程序设计（Python）

列 表

主讲：数据与目标工程学院 胡瑞娟 副教授

【案例】无人机目标识别，假设你是一名无人机操作员，正在执行一项侦察任务。任务目标是确认一个可疑区域内的敌方坦克数量。根据情报，坦克数量在1到100之间。由于战场环境的复杂性（如伪装、遮挡等），无人机需要多次侦察才能获得准确数字。

模拟过程：

（1）紧急侦察模式，侦察1次：每次侦察会传回一个估计数字：系统会根据真实数字给出反馈：“侦察结果偏低”、“侦察结果偏高”或“目标确认！”。

（2）标准侦察模式，限定次数n：由于无人机每次侦察都有被击落的风险，所以任务需要在有限次数内完成。

（3）持续侦察模式，不限定次数：直到确认目标为止，模拟不惜代价获取情报 。

（4）备用方案侦察模式，复活版：无人机有备用机或重新部署的机会，当无人机被击落（次数用尽）时，调用备用无人机（复活）继续侦察，重置侦察次数。

循环结构

案例引入

【案例背景】无人机侦察数据管理。在现代化军事行动中，无人机侦察系统负责收集敌方区域的各种情报数据。无人机每次执行侦察任务后，会传回大量不同类型的数据至指挥中心，包括**目标坐标位置、敌方装备数量、地形海拔高度、侦察时间戳、目标类型标识**等；指挥中心需要对这些数据进行存储、管理和分析，为指挥决策提供支持。

【任务】编写程序来**管理**这些数据，并实现以下功能：

1.**存储**侦察到的目标数据
2.**添加**新的侦察数据
3.**修改**目标数据
4.**删除**已确认处理的目标数据
5.**统计**不同类型目标的数量
6.**找出**某个区域的目标
7.对目标按坐标进行**排序**

【案例分析】问题1：**存储**侦察到的目标数据?

- **目标坐标位置** [120, 35], [125, 38], [118, 32],[122,40]
- **敌方装备数量** 5, 3, 8, 6
- **地形海拔高度** 5039, 5127, 5238, 5299
- **侦察时间戳** '08:30', '10:15', '13:45', '16:20'
- **目标类型标识** 'radar', 'missile', 'tank', 'barracks'

如何用一个变量存放一组数据?

组合类型数据

一组军事装备、军事目标、体能成绩、学员管理……

01 组合类型数据

1. 组合数据类型

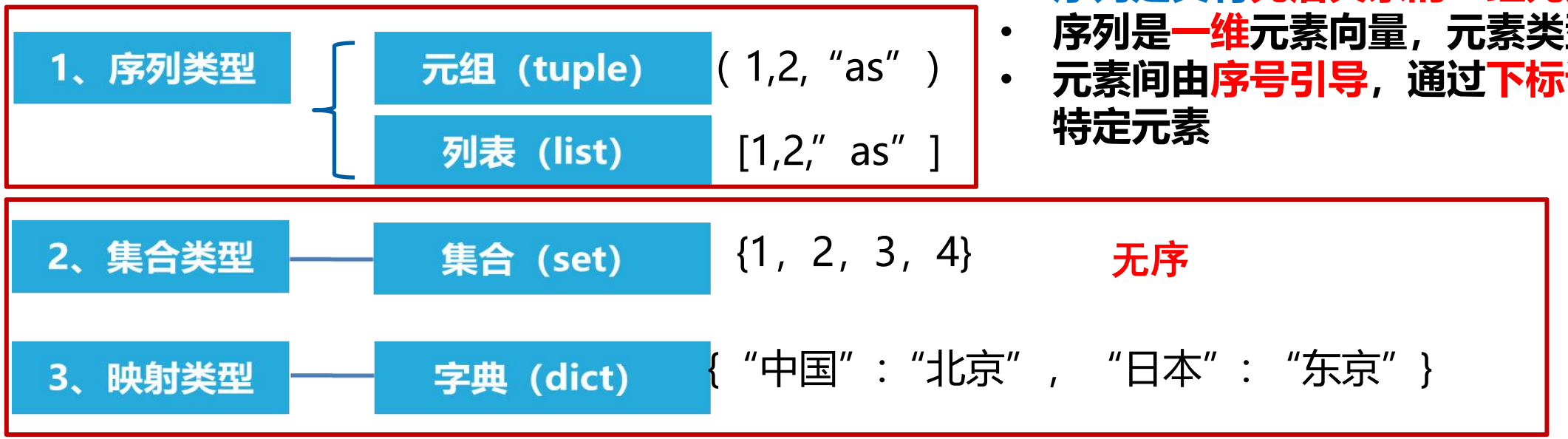
- **Python数据类型分为：**
- **基本数据类型：**
  - **数字类型（整数（int）、浮点数（float））**
  - **布尔类型（bool）**
  - **字符串类型(str)**
- **组合数据类型：**
  - **集合、元组、列表、字典**

数据是运算的核心
类型的不同决定着可以进行的**操作不同**

类型	示例
数字	1234, 3.14, 3+4j
字符串	'swfu', "I'm student", '"Python "'
布尔型	True, False
集合	set('abc'), {'a', 'b', 'c'}
元组	(2, -5, 6, '3')
列表	[1, 2, 3, '3']
字典	{1:'food' ,2:'taste', '3':'import'}
文件	f=open('data.dat', 'r')
空类型	None
编程单元类型	函数、模块、类

