Списки списков

Список тоже может быть элементом списка

Создадим список из трёх списков из четырёх чисел

```
s = [[0, 10, 20, 30], [40, 50, 60, 70], [80, 90, 100, 110]]


print(s[0])     # [0, 10, 20, 30]
print(s[2])     # [80, 90, 100, 110]

print(s[1][2])  # 60
```

Эту конструкцию можно переписать так:

```
s = [[0, 10, 20, 30],
     [40, 50, 60, 70],
     [80, 90, 100, 110]]
```

Тогда оно становится похоже на табличку 3x4.

Синонимы: двумерный список (в Python), двумерный массив, матрица

Рабоче-крестьянский способ создать матрицу

```
# Рабоче-крестьянский способ

n, m = 3, 4     # число строк и столбцов

matrix = []
for i in range(n):
    row = []
    for j in range(m):
        row.append(0)
    matrix.append(row)
print(matrix)   # [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]


# Более умный способ

n, m = 3, 4

matrix = []
for i in range(n):
    row = [0] * m
    matrix.append(row)
print(matrix)   # [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Рабоче-крестьянский пример

```
# Квадратная матрица, с диагональными элементами, равными 1

n = 4
matrix = []
for i in range(n):
    row = []
    for j in range(n):
        if i == j:
            row.append(1)
        else:
            row.append(0)
    matrix.append(row)

for row in matrix:
    print(row) #    Построчно печатается матрица

# [1, 0, 0, 0]
# [0, 1, 0, 0]
# [0, 0, 1, 0]
# [0, 0, 0, 1]
```

Более умный пример

```
# Квадратная матрица, с диагональными элементами, равными 1

n = 4

matrix = []
for i in range(n):
    row = [0] * i + [1] + [0] * (n - i - 1)
    matrix.append(row)

for row in matrix:
    print(row)

# [1, 0, 0, 0]
# [0, 1, 0, 0]
# [0, 0, 1, 0]
# [0, 0, 0, 1]
```

Ещё пример

```
n, m = 4, 8
matrix = []
for i in range(n):
    matrix.append([0] * m)
    for j in range(m):
        matrix[i][j] = (i + 1) ** (j + 1)


for row in matrix:
    print(row)


# [1, 1, 1, 1, 1, 1, 1, 1]
# [2, 4, 8, 16, 32, 64, 128, 256]
# [3, 9, 27, 81, 243, 729, 2187, 6561]
# [4, 16, 64, 256, 1024, 4096, 16384, 65536]
```

Рабоче-крестьянский способ напечатать матрицу

```
for row in matrix:
    for elem in row:
        print(elem, end=' ')
    print()


# 1 1 1 1 1 1 1 1
# 2 4 8 16 32 64 128 256
# 3 9 27 81 243 729 2187 6561
# 4 16 64 256 1024 4096 16384 65536


for row in matrix:
    for elem in row:
        print(f'{elem :5d}', end=' ')
    print()


#     1     1     1     1     1     1     1     1
#     2     4     8    16    32    64   128   256
#     3     9    27    81   243   729  2187  6561
#     4    16    64   256  1024  4096 16384 65536
```

Более умный способ напечатать матрицу

```
def print_matrix(mat, w=0):
    for row in mat:
        for elem in row:
            print(f'{elem :{w}}', end=' ')
        print()


print_matrix(matrix, w=5)


#     1     1     1     1     1     1     1     1
#     2     4     8    16    32    64   128   256
#     3     9    27    81   243   729  2187  6561
#     4    16    64   256  1024  4096 16384 65536
```

Ещё более умный способ напечатать матрицу

```
def max_matrix_elem_len(mat):
    max_len = 0
    for row in mat:
        for elem in row:
            max_len = max(max_len, len(str(elem)))
    return max_len


def print_matrix(mat):
    w = max_matrix_elem_len(mat)
    for row in mat:
        for elem in row:
            print(f'{elem :{w}}', end=' ')
        print()


print_matrix(matrix)

