合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

Python
数据分析及可视化-2

组合数据操作

HFUT计算机基础

2021.3

本章学习目标

- 熟练掌握列表对象及其常用方法
- 熟练掌握列表推导式语法和应用
- 熟练掌握切片操作
- 熟练掌握序列解包的语法和应用
- 熟练掌握生成器表达式的语法和应用
- 理解元组和列表的不同
- 熟练掌握字典对象及其常用操作
- 熟练掌握集合对象及其常用操作
- 熟练应用内置对象解决实际问题

组合数据对象

序列类型

序列类型是一维元素向量，元素之间存在先后关系，通过序号访问。

当需要访问序列中某特定值时，只需要通过下标标出即可。

$$\sum_{i=0}^{n-1} S_i$$

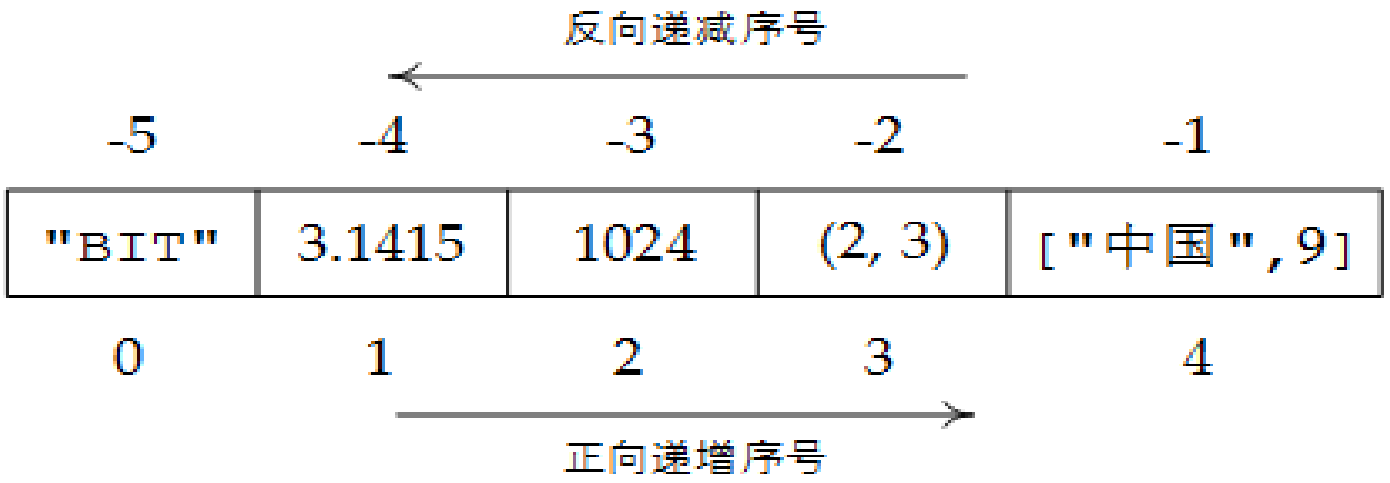
由于元素之间存在顺序关系，所以序列中可以存在相同数值但位置不同的元素。序列类型支持成员关系操作符(in)、长度计算函数(len())、分片([ ])，元素本身也可以是序列类型。

组合数据对象

序列类型

Python语言中有很多数据类型都是序列类型，其中比较重要的是：str（字符串）、tuple（元组）和list（列表）。

- 元组是包含0个或多个数据项的不可变序列类型。元组生成后是固定的，其中任何数据项不能替换或删除。
- 列表则是一个可以修改数据项的序列类型，使用也最灵活



组合数据对象

序列类型

序列类型常用操作符和函数

操作符	描述
x in s	如果x是s的元素，返回True，否则返回False
x not in s	如果x不是s的元素，返回True，否则返回False
s + t	连接s和t
s * n 或 n * s	将序列s复制n次
s[i]	索引，返回序列的第i个元素
s[i: j]	分片，返回包含序列s第i到j个元素的子序列（不包含第j个元素）
s[i: j: k]	步骤分片，返回包含序列s第i到j个元素以k为步数的子序列
len(s)	序列s的元素个数（长度）
min(s)	序列s中的最小元素
max(s)	序列s中的最大元素
s.index(x[, i[, j]])	序列s中从i开始到j位置中第一次出现元素x的位置
s.count(x)	序列s中出现x的总次数

列表操作

函数或方法	描述
ls[i] = x	替换列表ls第i数据项为x
ls[i: j] = lt	用列表lt替换列表ls中第i到j项数据（不含第j项，下同）
ls[i: j: k] = lt	用列表lt替换列表ls中第i到j以k为步的数据
del ls[i: j]	删除列表ls第i到j项数据，等价于ls[i: j]=[]
del ls[i: j: k]	删除列表ls第i到j以k为步的数据
ls += lt或ls.extend(lt)	将列表lt元素增加到列表ls中
ls *= n	更新列表ls，其元素重复n次
ls.append(x)	在列表ls最后增加一个元素x
ls.clear()	删除ls中所有元素
ls.copy()	生成一个新列表，复制ls中所有元素
ls.insert(i, x)	在列表ls第i位置增加元素x
ls.pop(i)	将列表ls中第i项元素取出并删除该元素
ls.remove(x)	将列表中出现的第一个元素x删除
ls.reverse()	列表ls中元素反转

