# 4.组合数据类型

## 4.1.列表

### 4.1.1.列表的表达

序列类型：内部元素有位置关系，能通过位置序号访问其中元素

列表是一个可以使用多种类型元素，支持元素的增、删、查、改操作的序列类型

ls = ["Python", 1989, **True**, {"version": 3.7}]

ls

['Python', 1989, True, {'version': 3.7}]

另一种产生方式：list(可迭代对象)

可迭代对象包括：字符串、元组、集合、range()等

字符串转列表

list("人工智能是未来的趋势")

['人', '工', '智', '能', '是', '未', '来', '的', '趋', '势']

元组转列表

list(("我", "们", "很", "像"))

['我', '们', '很', '像']

集合转列表

list({"李雷", "韩梅梅", "Jim", "Green"})

['Green', 'Jim', '李雷', '韩梅梅']

特殊的range()

**for** i **in** [0, 1, 2, 3, 4, 5]:

print(i)

0

1

2

3

4

5

**for** i **in** range(6):

print(i)

0

1

2

3

4

5

range(起始数字,中止数字,数字间隔)

如果起始数字缺省，默认为0

必须包含中止数字

数字间隔缺省，默认为1

**for** i **in** range(1, 11, 2):

print(i)

1

3

5

7

9

range()转列表

list(range(1, 11, 2))

[1, 3, 5, 7, 9]

### 4.1.2.列表的性质

列表的长度——len(列表)

ls = [1, 2, 3, 4, 5]

len(ls)

5

列表的索引——与同为序列类型的字符串完全相同

变量名[位置编号]

正向索引从0开始

反向索引从-1开始

cars = ["BYD", "BMW", "AUDI", "TOYOTA"]

print(cars[0])

print(cars[-1])

BYD

TOYOTA

列表的切片

变量名[开始位置：结束位置：切片间隔]

正向切片

cars = ["BYD", "BMW", "AUDI", "TOYOTA"]

print(cars[:3]) *# 前三个元素，开始位置缺省，默认为0；切片间隔缺省，默认为1*

['BYD', 'BMW', 'AUDI']

print(cars[1:4:2]) *# 第二个到第四个元素 前后索引差为2*

['BMW', 'TOYOTA']

print(cars[:]) *# 获取整个列表，结束位置缺省，默认取值到最后*

['BYD', 'BMW', 'AUDI', 'TOYOTA']

print(cars[-4:-2]) *# 获取前两个元素*

['BYD', 'BMW']

反向切片

cars = ["BYD", "BMW", "AUDI", "TOYOTA"]

print(cars[:-4:-1]) *# 开始位置缺省，默认为-1*

print(cars[::-1]) *# 获得反向列表*

['TOYOTA', 'AUDI', 'BMW']

['TOYOTA', 'AUDI', 'BMW', 'BYD']

### 4.1.3.列表的操作符

用 list1+lis2 的形式实现列表的拼接

a = [1, 2]

b = [3, 4]

a+b *# 该用法用的不多*

[1, 2, 3, 4]

用 n\*list 或 list\*n 实现列表的成倍复制

[0]\*10 *# 初始化列表的一种方式*

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

### 4.1.4.列表的操作方法

**增加元素**

在末尾增加元素——列表.append(待增元素)

languages = ["Python", "C++", "R"]

languages.append("Java")

languages

['Python', 'C++', 'R', 'Java']

在任意位置插入元素——列表.insert(位置编号，待增元素)  
 在位置编号相应元素前插入待增元素

languages.insert(1, "C")

languages

['Python', 'C', 'C++', 'R', 'Java']

在末尾整体并入另一列表——列表1.extend(列表2)

*# append 将列表2整体作为一个元素添加到列表1中*

languages.append(["Ruby", "PHP"])

languages

['Python', 'C', 'C++', 'R', 'Java', ['Ruby', 'PHP']]

*# extend 将待列表2内的元素逐个添加到列表1中*

languages = ['Python', 'C', 'C++', 'R', 'Java']

languages.extend(["Ruby", "PHP"])

languages

['Python', 'C', 'C++', 'R', 'Java', 'Ruby', 'PHP']

**删除元素**

删除列表i位置的元素  列表.pop(位置)

languages = ['Python', 'C', 'C++', 'R', 'Java']

languages.pop(1)

languages

['Python', 'C++', 'R', 'Java']

不写位置信息，默认删除最后一个元素

languages.pop()

languages

['Python', 'C++', 'R']

