Python 语言程序设计: 列表与元组

李宽
likuan@dgut.edu.cn

东莞理工学院

2019.10

目录

目录

集合类型 1/2

集合

由一个或多个确定的元素所构成的整体。

与数学概念一致：多个元素的无序组合
若 x 是集合 A 的元素，则记作 $x \in A$
集合中的元素有三个特征：

- 确定性：集合中的元素必须是确定的
- 互异性：集合中的元素互不相同
- 无序性：集合中的元素没有先后之分[1]

[1]集合 {3,4,5} 和 {3,5,4} 是同一个集合

集合类型 2/2

集合元素不能是可变数据结构

- 可变数据类型：
  列表 list、字典 dict, 集合 set
- 不可变数据类型：
  整型 int、浮点型 float、字符串型 string 和元组 tuple

目录

建立集合

集合以大括号 {} 表示, 元素间用逗号分割.
建立集合类型用 {} 或 set(), 建立空集合必须用 set()
{} 默认建的是空字典
create-set.py:

```
# 创建集合

# 方法 1 使用 set() 建立空集合
empty_set1=set()

# 方法 2 创建时指定元素，元素间用逗号隔开
set2 = {'huawei', 'xman', 19.87, 2017}
print(set2)

set3=set("12345") #set() 函数将字符串转化为集合
set4=set([1,2,3,4,5]) #set() 函数将列表转化为集合
set5=set((1,2,3,4,5)) #set() 函数将元组转化为集合

print(set3)
print(set4)
print(set5)
```

集合: 自动过滤

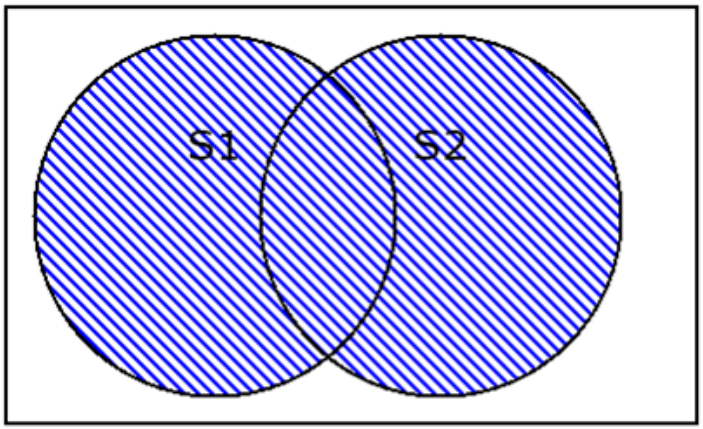
自动过滤：set 中的元素是无序的，并且重复元素在 set 中自动被过滤

remove-repeat.py

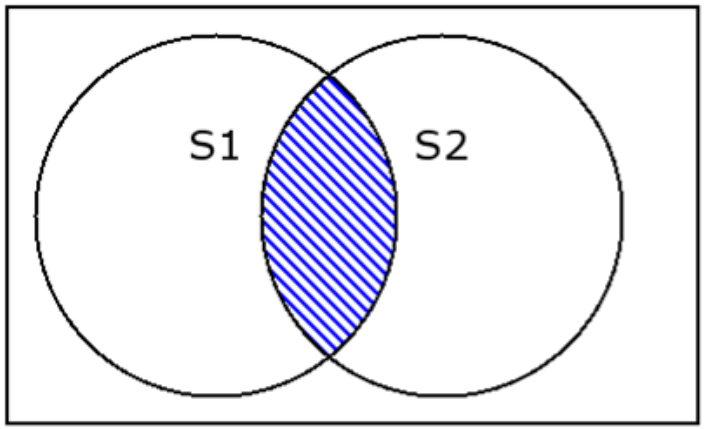
```
1  # 数据去重
2
3  # 建立列表，故意设置某些重复元素
4  ls1 = [1,2,3,"a","b",3,2,1,"c","a"]
5
6  print("原始列表:", ls1)
7
8  # 通过 set 函数将列表转换成集合
9  st1 = set(ls1)
10 print("集合:", st1)
11
12 # 还可通过 list 函数将集合转成列表
13 ls2 = list(st1)
14 print("列表:", ls2)
```

