



■ 第4章 Python的组合数据类型

4.1 列表

4.2 元组

4.3 字典

4.4 集合



- 理解列表、元组、字典的概念和特点
- 掌握对列表、元组、字典操作的相关方法
- 了解集合的概念、特点和对集合操作的相关方法



■ 组合数据类型

- ❖ 计算机不仅对单个变量表示的数据进行处理，更多时候，计算机需要对一组数据进行批量处理。
 - ◆ 给定一组单词：python, data, function, list, loop，计算并输出每个单词的长度
 - ◆ 给定一个学院学生信息，统计一下男女生比例；
 - ◆ 一次实验产生了很多组数据，对这些大量的数据进行分析。



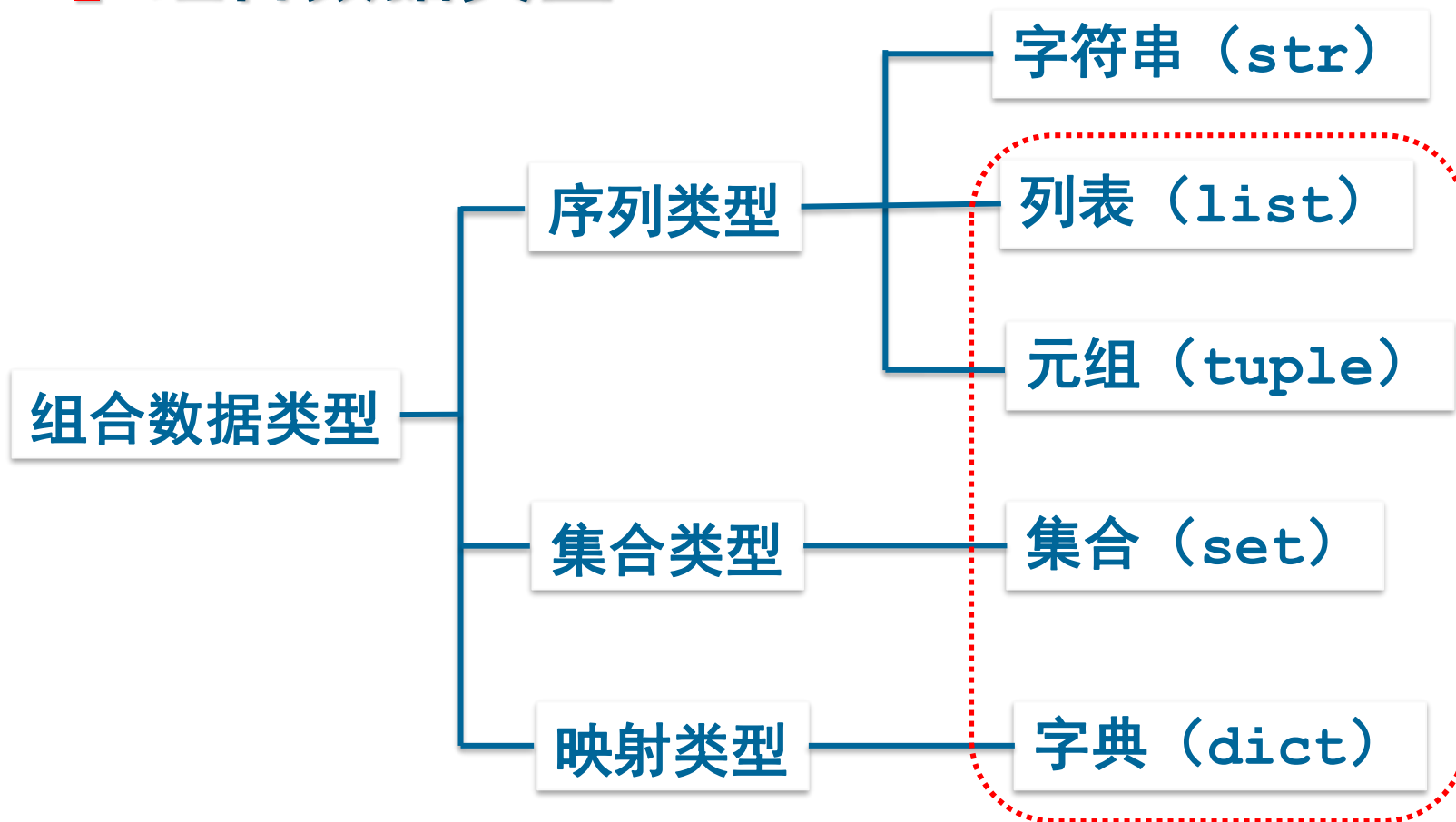
■ 组合数据类型

❖ 组合数据类型能够将多个同类型或不同类型的数据组织起来，通过单一的表示使数据操作更有序更容易。

- ◆ **序列类型：** 是一个元素向量，元素之间存在先后关系，通过序号访问，元素之间不排他（可有相同值）。
- ◆ **集合类型：** 是一个元素集合，元素之间无序，相同元素在集合中唯一存在。
- ◆ **映射类型：** 是“键-值”数据项的组合，每个元素是一个键值对，表示为（key, value）。



■ 组合数据类型





字符串

❖ 字符串是一个字符的序列

```
>>> str1 = "Hello"  
>>> str2 = "Python"  
>>> s = str1 + str2  
>>> print(s)  
HelloPython  
>>> type(s)  
<class 'str'>
```

❖ 求字符串的长

```
>>> s = "Python"  
>>> letter = s[0]  
>>> letter  
P  
>>> s = "Python"  
>>> print(len(s))  
6
```

字符

3

h

```
>>> x = input("x = ")  
x = 100  
>>> y = x - 1  
Traceback (most recent  
  File "<pyshell#22>",  
    y = x - 1  
TypeError: unsupported  
>>> x  
'100'  
>>> y = int(x) - 1  
>>> y  
99
```



■ 字符串

❖ 字符串遍历的方法

```
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(letter)
    index = index + 1
```

```
fruit = 'banana'
for letter in fruit:
    print(letter)
```

运行结果:

```
>>>
== RES'
b
a
n
a
n
a
```

❖ 逻辑运算符 `in` : 判断一个字符串是否存在于另一个字符串中。



H	e	l	l	o		W	o	r	l	d
0	1	2	3	4	5	6	7	8	9	10

字符串

❖ 字符串切片，可以截取

