

《Python程序设计》

Python数据类型

刘潇

机械科学与工程学院

2023年10月16日

2023秋季

Python数据类型

Python中常用的内置数据类型

数据类型	类型名称	示例
数字	int, float, complex	12, 1.1, 1.2e3, 1+2j
字符串 (序列)	str	"hello world" , 'python'
容器	列表 (序列)	[1, 2, 3], ['a' , 'b' , 'c']
	元组 (序列)	(1, 2, 3), ('a' , 'b' , 'c')
	字典 (映射)	{ 'username' : 'admin' , 'age' : 20 }

使用 `type()` 函数获取任何对象的数据类型

本节要点

- 掌握列表、元组、字典的常用操作和方法
- 深入理解列表、元组、字典的可变或不可变类型的差异
- 理解Python中的变量与对象间的共享引用和内存管理

主要内容

1. 列表

2. 元组

3. 字典

如何存储多个数据？

存储一个班学生的姓名

厉害了，我的班！
CLASS 4

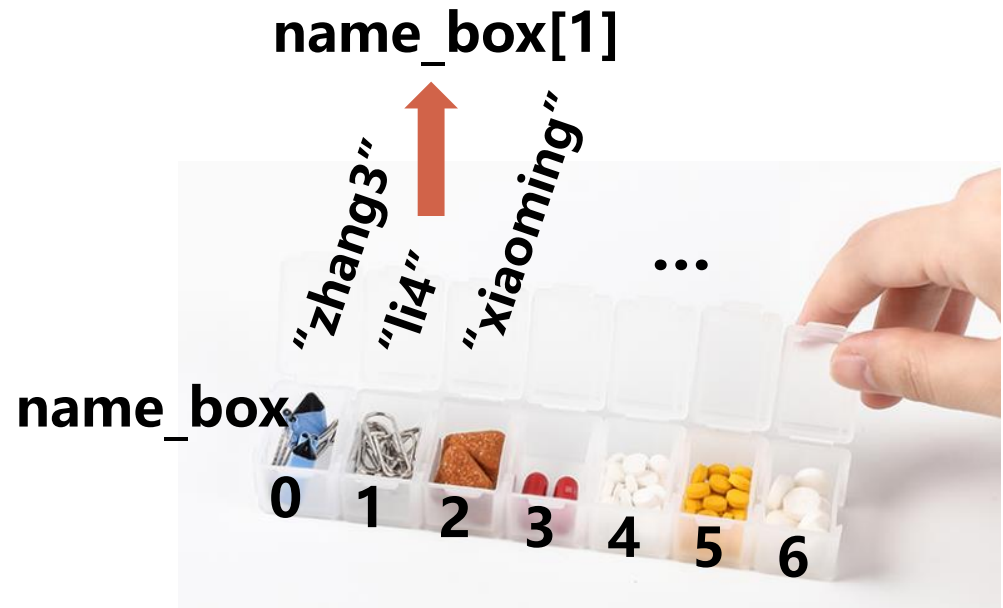


name_1 = "zhang3"

name_2 = "li4"

...

数据收纳盒



容器按照一定的规则存储数据，方便数据操作

列表

□ 列表(list): 一种**有序**和**可变**的集合, 用于存储一串数据, 允许重复的成员

```
list1 = [ "zhang3" , " li4" , "wang5" , 1, 5.5, 1+2j, 1]
```



- 所有元素放在**方括号**中, 相邻元素之间用**逗号**分隔
- 同一个列表中的元素的数据类型可以互不相同, 可以同时包含数字、字符串、列表、元组、字典、函数以及其他任意对象
- 使用 "=" 赋值创建列表, []表示空列表

利用列表创建数据库

创建学生数据库，第一个元素是姓名，第二个元素是年龄

列表

```
1 a = ["zhang3", "U202011054"]  
2 b = ["li4", "U202011055"]  
3 c = ["wang5", "U202011056"]  
4 student = [a, b, c]  
5 print(student)
```

```
[['zhang3', 'U202011054'], ['li4', 'U202011055'], ['wang5', 'U202011056']]
```

