字符串

字符串

字符串可以用双引号"，也可以用单引号’

In [89]:
```
var1 = "I love Python"
var2 = '我爱中华！'
中文变量名 = 'Python3直接支持中文等符号，包括标识符名，这个的确牛！'

print (var1, type(var1))
print (var2, type(var2))
print (中文变量名, type(中文变量名))
```

```
I love Python <class 'str'>
我爱中华！ <class 'str'>
Python3直接支持中文等符号，包括标识符名，这个的确牛！ <class 'str'>
```

字符串

能够在通过某一种标示的字符串中使用另外一种标示符

```
In [90]: print ("He said 'hello'")
         He said 'hello'
```

字符串

转义字符是反斜杠"\"

```
In [92]: # \n 是换行符

         print("This is first line\nThis is second line")
```

```
This is first line
This is second line
```

转义字符是反斜杠"\"，如果不想让反斜杠发生转义，可以在字符串前面添加一个 r，表示原始字符串

```
In [93]: print(r"This is first line \nThis is second line")
```

```
This is first line \nThis is second line
```

字符串

多行字符串可以通过三个连续的单引号(’’’)或是双引号(“””)来进行标示

In [94]:
```
str1 = '''first line
          second line
          third line'''
print (str1)
```

```
first line
          second line
          third line
```

In [95]:
```
str1 = '''first line
second line
third line'''
print (str1)
```

```
first line
second line
third line
```

字符串

使用 + 进行字符串链接:

```
In [96]: "Hello" + " World"
Out[96]: 'Hello World'
```

使用 * 进行字符串链接:

```
In [97]: "Bye" * 2
Out[97]: 'ByeBye'
```

字符串

Python中没有单独的单个 字符类型，要注意 字符串 和 数字 之间的区别:

```
In [98]: 4 + 5
Out[98]: 9

In [99]: "4" + "5"
Out[99]: '45'

In [100]: 4 + "5"
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-100-871c0c3bbca2> in <module>()
----> 1 4 + "5"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

List 列表

List 列表

List（列表） 是 Python 中使用最频繁的数据类型

列表是写在方括号 [ ] 之间、元素之间用逗号分隔开

```
In [101]: list1 = [1,2,3,4,5,6]

          print(list1)
```

```
[1, 2, 3, 4, 5, 6]
```

List 列表

列表中元素的类型可以不相同，它支持数字，字符串甚至可以包含列表（所谓嵌套）

In [102]:
```
list2 = [1,2,3,4,5,6,"hello python",[8,9,10,11,12]]

print(list2)
```

```
[1, 2, 3, 4, 5, 6, 'hello python', [8, 9, 10, 11, 12]]
```

列表元素访问

可以通过索引（下标）来访问列表元素

单个列表元素访问的语法格式为：列表名[下标]

```
In [103]: list1 = [1,2,3,4,5,6]

          list1[0]

Out[103]: 1

In [104]: list1[2]

Out[104]: 3
```

列表下标从0开始，-1表示倒数第1个

```
In [105]: list1[-1]

Out[105]: 6

In [106]: list1[-3]

Out[106]: 4
```

列表元素访问

下标访问不要越界

```
In [103]: list1 = [1,2,3,4,5,6]

          list1[0]

Out[103]: 1
```

```
In [107]: list1[6]

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-107-089331a3fd74> in <module>()
----> 1 list1[6]

IndexError: list index out of range
```

列表元素访问

列表截取（切片）的语法格式为：列表名[头下标:尾下标]

```
In [108]: list1 = [1,2,3,4,5,6]

          list1[0:3]
          # 列表截取（切片）返回一个包含所需内容的新列表

Out[108]: [1, 2, 3]
```

结果不包含尾下标那个元素！

```
In [109]: list1[-3:-1]

Out[109]: [4, 5]
```

切片步长

```
In [110]: list1[::2]

Out[110]: [1, 3, 5]
```

访问嵌套列表

访问嵌套列表元素：层层深入

In [111]:
```
list2 = [1,2,3,4,5,6,"hello python",[8,9,10,11,12]]

print(list2[-1][1:])
```

```
[9, 10, 11, 12]
```

字符串元素访问

字符串是一种特殊列表，可以按列表元素的访问方法来访问字符串中的元素

```
In [112]: str1 = "hello, hangzhou!"

          print(str1[2:5])
```

```
llo
```

```
In [113]: print(str1[-1])
```

```
!
```

Tuple 元组

Tuple 元组

元组（tuple）与列表类似，不同之处在于元组的元素不能修改

元组写在小括号 ( ) 里，元素之间用逗号隔开

元组中元素的类型可以不相同，和列表类似，也支持嵌套

In [114]:
```
tuple1 = (1,2,3,4,5,6,"hello python",[8,9,10,11,12],(13,14))

print(tuple1)
```

```
(1, 2, 3, 4, 5, 6, 'hello python', [8, 9, 10, 11, 12], (13, 14))
```

Tuple 元组

元组的元素访问和截取方式和列表相同，通过下标来操作

In [115]:
```
tuple1 = (1,2,3,4,5,6,"hello python",[8,9,10,11,12],(13,14))
print(tuple1[0])
print(tuple1[-1])
print(tuple1[6:-1])
print(tuple1[-1][-1])
```

```
1
(13, 14)
('hello python', [8, 9, 10, 11, 12])
14
```

Tuple 元组

元组一但定义好不能修改，是只读的

```
In [116]: tuple1[1] = 7
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-116-de8498ac87ac> in <module>()
----> 1 tuple1[1] = 7