格式：<string>[start

- ◆ start: 起始位置，默认为0
- ◆ end: 终止位置（但不包含）
- ◆ step: 步长，默认为1（同

```
>>> s = "Hello World"
>>> s[0:4]
'Hell'
>>> s[6:7]
'W'
>>> s[6:20]
'World'
>>> s[:4]
'Hell'
>>> s[6:]
'World'
>>> s[::2]
'HloWrld'
>>> s[8:0:-1]
'roW olle'
```



■ 字符串

```
>>> s = "hello world"
>>> dir(s)
['__add__', '__class__', '__contains__', '__delattr__', '__d
ir__', '__doc__', '__eq__', '__format__', '__ge__', '__getat
tribute__', '__getitem__', '__getnewargs__', '__gt__', '__ha
sh__', '__init__', '__init_subclass__', '__iter__', '__le__',
 '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__ne
w__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__',
 '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subcl
asshook__', 'capitalize', 'casefold', 'center', 'count', 'en
code', 'endswith', 'expandtabs', 'find', 'format', 'format_m
ap', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
'isidentifier', 'islower', 'isnumeric', 'isprintable', 'issp
ace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstri
p', 'maketrans', 'partition', 'replace', 'rfind', 'rindex',
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitli
nes', 'startswith', 'strip', 'swapcase', 'title', 'translate
', 'upper', 'zfill']
```




操作

含义

+

*

<string>[]

<string>[:]

len(<string>)

<string>.upper()

<string>.lower()

<string>.strip()

<string>.split()

<string>.join()

<string>.find()