列表

列表的所有元素放在一对方括号中，相邻元素之间使用逗号分隔。在Python中，同一个列表中元素的数据类型可以各不相同，可以同时包含整数、实数、复数、字符串等基本类型的元素，也可以包含列表、元组、字典、集合、函数或其他任意对象。一对空的方括号表示空列表。

创建列表

```
>>> lst=list()                          # 创建空列表
>>> list((3, 5, 7, 9, 11))             # 将元组转换为列表
[3, 5, 7, 9, 11]
>>> list(range(1, 10, 2))               # 将range对象转换为列表
[1, 3, 5, 7, 9]
>>> list(map(str, range(10)))           # 将map对象转换为列表
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
>>> list(zip('abcd', [1,2,3,4])) # 将zip对象转换为列表
[('a', 1), ('b', 2), ('c', 3), ('d', 4)]
```

创建列表

```
>>> list(enumerate('Python'))                # 将enumerate对象转换为列表
[(0, 'P'), (1, 'y'), (2, 't'), (3, 'h'), (4, 'o'), (5, 'n')]
>>> list(filter(str.isdigit, 'a1b2c3d456'))    # 将filter对象转换为列表
['1', '2', '3', '4', '5', '6']
>>> list('hello world')              # 将字符串转换为列表，每个字符转换为列中的一个元素
['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
>>> list({3, 7, 5})                  # 将集合转换为列表，集合中的元素是无序的
[3, 5, 7]
>>> x = [1, 2, 3]
>>> del x                            # 删除列表对象
>>> x                                # 对象删除后无法再访问，抛出异常
NameError: name 'x' is not defined
```

列表操作

与整数和字符串不同，列表要处理一组数据，因此，列表必须通过显式的数据赋值才能生成，简单将一个列表赋值给另一个列表不会生成新的列表对象，那么，这样的数据该怎样备份呢？

```
>>> ls = [425, "BIT", 1024]
>>> lt = ls
>>> ls[0] = 0
>>> lt
[0, 'BIT', 1024]
```

```
>>> id(ls)
67407824
>>> id(lt)
67407824
```

浅拷贝

```
>>> lt=ls.copy()
>>> ls[0]=0
>>> ls
[0, 'BIT', 1024]
>>> lt
[425, 'BIT', 1024]
```

```
>>> id(ls)
67407824
>>> id(lt)
61086936
```

```
>>> ls=[425, 'BIT', [12,3,9]]
>>> lt=ls.copy()
>>> ls[2][1]=0
>>> ls
[425, 'BIT', [12, 0, 9]]
>>> lt
[425, 'BIT', [12, 0, 9]]
```

深拷贝

```
>>> import copy
>>> lt=copy.deepcopy(ls)
>>> ls[2][1]=6
>>> ls
[425, 'BIT', [12, 6, 9]]
>>> lt
[425, 'BIT', [12, 0, 9]]
```

列表切片

切片公式：List [start : stop : step]

- start参数如省略，则默认为0。
- stop参数如省略，则默认到列表结尾所有元素，如不省略，则切片到此下标索引之前为止(不包括此下标索引的元素。)
- step参数为步长，省略则默认为1，也可以为负整数，表示 反向切片。

```
>>> lst = list(range(6))
>>> print(lst[:])
>>> print(lst[2:4])
>>> print(lst[1:5:2])
>>> print(lst[:3])
>>> print(lst[3:])
>>> print(lst[::3])
>>> print(lst[3:10])
>>> print(lst[0:4:-1])
>>> print(lst[-1:-3:-1])
>>> print(lst[::-1])
>>> print(lst[2::-1])
>>> print(lst[-3:])
>>> print(lst[:-5])
>>> lst[1:3]=["bit", "computer"]
>>> lst
[0,"bit","computer",3,4,5]
```

列表常用方法

append()、insert()、extend( )

```
>>> lst = [1, 2, 3, 4]
>>> lst.append(5)              #在列表尾部增加一个元素
>>> print(lst)                 # [1,2,3,4,5]
>>> Lst.append([11,12])        #将另一列表作为一个元素加入到当前列表的尾部
>>> print(lst)                 # [1,2,3,4,5,[11,12]]
>>> lst.insert(2, 1.5)         #在列表2下标位置插入一个元素
>>> print(lst)                 # [1,2,1.5,3,4,5,[11,12]]
>>> lst.extend([6, 7])         #将另一列表的多个元素顺序插入到当前列表的尾部
>>> print(lst)                 # [1,2,1.5,3,4,5,[11,12],6,7]
```

列表常用方法

pop()、remove()

```
lst = [1, 2, 3, 4, 5, 6]
print(lst.pop())                # 删除并返回最后一个元素
print(lst.pop(2))               # 删除并返回下标为2的元素，后面的元素向前移动
lst = [1, 2, 3, 2, 4, 2]
lst.remove(2)                   # 删除第一个2，后面的元素向前移动，该方法没有返回值
print(lst)
```

count()、index()

```
lst = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
print(lst.count(2))                     # 输出2
print(lst.index(4))                     # 输出6
print(lst.index(5))                     # 代码抛出异常，提示5 is not in list
```

