第七次课

目标

列表

元组

字典

回顾：

Python 中数据类型可以分为 数字型 和 非数字型

数字型

整型 (int)

浮点型（float）

布尔型（bool）

真 True 非 0 数 —— 非零即真

假 False 0

复数型 (complex)

主要用于科学计算，例如：平面场问题、波动问题、电感电容等问题

非数字型

字符串

列表

元组

字典

在 Python 中，所有 **非数字型变量** 都支持以下特点：

都是一个 **序列** sequence，也可以理解为 **容器**

取值 **[]**

遍历 **for in**

计算长度、最大/最小值、比较、删除

链接 **+** 和 重复 *****

切片

一 列表

1.1 列表的定义

1. 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**

2. 专门用于存储 **一串 信息**

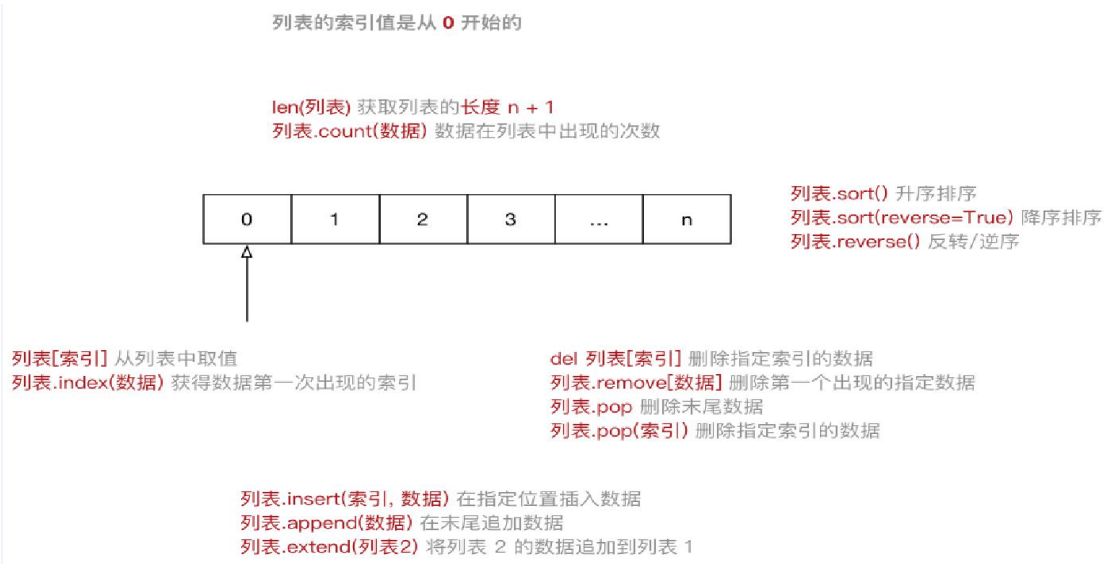
3. 列表用 **[]** 定义，**数据** 之间使用 **，**分隔

4. 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**

```
name_list = ["zhangsan", "lisi", "wangwu"]
```



1.2 列表的操作

在 ipython3 中定义一个 列表，例如：**name_list = []**

主要的方法如下：

```
name_list.append   name_list.count    name_list.insert   name_list.reverse

name_list.clear    name_list.extend   name_list.pop      name_list.sort

name_list.copy     name_list.index    name_list.remove
```

序号	分类	关键字 / 函数 / 方法	说明
1	增加	列表.insert(索引, 数据)	在指定位置插入数据
		列表.append(数据)	在末尾追加数据
		列表.extend(列表 2)	将列表 2 的数据追加到列表
2	修改	列表[索引] = 数据	修改指定索引的数据
3	删除	del 列表[索引]	删除指定索引的数据
		列表.remove[数据]	删除第一个出现的指定数据
		列表.pop	删除末尾数据
		列表.pop(索引)	删除指定索引数据
		列表.clear	清空列表
4	统计	len(列表)	列表长度
		列表.count(数据)	数据在列表中出现的次数
5	排序	列表.sort()	升序排序
		列表.sort(reverse=True)	降序排序
		列表.reverse()	逆序、反转