目录

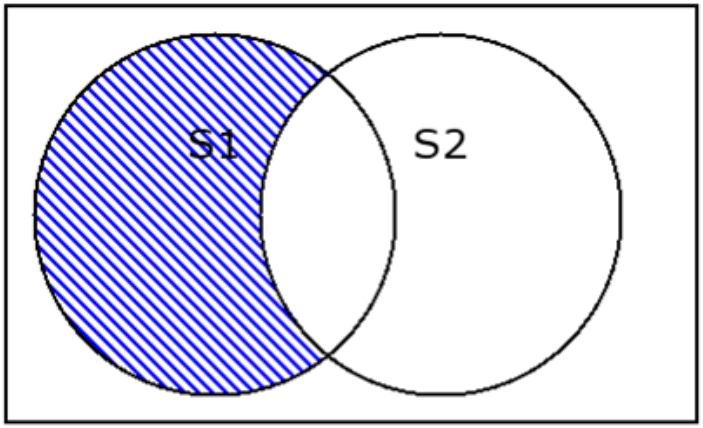
集合间的操作 1/4



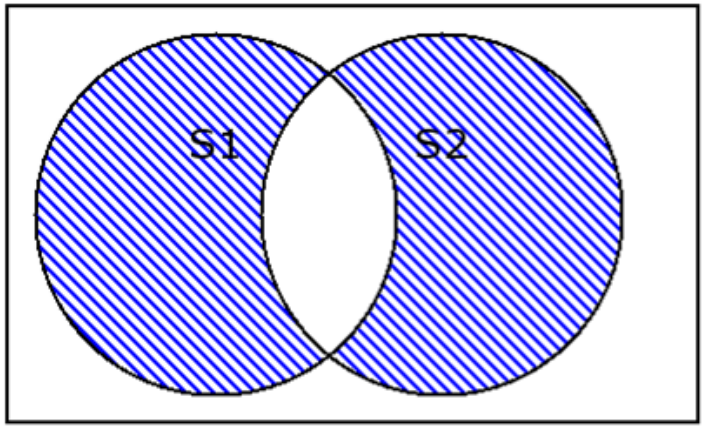
S1 | S2, 并, union



S1 & S2, 交, intersect



S1 - S2, 差, difference



S1 ^ S2, 补（对称差）, Symmetric difference

集合间的操作 2/4

集合间操作

操作符及应用	描述
S | T 或 S.union(T)	并
S - T 或 S.difference(T)	差
S & T 或 S.intersection(T)	交
S ∧ T	补 或 S.symmetric_diffence(T)
S <= T 或 S < T	返回 True/False，判断 S 和 T 的子集关系
S >= T 或 S > T	返回 True/False，判断 S 和 T 的包含关系

备注：前四种操作均返回一个新集合，不会影响 S 和 T

集合间操作 3/4

set-ops.py:

```
# 集合间的交并差补操作
fib=set((1,1,2,3,5,8,13))
prime=set((2,3,5,7,11,13))

print( fib | prime ) # 并
print(fib.union(prime)) # 并

print( fib & prime ) # 交
print(fib.intersection(prime)) # 交

print( fib - prime ) # 差
print(fib.difference(prime)) # 差

print( fib ^ prime ) # 补
print(fib.symmetric_difference(prime)) # 补
```

集合间的操作 4/4

增强操作符：

函数及方法	描述
S |= T	更新集合 S，包括在集合 S 和 T 中的所有元素
S -= T	更新集合 S，包括在集合 S 但不在集合 T 中的元素
S &= T	更新集合 S，包括同时在集合 S 和 T 中的元素
S ∧= T	更新集合 S，包括集合 S 和 T 中的非相同元素

目录

集合操作函数及方法 1/3

操作函数	描述
len(S)	返回集合 S 的元素个数
max(S)	返回集合 S 的元素最大值 (前提是可比较)
min(S)	返回集合 S 的元素最小值 (前提是可比较)
x in S	如果 x 在集合 S 中, 返回 True, 否则, 返回 False
x not in S	如果 x 不在集合 S 中, 返回 True, 否则, 返回 False
set(x)	将其他变量转变为集合类型

