

Python 语言程序设计

第三章

基本数据运算与函数



基本数据类型

01

02

运算符与表达式

常用内置函数

03

04

常用标准函数



01 基本数据类型

数值类型

- 1. 整型 (int)

整型是带正负号的整数数据。Python 3.X中并不严格区分整型和长整型，且没有长度限制。整型数据类型表示的数值范围仅与计算机支持的内存大小有关，所以它几乎包括了全部整数范围，远远超过了其他高级语言中整型数据的表示范围，给数据计算带来了很大的便利。

Python整型数的表示方法有以下几种：

- (1) 十进制整数，如：10, 255, -16;
- (2) 二进制整数，以 0B或0b开头的的数据，如：11, 100;
- (3) 八进制整数，以0O或0o开头的的数据，如：0O67, 0o17;
- (4) 十六进制整数，以0X或0x开头的的数据。如：0X4a, 0Xff。

数值类型

- 2. 浮点型 (float)

浮点数表示实数数据，由整数部分、小数点和小数部分组成。使用下面的语句可以输出当前系统下浮点数所能表示的最大数max和表示的最小数min。

```
>>> import sys
>>> sys.float_info.max
1.7976931348623157e+308
>>> sys.float_info.min
2.2250738585072014e-308
>>>
```

Python中的浮点数的表示方法如下：

(1) 十进制小数表示法，如，3.14，10.0，0.0等，注意：这里的0.0不是0，0表示一个整数，而0.0表示一个浮点数。

(2) 科学计数表示法，用字母e（或E）表示以10为底数的指数，用XeY表示 $X \times 10^Y$ 。

数值类型

- 3. 复数型 (complex)

复数型数据用来表示数学中的复数，复数由实数部分和虚数部分所组成的数，形如 $x=a+bj$ 。其中 a 、 b 为浮点数， a 是复数的实部， b 是复数的虚部。 j 为“虚数单位”， j 的平方等于-1。可以使用 $x.real$ 和 $x.imag$ 获得复数 x 的实部和虚部。例如：

- 4. 数值类型间的转换

在进行算术运算时，Python会自动完成数值类型间的转换。当参加运算的数值均为整型时，结果为整型。当有浮点型参与运算时，结果为浮点型。例如：

```
>>> x=12.3+45j
>>> x
(12.3+45j)
>>> x.real
12.3
>>> x.imag
45.0
>>>
```

```
>>>
>>> 99+1                                #整数相加得整数
100
>>> 99+1.000                            #整数与实数相加得实数
100.0
>>> 99-1.0                              #整数与实数相减得实数
98.0
>>> 99*1.0                              #整数与实数相乘得实数
99.0
>>> 99/1.0                              #整数与实数相除得实数
99.0
>>> >>>
```

字符串类型

- Python语言中的字符串类型是用引号括起来的一个或多个字符。用单引号 (') 和双引号 (") 括起来的字符串必须是单行字符串，用三引号 (") 括起来的可以是多行字符串。需要注意的是，引号必须是英文标点，且三引号由三个单引号组成。
- 例如：下面语句定义了三个字符串变量

```
>>> str='God Wants To Check The Air Quality'
>>> str1="God Wants To Check The Air Quality"
>>> str2=''God
Wants To Check
The Air Quality'''
>>>
```

布尔类型

- 布尔型数据用来表示具有两个确定状态的数据，它有真（True）和假（False）两个值。布尔型数据在计算机中用1或0来存储，1代表逻辑真，0代表逻辑假。而且，Python中值为“空”的数据，如一个空字符串、一个空的元组等，它们的布尔值均为False。

```
>>> x=True
>>> int(x)
1
>>> y=False
>>> int(y)
0
>>>
```

- 关系型表达式或逻辑型表达式的值为布尔型，在程序中通常用来表示条件。布尔型数据可以参与算术运算。

```
>>> x=1
>>> y=2
>>> x>y
False
>>> x+(x>y)
1
>>>
```

02 运算符与表达式

算术运算符

运算符	描述	实例 (a=10,b=20)
+	加, 两个数据相加	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
//	整除, 返回商的整数部分	9//2 输出结果 4 9.0//2.0 输出结果 4.0

关系运算符

运算符	描述	实例 (a=10,b=20)
==	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。
>	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。	(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。

赋值运算符

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 $c = c ** a$
//=	取整赋值运算符	$c //= a$ 等效于 $c = c // a$

