Python基础：

基本数据类型、表达式及输入输出

内容

- 标识符、变量及其赋值
- 基本数据类型
- 运算符及表达式
- 基本输入输出

Python标识符

标识符（Identifier)：在Python程序中自定义的变量、函数、类的符号/名称

- 允许采用大写字母、小写字母、数字、下划线(_)和汉字等字符，但标识符的首字符不能是数字，中间不能出现空格
- 大小写敏感　　myId≠myID
- 不能与保留字相同

汉字标识符：允许但不建议

Python保留字

保留字（Keyword)： 编程语言内部定义并保留使用的标识符

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

Python标识符

- 合法标识符：

  pythonPrint, Python_3_9, _python_ABC，python_你好

- 五个不同的标识符

  Python_3、python_3、PYTHON_3、PyThOn_3、pYtHoN_3

- 不合法的标识符

  Python.3、3_python、PYTHON 3、lambda

Python变量及其赋值

变量（Variable)：表示或指向特定值的标识符

变量赋值：　<变量> = <表达式>

- 不需要事先声明，变量的赋值就是其声明和定义的过程

```
x = 5
```

- 可以重新赋值直接改变变量的类型和值

```
x = 5
x = “abc”
```

Python变量：变量赋值

- 支持多重赋值

```
x = y = z = 5
```

- 赋值语句没有返回值

```
x = (y = y + 1)
```

- 支持多个变量同时赋值

```
x, y, z = 5, 'a', 1.5
```

Python中的常量

常量（Constant)：在程序执行过程中不能被改变的量

- Python中没有提供const修饰符，因此Python中没有常量，或者说，不能给常量命名
- Python程序中一般约定俗成以全大写的变量名表示这是一个常量，但这并非规则

内容

- 标识符、变量及其赋值
- 基本数据类型
- 运算符及表达式
- 基本输入输出

Python数据类型

数据类型	类型名称	示例
数字	int float complex	1234 3.14, 1.2e5 5+8j
布尔型	bool	True, False
字符串	str	‘hello world’ “C” ’’’这是一个字符串’’’
元组	tuple	（3，-5，8）
列表	list	[1, 2, 3] [‘a’, ‘b’, ‘c’]
集合	set	{-5，0，‘a’}
字典	dict	{1: “金牌”，2: “银牌“，3：“铜牌”}

基本数据类型

组合数据类型

基本数据类型

- 数字型
  - 整型
  - 浮点型
  - 复数类型
- 字符串
- 布尔型
- 空值
- 基本数据类型之间的转换

Python基本数据类型：整型

- 理论上可以任意大（取决于内存大小）

Python基本数据类型：整型

- 可以使用多种进制

进制	引导符	示例
十进制	无	0，-1，123
二进制	0b	0b10, 0b1101
八进制	0o	0o12，0o6573
十六进制	0x	0x4f, 0xed3c, 0xabcdef

Python基本数据类型：浮点型

- 必须有小数部分，小数部分可以为零
- 可以采用十进制或科学计数法表示
- 数值范围存在限制，小数精度也存在限制
- 浮点数间运算存在不确定尾数，不是bug

```
>>> x = 1.2
>>> y = 2.3
>>> x + y
3.5
>>> x + y == 3.5
True
```

```
>>> x = 4.2
>>> y = 2.1
>>> x + y
6.300000000000001
>>> x + y == 6.3
False
```

在浮点数的世界里不存在显而易见，不要轻易对两个浮点数进行直接比较，也不要进行精确的计算

Python基本数据类型：复数类型

- 与数学中的复数概念一致
- 由实部和虚部组成，用j或J表示虚部
- 实部和虚部的数值类型都是浮点型
- 可以使用Python内置函数real、imag求得复数的实部和虚部

```
>>> x = 5+6j
>>> y = 1-3J
>>> z = x + y
>>> z
(6+3j)
```

```
>>> z
(6+3j)
>>> z.real
6.0
>>> z.imag
3.0
```

Python数字类型的特殊写法

- 在数字中间的位置使用单个下划线作为分隔提高数字的可读性

```
x = 1_000_000
```

#类似于x = 1,000,000

- 下划线可以用于数字中间的任何位置，但不能用于数字头尾

```
>>> 1_000_000
1000000
>>> 1_2_34
1234
>>> 12_3+4_56j
(123+456j)
>>> 1_2.3_4_56
12.3456
```

```
>>> _123
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    _123
NameError: name '_123' is not defined

>>> 123_
SyntaxError: invalid decimal literal
```

基本数据类型

- 数字型
- 字符串
- 布尔型
- 空值
- 基本数据类型之间的转换

Python基本数据类型：字符串

- 用单引号、双引号或三引号括起来的任意文本

‘abc’　‘中国’　“Python”　“‘I said, “Let’s go!”’”

