组合数据类型

内容

- Python中变量的存储
- 列表类型
- 元组类型
- 集合类型
- 字典类型

Python中变量的存储

Python中所有的变量都是指针

- Python中所有可赋值的东西，即所有可出现在赋值语句中”=”左侧的东西都是指针
- 指针指向内存单元中的地址
- 对变量进行赋值的本质，即是让该变量指向某内存地址，该地址中存储着变量的值

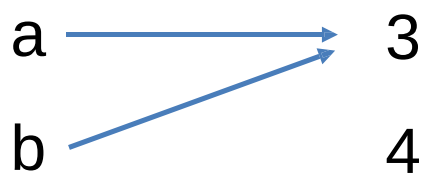
Python中变量的存储

- 对变量进行赋值，即让变量指向某地址

```
a = 3
b = 4
```

- 将某变量赋值为另一变量的本质，是让该变量指向另一变量指向的地址

```
b = a
```




```
b == a
b is a
```

Python中变量值的修改

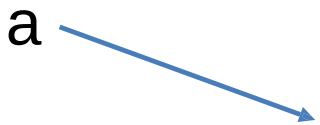
- 对变量的值进行修改，可能的方式有两种

```
a = 3
```

3

4

```
a = 4
```



3

4

修改变量指针指向的地址，在新地址中存储新的值。即，改变存储变量值的地址

```
a = 4
```

~~3~~4

4

修改变量指针当前指向地址中存储的值。即，不改变存储变量值的地址

不可变数据类型和可变数据类型

根据变量所指向内存地址中的值是否可以改变，将数据类型分为两种

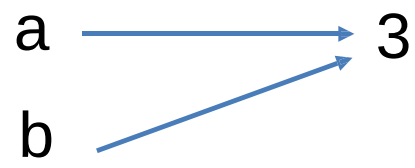
- 不可变数据类型：变量所指向内存地址中的值不能发生改变。即，对于此类变量，如果值发生改变，则内存地址发生改变。数字、布尔、字符串及元组类型是不可变数据类型

- 可变数据类型：变量所指向内存地址中的值可以发生改变。即，对于此类变量，如果值发生改变，内存地址一般不发生改变。集合、列表、字典是可变数据类型

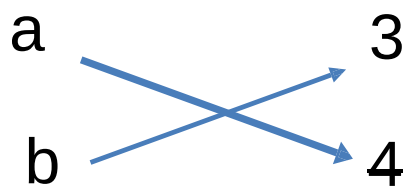
不可变数据类型变量不会相互影响

```
a = 3
```


```
b = a
```



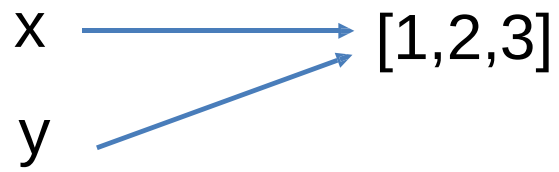

```
a = 4
```



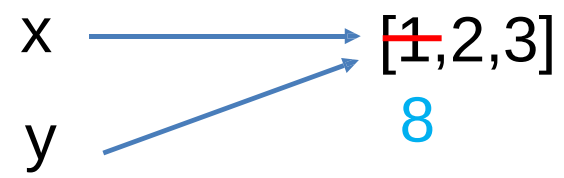
可变数据类型中可能存在的“陷阱”

```
x = [1,2,3]
```


```
y = x
```




```
x[0] = 8
```




```
>>> x = [1, 2, 3]
>>> y=x
>>> x[0] = 8
>>> x
[8, 2, 3]
>>> y
[8, 2, 3]
```

数据类型及其分类

数据类型	类型名称	示例
数字	int float complex	1234 3.14, 1.2e5 5+8j
布尔型	bool	True, False
字符串	str	‘hello world’ “C” ’’’这是一个字符串’’’
元组	tuple	（3，-5，8）
列表	list	[1,2,3] [‘a’, ‘b’, ‘c’]
集合	set	{-5，0，‘a’}
字典	dict	{1:”金牌“，2:”银牌“，3：”铜牌“}

