
# 内置数据结构1

北京理工大学计算机学院　高玉金

2019年3月

# Python中的内置数据结构

- 序列类型结构：元组tuple，序列list，字符串str（特殊只读）
- 集合类型结构：集合set
- 映射类型结构：字典dict
- 这些结构的元素一般无类型限制，而且大多可以多层嵌套，
- Python内置数据结构中没有类似C语言中的数组（其中元素为同种数据类型）
- 这些内置数据结构让Python的学习和使用变得简单，强大，但也有不足之处，比如性能较低，内存开销较大等

# 序列类型结构之列表list

- 列表list是一个数据容器，数据项之间存在先后关系，通过序号访问，数据项可重复，数据类型可以多样
- 列表list内的数据可修改
- 字符串可以看做是特殊（内容类型相同，只读）的列表list

# 生成列表list

- 通过中括号（[]）赋值产生
- 通过list()函数可以将一个可迭代对象强制转换为列表
- 列表推导式生成，如［ c for c in s if c.isdigit()]
  - 推导式又称解析式，是Python的一种独有特性
  - 推导式可以从一个数据序列构建另一个新的数据序列的结构体

```
>>> list()
[]
>>> list("china")
['c', 'h', 'i', 'n', 'a']
>>> la = [2,3]
>>> la
[2, 3]
```

```
>>> s="123abc456def"
>>> s2=[c for c in s if c.isdigit()]
>>> s2
['1', '2', '3', '4', '5', '6']
```

# 列表的使用

- in和not in，用于判断某元素是否在列表内
- 列表索引，如ls[i]=x，根据位置访问（读写）列表内的元素
- 切片，如ls[m:n:k]
- ls.append(x)，可在列表ls最后增加一个元素x
- ls.remove(x)，将列表中出现的第一个元素x删除
- **遍历列表元素：**

```
for <元素> in <列表>:
    语句
```

# 列表元素的排序

```
In [114]: ls=[4,2,6,3]

In [115]: ls.sort()

In [116]: ls
Out[116]: [2, 3, 4, 6]
```

- list.sort(key, reverse)排序，改变原列表
- newls=sorted(ls,key,reverse)排序，生成新列表
- sorted()方法可以用在任何数据类型的序列中，返回的总是一个列表形式
- key用复杂对象的某些值来对复杂对象的序列排序
- key参数来指定函数，此函数将在每个元素比较前被调用，此函数只有一个参数且返回一个值用来进行比较
- Reverse参数 True或Flase

# 字符串和列表的可变性（只读）比较

```
>>> a
'hello'
>>> ls=list(a)
>>> ls
['h', 'e', 'l', 'l', 'o']
>>> ls[0]='p'
>>> ls
['p', 'e', 'l', 'l', 'o']
>>> a
'hello'
>>> a[0]='p'
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    a[0]='p'
TypeError: 'str' object does not support item assignment
```

# 字符串与列表的内存分配比较

- 字符串为不变类型，相同值指向同一内存空间
- 列表为可变类型，即使值相同，也需要额外分配空间

```
In [85]: s1="abc"

In [86]: s2="abc"

In [87]: id(s1)
Out[87]: 2401854745040

In [88]: id(s2)
Out[88]: 2401854745040
```

```
In [89]: ls1=list(s1)

In [90]: id(ls1)
Out[90]: 2401945988680

In [91]: ls2=list(s1)

In [92]: id(ls2)
Out[92]: 2401946016904

In [93]: ls1 == ls2
Out[93]: True

In [94]: ls1 is ls2
Out[94]: False
```

# 元组的生成和限制

- 元组采用逗号和圆括号（可选）来表示
- 通过赋值或tuple()函数强制转换
- 元组生成后其元素**不可修改和删除，即**不可remove()，不可append()
- 其他功能和列表相似，如可以索引和切片
- 用dir()看一下tuple对象的方法，只有count和index两个

```
>>> d=(2)
>>> d
2
>>> type(d)
<class 'int'>
>>> d=(2,)
>>> type(d)
<class 'tuple'>
>>> d
(2,)
>>>
```

```
>>> d=2,3
>>> type(d)
<class 'tuple'>
>>> d,c = 2,3
>>> type(d)
<class 'int'>
>>>
```

```
In [117]: t=(2,3,4)

In [118]: t[0]
Out[118]: 2

In [119]: t[:-1]
Out[119]: (2, 3)

In [120]: t[0]=9
Traceback (most rec
```

# 元组的使用

- 主要用于表达固定数据项、函数多返回值return、多变量同步赋值、循环遍历等

```
def func(x):   #函数多返回值
    return x, x**3

a, b = 'dog', 'tiger'   #多变量同步赋值
a, b = (b, a)           #交换
for x, y in ((1,0), (2,5), (3,8)):
```