逻辑运算符

运算符	描述	实例(x=True,y=False)
and	与运算，仅当x=True且y=True时，x and y返回 True，其它情况均返回False。	(x and y) 返回False。
or	或运算，仅当x=False且y=False时，x or y返回False，其它情况均返回True。	(x or y) 返回True。
not	非运算，如果x=True，返回False。如果x= False，返回True。	not(x) 返回 False

位运算符

运算符	描述	实例(a=60,b=13)
&	按位与 (AND)：参与运算的两个值的两个相应位都为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 (OR)：参与运算的两个值的两个相应位有一个为1，则该位的结果为1；否则为0	(a b) 输出结果61, 二进制解释：0011 1101
^	按位异或 (XOR)：当两个对应的二进制位相异时，结果为1	(a ^ b) 输出结果49, 二进制解释：0011 0001
~	按位翻转/取反 (NOT)：对数据的每个二进制位取反，即把1变为0，把0变为1 由于用补码表示负数，所以 ~a= -(a+1)	(~a) 输出结果-61, 二进制解释：1100 0011, 是一个有符号二进制数的补码形式
<<	按位左移：运算数的各个二进制位全部左移若干位，高位丢弃，低位不补 0	a << 2 输出结果240, 二进制解释：1111 0000
>>	按位右移：运算数的各个二进制位全部右移若干位	a >> 2 输出结果15, 二进制解释：0000 1111

成员运算符

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。

身份运算符

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(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 。

身份运算符的实例:

```
>>> x,y=10,20
>>> x=y
>>> x is y           #说明x, y引用的是同一个数据对象20
True
>>>
```

表达式

运算符	描述
**	指数 (最高优先级)
~ + -	按位取反，正号和负号
* / % //	乘，除，取模和整除
+ -	加，减
>> <<	右移，左移运算符
&	位运算符
^	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 or and	逻辑运算符

表达式的运算实例，`not "Abc"=="abc" or 2+3>=5 and "23"<"3"`，
运算结果如下：

```
>>> not "Abc"=="abc" or 2+3>=5 and "23"<"3"  
True  
>>>
```

运算符优先级决定了运算的顺序，想要改变它们的运算顺序，
可以使用圆括号。

```
>>> not ("Abc"=="abc" or 2+3>=5 and "23"<"3")  
False  
>>>
```

03 常用内置函数

常用内置函数

abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	delattr()
hash()	memoryview()	set()		

常用内置函数

函数名	功能
abs(x)	求绝对值，参数可以是整型，也可以是复数
all(iterable)	元素都为真时，函数值为True；元素为空时，函数返回True
any(iterable)	元素有一个为真时，函数值为True；2、元素为空时，函数返回False
bin(x)	将整数x转换为二进制数
bool([x])	将x转换为布尔型Boolean类型
chr(x)	返回整数x对应的ASCII字符
complex([real[, imag]])	创建一个复数
dict([arg])	创建数据字典
dir([object])	返回当前范围内的变量或对象的详细列表
divmod(a, b)	返回一个由商和余数构成的元组，参数可以为整型或浮点型。
eval(str [,globals[,locals]])	将字符串str当成有效的表达式来求值并返回计算结果。
float([x])	将一个字符串或数转换为浮点数。如果无参数将返回0.0
format(value [, format_spec])	格式化输出字符串
help([object])	显示有关对象的帮助信息
hex(x)	将整数x转换为16进制数
id(object)	返回对象的唯一标识
input([prompt])	返回用户输入

常用内置函数

函数名	功能
<code>int([x[, base]])</code>	将一个数字或base类型的字符串转换成整数，base表示进制，默认为十进制
<code>len(s)</code>	返回集合长度
<code>list([iterable])</code>	生成一个列表
<code>locals()</code>	返回当前的变量列表
<code>max(iterable[, args...][key])</code>	返回集合中的最大值
<code>min(iterable[, args...][key])</code>	返回集合中的最小值
<code>oct(x)</code>	将一个数字转化为8进制数
<code>ord(s)</code>	返回单个字符的ASCII码，返回值是一个整数。
<code>open(name[, mode[, buffering]])</code>	打开文件
<code>pow(x, y[, z])</code>	返回x的y次幂
<code>print(value1, value2 ..., sep=' ', end='\n')</code>	按指定格式输出的函数
<code>range([start], stop[, step])</code>	产生一个序列，默认从0开始
<code>round(x[, n])</code>	将x进行四舍五入
<code>sorted(iterable[, cmp[, key[, reverse]])</code>	对集合排序
<code>str([object])</code>	转换为字符串类型
<code>sum(iterable[, start])</code>	对集合求和
<code>tuple([iterable])</code>	生成一个元组
<code>type(object)</code>	返回该对象的类型

