列表和元组

一、列表

在python中，有这样一些类型，它们的成员是有序排列的，并且可以通过下标访问成员，这些类型称之为**有序序列**，包括：列表、元组和字符串；其中列表的成员可修改，属于**可变序列**，字符串和元组，成员不可修改，属于**不可变序列**。序列有共同操作：

- 成员关系运算（in,not in)
- 连接操作（+）
- 重复操作（*）
- 切片操作（[::]）

1 概述

引出：存储一个数据可以采用变量

问题：需要同时存储多个数据，该怎么做?

```
#需求：有5个人的年龄，求平均年龄
age1 = 10
age2 = 32
age3 = 43
age4 = 18
age5 = 90
average = (age1 + age2 + age3 + age4 + age5) / 5
```

解决：容器【Python提供了一种数据结构list，可以同时存储多个数据】

本质：一种有序的集合

2 创建列表

语法：列表名 = [选项一，选项二，选项三.....]

说明：使用[]表示列表，列表名其实就是标识符

将列表中的列表选项被称为元素

列表中的元素分别被编了号，这些编号被称为索引【下标，角标】

列表索引：

从左向右开始编号：0,1,2....n-1

从右向左编号: -1,-2.....

-7 -6 -5 -4 -3 -2 -1

[10 , 20 , 30 , 40, 50 ,90, 80]

0 1 2 3 4 5 6

```
list1 = []  #创建一个空列表
list2 = [10,9,True,'张三']
list3 = list()  #创建一个空列表
list4 = list('hello world')  # ['h', 'e', 'l', 'l', 'o', ' ', 'w',
'o', 'r', 'l', 'd']
```

3.列表中元素的访问

列表中元素不能整体访问，只能单个存取

3.1取值

语法：列表名[索引]

```
list1 = [1,2,4,3,8]
num = list1[2]
print(num)

#下标越界，引发IndexError， 需要自己确定不要越界
print(list1[5])
```

3.2 修改

语法：列表名[索引] = 新的元素值

```
list1[2] = 888
print(list1)
```

3.3 遍历

列表的遍历就是访问列表的每一个元素，可以使用while循环和for-in循环。经常使用for-in循环遍历列表

for-in语法：

```
for 变量名 in 列表:
     语句块A
else:
      语句块B
```

说明：主要用于遍历序列【遍历：依次访问序列中的每一个元素,获取元素值】

工作原理：按照顺序获取列表中的每个元素，赋值给变量，再执行语句块A，以此类推，直到列表中的元素全部获取为止，循环终止后执行else语句块B。

- 不要在for-in循环中添加或删除元素

```
#只能获取元素，不能得到元素的下标
for value in list1:
    print(value,end=' ')
```

4.序列的通用操作

```
#1.列表组合  将两个列表拼接为一个列表
#直接使用加号
list1 = [43,65,76,6]
list2 = [45,77,90,11,2,4,66]
print(list1 + list2)  #[43,65,76,6,45,77,90,11,2,4,66]  产生一个新列表

#2.列表重复
#直接使用乘号 列表内容重复指定遍数，产生一个新列表
print(list1 * 2)  # [43,65,76,6, 43,65,76,6]

#3.成员操作
#成员运算符：in      not in
#运算的结果为布尔值
list3 = ["hello",False,189,"good"]
print(189 in list3) #True
print(180 not in list3)  #True
```

```
#4.列表的截取【分片，切片】
#语法：列表名[开始下标:结束下标：步长]，表示按指定步长获取从开始下标到结束下标之间的元素，
#       结果为一个新的列表
#注意：包头不包尾【前闭后开区间】    [开始下标,结束下标)
#步长：默认是1，正数表示从左向右取，负数表示从右向左取
list1 = [10,20,30,40,50,60]
print(list1[0:3])   #[10,20,30]
print(list1[:3])    #[10,20,30]
print(list1[:])     #[10,20,30,40,50,60] 从头取到尾
print(list1[::2])     # [10,30,50] 隔一个取一个
print(list1[3::-1])  #[40, 30, 20, 10]  从右向左取
print(list1[-1:-3:-1]) #[60, 50] 从右向左取
print(list1[2:]) #[30, 40, 50, 60] 从下标为2的元素开始取到末尾
print(list1[-3::-1])#[40, 30, 20, 10] 从右向左取到第一个元素

#5.列表长度
print(len(list1))

#6.列表最大值和最小值、求和
print(max(list1))
print(min(list1))
print(sum(list1))
```

5.列表的操作

列表操作包括：增加元素、删除元素、查找元素、列表的反转、列表的排序。

l1 = [10,20,30]

5.1 增加元素

```
#1. list.append(obj) 在列表末尾添加新的元素
l1.append(40) #可以是普通元素
#l1[len(l1):len(l1)] = [40] #等价于l1.append(40)
print(l1)     # [10,20,30,40]
l1.append([50,60]) #也可以是其它列表、元组、字典、集合等对象
print(l1)     # [10,20,30,40,[50, 60]]

#2. list.extend(obj)  在列表的末尾一次性追加另一个列表中的元素
#obj不能是单个数值，可以是列表等可迭代对象
l1.extend([70,80])  #只是将元素添加到l1末尾
```

```
print(l1)

#3. list.insert(i,obj) 在指定下标i的位置插入一个新元素，原来的元素后移，不会覆盖
l1.insert(1,-20)
print(l1)    #[10, -20, 20, 30, 70, 80]
l1.insert(0,[1,2,3])
print(l1)   #[[1, 2, 3], 10, -20, 20, 30, 70, 80]
```