列表内置函数

□ `len()`返回序列包含的元素个数，`min()`和`max()`分别返回列表中的最小值和最大值

列表内置函数

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 print(len(numbers))
3 print(min(numbers))
4 print(max(numbers))
```

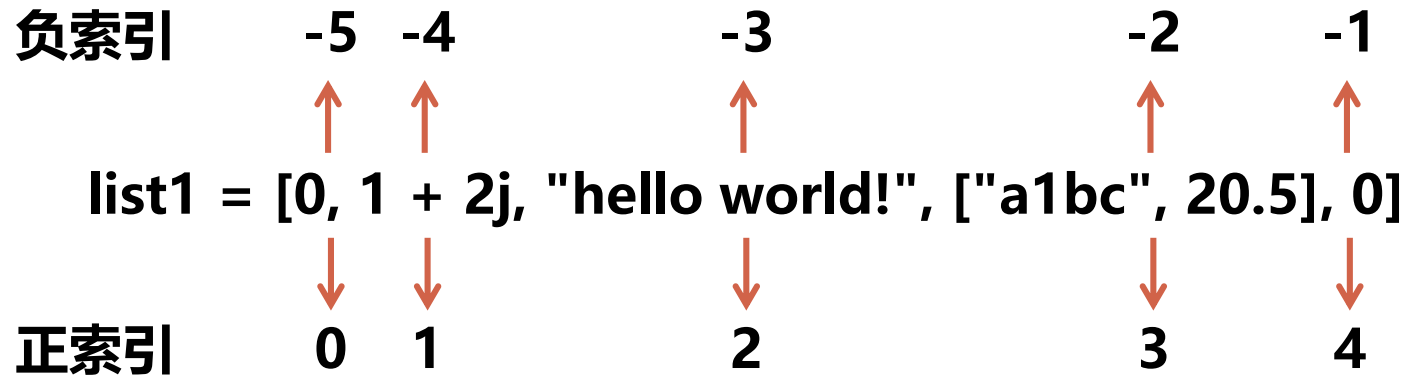
10

1

10

列表操作

□ 列表属于序列，字符串的索引、切片、相加、相乘和成员资格检查等操作都适用于列表



`print(list1[3][-1])`

`print(list1[3][-2].title())` (调用元素的方法)

`print(list1[3][-2][1])` (字符串 or 整数?)

`print(list1[5]) ?`

切片

列表切片操作

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 print(numbers[3:6])
3 print(numbers[:2])
4 print(numbers[5:])
5 print(numbers[:])
6 print(numbers[5:-2])
7 print(numbers[8:-4])
8 print(numbers[0:10:2])
9 print(numbers[10:0:-2])
10 print(numbers[::-4])
```

- 含第一个索引，不含第二个索引
- 可省略前后两个索引
- 反向索引切片
- 切片步长

[4, 5, 6]

[1, 2]

[6, 7, 8, 9, 10]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

[6, 7, 8]

[]

[1, 3, 5, 7, 9]

[10, 8, 6, 4, 2]

[1, 5, 9]

列表相加

□ 通过 “+” 拼接列表，列表只能与列表相加

列表相加操作

```
1 a = [1, 2, 3]
2 b = [4, 5, 6]
3 c = ["hello world!"]
4 d = "hello world!"
5 print(a + b)
6 print(a + c)
7 print(a + d)
```

```
[1, 2, 3, 4, 5, 6]
```

```
[1, 2, 3, 'hello world!']
```

```
TypeError                                Traceback (most recent call last)
<ipython-input-25-3a70906d19f4> in <module>
      5 print(a + b)
      6 print(a + c)
----> 7 print(a + d)
```

```
TypeError: can only concatenate list (not "str") to list
```

列表相乘

□ 列表与数x相乘时，重复列表x次来创建新列表

列表相乘操作

```
1 a = [1, 2, 3]
2 print(a * 3)
3
4 #创建空列表
5 b = [None]
6 c = b * 10
7 print(c)
8 print(type(c[1]))
```

[1, 2, 3, 1, 2, 3, 1, 2, 3]

[None, None, None, None, None, None, None, None, None, None]

<class 'NoneType'>

打印指定年月日的日期

打印指定年月日的日期

```
1 months = ["January", "February", "March", "April", \
2           "May", "June", "July", "August", \
3           "September", "October", "November", "December"]
4 day_endings = ["st", "nd", "rd"] + ["th"]*17 + ["st", "nd", "rd"] + ["th"]*7 + ["st"]
5
6 Year = input("Year: ")
7 Month = int(input("Month(1-12): "))
8 Day = int(input("Day(1-31): "))
9
10 #打印日期
11 month_name = months[Month - 1]
12 day_name = str(Day) + day_endings[Day - 1]
13 print(month_name, day_name, Year)
```

Year: 2020
Month(1-12): 8
Day(1-31): 30
August 30th 2020

成员资格检查

- 用运算符 “in” 检查特定的值是否包含在列表中，在返回 True，不在返回 False

成员资格检查

```
1 a = ["U202011054", 19]
2 b = ["U202011055", 20]
3 c = ["U202011056", 21]
4 student = [a, b, c]
5
6 ID = input("Student ID:")
7 age = int(input("Student age:"))
8 if [ID, age] in student:
9     print("OK!")
10 else:
11     print("ERROR!")
12
```

```
Student ID:U202011054
Student age:19
OK!
```

列表与字符串转换（字符串序列）

□ 利用函数list()将字符串转换为列表

□ 利用字符串方法"".join()将列表转化为字符串

列表与字符串转换

```
1 print(list("hello"))
2 print("".join(['h', 'e', 'l', 'l', 'o']))
3 print(list(['h', 'e', 'l', 'l', 'o']))
4
5 #元素是字符类型
6 print(list("hello1"))
7 print("".join(['h', 'e', 'l', 'l', 'o', 1]))
```

