## 🧩 Что такое «Сапёр»

«Сапёр» — это логическая игра, где поле состоит из клеток.  
Некоторые клетки — **заминированы** (\*), другие — **пустые** (.).  
Цель — открыть все пустые клетки, не наткнувшись на мину.

Когда ты открываешь пустую клетку, игра показывает **число** — сколько мин находится **в соседних 8 клетках** (по вертикали, горизонтали и диагонали).

Пример:

. \* .

. . .

\* . .

Ответ:

1 \* 1

2 3 1

\* 2 0

## 💻 Что делает код

Теперь перейдём к твоему коду.  
Он имитирует «логику сапёра»:  
принимает матрицу с минами (\*) и считает, сколько мин рядом с каждой клеткой.

### 🔹 1. Ввод размеров поля

n, m = map(int, input("Введите размеры матрицы (n m): ").split())

— пользователь вводит размеры, например 3 4 → значит поле 3 строки и 4 столбца.  
n — количество строк, m — количество столбцов.

### 🔹 2. Ввод самой матрицы

grid = [input().split() for \_ in range(n)]

— создаётся список из n строк, каждая строка — список символов (\* или .).  
Например, если ввести:

. \* .

. . .

\* . .

→ получится:

grid = [['.', '\*', '.'],

['.', '.', '.'],

['\*', '.', '.']]

### 🔹 3. Создание пустой матрицы для результата

result = [[0]\*m for \_ in range(n)]

— создаёт таблицу того же размера, но заполненную нулями, куда будут записываться числа мин.

### 🔹 4. Смещения соседей

dirs = [(-1,-1), (-1,0), (-1,1),

(0,-1), (0,1),

(1,-1), (1,0), (1,1)]

— это координаты **8 направлений** вокруг клетки (i, j):  
вверх-влево, вверх, вверх-вправо, влево, вправо, вниз-влево, вниз, вниз-вправо.

### 🔹 5. Подсчёт мин

for i in range(n):

for j in range(m):

if grid[i][j] == "\*":

result[i][j] = "\*"

else:

count = 0

for dx, dy in dirs:

x, y = i+dx, j+dy

if 0 <= x < n and 0 <= y < m and grid[x][y] == "\*":

count += 1

result[i][j] = count

Построчно:

* Проходим по каждой клетке.
* Если там мина → ставим "\*" в результат.
* Если нет → считаем, сколько рядом мин:
  + Проверяем 8 направлений.
  + Считаем только те, которые **в пределах поля** (0 <= x < n, 0 <= y < m).

### 🔹 6. Вывод результата

for row in result:

print(\*row)

— красиво выводит результат в виде таблицы.  
\*row распаковывает список, выводя элементы через пробел.

## 🔹 Пример работы

**Ввод:**

3 3

. \* .

. . .

\* . .

**Вывод:**

1 \* 1

2 3 1

\* 2 0

## **1. Что такое матрица**

В Python матрица — это **двумерная структура данных**, обычно представляется как **список списков**, где каждый вложенный список — это строка матрицы.

**Пример 2×3 матрицы:**

matrix = [

[1, 2, 3],

[4, 5, 6]

]

* matrix[0][1] → 2 (строка 0, столбец 1)
* len(matrix) → 2 (число строк)
* len(matrix[0]) → 3 (число столбцов)

## **2. Основные операции с матрицами**

### ****2.1 Создание матрицы****

# Пустая 3x3 матрица из нулей

matrix = [[0 for \_ in range(3)] for \_ in range(3)]

### ****2.2 Доступ и изменение элементов****

matrix[1][2] = 7 # Меняем элемент второй строки, третий столбец

print(matrix[1][2]) # 7

### ****2.3 Сложение матриц****

A = [[1, 2], [3, 4]]

B = [[5, 6], [7, 8]]

C = [[A[i][j] + B[i][j] for j in range(len(A[0]))] for i in range(len(A))]

print(C) # [[6, 8], [10, 12]]

### ****2.4 Умножение на число****

k = 3

C = [[k \* A[i][j] for j in range(len(A[0]))] for i in range(len(A))]

print(C) # [[3, 6], [9, 12]]

### ****2.5 Умножение матриц****

A = [[1, 2, 3], [4, 5, 6]]

B = [[7, 8], [9, 10], [11, 12]]

C = [[sum(A[i][k]\*B[k][j] for k in range(len(B))) for j in range(len(B[0]))] for i in range(len(A))]

print(C) # [[58, 64], [139, 154]]

### ****2.6 Транспонирование****

Меняем строки на столбцы:

A = [[1, 2, 3], [4, 5, 6]]

T = [[A[j][i] for j in range(len(A))] for i in range(len(A[0]))]

print(T) # [[1, 4], [2, 5], [3, 6]]

### ****Задание 3: Простая игра “Крестики-нолики” (текстовая)****

**Цель:** Практика списков, циклов, функций и условий.

**ТЗ:**

1. Игровое поле — 3x3, представлено списком списков.
2. Игроки по очереди ставят X и O.
3. После каждого хода программа выводит текущее поле.
4. Игра заканчивается, когда кто-то выиграл или поле заполнено.