1. 组合数据类型

分类：序列类型、集合类型、映射类型

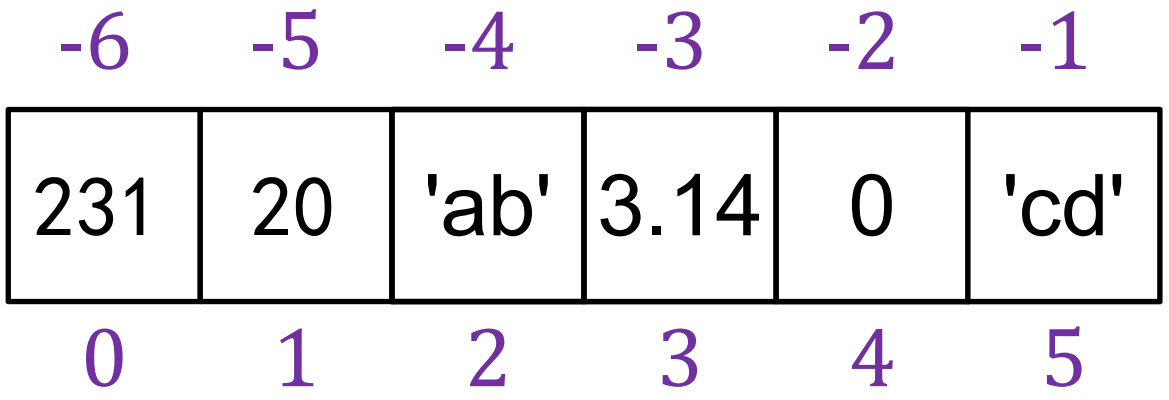


- 序列是具有先后关系的一组元素，有序
- 序列是一维元素向量，元素类型可以不同
- 元素间由序号引导，通过下标访问序列的特定元素

02

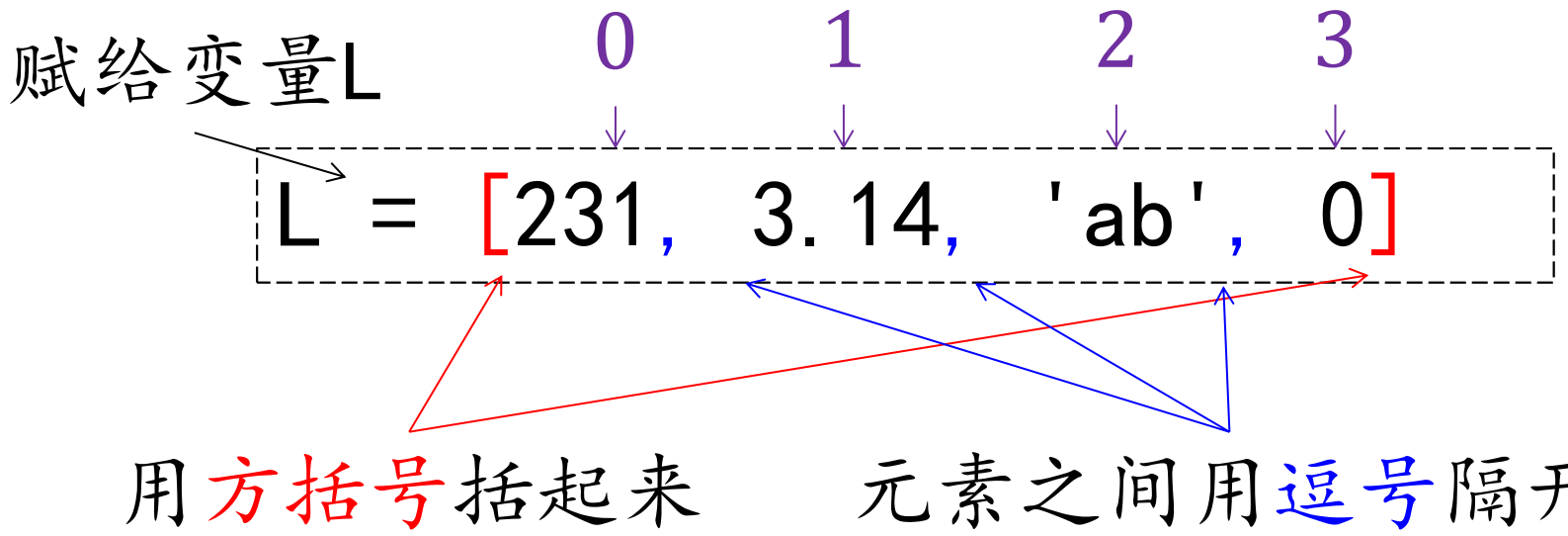
列表

- list
  - 包含若干元素
  - 元素是按序编号的（**从0开始**）
  - 利用编号可对元素进行增删改查等操作



创建列表

列表必须有元素吗？



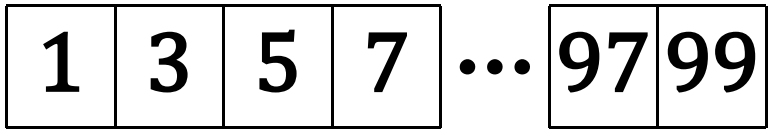
例：请创建如下列表，并打印出来

20	24	'P'	'Y'	11.19

```
L = [20, 24, 'P', 'Y', 11.19]
print(L)
```

- 创建列表
- L=[ ]
  - range()：用于生成整数等差数列
  - list()：转换成列表类型

例：请创建如下列表，并打印出来




```
L = list(range(1,100,2))
print(L)
```

【案例解决】问题1：**存储**侦察到的目标数据?

```
# 列表案例-问题1：创建侦察数据列表
target_coordinates = [[120, 35], [125, 38], [118, 32],[122,40]]  # 目标坐标(x,y)
enemy_equipment = [5, 3, 8, 6]  # 各区域敌方装备数量
terrain_elevation = [5039, 5127, 5238, 5299]  # 地形海拔高度(米)
recon_time = ['08:30', '10:15', '13:45', '16:20']  # 侦察时间
target_types = ['radar', 'missile', 'tank', 'barracks']  # 目标类型

print("侦察数据初始化完成")
print(f"目标坐标: {target_coordinates}")
print(f"装备数量: {enemy_equipment}")
```

【案例分析】问题2-4?

