第3章

数据类型

早期计算机主要用来处理各种数值计算问题。但是，计算机能处理的信息远不止数值，还可以处理文本、图形、音频、视频等各种数据。为了解决这些数据之间的转换、存储和计算，需要将不同数据定义为不同的数据类型。

3.1 数据类型——主要类型和特征

1. Python 的基本数据类型

Python 中的数据类型根据运算方法大体分为五种，一是数值类，如整数、浮点数、复数；二是字符串类，如字母、符号、汉字等；三是布尔类；四是混合类，包括数值和字符串，如列表、元组、集合、布尔值；五是字节类，主要用于二进制字节文件。Python 主要数据类型如表 3-1 所示。

表 3-1 Python 主要数据类型

数据类型	名称	说　明	示　例
int	整数	精度无限制，整数有效位可达数万位	0、50、－56789 等
float	浮点数	精度无限制，默认 16 位，精度可扩展	3.1415927、5.0 等
complex	复数	注意，复数的虚数部分用“j”表示	3＋2.7j
str	字符	由字符组成的不可修改元素，无长度限制	'hello'、'提示信息'等
list	列表	多种类型的可修改元素，最多 5.3 亿个元素	[4.0，'名称'，True]
tuple	元组	多种类型的不可修改元素	(4.0，'名称'，True)
dict	字典	由“键值对”(用:分隔)组成的有序元素	{'姓名':'张飞'，'年龄':30}
set	集合	无序且不重复的元素集合	{4.0，'名称'，True}
bool	布尔值	逻辑运算的值为 True(真)或 False(假)	a＞b and b＞c
bytes	字节码	由二进制字节组成的不可修改元素	b'\xe5\xa5\xbd'

2. 数据类型定义

Python 中变量的数据类型不需要在程序头部预先定义，变量赋值就是变量定义过程。如果变量没有赋值，则 Python 认为该变量不存在。

【例 3-1】　不同数据类型混用将造成程序运行错误，程序如下。

```
>>> x = 123                                  # 变量 x 赋值为数字
>>> y = '456'                                # 变量 y 赋值为字符串
>>> x + y                                    # 不同数据类型混合运算
    Traceback (most recent call last):…      # 抛出异常信息(输出略)
>>> x + z                                    # 变量 z 没有赋值,Python 认为该变量不存在
    Traceback (most recent call last):…      # 抛出异常信息(输出略)
```

3. 课程扩展：数据类型的特点

Python 主要数据类型的特点如表 3-2 所示。

表 3-2　Python 主要数据类型的特点

数据操作	字符串	列表	元组	字典
数据定义	s1＝'开卷有益'	c1＝[1,2,3,4,5]	t1＝(1,2,3,4,5)	d1＝{'a':1, 'b':2, 'c':3}
数据类型混用	不可	可以	可以	可以
元素索引号	有	有	有	无
访问单个元素	s2＝s1[2]	c2＝c1[2]	t2＝t1[2]	d2＝d1['a']
访问多个元素	s3＝s1[0:2]	c3＝c1[0:2]	t3＝t1[0:2]	不可(需要循环语句)
修改元素	不可	c1[0]＝8	不可	d1['c']＝8
添加元素	不可	c1.insert(0, 'ID')	不可	d1['d']＝10
删除元素	不可	c1.remove('ID')	不可	del d1['c']

4. 课程扩展：实际问题的数据分类

程序设计中会遇到很多实际问题，它们无法用程序语言定义的数据类型进行表示，如学生的性别、成绩的优良中差、地区的东南西北等。这些数据不便于程序处理，我们需要对这些数据进行合适的分类，根据不同的数据类型选用相应的处理方法。

(1) 定类数据。定类数据是指按照事物的某种属性对数据进行分类或分组。定类数据只能表示数据类别的差异，不能比较各类数据之间的大小，数据之间没有顺序或等级关系。定类数据只能计算数据频率，不能进行数据大小比较。例如，在统计男生和女生占总人数的比例时，可以将男生编码为 1，女生编码为 2，这里的 1、2 只表示数据类别不同，没有次序关系，这样就可以在程序中计算男生或女生的比例。其他应用案例如地区类型分为东部、中部、西部等。定类数据的应用参见案例 9，程序中用数字 1 表示选择“认罪”，数字 2 表示选择“不认罪”；也可以参见案例 33，程序中用数字 1、2 表示平声，用数字 3、4 表示仄声。