不可变数据类型

序列

可变数据类型

基本数据类型

组合数据类型

序列类型的共同特点

- 每个元素按照其在序列中的顺序都有相应的序号，代表它所在的位置。元素的序号有两个，一个正向，从0开始，依次加1；一个逆向，从-1开始，依次减1
- 可以使用元素序号进行双向索引
- 可以进行如下操作：+，*，切片
- 可以进行成员判断：in，not in
- 可以进行比较运算

内容

- Python中变量的存储
- 列表类型
- 元组类型
- 集合类型
- 字典类型

基本序列类型：列表（List）

列表是有序、可变的组合数据类型

[<item1>, <item2>...]

- 列表中的元素类型可以不同
- 可以对列表元素进行增删
- 可以对列表元素进行修改

```
>>> list1 = ['physics', 'chemistry', 1997, 2000]
>>> list1
['physics', 'chemistry', 1997, 2000]
>>> list1.append(8.29)
>>> list1
['physics', 'chemistry', 1997, 2000, 8.29]
>>> list1[2]= 'math'
>>> list1
['physics', 'chemistry', 'math', 2000, 8.29]
```

基本序列类型：列表（List）

列表（Python）	VS	数组（C）
• 元素类型可以不同		• 元素类型必须相同
• 长度可以改变		• 长度不可改变

列表的基本操作：列表创建

- 直接赋值

```
>>> list1 = ['physics', 'chemistry', 1997, 2000]
>>> list1
['physics', 'chemistry', 1997, 2000]
```

```
>>> list2 =[]
>>> list2
[]
```

- 通过list()函数将其他类型的数据转换为列表

```
>>> list3 = list((1,3,5,7))
>>> list3
[1, 3, 5, 7]
>>> list5 = list('Hello')
>>> list5
['H', 'e', 'l', 'l', 'o']
```

```
>>> list4 = list(range(2,8,2))
>>> list4
[2, 4, 6]
```

列表的基本操作：列表元素的增加

- append()方法

  list.append(x)

- extend()方法

  list.extend(t)

```
>>> list4
[2, 4, 6]
>>> list4.append(-1)
>>> list4
[2, 4, 6, -1]
>>> list4.extend((3,5))
>>> list4
[2, 4, 6, -1, 3, 5]
>>> list4.extend('ab')
>>> list4
[2, 4, 6, -1, 3, 5, 'a', 'b']
```

```
>>> list4.extend([True])
>>> list4
[2, 4, 6, -1, 3, 5, 'a', 'b', True]
```

列表的基本操作：列表元素的增加

- insert()方法

  list.insert(i, x)

```
>>> list5
[1, 2, 3]
>>> list5.insert(2,0)
>>> list5
[1, 2, 0, 3]
```

```
>>> list5.insert(-3,-5)
>>> list5
[1, -5, 2, 0, 3]

>>> list5.insert(10,8)
>>> list5
[1, -5, 2, 0, 3, 8]

>>> list5.insert(-12,7)
>>> list5
[7, 1, -5, 2, 0, 3, 8]
```

列表的基本操作：列表元素的增加

- 通过+在列表中增加元素

  list = list + [x, y...]

- 通过*在列表中增加重复元素

  list = list * n

```
>>> list3
[1, 3, 5, 7]
>>> list3 = list3+[9]
>>> list3
[1, 3, 5, 7, 9]

>>> list6=[2,5,8]
>>> list6 = list6*3
>>> list6
[2, 5, 8, 2, 5, 8, 2, 5, 8]
```

列表的基本操作：索引和切片

- 基本索引用于列表元素的查找、修改与删除

list[i]

list[i] = value

del list[i]

```
>>> list3=[1,3,5,7,9]
>>> x,y = list3[2],list3[-1]
>>> print(x,y)
5 9

>>> list3[1],list3[-2] = 0,8
>>> list3
[1, 0, 5, 8, 9]

>>> del list3[3]
>>> list3
[1, 0, 5, 9]
```

```
>>> list3[8]
Traceback (most recent call last):
  File "<pyshell#100>", line 1, in <module>
    list3[8]
IndexError: list index out of range
```

