## 第四章 列表、元组和字典 章节时长：30min

第一节 列表 6min

第二节 二维列表 3.5min

第三节 列表方法 7.5min

第四节 元组 2min

第五节 Python赋值语句技巧之序列解包 3min

第六节 字典 8min

## 第一节 列表

1. 定义一个列表（使用方括号）：

（1）字符串列表：

**变量名 = [‘字符串1’，‘字符串2’，‘字符串3’，……]**

**print（变量名）**

- 如：

name = ['John', 'Bob', 'Mosh', 'Sarah', 'Mary']

print（name）

（2）数字列表：

**变量名 = [数字，数字，数字，数字，……]**

**print（变量名）**

- 如：

numbers = [1, 2, 3, ,4, 5]

print（numbers）

2. 列表的索引

（1）作用：列表的索引可以帮我们访问列表中的单个元素

可参考字符串的索引——访问字符串中的单个字符

（2）格式（以字符串列表为例）：

① 索引单个元素

**变量名[索引]**

**print（变量名[索引]）**

* 如：

print（name[2]）

② 索引连续的一系列元素（使用冒号）

**变量名[索引1：索引2 ]**

* 如：print（name[2：]） # 只指定开始索引，就默认一直访问到末尾

print（name[2：4]） # 只访问开始索引所指项，不访问结束索引所指项

print（name[：4]） # 只指定结束索引，就默认从第一项开始访问

print（name[：]） # 开始和结束索引都不指定，默认访问所有项

print（name[0：]） # 默认访问所有项

（3）注意：

① 列表的索引不会修改原来的列表，而是获取一个副本并返回一个新列表

② 索引数字：

- 0索引第一项

- n索引第n+1项

- 负数是倒序索引，负几索引倒数第几项（如-1索引最后一项，-2索引倒数第二项）

3. 修改原列表中的单个元素（使用索引实现）

（1）格式：

names = ['John', 'Bob', 'Mosh', 'Sarah', 'Mary']   
**names[0] = 'Jon'**   
print(names)

（2）思路：先使用索引找到待修改元素，再用赋值语句直接重新赋值，达到修改目的；

此时原列表的内容已被修改。

4. 小练习：找出列表中的最大值（见代码整理）

- 本练习需要用到for循环、if语句的知识，可以待学完第五章相关知识后再回看本练习

## 第二节 二维列表

1. 作用：二维列表功能非常强大，并且在数据科学（data science）和机器学习（machine learning）中有着广泛的应用

2. 实质：二维列表本质上是一个列表，列表中的每一个元素都是另一个列表

- 类似于C语言中的二维数组

3. 应用举例：用二维列表定义一个矩阵（matrix）

（1）定义一个3x3矩阵：

**matrix = [**  # 外部列表  
 **[1, 2, 3],   
 [4, 5, 6],**  # 内部列表（每行）  
 **[7, 8, 9]   
]**

（2）访问矩阵中的单个数字：

- 格式：

**变量名[行索引][列索引]**

- 如：

matrix[0][1] # 访问第一行第一列的数字

- 注：索引从零起

（3）使用嵌套循环来遍历矩阵中的所有元素

- 注意：本部分知识涉及到5.4嵌套循环的知识，可以在学习完第五章相关内容（5.3或5.4）后再回看

- 思路：外部for循环的每次迭代访问一行；

内部for循环的每次迭代访问当前行的一个元素

## 第三节 列表方法

1. 概念：

- **列表方法**（list methods），也可称列表函数（list functions），是一系列可以对列表使用的操作

- 如：添加新项，删除一个原有项，检查一个项是否在列表中

2. append方法：

（1）功能：向列表中添加一个新项，该新项会被添加到列表末尾

（2）格式（以数字列表为例）：

**变量名.append（数字）**

- 如：

numbers.append（20）

3. insert方法：

（1）功能：向列表中的指定位置插入一个新项，该指定位置由索引指示

（2）格式：

**变量名.insert（索引，待插入数字）**

- 如：

numbers.insert(2, 20)

