PCS101　内建数据类型

概述

Python 提供的基本数据类型是比较丰富的，主要有：布尔类型、整型、浮点型、字符串、列表、元组、集合、字典，等等。下面就依次介绍它们的使用方法。

应用

空(None)

None 表示该值是一个空对象，且对其没有特别的操作，比如没有明确定义返回值的函式就返回 None。

布尔类型(Boolean)

在 Python 中，None、任何数值类型中的 0、空字符串''、空元组()、空列表[]、空字典{}都被当作 False，还有自定义的类型，如果它实现了__nonzero__()或__len__()方法且方法返回 0 或 False，则其实例也被当作 False，其他对象均为 True。

```
>>> bigornot = 1>2
>>> bigornot
False
>>> if []:
...     print "True"
... else:
...     print "False"
```

```
...
False
>>> if '':
...     print "True"
... else:
...     print "False"
...
False
```

整型(Int)

在 Python 内部对整数的处理分为普通整数和长整数，普通整数长度为机器位长，通常都是 32 位。超过这个范围的整数就自动当作长整数处理，而长整数表示范围就几乎完全没有限制了。

```
>>> a = []
>>> a is False
False
>>> a = 1
>>> type(a)
<type 'int'>
>>> b = 99999999999999999999999
>>> type(b)
<type 'long'>
>>> a
1
>>> b
99999999999999999999999L
```

浮点型(Float)

Python 的浮点数就是数学中的小数，类似 C 语言中的 double。在数值运算中，整数与浮点数运算的结果是浮点数，这就是所谓的“提升规则”，也就是“小”类型会被提升为“大”类型参与计算，许多语言都是这样的。在 Python 的所有运算中，对应类型所表示范围小的一般会被扩宽到范围大的类型，其依次为：int、long、 float、complex。其中，complex 包括像 list 这样的集合体。

```
>>> a = 0.000000001
>>> type(a)
<type 'float'>
>>> a
1.0000000000000001e-09
```

字符串(String)

Python 字符串既可以用单引号括起来也可以用双引号括起来，甚至还可以用三引号括起来。这样如果字符串里本身包含双引号，就可以用单引号来括，也可以用双引号加转义字符来括：

```
>>> 'My name is "python" '
'My name is "python" '
>>> 'My name is \"python\" '
'My name is "python" '
>>> "My name is \"python\" "
'My name is "python" '
```

同样，如果字符串里本身就包含单引号呢？可以使用双引号，也可以结合使用单引号和转义字符来实现：

```
>>> "My name is 'python' "
"My name is 'python' "
>>> 'My name is \'python\' '
"My name is 'python' "
```

三个引号的字符串就更方便了，中间甚至还可以换行！

```
>>> '''My
... name
... is
... "python"
... !
... '''
'My\nname\nis\n"python"\n!\n'
```

列表(List)

用符号[]表示列表，中间的元素可以是任何类型，用逗号分隔。list 类似于 C 语言中的数组，用于顺序存储结构。

```
>>> test = [1, 2, "yes"]
```

内建函式

1. append(x) 追加到链尾。
2. extend(L) 追加一个列表，等价于+=。
3. insert(i,x) 在位置 i 插入 x，其余元素向后推。如果 i 大于列表的长度，就在最后添加。如果 i 小于 0，就在最开始处添加。