列表的基本操作：索引和切片

- 切片用于多个列表元素的查找、修改与删除

list[start:stop:step]

list [start:stop:step] = [v1,v2,...]

```
>>> list3[1:3] = [1,2,3]
>>> list3
[-1, 1, 2, 3, 9]
>>> list3[1:3] = [5]
>>> list3
[-1, 5, 3, 9]
```

```
>>> list3
[1, 0, 5, 9]
>>> x,y = list3[0:3:2],list3[-1:-4:-1]
>>> x
[1, 5]
>>> y
[9, 5, 0]

>>> list3[0:3:2] = [-1,-6]
>>> list3
[-1, 0, -6, 9]
```

列表的基本操作：索引和切片

- 切片用于多个列表元素的查找、修改与删除

list[start:stop:step]
list [start:stop:step] = [v1,v2,...]

```
>>> list3=[6,7,8,9,10]
>>> list3[3:0:-1]=[3,2,1]
>>> list3
[6, 1, 2, 3, 10]
>>> list3[3:0:-1]=[3,2,1,0]
Traceback (most recent call last):
  File "<pyshell#143>", line 1, in <module>
    list3[3:0:-1]=[3,2,1,0]
ValueError: attempt to assign sequence of size 4 to extended slice of size 3
>>> list3[3:0:-1]=[3,2]
Traceback (most recent call last):
  File "<pyshell#144>", line 1, in <module>
    list3[3:0:-1]=[3,2]
ValueError: attempt to assign sequence of size 2 to extended slice of size 3
```

列表的基本操作：索引和切片

- 切片用于多个列表元素的查找、修改与删除

list[start:stop:step]

```
>>> list3 = [1,3,5,7,9]
>>> del list3[:4:2]
```

list [start:stop:step] = [v1,v2,...]

```
>>> list3
[3, 7, 9]
```

del list[start:stop:step]

```
>>> del list3[4:5]
>>> list3
[3, 7, 9]
```

列表的基本操作：索引和切片练习

```
>>> list3 = [1,3,5,7,9]
>>> list3[0:2] = [-1]
>>> list3
[-1, 5, 7, 9]

>>> list3[0:2] = [-2,-3,-4]
>>> list3
[-2, -3, -4, 7, 9]

>>> list3[-3:-1]=[]
>>> list3
[-2, -3, 9]
```

```
>>> list3
[-2, -3, 9]
>>> list3[5:8]
[]

>>> list3[5:8]=[2,3,4]
>>> list3
[-2, -3, 9, 2, 3, 4]

>>> list3[-12:-9]=[-10,-9]
>>> list3
[-10, -9, -2, -3, 9, 2, 3, 4]

>>> del list3[10:12]
>>> list3
[-10, -9, -2, -3, 9, 2, 3, 4]
```

列表元素的其它查询方法

- 使用index方法查询值出现的位置

list.index(x[,i[,j]])

```
>>> list3 = [1,2,1,3,5,1,7,9,11]
>>> list3.index(1)
0
>>> list3.index(1,2)
2
>>> list3.index(1,4,8)
5
>>> list3.index(1,6)
Traceback (most recent call last):
  File "<pyshell#126>", line 1, in <module>
    list3.index(1,6)
ValueError: 1 is not in list
```

列表元素的其它删除方法

- 使用pop方法删除默认或指定元素

list.pop([i])

```
>>> list3 = []
>>> list3.pop
<built-in method pop of list object at 0x024136C0>

>>> list3 = [1,2,3,4,5]
>>> list3.pop(7)
Traceback (most recent call last):
  File "<pyshell#90>", line 1, in <module>
    list3.pop(7)
IndexError: pop index out of range
```

```
>>> list3 = [1,2,3,4,5]
>>> x = list3.pop()
>>> x
5
>>> list3
[1, 2, 3, 4]
>>> y = list3.pop(2)
>>> y
3
>>> list3
[1, 2, 4]

>>> y = list3.pop(-1)
>>> y
4
>>> list3
[1, 2]
```

