# Python语言基础与应用

## 数据类型 / 容器类型：列表和元组

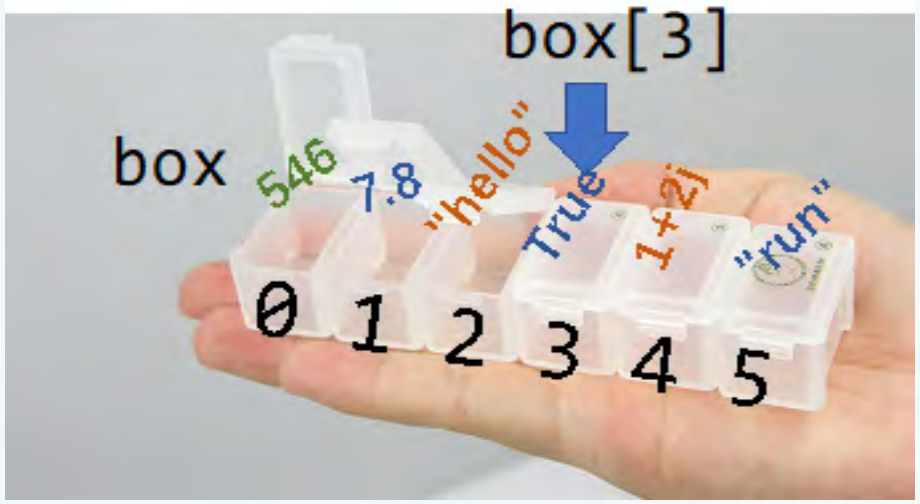
陈斌 北京大学 gischen@pku.edu.cn

# 容器类型：列表和元组

- 数据收纳盒
- 列表和元组的创建
- 列表和元组的操作

# 数据收纳盒

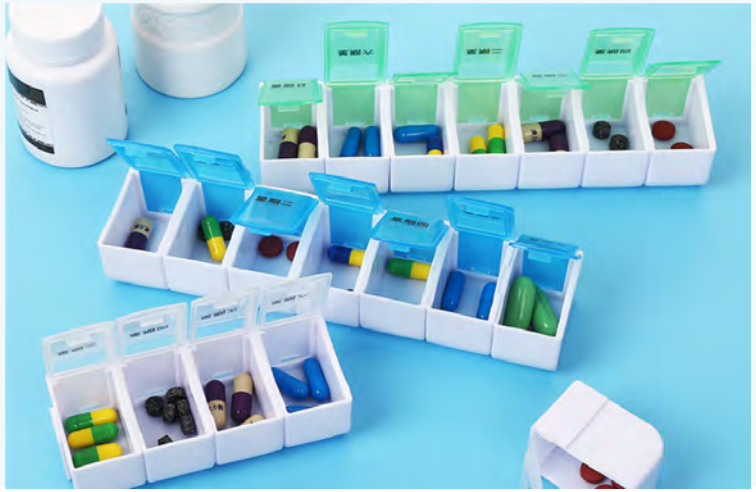
- 用来收纳数据对象的数据类型
- 以一种规则的下标索引方式（收纳盒名字+数字序号）访问到每个数据
- 这种收纳盒是一种序列



# 数据收纳盒

- 列表可以**删除、添加、替换、重排**序列中的元素（**可变类型**）
- 元组是不能再更新（**不可变**）序列

元组在保留列表大多数功能的同时，去掉了一些灵活性以换取更高的处理性能



# 列表和元组的创建

- **创建列表**

  方括号法[]，指明类型法list()

- **创建元组**

  圆括号法()，指明类型法tuple()

- **列表或元组中保存的各个数据称作元素(element)，类型没有限制**

# 列表的操作：增长/缩减

- **增长列表**

  append操作/insert操作/extend操作

- **缩减列表**

  pop操作/remove操作/clear操作

- **列表是一种可变容器，可以随意增减**

- **但并不是所有的数据容器都能像列表这样可以继续添加新元素**

# 列表的操作：重新组织

## reverse / sort 操作

reverse：把列表中的数据元素头尾反转重新排列
sort：把列表中的数据元素按照大小顺序重新排列

## reversed / sorted 操作

得到重新排列的列表，而不影响原来的列表

```
>>> num = [1,2,7,4,3,9,0]
>>> num.reverse()
>>> num
[0, 9, 3, 4, 7, 2, 1]
>>> num.sort()
>>> num
[0, 1, 2, 3, 4, 7, 9]
>>> num.sort(reverse = True)
>>> num
[9, 7, 4, 3, 2, 1, 0]
```

# 列表的方法

| 方法名称 | 使用例子 | 说明 |
| --- | --- | --- |
| append | alist.append(item) | 列表末尾添加元素 |
| insert | alist.insert(i,item) | 列表中i位置插入元素 |
| pop | alist.pop() | 删除最后一个元素，并返回其值 |
| pop | alist.pop(i) | 删除第i个元素，并返回其值 |
| sort | alist.sort() | 将表中元素排序 |
| reverse | alist.reverse() | 将表中元素反向排列 |
| del | del alist[i] | 删除第i个元素 |
| index | alist.index(item) | 找到item的首次出现位置 |
| count | alist.count(item) | 返回item在列表中出现的次数 |
| remove | alist.remove(item) | 将item的首次出现删除 |

# 列表和元组的操作

## 合并

加法运算+：连接两个列表 / 元组

乘法运算*：复制n次，生成新列表 / 元组

## 列表/元组大小

len()：列表 / 元组中元素的个数

# 列表和元组的操作：索引和切片

## 索引

```
alist[n]或atuple[n]
```

- 可以用赋值语句给列表中的任何一个位置重新赋值
- 但元组属于不可变类型，索引只能获取对应位置中的数据值，不可重新赋值

## 切片

```
alist[start : end : step]
atuple[start : end : step]
```

# 列表和元组的操作

```
>>> []
[]
>>> list()
[]
>>> alist = [1, True, (
>>> alist[0]
1
>>> alist + ["Hello"]
[1, True, 0.234, 'Hell(
>>> alist * 2
[1, True, 0.234, 1, Tr(
>>> len(alist)
3
>>> 1 in alist
True
>>> alist
[1, True, 0.234]
>>> alist[1:3]
[True, 0.234]
>>> alist[0:3:2]
[1, 0.234]
>>> alist[::-1]
[0.234, True, 1]
```

```
>>> ()
()
>>> tuple()
()
>>> atuple = (1, True, 0.234)
>>> atuple[0]
1
>>> atuple + ("Hello",)
(1, True, 0.234, 'Hello')
>>> atuple * 2
(1, True, 0.234, 1, True, 0.234)
>>> len(atuple)
3
>>> 1 in atuple
True
>>> atuple
(1, True, 0.234)
>>> atuple[1:3]
(True, 0.234)
>>> atuple[0:3:2]
(1, 0.234)
>>> atuple[::-1]
(0.234, True, 1)
```

# 列表和元组的操作：查找和计算

## 查找

- in操作：判断某个元素是否存在于列表/元组中
- index操作：指定的数据在列表/元组的哪个位置
- count操作：指定的数据在列表/元组中出现过几次

## 计算

sum函数：将列表中所有的数据元素累加

min/max函数：返回列表中最小/最大的数据元素