第四章 - 元组

张建章

阿里巴巴商学院

杭州师范大学

2024-09

元组（Tuple）是 Python 中一种重要的数据类型，用于存储多个数据项。与列表类似，元组也是一种序列类型，但与列表的可变性不同，元组是**不可变的**，一旦创建，元组中的元素不能被修改。这使得元组特别适合用于表示那些在程序运行期间不应改变的数据集，如数据库记录、配置参数等。

元组通过一组**圆括号** `()` 来创建，内部的元素用**逗号**分隔。例如，以下代码创建了一个包含三个元素的元组：

```
my_tuple = (1, 2, 3)
print(my_tuple)  # 输出: (1, 2, 3)
```

需要注意的是，**逗号**是定义元组的核心部分，即使只有一个元素的元组也需要逗号来区分。例如：

```
single_element_tuple = (1,)  # 必须加逗号
print(single_element_tuple)  # 输出：(1,)
```

如果省略逗号，Python 会将其视为普通的整数或其他类型，而不是元组。

元组还可以通过**不使用括号**的方式定义，但在多项运算或其他复杂场景中，为了提高可读性，通常建议加上圆括号：

```
my_tuple = 1, 2, 3
print(my_tuple)  # 输出：(1, 2, 3)
```

`tuple()` 函数是一个用于创建元组的内置函数。使用 `tuple()` 可以将其他可迭代对象（如列表、字符串、字典等）转换为元组。

1. 创建空元组

```
t1 = tuple()
print(t1)  # 输出: ()
```

2. 从列表创建元组

```
t2 = tuple([1, 4, 6])
print(t2)  # 输出: (1, 4, 6)
```

3. 从字符串创建元组

```
t3 = tuple('Python')
print(t3)  # 输出: ('P', 'y', 't', 'h', 'o', 'n')
```

元组是一种不可变的序列类型，常用于存储多个有序的值。元组的基本操作**与列表类似**，但由于其不可变性，不能对元组的元素进行修改。

1. 访问元组的元素

元组中的元素可以通过**索引**访问，索引从 0 开始。例如：

```
tuple1 = ('a', 'b', 'c')
print(tuple1[0])  # 输出: 'a'
```

2. 元组的不可变性

元组是不可变的，无法修改其中的元素。例如，试图修改元组元素会引发错误：

```
tuple1 = (1, 2, 3)
# tuple1[0] = 10  # 这将引发 TypeError
```

3. 嵌套元组

元组可以包含其他元组或可变对象。虽然元组本身是不可变的，但其包含的可变对象如列表等，仍然可以修改：

```
nested_tuple = (1, 2, [3, 4])
nested_tuple[2][0] = 'modified'
print(nested_tuple)  # 输出: (1, 2, ['modified', 4])
```

4. 元组的切片操作

元组支持**切片**操作，可以获取元组中的子集：

```
tuple1 = (1, 2, 3, 4, 5)
print(tuple1[1:3])  # 输出: (2, 3)
```

5. 元组的长度

```
tuple1 = (1, 2, 3)
print(len(tuple1))  # 输出: 3
```

6. 元组的遍历

可以使用 **for** 循环遍历元组中的元素：

```
tuple1 = (1, 2, 3)
for item in tuple1:
    print(item)
```

7. 检查元素是否存在于元组

```
tuple1 = (1, 2, 3)
print(2 in tuple1)  # 输出: True
```

元组虽然是不可变的，但提供了两种常用的内置方法：`count()` 和 `index()`，这些方法在处理元组数据时非常有用，特别是在分析和操作不需要修改的数据时。

1. `count()` 方法

`count()` 用于统计指定元素在元组中出现的次数。

```
# 创建包含重复元素的元组
vowels = ('a', 'e', 'i', 'o', 'i', 'u')

# 统计'i'出现的次数
count_i = vowels.count('i')
print(count_i)  # 输出: 2
```

2. `index()` 方法

`index()` 用于查找指定元素在元组中的索引位置，并返回第一个匹配项的索引。如果元素不存在，则会抛出 `ValueError`。其语法为：

```
tuple.index(element, start, end)
```

```python
# 创建一个元组
vowels = ('a', 'e', 'i', 'o', 'i', 'u')

# 查找'i'的索引
index_i = vowels.index('i')
print(index_i)  # 输出: 2
```

在该示例中，`index()` 返回元组中第一个'i' 的索引值

元组是不可变的序列类型之一，其支持许多适用于所有序列类型的操作。这些操作包括索引访问、切片、连接、重复、成员测试等，能够对序列进行常见的操作和查询，以下结合代码示例进行说明。

1. 索引访问 (Indexing)

元组的元素可以通过索引进行访问，索引从 0 开始。如果索引为负数，则表示从序列末尾开始计数。

```
my_tuple = (10, 20, 30, 40)
print(my_tuple[0])  # 输出: 10
print(my_tuple[-1])  # 输出: 40
```

2. 切片 (Slicing)

切片允许从序列中获取一个子序列，其格式为 $[start : end : step]$，其中 $start$ 是起始索引，end 是结束索引（不包括），$step$ 是步长（默认为 1）。

```
numbers = (0, 1, 2, 3, 4, 5)
subset = numbers[1:4]  # 输出: (1, 2, 3)
reversed_tuple = numbers[::-1]  # 输出: (5, 4, 3, 2, 1, 0)
```

切片操作返回一个新的元组，而不修改原始元组。

3. 连接 (Concatenation)

可以使用加号（+）将两个元组合并成一个新的元组。

```
tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)
concatenated_tuple = tuple1 + tuple2  # 输出: (1, 2, 3, 4, 5, 6)
```

4. 重复 (Repetition)

使用乘法符号（*）可以将元组重复多次生成一个新元组。

```
original_tuple = (10,)
repeated_tuple = original_tuple * 3  # 输出: (10, 10, 10)
```

5. 成员测试 (Membership Testing)

使用 `in` 运算符可以检查一个值是否存在于元组中。

```
my_tuple = ('apple', 'banana', 'cherry')
print('banana' in my_tuple)  # 输出: True
```

6. 打包与解包 (Packing and Unpacking)

打包（packing）和解包（unpacking）是处理序列（如列表和元组）的重要语法特性。打包是将多个值组合为一个序列，而解包则是将序列中的值提取到单独的变量中。

打包操作通过将多个值用逗号分隔在一起，可以创建一个元组。

```
# 打包
values = 1, 2, 3
print(values)  # 输出: (1, 2, 3)
```

解包操作使用赋值语句，将序列中的元素赋值给多个变量，变量的数量必须与序列中的元素数量相匹配。

```
a, b, c = (1, 2, 3)
print(a)  # 输出: 1
print(b)  # 输出: 2
print(c)  # 输出: 3
```

元组等序列可以被解包成多个变量。这种操作允许快速赋值，并且可以结合星号（*）将剩余的值赋给一个列表。

```
values = (1, 2, 3, 4)
a, b, *c = values  # a = 1, b = 2, c = [3, 4]
data = (1, 2, 3)
a, _, c = data # 这里使用了 _ 作为占位符来忽略中间的值
```

这种操作在处理多个返回值或动态参数传递时非常实用。

上述六种通用操作适用于所有 Python 中的序列类型，包括列表和字符串，而元组的不可变特性使其在某些情况下更具优势，例如用作函数的返回值或键值对。

THE END