['h', 'e', 'l', 'l', 'o']

hello

['h', 'e', 'l', 'l', 'o']

['h', 'e', 'l', 'l', 'o', '1']

TypeError

Traceback (most recent call last)

<ipython-input-11-ffb540ef8b56> in <module>

5 #元素是字符类型

6 print(list("hello1"))

----> 7 print("".join(['h', 'e', 'l', 'l', 'o', 1]))

TypeError: sequence item 5: expected str instance, int found

列表元素赋值

□ 利用索引给列表元素赋值

列表元素赋值

```
1 a = [1, 2, 3]
2 print(id(a))
3 a[1] = 3
4 print(a, id(a))
5 a[1] = [1, 2]
6 print(a, id(a))
7
8 b = a
9 print(b, id(b), id(a))
```



原位改变



列表异构



列表引用

```
97865664
[1, 3, 3] 97865664
[1, [1, 2], 3] 97865664
[1, [1, 2], 3] 97865664 97865664
```

id(object)函数是返回对象object在其生命周期内位于**内存中的地址**，id函数的参数类型是一个对象

给列表a的元素重新赋值，b会改变吗？

浅拷贝

浅拷贝

```
1 a = [1, 2, 3]
2 b = a
3 print(b, id(b), id(a))
4
5 #给a中的元素重新赋值
6 a[1] = 1
7 print(a, b, id(a), id(b))
```



浅拷贝，仅仅是创建了引用

```
[1, 2, 3] 96542400 96542400
[1, 1, 3] [1, 1, 3] 96542400 96542400
```

字符串引用（或者数字引用）

```
1 a = "123"
2 b = a
3 print(b, id(b), id(a))
4
5 #给a中的元素重新赋值
6 a = "456"
7 print(a, b, id(a), id(b))
```



字符串重新申请内存

```
123 96550320 96550320
456 123 96550256 96550320
```

深拷贝

深拷贝

```
1 a = [1, 2, 3]
2 b = [1, 2, 3]
3 print(id(a), id(b))
4
5 #给a中的元素重新赋值
6 a[1] = 1
7 print(a, b, id(a), id(b))
8
9 #深拷贝
10 b = a[:]
11 a[2] = 1
12 print(a, b, id(a), id(b))
```



利用切片赋值

```
95626240 95626496
[1, 1, 3] [1, 2, 3] 95626240 95626496
[1, 1, 1] [1, 1, 3] 95626240 95631360
```



非引用，新申请内存

切片赋值&删除元素

切片赋值&删除元素

```
1 name = list("perl")
2 name[1:] = list("ython")
3 print("".join(name))
4
5 #插入新元素
6 numbers = [1, 5]
7 numbers[1:1] = [2, 3, 4]
8 print(numbers)
9
10 #删除元素
11 numbers[1:4] = []
12 print(numbers)
13
14 del numbers[1]
15 print(numbers)
```



利用列表替换空切片



利用空列表替换

```
python
[1, 2, 3, 4, 5]
[1, 5]
[1]
```

列表方法

□ 增加元素: [1, 2, 3] + [4]

方法及使用	描述
list1.append()	向尾部添加 一个新元素 a = [1, 2, 3] a.append(4), a为[1, 2, 3, 4]
list1.extend()	同时将多个值附加到列表末尾 a = [1, 2, 3] b = [4, 5, 6] a.extend(b), a为[1, 2, 3, 4, 5, 6] 与a[len(a):] = b等效
list1.insert()	将一个元素 (第二个参数) 插入到列表的指定位置 (第一个参数) a = [1, 2, 3] a.insert(1, "hello world!") a为[1, "hello world!", 2, 3]

检索元素

方法及使用

描述

`list1.count()`

计算指定元素在列表中出现的次数

`a = [1, [1, 2], 1, 2, 3]`

`a.count(1)`结果为2

`list1.index()`

返回某个元素在列表中的准确位置，若不在列表中会报错，若有多个相同元素，返回最小的索引位置

`a = [1, 2, 3]`

`a.index(1)`结果为0

删除元素

方法及使用	描述
<code>list1.clear()</code>	原位清空列表 <code>a = [1, 2, 3]</code> <code>a.clear()</code> , <code>a</code> 为[] 与 <code>a[:] = []</code> 等效
<code>list1.remove()</code>	删除第一个为指定值的元素, 无返回值 <code>a = [1, 2, 3, 1]</code> <code>a.remove(1)</code> , <code>a</code> 为[2, 3, 1]
<code>list1.pop()</code>	从列表中删除一个元素, 并返回这一元素 默认为最后一个元素 <code>a = [1, 2, 3]</code> <code>a.pop()</code> 结果为3 <code>a</code> 为[1, 2]