#    1     1     1     1     1     1     1     1
#    2     4     8    16    32    64   128   256
#    3     9    27    81   243   729  2187  6561
#    4    16    64   256  1024  4096 16384 65536
```

Более умный способ создать матрицу

```
def zeros(n, m):
    result = []
    for i in range(n):
        result.append([0] * m)
    return result


matrix = zeros(3, 7)
print_matrix(matrix)


# 0 0 0 0 0 0 0
# 0 0 0 0 0 0 0
# 0 0 0 0 0 0 0
```

Пример: поэлементная сумма матриц

```
def mat_sum(a, b):
    n, m = len(a), len(a[0])
    result = zeros(n, m)

    for i in range(n):
        for j in range(m):
            result[i][j] = a[i][j] + b[i][j]
    return result

a = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
b = [[20, 30, 40],
     [90, 60, 90],
     [50, 60, 70]]

c = mat_sum(a, b)
print_matrix(c)

# 21 32 43
# 94 65 96
# 57 68 79
```

Пример: горизонтальное склеивание матриц

```
a = [[1, 2, 3, 8000],
     [4, 5, 6, 0],
     [7, 8, 9, 99]]
b = [[20, 30, 40],
     [90, 60, 90],
     [50, 60, 70]]

def hstack(a, b):
    n, ma, mb = len(a), len(a[0]), len(b[0])
    result = zeros(n, ma + mb)

    for i in range(n):
        for j in range(ma):
            result[i][j] = a[i][j]
        for j in range(mb):
            result[i][ma + j] = b[i][j]

    return result

c = hstack(a, b)
print_matrix(c)

#    1    2    3 8000   20   30   40
#    4    5    6    0   90   60   90
#    7    8    9   99   50   60   70
```

Рабоче-крестьянский способ ввести матрицу с клавиатуры

```
n = int(input('Введите кол-во строк матрицы: '))
matrix = []
for i in range(n):
    s = input(f'Строка {i}: ')

    row = []
    for elem in s.split():
        row.append(int(elem))
    matrix.append(row)

# Введите кол-во строк матрицы: 4
# Строка 0: 1 2 3 4
# Строка 1: 2 3 4 5
# Строка 2: 555 666 777 9990
# Строка 3: 10 29 39 49

print_matrix(matrix)

#    1    2    3    4
#    2    3    4    5
#  555  666  777 9990
#   10   29   39   49
```

Более умный способ ввести матрицу с клавиатуры

```
def input_matrix():
    matrix = []
    while row := input(f'Строка {len(matrix)}: '):
        matrix.append(list(map(int, row.split())))
    return matrix

matrix = input_matrix()

# Строка 0: 1 2 3 4
# Строка 1: 2 3 4 5
# Строка 2: 555 666 777 9990
# Строка 3: 10 29 39 49
# Строка 4:

print_matrix(matrix)

#    1    2    3    4
#    2    3    4    5
#  555  666  777 9990
#   10   29   39   49
```

Способ создать матрицу из случайных чисел

```
import random

def random_matrix(n, m):
    result = zeros(n, m)
    for i in range(n):
        for j in range(m):
            result[i][j] = random.randint(0, 9)
    return result

matrix = random_matrix(5, 10)
print_matrix(matrix)

# 0 7 4 6 4 1 3 7 9 4
# 2 3 0 9 7 6 6 4 9 3
# 0 2 7 5 2 6 6 0 0 5
# 3 8 7 6 8 8 5 3 4 6
# 9 6 0 4 8 3 6 2 7 5
```

Грабли

Вот так создавать матрицу неправильно
потому что в ней на самом деле n раз повторяется одна и та же строка

```
n, m = 3, 7
matrix = [[0] * m] * n

print_matrix(matrix)

# 0 0 0 0 0 0 0
# 0 0 0 0 0 0 0
# 0 0 0 0 0 0 0

matrix[0][4] = 1
print_matrix(matrix)

# 0 0 0 0 1 0 0
# 0 0 0 0 1 0 0
# 0 0 0 0 1 0 0
```