函数实例

1.数值运算函数

abs(x)函数: 返回一个x的绝对值, x参数可以是整型或浮点型数据。

divmod(a,b): 返回由a / b的商和余数构成的一个元组, a、b可以是整型或浮点型。

pow(x, y[, z]): 返回x的y次幂, 如果有参数z则返回x的y次幂与z的模。

$\text{pow}(x, y, z) \rightarrow \text{pow}(x, y) \% z$

$\text{pow}(x, y) \rightarrow x^{**}y$

round(x[, n]): 对x进行四舍五入, n保留的小数位, 若省略则只保留整数。

```
>>> print(abs(10),abs(-10))
10 10
>>> print(divmod(10,4),divmod(10.5,2.5))
(2, 2) (4.0, 0.5)
>>> print(pow(2,3),pow(2,3,4))
8 0
>>> print(round(3.1415926,3),round(3.1415926))
3.142 3
>>>
```

函数实例

2. 数制转换函数

`int([x])`: 将十进制数值取整 (截去小数部分)

```
>>>int(3.14)           # 3
>>>int(2e2)            # 200
>>>int(100, 2)         # 出错, base 被赋值后函数只接收字符串
```

`int([x, base])`: 将 “base” 进制的合法整数字符串转换成十进制整数

```
>>>int('23')           # 23, base参数省略时默认为十进制数
>>>int('23',16)        # 35, 将'23'做为16进制数
>>>int('Pythontab',8)   # 出错, 'Pythontab'不是个8进制数
```

字符串 `0x` 视作十六进制的符号, `0b` 视作二进制的符号

```
>>>int('0x10', 16)     # 16, 0x是十六进制的符号
>>>int('0b10',2)       # 2 , 0b是二进制的符号
```

函数实例

2. 数制转换函数

`bin(x)`: 将十进制整数转换成二进制的数字字符串

`oct(x)`: 将十进制整数转换成八进制的数字字符串

`hex(x)`: 将十进制整数转换成十六进制的数字字符串

```
>>> b=255
>>> print(bin(b),hex(b),oct(b))
0b11111111 0xff 0o377
>>> a=0b10110011      #int() 函数可以将2进数转换成10进制
>>> print(int(a))
179
>>> print(int('123', 16),int('123',8),int('123'))
291 83 123
>>>
```

函数实例

3.类型转换函数

bool([x]): 返回x布尔值。

```
>>> x,y=0,1
>>> print(bool(x),bool(y))
False True
>>> print(bool('abc'),bool(''))
True False
>>>
```

#0是False, 1是True

#字符串为True, 空串为False

float([x]): 返回一个浮点数。

```
>>> float('+1.23')
1.23
>>> float('    -12345\n')
-12345.0
>>> float('1e-003')
0.001
>>> float('+1E6')
1000000.0
>>> float('-Infinity')
-inf
```

函数实例

3.类型转换函数

ord(): 返回单个字符的ASCII码, 返回值是一个整数。

chr(i): 返回整数i对应的字符。与ord()函数互为反函数。

```
>>> x='A'
>>> y=97
>>> ord(x)
65
>>> chr(y)
'a'
>>> chr(ord(x))
'A'
>>>
```

eval(str [,globals[,locals]]): 将字符串str当成有效的表达式来求值并返回计算结果。

```
>>> x = 1
>>> eval('x+1')
2
>>> a='5,10'
>>> x,y=eval(a)
>>> print(x,y)
5 10
>>>
```

函数实例

4.集合运算函数

all()函数：元素均为True或元素为空时返回True。

any()函数：元素至少一个为True时返回True，元素为空时返回False。

```
>>> print(all([1,2,3]),all([0,1,2]),all([]))
```

```
True False True
```

```
>>> print(any([1,2,3]),any([0,1,2]),any([]))
```

```
True True False
```

```
>>>
```

max(iterable[, args...][key])：返回集合元素中的最大值。

min(iterable[, args...][key])：返回集合元素中的最小值。

```
>>> mylist=[5,9,98,20,126]
```

```
>>> print(max(mylist),min(mylist))
```

```
126 5
```

```
>>>
```

range([start], stop[, step])：产生一个序列，默认从0开始。

list([iterable])：产生一个列表。

tuple([iterable])：产生一个元组。

```
>>> x=range(1,10,2)      #生成一个1<=x<10的序列，步长为2
```

```
>>> list(x)             #由序列生成一个列表
```

```
[1, 3, 5, 7, 9]
```

```
>>> tuple(x)            #由序列生成一个元组
```

```
(1, 3, 5, 7, 9)
```

```
>>>
```

