3. 数据容器

Python 最的基本数据类型：**布尔型**、**整型**、**浮点型**以及**字符串型**。

本章将要提到的 **数据结构（容器）** 。在这一章中，我们会把之前所学的基本 Python 类型以更为复杂的方式组织起来。这些数据结构以后会经常用到。在编程中，最常见的工作就是将数据进行拆分或合并，将其加工为特定的形式

大多数编程语言都有特定的数据结构来存储由一系列元素组成的序列，这些元素以它们所处的位置为索引：从第一个到最后一个依次编号。

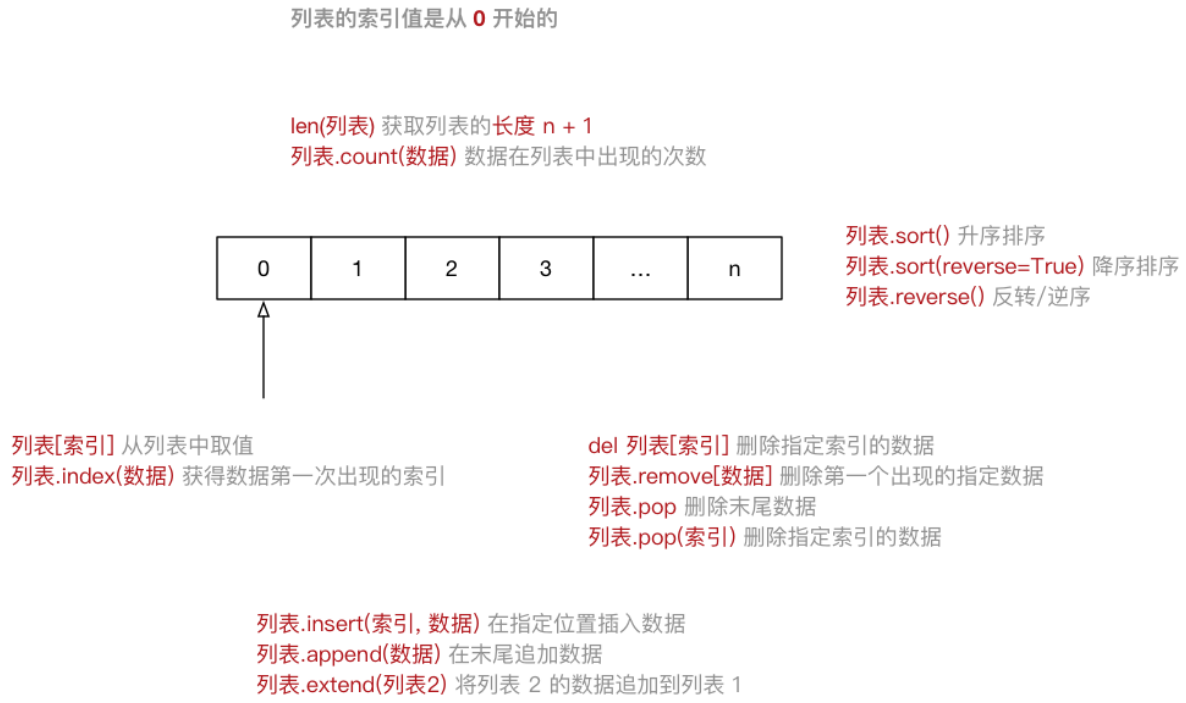
3.1 list（列表）

定义：列表是一种可变的、有序的数据结构，可以随时添加和删除其中的元素。

列表非常适合利用顺序和位置定位某一元素，尤其是当元素的顺序或内容经常发生改变时。与字符串不同，列表是可变的。你可以直接对原始列表进行修改：添加新元素、删除或覆盖已有元素。

创建列表

- `List`（列表） 是 `Python` 中使用 **最频繁** 的数据类型，在其他语言中通常叫做 **数组**
- 专门用于存储 **一串信息**
- 列表用 `[]` 定义，**数据** 之间使用 `,` 分隔
- 列表的 **索引** 从 `0` 开始。**索引** 就是数据在 **列表** 中的位置编号，**索引** 又可以被称为 **下标**



> 注意：从列表中取值时，如果 **超出索引范围**，程序会报错

```
# 用 [] 创建空列表
array2 = []
array2
```

创建一个列表

```
array3 = [1, 2, 3, 4, 5, 6, 7]
array3

# 列表中可以存放多种数据
array4 = [1, 2, 3, True, False, int, "str", array]
array
```

类型转化

将其他 **序列** 类型转化为列表

```
# 使用list()将其他数据类型转换成列表
s = 'hello world !'
list(s)

['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', ' ', '!']
```

