**一、基本款：设立变量存储最新的最小值**

查找列表[4,3,5,57,67,8]中最小元素

    核心思想：

        假设第一个就是最小值

        使用假设的和第二个进行比较,如果发现更小的,则替换假设的

        使用假设的和第三个进行比较,如果发现更小的,则替换假设的

        使用假设的和第...个进行比较,如果发现更小的,则替换假设的

        最后，假设的就是最小的.

"""

list01 = [4, 3, 5, 57, 67, 8]

**min\_value = list01[0]**

for i in range(1, len(list01)):

    if min\_value > list01[i]:

        min\_value = list01[i]

print(min\_value)

**二、清晰数字所代表的概念并加以利用**

"""打印出

    ######

    \*\*\*\*\*\*

    ######

    \*\*\*\*\*\*

    ######

    \*\*\*\*\*\*

    ######

"""

for r in range(7):

    for c in range(6):

**if r % 2 == 1:**

            print("\*", end=" ")

        else:

            print("#", end=" ")

    print()

**三、必须清楚给切片赋值**

**list1=【1,2】**

**list1=【1，2，3】 改list1列表的列表地址**

**list1[:2]=[1，2，3] 改list【0】，list【1】两个元素的地址**

**概括：列表赋值覆盖列表地址；列表切片覆盖列表下层元素地址，可以跨越——赋值只改主体地址的铁规**

"""

"""

# 二维列表

list01 = [

    [1, 2, 3, 4],

    [5, 6, 7, 8],

    [9, 10, 11, 12],

]

for line in list01:

    for element in line:

        print(element, end=" ")

print()

**经典的数列横向单方向趋势位移：满足条件的一定会移到数列的某一端）**

**特点：元素会根据循环内的判断条件反复多次进行比较和移动**

**注意点：索引的序号分别是什么**

list01=[7,3,4,65,90,1]

**for i in range(len(list01)-1):**

**for l in range(i+1,len(list01)):**

        print(list01[i],list01[l])

**if list01[i]<list01[l]:**

**list01[i],list01[l]=list01[l],list01[i]**

**print(list01)**

 """

        0元素移动到末尾

 """

    for i in range(len(list\_merge) - 1):

        for r in range(i + 1, len(list\_merge)):

            if list\_merge[i] == 0:

                # 通过全局变量，修改指向的列表

                list\_merge[i], list\_merge[r] = list\_merge[r], list\_merge[i]

def zero\_to\_end():

    """

        领元素移动到末尾

    """

    for i in range(len(list\_merge) - 1):

        for r in range(i + 1, len(list\_merge)):

            if list\_merge[i] == 0:

                # 通过全局变量，修改指向的列表

                list\_merge[i], list\_merge[r] = list\_merge[r], list\_merge[i]

list\_merge = [0, 0, 0, 0]

list\_map = [

    [2, 0, 0, 2],

    [0, 2, 0, 2],

    [4, 0, 2, 2],

    [2, 0, 4, 4],

def move\_left():

    global list\_merge

    for line in list\_map:

        # 将list\_map中每个元素(一维列表) 赋值给 全局变量(merge函数操作list\_merge)

        list\_merge = line

        merge()

def move\_right():

    global list\_merge

**for line in list\_map:在嵌套列表（大列表里有小列表）里用for item in做循环，循环内对小列表做切片赋值(就是例如item[1:2]=。。。)会直接改变大列表内小列表下层的元素，这和for item in的常规结果不同**

**list\_merge[::-1] =line** # 反向切片，创建新列表

        merge()

**line[::-1]= list\_merge #直接修改列表内的下层元素，不是修改列表地址，当然你想改一个切片的新建列表地址也没啥用**

def matrix\_transposition():

    # for r in range(1, len(list\_map)):

    #     for c in range(r, len(list\_map)):

    #         list\_map[r - 1][c], list\_map[c][r - 1] = list\_map[c][r - 1], list\_map[r - 1][c]

    for c in range(1, len(list\_map)):

        for r in range(c, len(list\_map)):

            list\_map[r][c - 1], list\_map[c - 1][r] = list\_map[c - 1][r], list\_map[r][c - 1]

def move\_up():

    matrix\_transposition()

    move\_left()

    matrix\_transposition()

def move\_down():

    matrix\_transposition()

    move\_right()

    matrix\_transposition()

# 执行

move\_right()

print(list\_map)

**四、简洁的列表推导式**

**用函数删除数列中大于10的数**

.def del\_greater\_10(list\_target):

return **[i for i in list\_target if i <= 10]**

print(del\_greater\_10(list01))

**五、用break跳出一层循环，来保证不报错**

**经典的for两层循环**

**如果某种操作后，该循环下的后续操作要报错，就用break，所以利用好跳出操作，不用害怕操作后报错**

# 定义函数, 删除列表中相同元素(只保留一个)

# list01 = [6, 54, 65, 677, 6, 65, 6, 65]

# def delete\_duplicates(list\_target):

#     for r in range(len(list\_target) - 1, -1, -1):

#         for c in range(r):

#             if list\_target[r] == list\_target[c]:

#                 del list\_target[r]

#               **break**

**上题中当r=1时c=0，当r=0时，c in range（0）c是取不到任何值的，但会报错，程序可以执行，这个特征为建立这种嵌套循环建立方便**

**六、和上面题目一样，利用集合的不重复特性**

def delete\_duplicates(list\_target):

    return set(list\_target)

# 测试

list01 = [6, 54, 65, 677, 6, 65, 6, 65]

list01 = delete\_duplicates(list01)

print(list01)

**和上面的题一样**

for i in range(len(list01)-1,-1,-1):

    number=list01.pop(i)

    if number not in list01:

        list01.append(number)

print(list01)

**和上面的题一样，不过是错误的pop使用示范**

**八、经典的末尾删除法**

list1=[1,2,3,4]

for i in range(len(list1)-1，-1，-1):

    del list1[i]

print(list1)

**思路：列表随着i增加长度会逐步减少1，力争让被删除位i，不在减少（影响）范围之内，否则就会因为超出范围而报错**

**九、经典的报错提前处理**