删除列表中的第一次出现的待删元素 列表.remove(待删元素)

languages = ['Python', 'C', 'R', 'C', 'Java']

languages.remove("C")

languages

['Python', 'R', 'C', 'Java']

languages = ['Python', 'C', 'R', 'C', 'Java']

**while** "C" **in** languages:

languages.remove("C")

languages

['Python', 'R', 'Java']

**查找元素**

列表中第一次出现待查元素的位置 列表.index(待查元素)

languages = ['Python', 'C', 'R','Java']

idx = languages.index("R")

idx

2

**修改元素**

通过"先索引后赋值"的方式，对元素进行修改 列表名[位置]=新值

languages = ['Python', 'C', 'R','Java']

languages[1] = "C++"

languages

['Python', 'C++', 'R', 'Java']

**列表的复制**

错误的方式

languages = ['Python', 'C', 'R','Java']

languages\_2 = languages

print(languages\_2)

['Python', 'C', 'R', 'Java']

languages.pop()

print(languages)

print(languages\_2)

['Python', 'C', 'R']

['Python', 'C', 'R']

正确的方式——浅拷贝

方法1：列表.copy()

languages = ['Python', 'C', 'R','Java']

languages\_2 = languages.copy()

languages.pop()

print(languages)

print(languages\_2)

['Python', 'C', 'R']

['Python', 'C', 'R', 'Java']

方法2：列表[ : ]

languages = ['Python', 'C', 'R','Java']

languages\_3 = languages[:]

languages.pop()

print(languages)

print(languages\_3)

['Python', 'C', 'R']

['Python', 'C', 'R', 'Java']

**列表的排序**

使用列表.sort()对列表进行永久排序

直接在列表上进行操作，无返回值

ls = [2, 5, 2, 8, 19, 3, 7]

ls.sort()

ls

[2, 2, 3, 5, 7, 8, 19]

递减排列

ls.sort(reverse = **True**)

ls

[19, 8, 7, 5, 3, 2, 2]

使用sorted(列表)对列表进行临时排序

原列表保持不变，返回排序后的列表

ls = [2, 5, 2, 8, 19, 3, 7]

ls\_2 = sorted(ls)

print(ls)

print(ls\_2)

[2, 5, 2, 8, 19, 3, 7]

[19, 8, 7, 5, 3, 2, 2]

sorted(ls, reverse = **True**)

[19, 8, 7, 5, 3, 2, 2]

**列表的翻转**

使用列表.reverse()对列表进行永久翻转

直接在列表上进行操作，无返回值

ls = [1, 2, 3, 4, 5]

print(ls[::-1])

ls

[5, 4, 3, 2, 1]

[1, 2, 3, 4, 5]

ls.reverse()

ls

[5, 4, 3, 2, 1]

**使用for循环对列表进行遍历**

ls = [1, 2, 3, 4, 5]

**for** i **in** ls:

print(i)

1

2

3

4

5

## 4.2.元组

### 4.2.1.元组的操作

元组是一个可以使用多种类型元素，一旦定义，内部元素不支持增、删和修改操作的序列类型

通俗的讲，可以将元组视作“不可变的列表”

除了不支持元素增加、元素删除、元素修改操作，其他操作与列表的操作完全一致

names = ("Peter", "Tom", "Mary")

### 4.2.2.元组的常见用处

打包与解包

**def** f1(x): *# 返回x的平方和立方*

**return** x\*\*2, x\*\*3 *# 实现打包返回*

print(f1(3))

print(type(f1(3))) *# 元组类型*

(9, 27)

<class 'tuple'>

a, b = f1(3) *# 实现解包赋值*

print(a)

print(b)

9

27

numbers = [201901, 201902, 201903]

name = ["小明", "小红", "小强"]

list(zip(numbers,name))

[(201901, '小明'), (201902, '小红'), (201903, '小强')]

**for** number,name **in** zip(numbers,name): *# 每次取到一个元组，立刻进行解包赋值*

print(number, name)