列表常用方法

列表案例

代码编写好后必须要经过反复测试，不能满足于几次测试结果正确。

```
>>> x = [1,2,1,2,1,2,1,2,1]
>>> for i in x:
        if i == 1:
            x.remove(i)
>>> print(x)
[2, 2, 2, 2]
```

```
>>> x = [1,2,1,2,1,1,1]
>>> for i in x:
        if i == 1:
            x.remove(i)
>>> print(x)
[2, 2, 1]
```

正确的代码：

```
>>> x = [1,2,1,2,1,1,1]
>>> for i in x[::]:          #切片
        if i == 1:
            x.remove(i)
```

或者：

```
>>> x = [1,2,1,2,1,1,1]:          #从后往前删
>>> for i in range(len(x)-1,-1,-1)
        if x[i]==1:
            del x[i]
```

列表常用方法

sort()、reverse()

```
import random
# 在range(10000)中任选10个不重复的随机数
data = random.sample(range(10000), 10)

>>> print(data)
>>> data.reverse()                    # 翻转，首尾交换，该方法没有返回值
>>> print(data)
>>> data.sort()                       # 按元素大小进行排序，该方法没有返回值
>>> print(data)
>>> data.sort(key=str)                # 按所有元素转换为字符串后的大小进行排序
>>> print(data)
```

列表推导式

列表推导式是用非常简洁的方式对列表或其他可迭代对象的元素进行遍历、过滤或再次计算，快速生成满足特定需求的新列表。

语法格式：[expression for expr1 in sequence1 if condition1
for expr2 in sequence2 if condition2
……..
for exprN in sequenceN if conditionN]

```
data = [2**i for i in range(64)]
```

等价于：

```
data = []
for i in range(64):
    data.append(2**i)
```

列表推导式

生成20以内的奇数列表

```
>>> data = []
>>> for num in range(20):
        if num%2 == 1:
            data.append(num)

>>> data = [num for num in range(20) if num%2==1]
```

使用随机函数生成一个三行五列的二维数组

```
>>> from random import random
>>> data = []
>>> for i in range(3):
        temp = []
        for j in range(5):
            temp.append(random())
        data.append(temp)

>>> data = [[random() for j in range(5)] for i in range(3)]
```

列表推导式

找出200以内所有的素数

```
>>> data=[]
>>> for i in range(3,200):
        for j in range(2,i):
            if i%j==0:
                break
        if j==i-1:
            data.append(i)
```

```
>>> data=[p for p in range(3,200) if 0 not in [p%d for d in range(2,int(p**0.5))]]
```

打印九九乘法口诀表

```
>>> for x in range(1,10):
    for y in range(1,x+1):
        print("%dx%d=%-4d"%(x,y,x*y),end="")
    print('')
```

```
>>> print('\n'.join([''.join(["%dx%d=%-4d"%(x,y,x*y) for y in range(1,x+1)])
                    for x in range(1,10)]))
```

元组与生成器表达式

元组的特点

- 可以通过把若干元素放在一对圆括号中创建元组，如果只有一个元素的话则需要多加一个逗号，例如(3,)。

- 也可以使用tuple()函数把列表、字典、集合、字符串以及range对象、map对象、zip对象或其他类似对象转换为元组。

- 很多内置函数、标准库函数和扩展库函数也会返回元组或者包含元组的对象。

元组与生成器表达式

元组与列表的区别

- 元组是不可变的，不能直接修改元组中元素的值，也不能为元组增加或删除元素。因此，元组没有提供append()、extend()和insert()等方法，也没有remove()和pop()方法。
- 元组的访问速度比列表更快，开销更小。如果定义了一系列常量值，主要用途只是对它们进行遍历或其他类似操作，那么一般建议使用元组而不用列表。
- 元组可以使得代码更加安全。例如，调用函数时使用元组传递参数可以防止在函数中修改元组，而使用列表则无法保证这一点。
- 元组可用作字典的键，也可以作为集合的元素，但列表不可以，包含列表的元组也不可以。

元组与生成器表达式

生成器表达式

生成器表达式的语法与列表推导式非常相似，在形式上，生成器表达式使用圆括号作为定界符，生成器表达式的计算结果是一个生成器对象。

生成器对象具有惰性求值的特点，犹如迭代对象，只在需要时生成新元素，相比列表推导式具有更高的运行效率，且对空间占用又非常少，尤其适合大数据处理。

```
>>> gen = (2**i for i in range(8))          # 创建生成器对象
>>> print(gen)                              # 迭代对象，看不到具有内容
>>> print(list(gen))                        # 转换为列表，用完了生成器对象中的所有元素
>>> print(tuple(gen))                       # 转换为元组，得到空元组
>>> gen = (2**i for i in range(8))          # 重新创建生成器对象
>>> print(next(gen))                        # 使用next()函数访问下一个元素
>>> print(next(gen))                        # 使用next()函数继续访问下一个元素
>>> for item in gen:                        # 使用for循环访问剩余的所有元素
        print(item, end=' ')
```