列表取值

使用[offset]获取与修改元素

```
# 直接获取
s[0]

s[-1]
```

根据索引位置修改内容

```
array3[0] = 5
```

列表切片

> 切片使用索引值来限定范围，从一个大的数据容器中切出小的字符串

Python中符合序列的有序序列都支持切片（slice），例如列表，字符串，元组。

切片 方法适用于 **字符串、列表、元组**

格式：`[start:stop:step]`

`[起始值:结束值:步长]`

- start：起始索引，从0开始，-1表示结束
- stop：结束索引
- step：步长，end-start，步长为正时，从左向右取值。步长为负时，反向取值

```
array1 = list(range(10))

# 指定区间切片
print(array1[2:5])
print(array1[2:-2])
```

```
# 从头开始切片
print(array1[0:5])

# 切片到末尾
print(array1[0:])

# 省略参数切全部内容
print(array1[:])

# 指定步长切片
print(array1[0:5:1])
print(array1[0:5:2])
```

3.2 tuple（元组）

创建元组

定义：元组是一种不可变的序列类型

元组和列表类似，但属于**不可变序列，元组一旦创建，用任何方法都不可以修改其元素。**

元组的定义方式和列表相同，但定义时所有元素是放在一对圆括号“（）”中，而不是方括号中。

元组没有列表中那么多方法可以使用，因为不可变，所以安全，速度比列表快。

```
# 使用 tuple() 创建元组
>>>tuple()
()

# 使用 () 创建元组
>>> ()
()

>>> type(())
tuple
>>> type(tuple())
tuple
```

元组中只包含一个元素时，需要在元素后面添加逗号

```
info_tuple = (50, )
```

元组取值与切片

- 元组的取值、切片与列表时一样使用
- 不能对元组的元素进行删除，但是可以删除整个元组：

```
In [1]: t = tuple("01234")

In [2]: t
```

```
Out[2]: ('0', '1', '2', '3', '4')

# 直接获取
In [3]: t[0]
Out[3]: '0'

In [4]: t[-1]
Out[4]: '4'
# 不能修改
In [5]: t[0] = 1
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-c8aeb8cd20ae> in <module>
----> 1 t[0] = 1

TypeError: 'tuple' object does not support item assignment

In [6]: t[0:5]
Out[6]: ('0', '1', '2', '3', '4')
```

用逗号创建只有一个元素的元组：（1，）

```
In [1]: (1,)
Out[1]: (1,)

In [2]: (1)
Out[2]: 1
```

3.3 列表的常用操作

列表常用操作

在 `ipython` 中定义一个 **列表**，例如：`l= list()`

输入 `l.` 按下 `TAB` 键， `ipython` 会提示 **字典** 能够使用的函数如下：

```
append()  count()   insert()  reverse()
clear()   extend()  pop()     sort()
copy()    index()   remove()
```

可以到官方网址查询使用方法：

分类	关键字 / 函数 / 方法	说明
增加	append()	添加元素至尾部
	insert()	在指定位置插入数据
删除	clear()	清空列表
	pop()	默认弹出末尾数据
	pop(index)	弹出指定索引数据
	remove(data)	移除指定数据
修改	extend(列表2)	将列表2 的数据追加到列表
查询	count(数据)	统计数据出现的次数
	index(内容)	查询内容所在位置
其他	copy()	将列表复制一份
	sort()	排序
	reverse()	逆序列表

insert、append需要达到熟练的程度，pop / sort 用的也比较多

列表大部分的方法都是就地操作（修改原来的内容），拷贝一份进行操作（不会修改原来的值，但是会返回一个新的内容）

案例：

```
In [7]: arr = list(range(1, 5))

# 添加元素到末尾
In [8]: arr.append(5)

In [9]: arr
Out[9]: [1, 2, 3, 4, 5]
# 插入元素到第一个
In [10]: arr.insert(0, 0)

In [11]: arr
Out[11]: [0, 1, 2, 3, 4, 5]
# 默认弹出最后一个元素
In [12]: arr.pop()
Out[12]: 5
# 指定弹出第一个元素
In [13]: arr.pop(0)
Out[13]: 0

In [14]: arr
Out[14]: [1, 2, 3, 4]
# 指定删除内容为 4 的元素
In [15]: arr.remove(4)


In [18]: arr
```

```
Out[18]: [1, 2, 3, 4, 5, 6]
# 查询内容为 4 的元素在第几个位置
In [19]: arr.index(4)
Out[19]: 3
# 排序后将内容输出（默认为升序）
In [20]: arr.sort(reverse=True)

In [21]: arr
Out[21]: [6, 5, 4, 3, 2, 1]
# 排序后将内容输出
In [22]: arr.sort()

In [23]: arr
Out[23]: [1, 2, 3, 4, 5, 6]
```

3.4 元组与列表的区别

- 元组一旦定义就不允许更改。
- 元组没有`append()`、`extend()`和`insert()`等方法，无法向元组中添加元素。
- 元组没有`remove()`或`pop()`方法，也无法对元组元素进行`del`操作，不能从元组中删除元素。
- 从效果上看，`tuple( )`冻结列表，而`list( )`融化元组。

