# 数据结构

IlleniumDillon

2024 年 6 月 2 日

## 1 基本数据类型

### 1.1 数字

- 整数　如：1, 2, 3, 4, 5　int
- 浮点数　如：1.0, 2.0, 3.0, 4.0, 5.0　float
- 复数　如：1+2j, 2+3j, 3+4j, 4+5j, 5+6j　complex

数字类型的操作符有：+，-，*，/，//，%，**等。

表 1: 数字类型的操作符

| 操作符 | 描述 | 结果的类型 |
|---|---|---|
| + | 加法 | 操作数全int，结果为int;有float，结果为float |
| - | 减法 | 操作数全int，结果为int;有float，结果为float |
| * | 乘法 | 操作数全int，结果为int;有float，结果为float |
| / | 除法 | 一定是float |
| // | 整除 | 操作数全int，结果为int;有float，结果为float |
| % | 取余 | 操作数全int，结果为int;有float，结果为float |
| ** | 幂 | 操作数全int，结果为int;有float，结果为float |

## 1.2 字符串

### 1.2.1 字符串表示

字符串*str*是一种序列类型，可以通过索引访问其中的元素。字符串是不可变的，不能修改其中的元素。它们可以用单引号或双引号括起来表示，也可以用三个单引号或三个双引号括起来表示多行字符串。

要标示引号本身，我们需要对它进行“转义”，即在前面加一个 或者使用不同类型的引号。

```
s = 'I\'m a student.'
s = "I'm a student."
s = '''I'm a student.'''
s = """I'm a student."""
```

除了引号之外，还有一些特殊字符称为转义字符，如： \$n$表示换行，\$t$表示制表符。

表 2: 转义字符

| 转义字符 | 含义 |
|---|---|
| \$n$ | 换行 |
| \$t$ | 制表符 |
| \$r$ | 回车 |
| \$b$ | 退格 |
| \$f$ | 换页 |
| \$v$ | 垂直制表符 |
| \\ | 反斜杠 |
| \' | 单引号 |
| \” | 双引号 |

如果不希望前置\的字符被转义，可以在字符串前加一个r或R，称为原始字符串，不会对字符串中的特殊字符进行转义。

原始字符串还有一个微妙的限制：一个原始字符串不能以奇数个 \ 字符结束。

```
s = r'C:\some\name'
```

### 1.2.2 字符串运算

字符串支持+和*运算，+表示连接，*表示重复，相邻的字符串字面值会自动连接。注意：自动连接只能用于两个字面值，不能用于变量或表达式。

```
s = 'Hello' + 'World' # HelloWorld
s = 'Hello' * 3       # HelloHelloHello
s = 'Hello' 'World'   # HelloWorld
```

字符串还支持索引和切片，索引从0开始，切片的范围是左闭右开区间。索引还支持负数，-1表示最后一个元素，-2表示倒数第二个元素，以此类推。

```
s = 'HelloWorld'
s[0]    # H
s[-1]   # d
s[1:5]  # ello
s[5:]   # World
s[:5]   # Hello
s[:]    # HelloWorld
s[1:5:2]# el
```

在使用索引时，如果索引超出范围，会引发IndexError异常；在使用切片时，如果切片超出范围，不会引发异常，而是返回一个空字符串。

字符串还支持in和not in运算符，用于判断一个字符串是否包含另一个字符串。

字符串还支持len()函数，用于返回字符串的长度。

更多字符串方法请参考Python官方文档。

```
s = 'HelloWorld'
'Hello' in s     # True
'Hello' not in s# False
```

### 1.2.3 字符串格式化

字符串格式化是将一个字符串中的占位符替换为其他值的过程。Python提供了多种字符串格式化的方法，如：

```
name = 'Alice'
age = 18
s = 'My name is %s, I am %d years old.' % (name, age)
s = 'My name is {}, I am {} years old.'.format(name, age)
s = f'My name is {name}, I am {age} years old.'
```

其中，%s表示字符串，%d表示整数，%f表示浮点数，%x表示十六进制整数。 format()方法中的表示占位符，可以使用位置参数或关键字参数。f-string是Python3.6引入的一种新的字符串格式化方法，使用f前缀，中的表达式会被计算。更多字符串格式化方法请参考Python官方文档。