5.2 删除元素

```
#1. list.pop(i) 删除指定下标的元素，如果下标越界，会出现IndexError
value = l1.pop()   #删除最后一个元素，并返回该元素的值
print(value)
l1.pop(0)   #删除下标为0的元素
print(l1)
```

```
#2. list.remove(x) 删除列表中第一个值等于x的元素，删除的时候是从左向右找到第一个等于x的元素删除
l1.remove(-20)
print(l1)
# l1.remove(100) 如果列表中没有该值，会引起错误：ValueError

#3 list.clear()  清空列表
l1.clear()
print(l1)  #[]

# 切片删除
l1[:2] = [] #删除l1[0]和l1[1]
```

5.3 查找元素

```
#1. list.index(x,start,end) 在[start  end)范围内查找第一个等于x的元素的下标
#参数说明：  x 要查找的元素；  start，开始下标；end 结束下标，不包含结束下标
#返回值：如果有值等于x的元素，返回其下标，如果不存在值等于x的元素，会引发ValueError
print(l1.index(10))    #1
print(l1.index(330,2,5))  #4

#2 list.count(x) 查找列表中x出现的次数，如果没有x，返回0
print(l1.count(30))   #2
print(l1.count(99))   #0 不存在99
```

5.4 列表反转

```
#把列表元素逆序排列
l1.reverse()
print(l1)
```

5.5 列表排序

```
#1. list.sort(key=None,reverse=None) 列表方法，实现列表就地排序，不产生新列表
#参数：key参数指明用哪个函数进行排序，默认值是None，用<进行比较  可选参数
      reserse：布尔值，默认值是None，也就是假，从小到大排序，如果设置为True，则从大到小排序，可选参数
l1 = [90,30,70,20,10,60]
print(l1)  # [90, 30, 70, 20, 10, 60]
l1.sort()
print(l1)  # [10, 20, 30, 60, 70, 90]
```

6 二维列表

就是列表的元素还是列表

```
list1 = [[1,2,3],[4,5,6]]
#获取元素
print(list1[0][0],list1[0][1],list1[0][2]) #1  2  3

#二维列表的遍历，循环嵌套
l1 = [[1,2],[3,4],[5,6]]
for elem in l1:
    for value in elem:
        print(value, end=' ')
    print('')
```

二、元组

元组和列表相似，但元组属于不可变序列，所以元组：

- 不能修改元素的值
- 元组用（）表示

1.1 创建元组

```
t1 = ()   #创建一个空元组
#或者
t1 = tuple() #空元组

t2 = (1,) #创建带有一个元素的元组，后面的逗号是必须的，否则无法区分是（）表达式还是元组
或者：t2 = 1,
t3 = (1,4,True,'hello')
t4 = 10,20,30 #t4 = (10,20,30)
t5 = tuple('abc')
```

1.2 成员访问

```
t1 = (10,7,12,23)
print(t1[0])
#print(t1[4]) 下标越界 IndexError

t2 = (1,2,[4,5])
t2[2][0] = 10  #元组的元素无法修改，但元素如果是可变列表，则列表元素是可以修改的
```

1.3 通用操作

```
#1.连接
t1 = (1,2,3)
t2 = (4,5,6)
t3 = t1 + t2
print(t3)  #(1,2,3,4,5,6)

#2.重复
print(t1 * 2)  #(1,2,3,1,2,3)

#3.切片
print(t3[0:3])  #(1,2,3)
print(t3[3:])   #(4,5,6)
print(t3[-1::-1]) (6,5,4,3,2,1)

#4.成员运算符
print(3 in t3)  #True
print(2 not in t3) #False

#5.元素个数
print(len(t3))

#6.最值
print(max(t3))  #6
print(min(t3))  #1
```

1.4 元组其它操作

- 元组和列表的转换

```
t1 = (10,20,30)
l1 = [1,2,3]

#列表转元组
print(tuple(l1))

#元组转列表
print(list(t1))
```

- 查找

```
t1 = (10,20,30,10)
print(t1.index(20))  #查找值等于20的第一个元素
print(t1.count(10))  #返回元组中10的个数
```

- 遍历

```
t1 = (10,20,30,10)
for value in t1:
    print(value)

#同时获取下标和值
for index,value in enumerate(t1):
    print(index, value)

#通过下标遍历
for i in range(len(t1)):
    print(t1[i])
```

1.5 二维元组

```
t1 = ((1,2,3),(4,5,6))

#遍历
for elem in t1:
    for value in elem:
        print(value, end = ' ')
    print('')
```

1.6 序列的解包

```
#元组解包
#变量个数和元素个数一致
t1 = (11,20)
a, b = t1
print(a,b)  #11 20

a, b = 2, 3
a,b,c = (1,2,3)

#变量个数和元素个数不同
#a=10,c=50,_获取（20,30）
a, *_, c = 10,20,30,50
print(a,c,_) #10 50 [20, 30]

#a=10，b=20,c获得剩余的元素
a, b, *c = 10,20,30,50
print(a,b,c) #10 20 [30, 50]

#*解包
print(*(1,2,3))  # 1 2 3

#range解包
a, b, c = range(3)  #a=0,b=1,c=2

#列表解包
a,*_b,c = [1,2,3,4,5]
print(a,_b,c) # 1 [2, 3, 4] 5

#字符串解包
a,b,c = '123'
print(a,b,c) # a='1',b='2',c='3'
```