列表排序

方法及使用

描述

`list1.reverse()`

按相反的顺序排列列表中的元素
原位修改，无返回值

`a = [1, 3, 2]`
`a.reverse()`, `a`为`[2, 3, 1]`

`list1.sort()`

对列表元素进行排序，默认为升序排列
原位修改，无返回值

`a = [1, 3, 2]`
`a.sort()`, `a`为`[1, 2, 3]`

计算标准差

计算列表[2,4,6,9,13]中数字的标准差S

$$S = \sqrt{\frac{\sum(\bar{x} - x_i)^2}{n-1}}$$

```
1 import math
2 numbers = [2, 4, 6, 9, 13]
3 total = 0
4 # 计算平均值
5 for number in numbers:
6     total += number
7 average = total/len(numbers)
8
9 # 计算标准差
10 total = 0
11 for number in numbers:
12     total += (number-average)**2
13 S = math.sqrt(total/(len(numbers)-1))
14 print(S)
```

4. 324349662087931

怎么删除列表中所有的2

为什么列表中还有2？

```
1 numbers = [2, 5, 2, 2, 5, 0, 99, 2, 2, 2]
2 for number in numbers:
3     if number == 2:
4         numbers.remove(2)
5         print(numbers)
6
7
8
9
10
```

```
[5, 2, 2, 5, 0, 99, 2, 2, 2]
[5, 2, 5, 0, 99, 2, 2, 2]
[5, 5, 0, 99, 2, 2, 2]
[5, 5, 0, 99, 2, 2]
[5, 2, 2, 5, 0, 99, 2, 2, 2]
[5, 2, 5, 0, 99, 2, 2, 2]
[5, 5, 0, 99, 2, 2, 2]
[5, 5, 0, 99, 2, 2]
[5, 5, 0, 99, 2]
[5, 5, 0, 99]
```

还有没有别的方法？

找出不超过n的所有素数

- 埃拉托斯特尼筛法：首先创建从2到n的数字列表，第一个数字从列表中删除，并作为素数公布，而且将该数字的所有倍数从列表中删除。此过程一直持续到列表为空。

找出不超过20的所有素数

```
1  # 创建2到20的数字列表
2  numbers = list(range(2, 21))
3  prime_numbers = []
4
5  while numbers: # 重复以下步骤
6      # 删除列表中的第一个数，声明为素数
7      prime_numbers.append(numbers[0])
8      del(numbers[0])
9
10     # 删除该数字的倍数
11     for number in numbers:
12         if number % prime_numbers[-1] == 0:
13             numbers.remove(number)
14
15     print(prime_numbers)
```

[2, 3, 5, 7, 11, 13, 17, 19]

元组

□ 元组tuple是不可修改的列表，代码更安全

元组 `tuple1 = (0, 1 + 2j, "hello world!", ["abc", 20.5], 0)`

有序的，逗号比小括号重要

```
1  #定义元组
2  tuple1 = (0, 1 + 2j, "hello world!", ["abc", 20.5], 0)
3  print(tuple1)
4  tuple2 = 0, 1 + 2j, "hello world!", ["abc", 20.5], 0
5  print(tuple2)
6
7  tuple3 = 12,
8  print(tuple3, type(tuple3))
9
10 number = (12)
11 print(number, type(number))
12
13 #空元组
14 tuple4 = ()
15 print(tuple4, type(tuple4))
```

可对元组进行索引、切片、相加、相乘和成员检查等操作

```
(0, (1+2j), 'hello world!', ['abc', 20.5], 0)
(0, (1+2j), 'hello world!', ['abc', 20.5], 0)
(12,) <class 'tuple'>
12 <class 'int'>
() <class 'tuple'>
```

元组是不可变的

修改元组中的元素

```
1 #修改元组中的元素
2 tuple1 = (0, 1 + 2j, "hello world!", ["abc", 20.5], 0)
3 tuple1[0] = 1
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-73-20f69565a395> in <module>
      1 #修改元组中的元素
      2 tuple1 = (0, 1 + 2j, "hello world!", ["abc", 20.5], 0)
----> 3 tuple1[0] = 1
```

```
TypeError: 'tuple' object does not support item assignment
```

tuple1[3][1]是否可修改呢?

元组中的列表可修改

修改元组中的元素

```
1 tuple1 = (0, 1 + 2j, "hello world!", ["abc", 20.5], 0)
2 print(id(tuple1))
3 tuple1[3][1] = 50.5
4 print(tuple1, id(tuple1))
```

95211984

(0, (1+2j), 'hello world!', ['abc', 50.5], 0) 95211984

原位修改

列表中的元组可修改吗？

修改列表中的元组

```
1 list1 = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]
2 list1[1][1] = 10
3 print(list1)
```

TypeError

Traceback (most recent call last)

<ipython-input-11-dd27c7595a80> in <module>

```
1 list1 = [(1, 2, 3), (4, 5, 6), (7, 8, 9)]
```

```
----> 2 list1[1][1] = 10
```

```
3 print(list1)
```