(2) 定序数据。定序是对事物之间等级或顺序差别的一种测度，用数字表示个体在某个有序状态中所处的位置。定序数据可以比较优劣或排序。定序数据不仅含有类别信息，还包含了次序信息。定序数据只能排序和比较，不能进行算术运算。例如，调查者对某事物的态度有同意、不同意、不发表意见等，可以通过程序计算同意的人数和比例，也可以对这些数据进行排序；学历数据可以分为小学、初中、高中、本科、研究生等；企业产品的销售额可以分为盈利、持平、亏损等。定序数据的应用参见案例 5。

3.2 数值——整数和浮点数的运算

1. 整数

Python 中的整数包括正整数和负整数，Python 支持大整数运算。32 位 Python 版本默认最大整数为 $2^{31}-1=2\,147\,483\,647$；64 位 Python 版本为 $2^{63}-1=9\,223\,372\,036\,854\,775\,807$。当整数值域超出这个范围时，Python 会自动转换为大整数计算（任意精度），整数有效位可达数万位（受操作系统内存管理的限制）。

【例 3-2】 Python 默认最大整数如下所示，超过默认值时，Python 会自动扩大位数。

```
>>> import sys                          # 导入标准模块
>>> sys.maxsize                         # 查看最大整数默认值
9223372036854775807
>>> 123 ** 100                          # 自动转为大整数运算（** 为指数运算）
9783880597…                             # 输出一个 209 位有效数字的大整数（输出略）
```

2. 浮点数的概念

浮点数就是带小数点的实数。Python 中，浮点数默认精度为小数点后 16 位，采用浮点数精度设置函数时，精度仅受内存大小限制。整数与浮点数的存储方式和计算方式都不同，浮点数采用 IEEE 754 标准规定方法存储。在 CPU（中央处理单元）中，整数由算术逻辑单元（ALU）执行运算，浮点数由浮点处理单元（FPU）执行运算。

3. 科学记数法

科学记数法以紧凑格式表示非常大或者非常小的数字。科学记数法用字母 e 或者 E 作为幂的符号，以 10 为基数（底数），后面紧跟的数表示多少次幂（可正可负）。

【例 3-3】 数值 $6\,100\,000=6.1\times10^{6}$，科学记数法为 6.1e+06（e 后面最少两位）。

【例 3-4】 数值 $0.000\,006=6.0\times10^{-6}$，科学记数法为 6e-06。

【例 3-5】 Python 默认最大和最小浮点数以及浮点数除法运算的程序如下：

```
>>> import sys                          # 导入标准模块
>>> sys.float_info.min                  # 系统定义的初始最大浮点数有效位
2.2250738585072014e-308                 # 说明：2.2e-308 表示 2.2×10^-308
>>> 25/7
3.571485714285716                       # 除法运算时，Python 自动将运算结果转为浮点数
>>> 0.1 + 0.2                           # 数值二-十进制转换时会产生截断误差
0.30000000000000004                     # 浮点数的精度为小数点后 16 位
>>> a = 6.1e+06                         # 用科学记数法给变量赋值
>>> a
6100000.0
```

3.3 字符串——最常用的数据类型

1. 字符串定义

字符串是以单引号、双引号或三引号引起来的符号，字符串中的符号称为元素，元素可

以是数字、字母、汉字等。Python 对字符串长度没有限制。字符串定义方法如下。

(1) 字符串必须用引号引起来,单引号(见例 3-6)、双引号(见例 3-7)、三引号(见例 3-7)都行,但是引号必须成对使用。引号是一种分隔符,不是字符串的一部分。

