()	Parenthesis
2	**	Exponent
3	+x, -x ,~x	Unary plus, Unary Minus, Bitwise negation
4	*, /, //, %	Multiplication, Division, Floor division, Modulus
5	+, -	Addition, Subtraction
6	<<, >>	Bitwise shift operator
7	&	Bitwise AND
8	^	Bitwise XOR
9	|	Bitwise OR
10	==, !=, >, >=, <, <=	Comparison
11	is, is not, in, not in	Identity, Membership
12	not	Logical NOT
13	and	Logical AND
14 (Lowest)	or	Logical OR

运算符优先级和结合性实例

```
>>> 3+5 * 4   #先乘后加
23
>>> 5 * 3/2   #从左向右
7.5
>>> 2**3**2   #从右向左
512
>>> 3<5 or a>3 #从左向右
True
```

语句

- Python语言常用的有赋值、if语句和for语句。语句通常是一行一条语句。如一行中有多条语句，则用分号（；）分开，如语句太长要跨行时，可以用续行符（\）跨行表示一个语句。
- 赋值语句

  赋值语句用于将名称绑定到特定对象（值）

  基本形式是“变量=值”的形式

```
x = 2              Assignment Statement
x = x + 2          Assignment with expression
x += 2
print(x)           print statement
```

if语句

```
if 逻辑表达式:
        语句块1
else:
        语句块2
```

```
x=int(input())
if x%2==0:
    print("偶数")
else:
    print("奇数")
```

Quiz: 计算水费

- 为鼓励居民节约用水，自来水公司采取按用水量阶梯式计价的办法，居民应交水费y（元）与月用水量x（吨）相关：当x不超过15吨时，y=4x/3；超过后，y=2.5x-17.5，小数部分保留2位。请编写程序实现水费的计算。