**Хранение матриц**

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются **матрицами** или **двумерными массивами**. В языке программирования Python таблицу можно представить в виде списка, каждый элемент которого тоже является списком, например, списком чисел.

**Пример:**

Если требуется создать числовую таблицу из двух строк и трёх столбцов, то это можно сделать следующим образом: a = [[2, 3, 4], [5, 6, 7]].

Здесь первый элемент списка a[0] является списком из чисел [2, 3, 4]. То есть a[0][0] равно 2, a[0][1] равно 3, a[0][2] равно 4. Таким образом, чтобы обратиться к элементу, расположенному в i-й строке и j-м столбце, то надо написать a[i][j]. Также можно использовать отрицательные индексы, например, элемент a[-1][-1] нашей таблицы — это элемент из последней строки и последнего столбца, и он равен числу 7.

**Обработка и вывод списка**

Для обработки и вывода списка, как правило, используются два вложенных цикла. Первый цикл перебирает все строки, второй цикл — элементы внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

**for** i **in** range(len(a)):

**for** j **in** range(len(a[i])):

**print**(a[i][j], end=' ')

**print**()

Обратим внимание на то, что пробелы будут выведены не только между числами в строке, но и после последнего элемента. При сдаче задач эти пробелы не мешают, поэтому можно об этом не беспокоиться.

Можно перебирать элементы списка не по индексу, а по значению. Тогда код вывода двумерного массива будет иметь вот такой вид:

**for** row **in** a:

**for** elem **in** row:

**print**(elem, end=' ')

**print**()

Кроме того, для вывода одной строки можно воспользоваться методом join:

**for** row **in** a:

**print**(' '.join(list(map(str, row))))

Наконец, можно использовать и упрощенный способ вывода списков:

**for** i **in** range(len(a)):

**print**(\*a[i])

**Пример:**

Используем два вложенных цикла для подсчета суммы всех чисел в таблице и нахождения минимального элемента в таблице:

S = 0

M = a[0][0]

**for** i **in** range(len(a)):

**for** j **in** range(len(a[i])):

S += a[i][j]

M = min(M, a[i][j])

**print**(S, M)

Для решения этой задачи можно использовать циклы не по индексам, а по элементам:

S = 0

M = a[0][0]

**for** row **in** a:

**for** elem **in** row:

S += elem

M = min(M, elem)

**print**(S, M)

Заметим, что если мы хотим параллельно изменять значения элементов в нашей таблице, то придётся использовать первый вариант. Действительно, переменная elem является дополнительной переменной, и если мы ее меняем, то элемент таблицы при этом не меняется. Таким образом, если мы хотим не только найти сумму чисел и минимум исходной таблицы, но и увеличить все элементы таблицы на 1, то для этого надо использовать первый вариант, и код будет выглядеть следующим образом:

S = 0

M = a[0][0]

**for** i **in** range(len(a)):

**for** j **in** range(len(a[i])):

S += a[i][j]

M = min(M, a[i][j])

a[i][j] += 1

**print**(S, M)