(2) 字符串内部有单引号时,外部必须用双引号,否则将导致语句错误。

(3) 语句中有特殊字符(如%)时,可在字符串前加 r,使字符串保持原样。

【例 3-6】 字符串定义示例程序如下:

```
>>> s1 = '与谁同坐,明月清风我。'          # 定义字符串(两个单引号为字符串定义符)
>>> s1                                    # 打印字符串
'与谁同坐,明月清风我。'
>>> s2 = '与谁同坐,'  '明月清风我。'      # 字符串用空格分隔时,属于同一个字符串
>>> s2                                    # 打印字符串
'与谁同坐,明月清风我。'
>>> s3 = '与谁同坐,', '明月清风我。'      # 字符串用逗号分隔时,自动转换为元组
>>> s3                                    # 打印元组
('与谁同坐,', '明月清风我。')
```

【例 3-7】 长字符串的定义,程序如下:

```
>>> URL = ['https://mp.csdn.net/mp_blog/'                    # 变量名 URL 为网址
...     'creation/editor/126873850']                          # 引号前空格不影响正确性
>>> URL                                                       # 打印网址
['https://mp.csdn.net/mp_blog/creation/editor/126873850']
>>> poetry = '''                                             # 用三引号定义字符串
... [宋] 苏轼《花影》                                        # 变量名 poetry 为诗歌
... 重重叠叠上瑶台,
... 几度呼童扫不开,
... 刚被太阳收拾去,
... 却教明月送将来。
... '''
>>> path = r"path = / % '相对路径'../n"                      # 引号外的 r 表示引号内
>>> path                                                     # 的字符串按原样输出
"path = / % '相对路径'../n"
```

2. 字符串索引号

字符串有两种索引方式,一是正向索引号(或称为下标),从 0 开始,从左往右读取;二是起始索引号从−1 开始,从右往左逆序读取。

【例 3-8】 字符串 s = '与谁同坐,明月清风我',索引号如图 3-1 所示。

正向索引号:	0	1	2	3	4	5	6	7	8	9
字符串元素:	与	谁	同	坐	,	明	月	清	风	我
反向索引号:	−10	−9	−8	−7	−6	−5	−4	−3	−2	−1

图 3-1　字符串中元素的索引号

3. 字符串切片方法

字符串切片就是读取字符串中的某些字符,字符串切片语法如下:

```
变量名[起始索引号:终止索引号:步长]
```

(1) 括号[ ]里的数字为索引号,没有冒号时表示读取单个元素。

(2) 正向切片时,起始索引号为0;负向切片时,起始索引号为-1。

(3) 正向切片时,终止索引号为右边最后一个字符的索引号。

(4) 步长表示切片时每次跳过几个元素,没有步长参数时,表示逐个读取元素。如s[0:9:2]表示在索引号0~9的元素中,读取索引号为0、2、4、6、8的元素。正数步长表示从左往右切片,负数步长表示从右到左切片。

【例3-9】 字符串正向切片程序如下:

```
>>> s = '与谁同坐,明月清风我'
>>> s[:]            # 切片所有元素
'与谁同坐,明月清风我'
>>> s[0:4]          # 切片前面4个元素
'与谁同坐'
>>> s[4:]           # 切片索引号4后的元素
',明月清风我'
>>> s[::2]          # 切片步长为2的元素
'与同,月风'
>>> s[4:2]          # 起点大于终点,越界错误
''                  # 输出为空
```

【例3-10】 字符串反向切片程序如下:

```
>>> s ='与谁同坐,明月清风我'
>>> s[-3:]          # 切片索引号-3起始的元素
'清风我'
>>> s[:-3]          # 切片起始到-3的元素
'与谁同坐,明月'
>>> s[-5:-3]        # 切片索引号-5到-3的元素
'明月'
>>> s[::-2]         # 切片步长为-2的元素
'我清明坐谁'
>>> s[-4:-6]        # 起大于终,越界错误
''                  # 输出为空
```

字符串切片时,只是读取了字符串的一部分,并没有改变源字符串,因为字符串是不可改变的。但是可以将切片的字符赋值给一个新的变量,形成新字符串。

4. 字符串切片函数:split()