4. remove(x) 删除第一个值为 x 的元素，如果不存在会抛出异常。
5. reverse() 反转序列。
6. pop(i) 返回并删除位置为 i 的元素，i 默认为最后一个元素(i 两边的[]表示 i 为可选的，实际不用输入)。
7. index(x) 返回 x 在列表中第一次出现的位置，不存在则抛出异常。
8. count(x) 返回 x 出现的次数。
9. sort() 排序。
10. len(List) 返回 List 的长度。
11. del list[i] 删除列表 list 中指定的第 i 个变量。

```
>>>test = [1,2,"yes"]
>>>test.append(1) # 追加到链尾
>>>test
[1, 2, 'yes', 1]
>>>test.extend([ 'no','maybe']) # 追加一个列表
>>>test
[1, 2, 'yes', 1, 'no', 'maybe']
>>> test.insert(0,'never') # 在位置 0 插入'never'
>>> test
['never', 1, 2, 'yes', 1, 'no', 'maybe']
>>> test.remove('no') # 删除第一个值为"no"的元素,如果不存在会抛出异常
>>> test
['never', 1, 2, 'yes', 1, 'maybe']
>>> test.reverse() # 反转序列
>>> test
['maybe', 1, 'yes', 2, 1, 'never']
>>> test.pop() # 返回并删除位置为 i 的元素,i 默认为最后一个元素
'never'
>>> test
['maybe', 1, 'yes', 2, 1]
>>> test.index('yes') # 返回第一个值为'yes'的元素,不存在则抛出异常
2
>>> test.count(1) # 返回 1 出现的次数
2
>>>test.sort() # 排序
>>> test
[1, 1, 2, 'maybe', 'yes']
>>> len(test)
5
```

```
>>> del test[2] # 删除
>>> test
[1, 1, 'maybe', 'yes']
>>>
```

切片

切片指的是抽取序列的一部分，其形式为：list[start:end:step]。其抽取规则是：一般默认的步长为 1，但也可自定义。在默认步长的情况下，抽取的部分应该是 list 中从 start 一直到(end-1)；step=1 时，抽取的部分应该是 list 中从 start 开始，每次加上 step，直到 end 为止。当 start 没有给出时，默认从 list 的第一个元素开始；当 end=-1 时表示 list 的最后一个元素，依此类推。

```
>>> test = ['never', 1, 2, 'yes', 1, 'no', 'maybe']
>>> test[0:3] # 包括 test[0],不包括 test[3]
['never', 1, 2]
>>> test[0:6:2] # 包括 test[0],不包括 test[6],而且步长为 2
['never', 2, 1]
>>> test[:-1] # 包括开始,不包括最后一个
['never', 1, 2, 'yes', 1, 'no']
>>> test[-3:] # 抽取最后 3 个
[1, 'no', 'maybe']
>>>test[::-1] # 倒序排列
['maybe', 'no', 1, 'yes', 2, 1, 'never']
```

列表推导式

直接通过 for 循环生成一个 list，形式如下：[<expr1> for k in L if <expr2>]，表示在列表 L 中，如果 expr2 为真，就循环执行 expr1 语句并产生一个列表，此为列表推导式。

```
>>>freshfruit = ['  banana  ','   loganberry  ']
>>>[weapon.strip() for weapon in freshfruit]  # strip()是去除字符串两端多于
                                  # 空格，该句是去除序列中的所有字串两端多余的空格
['banana', 'loganberry']
>>>vec = [2,4,6]
>>>[3*x for x in vec if x>3]  # 大于 3 的元素乘上 3 作为新列表元素
[12, 18]
>>>[(x,x**2) for x in vec] # 循环变量要是一个 sequence,而[x,x**2 for x in vec]
                           # 是错误的
[(2,4),(4,16),(6,36)]
```

```
>>>vec2 = [4,3,-9]
>>>[x*y for x in vec for y in vec2] # vec 与 vec2 元素相乘
[8, 6, -18, 16, 12, -36, 24, 18, -54]
>>>[vec[i]+vec2[i] for i in range(len(vec))] # vec 与 vec2 元素相加
[6, 7, -3]
>>>[str(round(355/113.0,i)) for i in range(1,6)]  # str()是转换类型为可以
                        # 打印的字符，round(x,n)表示对 x 保留 n 位小数(四舍五入)
['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

元组(tuple)

元组是和列表相似的数据结构，但是它一旦初始化就不能更改，速度比 list 快，同时 tuple 不提供动态内存管理的功能。而且，要理解以下规则：

- tuple 可以用下标返回一个元素或子 tuple；
- 表示只含有一个元素的 tuple 的方法是：(d,)后面有个逗号，用来和单独的变量相区分。

