Python_数据容器

数据容器：列表（List）、元组（Tuple）、字典（Dictionary）、集合（Set）

和列表、元组、字符串等定义基本相同：

- 列表使用：[]
- 元组使用：()
- 字符串使用：""
- 集合使用：{}

列表list：有序可变集合

- tip：列表可以一次存贮多个数据，且可以为不同的数据类型，支持嵌套

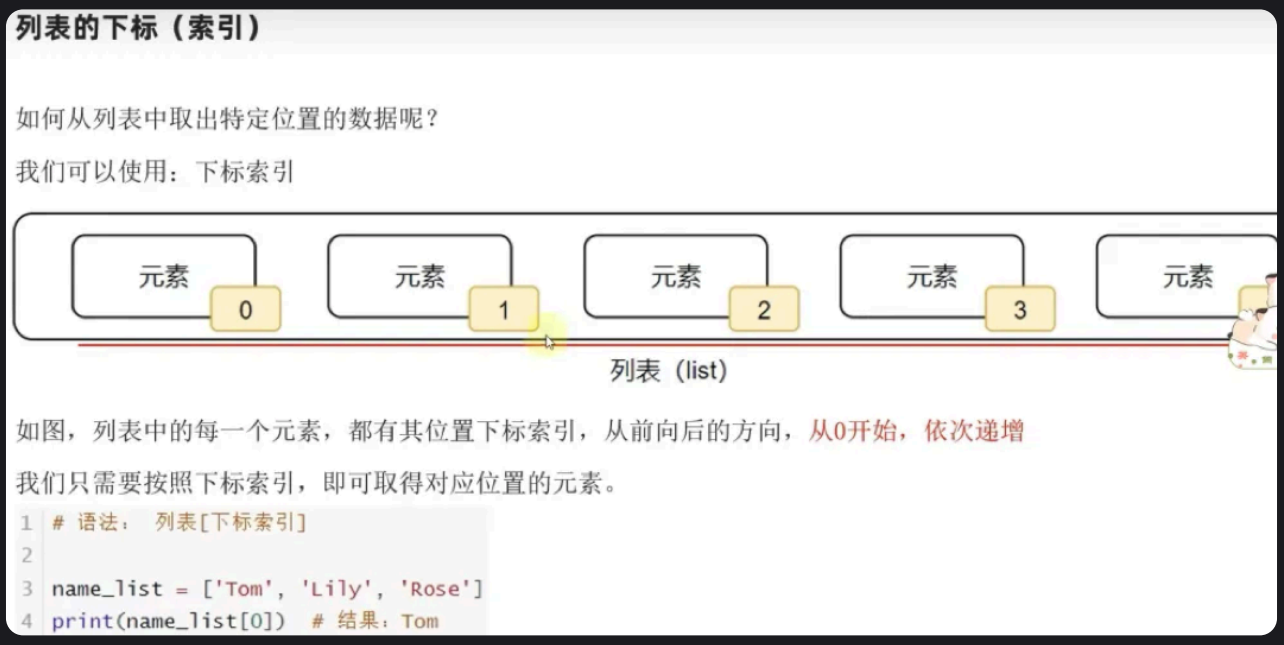
定义列表：

```python
name_list =['itheima','itcast','python']
print(name_list)
print(type(name_list))

'''
itheima
<class 'list'>
'''
```

定义空列表：

```python
name_list =[] #定义空列表1
name_list =list() #定义空列表2
```

列表的下标：



或者，可以反向索引，也就是从后向前：从-1开始，依次递减（-1、-2、-3.....）

如图，从后向前，下标索引为：-1、-2、-3，依次递减。

```python
# 语法：  列表[标号]

name_list = ['Tom', 'Lily', 'Rose']
print(name_list[-1])  # 结果：Rose
print(name_list[-2])  # 结果：Lily
print(name_list[-3])  # 结果：Tom
```