元组与生成器表达式

序列解包

序列解包的本质是对多个变量同时进行赋值，即把一个序列或可迭代对象中的多个元素的传下同时赋值给多个变量，要求等号左侧变量的数量和等号右侧值的数量必须一致。

```
>>> x, y, z = 1, 2, 3                    # 多个变量同时赋值
>>> x, y, z = (False, 3.5, 'exp')        # 元组支持序列解包
>>> x, y, z = [1, 2, 3]                  # 列表支持序列解包
>>> x, y = y, x                          # 交换两个变量的值
>>> x, y, z = map(int, '123')            # map对象支持序列解包

>>> data = {'a': 97, 'b': 98}
>>> x1, y1 = data                        # 默认使用字典的“键”进行序列解包
>>> x1, y1 = data.keys()                 # 使用字典的“键”进行序列解包
>>> x2,y2 = data.items()                 # 使用字典的“键值”进行序列解包
>>> x3,y3 = data.values()                # 使用字典的“值”进行序列解包
```

字典

字典的特点

- 字典属于容器对象，其中包含若干元素，每个元素由“键”和“值”两部分构成，之间使用“:”隔开，表示一种对应关系。
- 字典中不同元素之间用逗号分隔，所有元素放在一对大括号中。
- 字典中元素的“键”可以是任意不可变的数据，即可变数据不能作为字典的“键”。
- 字典中的“键”不允许重复，而“值”可以重复。

```
>>> data = dict()
>>> data = dict.fromkeys([1, 2, 3, 4])          # 以指定的数据为“键”，“值”为空
>>> data = dict(name='张三', age=18, sex='M')
>>> data = dict(zip('abcd', [97,98,99,100]))
>>> data = {ch:ord(ch) for ch in 'abcd'}        # 字典推导式
```

字典元素访问

```
>>> data = dict(name='张三', age=18, sex='M')
>>> print(data['name'])                          # 使用“键”作为下标，访问“值”
>>> print(data.get('age'))
>>> print(data.get('address', '不存在这个键'))    # “键”不存在，返回默认值
>>> print(list(data.keys()))                     # 把所有的“键”转换为列表
>>> print(list(data.values()))                   # 把所有的“值”转换为列表
>>> print(list(data.items()))                    # 把所有的元素转换为列表
>>> for key, value in data.items():              # 遍历字典的“键：值”元素
        print(key, value, sep='\t')
```

字典元素修改、添加与删除

- 当以指定“键”为下标为字典元素赋值时，有两种含义：
  1）若该“键”存在，表示修改该“键”对应的值；
  2）若不存在，表示添加一个新元素。

```
>>> sock = {'IP': '127.0.0.1', 'port': 80}
>>> sock['port'] = 8080                    # 修改已有元素的“值”
>>> sock['protocol'] = 'TCP'               # 增加新元素
```

- 使用字典对象的update()方法可以将另一个字典的元素一次性全部添加到当前字典对象，如果两个字典中存在相同的“键”，则以另一个字典中的“值”为准对当前字典进行更新。

```
>>> sock = {'IP': '127.0.0.1', 'port': 80}
>>> sock.update({'IP':'192.168.9.62', 'protocol':'TCP'})
# 更新了一个元素的“值”，增加了一个新元素
```

字典元素修改、添加与删除

可以使用字典对象的pop()方法删除指定“键”对应的字典元素，同时返回该元素对应的“值”。字典方法popitem()方法用于删除字典最后一个元素并返回一个由被删除元素的键和值构成的元组。另外，也可以使用del删除指定的“键”对应的元素。

```
>>> sock = {'IP': '192.168.9.62', 'port': 80, 'protocol': 'TCP'}
>>> print(sock.pop('IP'))                    # 删除并返回指定“键”的元素
>>> print(sock.popitem())                    # 删除并返回一个元素
>>> del sock['port']                         # 删除指定“键”的元素
>>> print(sock)
```

集合概述

- Python集合是无序、可变的容器对象，所有元素放在一对大括号中，元素之间使用逗号分隔，同一个集合内的每个元素都是唯一的，不允许重复。
- 集合中只能包含数字、字符串、元组等不可变类型的数据，而不能包含列表、字典、集合等可变类型的数据，包含列表等可变类型数据的元组也不能作为集合的元素。
- 集合中的元素是无序的，元素存储顺序和添加顺序并不一致。
- 集合不支持使用下标直接访问特定位置上的元素，也不支持使用random中的choice()函数从集合中随机选取元素，但支持使用random模块中的sample()函数随机选取部分元素。

集合常用的方法

add()、update()

```
>>> data = {30, 40, 50}
>>> data.add(20)                    # 增加新元素20
>>> data.add(50)                    # 集合中已包含50，会忽略本次操作
>>> data.update({40, 60})           # 增加另一个集合中新元素60，忽略已有40
>>> print(data)
```

pop()、remove()、discard()

```
>>> data = {30, 60, 90}
>>> data.remove(30)                 # 删除指定元素30，如果不存在，将报错
>>> data.discard(30)                # 删除指定元素30，如果不存在，则忽略本次操作
>>> print(data.pop())               # 删除并返回集合中的一个元素
>>> print(data)
```