<string>.replace()

for <var> in <string>

左拉

```
>>> s='my name is suohairui'
>>> s.replace('my','My')
'My name is suohairui.'
>>> s
'my name is suohairui.'
>>> r=s.replace('my','My')
>>> r
'My name is suohairui.'
>>> s.replace('m','M',2)
'My naMe is suohairui.'
>>> r
'My name is suohairui.'
>>> s
'my name is suohairui.'
>>> r.replace('s','S')
'My name iS Suohairui.'
>>> s.replace('i','I')
'my name Is suohaIruI.'
>>> r.replace('suo','Suo')
'My name is Suohairui.'
```

按



■ 字符串

❖ 内置的字符串处

操作

<string>.capitalize()

<string>.center(width)

<string>.startswith()

<string>.endswith()

```
>>> s = "Hello Bob"
>>> s.replace('Bob', 'Jane')
'Hello Jane'
>>> s1 = s.center(20)
>>> s1
'      Hello Bob      '
>>> s1.lstrip()
'Hello Bob      '
>>> s1.rstrip()
'      Hello Bob'
>>> s1.strip()
'Hello Bob'
```

```
>>> b=' dddd fdfd' 两端默认为空
>>> b.title()
'Dddd Fdfd'
>>> b.capitalize() 定字符串开头
'Dddd fdfd'
```

判断字符串是否以指定字符串结尾

help(str.函数名)命令：查询某个函数的含义 引

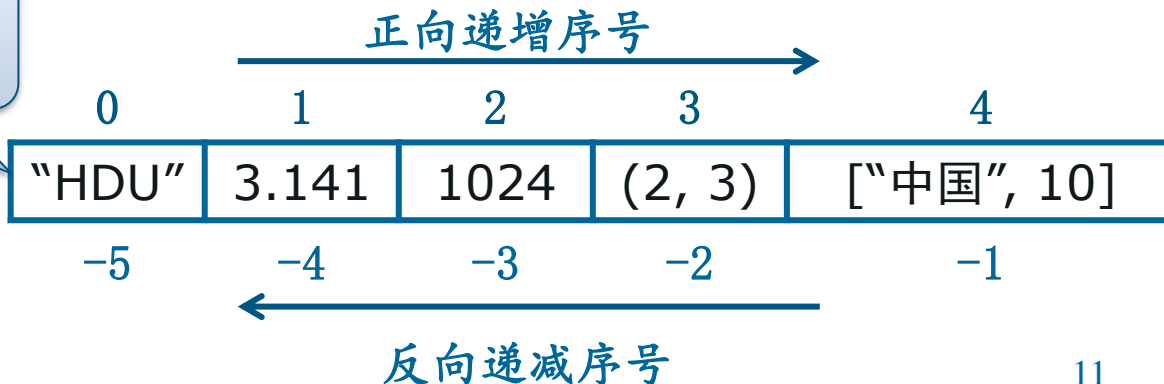


■ 序列类型

除了str（字符串），Python语言中还有另外两个重要的序列类型：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。
- ❖ 列表则是可以修改数据项的序列类型，使用也最灵活。

可以存储不同
类型的数据项





■ 序列类型

序列类型有12个通用的操作符和函数

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



■ 4.1 列表

列表（list）是包含0个或多个数据项的有序序列，每个数据项称为元素。

- ❖ 长度和内容都是**可变**的。可对列表中的数据项进行增加、删除或替换。
- ❖ 是**有序**的，可以通过元素的索引，访问数据项。
- ❖ 没有长度限制，**元素类型可以不同**，使用非常灵活。



4.1 列表

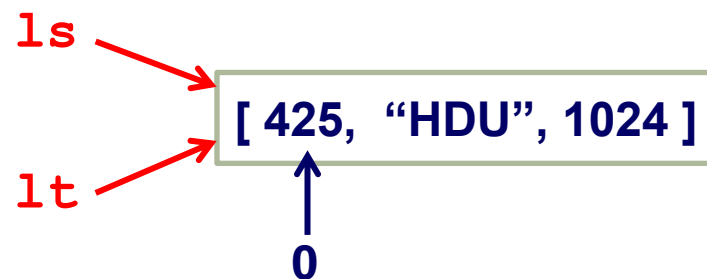

❖ 创建列表

```
>>> ls = [425, "HDU", [10, "CS"], 425]
>>> ls
[425, 'HDU', [10, 'CS'], 425]
>>> ls[2][-1][0]
'C'
```