为什么？

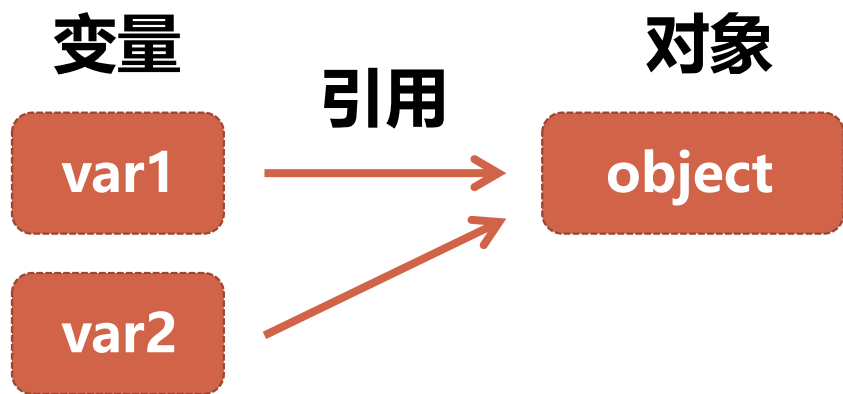
TypeError: 'tuple' object does not support item assignment

变量与对象

□ Python解释器内存中有个对象池，所有对象都放在这个池子里。Python中对变量的赋值是**引用对象**的过程



变量指针指向具体对象的内存空间，取对象的值。



1、Python**缓存了整数和短字符串**，因此每个对象在内存中只存有一份，引用所指对象就是相同的，即使使用赋值语句，也只是创造新的引用，而不是对象本身；

2、Python**没有缓存长字符串、列表及其他对象**，可以有多个相同的对象，可以使用赋值语句创建出新的对象。

变量与对象

变量与对象

整数10为一个对象。而number是一个引用。利用赋值语句，引用number指向对象10

```
1 # 数字和短字符串
2 number = 10
3 print(id(number), id(10))
4 b = 10
5 print(id(b), id(number), id(10))
6
7 # 长字符串、列表等
8 string = "hello world "*5
9 c = "hello world "*5
10 print(id(c), id(string))
```

相同的内存地址

不同的内存地址

```
8791159027776 8791159027776
8791159027776 8791159027776 8791159027776
98675376 98675152
```

变量与对象

- Python的对象分为两种：可变对象（列表、字典）和不可变对象（数字、字符串、元组）。
- 可以用变量名去使用不可变对象，但不能直接改变对象里面的内容。对其修改只能重新赋值，而且是在新的内存中创建新的对象后重新赋值

```
1 # 重新赋值
2 number = 11
3 print(id(number), id(11), id(10))
4
5 # 不可变对象不能修改
6 string[1] = "s"
```

新申请一段内存来存储对象11，
再让number去指向对象11

8791159027808 8791159027808 8791159027776

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-26-b4545e69e4db> in <module>
      4
      5 # 不可变对象不能修改
----> 6 string[1] = "s"
```

TypeError: 'str' object does not support item assignment

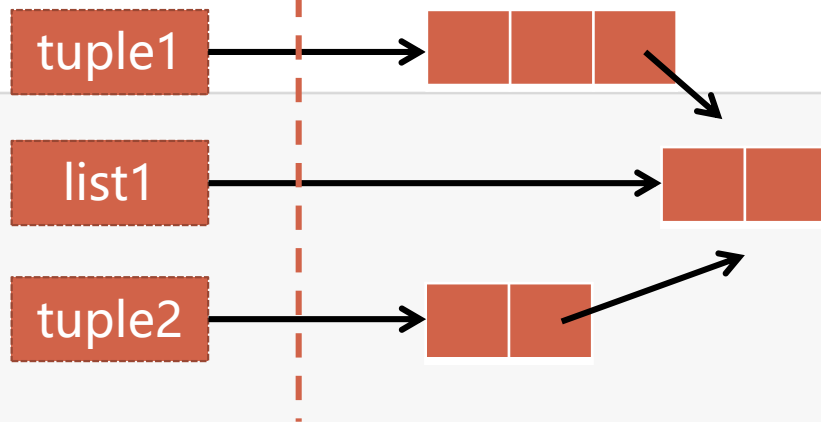
共享的引用

- 元组本身是不可变对象，当元组变量与对象建立引用关系后，元组对象地址的内容不可变了
- 当元组其中的元素是对象引用，如果这个对象本身是可变的，可以利用元组的索引修改该元素的引用，进而修改可变元素对象，而且**与之共享引用的名字都将改变**

```
1 number = 10
2 string = "hello world"
3 list1 = ["abc", 20.5]
4 tuple1 = (number, string, list1)
5 tuple2 = (list1, string)
6 print(list1, tuple1, tuple2)
7 tuple1[2][1] = 50.5
8 print(list1, tuple1, tuple2)
```

变量

对象



