

# 第 2 章 Python 语言基础

南通职业大学 朱亚林



# 内容提要

01

标识符、常  
量与变量

02

运算符与  
表达式

03

基本输入与  
输出方法

# 1. 标识符、常量与变量

标识符：

- 标识符是在程序中用来标识诸如变量、函数、类、对象等名字的符号
- Python规定，标识符只能由字母、数字和下划线组成，且必须由字母或下划线开头，不能和关键字同名。
- Python中大小写敏感（代表不同字符）
- 合法的标识符：
  - A , x1 , \_123 , name , abc
- 不合法的标识符：
  - G.U.I , 2end , for , from#12



# 1. 标识符、常量与变量

Python关键字（35个）：

```
>>> import keyword  
>>> print(keyword.kwlist)
```

```
['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']
```

# 标识符的命名规则

01

Python是大小写敏感的语言，  
大写字母和小写字母被认为  
是不同的字符

02

一般情况下，经常用小写字  
母对变量、对象和函数进行  
命名

03

标识符命名时尽可能做到见  
名知义，常选用英文单词或  
拼音缩写的形式

04

应尽量避免用易混淆单个字  
符作为标识符，如数字0和字  
母O

05

关键字具有特殊意义，不可  
以再用来对自定义标识命名

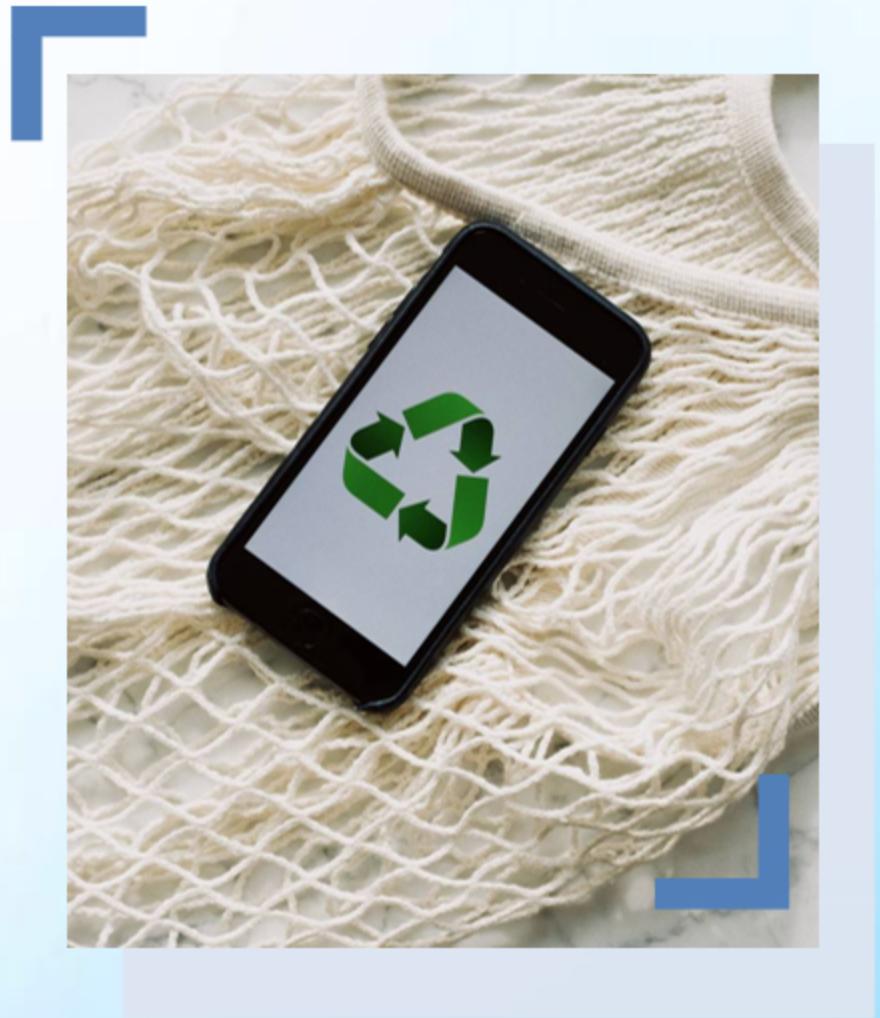
06

Python中使用下划线\_表示  
上次运算的结果

# 1. 标识符、常量与变量

常量：

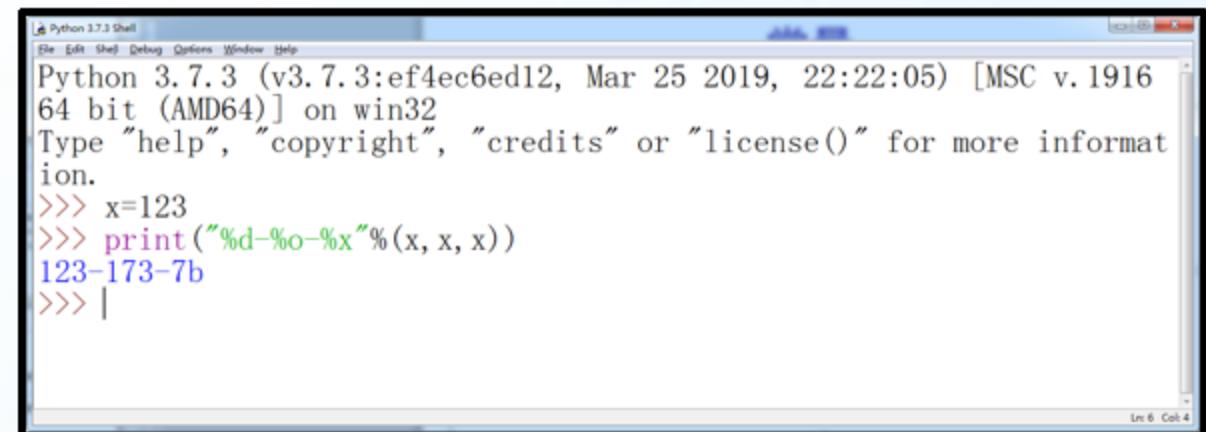
