

# 笔记

版权：智泊AI

作者：Jeff

## 一、tuple元组

### 1.概述

和列表相似，本质上是一种有序的集合

元组和列表的不同之处：

a.列表：[] 元组：()

b.列表中的元素可以进行增加和删除操作，但是，元组中的元素不能修改

【元素：一旦被初始化，将不能发生改变】

### 2.创建元组

创建列表：

创建空列表：list1 = []

创建有元素的列表：list1 = [元素1, 元素2, ...]

创建元组

创建空元组：tuple1 = ()

创建有元素的元组：tuple1 = (元素1, 元素2, ...)

代码演示：

```
# 1.创建空元组
tuple1 = ()
print(type(tuple1)) # <class 'tuple'>

# 2.创建带有元素的元组
tuple2 = (12,34,6,87)
print(tuple2)
print(type(tuple2)) # <class 'tuple'>

# 3.元组中的元素可以是各种类型
tuple3 = (12,34,4.12,"hello",True)
print(tuple3)

# 注意:创建的元组只有一个元素时，会在元素的后面加上一个逗号，
tuple4 = (2)
print(tuple4)
print(type(tuple4)) # <class 'int'>
```

```
tuple5 = (3,)
print(tuple5)
print(type(tuple5)) #<class 'tuple'>
```

### 3.元组元素的访问

代码演示：

```
tuple1 = (14,32,35,7,87)
# 1.访问元组的元素,使用下标访问,下标默认从0开始
print(tuple1[1])
# print(tuple1[5]) # tuple index out of range 索引越界

print(tuple1[-1]) # 87 访问元组的最后一个元素 下标是-1
print(tuple1[-3]) # 35

# 2. 元组的元素的值不能进行修改
# tuple1[2] = 99
# print(tuple1) # 'tuple' object does not support item assignment
```

### 4.元组操作

代码演示：

```
# 1.合并元组 +
tuple1 = (12,34,56)
tuple2 = (3,12,56,"hello")
print(tuple1 + tuple2)

# 2.重复元组中的元素 *
tuple3 = (23,45,67)
print(tuple3 * 4)

# 3.判断指定元素是否在元组中 使用成员运算符 in 和 not in
print(56 in tuple2)
if "hello" in tuple2:
    print("终于找到你")
else:
    print("你在哪里呢!")

# 4.元组的截取(切片)
tuple4 = (12,3,5,7,98)
print(tuple4[1:4]) # (3, 5, 7)
print(tuple4[-1:]) # (98,)
print(tuple4[:2]) # (12, 3)
```

## 5.元组功能

代码演示：

```
# 元组常见的功能
#1.len  获取元组的长度
print(len(tuple4))  # 5

# 2.获取元组中的最大值max()和最小值min()
print(max(tuple4))  # 98
print(min(tuple4))  # 3

# 3.其他数据类型转换为元组  tuple()
list1 = [12,34,57,89]
print(type(list1))  # <class 'list'>
print(type(tuple(list1)))  # <class 'tuple'>

# 4.遍历元组
# 第一种方式：for-in
for i in tuple4:
    print(i)

# 第二种方式：通过下标访问
for i in range(len(tuple4)):
    print(tuple4[i])

# 第三种方式：enumerate() 返回索引和元素
for i,n in enumerate(tuple4):
    print(i,n)
```

## 二、字典dict

### 1.概念

列表和元组的使用缺点：当存储的数据要动态添加、删除的时候，我们一般使用列表，但是列表有时会遇到一些麻烦

解决方案：既能存储多个数据，还能在访问元素的很方便的定位到需要的元素，采用字典

语法：{键1: 值1, 键2: 值2, 键3: 值3, ..., 键n: 值n}

说明：键值对: key-value

- 字典和列表类似，都可以用来存储多个数据
- 在列表中查找某个元素时，是根据下标进行的；字典中找某个元素时，是根据'名字'（就是冒号:前面的那个值，例如上面代码中的'name'、'id'、'sex'）
- 字典中的每个元素都由2部分组成，键:值。例如 'name': '班长', 'name'为键, '班长'为值

- 键可以使用数字、布尔值、元组，字符串等不可变数据类型，但是一般习惯使用字符串，切记不能使用列表等可变数据类型
- 每个字典里的key都是唯一的，如果出现了多个相同的key,后面的value会覆盖之前的value

习惯使用场景：

- 列表更适合保存相似数据，比如多个商品、多个姓名、多个时间
- 字典更适合保存不同数据，比如一个商品的不同信息、一个人的不同信息

## 2.定义字典

```
# 1.定义空字典 {}
dict1 = {}
print(type(dict1)) # <class 'dict'>

# 2.定义非空字典
# 第一种定义字典的方式
dict2 = {"name": "小明", "age": 25, "sex": "男", "love": "篮球"} # 最常用
print(dict2)
print(type(dict2))
print(dict2["name"], dict2["love"]) # 访问字典
```