- ◆ 将逗号分隔的不同数据项，使用中括号 [] 括起来即可创建列表。
- ◆ 也可以通过 list() 函数将元组或字符串转化成列表。
- ◆ 直接使用 list() 函数会返回一个空列表。

```
>>> list((425, 'HDU', [10, 'CS'], 425))
[425, 'HDU', [10, 'CS'], 425]
>>> list("Hello")
['H', 'e', 'l', 'l', 'o']
>>> list()
[]
```

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



❖ 列表赋值

- ◆ 通过显式的数据赋值才能生成一个列表对象。
- ◆ 简单将一个列表的值给另一个列表变量，不会生成新的列表对象。

```
>>> ls = [425, "HDU", 1024]    #用数据赋值产生列表ls
>>> lt = ls                  #lt是ls所对应的数据的引用，lt并不包含真实数据
>>> ls[0] = 0
>>> lt
[0, 'HDU', 1024]
```

操作	含义
<code>ls[i] = x</code>	替换列表 <code>ls</code> 第 <i>i</i> 数据项为 <i>x</i>
<code>ls[i:j] = lt</code>	用列表 <code>lt</code> 替换列表 <code>ls</code> 中第 <i>i</i> 到 <i>j</i> 项（不含）的数据
<code>ls[i:j:k] = lt</code>	用列表 <code>lt</code> 替换列表 <code>ls</code> 中第 <i>i</i> 到 <i>j</i> （不含）以 <i>k</i> 为步长的数据
<code>del ls[i:j]</code>	删除列表 <code>ls</code> 第 <i>i</i> 到 <i>j</i> 项数据，等价于 <code>ls[i:j]=[]</code>
<code>del ls[i:j:k]</code>	删除列表 <code>ls</code> 第 <i>i</i> 到 <i>j</i> （不含）以 <i>k</i> 为步长的数据
<code>del ls</code>	删除列表
<code>ls += lt</code> 或 <code>ls.extend(lt)</code>	将列表 <code>lt</code> 元素增加到列表 <code>ls</code> 中
<code>ls *= n</code>	更新列表 <code>ls</code> ，其元素重复 <i>n</i> 次
<code>ls.append(x)</code>	在列表末尾添加一个元素 <i>x</i>
<code>ls.clear()</code>	删除 <code>ls</code> 中的所有元素，等价于 <code>del a[:]</code>
<code>list.copy()</code>	生成一个新列表，复制 <code>ls</code> 中所有元素
<code>ls.insert(i, x)</code>	在列表 <code>ls</code> 第 <i>i</i> 位置增加元素 <i>x</i>
<code>list.pop(i)</code>	将列表 <code>ls</code> 中第 <i>i</i> 项元素取出，并删除该元素。若不指定索引值，则取出并删除最后一个元素
<code>ls.remove(x)</code>	将列表中第一个值为 <i>x</i> 的元素删除，若不存在则出错
<code>list.reverse()</code>	将列表 <code>ls</code> 中的元素反转

操作	含义
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（不含）的数据
ls[i:j:k] = lt	用列表lt替换列表ls中第i到j（不含）以k为步长的数据
del ls[i:j]	删除列表ls第i到j项（不含）
del ls[i:j:k]	删除列表ls中第i到j（不含）以k为步长的数据
del ls	删除列表ls
ls += lt或 ls.extend(lt)	将列表lt追加到列表ls末尾
ls *= n	更新列表ls，重复n次
ls.append(x)	在列表ls末尾添加元素x
ls.clear()	删除ls中所有元素
list.copy()	生成一个ls的副本
ls.insert(i, x)	在列表ls中第i个元素前插入元素x
list.pop(i)	将列表ls中第i个元素删除并返回该元素索引值
ls.remove(x)	将列表中第一个元素x删除
list.reverse()	将列表ls中元素逆序

```
>>> a=list(b)
>>> a
['f', 'd', 'a', 'd', 'f']
>>> c=a.copy()
>>> c
['f', 'd', 'a', 'd', 'f']
>>> a[1]=1
>>> a
['f', 1, 'a', 'd', 'f']
>>> c
['f', 'd', 'a', 'd', 'f']
>>> b=a
>>> b
['f', 1, 'a', 'd', 'f']
>>> a[3]=3
>>> a
['f', 1, 'a', 3, 'f']
>>> b
['f', 1, 'a', 3, 'f']
```



