

## Вложенные списки

- 1 Вложенные списки. Двумерные вложенные списки (матрицы)
- 2 Создание двумерного списка
- 3 Перебор элементов двумерного списка. Вывод списка на экран
- 4 Матрицы

### Аннотация

Мы уже упоминали о том, что элементами списка могут быть любые объекты — числа, строки, кортежи, множества и даже другие списки. Сегодня мы рассмотрим подробнее списки, элементами которых являются другие (вложенные) списки.

## 1. Вложенные списки. Двумерные вложенные списки (матрицы)

Язык Python не ограничивает нас в уровнях вложенности: элементами списка могут быть списки, их элементами могут быть другие списки, элементами которых в свою очередь могут быть другие списки и т. д. Но для решения практических задач сначала важно научиться работать с двумерными списками.

С помощью таких списков очень удобно представить прямоугольную таблицу (матрицу) — каждый вложенный список при этом будет являться строкой. Именно такая структура данных используется, например, для представления игровых полей при программировании таких игр, как шахматы, крестики-нолики, морской бой, 2048.

## 2. Создание двумерного списка

### Создание двумерного списка

Важно понять, что список списков принципиально ничем не отличается, например, от списка чисел. Чтобы задать список списков в программе, мы также перечисляем элементы через запятую в квадратных скобках:

```
table = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Если элементы списка вводятся через клавиатуру (каждая строка таблицы на отдельной строке, всего  $n$  строк, числа в строке разделяются пробелами), для ввода списка можно использовать следующий код:

```
table = []
for i in range(n):
    row = [int(el) for el in input().split()]
    table.append(row)
```

В этом примере мы используем метод `append`, передавая ему в качестве аргумента другой список. Так у нас получается список списков.

## Списочные выражения

Для создания вложенных списков можно использовать списочные выражения. Например, список из предыдущего примера можно создать так:

```
table = [[int(el) for el in input().split()] for i in range(n)]
```

Попробуем теперь составить список размером  $10 \times 10$  элементов, заполненный нулями (такая задача нередко возникает при написании различных программ). Может показаться, что сработает конструкция `a = [[0] * 10] * 10`, но это не так. Попробуйте понять почему.

*Подсказка: создайте такой список, измените в нем один элемент и посмотрите, что получилось.*

Самый короткий способ выполнить такую задачу — при помощи списочного выражения:

```
[[0] * 10 for _ in range(10)]
```

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

## Важно!

Обратите внимание: в этом примере используется переменная `_`. Это вполне законное имя переменной, как и, например, `i`. Однако по соглашению оно используется для переменной-счетчика только в том случае, когда принимаемые этой переменной значения не важны, а важно лишь количество итераций.

Подобное имя переменной можно было бы использовать и в первом примере списочного выражения:

```
table = [[int(el) for el in input().split()] for _ in range(n)]
```

### 3. Перебор элементов двумерного списка. Вывод списка на экран

Для доступа к элементу списка мы должны указать индекс этого элемента в квадратных скобках. В случае двумерных вложенных списков мы должны указать два индекса (каждый в отдельных квадратных скобках), в случае трехмерного списка — три индекса и т. д. В двумерном случае сначала указывается номер строки, затем — номер столбца (сначала выбирается вложенный список, а затем — элемент из него).

```
table = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(table[0][0], table[0][1], table[1][0])
```

Для того чтобы перебрать все элементы матрицы (чтобы, например, вывести их на экран), обычно используются вложенные циклы. Например, список из предыдущего примера можно вывести на экран таким образом:

```
for i in range(3):
    for j in range(3):
        print(table[i][j], end='\\t')
    print()
```

В этом примере мы перебирали индексы элементов. А что будет, если перебирать сами элементы? Например, если мы хотим подсчитать сумму всех элементов матрицы, можно написать такой цикл:

```
s = 0
for row in table:
    s += sum(row)
print(s)
```

### 4. Матрицы

В некоторых задачах этого урока вам встретится важный математический объект, который называется «матрица».

#### Матрица

Матрица — прямоугольная табличка, заполненная какими-то значениями, обычно числами.

В математике вам встретится множество различных применений матриц, поскольку с их помощью многие задачи гораздо проще сформулировать и решить. Мы же сконцентрируемся на том, как хранить матрицу в памяти компьютера.

В первую очередь от матрицы нам нужно уметь получать элемент в  $i$ -й строке и  $j$ -м столбце. Чтобы этого добиться, обычно поступают так: заводят список строк матрицы, а каждая строка матрицы сама по себе тоже является списком элементов. То есть мы получили список списков чисел. Теперь, чтобы получить элемент, нам достаточно из списка строк матрицы выбрать  $i$ -ю и из этой строки взять  $j$ -й элемент.

Давайте заведем простую матрицу  $M$  размера  $2 \times 3$  (2 строки и 3 столбца) и получим элемент на позиции (1, 3). Обратите внимание: в математике нумерация строк и столбцов идет с единицы, а не с нуля. И, по договоренности среди математиков, сначала всегда указывается строка, а лишь затем — столбец. Элемент на  $i$ -ой строке,  $j$ -м столбце матрицы  $M$  в математике обозначается  $M_{i,j}$ .

Итак:

```
matrix = [[1, 2, 3],  
          [2, 4, 6]]  
print(matrix[0][2]) # => 3
```

`matrix` — вся матрица, `matrix[0]` — список значений в первой строке, `matrix[0][2]` — элемент в третьем столбце в этой строке.

Чтобы перебрать элементы матрицы, приходится использовать двойные циклы. Например, выведем на экран все элементы матрицы, перебирая их по столбцам:

```
for col in range(3):  
    for row in range(2):  
        print(matrix[row][col])
```

## Справка

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках сервиса, принадлежат АНО ДПО «Образовательные технологии Яндекса». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «Образовательные технологии Яндекса».

[Пользовательское соглашение](#).

© 2018 – 2024 ООО «Яндекс»