列表元素的其它删除方法

- 使用remove方法删除指定元素

list.remove(x)

```
>>> list4 = [1,3,5,1,7,9]
>>> list4.remove(1)
>>> list4
[3, 5, 1, 7, 9]
```

```
>>> list4.remove(2)
Traceback (most recent call last):
  File "<pyshell#94>", line 1, in <module>
    list4.remove(2)
ValueError: list.remove(x): x not in list
```

列表元素的其它删除方法

- 使用clear方法删除全部元素

  list.clear()

- 使用del list[:] 删除全部元素

```
>>> list4
[5, 1, 7, 9]
>>> list4.clear()
>>> list4
[]
```

```
>>> list3 = [1,2,3]
>>> del list3[:]
>>> list3
[]
```

列表的基本操作：列表删除

- 使用del删除整个列表

del list

```
>>> list3 = [1,2,3]
>>> del list3
>>> list3
Traceback (most recent call last):
  File "<pyshell#107>", line 1, in <module>
    list3
NameError: name 'list3' is not defined
```

列表的成员判断操作

- 使用in判断元素是否在列表中

  x in list

```
>>> list3 = [1,'a',5]
>>> 'a' in list3
True
>>> 8 in list3
False
```

- 使用 not in判断元素是否不在列表中

  x not in list

```
>>> list3 = [1,'a',5]
>>> 'a' not in list3
False
>>> 8 not in list3
True
```

列表的比较操作

列表之间可以通过比较运算符“比大小”

- 只有同一位置类型相同的列表才能进行比较
- 从第一个元素顺序开始比较，如果相等，则继续，返回第一个不相等元素比较的结果
- 如果所有元素比较均相等，则长的列表大，一样长则两列表相等

```
>>> [1,2]<[1,'a']
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    [1,2]<[1,'a']
TypeError: unorderable types: int() < str()
>>> s1 = [3,"abc",4]
>>> s2 = [3,"abcc",2]
>>> s3 = [3,"abc",2]
>>> s4 = [3, "abc",4,"c"]
>>> print(s1==s2,s1!=s2,s1>s2,s1<s2)
False True False True
```

s1、s2、s3和s4的大小关系？ s3<s1<s4<s2

列表的统计操作

- 通过len函数求列表的长度

  l = len(list)

- 通过max、min函数求列表元素中的最大值、最小值

  ma = max(list)

  mi = min(list)

```
>>> list4 = ['aab', 3, -2, True]
>>> len(list4)
4
>>> list5 = []
>>> len(list5)
0
>>> list7 = [0,8,-2,3,-10,1]
>>> max(list7)
8
>>> min(list7)
-10
>>> list6 = ['c','f','E','H']
>>> print(max(list6),min(list6))
f E
```

列表的统计操作

- 通过count()方法求值出现的次数

n = list.count(x)

```
>>> list3 = [1,0,'a',1,-2,False,-1,1,1,0]
>>> list3.count(1)
4

>>> list3.count("a")
1

>>> list3.count(2)
0

>>> list3.count(True)
4

>>> list3.count(False)
3
```

列表的布尔操作

- 通过all()函数判断列表中元素的布尔值是否全为真

all(list)

```
>>> list2 = [1,'a',-1,True]
>>> all(list2)
True

>>> list3 = ["",1]
>>> all(list3)
False

>>> list4 = [1, 0]
>>> all(list4)
False

>>> list5 = []
>>> all(list5)
True
```

列表的布尔操作

- 通过any()函数判断列表中元素的布尔值是否有真

any(list)

```
>>> list2 = [0,"",False,[]]
>>> any(list2)
False

>>> list3 = [0, 1]
>>> any(list3)
True
```

```
>>> list4 = []
>>> any(list4)
False

>>> list5=[]
>>> all(list5)
True
>>> any(list5)
False
```