集合应用案例

使用集合快速提取序列中单一元素

```
>>> import random
>>> listRandom = [random.choice(range(10000)) for i in range(100)]
>>> noRepeat = []
>>> for i in listRandom :
        if i not in noRepeat :          #通过子集运算，过滤重复数据
            noRepeat.append(i)
>>> len(listRandom)
>>> len(noRepeat)

>>> newSet = set(listRandom)
```

常用内置函数

◆内置函数（BIF，built-in functions）是Python内置对象类型之一，不需要额外导入任何模块即可直接使用，这些内置对象都封装在内置模块__builtins__之中，使用内置函数dir()可以查看所有内置函数和内置对象，注意builtins两侧各有两个下划线，一共4个。

```
>>> print(dir(__builtins__))
```

◆使用help(函数名)可以查看某个函数的用法。

```
>>> help(sum)
Help on built-in function sum in module builtins:
sum(iterable, start=0, /)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers
    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

常用内置函数—类型转换

int()、float()、complex()

```
>>> print(int(3.5))                  # 获取实数的整数部分
>>> print(int('119'))                # 把整数字符串转换为整数
>>> print(int('1111', 2))            # 把1111看作二进制数，转换为十进制数
>>> print(int('1111', 8))            # 把1111看作八进制数，转换为十进制数
>>> print(int('1111', 16))           # 把1111看作十六进制数，转换为十进制数
>>> print(int('  9\n'))              # 自动忽略字符串两个的空白字符
>>> print(float('3.1415926'))        # 把字符串转换为实数
>>> print(float('-inf'))             # 负无穷大
>>> print(complex(3, 4))             # 复数
>>> print(complex(6j))
>>> print(complex('3'))
```

bin()、oct()、hex()

```
>>> print(bin(8888))                 # 把整数转换为二进制
>>> print(oct(8888))                 # 把整数转换为八进制
>>> print(hex(8888))                 # 把整数转换为十六进制
```

常用内置函数—类型转换

ord()、chr()、str()

```
>>> print(ord('a'))                  # 返回字符的ASCII码
>>> print(ord('冷'))                 # 返回汉字字符的Unicode编码    20919
>>> print(chr(65))                   # 返回指定ASCII码对应的字符
>>> print(chr(20919))                # 返回指定Unicode编码对应的汉字
>>> print(str([1, 2, 3, 4]))         # 把列表转换为字符串
>>> print(str({1, 2, 3, 4}))         # 把集合转换为字符串
```

list()、tuple()、dict()、set()

```
>>> print(list(),tuple(),dict(),set())          #[] () {} set()
>>> s = {3, 2, 1, 4}
>>> print(list(s), tuple(s))                    #[1, 2, 3, 4] (1, 2, 3, 4)
>>> lst = [1, 1, 2, 2, 3, 4]
>>> print(tuple(lst), set(lst))                 #(1, 1, 2, 2, 3, 4) {1, 2, 3, 4}
>>> print(dict(name='Li',sex='Male',age=41))    #{'name': 'Li', 'sex': 'Male', 'age': 41}
>>> print(list('123456'))                       #['1', '2', '3', '4', '5', '6']
```

注：list()会把字符串中每个字符都转换为列表中的元素，tuple()、set()函数也具有类似的特点。

常用内置函数—极值

max()、min()

```
>>> data = [3, 22, 111]
>>> print(max(data))                    # 输出列表中数值元素最大值的元素   111
>>> print(min(data))                    # 输出列表中数值元素最小值的元素   3
>>> print(max(data, key=str))           # 把列表中数值型元素作为字符型，输出最大值元素   3

>>> data = ['3', '22', '111']
>>> print(max(data))                    # 输出列表中字符元素最大值的元素   '3'
>>> print(max(data,key=len))            # 输出列表中字符串长度最大的元素   '111'
>>> data = [1, 1, 1, 2, 2, 1, 3, 1]
>>> print(max(set(data), key=data.count))          # 输出列表中出现次数最多的元素 1
>>> print(max(range(len(data)), key=data.__getitem__))
    # 最大元素的索引位置，列表方法__getitem__()用于获取指定位置的值    6
>>> print(data.index(max(data)))        #最大元素的位置索引   6
```

常用内置函数—元素数量、求和

len()、sum()

```
>>> data = [1, 2, 3, 4] #列表
>>> print(len(data))
>>> print(sum(data))
>>> data = (1, 2, 3)     #元组
>>> print(len(data))
>>> print(sum(data))
>>> data = {1, 2, 3}     #集合
>>> print(len(data))
>>> print(sum(data))
>>> data ='Readability counts.'   #字符串
>>> print(len(data))
```

```
>>> data = {97:'a', 65:'A', 48:'0'}   #字典
>>> print(len(data))
>>> print(sum(data))
```

注：对字典数据求和，默认是对其键(data.keys())求和，其键必须为数值。

```
>>> data = {'a':97, 'A':65, '0':48}   #字典
>>> print(sum(data.values()))
```

常用内置函数—基本输入输出

input([prompt])

内置函数**input([prompt])**用来接收用户的键盘输入，**prompt**为字符型输入提示信息，不论用户输入什么内容，**input()**一律返回字符串。所以，对于非字符类型的数据，还需使用其他用内置函数如**int()**、**float()**或**eval()**对用户输入的内容进行类型转换。