# 新项在新列表的索引为2，即将新项插入到原列表索引为2的项前面

4. remove方法：

（1）功能：删去列表中指定元素

（2）格式：

**变量名.remove（待删除数字）**

- 如：

numbers.remove（5）

5. clear方法：

（1）功能：清除列表中的所有元素

（2）格式：

**变量名.clear（）**

- 如：

numbers.clear（）

6. pop方法：

（1）功能：删去列表中的最后一个元素

（2）格式：

**变量名.pop（）**

- 如：

numbers.pop（）

7. index方法：

（1）功能：检查列表中是否存在某个元素，如果存在，则返回其在列表中的索引

（2）格式：

**变量名.index（待查找数字）**

- 如：

numbers.index（5）

（3）局限：如果待查找的元素不存在，会报错

8. in操作符：

（1）功能：检查列表中是否存在某个元素，返回值是一个布尔值。如果存在，则返回True，不存在则返回False

（2）格式：

**待查找元素 in 变量名**

print（待查找元素 in 变量名）

- 如：

print（50 in numbers）

（3）优势：与index方法相比，不会报错，使代码更加安全

9. count方法：

（1）功能：计算值相同的元素在列表中出现的次数，返回计算所得的次数

（2）格式：

**变量名.count（元素的值)**

print（变量名.count（元素的值)）

- 如：

print（numbers.count（5））

10. sort方法：

（1）功能：将列表中的所有元素按值大小升序排列

（2）格式：

**变量名.sort（)**

print（变量名）

- 如：

numbers.sort（）

print（numbers）

11.reverse方法：

（1）功能：对列表倒序排列，即把列表原来的顺序反过来

（2）格式：

**变量名.reverse（元素的值)**

print（变量名）

- 如（与sort方法结合，实现按值降序排列）：

numbers.sort（） # 先按值升序

numbers.reverse（） # 逆序排列——升序变降序

print（numbers）

12. copy方法：

（1）功能：拷贝一个列表，生成它的一个新副本

（2）格式：

**变量名2 = 变量名1.copy（)**

- 如：

numbers2 = numbers.copy() # numbers和numbers2是两个独立的列表  
numbers.append(10) # 对原列表的操作不会影响到副本numbers2

