**1. Уникальные элементы (множества и счётчики)**

* **Множества (set)** в Python содержат только уникальные элементы.
* nums = [1, 2, 2, 3]
* print(set(nums)) # {1, 2, 3}
* Метод .count(x) у списка возвращает, сколько раз встречается элемент.
* Для более быстрого подсчёта можно использовать collections.Counter:
* from collections import Counter
* c = Counter(nums)
* print(c) # Counter({2: 2, 1: 1, 3: 1})

👉 Подсказка: ищем элементы, у которых количество в списке равно 1.

**🔹 2. Анаграммы (работа со строками)**

* **Строка** в Python — это последовательность символов.
* Можно сортировать символы:
* word = "silent"
* print(sorted(word)) # ['e', 'i', 'l', 'n', 's', 't']
* Если отсортированные символы у двух слов совпадают — это анаграмма.
* Вариант: сравнивать через Counter, чтобы учитывать количество букв.

👉 Подсказка: анаграмма = одинаковый «набор букв».

**🔹 3. Магический квадрат (матрицы)**

* **Матрица** в Python хранится как список списков:
* matrix = [
* [2, 7, 6],
* [9, 5, 1],
* [4, 3, 8]
* ]
* Можно суммировать строки и столбцы:
* sum(matrix[0]) # сумма первой строки
* sum(matrix[row][0] for row in range(3)) # сумма первого столбца
* Диагонали берём с индексами [i][i] и [i][n-1-i].
* Если все суммы одинаковы → магический квадрат.

👉 Подсказка: надо проверить все строки, все столбцы и диагонали.

**🔹 4. Сжатие строки (RLE — Run Length Encoding)**

* Суть RLE: заменить **повторяющиеся символы** на «символ + количество повторов».
* "aaabb" → "a3b2"
* Идея: пройти строку слева направо, считать количество одинаковых подряд символов, записывать результат и обнулять счётчик при смене символа.
* Трюк: не забыть обработать **последнюю группу символов**.

👉 Подсказка: нужен цикл по символам + счётчик подряд идущих одинаковых букв.

**🔹 5. Сапёр (матрица + соседние клетки)**

* Игровое поле = матрица (list списков).
* Чтобы посчитать количество мин вокруг клетки, надо проверить **8 соседних клеток**.
* Можно описать направления сдвига:
* directions = [(-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1)]
* Для каждой клетки проверяем соседей по этим сдвигам.
* Важно: не выходить за границы матрицы → проверка индексов.

👉 Подсказка: используем «двойной цикл» по строкам и столбцам, а внутри — цикл по 8 направлениям.