```
>>>t = 1234,5567,'hello' # t=(1234,5567,'hello')的简写
>>>x,y,z = t # 拆分操作可以应用于所有 sequence
>>>x
1234
>>>u = t,(1,2,3)
>>>u
((1234,5567,'hello'),(1,2,3))
>>>empty = () # 空元组
>>>singleton = 'hi', # 单个元素的元组,注意逗号
>>> singleton
('hi',)
>>> print t[2]
hello
>>> print t[:2]
(1234,5567)
```

通过元组可以很简单地进行数据交换。比如：

```
a = 1
b = 2
a,b = b,a
```

集合(Set)

集合是无序的，不重复的元素集，类似数学中的集合，可进行逻辑运算和算术运算。

```
>>> s = set(['a', 'b', 'c'])  # 集合对象 s
>>> len(s)  # 集合 s 的长度
3
```

```
>>> 'a' in s  # 元素'a'在集合 s 中, 返回布尔类型
True
>>> 'd' not in s  # 元素'd'不在集合 s 中, 返回布尔类型
True
>>> t = set(['a', 'b', 'c', 'd'])  # 集合对象 t
>>> s.issubset(t)  # s 是否是 t 的子集, 等价于 s <= t
True
>>> s.issuperset(t)  # s 是否是 t 的超集, 等价于 s >= t
False
>>> s.union(t)  # 集合的并, 等价于 s | t
set(['a', 'c', 'b', 'd'])
>>> s | t
set(['a', 'c', 'b', 'd'])
>>> s.intersection(t)  # 集合的交, 等价于 s & t
set(['a', 'c', 'b'])
>>> s & t
set(['a', 'c', 'b'])
>>> s.difference(t)  # 集合的差, 等价于 s - t
set([])
>>> s - t
set([])
>>> t.difference(s)
set(['d'])
>>> t - s
set(['d'])
>>> s.symmetric_difference(t)  # 集合的异或, 等价于 s ^ t
set(['d'])
>>> s ^ t
set(['d'])
```

字典(dict)

字典是一种无序存储结构，包括关键字（key）和关键字对应的值（value）。 字典的格式为：dictionary = {key:value}。关键字为不可变类型，如字符串、整数、只包含不可变对象的元组。列表等不能作为关键字。如果列表中存在关键字对，可以用 dict()直接构造字典，而这样的列表对通常是由列表推导式生成的。

```
>>> tel = {'jack':4098, 'sape':4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'jack': 4098, 'guido': 4127}
>>> tel['jack'] # 如果 jack 不存在,会抛出 KeyError
4098
```

```
>>> tel.get("zsp", 5000) # 如果"zsp"为 tel 的键则返回其值,否则返回 5000
5000
>>> del tel['sape'] # 删除键'sape'和其对应的值
>>> tel.keys()
['jack', 'guido']
>>> "jack" in tel # 判断"jack"是否 tel 的键
True
>>> "zsp" not in tel
True
>>> for k,v in tel.iteritems():  # 同理 tel.iterkeys()为键的迭代
器,tel.itervalues()为值的迭代器
...     print k,v
...
jack 4098
guido 4127
>>> tel.copy() # 复制一份 tel
{'jack': 4098, 'guido': 4127}
>>> tel.fromkeys([1, 2], 0)
{1: 0, 2: 0}
>>> tel.popitem() # 弹出一项
('jack', 4098)
```

小结

Python 中的数据类型比较丰富，主要有上述这些。还有很多关于数据类型的知识在这里没有描述，可以进一步参考以下网址：

- 内置数据类型：*http://wiki.woodpecker.org.cn/moin/ObpLovelyPython/LpyQLearn-2-data*
  精巧地址：*http://bit.ly/1FsXZf*
- Python 绝对简明手册：*http://wiki.woodpecker.org.cn/moin/PyAbsolutelyZipManual*
  精巧地址：*http://bit.ly/2EEz6I*