```
['abc', 20.5] (10, 'hello world', ['abc', 20.5]) (['abc', 20.5], 'hello world')
['abc', 50.5] (10, 'hello world', ['abc', 50.5]) (['abc', 50.5], 'hello world')
```

元组常用函数

元组常用函数

```
1 tuple1 = (2, 5, 2, 2, 5, 0, 99, 2, 2, 2)
2 # 元组长度
3 print(len(tuple1))
4
5 # 最大值
6 print(max(tuple1))
7
8 # 最小值
9 print(min(tuple1))
10
11 # 序列转换为元组
12 list1 = ["h", "e", "l", "l", "o"]
13 strings1 = "hello"
14 print(tuple(list1))
15 print(tuple(strings1))
```

```
10
99
0
('h', 'e', 'l', 'l', 'o')
('h', 'e', 'l', 'l', 'o')
```

元组方法

□ 用于修改序列的方法不适用于元组，如append、extend、insert、remove、pop、reverse、sort等

元组方法

```
1 tuple1 = (2, 5, 2, 2, 5, 0, 99, 2, 2, 2)
2 tuple1.append(2)
```

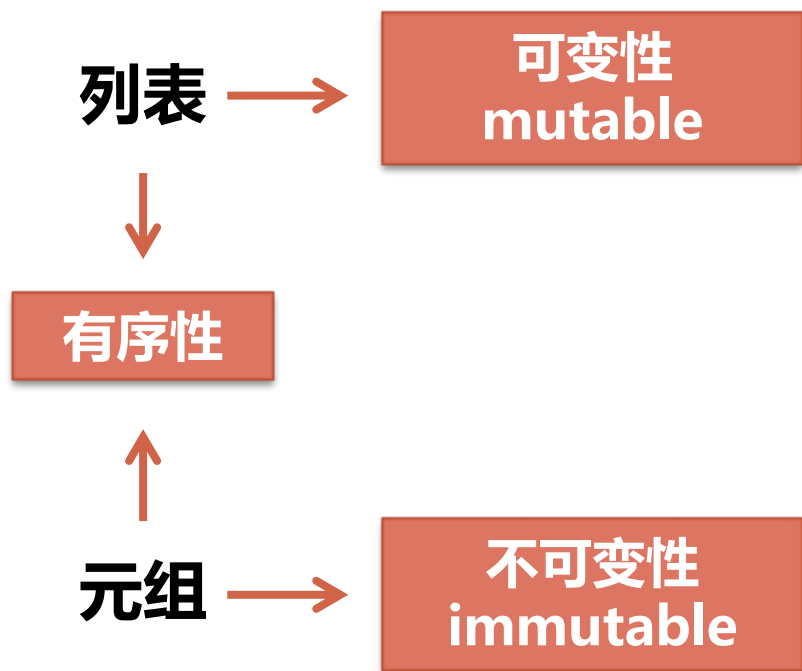
```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-10-27ccc6204b61> in <module>
      1 tuple1 = (2, 5, 2, 2, 5, 0, 99, 2, 2, 2)
----> 2 tuple1.append(2)
```

```
AttributeError: 'tuple' object has no attribute 'append'
```

count()和index()可用

列表可变与元组不可变

- 列表是可变的，列表创建后允许对其**原位**进行修改、增加、删除、排序等操作



元组牺牲了可变性，增加了数据处理性能和安全性

字典

- 元组dict是Python中唯一的内置映射类型，是一个无序、可变和有索引的集合
- 字典中的值不按顺序排列，而是存储在键下

创建字典

```
1 names = ["zhang3", "li4", "wang5"]
2 numbers = [1001, 1002, 1003]
3
4 #li4的编号是
5 print(numbers[names.index("li4")])
6
7 #创建字典staffbook
8 staffbook = {"zhang3":1001, "li4":1002, "wang5":1003}
9 print(staffbook["li4"])
```

字典用花括号{}表示

1002
1002

字典中的项，用逗号隔开
项由键和值组成，用冒号隔开

字典可以看成是元素对构成的列表，查找效率高

利用函数dict创建字典

利用函数dict创建字典

```
1 staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
2 print(staffbook1)
3
4 #从字典创建字典
5 staffbook2 = dict(staffbook1)
6 staffbook3 = staffbook1
7 print(staffbook2, staffbook3)
8 print(id(staffbook2), id(staffbook3), id(staffbook1))
9
10 #从键-值对序列创建字典
11 staffbook4 = dict([("zhang3",1001), ("li4",1002), ("wang5",1003)])
12 print(staffbook4)
13 staffbook5 = dict([("zhang3",1001), ("li4",1002), ("wang5",1003)])
14 print(staffbook5)
15
16 #利用关键字实参创建字典
17 staffbook6 = dict(zhang3=1001, li4=1002, wang5=1003)
18 print(staffbook6)
```

```
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003} {'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
99064128 99062784 99062784
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
```

字典中的键-值对

□ 字典中的键是互不相同的，且是不可变类型（数字、字符串或元组）

字典中的键-值对