4.1 列表



❖ 列表

```
>>> vlist = list(range(5))
>>> vlist
[0, 1, 2, 3, 4]
>>> len(vlist[2:])
3
>>> 2 in vlist
True
>>> vlist[3] = "python"
>>> vlist
[0, 1, 2, 'python', 4]
>>> vlist[1:3] = ["hdu", "computer"]
>>> vlist
[0, 'hdu', 'computer', 'python', 4]
>>> vlist[4:0:-1]
[4, 'python', 'computer', 'hdu']
```



4.1 列表

❖ 列表的基本操作

◆ 列表元素的更改

➤ 当使用一个列表改变另一个列表值时，Python不求两个列表长度一样，遵循“多增少减”的原则。

```
>>> vlist[1:3] = ["new_hdu", "new_computer", 123]
>>> vlist
[0, 'new_hdu', 'new_computer', 123, 'python', 4]
>>> vlist[1:3] = ["fewer"]
>>> vlist
[0, 'fewer', 123, 'python', 4]
```



4.1 列表

❖ 列表的基本操作

◆ 列表元素的复制与拼接

◆ 列表元素的求和：用for...in语句对其元素进行遍历

```
>>> zeros = [0]*6
>>> zeros
[0, 0, 0, 0, 0, 0]
>>> print( 2*['a'] + 3*[1, 'c'])
['a', 'a', 1, 'c', 1, 'c', 1, 'c']
```

```
>>> s = list(range(10))
>>> sum = 0
>>> for i in s:
>>>     sum += i

>>> print("sum is", sum, end="")
sum is 45
```



4.1 列表

❖ 删除列表中

◆ **del:** Py

表元素、

◆ **pop():**

```
>>> a = [-1, 1, 66.25, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 1234.5]
>>> del a[2:4]
>>> a
[1, 66.25]
>>> del a[:]
>>> a
[]
>>> del a
>>> a
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined

>>> a = [-1, 1, 66.25, 333, 1234.5]
>>> a.pop()
1234.5
>>> a
[-1, 1, 66.25, 333]
>>> a.pop(0)
-1
>>> a
[1, 66.25, 333]
```

列表功能默写

- 定义空列表lt
- 向lt新增5个元素
- 修改lt中第2个元素
- 向lt中第2个位置增加一个元素
- 从lt中第1个位置删除一个元素
- 删除lt中第1-3位置元素
- 判断lt中是否包含数字0
- 向lt新增数字0
- 返回数字0所在lt中的索引
- lt的长度
- lt中最大元素
- 清空lt

列表功能默写

- 定义空列表lt `>>> lt = []`
- 向lt新增5个元素 `>>> lt += [1,2,3,4,5]`
- 修改lt中第2个元素 `>>> lt[2] = 6`
- 向lt中第2个位置增加一个元素 `>>> lt.insert(2, 7)`
- 从lt中第1个位置删除一个元素 `>>> del lt[1]`
- 删除lt中第1-3位置元素 `>>> del lt[1:4]`
- 判断lt中是否包含数字0 `>>> 0 in lt`
- 向lt新增数字0 `>>> lt.append(0)`
- 返回数字0所在lt中的索引 `>>> lt.index(0)`
- lt的长度 `>>> len(lt)`
- lt中最大元素 `>>> max(lt)`
- 清空lt `>>> lt.clear()`