集合操作函数及方法 2/3

集合类型方法	描述
S.add(x)	如果元素x 不在集合 S 中，将 x 增加到 S
S.update(T)	合并集合T 中的元素到集合 S 中[2]
S.discard(x)	移除 S 中的元素 x，如果 x 不在 S 中，不报错
S.remove(x)	移除 S 中元素 x，如果 x 不在 S 中，产生 KeyError 异常
S.clear()	移除 S 中所有元素
S.pop()	随机返回 S 的一个元素，更新 S，[3]
S.copy()	返回集合 S 的一个副本

[2]等价于 |=

[3]若 S 为空产生 KeyError 异常

集合操作函数及方法 3/3

set-funcs.py:

```
# 集合间的交并差补操作
s={1,2,3}

s.add(4) # 增加 1 个新的元素
print(s)

s.update({3,4,5}) # 等价于 s |= {3,4,5}
print(s)

s.discard(5) # 移除元素 5，如果没有，不报错
print(s)

s.remove(5) # 移除元素 5，如果没有，报错 KeyError
```

目录

集合类型应用场景: 包含关系比较

```
1  # 包含关系比较
2
3  # 创建集合
4  st1={1,2,(8,9,10),"莞工"}
5
6  print("1:", 1 in st1) # 判断元素 1 是否在集合中
7  print("2:", {1} in st1) # 判断元素 {1} 是否在集合中
8
9  #(8,9) 和 (8,9,10) 是两个不同的元组
10 print("3:", (8,9) in st1) # 判断元素 (8,9) 是否在集合中
11 print("4:", (8,9,10) in st1) # 判断元素 (8,9,10) 是否在集合中
12
13 print("5:", {1,2} <= st1)
14
15 # 错误，只有集合可比较，表示包含关系
16 #print((1,2) <= st1)
```

集合类型应用场景: 数据去重 1/4

充分利用集合类型所有元素无重复这个性质, 对比列表与集合

rand-list.py:

```
# 集合去重
import random # 导入 random 库

list1 = [] # 建立空列表

while True:
    num = random.randint(1,100) # 在 [1,100] 之间取随机整数
    for e in list1: # 对列表来说, 需要先判断 num 是否存在
        if e == num: # 如果相等, 说明已存在
            break;
    else: # 不被 break 跳出时执行
        list1.append(num)

    if len(list1) >= 10: # 取够 10 个退出
        break

print(list1)
```

集合类型应用场景: 数据去重 2/4

rand-list-rev.py:

```
# 集合去重
import random # 导入 random 库

list1 = [] # 建立空列表

while True:
    num = random.randint(1,100) # 在 [1,100] 之间取随机整数

    if num not in list1: # 直接用 not in 判断 num 是否在 list1 中
        list1.append(num)

        if len(list1) >= 10: # 取够 10 个退出
            break

print(list1)
```

集合类型应用场景: 数据去重 3/3

rand-set.py:

```
1  # 集合去重
2  import random
3
4  set1 = set()  # 建立空集合
5
6  while True:
7      num = random.randint(1,100) # 返回 [1,100] 之间的随机数
8      set1.add(num) # 直接添加，自动过滤重复值
9      if len(set1) >= 10: # 随机数个数满足要求时，退出循环
10         break
11
12 print(set1)
```

集合类型应用场景: 数据去重 4/4

rand.py:

```
1 import random # 导入 random 库
2
3 # 直接调用 random 库的 sample 函数
4 # 两个参数: 1st 序列或集合 2nd 个数
5 # 意义: 从某序列或集合中取若干个不重复的值
6 l = random.sample(range(1,101),10)
7 print(l)
```