```
>>> num = int(input('请输入一个大于2的自然数：'))
>>> if num%2 ==1:                    # 对2的余数为1的整数为奇数，能被2整除的整数为偶数
        print('这是个奇数。')
    else:
        print('这是个偶数。')
```

eval()

内置函数**eval()**是用来执行一个字符串表达式，并返回表达式运算的结果。

```
>>> print(eval('8**2'))
>>> print(eval('[1, 2, 3, 4, 5]'))
>>> print(eval('{1, 2, 3, 4}'))
```

```
>>> lst = eval(input('请输入一个自然数的列表：'))
>>> print('列表中所有元素之和为：', sum(lst))
```

常用内置函数—基本输入输出

print()

内置函数print()用于以指定的格式输出信息，语法格式为：

print(value1, value2, ..., sep=' ', end='\n')

其中，**sep**参数之前为需要输出的内容（可以有多个）；**sep**参数用于指定数据之间的分隔符，如果不指定则默认为空格；**end**参数表示输出完所有数据之后的结束符，如果不指定则默认为换行符。

```
>>> print(1, 2, 3, 4, 5)            # 默认情况，使用空格作为分隔符
>>> print(1, 2, 3, 4, 5, sep=',')   # 指定使用逗号作为分隔符
>>> print(3, 5, 7, end=' ')         # 输出完所有数据之后，以空格结束，不换行
```

常用内置函数—枚举与迭代

range([start,] stop [,step])

```
>>> range1 = range(4)              # 只指定stop为4，start默认为0，step默认为1
>>> range2 = range(5, 8)           # 指定start=5和stop=8，step默认为1
>>> range3 = range(3, 20, 4)       # 指定start=3、stop=20和step=4
>>> range4 = range(20, 0, -3)      # step也可以是负数
>>> print(range1, range2, range3, range4)
>>> print(range4[2])
>>> print(list(range1), list(range2), list(range3), list(range4))
>>> for i in range(10):
        print(i, end=' ')
```

常用内置函数—排序、逆序

sorted

函数**sorted()**可以对列表、元组、字典、集合或其他可迭代对象进行排序，并返回新列表，支持使用**key**参数指定排序规则，**key**参数的值可以是函数、类、**lambda**表达式、方法等可调用对象。另外，还可以使用**reverse**参数指定是升序**(reverse=False)**排序还是降序**(reverse=True)**排序，默认为升序排序。

```
>>> import random
>>> data = list(range(20))
>>> random.shuffle(data)                          # 随机打乱顺序
>>> print(data)
>>> print(sorted(data))                           # 升序排序
>>> print(sorted(data, key=str))                  # 按转换成字符串后的大小升序排序
>>> print(sorted(data, key=str, reverse=True))    # 按转换成字符串后的大小降序排序
```

常用内置函数—排序、逆序

reversed()

reversed()可以对可迭代对象(生成器对象和具有惰性求值特性的zip、map、enumerate、filter、reversed等类似对象除外)进行翻转并返回可迭代的reversed对象。在使用时应注意，reversed对象具有惰性求值特点，其中的元素只能使用一次，并且不支持使用内置函数len()计算元素个数，也不支持使用内置函数reversed()再次翻转。

```
>>> import random
>>> data = list(range(20))          # 创建列表
>>> random.shuffle(data)                    # 随机打乱顺序
>>> print(data)
>>> reversedData = reversed(data) # 生成reversed对象
>>> print(reversedData)
>>> print(list(reversedData))       # 根据reversed对象得到列表
>>> print(tuple(reversedData))      # 空元组，reversed对象中元素只能使用一次
```

常用内置函数—枚举与迭代

enumerate()

内置函数enumerate()函数用来枚举可迭代对象中的元素，返回可迭代的enumerate对象，其中每个元素都是包含索引和值的元组。

```
>>> list(enumerate('abcd'))                              #枚举字符串中的元素
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
>>> list(enumerate(['Python', 'Greate']))                #枚举列表中的元素
[(0, 'Python'), (1, 'Greate')]
>>> list(enumerate({'a':97, 'b':98, 'c':99}.items()))    #枚举字典中的元素
[(0, ('c', 99)), (1, ('a', 97)), (2, ('b', 98))]
>>> for index, value in enumerate(range(10, 15)):        #枚举range对象中的元素
    print((index, value), end=' ')
(0, 10) (1, 11) (2, 12) (3, 13) (4, 14)
```

常用内置函数—枚举与迭代

zip(sqe1,sqe2)

内置函数zip()函数用来把多个可迭代对象中的元素压缩到一起，返回一个可迭代的zip对象，其中每个元素都是包含原来的多个可迭代对象对应位置上元素的元组，如同拉拉链一样。