4.1 列表

❖ 列表解析

一种快速生成列表的简便方法。

```
>>> square = []  
>>> for x in range(10):  
    square.append(x**2)
```

```
>>> square  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
>>> square = [x**2 for x in range(10)]  
>>> square  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```




4.1 列表

❖ 列表解析

是Python语言强有力的语法之一，常用于从集合对象中有选择地获取并计算元素。虽然在多数情况下可以使用for、if等语句组合完成同样的任务，但列表解析书写的代码更简洁。

一般格式：

```
[<表达式> for x1 in <序列1> [... for xN in <序列N> if <条件>]
```

```
>>> [ x**2 for x in range(10) ]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```



4.1 列表

❖ 列表解析

【例】输入以空格作为间隔的一组整数。

```
>>> nums = [int(x) for x in input().split()]
12 56 34 2 78 9 10
>>> nums
[12, 56, 34, 2, 78, 9, 10]
```

```
>>> a, b, c = [eval(x) for x in input().split()]
10 3.14 5
>>> a
10
>>> b
3.14
>>> c
5
```



❖ 输出0-10之间所有偶数的平方

```
>>> [x**2 for x in range(0, 11, 2)]  
[0, 4, 16, 36, 64, 100]
```

❖ 输出x*y的值，x的范围是[1, 5]，y的范围[1, 3]

```
>>> [x*y for x in range(1, 6) for y in range(1, 4)]  
[1, 2, 3, 2, 4, 6, 3, 6, 9, 4, 8, 12, 5, 10, 15]
```



4.1 列表

【例】奇偶分家 #奇数偶数分家 给定一个非负整数列表，将数据按奇偶分成两组，分两行显示

输入样例：

88 74 101 26 1

输出样例：

```
n = input('请输入一组非负整数: ').split()
for i in n:
    if int(i)%2==0:
        print(i, end=' ')
    print()
for j in n:
    if int(j)%2 !=0:
        print(j, end=' ')
```

简化

```
#奇数偶数分家
n = [int(x) for x in input('请输入一组非负整数: ').split()]
a=[]
b=[]
for i in n:
    if i%2==0:
        a.append(i)
    else:
        b.append(i)
print(a)
print(b)
```

#奇数偶数分家

```
n = input('请输入一组非负整数: ').split()
a=[]
b=[]
for i in n:
    if int(i)%2==0:
        a.append(i)
    else:
        b.append(i)
print(a)
print(b)
```

#奇数偶数分家

```
n = [int(x) for x in input('请输入一组非负整数: ').split()]
a=[i for i in n if i%2==0]
b=[j for j in n if j%2!=0]
print(a)
print(b)
```

```
ls = [int(x) for x in input('nums: ').split()]
odd = [x for x in ls if x%2==1]
print('odd: ', odd)
even = [x for x in ls if x%2==0]
print('even: ', even)
```



4.2 元组

❖ 元组 (tuple) 是Python中另一种内置的存储有序数据的结构。元组可以存储不同类型的数据。

```
>>> tup1 = 1, 2, 3
>>> tup1
(1, 2, 3)
>>> tup2 = ("physics", 0x001100, 2018, tup1)
>>> tup2
('physics', 4352, 2018, (1, 2, 3))
>>> tup2[2]
2018
>>> tup2[-1][1]
2
>>> v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
```

元组允许嵌套

元组中可包含可变数据类型的元素

```
>>> a=(1, 2, 3)
>>> a[0]=88
Traceback (most recent call last):
  File "<pyshell#30>", line 1, in <module>
    a[0]=88
TypeError: 'tuple' object does not support item assignment
```

```
a=([1, 2, 3], 5, 6)
```

```
a[0]=[7, 8, 9]
```

```
a[0][0]
```



```
>>> import math
>>> for x, y in ((1,0), (2,5), (3,8)): #循环遍历
    print(math.hypot(x,y)) #求多个坐标值到原点的距离
```

```
1.0
5.385164807134504
8.54400374531753
```

```
>>> def func(x): #函数多返回值
    return x, x**3

>>> func(8)
(8, 512)
```

```
>>> a, b = "dog", "tiger" #多变量同步赋值
>>> a, b = (b, a) #多变量同步赋值, 括号可省
>>> a
'tiger'
>>> b
'dog'
```