1.**存储**侦察到的目标数据

2.**添加**新的侦察数据

3.**修改**目标数据

4.**删除**已确认处理的目标数据

5.**统计**不同类型目标的数量

6.**找出**某个区域的目标

7.对目标按坐标进行**排序**

读取元素

- x=**L[i]** ：读取第i个元素
- x=**L[i:j]** ：读取第i到第j-1个元素
- x=**L[i:j:s]** ：第i到第j-1每隔s取个元素
- i、j可为负数

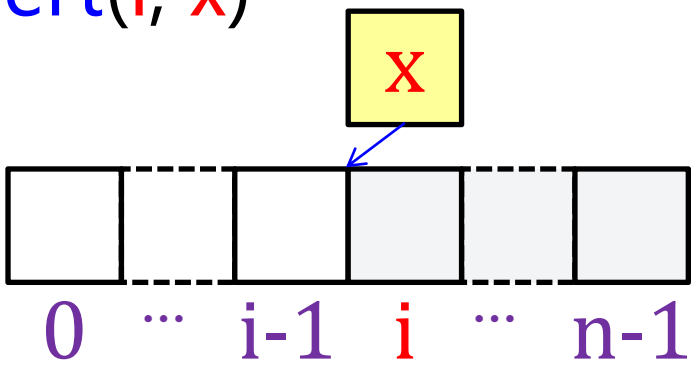
-6	-5	-4	-3	-2	-1
'P'	'y'	't'	'h'	'o'	'n'
0	1	2	3	4	5

```
L = ['P', 'y', 't', 'h', 'o', 'n']
a = L[4]      #'o'
b = L[0:3]    #['P', 'y', 't']
c = L[0:4:2]#['P', 't']
d = L[-1]
```

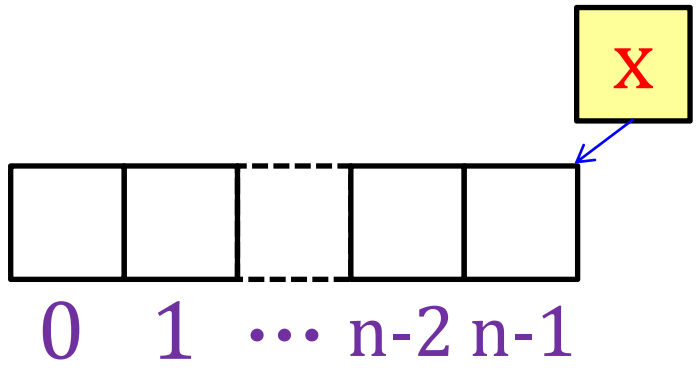
```
L[1] = 'Y'
L[2:5] = ['T', 'H', 'O']
L[-1] = 'N'
print(L)
```

添加元素

- L.insert(i, x)

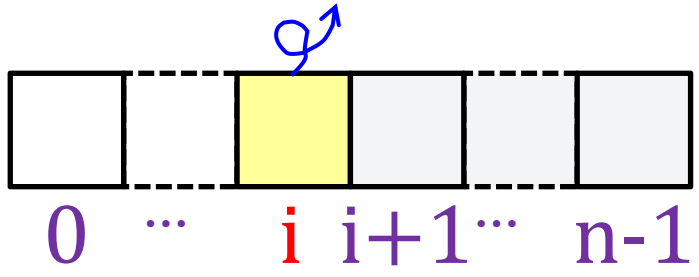


- L. append(x)

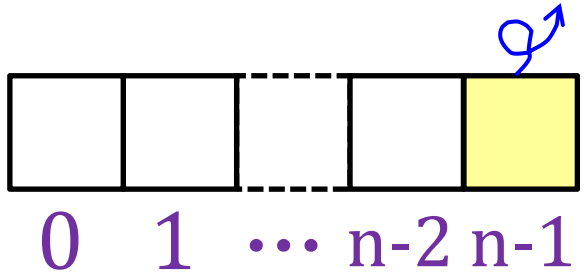


删除元素

- L.pop(i)



- L. pop()



添加/删除元素

- L.insert(i, x) ： 将x添加到L的第i处
- L.append(x) ： 将x添加到L的尾部
- L.pop(i) ： 删除L的第i个元素
- L.pop() ： 删除L最后一个元素

```
L = ['m', 'i', 'l', 'e']
L.insert(0, 's')
L.append('s')
print(L)
```

```
L = ['s', 'm', 'i', 'l', 'e', 's']
L.pop(0)
L.pop()
print(L)
```

【案例解决】问题2-4

2. **添加**新侦察到的目标
(1)目标坐标：在末尾添加[130,40]
(2)装备数量：在第三个位置插入7

3.**修改**
(1)海拔高度：第二个修改为5150
(2)目标类型：第一个改为’radar_station’

