tuple元组

创建元组

1. 使用`()`创建元组
2. 使用`tuple()`函数创建元组:偏向于将某个类型转换为元组
3. 元组 VS 列表：元组的元素不能修改，而列表的元素可以修改；元组使用`()`，列表使用`[]`
4. 元组连接(合并)/复制

In [1]:
```
# 使用`()`创建元组
tuple1 = (10,20,30)
display(tuple1)
#若元素只有一个元素，需要在元素后加逗号
tuple2 = (10)
tuple3 = (10,)
print(type(tuple2))
print(type(tuple3))
```

```
(10, 20, 30)
<class 'int'>
<class 'tuple'>
```

In [2]:
```
# 使用`tuple()`函数创建元组:偏向于将某个类型转换为元组
tuple2 = tuple({'a':11,'b':22,'c':33})
tuple3 = tuple((1,2,3))
display(tuple2)
display(tuple3)
```

```
('a', 'b', 'c')
(1, 2, 3)
```

In [3]:
```
# 元组连接
x = (1,2,3)
y = ('a','b','c')
print(x+y)
```

```
(1, 2, 3, 'a', 'b', 'c')
```

In [4]:
```
# 元组复制*
x = ('Hello',)
print(x*5)
```

```
('Hello', 'Hello', 'Hello', 'Hello', 'Hello')
```

In [5]:
```
# 元组值不可更改
tup = ('A', 'B', 'C')
print(id(tup))
# 查看内存地址
tup = (1, 2, 3)
print(id(tup))
```

```
4410797504
4410897984
```

> 从以上实例可以看出，重新赋值的元组 tup ，绑定到了新的对象了，不是修改了原来的对象。

若要更改元组值，可以先将元组转换为列表，更改列表值，然后再将其转换回元组。

In [6]:
```
fruit_tuple = ('apple', 'pear', 'cherry')
fruit_list = list(fruit_tuple)
fruit_list[2] = 'banana'
fruit_tuple = tuple(fruit_list)
print(fruit_tuple)
```

```
('apple', 'pear', 'banana')
```

访问元组

1. 下标索引访问

In [7]:
```
tuple_name = ('wzq', 'lgl', 'gz', 'whl', 'sj', 'hxw')
print(tuple_name[0])
print(tuple_name[-1])
```

```
wzq
hxw
```

2. 切片访问

使用切片访问元组的格式为 `tuple_name[strat : end : step]` ，其中， `start` 表示起始索引， `end` 表示结束索引， `step` 表示步长。

是左闭右开区间，即访问不了 `end` 代表的元素

In [8]:
```
tuple_name = ('wzq', 'lgl', 'gz', 'whl', 'sj', 'hxw')
print(tuple_name[1:5:2])
print(tuple_name[-6:-1:3])
```

```
('lgl', 'whl')
('wzq', 'whl')
```

3. `for` 循环遍历元组

In [9]:
```
fruit_tuple = ('apple', 'pear', 'cherry')
for i in fruit_tuple:
    print(i)
```

```
apple
pear
cherry
```

4. 检查元素是否存在

In [10]:
```
# 检查元组中是否存在'apple'
fruit_tuple = ('apple', 'pear', 'cherry')
```

```
print('apple' in fruit_tuple)
```

```
True
```

内置函数

1. `print()`
2. `len()`
3. `type()`
4. `tuple()`
5. `max()` `min()`
6. `del 元组名` # del函数删除整个元组

内置方法

1. `count()` # 返回元组中指定值出现的次数
2. `index()` # 在元组中搜索指定的值并返回它被找到的位置