- 整型常量
- 实型常量
- 字符型常量
- 布尔型常量
- 复数型常量



# 常量

整型常量：

- 十进制形式：123
- 八进制形式：数码为0~7，以0o或0O开头，通常是无符号数。 0o123（十进制为83）
- 十六进制形式：数码为0~9，A~F（或a~f，代表10~15），以0x或0X开头。 0x123（十进制为291）



The screenshot shows a Windows-style window titled "Python 3.7.3 Shell". The window title bar includes "File Edit Shelf Debug Options Window Help" and a progress bar. The main area displays the Python interpreter's startup message and a command-line session:

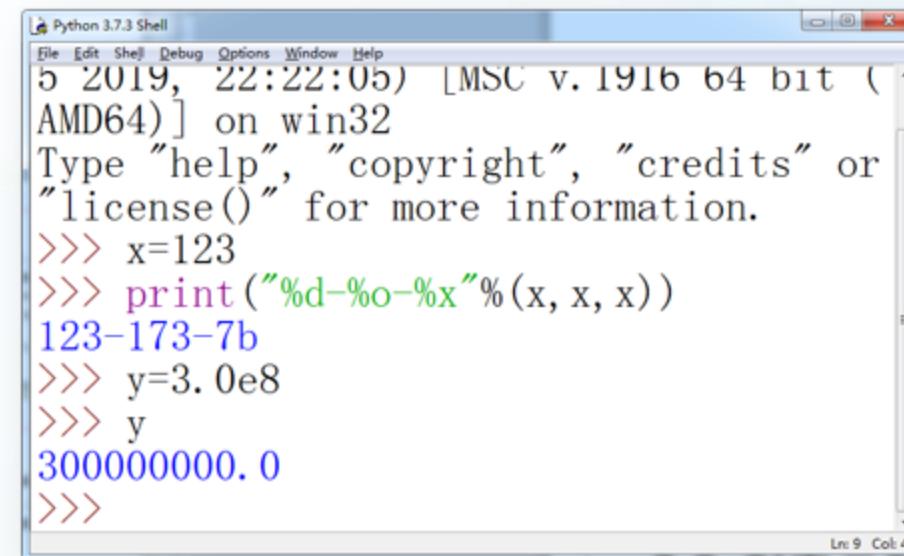
```
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916  
64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> x=123  
>>> print("%d-%o-%x"%(x, x, x))  
123-173-7b  
>>> |
```

The output shows the decimal value 123 followed by its octal representation (173) and hexadecimal representation (7b).