```
1 staffbook1 = {"zhang3":1001, "zhang3":1002, "wang5":1003}
2 print(staffbook1)
3
4 staffbook1 = {"zhang3":1001, "li4":1001, "wang5":1003}
5 print(staffbook1)
6
7 staffbook2 = {"zhang3":1001, ["zhang3"]:1002, "wang5":1003}
8 print(staffbook2)
```

```
{'zhang3': 1002, 'wang5': 1003}
{'zhang3': 1001, 'li4': 1001, 'wang5': 1003}
```

```
TypeError                                Traceback (most recent call last)
<ipython-input-108-a99c23e844c3> in <module>
      5 print(staffbook1)
      6
----> 7 staffbook2 = {"zhang3":1001, ["zhang3"]:1002, "wang5":1003}
      8 print(staffbook2)
```

```
TypeError: unhashable type: 'list'
```

字典操作

□ 序列的操作如索引、切片、相加、相乘不适用于字典

字典操作

```
1 staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
2 print(len(staffbook1))
3
4 print(staffbook1["zhang3"])
5
6 staffbook1["zhang3"] = 1005
7 print(staffbook1)
8
9 del(staffbook1["zhang3"])
10 print(staffbook1)
11
12 print("zhang3" in staffbook1)
```

字典长度

返回键对应的值

给对应的键赋值

删除对应的键-值对

成员检查

```
3
1001
{'zhang3': 1005, 'li4': 1002, 'wang5': 1003}
{'li4': 1002, 'wang5': 1003}
False
```


字典赋值

□ 字典可以给不存在的键赋值，相当于在字典中创建一个新项

字典中创建新项

```
1 staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
2 #创建新项
3 staffbook1["ma6"] = 1004
4 print(staffbook1)
5
6 list1 = [("zhang3",1001), ("li4",1002), ("wang5",1003)]
7 list1[3] = ("ma6", 1003)
```

超出列表索引范围

```
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003, 'ma6': 1004}
```

```
-----
IndexError                                Traceback (most recent call last)
<ipython-input-118-2ead1583a2bc> in <module>
      5
      6 list1 = [("zhang3",1001), ("li4",1002), ("wang5",1003)]
----> 7 list1[3] = ("ma6", 1003)
```

```
IndexError: list assignment index out of range
```

数据库字典示例

数据库字典示例

创建嵌套字典

```
1 #人员数据库
2 people = {
3     "zhang3":{
4         "staff_number":1001,
5         "address":"Road001"
6     },
7     "li4":{
8         "staff_number":1002,
9         "address":"Road002"
10    },
11    "wang5":{
12        "staff_number":1003,
13        "address":"Road003"
14    }
15 }
16
17 #查询数据库
18 name = input("Name: ")
19 request = input("Staff_number (S) or Address (A): ")
20
21 if name in people:
22     if request == "A":
23         print("{}'s {} is {}".format(name, "address", people[name]["address"]))
24     if request == "S":
25         print("{}'s {} is {}".format(name, "staff_number", people[name]["staff_number"]))
26
```