函数实例

5.帮助函数

`help([object])`: 查看函数用法的详细信息。

```
>>> help(round)           #查看round函数的使用说明
```

```
Help on built-in function round in module builtins:
```

```
round(...)
```

```
    round(number[, ndigits]) -> number
```

```
    Round a number to a given precision in decimal digits (default 0 digits).
```

```
    This returns an int when called with one argument, otherwise the
```

```
    same type as the number. ndigits may be negative.
```

```
>>>
```

04 常用标准函数

random模块

函数	描述
<code>seed([a])</code>	初始化随机数生成器的种子，默认为系统时间。需要在生成随机数之前调用此函数。
<code>random()</code>	生成一个在[0.0,1.0)之间的随机浮点数
<code>randint(a,b)</code>	生成一个[a,b]之间的随机整数，其中 $a \leq b$ 。
<code>randrange</code> <code>([start,]stop [,step])</code>	生成一个[start,stop)，以step为步长的范围内的随机整数，step默认值为1，等价于 <code>choice(range(m,n,step))</code> 。
<code>uniform(a,b)</code>	生成一个[a,b]之间的随机浮点数，a可以大于b
<code>choice(seq)</code>	从非空序列seq中随机挑选一个元素
<code>sample(seq,k)</code>	从序列seq中随机获取k个元素，并返回一个新序列
<code>shuffle(seq)</code>	将序列seq的所有元素随机排序，并修改原序列
<code>getrandbits(k)</code>	生成一个k比特长的随机整数

random模块

函数的应用实例：

```
>>> from random import *
>>> random()
0.14836545714990013
>>> randint(1,10)
5
>>> x=[1,2,3,4,5]
>>> sample(x,2)
[5, 1]
>>> shuffle(x)
>>> x
[2, 5, 4, 3, 1]
>>>
```

```
#导入random模块的所有函数
#生成一个在[0.0,1.0)之间的随机浮点数
#生成一个[1, 10]之间的随机整数
#x赋值为一个序列
#从x序列中随机挑选2个元素生成新序列
#将x序列的元素随机排序
```

random模块

- random是如何生成随机数的呢？Python的随机数是用确定性的算法计算出来在[0.0,1.0) 范围均匀分布的随机数序列，它并不是真正的随机被称为伪随机数。伪随机数具有类似于随机数的统计特征，如均匀性、独立性等。因此在模拟研究中可以采用伪随机数代替真正的随机数，从而提高模拟效率。
- 在计算伪随机数时，若使用的初值（种子）不变，那么伪随机数的序列也不变。

```
>>> from random import *
>>> seed(2)                #随机种子为2
>>> random()
0.9560342718892494         #随机序列中第一个数
>>> random()
0.9478274870593494         #随机序列中第二个数
>>> random()
0.05655136772680869        #随机序列中第三个数
>>> seed(2)                #重新生成随机序列
>>> random()
0.9560342718892494         #新生成的随机序列的第一个数
>>>
```

随机数种子
seed(2)

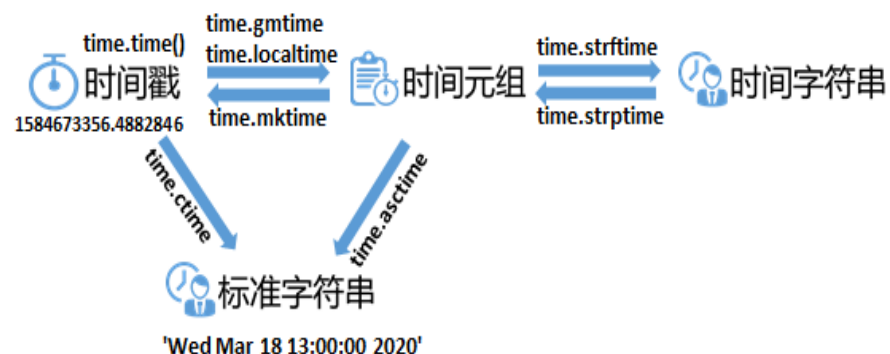
随机序列

0.9560342718892494
0.9478274870593494
0.05655136772680869
0.08487199515892163
0.8354988781294496
0.7359699890685233
0.6697304014402209
0.3081364575891442
0.6059441656784624
0.6068017336408379