## 1.3 列表

Python 支持多种复合数据类型，可将不同值组合在一起。最常用的列表，是用方括号标注，逗号分隔的一组值。列表可以包含不同类型的元素，但一般情况下，各个元素的类型相同。

```
a = [1, 2, 3, 4, 5]
b = ['a', 'b', 'c', 'd', 'e']
c = [1, 'a', 2, 'b', 3, 'c']
```

列表支持索引和切片，索引从0开始，切片的范围是左闭右开区间。列表还支持+和*运算，+表示连接，*表示重复。

```
a = [1, 2, 3, 4, 5]
a[0]    # 1
a[-1]   # 5
a[1:3]  # [2, 3]
a[3:]   # [4, 5]
a[:3]   # [1, 2, 3]
a[:]    # [1, 2, 3, 4, 5]
a[1:5:2]# [2, 4]
a + [6, 7, 8]   # [1, 2, 3, 4, 5, 6, 7, 8]
a * 3           # [1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

列表还支持in和not in运算符，用于判断一个元素是否在列表中。列表还支持len()函数，用于返回列表的长度。

与字符串不同，列表是可变的，可以修改其中的元素，也可以添加或删除元素。通过索引或切片可以修改元素的值，通过append()方法可以在列表末尾添加元素， Python 中的简单赋值绝不会复制数据。当你将一个列表赋值给一个变量时，该变量将引用现有的列表。通过一个变量对列表所做的任何更改都会被引用它的所有其他变量看到。如果你想复制列表或者其中的一部分，请使用切片操作符来制作副本。

# 2 列表详述

## 2.1 列表方法

列表数据类型支持很多方法，列表对象的所有方法所示如下：

**list.append(x)** 在列表的末尾添加一个元素。相当于a[len(a):] = [x]。

**list.extend(iterable)** 将可迭代对象的元素添加到列表的末尾。相当于a[len(a):] = iterable。

**list.insert(i, x)** 在指定位置插入一个元素。第一个参数是要插入的元素的索引，所以a.insert(0, x) 插入列表头部，a.insert(len(a), x) 等同于a.append(x)。

**list.remove(x)** 删除列表中第一个值为 x 的元素。如果没有这样的元素，触发 ValueError 异常。

**list.pop([i])** 删除列表中给定位置的元素并返回它。如果没有给定位置，a.pop() 将会删除并返回列表中的最后一个元素。如果索引超出范围，会引发IndexError异常。

**list.clear()** 移除列表中的所有元素。相当于 del a[:].

**list.index(x[, start[, end]])** 返回列表中第一个值为 x 的元素的索引。如果没有这样的元素将会触发 ValueError 异常。可选参数 start 和 end 是切片符号，用于将搜索限制为列表的特定子序列。返回的索引是相对于整个序列的开始计算的，而不是 start 参数。

**list.count(x)** 返回 x 在列表中出现的次数。

**list.sort(key=None, reverse=False)** 对列表中的元素进行排序, key 是一个只有一个参数的函数，用来为每个元素提取比较键(reverse=True 降序，reverse=False 升序)。列表中的元素必须是可比较的。否则会引发 TypeError 异常。

**list.reverse()** 反转列表中的元素。

**list.copy()** 返回列表的一个浅拷贝。相当于 a[:]。

## 2.2 列表推导

列表推导是一种从其他列表创建新列表的方法，可以使用一个表达式来过滤和转换列表中的元素。列表推导的语法如下：

```
a = [1, 2, 3, 4, 5]
b = [x * x for x in a]  # [1, 4, 9, 16, 25]
c = [x for x in a if x % 2 == 0] # [2, 4]
```

列表推导还支持嵌套，可以使用多个for子句。