列表的布尔操作

```
>>> list5=[]
>>> all(list5)
True
>>> any(list5)
False
```

```
def all(iterable):
    for element in iterable:
        if not element:
            return False
    return True
```

```
def any(iterable):
    for element in iterable:
        if element:
            return True
    return False
```

https://docs.python.org/3/library/functions.html#all

列表的排序操作

- 通过sort()方法对列表元素进行排序，并修改列表

  list.sort(key = None, reverse = False)

```
>>> list1 = [0,8,-2,3,-5]
>>> list1.sort()

>>> list1
[-5, -2, 0, 3, 8]
```

```
>>> list1 = [0,8,-2,3,-5]
>>> list1.sort(reverse=True)

>>> list1
[8, 3, 0, -2, -5]
```

列表的排序操作

- 通过参数key修改排序依据

  list.sort(key = func, reverse = False)

```
def k1( x ):
    return( x * x - 4 * x + 4 )
```

```
>>> s=[1, 2, 5, 4, 3]
>>> s.sort(key=k1)
>>> s
[2, 1, 3, 4, 5]
```

```
for i in s:
    print("i=",i,"k1(i)=",k1(i))
```

```
i= 1 k1(i)= 1
i= 2 k1(i)= 0
i= 5 k1(i)= 9
i= 4 k1(i)= 4
i= 3 k1(i)= 1
```

列表的排序操作

- 通过sorted()函数对列表元素进行排序，返回新列表

sorted(list, key = None, reverse = False)

```
>>> list1 = [0,8,-2,3,-5]
>>> sorted(list1)
[-5, -2, 0, 3, 8]
>>> list1
[0, 8, -2, 3, -5]
>>> sorted(list1,key=k1,reverse=True)
[-5, 8, -2, 0, 3]
```

```
for i in list1:
    print("i=",i,"k1(i)=",k1(i))
```

```
i= 0 k1(i)= 4
i= 8 k1(i)= 36
i= -2 k1(i)= 16
i= 3 k1(i)= 1
i= -5 k1(i)= 49
```

列表的反序操作

- 通过reverse()方法对列表元素进行反序操作，并修改列表

  list.reverse()

```
>>> list1 = [1, 'a', 3, -2, 'ccd']
>>> list1.reverse()
>>> list1
['ccd', -2, 3, 'a', 1]
```

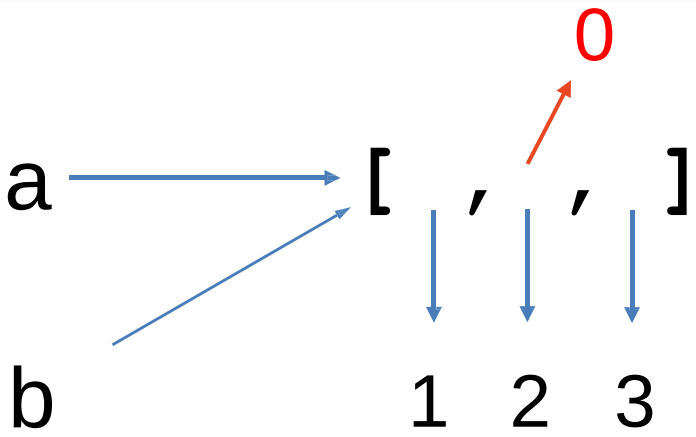
列表的复制

```
>>> a[1]=0
>>> id(a[1]),id(b[1])
(1609861568, 1609861568)
```

- 直接赋值

list2 = list1

```
>>> a = [1, 2, 3]
>>> b = a
>>> b
[1, 2, 3]
>>> a[1] = 0
>>> a
[1, 0, 3]
>>> b
[1, 0, 3]
```