## 3.字典的操作

```
# 1.访问字典中的元素
# 第一种方式：直接通过下标访问
dict1 = {"name": "中国医生", "author": "刘伟强", "person": "张涵予"}
print(dict1['author'])
# print(dict1['money']) # 访问字典中不存在的key时,直接报错

# 第二种方式:通过get()方法获取
print(dict1.get('name'))
print(dict1.get('money')) # None 访问字典中不存在的key时,返回None
print(dict1.get('money', 10000)) # 访问字典中不存在的key时,若传递了第二个参数,则使用第二个参数的值

#2. 获取字典的长度 len()
print(len(dict1))

# 3.获取字典中所有的key
print(dict1.keys()) # dict_keys(['name', 'author', 'person'])

# 4.获取字典中所有的value
print(dict1.values()) # dict_values(['中国医生', '刘伟强', '张涵予'])

# 5.获取字典中的所有的key和value items()
print(dict1.items()) # dict_items([('name', '中国医生'), ('author', '刘伟强'), ('person', '张涵予')])
```

```
# 6.遍历字典
# 第一种方式: for in
for key in dict1: # 遍历字典中所有的key
    print(key)

# 第二种方式:遍历 字典中所有的值
for v in dict1.values():
    print(v)

# 第三种方式: items 遍历字典中所有的key和value
for k,v in dict1.items():
    print(k, '----', v)


# 合并字典 update()
dict2 = {"name": "刘哥", "money": "1999", "age": 23}
dict3 = {"sex": "男"}
dict2.update(dict3)
print(dict2)


dict4 = {"name": "iPhone17", "money": 10000, "color": "红色"}
# 增
dict4["size"] = 6.8
print(dict4)

# 改
dict4['color'] = "骚粉"
print(dict4)

# 删
# 第一种: pop() 删除指定的元素
# dict4.pop("color")
print(dict4)
# 第二种: popitem() 随机返回并删除字典中的最后一对key和value
dict4.popitem()
print(dict4)

# clear() 清空字典
dict4.clear()
print(dict4)
```

## 三、赋值、深拷贝、浅拷贝

#深浅拷贝的可视化视图

<http://pythontutor.com/live.html#mode=edit>

# 赋值：其实就是对象的引用(别名)

```
list = [12,34,57,9]
```

```
list1 = list
```

```
list[1] = 78
```

```
# print(list,list1)
```

# 浅拷贝：拷贝父对象,不会拷贝对象内部的子对象.浅拷贝一维列表的时候,前后两个列表是独立的.

```
import copy
```

```
a = [12,35,98,23] # 一维列表
```

```
b = a.copy()
```

```
a[1] = 67
```

```
# print(a,b)
```

# 浅拷贝在拷贝二维列表的时候,只能拷贝最外层列表,不能拷贝父对象中的子对象,当修改子对象中的值的时候,新拷贝的对象也会发生变化

```
c = [14,53,25,[31,89,26],42] # 二维列表
```

```
d = c.copy()
```

```
c[3][1] = 11
```

```
print(c,d)
```

# 若要解决浅拷贝处理二维列表时的问题,需要使用深拷贝解决

```
e = [14,53,25,[31,89,26],42] # 二维列表
```

```
f = copy.deepcopy(e)
```

```
e[3][1] = 11
```

```
print(e,f)
```

## 四、set集合【了解】

### 1.概述

和数学上的集合基本是一样的,

特点:不允许有重复元素,可以进行交集,并集,差集的运算

本质: 无序,无重复元素的集合

## 2.创建

set(列表或者元组或者字典)

代码演示：

```
# 创建空集合
set1 = set()
print(set1)
print(type(set1)) #<class 'set'>

# 创建包含元素的集合
set2 = {12,345,633,21}
set3 = {"hello","world","everyone"}
print(set2, set3)
print(type(set2)) # <class 'set'>
```

## 3.操作

```
# 获取集合的长度 len()
print(len(set2)) # 4

# 集合不能通过下标访问元素
# print(set2[3])
```

### 3.1添加

代码演示：

```
set2 = {12,345,633,21}
# 向集合中添加一个元素 add()
set2.add(98)
print(set2)

# 通过update() 向集合中添加多个元素 追加的元素以列表的形式出现
set2.update([1,2,3])
print(set2)
```

### 3.2删除

代码演示：

```
set2 = {12,345,633,21}
# pop() 随机删除一个
set2.pop()
print(set2)

# remove() 删除指定的元素,传入的参数是要删除的元素,如果删除的元素不存在,会报错
# set2.remove(22)
```

```
set2.remove(2)
print(set2)

# discard() 删除指定的元素,传入的参数是要删除的元素,如果删除的元素不存在,不会报错
set2.discard(21)
# set2.discard(22)
print(set2)

# clear() 清空集合
set2.clear()
#print(set2)
```

### 3.3遍历

代码演示:

```
set2 = {12,345,633,21}
for i in set2:
    print(i)
```

### 3.4交集和并集

代码演示:

```
# 集合之间的关系
set5 = {12,34,56,23,86}
set4 = {23,45,25,12,41}
print(set5 & set4) # 交集
print(set5 - set4) # 差集
print(set5 | set4) # 并集
print(set4 > set5) # set4是否包含set5
print(set4 < set5) # set5是否包含set4
```