201901 小明

201902 小红

201903 小强

## 4.3.字典

### 4.3.1.字典的创建

映射类型： 通过“键”-“值”的映射实现数据存储和查找

常规的字典是无序的

字典键的要求

1、字典的键不能重复

students = {201901: '小明', 201902: '小红', 201903: '小强'}

students

{201901: '小红', 201903: '小强'}

2、字典的键必须是不可变类型，如果键可变，就找不到对应存储的值了

不可变类型：数字、字符串、元组。  一旦确定，它自己就是它自己，变了就不是它了。

可变类型：列表、字典、集合。  一旦确定，还可以随意增删改。

d1 = {1: 3}

d2 = {"s": 3}

d3 = {(1,2,3): 3}

d = {[1, 2]: 3}

**TypeError**: unhashable type: 'list'

d = {{1:2}: 3}

**TypeError**: unhashable type: 'dict'

d = {{1, 2}: 3}

**TypeError**: unhashable type: 'set'

### 4.3.2.字典的性质

字典的长度——键值对的个数

students = {201901: '小明', 201902: '小红', 201903: '小强'}

len(students)

3

字典的索引

通过 字典[键] 的形式来获取对应的值

students = {201901: '小明', 201902: '小红', 201903: '小强'}

students[201902]

'小红'

### 4.3.3.字典的操作方法

**增加键值对**

变量名[新键] = 新值

students = {201901: '小明', 201902: '小红', 201903: '小强'}

students[201904] = "小雪"

students

{201901: '小明', 201902: '小红', 201903: '小强', 201904: '小雪'}

删除键值对

通过del 变量名[待删除键]

students = {201901: '小明', 201902: '小红', 201903: '小强'}

**del** students[201903]

students

{201901: '小明', 201902: '小红'}

通过变量名.pop(待删除键)

students = {201901: '小明', 201902: '小红', 201903: '小强'}

value = students.pop(201903) *# 删除键值对，同时获得删除键值对的值*

print(value)

print(students)

小强

{201901: '小明', 201902: '小红'}

变量名.popitem() 随机删除一个键值对，并以元组返回删除键值对

students = {201901: '小明', 201902: '小红', 201903: '小强'}

key, value = students.popitem()

print(key, value)

print(students)

201903 小强

{201901: '小明', 201902: '小红'}

**修改值**

通过先索引后赋值的方式对相应的值进行修改

students = {201901: '小明', 201902: '小红', 201903: '小强'}

students[201902] = "小雪"

students

{201901: '小明', 201902: '小雪', 201903: '小强'}

**d.get( )方法**

d.get(key,default) 从字典d中获取键key对应的值，如果没有这个键，则返回default

小例子：统计"牛奶奶找刘奶奶买牛奶"中字符的出现频率

s = "牛奶奶找刘奶奶买牛奶"

d = {}

print(d)

**for** i **in** s:

d[i] = d.get(i, 0)+1

print(d)

*# print(d)*

{}

{'牛': 1}

{'牛': 1, '奶': 1}

{'牛': 1, '奶': 2}

{'牛': 1, '奶': 2, '找': 1}

{'牛': 1, '奶': 2, '找': 1, '刘': 1}

{'牛': 1, '奶': 3, '找': 1, '刘': 1}

{'牛': 1, '奶': 4, '找': 1, '刘': 1}

{'牛': 1, '奶': 4, '找': 1, '刘': 1, '买': 1}

{'牛': 2, '奶': 4, '找': 1, '刘': 1, '买': 1}

{'牛': 2, '奶': 5, '找': 1, '刘': 1, '买': 1}

**d.keys( ) d.values( )方法**

students = {201901: '小明', 201902: '小红', 201903: '小强'}

print(list(students.keys()))

print(list(students.values()))

[201901, 201902, 201903]

['小明', '小红', '小强']

**d.items( )方法及字典的遍历**

print(list(students.items()))

**for** k, v **in** students.items():

print(k, v)

[(201901, '小明'), (201902, '小红'), (201903, '小强')]