函数split()可以对字符串切片,并以列表的形式返回,函数语法如下:

```
split('切分符', 切分次数)[n]
```

(1) 参数"'切分符'"为空格、逗号、换行符等。不带参数时默认以空格进行切分。

(2) 参数"切分次数"为可选参数,默认为-1,即对整个字符串进行切分。

(3) 参数"[n]"为切分第n个字符,正数从左向右切分,负数时从右向左切分。

(4) 返回值:函数返回切分后的字符串列表(注意:切分不会改变源字符串)。

【例 3-11】　对字符串“百度 www.baidu.com”进行各种切分，程序如下：

```
>>> s = '百度 www.baidu.com'                    # 定义字符串变量
>>> s.split(' ')                                # 空格切分符(默认),对字符串按空格切分
['百度', 'www.baidu.com']                       # 切分为两个元素
>>> s.split('.')                                # 点切分符,对字符串按点切分
['百度 www', 'baidu', 'com']                    # 切分为 3 个元素,'百度 www'为同一个元素
>>> s.split('.', 3)[1]                          # 点切分符,切分 3 次,[1]为切片左起第 2 个元素
'baidu'
>>> s.split('.')[-2]                            # 点切分符,全部切分,[-2]为切片右起第 2 个元素
'baidu'
>>> s.split()[1]                                # 空切分符,全部切分,[1]为切片左起第 2 个元素
'www.baidu.com'
```

5. 字符串连接函数：join()

函数 join()用于两个字符串的连接，它是函数 split()的逆方法，语法如下：

```
1  newstr ='分隔符'.join('字符串')
```

(1) 参数“'分隔符'”为字符串合并时，用于分隔的符号，如空格、点号、冒号等。

(2) 参数“序列”为合并操作的源字符串，如字符串、列表、元组、字典等。

(3) 返回值 newstr 为连接后带分隔符的新字符串。

【例 3-12】　对字符串进行连接(生成新字符串)，程序如下：

```
>>> s1 = '青山相待,'                            # 定义字符串
>>> s2 = '白云相爱。'                           # 定义字符串
>>> s1 + s2                                     # 字符串连接
'青山相待,白云相爱。'
>>> path = 'D:\\test\\资源\\'                   # 定义字符串
>>> print(path + ''.join('琴诗.txt'))           # 合并字符串,分隔符''为两个单引号
D:\test\资源\琴诗.txt
>>> URL = ['www','baidu','com']                 # 定义字符串
>>> '.'.join(URL)                               # 用点号(.)进行字符串分隔
'www.baidu.com'
```

6. 删除首尾字符函数：strip()

【例 3-13】　s1＝'7. 删除首尾指定字符\n'，将字符串“7.”删除，程序如下：

```
>>> s1 = '7.删除首尾指定字符\n'                 # 为字符串变量 s1 赋值
>>> s2 = s1.strip('7. ')                        # 删除字符串 s1 中的“'7. '”
>>> s3 = s2.strip('\n')                         # 删除字符串尾部的换行符“'\n'”
```

说明：函数 strip()不能删除字符串中间的空格或字符。

7. 字符串长度计算函数：len()

【例 3-14】　字符串长度计算，程序如下：

```
>>> s = '故乡一望一心酸'                       # 为字符串变量 s 赋值
>>> len(s)                                      # 计算字符串长度
7
```

3.4 列表——功能强大的数据类型

列表是一种功能强大的数据类型，列表中元素的数据类型可以相同或不相同，元素可以是数字、字符串、元组、字典等，或者它们的混合体。32 位 Python 中列表元素最多为 $2^{29}=536\ 870\ 912$ 个，而且每个元素的大小没有限制。

1. 列表的索引号

列表中每个元素都有一个索引号，索引号在 C、Java 等语言中称为数组下标。元素可以通过索引号进行访问（读写）或遍历（顺序访问其中每个元素）。索引分为正向索引和反向索引。正向索引中，第 1 个元素索引号为 0，第 2 个元素索引号为 1，其余以此类推；反向索引中，最后 1 个元素索引号为－1，其余以此类推。