- 在默认的情况下，当未设置种子时，random将以系统时间作为种子，这时产生的随机序列就是随时变化的了。

time模块

- 1. 时间表示与函数
- time模块提供与时间有关的函数。time模块有三种时间的表示，时间戳、时间元组和格式化时间字符串。



- 时间戳 (timestamp)：时间戳表示从格林尼治时间1970年1月1日00:00:00开始到现在所经过的秒数，其值为float类型。
- 时间元组 (struct_time)：元组共有9个元素，时间戳和格式化时间字符串之间的转化必须通过时间元组才能实现，它是三种时间表示的中间过程。
- 格式化时间 (format time)：以格式化的结构字符串使时间更具可读性。包括自定义的时间字符串和标准字符串。

time模块

函数	描述
time()	返回当前系统时间的时间戳（1970元年后的浮点秒数）
gmtime([secs])	将一个时间戳转换为UTC时区的时间元组，无参数时，默认为当前系统时间
localtime([secs])	将一个时间戳转换为当前时区的时间元组，无参数时，默认为当前系统时间
mktime(tupletime)	将一个时间元组转换为时间戳
strftime(fmt[,tupletime])	将一个时间元组，按指定格式fmt转换为时间字符串，即string format time。无参数时，默认为当前系统时间
strptime(str[,fmt])	将一个时间字符串解析为时间元组，即string parse time
asctime([tupletime])	将一个时间元组表示为24位的标准格式的时间字符串。无参数时，默认为当前系统时间
ctime([secs])	将一个时间戳表示为24位的标准格式的时间字符串。无参数时相当于asctime()

time模块

```
>>> import time
```

```
>>> time.time()
```

#返回当前时间的时间戳，是一个浮点数

```
1611557088.1828556
```

```
>>> time.localtime()
```

#返回当前本地时间的时间元组，元组有9个元素

```
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=14, tm_min=45, tm_sec=20,  
tm_wday=0, tm_yday=25, tm_isdst=0)
```

```
>>> time.mktime(time.localtime())
```

#将当前本地时间元组转换为时间戳

```
1611557166.0
```

```
>>> time.asctime()
```

#返回当前时间的标准字符串

```
'Mon Jan 25 14:46:49 2021'
```

```
>>> time.strftime('%Y年%m月%d日')
```

#将当前时间显示为自定义格式的时间字符串

```
'2021年01月25日'
```



time模块



- 2. 时间元组
- 时间元组由9个元素组成，每个元素有自己的名称和下标，下标从0开始。时间元组各元素的构成说明见表3.13。调用时间元组的元素可以使用元素的下标或者元素的名称。如：

```
>>> import time
>>> t1=time.localtime()           #返回本地时间的时间元组
>>> t1                             #t1为本地时间的时间元组
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=15, tm_min=7, tm_sec=51,
tm_wday=0, tm_yday=25, tm_isdst=0)
>>> print(t1[0],t1[1],t1.tm_mday)  #显示输出时间元组中的元素
2021 1 25
>>>
```



下标	元素名称	描述
0	tm_year	年份 (0000-9999) , 如2020
1	tm_mon	月份 (1-12)
2	tm_mday	一个月中的第几天 (1-31)
3	tm_hour	小时 (0-23)
4	tm_min	分钟 (0-59)
5	tm_sec	秒 (0-61) , 60、61是闰秒
6	tm_wday	一个星期中的第几天 (0-6) , 0是星期一
7	tm_yday	一年中的第几天 (1-366) , 366是闰年
8	tm_isdst	1-夏令时,0-非夏令时, -1-不确定, 默认值为-1



time模块

- 3. 时间格式符
- 格式化时间中的标准时间字符串长度固定为24位，显示的时间格式也是固定不变
- 如果想要自定义时间元组的显示格式，则需要使用带有格式符的时间字符串来完成。

```
>>> import time
>>> time.ctime()
'Mon Jan 25 15:35:02 2021'
>>> t2=time.localtime()           #获取当前时间的时间元组t2
>>> t2
time.struct_time(tm_year=2021, tm_mon=1, tm_mday=25, tm_hour=15, tm_min=35,
tm_sec=24, tm_wday=0, tm_yday=25, tm_isdst=0)
>>> time.strftime('%Y年%m月%d日 %A',t2)   #按时间格式符的样式显示时间t2
'2021年01月25日 Monday'
>>>
```