4.删除
(1)删除第4个装备数量
(2)删除最后一个侦察时间

```
# 列表案例-问题1：创建侦察数据列表
target_coordinates = [[120, 35], [125, 38], [118, 32],[122,40]]  # 目标坐标(x,y)
enemy_equipment = [5, 3, 8, 6]  # 各区域敌方装备数量
terrain_elevation = [5039, 5127, 5238, 5299]  # 地形海拔高度(米)
recon_time = ['08:30', '10:15', '13:45', '16:20']  # 侦察时间
target_types = ['radar', 'missile', 'tank', 'barracks']  # 目标类型
```

```
# 添加新侦察到的目标
target_coordinates.append([130, 40])
enemy_equipment.insert(2, 7)  # 在索引2处插入新数据

# 修改错误数据
terrain_elevation[1] = 5150  # 修正第二个海拔数据
target_types[0] = 'radar_station'  # 更新目标类型名称

# 删除无效数据
invalid_equipment = enemy_equipment.pop(3)  # 删除索引3的数据
last_time = recon_time.pop()  # 删除最后一个时间

print(f"更新后目标坐标: {target_coordinates}")
print(f"修正后海拔数据: {terrain_elevation}")
```

```
更新后目标坐标: [[120, 35], [125, 38], [118, 32], [122, 40], [130, 40]]
修正后海拔数据: [5039, 5150, 5238, 5299]
```

2.2 列表操作元素

【案例分析】问题5-7?

```
# 列表案例-问题1：创建侦察数据列表
target_coordinates = [[120, 35], [125, 38], [118, 32],[122,40]]  # 目标坐标(x,y)
enemy_equipment = [5, 3, 8, 6]  # 各区域敌方装备数量
terrain_elevation = [5039, 5127, 5238, 5299]  # 地形海拔高度(米)
recon_time = ['08:30', '10:15', '13:45', '16:20']  # 侦察时间
target_types = ['radar', 'missile', 'tank', 'barracks']  # 目标类型
```

1.**存储**侦察到的目标数据

2.**添加**新的侦察数据

3.**修改**目标数据

4.**删除**已确认处理的目标数据

5.**统计**不同类型目标的数量

6.**找出**某个区域的目标

7.对目标按坐标进行**排序**

其它操作

- L.**index**(x) ： 返回L中首个x的编号
- L.**remove**(x) ： 删除L中首个x
- L.**count**(x) ： 返回L中x出现的次数

```
students=['jack','tom','jack','amy','kim','sunny']
print(students.count('jack'))
students.remove('jack')
print(students.index('jack'))
```

- 其它操作
  - x **in** L ： 判断x是否为L中的元素
- 判断两个列表之间大小?
  - 逐个元素比较，根据第一次遇到的不同元素之间的关系判定大小
  - 字符串比较规则与列表相同（大写字母较小）

```
[1,2,10]          >   [1,2,3,4,5,6,7,8]
'abc'             >   'ABCD'
list(range(100))  <   list(range(1,2))
'11'              ?   '8'
```

其它操作

- L.**reverse**() ：逆序排列（原顺序）
- L.**sort**() ：升序排列（比大小）【注意：原地排序】
- **sorted**(L) ：升序排列【注意：生成新列表】

```
students=['jack','tom','jack','amy','kim','sunny']
students.reverse()
print(students)
students.sort()
print(students)
```

```
['sunny', 'kim', 'amy', 'jack', 'tom', 'jack']
['amy', 'jack', 'jack', 'kim', 'sunny', 'tom']
```

其它操作

- L.**extend**(L2) :扩展列表，在末尾追加列表
- L.**clear**() :清空列表
- L.**copy**() :复制列表

```
students = ['jack', 'tom','kim', 'sunny']
L2 = ['lily', 'bob']
students.extend(L2)
print(students)
students.clear()
print(students)
```

```
['jack', 'tom', 'kim', 'sunny', 'lily', 'bob']
[]
```

其它操作

- L1 + L2 ：将L1和L2拼接起来
- L * n ：将n个L拼接起来
- **len**(L) ：返回L的长度（元素个数）

```
L1 = ['jack', 'tom', 'amy']
L2 = ['kim', 'sunny']
print(L1+L2)
print(L2*3)
print(len(L1))
```

```
['jack', 'tom', 'amy', 'kim', 'sunny']
['kim', 'sunny', 'kim', 'sunny', 'kim', 'sunny']
3
```

其它操作

- **max**(L)　　：返回L中最大的元素
- **min**(L)　　：返回L中最小的元素
- **sum**(L)　　：返回L中所有元素之和

```
numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
print(max(numbers))
print(min(numbers))
print(sum(numbers))
```

```
9
1
44
```

2.3 列表操作

L, x = [231, 3.14, 41, 13, 0], 13

- a = L.**extend**(L)
- a = x **in** L
- a = L.**count**(x)
- L.**remove**(x)
- L.**reverse**()
- L.**sort**()
- a = L + L
- a = L * 3
- a = len(L)
- a = max(L)
- a = min(L)
- a = sum(L)

L是列表，x表示元素，a为某类型变量

【举一反三】代码运行后，x值为多少?

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

A [2,1,2,1,1,1]

B [2,2]

C [2,2,1]

D [2,2,1,1]

【**易错点分析**】:删除元素元素前移，但访问的序号增加，导致**删除的元素的后面的第一个元素没有被访问**。如何解决?