【例 3-15】 列表 lst=['枯藤', '老树', '昏鸦', '小桥流水人家']，索引号如图 3-2 所示。

正向索引号：	0	1	2	3
列表元素：	'枯藤'	'老树'	'昏鸦'	'小桥流水人家'
反向索引号：	−4	−3	−2	−1

图 3-2 列表中元素的索引号

2. 定义列表

列表中的元素用方括号[ ]括起来，元素之间以英文逗号分隔。列表元素可以是 Python 支持的任意数据类型，如数字、字符串、布尔值、列表、元组、字典等。

列表定义和输出语法如下：

```
1  列表名 = [元素 1, 元素 2, 元素 3, …,元素 n]          # 定义列表
2  列表名[索引号]                                      # 对已定义列表,按索引号输出列表
3  列表名[起始索引号:终止索引号]                       # 按起始和终止索引号输出列表
```

【例 3-16】 列表元素以逗号分隔，每个逗号之间为同一个元素，程序如下：

```
>>> lst1 = ['枯藤', '老树', '昏鸦', '小桥流水人家']      # 定义列表
>>> lst1                                                # 打印列表
['枯藤', '老树', '昏鸦', '小桥流水人家']
>>> lst2 = ['枯藤'  '老树'  '昏鸦', '小桥流水人家']     # 元素用空格分隔时属于同一元素
>>> lst2                                                # 打印列表
['枯藤老树昏鸦', '小桥流水人家']
>>> lst3 = ['枯藤', '老树', '昏鸦'], ['小桥流水人家']   # 列表嵌套定义时,将转换为元组
>>> lst3                                                # 打印元组
(['枯藤', '老树', '昏鸦'], ['小桥流水人家'])
>>> lst4 = ['圆周率', 3.14159, 520, True]               # 列表中元素的数据类型可以混用
>>> lst4                                                # 打印列表
['圆周率', 3.14159, 520, True]
```

3. 列表切片语法

列表切片是指按指定顺序读取列表中某些元素，得到一个新列表，语法如下：

```
1  列表名[切片起始索引号:切片终止索引号:步长]
```

列表切片通过冒号和数字的组合来访问列表中的元素。

(1) 列表名为要进行切片的源列表。

(2) 方括号[ ]里的数字为索引号,冒号为分隔符。第一个冒号前面的数字表示起始索引号(默认为 0);第一个冒号后面的数字表示终止索引号(默认为最后一个元素);第二个冒号后面的数字表示步长(默认为 1)。没有冒号则表示对列表中的单个元素进行访问。

(3) 正向切片时,起始索引号为 0;负向切片时,起始索引号为－1。

(4) 正向切片时,终止索引号为右边最后一个字符的索引号;负向切片时,终止索引号为左边最后一个字符的索引号。

(5) 步长表示切片时每次跳过几个元素,没有步长参数时,表示逐个读取元素(即步长默认值为 1)。如 lst[0:3:2]表示在索引号为 0～3 的元素中,读取索引号为 0、2 的元素。正数步长表示从左往右切片,负数步长表示从右到左切片。