列表常用操作：

编号	使用方式	作用
1	列表.append(元素)	向列表中追加一个元素
2	列表.extend(容器)	将数据容器的内容依次取出，追加到列表尾部
3	列表.insert(下标, 元素)	在指定下标处，插入指定的元素
4	del 列表[下标]	删除列表指定下标元素
5	列表.pop(下标)	删除列表指定下标元素
6	列表.remove(元素)	从前向后，删除此元素第一个匹配项
7	列表.clear()	清空列表
8	列表.count(元素)	统计此元素在列表中出现的次数
9	列表.index(元素)	查找指定元素在列表的下标 找不到报错ValueError
10	len(列表)	统计容器内有多少元素

```python
name_list =['itheima','itcast','python']
print(name_list)

name_list.append('abc')
print(name_list)

del name_list[0]
pop=name_list.pop[2]
print(name_list)
print(pop)

index = name_list.index('itcast')
print(index)
```

```
print(name_list)
print(name_list[index])
print(f'在第{index}个')
print(len(name_list))
print(count(itcast))

'''
['itheima', 'itcast', 'python']
['itheima', 'itcast', 'python', 'abc']
['itcast', 'python']
abc
0
['itcast', 'python']
itcast
在第0个
2
1
'''
```

For\while 循环遍历列表：

```
name_list = ['itheima','itcast','python']
for element in name_list:
    print(element)
index = 0
while index < len(name_list):
    print(name_list[index])
    index += 1

'''
itheima
itcast
python
itheima
itcast
python
'''
```

元组tuple:不可以修改的list

元组定义：

```
my_tuple = ('a','b','c')
```

定义空元组：

```
my_tuple = ()
my_tuple = tuple
```

元组操作：与列表操作几乎一致，不过操作较少

元组的相关操作

编号	方法	作用
1	index()	查找某个数据，如果数据存在返回对应的下标，否则报错
2	count()	统计某个数据在当前元组出现的次数
3	len(元组)	统计元组内的元素个数

```
# 根据下标（索引）取出数据
t1 = (1, 2, 'hello')
print(t1[2]) # 结果：'hello'

# 根据index()，查找特定元素的第一个匹配项
t1 = (1, 2, 'hello', 3, 4, 'hello')
print(t1.index('hello'))    # 结果： 2

# 统计某个数据在元组内出现的次数
t1 = (1, 2, 'hello', 3, 4, 'hello')
print(t1.count('hello')) # 结果：2

# 统计元组内的元素个数
t1 = (1, 2, 3)
print(len(t1))  # 结果 3
```

元组的遍历：同列表

```
t4 = ("i", "am", "good", "student", "17")
for element in t4:
    print(element)
index = 0
while index < len(t4):
    print(t4[index])
    index += 1
'''
```

```
i
am
good
student
17
i
am
good
student
17
'''
```

字典dict：

- 由两部分组成，一般通过字key 找到信息值value
- 字典的key是不能重复的 如果出现重复的情况的话 后面的会覆盖前面的

老师有一份名单，记录了学生的姓名和考试总成绩。

姓名	成绩
王力鸿	77
周杰轮	88
林俊节	99

现在需要将其通过Python录入至程序中，并可以通过学生姓名检索学生的成绩。

使用字典最为合适：

可以通过Key（学生姓名），取到对应的Value（考试成绩）

字典的定义：

```
dic_score ={
    'a':{"1":"100","2":'99'},
    'b': {"1": "95","2": '85'},
    'c':{"1":"50","2":"60"}
}
```

字典的定义，同样使用{}，不过存储的元素是一个个的：键值对，如下语法：

```
# 定义字典字面量
{key: value, key: value, ....., key: value}
# 定义字典变量
my_dict = {key: value, key: value, ....., key: value}
# 定义空字典
my_dict = {}           # 空字典定义方式1
my_dict = dict()       # 空字典定义方式2
```