```
>>> data = zip('1234', [1, 2, 3, 4, 5, 6])
>>> print(data)
# 在转换为列表时，使用了zip对象中的全部元素，zip对象中不再包含任何内容
>>> print(list(data))
# 如果需要再次访问其中的元素，必须重新创建zip对象
>>> data = zip('1234', [1, 2, 3, 4, 5, 6])
>>> print(tuple(data))
>>> data = zip('1234', [1, 2, 3, 4, 5, 6])
>>> for item in data:      # zip对象是可迭代的，可以使用for循环逐个遍历和访问其中的元素
        print(item)
```

常用内置函数—枚举与迭代

map(func, seq1)

map()函数是将func作用于seq中的每一个元素，并将所有的调用的结果作为一个可迭代map对象返回。

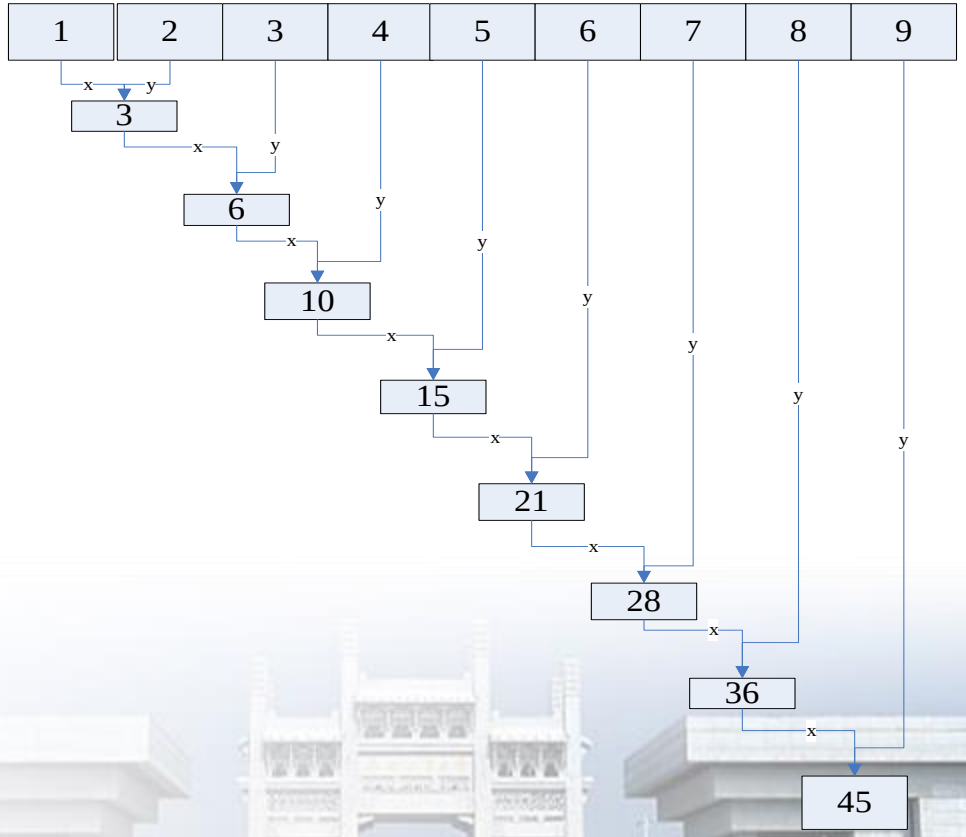
```
>>> print(map(str, range(5)))
>>> print(list(map(str, range(5))))
>>> print(list(map(len, ['abc', '1234', 'test'])))

# 使用operator标准库中的add运算add运算相当于运算符+
# 如果map()函数的第一个参数func能够接收两个参数，则可以映射到两个序列上
>>> from operator import add
>>> for num in map(add, range(5), range(5,10)):
        print(num)
```

常用内置函数—枚举与迭代

reduce()

标准库functools中的函数reduce()可以将一个接收2个参数的函数以迭代累积的方式从左到右依次作用到一个序列或迭代器对象的所有元素上，并且允许指定一个初始值。




```
>>> from functools import reduce
>>> from operator import add, mul, or_
>>> seq = list(range(1,10))
>>> reduce(lambda x, y: x+y, seq)
45

>>> print(reduce(add, seq))      # 累加seq中的数字
>>> print(reduce(mul, seq))      # 累乘seq中的数字
>>> seq = [{1}, {2}, {3}, {4}]
>>> print(reduce(or_, seq))      # 对seq中的集合连续进行并集运算
```

常用内置函数—枚举与迭代

filter()

内置函数filter()将一个单参数函数作用到一个序列上，返回该序列中使得该函数返回值为True的那些元素组成的filter对象，如果指定函数为None，则返回序列中等价于True的元素。