(6) 切片遵循“左闭右开”原则,即切片结果不包括“终止索引号”的元素。正向切片时,元素“起始索引号”要小于“终止索引号”,否则将输出空列表。

注意:列表定义与列表切片是不同的操作,它们的区别如图 3-3 所示。

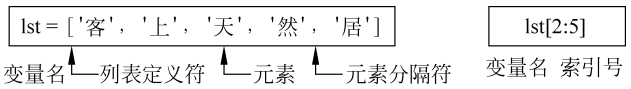

图 3-3　列表定义(左)和列表切片(右)的区别

4. 程序设计:列表切片

【例 3-17】 lst=['宝玉', '黛玉', '宝钗', '晴雯', '王熙凤', '袭人'],列表切片如下:

```
>>> lst = ['宝玉', '黛玉', '宝钗', '晴雯', '王熙凤', '袭人']  # 定义列表(符号[ ]为列表定义符)
>>> lst[:]                                                    # 切片全部元素
    ['宝玉', '黛玉', '宝钗', '晴雯', '王熙凤', '袭人']
>>> lst[3:]                                                   # 切片自索引号 3 起,至尾部
                                                              # 止,步长默认为 1
    ['晴雯', '王熙凤', '袭人']
>>> lst[3]                                                    # 只切片索引号为 3 的元素
    ['晴雯']
>>> lst[1:4]                                                  # 切片自索引号 1 起,至索引号 4
                                                              # 止,步长默认为 1
    ['黛玉', '宝钗', '晴雯']
>>> lst[1:4:2]                                                # 切片自索引号 1 起,至索引号 4
                                                              # 止,步长为 2
    ['黛玉', '晴雯']
>>> lst[::2]                                                  # 切片自索引号 0 起,至尾部
                                                              # 止,步长为 2
    ['宝玉', '宝钗', '王熙凤']
>>> lst[::-1]                                                 # 全部元素逆序输出
    ['袭人', '王熙凤', '晴雯', '宝钗', '黛玉', '宝玉']
>>> lst[-2:-5]                                                # 切片自索引号－2 起,至索引号
                                                              # －5 止,越界错误
    []                                                        # 输出为空
```

```
>>> lst[-5:-2]                                    # 切片自索引号-5起,至索引号
                                                  # -2止
['黛玉', '宝钗', '晴雯']
```

注意：切片时，如果偏移量越界（索引号超出列表最大长度），Python 仍然按照最大长度处理。如切片索引号大于列表的最大长度（越界）时，Python 仍然会按照最大索引号处理。

5. 在列表中添加元素

向列表中添加或从中删除元素时，Python 会对列表自动进行存储大小和索引号顺序调整。为了提高程序运行效率，应当尽量从列表尾部进行元素添加或删除操作。

向列表中添加元素的函数有 append()、extend()和 insert()，语法如下：

```
1  列表名.append(元素)                    # 在列表尾部添加1个元素
2  列表名.extend([元素列表])              # 在列表尾部添加多个元素
3  列表名.insert(索引号, 元素)            # 在列表指定位置插入元素或列表
```

【例 3-18】 用函数 append()在列表末尾添加一个元素，程序如下：

```
>>> lst = ['宝玉', '黛玉', '宝钗']          # 定义列表lst(列表有3个元素)
>>> lst.append('晴雯')                      # 在列表尾部添加1个元素'晴雯'
>>> lst                                     # 输出列表lst
['宝玉', '黛玉', '宝钗', '晴雯']           # 列表添加元素时,列表会自动扩展
```

【例 3-19】 用函数 extend()在列表末尾添加多个元素，程序如下：

```
>>> lst = ['官', '君莫想']                  # 定义列表lst
>>> lst.extend(['钱', '君莫想'])            # 在列表尾部添加元素'钱'和'君莫想'
>>> lst
['官', '君莫想', '钱', '君莫想']
```

【例 3-20】 用函数 insert()在列表指定位置插入元素，程序如下：

```
>>> lst = ['日日', '无事事', '亦茫茫']      # 定义列表lst
>>> lst.insert(2, '忙忙')                   # 在列表索引号2的位置添加元素'忙忙'
>>> lst                                     # 输出列表
['日日', '无事事', '忙忙', '亦茫茫']
```

说明：插入元素时，若索引号为－1，则在最后一个元素之后插入。

6. 多个列表的连接

【例 3-21】 用加法（＋）连接列表，程序如下：

```
>>> lst1 = ['寒蝉凄切', '对长亭晚']         # 定义列表lst1
>>> lst2 = ['骤雨初歇']                     # 定义列表lst2
>>> lst1 + lst2                             # lst1和lst2连接
['寒蝉凄切', '对长亭晚', '骤雨初歇']       # 输出列表
```

7. 列表的嵌套

【例 3-22】 列表嵌套即在列表里再定义列表，程序如下：

```
>>> lst1 = ['孙悟空', '猪八戒', '沙和尚']   # 为列表lst1赋值
>>> lst2 = [1000, 700, 300]                 # 为列表lst2赋值
```

```
>>> lst3 = [lst1, lst2]                          # 连接列表 lst1 和 lst2,形成嵌套列表 lst3
>>> lst3                                         # 输出嵌套列表 lst3
[['孙悟空', '猪八戒', '沙和尚'], [1000,
700, 300]]
```