201901 小明

201902 小红

201903 小强

## 4.4.集合

### 4.4.1.集合的创建

集合是一系列互不相等元素的无序集合

元素必须是不可变类型：数字，字符串或元组，可视作字典的键

可以看做是没有值，或者值为None的字典

students = {"小明", "小红", "小强", "小明"} *#可用于去重*

students

{'小强', '小明', '小红'}

### 4.4.2.集合的运算

小例子 通过集合进行交集并集的运算

Chinese\_A = {"刘德华", "张学友", "张曼玉", "钟楚红", "古天乐", "林青霞"}

Chinese\_A

{'刘德华', '古天乐', '张学友', '张曼玉', '林青霞', '钟楚红'}

Math\_A = {"林青霞", "郭富城", "王祖贤", "刘德华", "张曼玉", "黎明"}

Math\_A

{'刘德华', '张曼玉', '林青霞', '王祖贤', '郭富城', '黎明'}

交集：语文和数学两门均为A的学员

S & T 返回一个新集合，包括同时在集合S和T中的元素

Chinese\_A & Math\_A

{'刘德华', '张曼玉', '林青霞'}

并集：语文或数学至少一门为A的学员

S | T 返回一个新集合，包括集合S和T中的所有元素

Chinese\_A | Math\_A

{'刘德华', '古天乐', '张学友', '张曼玉', '林青霞', '王祖贤', '郭富城', '钟楚红', '黎明'}

异或：语文数学只有一门为A的学员

S ^ T 返回一个新集合，包括集合S和T中的非共同元素

Chinese\_A ^ Math\_A

{'古天乐', '张学友', '王祖贤', '郭富城', '钟楚红', '黎明'}

差集：语文为A，数学不为A的学员

S - T 返回一个新集合，包括在集合S但不在集合T中的元素

Chinese\_A - Math\_A

{'古天乐', '张学友', '钟楚红'}

差集：数学为A，语文不为A的学员

Math\_A - Chinese\_A

{'王祖贤', '郭富城', '黎明'}

### 4.4.3.集合的操作方法

**增加元素**——S.add(x)

stars = {"刘德华", "张学友", "张曼玉"}

stars.add("王祖贤")

stars

{'刘德华', '张学友', '张曼玉', '王祖贤'}

**移除元素**——S.remove(x)

stars.remove("王祖贤")

stars

{'刘德华', '张学友', '张曼玉'}

**集合的长度**——len(S)

len(stars)

3

**集合的遍历**——借助for循环

**for** star **in** stars:

print(star)

张学友

张曼玉

刘德华

## 4.5.作业练习

列表：

1、产生由100以内偶数构成的列表。

2、Is=["a","b","c","d","e"]

\*分别用正向索引和反向索引获得元素"c"

\*用正向切片获得["b","d"]

\*用反向切片实现Is的反向列表

3、cars=["BYD","GEELY"]，对列表cars依次执行以下列表操作：

\*在列表结尾增加元素"TOYOTA"；

\*在"BYD"和"GEELY"直接增加元素"CHERY"；

\*在列表结尾增加["BMW","BENZ"]中的元素；

\*删除列表最后一个元素和第四个"元素；

\*删除元素"GEELY"；

\*查找元素"BMW"在列表中的位置索引；

\*将元素"CHERY"修改为"QQ"；

\*通过两种方法对列表进行复制；

\*对列表分布进行永久排序和临时排序；

\*对列表进行翻转；

\*对列表进行遍历，并按下列格式进行输出：

My first car is BYD。

元祖：

4、元组为什么被称为"不可变的列表"；

5、foods=["bread","fish","potato"]

prices=[2.4,9.8,0.9].

按照以下格式进行输出：

The price of bread is 2.4。

字典：

6、判断以下字典的定义是否合法，如果不合法，指出原因。

A{(1,2):3} B{[1,2]:3} C{"price":2.4} D{1:"fisrt"}

7、favorite\_fruits={"Judy":"watermelon","Jen":"banana","Sarah":"orange"}

请依次执行下列操作

\*获得字典favorite\_fruits的长度；

\*获得"Jen"最爱的水果名；

\*增加"Tom"最爱的水果"peach","Bob"最爱的水果"Tomato"；

\*删除"Judy"及其最爱的水果名；

\*随机删除一个键值对，并捕获被删除的值；

\*将"Sarah"的最爱修改为"watermelon"；

\*遍历字典favorite\_fruits，并按下列格式进行输出：

Sarah'favorite fruit is watermelon

8、统计下列绕口令中字符出现的频次：

八百标兵奔北坡，

北坡八百炮兵炮。

标兵怕碰炮兵炮，

炮兵怕把标兵碰。

集合：

9、vegetables={"tomato", "cabbage", "cucumber", "meat"}

fruits={"banana", "orange", "tomato", "cucumber"}

\*将集合vegetables中的"meat"删除；

\*将"eggplant"加入到集合vegetables中；

\*两个集合中，哪些即属于蔬菜，又属于水果？

\*两个集合中，所有的蔬菜和水果有哪些？

\*两个集合中，哪些只属于蔬菜或者只属于水果？

\*两个集合中，哪些只属于蔬菜？

答案：

1.

ans = []

**for** i **in** range(100):

**if** i % 2 == 0:

ans.append(i)

print(ans)

[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98]

2.