## 第四章 列表、元组和字典

第四节 元组 2min

第五节 Python赋值语句技巧之序列解包 3min

第六节 字典 8min

## 第四节 元组tuples

1. 定义：

- 元组是有序且不可更改的集合。在 Python 中，元组是用圆括号编写的。

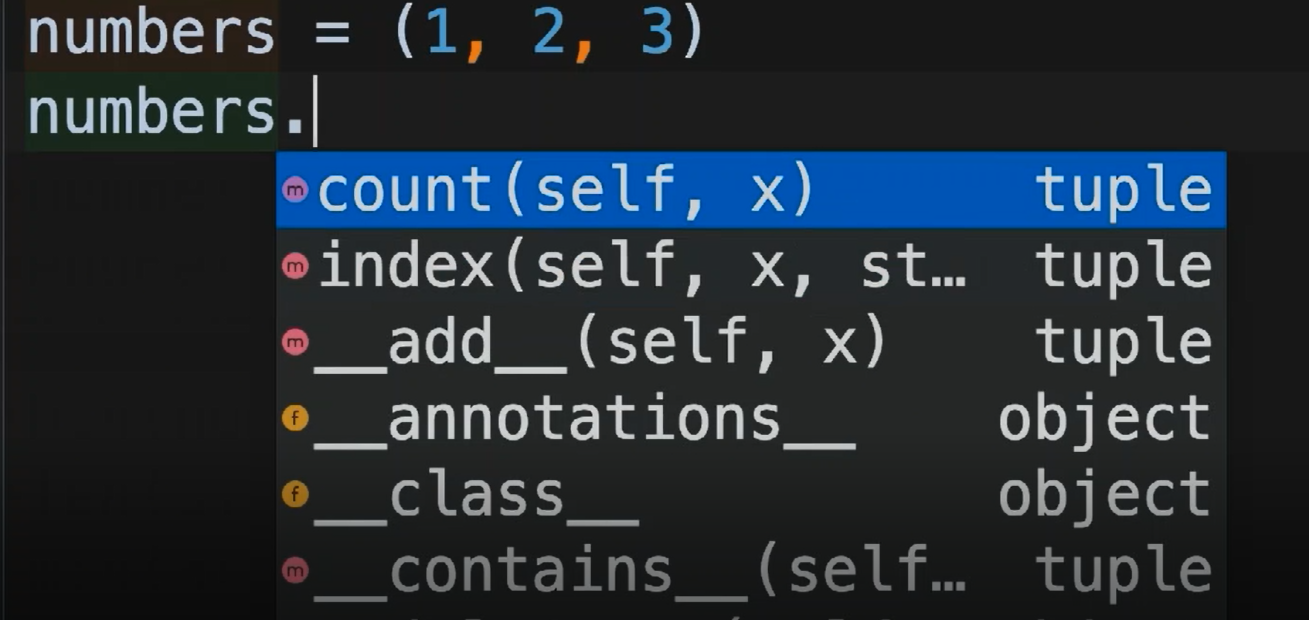
2. 作用：

- 可以用来存储一系列项，与列表相似（具体知识点和操作可以参考列表）

3. 特点：

（1）与列表不同的是，元组是不可变的（immutable），我们只能得到元组内部的信息，而不能修改元组中的原有内容，如添加新项、删除项；

（2）根据（1）中特点，可知元组没有append、 insert、remove、clear、pop、sort、reverse等修改类方法，只有count和index两个方法（可复习4.3列表方法）



（3）列表用方括号定义，但是元组用圆括号定义

（4）不能使用赋值语句改变元组中的项的值，否则会报错

4. 一个适于使用元组来代替列表的情况：

- 如果你想要创建一系列项，并且想确保在编写程序过程中你不会不小心修改了那个列表，那么可以使用元组代替列表来

**【元组补充知识】**<https://www.w3school.com.cn/python/python_tuples.asp>

相关笔记详见代码整理

## 第五节 Python赋值语句技巧之序列解包

1. 使用情况：

（1）想要分别获取一个集合中单个元素的值，并且将它们存储在不同变量中

（2）可以应用在元组tuple或列表list

2. 格式（以元组为例）：

coordinates = (1, 2, 3)

**x, y, z = coordinates** # unpacking，变量的顺序与元组内部顺序相对应

- 等价于：

x = coordinates[0]  
y = coordinates[1]  
z = coordinates[2]

## 第六节 字典 Dictionaries

1. 使用情况：

- 想要存储以一组“键值对”为形式的信息

【键值对】key-value pairs 或 （key，value）pairs 即 （关键字，值）对

- 如：一个客户有很多属性，如名字、邮件地址、电话号码等，每一个属性都有一个值，这一组关键字（即上述的“属性”）和值就构成了“key value pairs”，可以用字典来定义

2. 格式：

**变量名 = {**

**“键名称1” ： 值1，** # 此处相同名称的键只能被赋值一次

**“键名称2” ： 值2，**

**“键名称3” ： 值3**

**}**

3. 注意：

（1）每个键在字典中都是唯一的，不能被重复赋值

（2）键的值可以是很多类型，如一个字符串、一个数字、一个布尔值、一个列表……

4. 访问字典中的每一项（使用方括号）：

（1）格式：

**变量名[“键名称”]**  # 会返回该键对应的值

**print（变量名[“键名称”]）**

（2）局限：如果输入不存在的键名，或错误输入键名，系统会报错；

改进：使用get方法

5. get方法：

（1）一般格式：

**变量名.get（“键名”）**

**print（变量名.get（“键名”））**

（2）返回值：

- 如果键名存在且输入正确，返回该键所对应的值

- 如果键名不存在或输入错误（包括大小写错误），返回None值

- 【None值】代表“值缺失”，即“此处没有值”

（3）访问失败时，赋默认值：

① 格式：

**变量名.get（“键名”，“默认值”）**

**print（变量名.get（“键名”，“默认值”））**

② 使用情况：输入不存在的键名时，为了不获得None值，可以将一个默认值赋给该键名

* 如：

**print（customer.get（“name”，“Jan 1 1980”））**

6. 给字典中的原有的键值对重新赋值

- 格式：

**变量名[“键名”] = 新值**

- 如：

**customer[“name”] = “Jack Smith”**

7. 向字典中添加一个新的键值对，并且赋值

- 格式：

**变量名[“新键名”] = 值**

- 如：

**customer[“birthdate”] = “Jan 1 1980”**