Матрицы продолжение

In [10]:

```
n = 4
m = 3
a = [[0] * m] * n
a[0][0] = 7
print(a[1][0])
print(a)
```

```
7
[[7, 0, 0], [7, 0, 0], [7, 0, 0], [7, 0, 0]]
```

[0] * m возвращает ссылку на список из n нулей, но последующее повторение элемента создает список из n элементов, которые являются ссылкой на один и тот же список

In [14]:

```
a = []
for i in range(n):
    a.append([])
    for j in range(m):
        a[i].append(0)
print(a)
a[0][0] = 7
print(a[1][0])
print(a)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
0
[[7, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

In [15]:

```
# сумма элементов каждой строки,
# количество положительных элементов во всей матрице
n = int(input("input rows: "))
m = int(input("input columns: "))
print("input matrix, one element in row")
a = []
for i in range(n):
    a.append([])
    for j in range(m):
        a[i].append(int(input()))

print("in one row")
print(a, end="\n\n\n")

for q in a:
    print(q)
count = 0
for i in range(n):
    sum = 0
    for j in range(m):
        sum += a[i][j]
        if a[i][j] > 0:
            count += 1
    print(sum)
print(count)
```

```
input rows: 2
input columns: 2
input matrix, one element in row
1
1
-1
-1
in one row
[[1, 1], [-1, -1]]


[1, 1]
```

```
[-1, -1]
2
-2
2
```

Чаще суммы элементов надо сохранять в массиве, чтобы потом можно было обрабатывать этот список

Результаты надо выводить с поясняющим текстом

In [30]:

```
x = [[-1, 2, 7, 9],
     [-1, 2, 7, 2],
     [-1, 2, 7, 6],
     [-1, 2, 7, 0]]

# поменять местами два столбца

n = 4
print("ИСходная матрица")
for q in x:
    print(q)
print()
# с помощью средств python
for i in range(n):
    x[i][2], x[i][1] = x[i][1], x[i][2]
for q in x:
    print(q)
print()
# поменять местами две строки
# с помощью средств python
print("Исходная матрица")
for q in x:
    print(q)
print()
x[0], x[3] = x[3], x[0]
for q in x:
    print(q)
print("\n\n")

# копия строки
row = x[1][:]
print("Row =", row)
print()

# копия стобца
column = [row[0] for row in x]
print("Column", column, end="\n\n")
# главная диагональ
diag = [x[i][i] for i in range(n)]
print("Main diag", diag)
```

```
ИСходная матрица
[-1, 2, 7, 9]
[-1, 2, 7, 2]
[-1, 2, 7, 6]
[-1, 2, 7, 0]

[-1, 7, 2, 9]
[-1, 7, 2, 2]
[-1, 7, 2, 6]
[-1, 7, 2, 0]

Исходная матрица
[-1, 7, 2, 9]
[-1, 7, 2, 2]
[-1, 7, 2, 6]
[-1, 7, 2, 0]

[-1, 7, 2, 0]
[-1, 7, 2, 2]
[-1, 7, 2, 6]
[-1, 7, 2, 9]



Row = [-1, 7, 2, 2]

Column [-1, -1, -1, -1]
```

Main diag [-1, 7, 2, 9]

In [39]:

```
x = [[-1, 2, 7, 9],
     [-1, 2, 7, 2],
     [-1, 2, 7, 6],
     [-1, 2, 7, 0]]
for q in x:
    print(q)

print("элементы на главной диагонали и под ней")
for i in range(len(x)):
    for j in range(i + 1):
        print(x[i][j], end=" ")
    print()

print("элементы на главной диагонали и над ней")
for i in range(len(x)):
    print(" " * i * 6, end="")
    for j in range(i, len(x)):
        print("{: 6d}".format(x[i][j]), end = "")
    print()
```

```
[-1, 2, 7, 9]
[-1, 2, 7, 2]
[-1, 2, 7, 6]
[-1, 2, 7, 0]
элементы на главной диагонали и под ней
-1
-1 2
-1 2 7
-1 2 7 0
элементы на главной диагонали и над ней
    -1     2     7     9
           2     7     2
                 7     6
                       0
```

In [41]:

```
'''
сформировать матрицу
[1 0 0 0]
[2 1 0 0]
[2 2 1 0]
[2 2 2 1]
'''
n = 4
a = [[0] * n for i in range(n)]

for i in range(n):
    for j in range(n):
        if i < j:
            a[i][j] = 0
        elif i > j:
            a[i][j] = 2
        else:
            a[i][j] = 1

for q in a:
    print(q)
```

```
[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]
```

In [43]:

```
n = 4
a = [[0] * n for i in range(n)]

for i in range(n):
    a[i][i] = 1
    for j in range(i):
```

```
        a[i][j] = 2
    for j in range(i+1, n):
        a[i][j] = 0

for q in a:
    print(q)
```

```
[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]
```

In [45]:

```
n = 4
a = [[0] * n for i in range(n)]
for i in range(n):
    a[i][i] = 1
    for j in range(i):
        a[i][j] = 2
        a[j][i] = 0
for q in a:
    print(q)
```

```
[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]
```

In [46]:

```
# С ипользованием python

n = 4
a = [[0] * n for i in range(n)]

for i in range(n):
    a[i] = [2] * i + [1] + [0] * (n - i - 1)

for q in a:
    print(q)
```

```
[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]
```

In [47]:

```
n = 4
a = [[2] * i + [1] + [0] * (n - i - 1) for i in range(n)]

for q in a:
    print(q)
```

```
[1, 0, 0, 0]
[2, 1, 0, 0]
[2, 2, 1, 0]
[2, 2, 2, 1]
```

Срезы (slice)

Срезы полезны лд ясоздания копий последовательностей или их частей, что часто позволяет создавать эффективные программы.

Представим срез в общем виде для списка [i:j], где между двумя индексами стоит символ двоеточие, а индексы пока имеют положительное значение.

Результатом действия списка [i:j] будет список из элементов с индексами большими или равными i и меньшими j. Когда i или j бльше длины спика, то соответствующие индекс неявно заменяется на длину спика, то есть большие индексы укорачиваются.

In [48]:

```
x = [1, 2, 3, 4, 5]
r = x[:]  # копия списка x
```

```
print(r)
r1 = x[0:3] # формируется спико из трех элементов x[0], x[1], x[2]
print(r)
r2 = x[1:]  # все элементы кроме первого
print(r2)
r3 = x[:3]  # первые три элемен массива
print(r3)
r4 = x[1:10]  # результат равен x[2, 3, 4, 5]
print(r4)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
[2, 3, 4, 5]
[1, 2, 3]
[2, 3, 4, 5]
```

При равенстве индексов или когда первый больше второго, срез будет пустым

Один или оба индекса могут быть отрицательными. при этом надо как бы заменить индекс на длину списка + индекс

In [53]:

```
x = [1, 2, 3, 4, 5]
r5 = x[:-1]
print(r5)
r6 = x[-3: ]
print(r6)
```

```
[1, 2, 3, 4]
[3, 4, 5]
```

Срезы могут быть и с тремя параметрами при этом третий индекс определяет приращение для индекса, заключаемого в срез

In [57]:

```
x = [1, 2, 3, 4, 5]
r7 = x[::2]
r8 = x[::-1]
print(r7, r8, sep="\n")
r9 = x[1:4:2]
print(r9)
```

```
[1, 3, 5]
[5, 4, 3, 2, 1]
[2, 4]
```

In [60]:

```
x = list(range(10))
del x[3:-2:2]
print(x)
```

```
[0, 1, 2, 4, 6, 8, 9]
```