```
a = [1, 2, 3, 4, 5]
b = [(x, y) for x in a for y in a]
'''
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5),
 (2, 1), (2, 2), (2, 3), (2, 4), (2, 5),
 (3, 1), (3, 2), (3, 3), (3, 4), (3, 5),
 (4, 1), (4, 2), (4, 3), (4, 4), (4, 5),
 (5, 1), (5, 2), (5, 3), (5, 4), (5, 5)]
'''
c = [(x, y) for x in a for y in a if x != y]
'''
[(1, 2), (1, 3), (1, 4), (1, 5),
 (2, 1), (2, 3), (2, 4), (2, 5),
 (3, 1), (3, 2), (3, 4), (3, 5),
 (4, 1), (4, 2), (4, 3), (4, 5),
 (5, 1), (5, 2), (5, 3), (5, 4)]
'''
```

列表推导式可以使用复杂的表达式和嵌套函数：

```
[str(round(pi, i)) for i in range(1, 6)]
# ['3.1', '3.14', '3.142', '3.1416', '3.14159']
```

列表推导式中的初始表达式可以是任何表达式，甚至可以是另一个列表推导式。

```
matrix = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [9, 10, 11, 12]
]
transposed = [[row[i] for row in matrix] for i in range(4)]
'''
[[1, 5, 9],
 [2, 6, 10],
 [3, 7, 11],
 [4, 8, 12]]
'''
#等价于
transposed = []
for i in range(4):
    transposed.append([row[i] for row in matrix])
```

### 2.3 del语句

有一种方式可以按索引而不是值从列表中移除条目: del 语句。这与返回一个值的 pop() 方法不同。 del 语句也可用于从列表中移除切片或清空整个列表。

## 3 元组

元组是由逗号分隔的一组值，通常用圆括号括起来表示。元组是不可变的，不能修改其中的元素。元组可以包含不同类型的元素，但一般情况下，各个元素的类型相同。

元组由多个用逗号隔开的值组成,例如：

构造 0 个或 1 个元素的元组比较特殊，需要使用额外的逗号来消除歧义。

```
a = (1, 2, 3, 4, 5)
b = ('a', 'b', 'c', 'd', 'e')
c = a, b # ((1, 2, 3, 4, 5), ('a', 'b', 'c', 'd', 'e'))
```

```
a = () # 空元组
b = (1,) # 一个元素的元组
c = 1, # 一个元素的元组
```

# 4 集合

集合是一种无序且不重复的元素集合，是可变数据类型。集合对象支持数学上的集合操作，如并集、交集、差集和对称差集。

集合是由花括号括起来的一组值，逗号分隔，或者使用set()函数创建。

```
a = {1, 2, 3, 4, 5}
b = set([1, 2, 3, 4, 5])
c = set() # 空集合
# 集合运算
a = {1, 2, 3, 4, 5}
b = {4, 5, 6, 7, 8}
a | b # {1, 2, 3, 4, 5, 6, 7, 8} 并集
a & b # {4, 5} 交集
a - b # {1, 2, 3} 差集
a ^ b # {1, 2, 3, 6, 7, 8} 对称差集
```

空集合必须使用set()函数创建，不能使用创建，因为创建的是空字典。

集合支持in和not in运算符，用于判断一个元素是否在集合中。

集合支持推导式，可以使用一个表达式来过滤和转换集合中的元素。

# 5 字典

不同于以固定范围的数字进行索引的序列，字典是以键进行索引的，键可以是任何不可变类型；字符串和数字总是可以作为键。如果一个元组只包含字符串、数字或元组则也可以作为键；如果一个元组直接或间接地包含了任何可变对象，则不能作为键。列表不能作为键，因为列表可以使用索引赋值、切片赋值或者 append() 和 extend() 等方法进行原地修改列表。

字典是由花括号括起来的一组键值对，逗号分隔，或者使用dict()函数创建。字典的主要用途是通过关键字存储、提取值。用 del 可以删除键值对。用已存在的关键字存储值，与该关键字关联的旧值会被取代。通过不存在的键提取值，则会报错。

对字典执行 list(d) 操作，返回该字典中所有键的列表，按插入次序排列（如需排序，请使用 sorted(d)）。检查字典里是否存在某个键，使用关键字 in。

字典推导可以用来创建任意键和值的表达式字典，如下所示：

```
a = {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
b = dict([(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd'), (5, 'e')])
c = dict(a=1, b=2, c=3, d=4, e=5)
d = {x: x**2 for x in (1, 2, 3, 4, 5)}
# {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```