集合类型应用场景: 集合遍历

```
1 for <item> in <set>:
2     ...
```

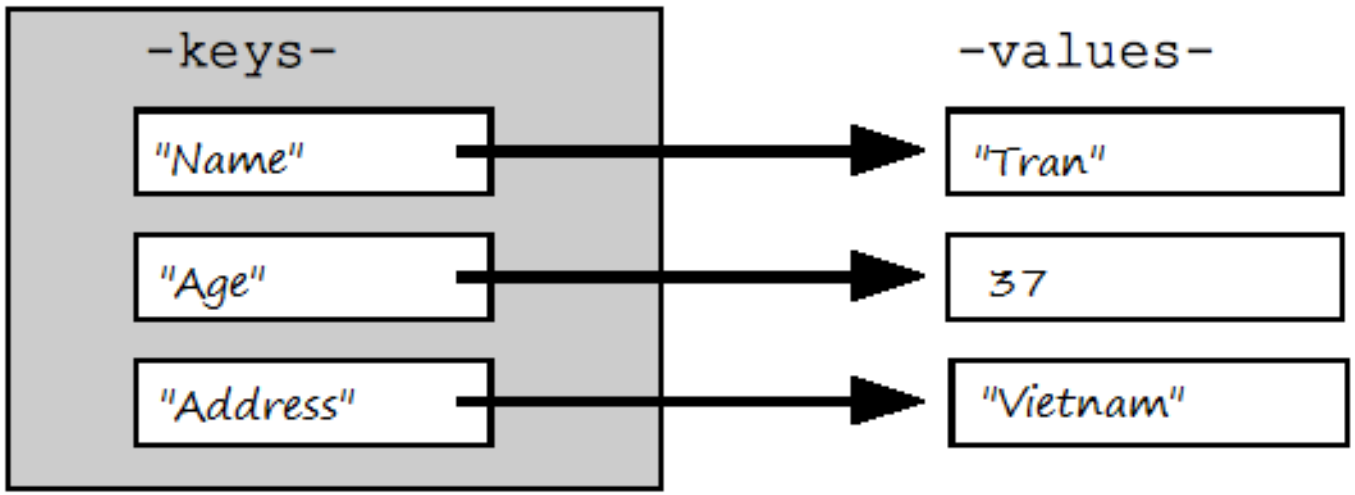
目录

字典类型 1/2

字典类型

包含若干"键: 值" 元素的无序可变序列

关键词：key(键):value(索引) 对, 表示" 映射" 关系



字典类型 2/2

性质：

- 键是唯一的（不可重复）
  可以是字符串、数字（int、float.....）、元组等不可变数据类型
  不可是列表, 集合或字典等可变数据类型
- 值可重复，可以是可变类型，也可是不可变类型
- 不维护元素的先后顺序，关注在键和值的对应关系[4]

对比:

- 有序序列类型 (列表, 元组, 字符串):
  由 Python 内部维护 0...n(正向) 或-1,-2...(逆向) 作数据的默认索引
- 字典类型则
  用户为数据自定义索引

字典类型中键 (key) 是数据索引的扩展

[4]如要维护字典中元素的先后关系可使用 collections 的 OrderedDict 类

目录

字典类型: 创建

字典是键值对的集合，键值对之间无序, 以逗号分隔
采用 {} 和 dict() 创建，键值对用冒号表示

```
{<键1>:<值1>, <键2>:<值2>, <键3>:<值3>, ... ,<键n>:<值n>}
# 大括号，且键值对之间用冒号（:）隔开
phones = {'Jack':'0571','James':'7856','Paul':'2364'}
```

目录

字典类型: 访问元素

dict-elements.py:

```
# 访问字典中的元素

# 创建字典 aDict
aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

print(aDict['age'])  # 指定的" 键" 存在, 返回对应的" 值"
# 指定的" 键" 不存在, 抛出 KeyError 异常
#print(aDict['address'])

# 如果字典中存在该" 键" 则返回对应的" 值"
print(aDict.get('age'))
# 如果字典中不存在" 键" 则返回指定的默认值
print(aDict.get('address','莞工'))
```

目录

字典类型: 添加删除元素 1/4

除访问字典中的元素外，还可修改字典中的元素或增加元素

- 若该键存在，修改该键对应的值
- 若该键不存在，增加一个新的键值对 (添加一个新元素)