del 关键字（科普）

使用 del 关键字(delete) 同样可以删除列表中元素

del 关键字本质上是用来 **将一个变量从内存中删除的**

如果使用 del 关键字将变量从内存中删除，后续的代码就不能再使用这个变量了

```
del name_list[1]
```

在日常开发中，要从列表删除数据，建议 使用列表提供的方法

1.3 循环遍历

遍历 就是 **从头到尾** 依次 从 **列表** 中获取数据

在 **循环体内部** 针对 **每一个元素**，执行相同的操作

在 Python 中为了提高列表的遍历效率，专门提供的 **迭代 iteration** 遍历

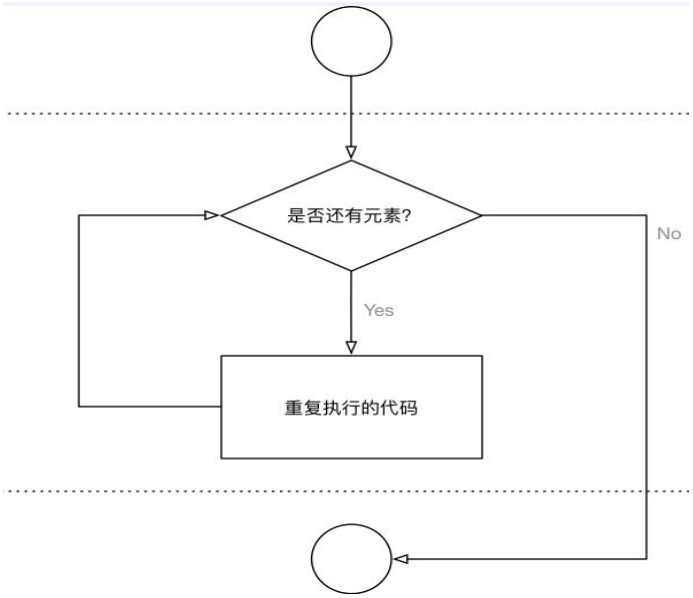
使用 for 就能够实现迭代遍历

```
# for 循环内部使用的变量 in 列表

for name in name_list:

    循环内部针对列表元素进行操作

    print(name)
```



1.4 应用场景

尽管 Python 的 **列表** 中可以 **存储不同类型的数据**

但是在开发中，更多的应用场景是

列表 **存储相同类型的数据**

通过 迭代遍历，在循环体内部，针对列表中的每一项元素，**执行相同的操作**

二、元组

2.1 元组的定义

Tuple（元组）与列表类似，不同之处在于元组的 元素不能修改

元组 表示多个元素组成的序列

元组 在 Python 开发中，有特定的应用场景

用于存储 **一串 信息，数据** 之间使用 **，** 分隔

元组用 **()** 定义

元组的 **索引** 从 **0** 开始

索引 就是数据在 元组 中的位置编号

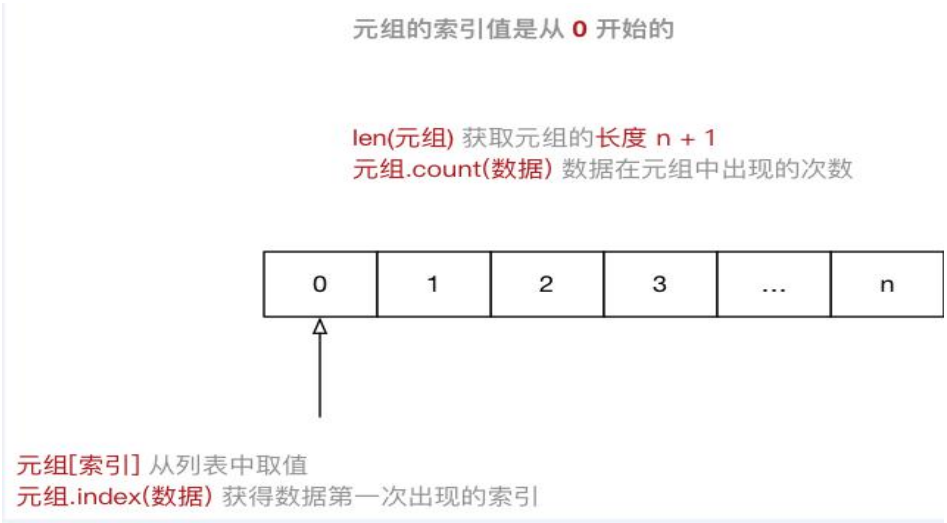
```
info_tuple = ("zhangsan", 18, 1.75)
```