# 常量

## 实型常量：

- 常规形式：如1.23， -6.78
- 指数形式：采用字母e或E连接两个数，要求字母E前后均要有数，且E之后为整数。这种形式在数学上称为科学计数法。如：3.0e8(表示 $3.0 \times 10^8$ )



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
5 2019, 22:22:05) [MSC v. 1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or
"license()" for more information.
>>> x=123
>>> print("%d-%o-%x"%(x, x, x))
123-173-7b
>>> y=3. 0e8
>>> y
300000000. 0
>>>
```

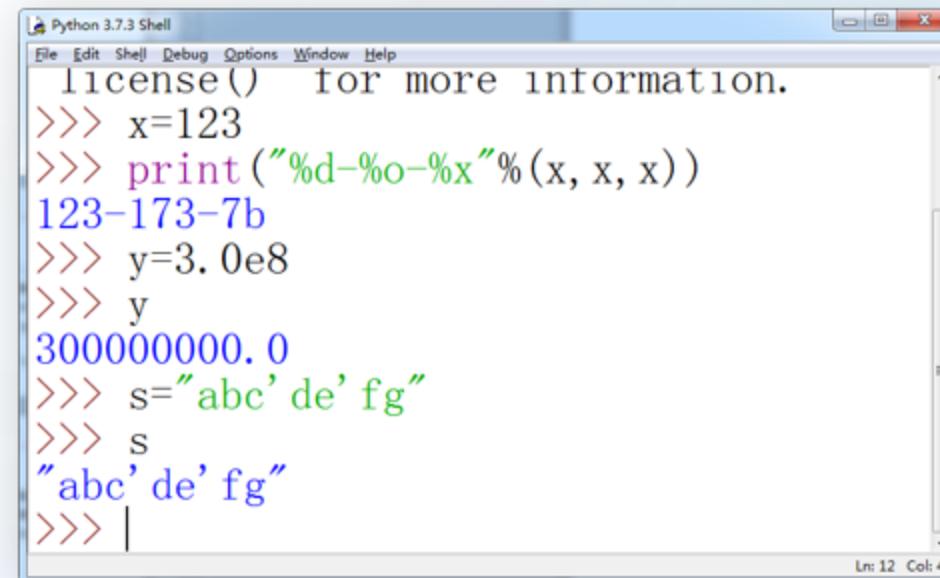
The screenshot shows the Python 3.7.3 Shell window. It displays the Python version and architecture, followed by a help message. Then, it shows two print statements. The first print statement uses the format string "%d-%o-%x" with three arguments (x, x, x). The output is "123-173-7b", where 123 is in decimal, 173 is in octal, and 7b is in hexadecimal. The second print statement sets a variable y to 3.0e8 and then prints it, resulting in 300000000.0.

# 常量

## 字符串常量：

用一对单引号、双引号或三引号进行字符串的表示，其中单引号和双引号引起的字符串需在一行内写完，而三引号引起的字符串可以是多行的。

如：“Hello World”， “abc'de'fg”，  
'abc"de"fg'



The screenshot shows a Python 3.7.3 Shell window. It displays the following code and its output:

```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
license() for more information.