ls = ["a", "b", "c", "d", "e"]

ls[2]

'c'

ls[-3]

'c'

ls[1::2]

['b', 'd']

ls[::-1]

['e', 'd', 'c', 'b', 'a']

3.

cars = ["BYD", "GEELY"]

cars.append("TOYOTA")

cars

['BYD', 'GEELY', 'TOYOTA']

cars.insert(1, "CHERY")

cars

['BYD', 'CHERY', 'GEELY', 'TOYOTA']

cars.extend(["BMW", "BENZ"])

cars

['BYD', 'CHERY', 'GEELY', 'TOYOTA', 'BMW', 'BENZ']

cars.pop()

cars.remove(cars[3])

cars

['BYD', 'CHERY', 'GEELY', 'BMW']

cars.remove("GEELY")

cars

['BYD', 'CHERY', 'BMW']

cars.index("BMW")

2

cars[1] = "QQ"

cars

['BYD', 'QQ', 'BMW']

cars1 = cars.copy()

cars

['BYD', 'QQ', 'BMW']

cars2 = cars[:]

cars2

['BYD', 'QQ', 'BMW']

sorted(cars)

['BMW', 'BYD', 'QQ']

cars

['BYD', 'QQ', 'BMW']

cars.sort()

cars

['BMW', 'BYD', 'QQ']

cars[::-1]

['QQ', 'BYD', 'BMW']

**for** i **in** cars:

print("My first car is **{}**".format(i))

My first car is BMW

My first car is BYD

My first car is QQ

4. 因为元祖的元素无法改变，但是其他操作与列表一致。

5.

food = ["bread", "fish", "potato"]

prices = [2.4, 9.8, 0.9]

**for** food, price **in** zip(food, prices):

print("The price of **{}** is **{}**".format(food, price))

The price of bread is 2.4

The price of fish is 9.8

The price of potato is 0.9

1. B不合法，因为key只能是不可变的

7.

favorite\_fruits = {"Judy": "watermelon", "Jen": "banana", "Sarah": "orange" }

len(favorite\_fruits)

3

favorite\_fruits["Jen"]

'banana'

favorite\_fruits["Tom"] = "peach"

favorite\_fruits

{'Judy': 'watermelon', 'Jen': 'banana', 'Sarah': 'orange', 'Tom': 'peach'}

favorite\_fruits["Bob"] = "Tomato"

favorite\_fruits

{'Judy': 'watermelon',

'Jen': 'banana',

'Sarah': 'orange',

'Tom': 'peach',

'Bob': 'Tomato'}

favorite\_fruits.pop("Judy")

favorite\_fruits

{'Jen': 'banana', 'Sarah': 'orange', 'Tom': 'peach', 'Bob': 'Tomato'}

v = favorite\_fruits.popitem()

v

favorite\_fruits

{'Jen': 'banana', 'Sarah': 'orange', 'Tom': 'peach'}

favorite\_fruits["Sarah"] = "watermelon"

favorite\_fruits

{'Jen': 'banana', 'Sarah': 'watermelon', 'Tom': 'peach'}

**for** k, v **in** favorite\_fruits.items():

print("**{}**’favorite fruit is**{}**".format(k, v))

Jen’favorite fruit isbanana

Sarah’favorite fruit iswatermelon

Tom’favorite fruit ispeach

8.

s = "八百标兵奔北坡，北坡八百炮兵炮。标兵怕碰炮兵炮，炮兵怕把标兵碰。"

d = {}

**for** i **in** s:

d[i] = d.get(i, 0) + 1

d

{'八': 2,

'百': 2,

'标': 3,

'兵': 6,

'奔': 1,

'北': 2,

'坡': 2,

'，': 2,

'炮': 5,

'。': 2,

'怕': 2,

'碰': 2,

'把': 1}

9.

vegetables = {"tomato", "cabbage", "cucumber", "meat"}

fruits = {"banana", "orange", "tomato", "cucumber"}

vegetables.remove("meat")

vegetables

{'cabbage', 'cucumber', 'tomato'}

vegetables.add("eggplant")

vegetables

{'cabbage', 'cucumber', 'eggplant', 'tomato'}

vegetables & fruits

{'cucumber', 'tomato'}

vegetables | fruits

{'banana', 'cabbage', 'cucumber', 'eggplant', 'orange', 'tomato'}

vegetables ^ fruits

{'banana', 'cabbage', 'eggplant', 'orange'}

vegetables - fruits

{'cabbage', 'eggplant'}