字典类型: 添加删除元素 2/4

```
# 字典中元素的增删改

# 访问字典中的元素
aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

aDict['age']=40 # 对应键值存在时，修改元素的值
aDict['address']="莞工" # 对应键值不存在时，添加新元素
print(aDict)

bDict = {"age":38, 7:"xyz"} # 定义另外 1 个字典
# 将另一个字典的所有的键值对一次性全部添加到当前字典对象中
aDict.update(bDict)
print(aDict)

x = aDict.setdefault("age",36)
print(x)
print(aDict)
```

字典类型：添加删除元素 3/4

```
D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D
```

返回指定键对应的值
如果字典中不存在该键，就添加一个新元素并设置该键对应的值

```
# 字典中 setdefault 用法

aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

# 在键值"age" 存在的情况下，设置的 36 不起作用
x = aDict.setdefault("age",36)
print(x) # 打印返回值
print(aDict) # 打印字典

# 不存在对应的键值，新增元素
y = aDict.setdefault("nationality","China")
print(y) # 打印返回值
print(aDict) # 打印字典
```

字典类型：添加删除元素 4/4

```
# 字典中 pop,popitem 用法

aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

del aDict["age"] # 删除指定元素
print(aDict)

# 弹出指定键对应的元素，也支持默认值，如 pop("sex","male")
aDict.pop("sex")
print(aDict)

# 无参数，弹出一个元素（随机），故每次结果可能不同
aDict.popitem()
print(aDict)
```

补充：del aDict["age"] 甚至 del aDict

目录

字典类型：遍历 1/2

对字典对象进行迭代或遍历时默认是遍历字典的" 键"

```
# 字典遍历

aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

for item in aDict:    # 遍历的是键
    print(item, end=" : ")
    print(aDict[item], end=" , ")
print()

# 下列用法与上面等价
for item in aDict.keys():    # 遍历的是键
    print(item, end=" : ")
    print(aDict[item], end=" , ")
print()
```

字典类型：遍历 2/2

遍历字典的值:

```
# 字典遍历

aDict = {'age':39, 'score': [98,97],
         'name': 'Dong','sex':'male'}

#values() 以列表形式返回字典中的所有值
for item in aDict.values():
    print(item, end=" ")
print()

# 下列用法与上面等价
#items() 以列表形式将键和值二元元组 tuple 返回
for item in aDict.items():
    print(item[0], end=" : ")
    print(item[1], end=" , ")
print()
```

目录

字典类型函数及方法

函数及方法	描述
k in d[k]	判断键是否在字典中, 在返回 True, 否则返回 False
k not in d[k]	判断键是否在字典中, 不在返回 True, 否则返回 False
d.clear()	删除所有键值对
d.copy()	返回字典的浅复制[5]
len(d)	返回字典中键值对的个数

[5]后续课程会介绍关于深浅复制的区别

目录

应用场景: 字典存储电话号码 1/2

姓名	电话
赵云	133 0928 3335
乔丹	188 0731 7878
C 罗	150 9348 8129
韦德	192 8293 7665
...	...

问题: 如何合理组织存储类似存在关系的结构?

应用场景: 字典存储电话号码 2/2

```
1  # 字典存储电话号码
2  contacts = {"赵云":"133 0928 3335",
3              "乔丹":"188 0731 7878",
4              "C 罗":"150 9348 8129",
5              "韦德":"192 8293 7665"}
6
7  name = input("输入姓名：")
8
9  if name in contacts:    #in 默认是检查键 (key) 是否在字典中
10     print(name, ":", contacts[name])
11 else:
12     print("未找到" + name)
```

字典类型: 总结

1: 字典是什么?
键值对的集合 + 一系列操作

2: 什么时候用字典？

- 存储一组键-值对。
- 频繁地用键查值。

3: 字典的用法:

- 定义字典 dict = {键: 值, ···}
- 用键查值 dict[键
- 检查键在字典中吗：键 in dict

建议预习内容

第一部分 (序列结构部分收尾, 疑难点):

- Python 中一切皆对象, 可变类型 & 不可变类型
- 深复制 & 浅复制
- 列表推导式
- 生成器推导式
- 序列解包

第二部分 (函数):

- 函数的定义及调用
- 函数的参数
- 变量的作用域