>>> x=123
>>> print("%d-%o-%x"%(x, x, x))
123-173-7b
>>> y=3.0e8
>>> y
300000000.0
>>> s="abc' de' fg"
>>> s
"abc' de' fg"
>>> |
```

The window title is "Python 3.7.3 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. A status bar at the bottom right shows "Ln: 12 Col: 4". The code area contains several lines of Python code demonstrating different string representations and their printed forms.

# 常量

字符型常量：

- 转义字符：以“\”(反斜杠)开头的特殊的表示方法
  1. 转义字符常量'\n'、'\101'、'\x41'等只能表示一个字符；
  2. '\101'和'\0x41'均表示字符'A'
  3. 转义字符常量'\n'、'\101'、'\x41'等只能表示一个字符；
  4. '\101'和'\0x41'均表示字符'A'

字符形式	含 义
\n	回车换行，将当前位置移到下一行开头。
\t	横向跳到下一个制表位置，相当于 Tab 键。
\b	退格，将当前位置退回到前一列。
\r	回车，将当前位置移到当前行的开头。
\f	走纸换页，将当前位置移到下页开头。
\\\	表示反斜杠符“\”。
\'	表示单引号。
\"	表示双引号。
\ddd	1~3位八进制所代表的字符。
\xhh	1~2位十六进制所代表的字符。

The screenshot shows a Python 3.7.3 Shell window. The user has typed:

```
>>> c1='\'101', c2='\'x41'  
SyntaxError: can't assign to literal  
>>> c1='\'101'  
>>> c2='\'x41'  
>>> print(c1, c2)  
A A  
>>> c1, c2  
(A, A)  
>>>
```

The command `c1='\'101'` results in a `SyntaxError: can't assign to literal`. Subsequent assignments to `c1` and `c2` without the leading backslash are successful, and the final `print` command outputs `A A`.

# 1. 标识符、常量与变量

## 布尔型常量：

只有两个：真（True）和假（False），书写时注意区分大写。这两常量一般用于描述逻辑判断的结果，如关系表达式或逻辑表达式。

```
>>> print(5>3)
```

```
True
```

```
>>> print(5<3)
```

```
False
```

# 常量

- 复数型常量：

和数学上表示含义一样， Python中的复数也由实部和虚部组成， 形式为：  $a + bj$  或 `complex(a , b)`。如： $3 + 5j$ 。从 $z=a + bj$ 中提取实部和虚部， 可用`z.real`和`z.imag`方式。

```
>>> c=complex(3,5)
>>> c
(3+5j)
>>> c.real
3.0
>>> c.imag
5.0
>>> |
```

```
>>> d = 1 + 2j
>>> d.real
1.0
>>> d.imag
2.0
>>> |
```

# 变量

- 变量结构:

对于Python而言，一切变量都是**对象**，  
变量的存储，采用了引用语义的方式，  
变量存储的只是一个变量的内存地址，  
而不是这个变量的值本身。Python解释  
器会为每个变量分配大小一致的内存，  
用于保存变量引用对象的地址。



# 变量

## 变量赋值：

01

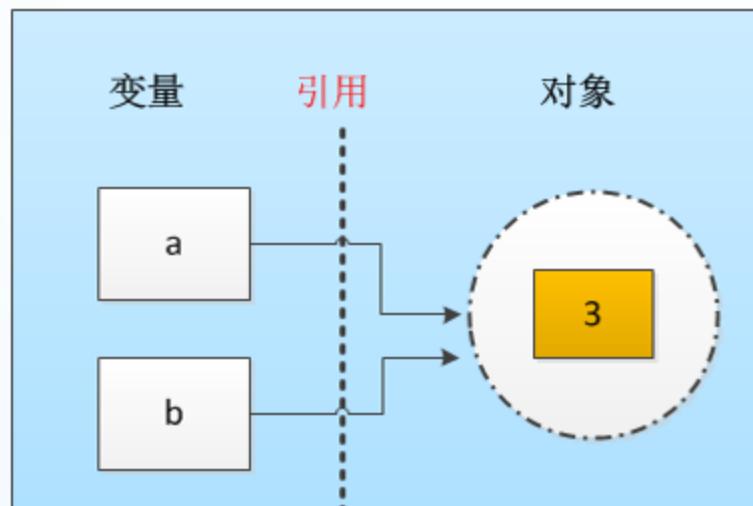
变量一旦被赋值，就完成了定义和创建过程。

02

Python允许为多个变量同时赋值，如：

- `a,b,c=1,2, "Python"`
- 表示两个整数1和2分别配送给变量a和b，字符串"Python"配送给变量c。。

# 变量