- 解题思路一：

访问和删除的列表**分开**

```
ls=[1, 2, 1, 2, 1, 1, 1]
ls1=ls.copy()  #复制列表
#删除列表中的1
for i in  ls1  :
    if i==1:
        ls  .remove(i)
```

```
ls: [2, 2]
ls1: [1, 2, 1, 2, 1, 1, 1]
```

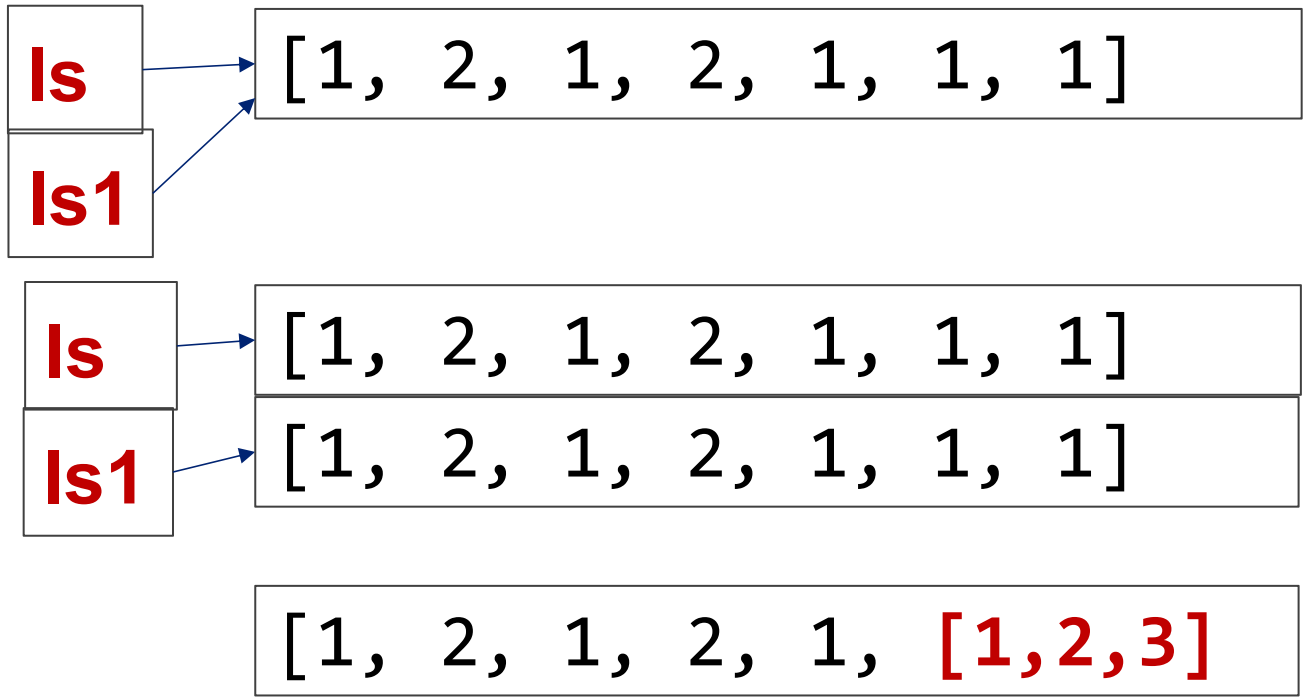
- 解题思路二：

从后面**倒着**删除

```
ls=[1, 2, 1, 2, 1, 1, 1]

#删除列表中的1
for i in  range(len(ls)-1,-1,-1):
    if ls[i]  ==1:
        ls.remove(ls[i] )
```

?

- 赋值复制
  - ls1=ls
- 浅复制
  - ls1=ls[:]
  - ls1=ls.copy()
- 深复制
  - ls2=copy.deepcopy(ls)



ls[5][1]=10

ls:
ls1:浅拷贝
ls2:深拷贝

```
[1, 2, 1, 2, 1, [1, 10, 3]]
[1, 2, 1, 2, 1, [1, 10, 3]]
[1, 2, 1, 2, 1, [1, 2, 3]]
```

- 解题思路三：把不是1的**另外**保存

```
ls=[1, 2, 1, 2, 1, 1, 1]
#删除列表中1
new_list=[]    #新建空列表
for i in ls:
    if  i!=1:
        new_list.append(i) #追加到新列表
```

【举一反三】代码运行后，x2值为多少?

A [2,1,2,1,1,1]

B [2,2]

C [2,2,1]

D [2,2,1,1]

```
x1 = [1, 2, 1, 2, 1, 1, 1]
x2 = []
for i in x1:
    if i != 1:
        x2.append(i)
```

- 计算1+2+3+…+100的值

=5050

```
L = list(range(1,101))
s = sum(L)
print(s)
```

- 求L中最大的两个数的和

=229

```
L = [101, 25, 38, 29, 108, 121]
L.sort()
s = L[-1]+L[-2]
print(s)
```

- 列表ID中存放学号，Name中存放对应的姓名，输入学号，若存在则打印姓名，否则提示'学号不存在'

```
ID = [1, 2, 3, 4, 5]
Name = ['zhao', 'qian', 'sun', 'li', 'zhou']
x = eval(input('学号: '))
if x in ID:
    print('姓名: ', Name[ID.index(x)])
else:
    print('学号不存在')
```