元组的优点

- **元组的速度比列表更快**。如果定义了一系列常量值，而所需做的仅是对它进行遍历，那么一般使用元组而不用列表。
- 元组对不需要改变的数据进行 “写保护” 将使得代码 **更加安全** 。
- **元组可用作字典的“键”，也可以作为集合的元素。列表永远不能当做字典键**使用，也不能作为集合的元素，因为列表不是不可变的。
- 不可变

3.5 dict（字典）

定义：字典是一种可变的、无序的、键值对的、复杂的数据容器

Python 中的字典是Python中一个键值映射的数据结构。

字典是一种可变无序数据容器，且可存储任意类型对象。字典的每个键值 **key=>value** 对用冒号 **:** 分割，每个键值对之间用逗号 **,** 分割，整个字典包括在花括号 **{}** 中

字典的定义

字典用`{}`定义

字典使用 **键值对** 存储数据，键值对之间使用`,`分隔

- **键** `key` 是索引
- **值** `value` 是数据
- **键** 和 **值** 之间使用`:`分隔
- **键必须是唯一的**
- **值** 可以取任何数据类型，但 **键** 只能使用 **字符串**、**数字**或 **元组**

```
dict1 = {
    "name": "小明",
    "age": 18,
    "gender": True,
    "height": 1.75
}
```

字典常用操作

使用 `字典['键']` 可以取到字典里面的内容。

```
print(dict1['name'])
```

使用 `字典['键'] = 值` 修改字典内容。

```
dict1['height'] = 0
print(dict1['height'])
```

当键不存在时添加内容。

```
dict1['weight'] = '小红'
print(dict)
```

字典常用方法

```
clear       items       setdefault
copy        keys        update
fromkeys    pop         values
get         popitem
```

> 有关 **字典** 的 **常用操作** 可以参照上图练习

方法	用法
get(key, default=None)	返回指定键的值，如果值不在字典中返回default值
keys()	以列表返回一个字典所有的键
values()	以列表返回字典中的所有值
update(dict2)	把字典dict2的键/值对更新到dict里
items()	以列表返回可遍历的(键, 值) 元组数组

循环遍历

遍历 就是 **依次** 从 **字典** 中获取所有键值对

```
# for 循环内部使用的 `key 的变量` in 字典

for k, v in dict1.items():
    print(k, v)

for k in dict1.keys():
    print(k, dict1[k])
```

> 提示：在实际开发中，由于字典中每一个键值对保存数据的类型是不同的，所以针对字典的循环遍历需求并不是很多

案例：对Python之禅的单词计数

统计python之禅中每个字符出现的次数

> ```
> from this import s
> ```

```
from this import s

d = {}
for i in s:
    if i not in d.keys():
        d[i] = 1
    else:
        d[i] += 1

for k, v in d.items():
    print(f"字符 {k} 出现了 {v} 次")
```

拓展：对字典进行排序输出

```
# 1. 字典是无序的，需要改变结构
l = list(d.items())


def func(temp):
    return temp[1]


# 2. 需要指定列表里面的元素的第二个值进行排序
l.sort(key=func, reverse=True)

print(l)
for i in l:
    print(f"字符 {i[0]} 出现了 {i[1]} 次")
```

3.6 set（集合）

> 与数学中的集合功能一样

集合（set）是一种无序的、可变的、不可重复的数据类型。

集合用 `{}` 创建，一般用作于去重

```
# 集合创建
set1 = {1, 2, 2, 3, 5, 6, 4, 4}
print(set1)

# 集合对列表进行去重
li = [1, 2, 2, 3, 5, 6, 4, 4]
li = list(set(li))
print(li)
```

集合运算（了解）

可以进行数学中的集合运算

```
set2 = {2, 3, 4}
set3 = {3, 4, 5}
# 交 and
print(set2 & set3)
# 并 or
print(set2 | set3)
# 差集
print(set1 - set2)
# 异或集
print(set2 ^ set3)
```

3.7 拓展

列表推导式

推导式comprehensions（又称解析式），是Python的一种独有特性。推导式是可以从一个数据序列构建另一个新的数据序列的结构体。

```
"""
    猫眼top100的网址
    第一页：https://maoyan.com/board/4?offset=0
    第二页：https://maoyan.com/board/4?offset=10
    第三页：https://maoyan.com/board/4?offset=20
    ....
    第十页：https://maoyan.com/board/4?offset=90

    请构建所有的请求地址
"""

url_list1 = []
base_url = 'https://maoyan.com/board/4?offset={page}'
for page in range(10):
```

```
    url_list1.append(base_url.format(page=page*10))

print(url_list1)

print([base_url.format(page=page*10) for page in range(10)])
```

序列解包

解包在英文里叫做 Unpacking，就是将容器里面的元素释放出来。Python 中的解包是自动完成的，例如：

```
# 元组解包
a, b, c = (1, 2, 3)
print(a, b, c)

# 列表解包
a, b, c = [1, 2, 3]
print(a, b, c)

# 用 _ 收集不用的变量（了解）
_, _, c = tuple('abc')

# 另外一种解包方式
print(*range(10))
```

`_` 是被舍弃的变量