```
>>> a=3  
>>> b=3  
>>> id(a)  
8791303021264  
>>> id(b)  
8791303021264
```

在python中，一切都是对象，一切都是对象的引用

如下图所示，Python将执行三个步骤来完成`a = 3`的赋值操作：

1. 创建变量a；
2. 创建一个对象(分配一块内存)，来存储值3；
3. 将变量与对象，通过指针连接起来，从变量到对象的连接称之为引用(变量引用对象)；
4. 后面再创建变量b来存储3，则依然指向同一个对象3。

# 变量

”

## 特殊变量：

主要是指以下划线作为变量名前缀或后缀的变量。

1. `_xxx`形式的变量：以单下划线开头的变量表示变量是私有的，模块或类外不允许使用。
2. `__xxx`形式的变量：以双下划线开头的成员变量表示类的私有变量，只有类对象自己能访问，连子类对象也不能访问到这个数据。
3. `__xxx__`形式的标识符：表示系统定义的专用标识，如`__init__()`代表类的构造函数。



## 2. 运算符与表达式

Python运算符主要分为以下类别：

1. 算术运算符：+、-、\*、/、%、\*\*、//
2. 关系运算符：<、<=、>、>=、==、!=
3. 逻辑运算符：and 、or、 not
4. 赋值运算符：=、复合赋值运算符
5. 位运算符：&、|、^、~、<<、>>
6. 成员运算符：in 、not in
7. 身份运算符：is 、is not

# 算术运算符

名称	运算符	功    能	实    例	优先级
加	<code>+</code>	两个对象相加	<code>x+y</code> , 结果为 7	优先级相同, 比下面的运算符优先级低
减	<code>-</code>	得到负数或实现两数相减	<code>x-y</code> , 结果为 3	
乘	<code>*</code>	两数相乘或者返回一个被重复若干次的字符串	<code>x*y</code> , 结果为 10	
除	<code>/</code>	两个对象相除, 实现 <code>x</code> 除以 <code>y</code>	<code>x/y</code> , 结果为 2.5	
求余	<code>%</code>	取模运算, 求 <code>x</code> 除以 <code>y</code> 的余数, 符号同 <code>y</code>	<code>x%y</code> , 结果为 1 <code>5%-2</code> , 结果为 -1	优先级相同, 比上面的运算符优先级高
幂	<code>**</code>	幂运算, 返回 <code>x</code> 的 <code>y</code> 次幂	<code>x**y</code> , 结果为 25	
整除	<code>//</code>	整除运算, 返回商的整数部分(向下取整)	<code>9//2</code> 的结果为 4 <code>9.0//2.0</code> 的结果为 4.0	

```
>>> x,y=3,4
>>> x+y
7
>>> x/y
0.75
>>> x//y
0
>>> x%y
3
>>> x**y
81
>>> |
```

# 关系运算符

名称	运算符	功    能	实    例	优先级
小于	<	用于判断 $x$ 是否小于 $y$	$x < y$ ,结果为 False	
小于等于	$\leq$	用于判断 $x$ 是否小于等于 $y$	$x \leq y$ ,结果为 False	
大于	>	用于判断 $x$ 是否大于 $y$	$x > y$ ,结果为 True	
大于等于	$\geq$	用于判断 $x$ 是否大于等于 $y$	$x \geq y$ ,结果为 True	
等于	$\equiv$	用于判断 $x$ 是否等于 $y$	$x \equiv y$ ,结果为 False	
不等于	$\neq$	用于判断 $x$ 是否不等于 $y$	$x \neq y$ ,结果为 True	