【案例解决】问题5-7

5.统计不同类型目标的数量
(1)统计最大装备数、最小装备数
(2)统计装备总数

6.找出某个区域的目标
(1)查询特定目标位置：查找目标[125,38]在列表的第几个位置

7.对目标按坐标进行排序
(1)按装备数量升序排列
(2)按装备数量降序排列

```
最大装备数: 7, 最小装备数: 3
装备总数: 21
目标位于列表第1个位置
装备数量排序: [3, 5, 6, 7]
```

```python
# 列表案例-问题1：创建侦察数据列表
target_coordinates = [[120, 35], [125, 38], [118, 32],[122,40]]  # 目标坐标(x,y)
enemy_equipment = [5, 3, 8, 6]  # 各区域敌方装备数量
terrain_elevation = [5039, 5127, 5238, 5299]  # 地形海拔高度(米)
recon_time = ['08:30', '10:15', '13:45', '16:20']  # 侦察时间
target_types = ['radar', 'missile', 'tank', 'barracks']  # 目标类型
```

```python
# 统计装备数据
max_equipment = max(enemy_equipment)
min_equipment = min(enemy_equipment)
total_equipment = sum(enemy_equipment)
print(f"最大装备数: {max_equipment}, 最小装备数: {min_equipment}")
print(f"装备总数: {total_equipment}")
```

```python
# 查询特定目标位置
if [125, 38] in target_coordinates:
    position = target_coordinates.index([125, 38])
    print(f"目标位于列表第{position}个位置")
```

```python
# 数据排序分析
sorted_equipment = sorted(enemy_equipment)  # 升序排列
reverse_sorted = sorted(enemy_equipment, reverse=True)  # 降序排列
print(f"装备数量排序: {sorted_equipment}")
```

【举一反三】计算L中所有元素的平方和

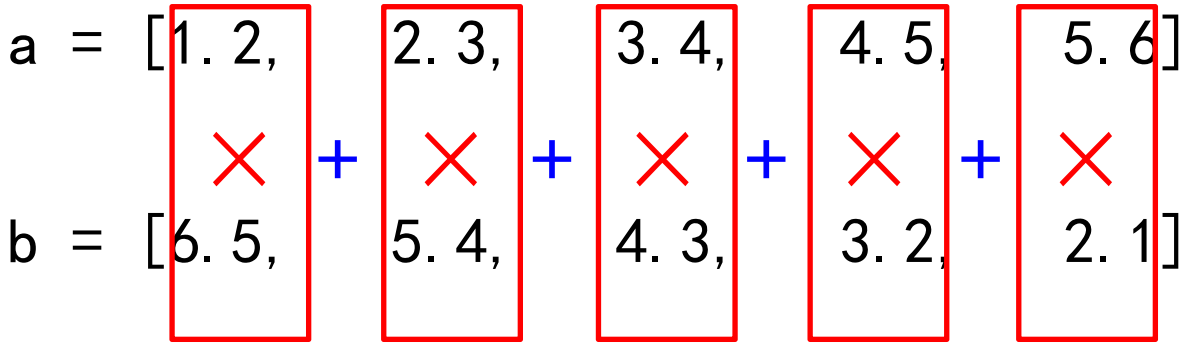
```
L = [101, 25, 38, 29, 108, 121]
i, s = 0, 0
while i<len(L):
    s=s+L[i]**2
    i=i+1
print(s)
```

```
L = [101, 25, 38, 29, 108, 121]
s = 0
for x in L:
    s=s+x**2
print(s)
```

思考：for循环如何用索引访问元素?

```
# 使用 range(len(L)) 生成索引序列
for i in range(len(L)):
    s = s + L[i]**2
```

【举一反三】利用for循环计算向量a、b的内积




```
a = [1.2, 2.3, 3.4, 4.5, 5.6]
b = [6.5, 5.4, 4.3, 3.2, 2.1]
m = 0
for i in range(len(a)):
    m = m+a[i]*b[i]
print(m)
```

2.4 列表遍历

【案例解决】遍历

```
=== 各区域威胁评估 ===
区域1: 5个装备, 威胁值: 50
区域2: 3个装备, 威胁值: 30
区域3: 7个装备, 威胁值: 70
区域4: 2个装备, 威胁值: 20
区域5: 6个装备, 威胁值: 60
```

```
=== 高海拔区域分析 ===
高海拔区域: 5238米
高海拔区域: 5259米
高海拔区域: 5299米
总共3个高海拔区域
```

为了评估战场情况，我们需要：

(1)对每个区域的威胁水平进行评估（威胁值 = 装备数量 × 10）。

(2)统计高海拔区域（海拔高于5200米）的数量。

```python
print("\n=== 高海拔区域分析 ===")

# 筛选出所有高海拔区域
high_elevations = [elev for elev in terrain_elevation if elev > 5200]
high_elevation_count = len(high_elevations)

# 打印每个高海拔区域
for elevation in high_elevations:
    print(f"高海拔区域: {elevation}米")

print(f"总共{high_elevation_count}个高海拔区域")
```

```python
# 方法1: while循环遍历
print("=== 各区域威胁评估 ===")
i = 0
while i < len(enemy_equipment):
    threat_level = enemy_equipment[i] * 10  # 威胁系数计算
    print(f"区域{i+1}: {enemy_equipment[i]}个装备, 威胁值: {threat_level}")
    i += 1

# 方法2: for循环遍历
print("\n=== 高海拔区域分析 ===")
high_elevation_count = 0
for elevation in terrain_elevation:
    if elevation > 5200:  # 海拔高于5200米的区域
        high_elevation_count += 1
        print(f"高海拔区域: {elevation}米")

print(f"总共{high_elevation_count}个高海拔区域")
```