```
>>> seq = ['abcd', '1234', '.,?!', '']
>>> filt=filter(str.isdigit, seq)                  # 只保留数字字符
>>> print(list(filt))                              # 输出可迭代对象的列表
>>> print(tuple(filt))                             #输出可迭代对象的元组
>>> print(list(filter(str.isalpha, seq)))          # 只保留英文字母字符串
>>> print(list(filter(str.isalnum, seq)))          # 只保留数字字符串和英文字符串
>>> print(list(filter(None, seq)))                 # 只保留等价于False的元素
```

Python常用内置函数

函数	功能简要说明
abs(x)	返回数字x的绝对值或复数x的模
all(iterable)	如果对于可迭代对象中所有元素x都等价于True，也就是对于所有元素x都有bool(x)等于True，则返回True。对于空的可迭代对象也返回True
any(iterable)	只要可迭代对象iterable中存在元素x使得bool(x)为True，则返回True。对于空的可迭代对象，返回False
ascii(obj)	把对象转换为ASCII码表示形式，必要的时候使用转义字符来表示特定的字符
bin(x)	把整数x转换为二进制串表示形式
bool(x)	返回与x等价的布尔值True或False
bytes(x)	生成字节串，或把指定对象x转换为字节串表示形式
compile()	用于把Python代码编译成可被exec()或eval()函数执行的代码对象
complex(real, [imag])	返回复数
chr(x)	返回Unicode编码为x的字符
ord('x')	返回字符x的Unicode编码

Python常用内置函数

函数	功能简要说明
delattr(obj, name)	删除属性，等价于del obj.name
dir(obj)	返回指定对象或模块obj的成员列表，如果不带参数则返回当前作用域内所有标识符
divmod(x, y)	返回包含整商和余数的元组((x-x%y)/y, x%y)
enumerate(iterable[, start])	返回包含元素形式为(0, iterable[0]), (1, iterable[1]), (2, iterable[2]), ...的迭代器对象
eval(s[, globals[, locals]])	计算并返回字符串s中表达式的值
exec(x)	执行代码或代码对象x
exit()	退出当前解释器环境
filter(func, seq)	返回filter对象，其中包含序列seq中使得单参数函数func返回值为True的那些元素，如果函数func为None则返回包含seq中等价于True的元素的filter对象
float(x)	把整数或字符串x转换为浮点数并返回
frozenset([x])	创建不可变的集合对象
getattr(obj, name[, default])	获取对象中指定属性的值，等价于obj.name，如果不存在指定属性则返回default的值，如果要访问的属性不存在并且没有指定default则抛出异常

Python常用内置函数

函数	功能简要说明
globals()	返回包含当前作用域内全局变量及其值的字典
hasattr(obj, name)	测试对象obj是否具有名为name的成员
hash(x)	返回对象x的哈希值，如果x不可哈希则抛出异常
help(obj)	返回对象obj的帮助信息
hex(x)	把整数x转换为十六进制串
id(obj)	返回对象obj的标识（内存地址）
input([提示])	显示提示，接收键盘输入的内容，返回字符串
int(x[, d])	返回实数（float）、分数（Fraction）或高精度实数（Decimal）x的整数部分，或把d进制的字符串x转换为十进制并返回，d默认为十进制
isinstance(obj, class-or-type-or-tuple)	测试对象obj是否属于指定类型（如果有多个类型的话需要放到元组中）的实例
iter(...)	返回指定对象的可迭代对象
len(obj)	返回对象obj包含的元素个数，适用于列表、元组、集合、字典、字符串以及range对象和其他可迭代对象

Python常用内置函数

函数	功能简要说明
list([x])、set([x])、tuple([x])、dict([x])	把对象x转换为列表、集合、元组或字典并返回，或生成空列表、空集合、空元组、空字典
locals()	返回包含当前作用域内局部变量及其值的字典
map(func, *iterables)	返回包含若干函数值的map对象，函数func的参数分别来自于iterables指定的每个迭代对象，
reduce(func, sequence[, initial])	将双参数的函数func以迭代的方式从左到右依次应用至序列seq中每个元素，最终返回单个值作为结果。在Python 2.x中该函数为内置函数，在Python 3.x中需要从functools中导入reduce函数再使用
max(x)、 min(x)	返回可迭代对象x中的最大值、最小值，要求x中的所有元素之间可比较大小，允许指定排序规则和x为空时返回的默认值。
sum(x, start=0)	返回序列x中所有元素之和，返回start+sum(x)
next(iterator[, default])	返回可迭代对象x中的下一个元素，允许指定迭代结束之后继续迭代时返回的默认值
oct(x)	把整数x转换为八进制串
str(obj)	把对象obj直接转换为字符串
open(name[, mode])	以指定模式mode打开文件name并返回文件对象

Python常用内置函数

函数	功能简要说明
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)	基本输出函数
range([start,] end [, step])	返回range对象，其中包含左闭右开区间[start,end)内以step为步长的整数
reversed(seq)	返回seq（可以是列表、元组、字符串、range以及其他可迭代对象）中所有元素逆序后的迭代器对象
round(x [, 小数位数])	对x进行四舍五入，若不指定小数位数，则返回整数
pow(x, y, z=None)	返回x的y次方，等价于x ** y或(x ** y) % z
sorted(iterable, key=None, reverse=False)	返回排序后的列表，其中iterable表示要排序的序列或迭代对象，key用来指定排序规则或依据，reverse用来指定升序或降序。该函数不改变iterable内任何元素的顺序
type(obj)	返回对象obj的类型
zip(seq1 [, seq2 [...]])	返回zip对象，其中元素为(seq1[i], seq2[i], ...)形式的元组，最终结果中包含的元素个数取决于所有参数序列或可迭代对象中最短的那个

合肥工業大學
HEFEI UNIVERSITY OF TECHNOLOGY

祝学习进步！

计算机基础教育研究所

2021.3