‘’　“”　“‘’”　“‘“”’”

- 一对三单引号或三双引号表示的字符串支持换行、复杂排版格式；也可以在程序中表示较长的注释

```
>>> s = """When I was young
...     I'd listen to the radio
...     Waiting for my favorite songs"""
```

```
>>> print(s)
When I was young
    I'd listen to the radio
    Waiting for my favorite songs
```

Python支持转义字符

转义字符	含义	转义字符	含义
\n	换行符	\\	一个\
\t	横向制表符	\ooo	3位八进制数对应的字符，数字可以是1~3个
\v	纵向制表符	\xhh	2位十六进制数对应的字符
\r	回车	\uhhhh	4位十六进制数对应的字符
\’	单引号	\Uxxxxxxxx	8位十六进制数对应的字符，要求不大于\U0010FFFF
\”	双引号		

转义字符的使用

```
>>> s = "You raise me up \nso I can stand on mountains"
>>> print(s)
You raise me up
so I can stand on mountains
```

转义字符的原始表达（不转义）

字符串界定符前加r或R代表不进行转义

```
>>> s = r'换行符是\n。'
>>> print(s)
换行符是\n。
```

Python字符串：字符串序列

Python字符串本质上是由长度为1的字符串（字符）组成的字符串序列，支持双向索引

0	1	2	3	4	5	6	7	8	9	10
H	**e**	**l**	**l**	**o**		**W**	**o**	**r**	**l**	**d**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s = 'Hello World'
>>> print(s[0])
H
```

```
>>> print(s[8], s[5], s[10])
r   d
```

```
>>> print(s[-11], s[-1], s[-4])
H d o
```

Python字符串：通过索引进行“切片”

s[start:stop]

0	1	2	3	4	5	6	7	8	9	10
H	**e**	**l**	**l**	**o**		**W**	**o**	**r**	**l**	**d**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> print(s[1:7])
ello W
>>>
>>> print(s[-5:-3])
Wo
```

```
>>> print(s[0:])
Hello World
>>>
>>> print(s[:8])
Hello Wo
```

```
>>> print(s[-1:])
d
>>>
>>> print(s[:-1])
Hello Worl
```

Python字符串

不能通过索引对字符串中的字符进行修改

0	1	2	3	4	5	6	7	8	9	10
H	**e**	**l**	**l**	**o**		**W**	**o**	**r**	**l**	**d**
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
>>> s = 'Hello World'
>>> s[2] = 'a'
Traceback (most recent call last):
  File "<pyshell#87>", line 1, in <module>
    s[2] = 'a'
TypeError: 'str' object does not support item assignment
```

Python字符串：修改字符串中字符的值

```
>>> s = 'Hello World'
>>> s1 = s.replace('o', 'O')
>>> print(s1)
HellO WOrld
>>>
>>> print(s)
Hello World
```

通过s.replace(old,new)可以将字符串s中所有old字符替换new

替换并不改变原字符串的值

```
>>> s2 = s.replace('l', 'L', 1)
>>> print(s2)
HeLlo World
```

如果希望控制被替换字符的个数，可以在replace函数中增加一个参数n，表示只对前n个字符执行替换

基本数据类型

- 数字型
- 字符串
- 布尔型
- 空值
- 基本数据类型之间的转换

Python基本数据类型：布尔型

- 表示与计算都与布尔代数完全一致
- 只有两个值：True，False

```
>>> 3.1 > 3
True
```

```
>>> True and False
False
```

```
>>> True == 1
True
>>> False == 0
True
```

```
>>> s1 = 'Hello World'
>>> s2 = 'hello World'
>>> s1 == s2
False
```

```
>>> True or False
True
```

基本数据类型

- 数字型
- 字符串
- 布尔型
- 空值
- 基本数据类型之间的转换

Python基本数据类型：空值

- 只有一个值：None

```
>>> a = None
>>> print(a)
None
```

```
>>> a = None
>>> type(a)
<class 'NoneType'>
```

```
>>> a
>>>
```

- 不支持任何运算，没有任何内置方法，也没有什么有用的属性
- 其布尔值总是False

```
>>> a = None
>>> bool(a)
False
```

```
>>> bool(None)
False
```

基本数据类型

- 数字型
- 字符串
- 布尔型
- 空值
- 基本数据类型之间的转换

Python基本数据类型的转换：数值型之间

- 对于数值型数据，在进行混合运算时会发生“隐式”类型转换，转换为“最大”的类型

```
>>> a = 2
>>> b = 3.5
>>> c = a + b
>>> print(c)
5.5
```

```
>>> a = 2
>>> type(a)
<class 'int'>
>>>
>>> a = a + 10j
>>> type(a)
<class 'complex'>
```

数值与字符串的数据类型转换

- bin()、oct()、hex()分别把其它整数类型转换为二进制、八进制和十六进制形式

```
>>> x = 1024
>>> y = bin(x)
>>> print(y)
0b10000000000
```

```
>>> y, z = oct(x), hex(x)
>>> print(y, z)
0o2000 0x400
```

```
>>> type(x), type(y)
...
(<class 'int'>, <class 'str'>)
```

- int()、float()分别把其它类型转换为整型和浮点型

```
>>> s = '123'
>>> x ,y = int(s), float(s)
>>> print(x, y)
123 123.0
```

```
>>> s = '123.456'
>>> x, y = float(s), int(x)
>>> print(x, y)
123.456 123
```

数值与字符串的数据类型转换

- str() 把其它类型转换为字符串

```
>>> x, y, z = 5, -1.6, 3+8.3j
>>> print(x, y, z)
5 -1.6 (3+8.3j)
>>> print(str(x), str(y), str(z))
5 -1.6 (3+8.3j)
```

```
>>> str(x)
'5'
>>> str(y)
'-1.6'
>>> str(z)
'(3+8.3j)'
```

- eval(x)将字符串x看作一个Python表达式，并求其值

```
>>> eval('3 + 4')
7
```

```
>>> x, y = 3, 4
>>> eval('x + y')
7
```