TypeError: 'tuple' object does not support item assignment
```

Set 集合

Set 集合

集合（set）是一个无序、且不含重复元素的序列

集合主要用来进行成员关系测试和删除重复元素

可以使用大括号 { } 或者 set() 函数 创建集合

```
In [117]: # 自动去除重复元素
          set1 = {1,3,5,5,3,1}

          print(set1)
```

```
{1, 3, 5}
```

```
In [118]: 5 in set1
```

```
Out[118]: True
```

```
In [119]: 8 in set1
```

```
Out[119]: False
```

集合操作

集合的操作：交、并、差、补

```
In [120]: set1 = {1,2,3}
          set2 = {2,4,5}

In [121]: #集合的并
          set1 | set2
Out[121]: {1, 2, 3, 4, 5}

In [122]: #集合的交
          set1 & set2
Out[122]: {2}

In [123]: #集合的差
          set1 - set2
Out[123]: {1, 3}

In [124]: #集合的补，两个集合中不同时存在的元素集合
          set1 ^ set2
Out[124]: {1, 3, 4, 5}

In [125]: (set1|set2)-(set1&set2)
Out[125]: {1, 3, 4, 5}
```

Dictionary字典

Dictionary字典

字典是一种映射类型，用"{ }"标识，它是一个无序的 键(key) : 值(value)对 集合

键(key)必须使用不可变类型，在同一个字典中，键(key)是唯一的

字典当中的元素是通过键来存取的

```
In [126]: dict1 = {"name":"giggle","height":176,"weight":72}

In [127]: dict1["height"]
Out[127]: 176
```

字典的操作

In [128]:
```
# 修改字典某项的值

dict1["weight"]=73
print(dict1)
```

```
{'weight': 73, 'name': 'giggle', 'height': 176}
```

In [129]:
```
# 在字典中增加一项

dict1["sex"]="M"
print(dict1)
```

```
{'weight': 73, 'name': 'giggle', 'height': 176, 'sex': 'M'}
```

构建字典

构建空字典

In [130]:
```
# 构建空字典

dict2 ={}
print(dict2)
```

```
{}
```

通过元组序列构造字典

In [131]:
```
# 通过元组序列构造字典

dict2 = dict([('name','giggle'),('height',176)])
print(dict2)
```

```
{'name': 'giggle', 'height': 176}
```

字典内建方法

字典类型也有一些内置的函数，例如clear()、keys()、values()

```
In [132]: dict2.keys()
Out[132]: dict_keys(['name', 'height'])

In [133]: dict2.values()
Out[133]: dict_values(['giggle', 176])

In [134]: dict2.clear()
          print(dict2)
          {}
```

print的格式化输出

print的字符串格式化符号

- %c 格式化字符及其ASCII码
- %s 格式化字符串
- %d 格式化整数
- %u 格式化无符号整型
- %o 格式化无符号八进制数
- %x 格式化无符号十六进制数
- %X 格式化无符号十六进制数（大写）
- %f 格式化浮点数字，可指定小数点后的精度
- %e 用科学计数法格式化浮点数
- %E 作用同%e，用科学计数法格式化浮点数
- %g %f 和 %e 的简写
- %G %f 和 %E 的简写

格式化操作符辅助指令

- m.n. m 是显示的最小总宽度（如果指定的话）,n 是小数点后的位数(如果指定的话)
- *定义宽度或者小数点精度
- -用做左对齐
- +在正数前面显示加号 +
- 在正数前面显示空格#
  - 在八进制数前面显示零('0')
  - 在十六进制前面显示'0x'或者'0X'(取决于用的是'x'还是'X')
- '%%'输出一个单一的'%'
- (var)映射变量(字典参数)

print的格式化输出

如果想通过变量来填充格式控制字符串，那么可以使用运算符(%)和一个元组，在目标字符串中从左至右使用%来指代变量的位置

```
In [135]: print ("I like %s and can eat %.2f kg." % ("orange", 1.5))
          I like orange and can eat 1.50 kg.
```

print的格式化输出

```
In [136]: pi = 3.1415926
          print("pi = %.*f" % (5,pi)) # 用*从后面的元组中读取字段宽度或精度

          pi = 3.14159
```

使用字典来对应填充

```
In [137]: print ("I like %(fruit_name)s and can eat %(weight)8.2f kg." % {"fruit_name": "orange", "weight": 1.5})

          I like orange and can eat     1.50 kg.
```

类型转换

类型转换

数据类型的转换，只需要将数据类型作为函数名即可使用

这些函数返回一个新的对象，表示转换的值，如： int(), float(), 和 str() 等

```
In [138]: x = "6"
          print(x, type(x))
          x = int("6")
          print(x, type(x))

          6 <class 'str'>
          6 <class 'int'>
```

```
In [139]: float("1.25")

Out[139]: 1.25

In [140]: str(4)

Out[140]: '4'
```

类型转换

字符和数字的转换

```
In [141]: int("a")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-141-91097a4105a2> in <module>()
----> 1 int("a")

ValueError: invalid literal for int() with base 10: 'a'
```

字符和数字的转换，通过 **ord()** 和 **chr()**

```
In [142]: ord("a")
Out[142]: 97
```

```
In [143]: chr(65)
Out[143]: 'A'
```

表达式计算

超强的表达式计算（表达式字符串到数值的转换）

```
In [144]: x = 8
          calc = "5*x+9"
          eval(calc)

Out[144]: 49
```