8. 判断列表

【例 3-23】 利用布尔函数判断列表是否为“空”,程序如下:

```
>>> lst = [‘比人心’, ‘山未险’]                    # 定义列表 lst
>>> bool(lst)                                    # 利用布尔函数,判断列表是否为空
True                                             # True 表示列表不为空,False 表示列表为空
```

【例 3-24】 利用 index()函数判断元素在列表中的位置,程序如下:

```
>>> lst = ['情', '深', '深', '雨', '蒙', '蒙']    # 定义列表 lst
>>> lst.index('蒙')                              # 查找字符串中元素“'蒙'”的索引号
4                                                # 找到的第 1 个元素(索引号为 4)
>>> lst.count('深')                              # 统计字符串中元素“'深'”出现的次数
2
```

9. 修改列表中元素

修改列表元素的语法如下:

```
1  变量名[元素索引号] = 修改内容
```

【例 3-25】 在列表 lst 中,将元素“栏杆”修改为“阑干”,程序如下:

```
>>> lst = ['把吴钩看了', '栏杆拍遍', '无人会', '登临意']    # 定义列表 lst
>>> lst[1] = '阑干拍遍'                                    # 修改索引号为 1 的元素值
>>> lst
['把吴钩看了', '阑干拍遍', '无人会', '登临意']            # 修改列表中元素
```

读取、删除、修改列表时,容易出现越界错误。列表切片时不提示越界错误。提示偏移量越界的操作有 lst[偏移量]、del lst[偏移量]、lst.remove(值)、lst.pop(偏移量),如果偏移量越界,这些操作都会报错。

10. 删除列表元素

删除列表中指定索引号的元素,可以使用函数 pop(),语法如下:

```
1  列表名.pop(元素索引号)                    # 语法 1
2  del 列表名[元素索引号]                    # 语法 2
```

【例 3-26】 删除列表中指定位置元素的函数有 pop()、del,程序如下:

```
>>> lst = ['天不教人客梦安', '昨夜春寒', '今夜春寒']    # 定义列表 lst
>>> lst.pop(2)                                        # 删除列表中索引号为 2 的元素
'今夜春寒'
>>> lst
['天不教人客梦安', '昨夜春寒']
>>> del lst[1]                                        # 删除列表中索引号为 1 的元素
>>> lst
['天不教人客梦安']
```

3.5 元组——不可修改的数据类型

1. 元组定义语法

元组也是一种数据存储容器。元组与列表的区别在于元组中的元素不能修改，而列表中的元素可以修改；元组和列表的定义符号不一样，元组使用圆括号定义，而列表使用方括号定义。元组定义语法如下：

```
元组名 = (元素 1, 元素 2, 元素 3, …, 元素 n)
```

2. 元组定义案例

【例 3-27】 定义元组的简单示例，程序如下：

```
>>> tup1 = ('春', '风', '吹', '又', '生')    # 定义元组 tup1(左右两个圆括号为元组定义符)
>>> tup1                                     # 输出元组
('春', '风', '吹', '又', '生')
>>> tup2 = (12, 34, 56, 78)                  # 定义元组
>>> tup2                                     # 输出元组
(12, 34, 56, 78)
```

元组只有一个元素时，必须在元素后面增加一个逗号(见例 3-28)。如果没有逗号，Python 会假定这只是一对额外的圆括号，不会定义一个元组。

在不引起语法错误的情况下，可以用逗号分隔的一组值定义元组(见例 3-28 第 2 行)。也就是说，在没有歧义的情况下，元组可以没有括号。

【例 3-28】 元组的特殊定义方法，程序如下：

```
>>> tup1 = (0,)                              # 元组为单个元素时，必须加逗号以防出错
>>> tup2 = 1, 2, 3, 4, 5                     # 定义元组时，可以省略圆括号()
>>> tup2                                     # 输出元组
(1, 2, 3, 4, 5)
```

3. 元组切片方法

元组切片方式与列表相同，元组索引号也与列表相同。元组访问语法如下：

```
元组名[索引号]
```

【例 3-29】 切片访问元组中的某个元素，程序如下：

```
>>> tup1 = ('月', '是', '故', '乡', '明')    # 定义元组 tup1
>>> tup1[2]                                  # 访问元组中第 3 个元素(元组可读不可写)
'故'
>>> tup2 = (1, 2, 3, 4, 5)                   # 定义元组 tup2
>>> tup2[2]                                  # 访问元组中第 3 个元素
3
```