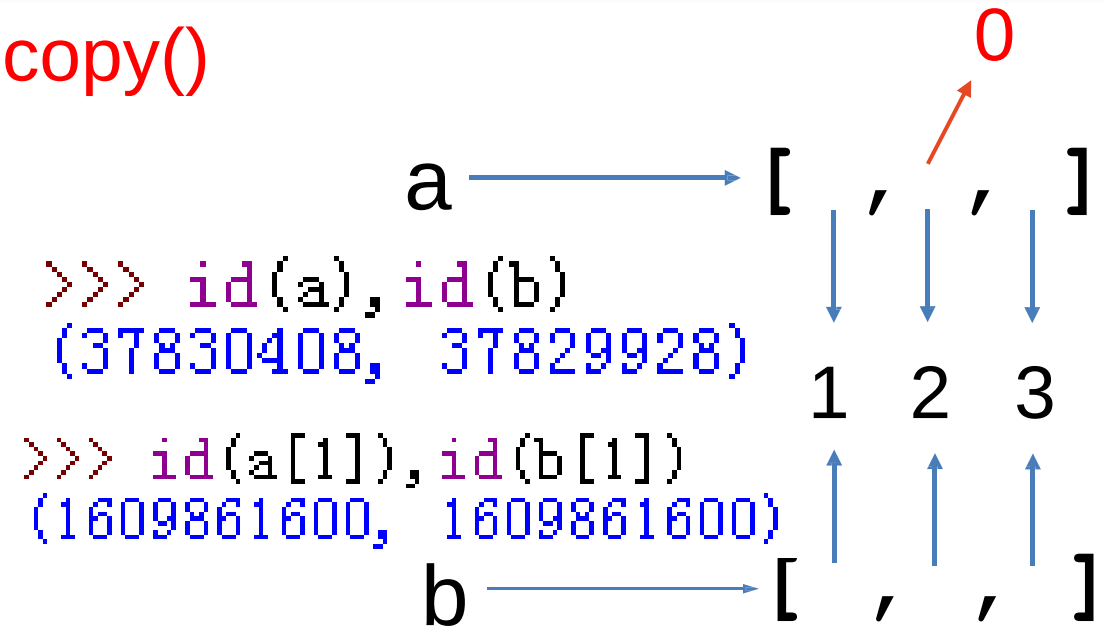

```
>>> id(a),id(b)
(37830608, 37830608)
>>> id(a[1]),id(b[1])
(1609861600, 1609861600)
```

列表的复制

- 浅复制　　list2 = list1.copy()

```
>>> a = [1, 2, 3]
>>> b = a.copy()
>>> b
[1, 2, 3]
>>> a[1]=0
>>> a
[1, 0, 3]
>>> b
[1, 2, 3]
```

```
>>> a[1]=0
```




```
>>> id(a),id(b)
(37830408, 37829928)
>>> id(a[1]),id(b[1])
(1609861600, 1609861600)
```

```
>>> id(a[1]),id(b[1])
(1609861568, 1609861600)
```

列表的复制

- 浅复制　　list2 = list1.copy()