创建空元组

```
info_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, )
```



2.2 元组的常用操作

在 python3 中定义一个 元组，例如：info = ()

```
info.count  info.index
```

2.3 循环遍历

取值 就是从 **元组** 中**获取存储**在**指定位置**的**数据**

遍历 就是 **从头到尾 依次** 从 **元组** 中**获取数据**

```

```

```
# for 循环内部使用的变量 in 元组 for item in info:

    循环内部针对元组元素进行操作

    print(item)
```

在 Python 中，可以使用 for 循环遍历所有非数字型类型的变量：列表、元组、字典 以及字符串

提示：在实际开发中，除非 **能够确认元组中的数据类型，否则针对元组的循环遍历需求并不是很多**

2.4 应用场景

尽管可以使用 for in 遍历 元组

但是在开发中，更多的应用场景是：

函数的 **参数** 和 **返回值**，一个函数可以接收 **任意多个参数**，或者 **一次返回多个数据**

有关 函数的参数 和 返回值，在后续 函数高级 给大家介绍

格式字符串，格式化字符串后面的 **()** 本质上就是一个**元组**

让列表不可以被修改，以保护数据安全

```
info = ("zhangsan", 18)

print("%s 的年龄是 %d" % info)
```

补充知识点：

元组和列表之间的转换

使用 list 函数可以把元组转换成列表

```
list(元组)
```

使用 tuple 函数可以把列表转换成元组

```
tuple(列表)
```

三、字典

3.1 字典的定义

dictionary（字典） 是 **除列表以外** Python 之中 **最灵活** 的数据类型

字典同样可以用来 **存储多个数据**

通常用于**存储** 描述一个 **物体** 的**相关信息**

和列表的区别

列表 是 **有序** 的对象集合

字典 是 **无序** 的对象集合

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**

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

len(字典) 获取字典的 键值对数量

key	value
name	小明
age	18
gender	True
height	1.75

字典.keys() 所有 key 列表
字典.values() 所有 value 列表
字典.items() 所有 (key, value) 元组列表

字典[key] 可以从字典中取值，key 不存在会报错
字典.get(key) 可以从字典中取值，key 不存在不会报错

del 字典[key] 删除指定键值对，key 不存在会报错
字典.pop(key) 删除指定键值对，key 不存在会报错
字典.popitem() 随机删除一个键值对
字典.clear() 清空字典

字典[key] = value
    如果 key 存在，修改数据
    如果 key 不存，新建键值对
字典.setdefault(key, value)
    如果 key 存在，不会修改数据
    如果 key 不存在，新建键值对
字典.update(字典2) 将字典 2 的数据合并到字典 1

3.2 字典常用操作

在 ipython3 中定义一个 字典，例如：**xiaoming = {}**

```
xiaoming.clear      xiaoming.items      xiaoming.setdefault

xiaoming.copy       xiaoming.keys       xiaoming.update

xiaoming.fromkeys   xiaoming.pop        xiaoming.values

xiaoming.get        xiaoming.popitem
```

3.3 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 in xiaoming:

    print("%s: %s" % (k, xiaoming[k]))
```

3.4 应用场景

尽管可以使用 for in 遍历 **字典**

但是在开发中，更多的应用场景是：

使用 **多个键值对**，存储 **描述一个 物体 的相关信息** —— 描述更复杂的数据信息

将 **多个字典** 放在 **一个列表** 中，再进行遍历，在循环体内部针对每一个字典进行 **相同的处理**

```
card_list = [{"name": "张三",

             "qq": "12345",

             "phone": "110"},

            {"name": "李四",

             "qq": "54321",

             "phone": "10086"}

            ]
```