通过对**一个序列进行操作并筛选出符合条件的元素**，从而生成一个新的列表。

03 列表推导式

3. 列表推导式

- 例：找出100以内所有能被2整除的数。

```
L = [ x for x in range(101) if x%2==0]
```


```
L = []
for x in range(0,101):
    if x%2==0:
        L.append(x)
```

- **列表推导式**使用非常简洁的方式来快速生成满足特定需求的列表，代码具有非常强的可读性

L= [表达式 for 变量 in 序列 if 条件表达式]

③定义最终元素的表达式

①for循环初步定义列表

②if语句过滤初步定义的列表

3. 列表推导式

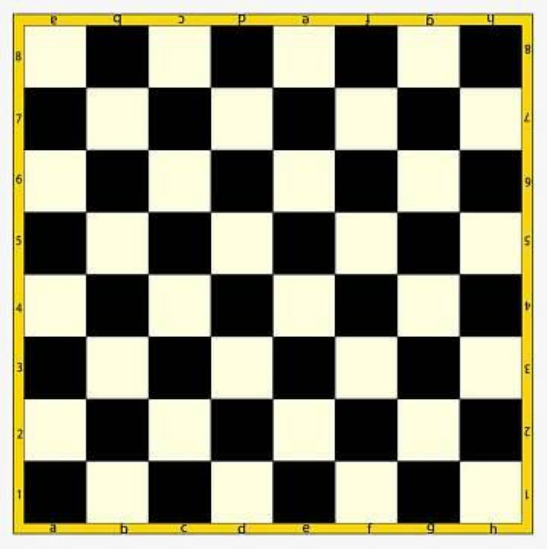
```
1. list1 = [1, 2, 3, 4, 5]
2. list2 = [4, 5, 6, 7, 8]
3. # 创建一个包含两个列表中共同元素的新列表
4. common_elements=[]
5. for x in list1:
6.     if x in list2:
7.         common_elements.append(x)
```

```
1. # 列表推导式写法哪个对?
2. common_elements = [for x in list1: if x in list2]
3. common_elements = [x for x in list1: if x in list2]
4. common_elements = [x for x in list1 if x in list2]
```

3. 列表推导式

- 阿凡提与国王比赛下国际象棋，国王说要是自己输了的话阿凡提想要什么他都可以拿得出来。阿凡提说那我就要点米吧：在第一个格子里放1粒米，第二个格子里放2粒米，第三个格子里放4粒米，以此类推，后面每个格子的米都是前一个的2倍，一直把64个格子都放满需要多少粒米呢？写出列表推导式。

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



3. 列表推导式

- 计算L中所有奇数的平方和

```
L = [101, 25, 38, 29, 108, 121]
```

```
s=0
for i in L:
    if i %2 == 1:
        s+=i**2
print(s)
```

```
sum([i**2 for i in L if i%2 == 1])
```

3. 列表推导式

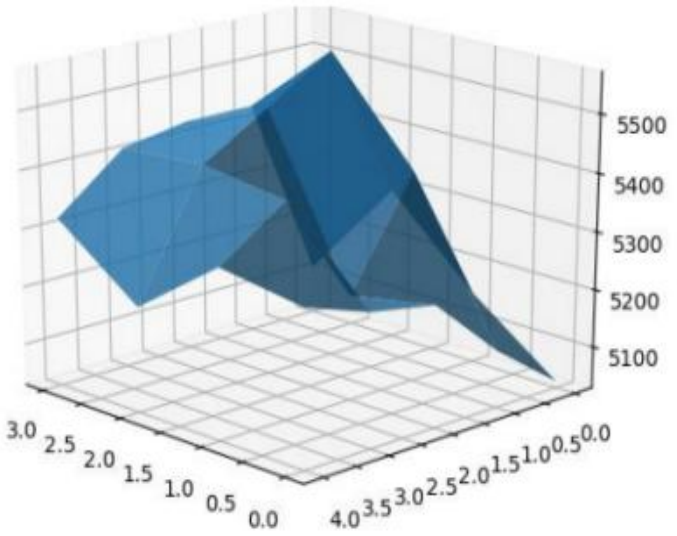
```python
# 生成高威胁区域列表
high_threat_areas = [eq for eq in enemy_equipment if eq > 5]
print(f"高威胁区域装备数: {high_threat_areas}")

# 计算各区域威胁平方和（用于风险评估）
risk_assessment = sum([eq**2 for eq in enemy_equipment])
print(f"风险评估指数: {risk_assessment}")

# 生成侦察简报
recon_report = [f"区域{i+1}: {eq}个目标" for i, eq in enumerate(enemy_equipment)]
print("侦察简报:")
for report in recon_report:
    print(report)
```

```
高威胁区域装备数: [7, 6]
风险评估指数: 123
侦察简报:
区域1: 5个目标
区域2: 3个目标
区域3: 7个目标
区域4: 2个目标
区域5: 6个目标
```

进一步思考

地形导航是无人机设计的关键部分，无人机系统中存储了任务区域的地形信息，根据当前所处位置和目的地位置，无人机可以自动规划出最佳飞行路线。

在某无人机中，把任务区域划分成**二维网格**，并记录每个网格的海拔高度信息。



如图所示，任务区域被分成 4×5 个网格，坐标(x, y)表示第 x 行第 y 列的网格，如坐标为(0, 0)的网格海拔高度为 5039 米，坐标为(0, 1)的网格海拔高度为 5127 米。