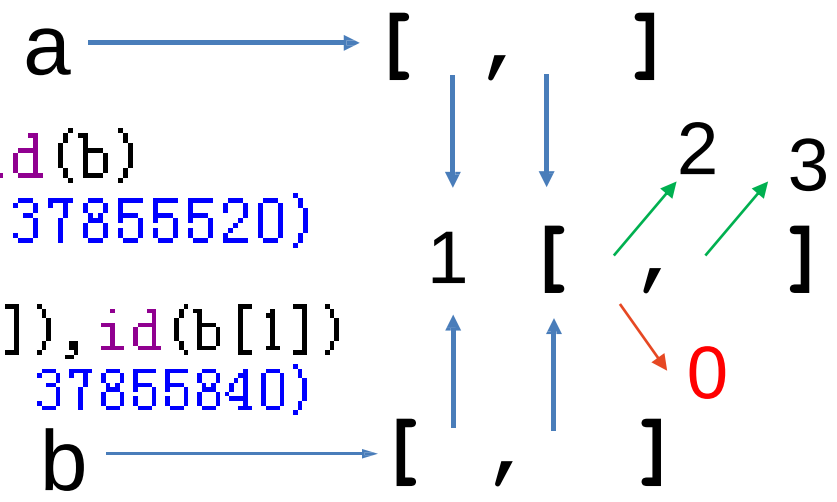
```
>>> a = [1, [2,3]]
>>> b = a.copy()
>>> b
[1, [2, 3]]

>>> a[1][0] = 0
>>> a
[1, [0, 3]]
>>> b
[1, [0, 3]]
```

```
>>> id(a[1][0]),id(b[1][0])
(1609861568, 1609861568)
```

```
>>> a[1][0] = 0
```

```
>>> id(a),id(b)
(37829968, 37855520)
```

```
>>> id(a[1]),id(b[1])
(37855840, 37855840)
```




```
>>> id(a[1][0]), id(b[1][0])
(1609861600, 1609861600)
```

多重列表乘法操作存在的问题

```
>>> x = [[1,2],3]
>>> y = x*3
>>> y
[[1, 2], 3, [1, 2], 3, [1, 2], 3]

>>> x[0][1]=5
>>> x
[[1, 5], 3]
>>> y
[[1, 5], 3, [1, 5], 3, [1, 5], 3]
```

