第4章 Python的组合数据类型

本章教学目标：

- 理解序列型、映射型组合数据的概念和特点。
- 掌握对序列型、映射型组合数据操作的相关方法。
- 了解集合型数据的概念、特点和对集合操作的相关方法。

4.1 序列型组合数据

- 序列型组合数据是在一维空间上的元素向量，元素之间有顺序关系，元素之间不排他
- 可以通过索引序号访问元素
- 序列型组合数据包括字符串、列表和元组

4.1 序列型组合数据

操　作	意　义
s1+s2	将s1和s2连接成一个序列
s * n 或 n * s	将序列s复制n次并连接成一个序列
s[i]	序列s的第i个元素（从0起算）
s [start : stop : step]	序列s的第start个元素（含）起至第stop个元素（不含），间隔为step的子序列。若start为0可省略，step为1可省略。
len(s)	返回序列s的元素数（长度）
min(s) 和 max(s)	返回序列s中最小和最大的元素项
s.index(x) 和 s.count(x)	元素x在序列s中第一次出现的位置和出现的次数
x in s 和 x not in s	判断x是否是序列s的元素，返回True或False

4.1.1 字符串

- 字符串**索引**

利用方括号运算符[ ]可以通过索引值得到相应位置（下标）的字符

- 从前往后的正向索引，*n*个字符的字符串，其索引值从0至*n*-1
- 从后向前的负数索引，n个字符的字符串，其索引值从-1至-n

```
>>> s='Python'
>>> print(s[0],s[5])  # 注意不能越界
P n
>>> print(s[-1],s[-6])
n P
>>> s[6]  # 下标越界了
Traceback (most recent call last):
  File "<pyshell#158>", line 1, in
<module>
    s[6]
IndexError: string index out of range
```

4.1.1 字符串

- 字符串**切片（slice）**

  从字符串中提取子串***[start:end:step]***

  - 第一个数字表示切片开始位置（默认为0）
  - 第二个数字表示切片截止位置（但不包含这个位置，默认为字符串长度）***数字前面切一刀***
  - 第三个数字表示切片的步长（默认为1），当步长省略时，可以顺便省略最后一个冒号

```
>>> a = 'Python'
>>> a[1:4]
'yth'
>>> a
'Python'
>>> a[:4]
'Pyth'
>>> a[1:]
'ython'
```

```
>>> a[::]
'Python'
>>> a[::2]
'Pto'
>>> a[::-1]
'nohtyP'
>>> a[:100]
'Python'
>>> a[100:]
''
```

4.1.1 字符串

- 【例4-1】 格式化输出字符串示例。

```
>>> test =5000
>>> print("%6d" % test)          5000
>>> print("%2d" % test)          5000
>>> print("%-6d" % test)         5000
>>> print("%+6d" % test)         +5000
>>> print("%06d" % test)         005000
>>> print("%#o" % test)          0o11610
>>> print("%#x" % test)          0x1388

>>> test=128.3656
>>> print("%6.2f" % test)        128.37
>>> print("%3.1f" % test)        128.4
>>> print("%.3e" % test)         1.284e+02

>>> test="上海是一个美丽的城市"
>>> print("%5.2s" % test)          上海
```

4.1.1 字符串

- 用format()方法进行字符串格式化

{ <序号>：<占位符> <对齐符> <总长度> <千位分隔> <截断位数> <数字类型> }

  - <序号>为接收参数的位置顺序，缺省为自然先后顺序，后面冒号通常不要省略；
  - <占位符>是用于填满整个字符串长度的单个字符；
  - <对齐符>是参数文本在整个字符串中的对齐方式，“^”表示居中对齐，“<”为左对齐，“>”为右对齐；
  - <总长度>是生成字符串的总字符数；
  - <千位分隔>用于整数或浮点数每隔三位数字的分隔字符；
  - <截断位数>若用于浮点数则为小数部分的位数，若用于字符串则为最大输出长度；
  - <数字类型>包括“b”、“c”、“d”、“o”、“x”、“X”表示整数输出进制或编码字符串，“f”、“e”、“E”、“%”表示浮点数及其指数形式或百分比形式。

4.1.1 字符串

- format()方法举例

```
>>> '{:>8}'.format('123') #总长度8字符，右对齐
'     123'

>>> '{:*^10}'.format('123')#总长度10字符，居中对齐，星号填充
'***123****'

>>> '{:_^24,}'.format(12345.67890)#居中，下画线填充，千分分隔
'_______12,345.6789________'

>>> '{:.3f}'.format(1.23456789)    #保留3位小数
'1.235'

>>> '{:.3}'.format('甲乙丙丁戊己庚辛')    #截断输出3字符
'甲乙丙'
```

4.1.1 字符串

- format()方法举例(续)

```
>>> '{:X}'.format(1234)  #字符串输出大写十六进制
'4D2'

>>> '{:e}'.format(0.0000001234)
'1.234000e-07'

>>> '{:%}'.format(0.12345)
'12.345000%`
```

格式限定表达式也支持按序号接收参数

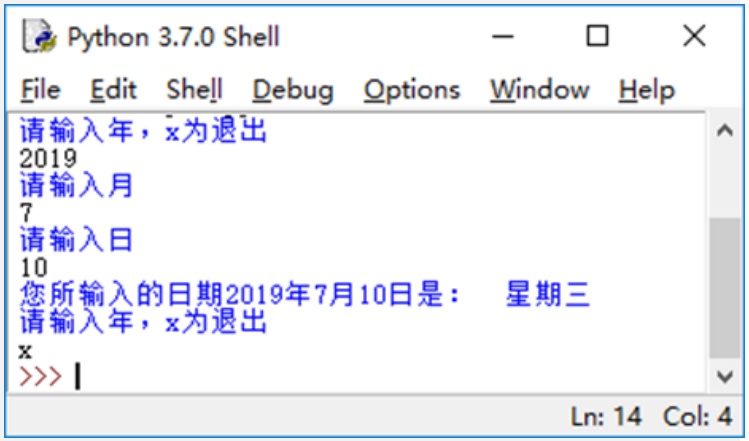
```
>>> '{0:{1}{3}{2}}'.format('甲乙丙丁','-',30,'^')
'------------甲乙丙丁------------'
```

4.1.1 字符串

- 【例4-2】 字符串综合举例。利用Python内置库calendar，待输入年、月、日后打印输出该日的星期。

```
import calendar
s='星期一星期二星期三星期四星期五星期六星期日'
while True:
    y=input('请输入年，x为退出\n')

    if y in ('x','X'):
        break
    else:
        m=input('请输入月\n')
        d=input('请输入日\n')
        i=calendar.weekday(int(y),int(m),int(d))
        print('您所输入的日期{0}年{1}月{2}日是：{3:>5}'
              .format(y,m,d,s[i*3:i*3+3]))
```



4.1.1 字符串

- 输入四个字符串，求这些字符串的最大长度。

```
length=0
for i in range(4):
    a=len(input())
    if a>length:
      length=a
print(length)
```