```
Name: zhang3
Staff_number (S) or Address (A): S
zhang3's staff_number is 1001
```

字符串格式设置用于字典format_map

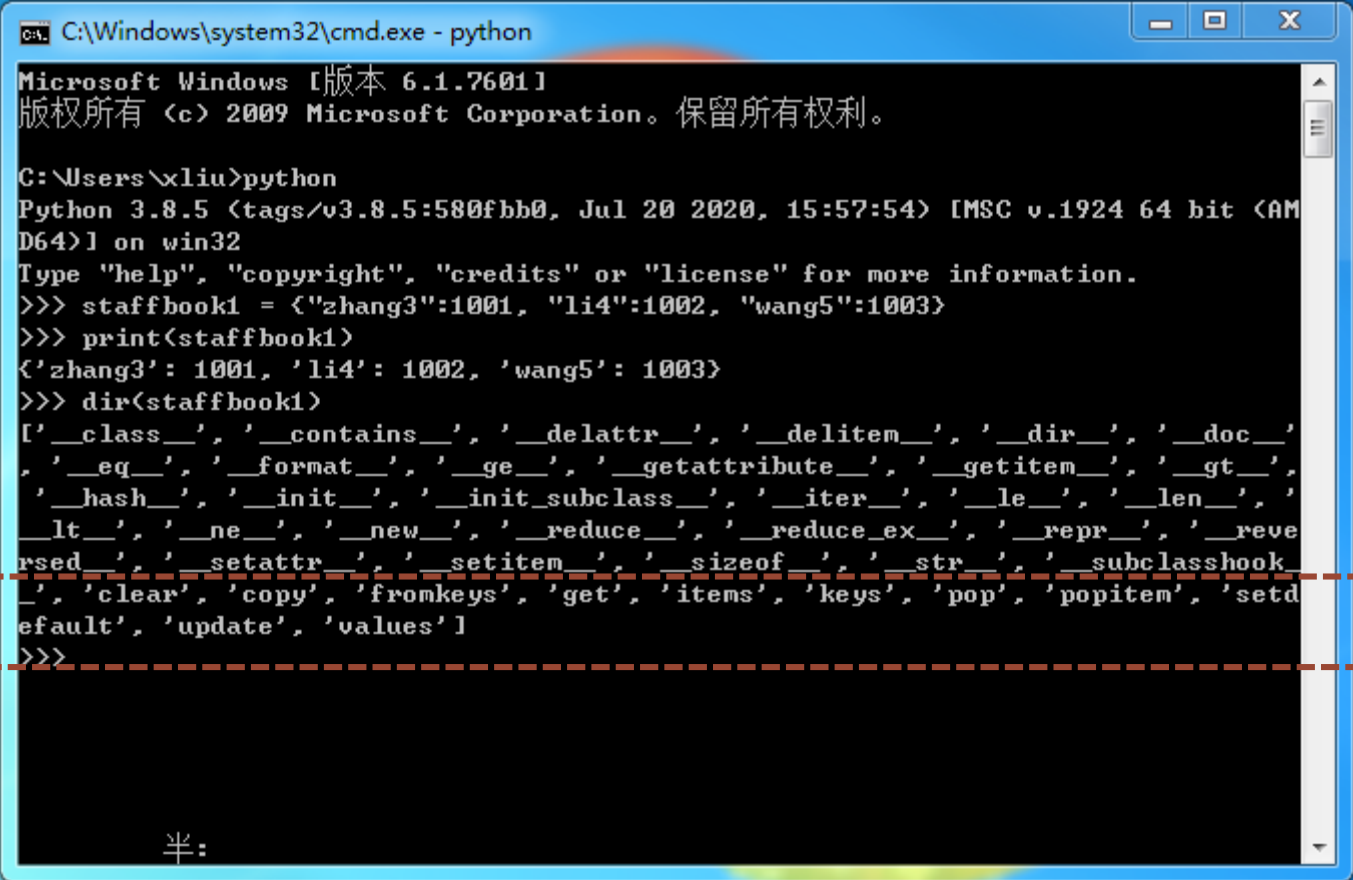
字符串格式设置用于字典format_map

```
1 web_template = '''<html>
2 <head><title>{title}</title></head>
3 <body>
4 <h1>{title}</h1>
5 <p>{text}</p>
6 </body>
7 </html>
8 '''
9 data = {"title": "My home page", "text": "Welcome to my home page!"}
10 print(web_template.format_map(data))
```

```
<html>
<head><title>My home page</title></head>
<body>
<h1>My home page</h1>
<p>Welcome to my home page!</p>
</body>
</html>
```

字典方法

□ 字典方法的使用与其他数据类型相同：字典名.方法()



```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

C:\Users\xliu>python
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
>>> print(staffbook1)
{'zhang3': 1001, 'li4': 1002, 'wang5': 1003}
>>> dir(staffbook1)
['_class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__',
 '__lt__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
 '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__',
 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault',
 'update', 'values']
>>>
```

clear: 删除所有字典项; copy: 复制字典; fromkeys: 从键列表创建字典

keys, values, items

字典方法keys, values, items

```
1 staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
2 #返回键或值或项的字典视图
3 print(staffbook1.keys())
4 print(staffbook1.values())
5 print(staffbook1.items())
6
7 #返回键或值或项的列表
8 print(list(staffbook1.keys()))
9 print(list(staffbook1.values()))
10 print(list(staffbook1.items()))
```

字典视图

```
dict_keys(['zhang3', 'li4', 'wang5'])
dict_values([1001, 1002, 1003])
dict_items([('zhang3', 1001), ('li4', 1002), ('wang5', 1003)])
['zhang3', 'li4', 'wang5']
[1001, 1002, 1003]
[('zhang3', 1001), ('li4', 1002), ('wang5', 1003)]
```

get

- 使用get访问不存在的键时，返回None，为访问字典提供了宽松的环境

字典方法get

```
1 staffbook1 = {"zhang3":1001, "li4":1002, "wang5":1003}
2 print(staffbook1.get("ma6"))
3 print(staffbook1.get("ma6", "ma6不在字典中"))
4 print(staffbook1["ma6"])
```

指定返回值

None
ma6不在字典中

```
KeyError                                Traceback (most recent call last)
<ipython-input-9-f91104fa4e07> in <module>
      2 print(staffbook1.get("ma6"))
      3 print(staffbook1.get("ma6", "ma6不在字典中"))
----> 4 print(staffbook1["ma6"])
```

KeyError: 'ma6'

小结

□ 列表：序列操作、方法、可变类型

□ 元组：序列操作、方法、不可变类型

□ 字典：映射操作、方法、可变类型

注意：Python中的浅拷贝&深拷贝，共享引用

下一节：语句控制