4.2 元组

❖ 创建元组

用逗号分隔一些值，那么将自动创建元组，元组通常用圆括号()括起来。换句话说，任意类型的对象

```
>>> empty=() #创建空元组
>>> single = "hello"
>>> len(empty)
0
>>> len(single)
5
>>> single = "hello", #加了逗号,
>>> len(single)
1
>>> single
('hello',)
```

```
>>> a=12
>>> type(a)
<class 'int'>
>>> a=12,
>>> type(a)
<class 'tuple'>
```

元组



```
>>> ls = list()
>>> dir(ls)
>>> tup = tuple()
>>> dir(tup)
```

操作	
<code><tup>[i]</code>	索引，求元组 <code>tup</code> 中第 <code>i</code> 数据项
<code><tup>[i: j: k]</code>	切片，求元组 <code>tup</code> 中第 <code>i</code> 到 <code>j</code> 项（不含）以 <code>k</code> 为步长的子元组
<code><tup1> + <tup2></code>	将两个元组连接
<code><tup> * n</code>	将元组 <code>tup</code> 复制 <code>n</code> 次
<code>for <var> in <tup>:</code>	对元组 <code>tup</code> 中的元素进行遍历
<code><expr> in <tup></code>	查找元组 <code>tup</code> 中是否存在 <code><expr></code> ，返回值为布尔类型
<code>del <tup></code>	删除元组 <code>tup</code>
<code>len(<tup>)</code>	求元组 <code>tup</code> 的长度
<code>max(<tup>)</code>	返回元组中最大值
<code>min(<tup>)</code>	返回元组中最小值
<code><tup>.count(x)</code>	统计元组 <code>tup</code> 中出现元素 <code>x</code> 的总次数
<code><tup>.index(x[, i[, j]])</code>	元组 <code>tup</code> 从 <code>i</code> 开始到 <code>j</code> 位置中第一次出现元素 <code>x</code> 的位置



■ 4.2 元组

❖ 元组与列表的相互转换

- ◆ 内置的**tuple()**函数接收一个列表，可返回一个包含相同元素的元组。
- ◆ **list()** 函数接收一个元组并返回一个列表。
- ◆ 从元组与列表的性质来看，**tuple()**相当于冻结一个列表，而**list()**相当于解冻一个元组。



4.2 元组

❖ 元组与列表的相互转换

【例】 分别从两个列表中取不相等的两个元素组合成元组类型元素的新列表。（用程序、解析列表分别实现）

```
combs = []  
for x in [1,2,3]:  
    for y in [3,1,4]:  
        if x != y:  
            combs.append((x, y))
```

```
>>> [(x, y) for x in [1,2,3] for y in [3,1,4] if x!=y]  
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```



■ 4.2 元组

❖ 元组与列表的相互转换

【例】 利用解析列表生成九九乘法表（由元组元素组成的列表，无需换行）。

```
>>> [(x, y, x*y) for x in range(1,10) for y in range(1,10) if x<=y]
```

思考一下，怎样在列表解析的方法中实现换行？

```
ls=[' {}*{}={} '.format(i, j, i*j) for i in range(1, 10) for j in range(i, 10)]
```

```
ls=[' {}*{}={} '.format(i, j, i*j) for i in range(1, 10) for j in range(i, 10)]  
for i in range(9, 0, -1):  
    for j in range(i):  
        print(ls.pop(0), end=' \t')  
    print()
```

```
ls=[' {}*{}={} '.format(i, j, i*j) for i in range(1, 10) for j in range(i, 10)]  
for i in range(9, 0, -1):  
    print(ls[:i])  
    del ls[:i]
```

```
ls=[print(' {}*{}={} '.format(i, j, i*j), end=' \t' if j<9 else '\n')\n    for i in range(1, 10) for j in range(i, 10)]
```