3.6　字典——键值对数据类型

1. 字典的特征

字典是 Python 中一种重要的数据结构。字典中每个元素分为两部分，前半部分称为“键”(key)，后半部分称为“值”(value)。例如{'姓名':'张飞'}元素中，“姓名”称为“键”，“张飞”称为“值”，它们一起称为“键值对”。字典是“键值对”元素的集合。字典类似于表格中的一行(记录)，“键”相当于表格的字段名称；“值”相当于表格单元格中的值。

列表、元组、字典都是有序对象集合。它们之间的区别在于字典中的元素通过“键”进行查找，而列表中的元素通过索引号进行查找。

2. 字典定义语法

字典中每个“键值对”(key-value)用冒号“:”分隔，每个“键值对”之间用逗号(,)分隔，整个字典包含在花括号{ }中(见图 3-4)。字典定义语法如下：

```
字典名 = {键 1:值 1, 键 2:值 2, …,键 k:值 v}
```

字典中，键可以是字符串、数字、元组等数据类型。键不能重复，如果键有重复，最后一个键值对会替换前面的键值对；值可以重复；键和值的数据类型没有限制。

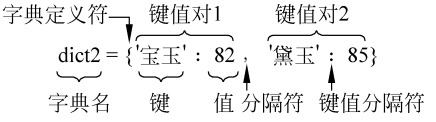

图 3-4　字典案例示意图

3. 字典定义案例

【例 3-30】 定义字典的各种方法示例，程序如下：

```
>>> dict1 = {'姓名':'张飞', '战斗力':1000}   # "'姓名'"是键,"'张飞'"是值,它们是键值对
>>> dict2 = {'宝玉':82, '黛玉':85}          # 同一字典中,键不能重复,值允许重复
>>> dict2['宝钗'] = 88                      # 在字典 dict2 中添加一个键值对{'宝钗':88}
>>> dict3 = {(80, 90):'优良', 60:'及格'}    # 键也可以是元组,如(80, 90)
```

可以通过字典中的“键”查找字典中的“值”。

4. 访问字典元素

字典中元素虽然是有序的(Python 3.6 以后的版本)，但是字典中的元素没有索引号，因此访问字典中的元素可以由键查找值。字典访问语法如下：

```
字典名[键名]
```

【例 3-31】 在字典中读取人物身高信息，程序如下：

```
>>> people = {'姓名':'张飞', '性别':'男', '身高':'180cm'}
>>> people['身高']                                    # 访问字典,由键查找值
'180cm'
```

5. 修改字典中的元素

修改字典是指添加新键值对、修改或删除已有键值对。

【例 3-32】 向字典中添加新内容，程序如下：

```
1   dict1 = {'姓名':'关云长', '年龄':40, '战斗力':'一级'}          # 定义字典
2   dict1['年龄'] = 48                                          # 修改字典条目
3   dict1['宝贝'] = '赤兔马'                                    # 尾部新增键值对
4   print('字典:', dict1)                                       # 输出字典
>>>  字典:{'姓名': '关云长', '年龄': 48, '战斗力': '一级', '宝贝': '赤兔马'}   # 程序输出
```

6. 程序设计：成绩统计序列输出

【例 3-33】 某班成绩如表 3-3 所示，用字典数据类型输出学生成绩。

表 3-3 某班考试成绩（部分）

姓名	语文	数学
周小碧	85	92
秦天际	76	55
韩织烟	92	88

输出学生成绩程序如下：

```
1   students = [                                        # 定义列表
2       {'姓名': '周小碧', '语文': 85, '数学': 92},      # 定义字典
3       {'姓名': '秦天际', '语文': 76, '数学': 55},
4       {'姓名': '韩织烟', '语文': 92, '数学': 88},
5   ]
6   print('考试成绩如下:', students)                      # 打印输出
>>>  考试成绩如下:[{'姓名': '周小碧', '语文': 85, …      # 程序输出(略)
```

说明：平均成绩计算参见案例 6。