time模块

格式符	描述
%a	本地星期名称的简写 (如星期四为Thu)
%A	本地星期名称的全称 (如星期四为Thursday)
%b	本地月份名称的简写 (如八月份为agu)
%B	本地月份名称的全称 (如八月份为august)
%c	本地相应的日期和时间的字符串表示
%d	一个月中的第几天 (01-31)
%H	一天中的第几个小时 (24小时制, 00-23)
%I	第几个小时 (12小时制, 0-11)
%j	一年中的第几天 (001-366)
%m	月份 (01-12)
%M	分钟数 (00-59)
%p	本地am或者pm的相应符
%S	秒 (00-61)
%U	一年中的星期数。 (00-53星期天是一个星期的开始。) 第一个星期天之前的所有天数都放在第0周
%w	一个星期中的第几天 (0-6, 0是星期天)
%W	和%U基本相同, 不同的是%W以星期一为一个星期的开始
%x	本地相应日期字符串 (如15/08/01)
%X	本地相应时间字符串 (如08:08:10)
%y	去掉世纪的年份 (00-99) 两个数字表示的年份
%Y	完整的年份 (4个数字表示年份)
%z	与UTC时间的间隔 (如果是本地时间, 返回空字符串)
%Z	时区的名字 (如果是本地时间, 返回空字符串)
%%	"%" 字符

math模块

函数	返回值
<u>ceil(x)</u>	对x的向上取整，如ceil(4.1) 返回 5。
<u>exp(x)</u>	返回e的x次幂， 例如，exp(1)返回2.718281828459045。
<u>fabs(x)</u>	返回x的绝对值(浮点数)，如fabs(-10) 返回10.0。
<u>floor(x)</u>	对x的向下取整，如floor(4.9)返回4。
<u>fmod (x,y)</u>	返回x/y的余数（浮点数），如fmod(7,4)返回3.0。
<u>log(x[,base])</u>	返回x的自然对数，如log(e)返回1.0，可以用base参数改变对数的底，如，log(100,10)返回2.0。
<u>log10(x)</u>	返回10为基数的x的对数，如log10(100)返回 2.0。
<u>max(x1, x2,...)</u>	返回给定参数的最大值，参数可以为序列。
<u>min(x1, x2,...)</u>	返回给定参数的最小值，参数可以为序列。
<u>pow(x, y)</u>	返回x的y次幂，x**y 运算后的值。
<u>round(x [,n])</u>	返回浮点数x的四舍五入值，如给出n值，则代表舍入到小数点后的位数。
<u>sqrt(x)</u>	返回数字x的平方根，数字可以为负数，返回类型为实数，如sqrt(4)返回2.0。

math模块

函数的应用实例:

```
>>> import math
>>> print(math.fabs(-4),math.ceil(4.1),math.floor(4.9))
4.0 5 4
>>> print(math.pow(3,4),math.fmod(7,4),math.log(100,10))
81.0 3.0 2.0
```

calendar模块

函数	描述
setfirstweekday(weekday)	设置每周的第一天的日期码。默认为0（星期一）到6（星期日）。
firstweekday()	返回当前每周起始日期的设置。默认返回0，即星期一。
isleap(year)	如果year是闰年则返回True，否则为false。
leapdays(y1,y2)	返回在y1， y2两年之间的闰年总数。
month(year,month,w=2,l=1)	返回指定年月的日历。多行字符串格式，两行标题，一周一行。
monthcalendar(year,month)	返回一个整数列表。每个子列表代表一个星期。
monthrange(year,month)	返回两个整数。第一个是该月的是星期几，第二个是该月的日期码。
weekday(year,month,day)	返回指定日期的星期代码。

calendar模块

函数应用实例：

```
>>> from calendar import *      #导入 calendar 模块的所有函数
>>> print(month(2017,5))         #多行格式输出 2017 年 5 月的日历
```

```
    May 2017
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

```
>>> print(weekday(2017,5,20))    #返回 2017.5.20 星期代码，5 代表星期六
5
>>> print(monthrange(2017,5))    #2017 年 5 月第一天是星期一，有 31 天
(0, 31)
>>>
```

THANK YOU

The user can demonstrate on a projector or computer, or print the presentation and make it into a film to be used in a wider field