```
>>> 8 > 6  
True  
>>> 8 < 6  
False  
>>> 8 == 6  
False  
>>> 8 != 6  
True  
>>> 8 == 8  
True
```

# 逻辑运算符

名称	运算符	功 能	实 例	优先级
逻辑非	not	单目运算符，用于返回 not x 的结果，x 为非 0 值时，结果为 False，x 为 0 值时，结果为 True。	not y, 结果为 False	高 ↑ ↓ 低
逻辑与	and	双目运算符，用于返回 x and y 的结果，只有 x 和 y 的值均为非 0 值时，结果才是 True，其它情况结果全为 False。	x and y, 结果为 True	
逻辑或	or	双目运算符，用于返回 x or y 的结果，只有 x 和 y 的值均为 0 值时，结果才是 False，其它情况结果全为 True。	x or y, 结果为 True	

X <sup>+</sup>	y <sup>+</sup>	not x <sup>+</sup>	x and y <sup>+</sup>	x or y <sup>+</sup>
真 <sup>+</sup>	真 <sup>+</sup>	假 <sup>+</sup>	真 <sup>+</sup>	真 <sup>+</sup>
真 <sup>+</sup>	假 <sup>+</sup>	假 <sup>+</sup>	假 <sup>+</sup>	真 <sup>+</sup>
假 <sup>+</sup>	真 <sup>+</sup>	真 <sup>+</sup>	假 <sup>+</sup>	真 <sup>+</sup>
假 <sup>+</sup>	假 <sup>+</sup>	真 <sup>+</sup>	假 <sup>+</sup>	假 <sup>+</sup>

# 赋值运算符

## 一. 基本赋值运算符

- 格式：变量 = 表达式

## 二. 复合赋值运算符

- Python语言中，基本赋值运算符用“=”与7种算术运算符（+、-、\*、/、%、\*\*、//）和5种位运算符（&、|、^、<<、>>）结合成12种复合赋值运算符，其功能是先完成算术或位运算，然后再赋值。例：

a+=b 等价于 a=a+b

a-=b 等价于 a=a-b

a\*\*=b 等价于 a=a\*\*b

a//=b 等价于 a=a//b

a&=b 等价于 a=a&b

```
>>> x,y = 8,6
>>> x+=2
>>> x
10
>>> x//=y
>>> x
1
```

# 成员运算符

Python的成员运算符用于验证给定的值在指定范围内是否存在，分别是in和not in，运算规则如表所示。

名称	运算符	功    能
存在	in	给定的值在指定范围内则返回 True，否则返回 False
不存在	not in	给定的值不在指定范围内则返回 True，否则返回 False

```
>>> list = [1,2,3,4,5]
>>> x = 2
>>> x in list
True
>>> x not in list
False
```

# 身份运算符

Python的身份运算符用于测试两个变量是否引用同一个对象，分别是is和is not，运算规则如表所示。

名称	运算符	功    能
是	is	双目运算符，判断两个变量是否引用同一个对象，若是则返回 True，否则返回 False
不是	is not	双目运算符，判断两个变量是否引用同一个对象，若不是则返回 True，否则返回 False

```
>>> x,y = 8,8  
>>> x is y  
True  
>>> x is not y  
False
```

# 运算符优先级 和结合性

常用运算符的优先级和结合性

优先级 ↑	运算符	结合性
高 ↑ 	( )	从左至右 ↗
	**	
	~、 +、 -	
	*、 /、 %、 //	
	+、 -	
	<<、 >>	
	&	
	^、	
	<、 <=、 >、 >=	
	==、 !=	
	=、 +=、 -=、 *=、 /=、 %=、 **=、 //=	从右至左 ↘
	is、 is not	从左至右 ↗
	in、 not in	
	not	从右至左 ↘
	and	从左至右 ↗
	or	

# 3. 基本输入与输出方法

## 数据输出

### 一. print()函数

基本格式如下：

- `print([obj1,...][,sep=' '][,end='\n'][,file=sys.stdout])`
- 例：

```
>>> print(123,'abc',456,'def',sep='#' )  
123#abc#456#def
```

`print(格式控制字符串%(输出项1, 输出项2, ..., 输出项n))`

- 例：