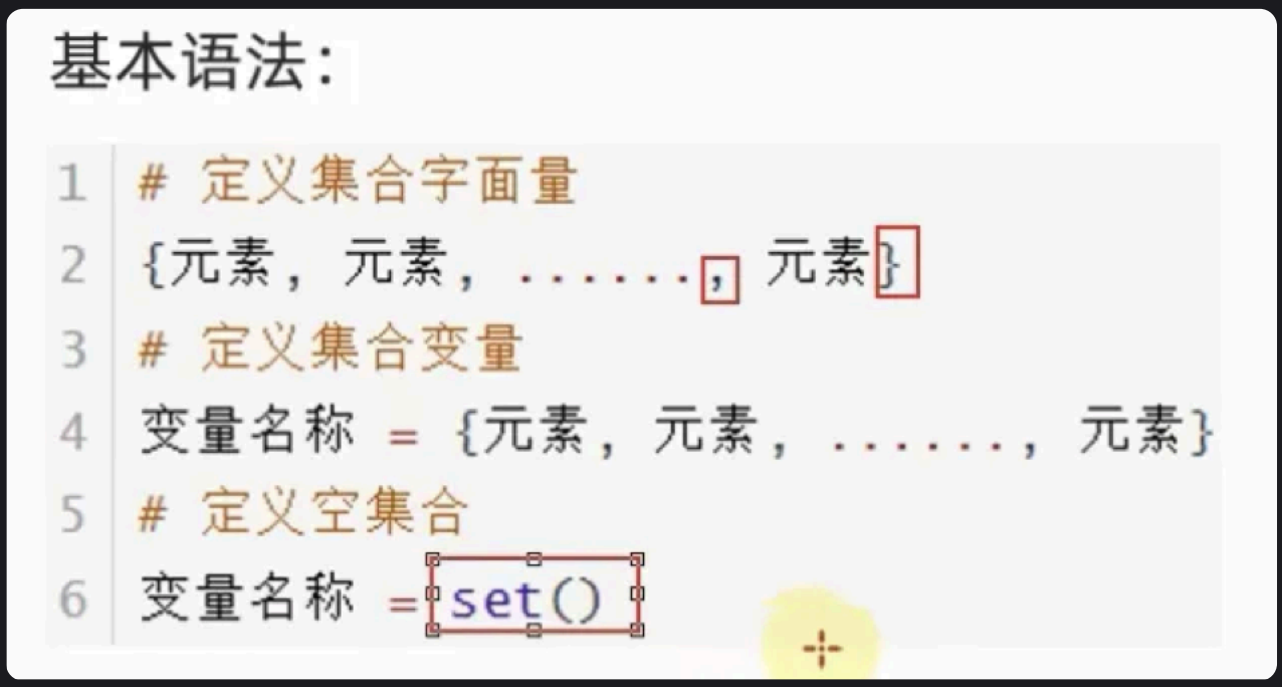
字典的使用与循环：

```
dic_score ={
    'a':{"1":"100",
    "2":'99'},
    'b': {"1": "95",
          "2": '85'},
    'c':{"1":"50",
         "2":"60"}}
print(dic_score)
print(type(dic_score))
print(dic_score['a'])

#方法一：得到所有的key进行遍历
keys = dic_score.keys()
print(keys)
for i in keys:
    print(dic_score[i])

#方法二：直接遍历
for key in stu_score:
    print(key)
    print(stu_score[key])
```

集合set：

- 最主要的特点就是不支持元素重复 一般用来去重 但是他的内容和前面的地方不一样的点在于其无序性的特点
- 定义空集合一定是 set（） 因为{} 是空字典

基本语法：



集合操作：

集合常用功能总结

编号	操作	说明
1	集合.add(元素)	集合内添加一个元素
2	集合.remove(元素)	移除集合内指定的元素
3	集合.pop()	从集合中随机取出一个元素
4	集合.clear()	将集合清空
5	集合1.difference(集合2)	得到一个新集合，内含2个集合的差集 原有的2个集合内容不变
6	集合1.difference_update(集合2)	在集合1中，删除集合2中存在的元素 集合1被修改，集合2不变
7	集合1.union(集合2)	得到1个新集合，内含2个集合的全部元素 原有的2个集合内容不变

集合运算：高中内容

- 交集（Intersection）：返回两个集合中都包含的元素，即它们的公共元素。格式：set1 & set2 或 set1.intersection(set2)
- 并集（Union）：并集运算会返回两个集合的所有元素，但不会重复，包含两者中的所有唯一元素。格式：set1 | set2 或 set1.union(set2)
- 差集（Difference）：差集运算会返回存在于第一个集合但不存在于第二个集合中的元素。格式：set1 - set2 或 set1.difference(set2)

集合的遍历：

```
#方法一
set1 = {1, 2, 3, 4, 5}
for element in set1:
    print(f"集合的元素有: {element}")
```

Range遍历：

```
#list
my_list = ['apple', 'banana', 'cherry']
# 使用 range() 和 len() 来按索引访问列表
for i in range(len(my_list)):
    print(f"索引 {i} 对应的元素是 {my_list[i]}")
#tuple
my_tuple = (10, 20, 30)
# 使用 range() 按索引访问元组元素
for i in range(len(my_tuple)):
    print(f"索引 {i} 对应的元素是 {my_tuple[i]}")
```