请统计统计任务区域 grid 内的山峰数量（不考虑边界点）。

	0	1	2	3	4
0	5039	5127	5238	5259	5299
1	5150	5392	5210	5401	5321
2	5290	5560	5490	5421	5210
3	5110	5429	5430	5411	5319

二维列表

04 二维列表

6 二维列表

- 二维列表是列表的嵌套，即一个列表中每个元素又是一个列表。它可以形象地看作是一个具有行和列的表格，用于存储和处理二维数据结构。

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
```

6 二维列表

```
matrix = [[1, 2, 3],
          [4, 5, 6],
          [7, 8, 9]]
```

行下标 \ 列下标	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

6 二维列表的访问

- 二维列表中的**元素类型可以各不相同**

matrix=[[1,2],[“zhangsan”,”lisi”],[9.4,8.2]]

- 二维列表访问时**通过两个下标进行引用**

列表名[行下标][列下标]

print(matrix[1][1])

输出：lisi

【思考】二维列表中各个一维列表的长度必须相同吗？

```
x = [[1], [4, 5, 6], [7, 8, 9,10]]  ?
```

6 创建二维列表

- **创建二维列表**的三种方法

1.直接初始化

```
matrix=[[0,1,2],
        [3,4,5]
        [6,7,8]]
print(matrix)
```

2.先创建空列表再逐个添加元素

```
matrix=[]
count=0
for i in range(3):
    row = []
    for j in range(3):
        row.append(count)
        count+=1
    matrix.append(row)
print(matrix)
```

3.使用**列表推导式**

```
matrix=[[i*3+j for j in range(3)] for i in range(3)]
print(matrix)
```

```
[[0, 1, 2], [3, 4, 5], [6, 7, 8]]
```

6 二维列表的赋值

- 得到一个m行n列的二维列表之后，如何对列表中的元素进行赋值？

```
[1, 2, 3, 4]
[2, 4, 6, 8]
[3, 6, 9, 12]
```

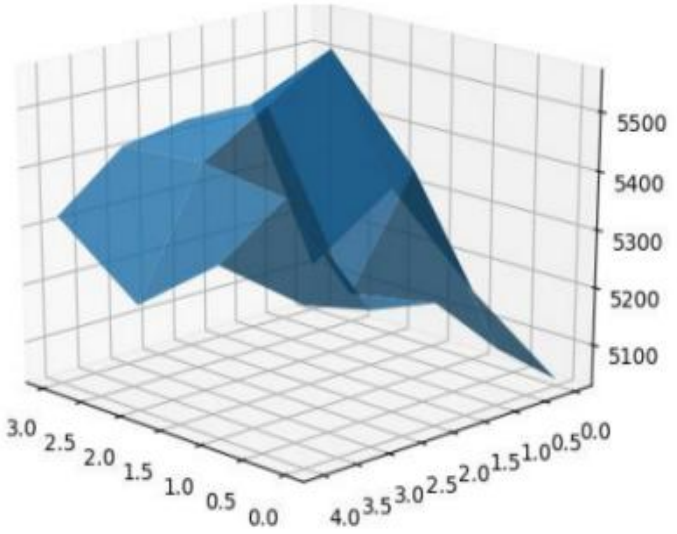
```
m= int(input())
n = int(input())
a=[[0]*n]*m            #相当于复制m行
for i in range(m):
    for j in range(n):
        a[i][j]=(i+1)*(j+1)
print(a)
```

```
a=[[0]*n for i in range(m)]
```

问题

地形导航是无人机设计的关键部分，无人机系统中存储了任务区域的地形信息，根据当前所处位置和目的地位置，无人机可以自动规划出最佳飞行路线。

在某无人机中，把任务区域划分成二维网格，并记录每个网格的海拔高度信息。



如图所示，任务区域被分成 4×5 个网格，坐标(x, y)表示第 x 行第 y 列的网格，如坐标为(0, 0)的网格海拔高度为 5039 米，坐标为(0, 1)的网格海拔高度为 5127 米。

请统计统计任务区域 grid 内的山峰数量（不考虑边界点）。

	0	1	2	3	4
0	5039	5127	5238	5259	5299
1	5150	5392	5210	5401	5321
2	5290	5560	5490	5421	5210
3	5110	5429	5430	5411	5319

6 二维列表的运用

```
mt=[[5039,5127,5238,5259,5299],
    [5150,5329,5210,5401,5321],
    [5290,5560,5490,5421,5210],
    [5110,5429,5430,5411,5319]]
num_row=len(mt)
num_col=len(mt[0])
count=0
for i in range(1,num_row-1):
    for j in range(1,num_col-1):
        co=mt[i][j]
        up=mt[i-1][j]
        dw=mt[i+1][j]
        lf=mt[i][j-1]
        rt=mt[i][j+1]
        if lf<co and rt<co and up<co and dw<co:
            count+=1
print(count)
```

总结

- 列表不仅是数据的容器，更是思维的训练场。
  - 在append()中学会积累
  - 在pop()中懂得取舍
  - 在切片中掌握分寸
  - 在遍历中培养耐心
  - 在推导中追求简洁
  - 在多维中拓展视野
- 通过有序组织、灵活操作、高效处理，让数据为我们所用，让代码为创新服务！

  让Python列表的智慧成为解决问题的利器，在数字时代中游刃有余!

下课并不代表思考的终止
期待我们下次的思想碰撞