```
print("a=%d,b=%d"%(a,b))
```

指定输出结尾符号

```
print('area',end='=')
```

### 二. format()函数

基本语法是通过 {} 和 : 来代替以前的 % 。

```
print("{1} {0} {1}".format("hello", "world")) # 设置指定位置  
world hello world
```



### 3. 基本输入与输出方法

Python常用格式说明符如表所示

格式符	功能说明
d或i	以带符号的十进制整数形式输出（正数省略符号）
o	以八进制无符号整数形式输出整数（不输出前导0）
x或X	以十六进制无符号整数形式输出整数（不输出前导0x）。
c	以字符形式输出一个字符
s	以字符串形式输出
f	以小数形式输出实数，默认带6位小数
e或E	以标准指数形式输出实数，数字部分含1位整数、6位小数
g或G	根据给定的值和精度，系统自动选择f和e中较紧凑的格式输出
m	域宽，十进制整数，用于指定输出数据所占宽度。若m大于数据实际宽度，输出时前面补空格；若m小于实际宽度，按实际位数输出。 小数点占1位
n	附加域宽，十进制整数，用于指定实型数据的小数所占宽度。若n大于小数实际宽度，输出时小数部分后面补0；若n小于小数的实际宽度，输出时将小数部分多余的位按四舍五入处理。字符串则截取。
-	输出数据左对齐，默认为右对齐
+	输出正数时，以+开头
#	作为o、x的前缀，输出结果前面加上前导0、0x

# 基本输出 实例

```
>>> a=10
>>> print("a=%d,a=%o,a=%x"%(a,a,a))
a=10,a=12,a=a
>>> b=1.23
>>> print("b=%f"%b)
b=1.230000
>>> print("%4d-%02d-%02d"%(2023,9,8))
2023-09-08
>>> area=6.268
>>> print('area=%6.2f'%area)
area= 6.27
```

### 3. 基本输入与输出方法

#### 二. 数据输入

- 当用户想从计算机输入设备（如键盘）上读取数据时，Python 3.x提供了input()函数，其格式如下：
- input([prompt])
- 该函数返回的都是字符串，若需要输入数值，则需进行类型转换。
- 例：
- a,b = eval(input('输入两个数，逗号隔开：'))
- eval()函数将输入的字符串转换成数字

```
>>> a,b = eval(input('请输入两个数，逗号隔开：'))
请输入两个数，逗号隔开： 3,4
>>> a
3
>>> b
4
>>> a+b
7
>>> |
```

# 小结

”

标识符、常量与变量:介绍了标识的定义方法、常量的分类和变量的用法;

1

运算符与表达式:  
介绍了七类运算符的运算规则;

2

基本输入与输出方法:介绍了Python数据在交互环境下的输入和输出方法。

3



# 练习

1.下列标识符合法的是\_\_\_\_。

- A、 var-name
- B、 !@#\$%
- C、 \_100
- D、 elif





# 练习

2.下面不属于Python保留字的是\_\_\_\_\_。

- A、 def
- B、 elif
- C、 type
- D、 import





# 练习

3. 已知 $x = 43, y = \text{False}$ ; 则表达式

$x >= y \text{ and } 'A' < 'B'$

的值是\_\_\_\_\_。

A、 False

B、 语法错

C、 True

D、 "假"





# 练习

4. 下面语句的输出结果是\_\_\_\_。

>>>-5//3

- A、1
- B、2
- C、-1
- D、-2





# 练习

5. 下面语句的输出结果是\_\_\_\_。

```
>>> '{:.4e}'.format(1234.56789)
```

- A、 '1.2345e+03'
- B、 '1234.5679'
- C、 '1.2346e+03'
- D、 '1.2345e+03'





# 练习

6. 关于数据输入及其处理，以下说法正确的是

\_\_\_\_\_。

- A、在Python中语句x,y=1是合法的
- B、input函数从控制台获得用户的一行输入，以输入值的类型返回
- C、在Python中语句x=y=z=1不合法
- D、print函数用于输出运算结果