#四个元素[1,2]是对同一个子列表的引用

#通过修改子列表可以同时修改四个元素的值（修改y[0][1]效果相同）

列表的复制

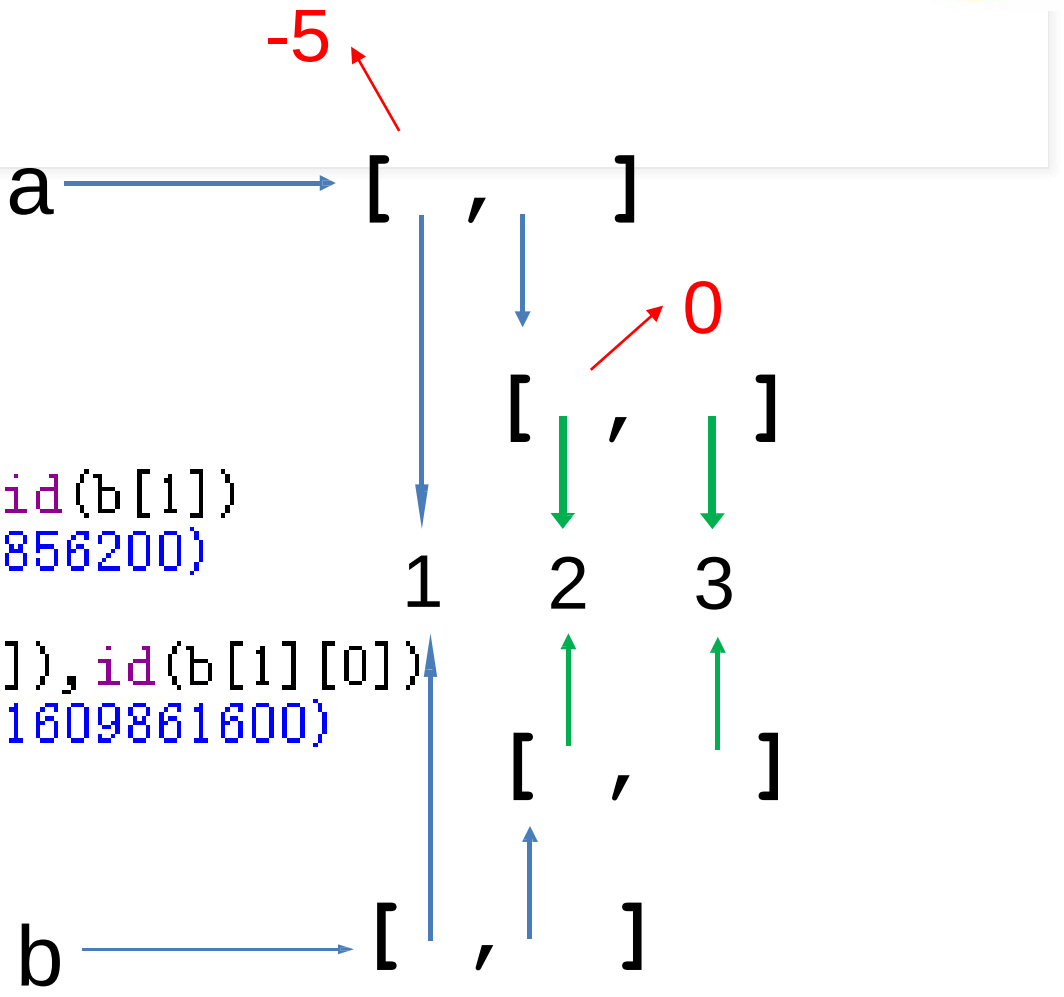
- 深复制 *import copy*

  list2 = copy.deepcopy(list1)

```
>>> import copy
>>> a = [1,[2,3]]
>>> b = copy.deepcopy(a)
>>> b
[1, [2, 3]]
>>> a[0],a[1][0] = -5, 0
>>> a
[-5, [0, 3]]
>>> b
[1, [2, 3]]
```

```
>>> id(a[1]),id(b[1])
(37856080, 37856200)
>>>
>>> id(a[1][0]),id(b[1][0])
(1609861600, 1609861600)
```

```
>>> a[0],a[1][0] = -5, 0
```




```
>>> id(a[1][0]),id(b[1][0])
(1609861568, 1609861600)
```

列表常用方法总结

函数或方法	描述
list.append(x)	将x添加到列表末尾，列表长度加1
list.insert(i,x)	在序号为i的位置插入对象x，i以后的元素后移
list.extend(t)	把可迭代对象t附加到s的尾部
list.copy()	复制出一个新列表，返回给调用者，复制过程为浅拷贝
del list[i]	删除序号为i的元素，i以后的元素前移
list.pop[(i])	删除列表末尾或序号为i的位置的元素，返回该元素的值
list.remove(x)	删除列表中第一次出现的x值
list.clear()	清空列表元素
del list	删除列表
list.sort()	对列表排序，修改列表
sorted(list)	对列表排序，生成新列表
list.reverse()	将列表反序，修改列表

列表解析（列表推导式）

通过对一个或多个列表中的每个元素应用某种操作，从而快速将一个或多个列表映射为另一个列表

[exp for v1 in s1 (for v2 in s2) … …(if exp1 if exp2)… …]

```
>>> x = [1, 2, 3]
>>> y = [2, 5, 4]

>>> [a*b for a in x for b in y if a<b]
[2, 5, 4, 10, 8, 15, 12]
```

列表解析（列表推导式）

```
>>> list1 = [x**2 for x in range(5)]
>>> list1
[0, 1, 4, 9, 16]

>>> list2 = [x for x in range(3,30,2) if x%3 == 0]
>>> list2
[3, 9, 15, 21, 27]

>>> list1 = [ i**2 if i<5 else i*2 for i in range(10)]
>>> list1
[0, 1, 4, 9, 16, 10, 12, 14, 16, 18]
```

列表解析（列表推导式）

```
>>> x,y = [1,3,5], ['b','d','a']
>>> list2 = [str(i)+j for i in x for j in y]
>>> list2
['1b', '1d', '1a', '3b', '3d', '3a', '5b', '5d', '5a']

>>> list3 = [[j*2,i] for i in x for j in y if i>1]
>>> list3
[['bb', 3], ['dd', 3], ['aa', 3], ['bb', 5], ['dd', 5], ['aa', 5]]

>>> mat = [[1,2,3], [4,5,6], [7,8,9]]
>>> list3 = [[row[i] for row in mat] for i in (0,1,2)]
>>> list3
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```