1. **进一步学习列表**

**第一部分**

**一、列表常用方法**

|  |  |
| --- | --- |
| list.append(x) | 把一个元素添加到列表的结尾，相当于 a[len(a):] = [x]。 |
| list.extend(L) | 通过添加指定列表的所有元素来扩充列表，相当于 a[len(a):] = L。 |
| list.insert(i, x) | 在指定位置插入一个元素。第一个参数是准备插入到其前面的那个元素的索引，例如  a.insert(0, x) 会插入到整个列表之前，而 a.insert(len(a), x) 相当于 a.append(x) 。 |
| list.remove(x) | 删除列表中值为 x 的第一个元素。如果没有这样的元素，就会返回一个错误。 |
| list.pop([i]) | 从列表的指定位置删除元素，并将其返回。如果没有指定索引，a.pop()返回最后一个元素。  元素随即从列表中被删除。（方法中 i 两边的方括号表示这个参数是可选的，而不是要求你输入  一对方括号，你会经常在 Python 库参考手册中遇到这样的标记。） |
| list.clear() | 移除列表中的所有项，等于del a[:]。 |
| list.index(x) | 返回列表中第一个值为 x 的元素的索引。如果没有匹配的元素就会返回一个错误。 |
| list.count(x) | 返回 x 在列表中出现的次数。 |
| list.sort() | 对列表中的元素进行排序。 |
| list.reverse() | 倒排列表中的元素。 |
| list.copy() | 返回列表的浅复制，等于a[:]。 |

**二、将列表当做堆栈使用**

列表方法使得列表可以很方便的作为一个堆栈来使用，堆栈作为特定的数据结构，最先进入的元素最后一个被释放（后进先出:Last In First Out ).

用 append() 方法可以把一个元素添加到堆栈顶。用不指定索引的 pop() 方法可以把一个元素从堆栈顶释放出来。

*#encoding=utf-8*list1=[1,2,3,4,5,6,7,8]  
list1.append(9)  
print(list1)  
list1.pop()  
print(list1)

**三、将列表当作队列使用**

First Input First Output的缩写，先入先出队列

也可以把列表当做队列用，只是在队列里第一加入的元素，第一个取出来；但是拿列表用作这样的目的效率不高。在列表的最后添加或者弹出元素速度快，然而在列表里插入或者从头部弹出速度却不快（因为所有其他的元素都得一个一个地移动）。

from collections import deque

a=['rose','tom','appr','wes']

queue=deque(a)

queue.append("Lily")

print(queue)

queue.appendleft("dfer")

print(queue)

print(queue.popleft())

print(queue)

print(queue.pop())

print(queue)

执行结果：

deque(['rose', 'tom', 'appr', 'wes', 'Lily'])

deque(['dfer', 'rose', 'tom', 'appr', 'wes', 'Lily'])

dfer

deque(['rose', 'tom', 'appr', 'wes', 'Lily'])

Lily

deque(['rose', 'tom', 'appr', 'wes'])

在列表的最后添加或者弹出元素，执行的速度比较快。

from collections import deque

a=['rose','tom','appr','wes']

queue=deque(a)

queue.append("Lily")

print(queue)

print(queue.pop())

print(queue)

**第二部分**

**四、列表推导式**

列表推导式提供了从序列创建列表的简单途径。通常应用程序将一些操作应用于某个序列的每个元素，用其获得的结果作为生成新列表的元素，或者根据确定的判定条件创建子序列。

每个列表推导式都在 for 之后跟一个表达式，然后有零到多个 for 或 if 子句。返回结果是一个根据表达从其后的 for 和 if 上下文环境中生成出来的列表。如果希望表达式推导出一个元组，就必须使用括号。

1、创建一个新列表，新列表是旧列表的所有元素的平方

a=[1,2,3,4]

print [x\*\*2 for x in a]

[1,4,9,16]

2、a=[1,2,3,4]

b=[34,56,78,12]

c = [x+y for x in a for y in b]

print(c)

[35, 57, 79, 13, 36, 58, 80, 14, 37, 59, 81, 15, 38, 60, 82, 16]

3、a=[1,2,3,4]

print [[x\*\*2 ]for x in a]

[[1], [4], [9], [16]]

4、if语句过滤

a=[1,2,3,4,67,45,23]

print [[x\*\*2]for x in a if x<30]

[[1], [4], [9], [16], [529]]

5、可调用列表元素的方法

freshfruit = [' banana', ' loganberry ', 'passion fruit ']

# strip() 方法用于移除字符串头尾指定的字符

print [weapon.strip() for weapon in freshfruit]

5、矩阵操作

#通过列表的嵌套，定义一个矩阵

matrix = [

[1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

]

#打印矩阵

print(matrix)

#遍历矩阵

for row in matrix:

for j in row:

print(j)

print("#######")

#遍历矩阵

for row in matrix:

for i in range(4):

print(row[i])

print("$$$$$$$$$$$$$")

#矩阵转化

for i in range(4):

for row in matrix:

print(row[i])

print("$$$$$$$$$$$$$")

6、在序列中遍历时，索引位置和对应值可以使用 enumerate() 函数同时得到：

a=[23,14,27,73,67,90,24]

for i,v in enumerate(a):

print(i,v)

(0, 23)

(1, 14)

(2, 27)

(3, 73)

(4, 67)

(5, 90)

(6, 24)

7、同时遍历两个或更多的序列，可以使用 zip() 组合

questions = ['name', 'quest', 'favorite color']

answers = ['lancelot', 'the holy grail', 'blue']

for q, a in zip(questions, answers):

print('What is your {0}? It is {1}.'.format(q, a))

What is your name? It is lancelot.

What is your quest? It is the holy grail.

What is your favorite color? It is blue.

8、要反向遍历一个序列，首先指定这个序列，然后调用 reversed() 函数：

questions = ['name', 'quest', 'favorite color']

for t in reversed(questions):

print(t)