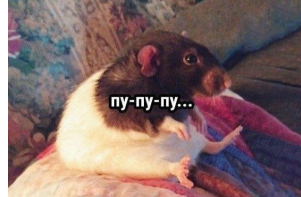


Вложенные списки

Повторение



1. На вход программе подается строка текста, содержащая слова. Напишите программу, которая выводит слова введенной строки в столбик.
2. На вход программе подается строка текста. Напишите программу, использующую списочное выражение, которая выводит все цифровые символы данной строки.
3. На вход программе подается строка текста, содержащая целые числа. Напишите программу, использующую списочное выражение, которая выведет квадраты четных чисел, которые не оканчиваются на цифру 4.
4. На вход программе подается строка текста, содержащая натуральные числа. Напишите программу, которая вставляет между каждым числом знак +, а затем вычисляет сумму полученных чисел.

Вложенные списки в python

- Иногда для правильного представления набора данных простого одномерного массива недостаточно.
- В таких случаях используют двумерные массивы, матрицы и многомерные массивы.
- Однако в Python 3 массивов, матриц, по сути, не существует.
- Но это не проблема, так как базовые возможности платформы позволяют легко создавать двумерные списки.

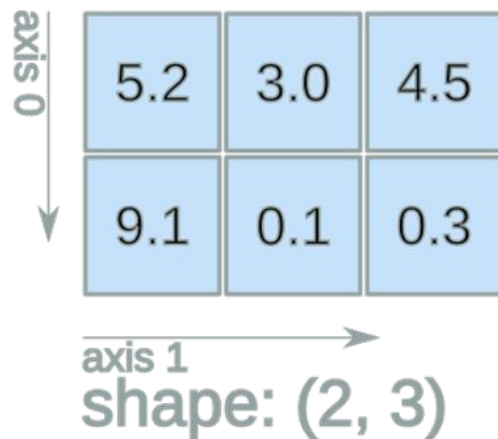
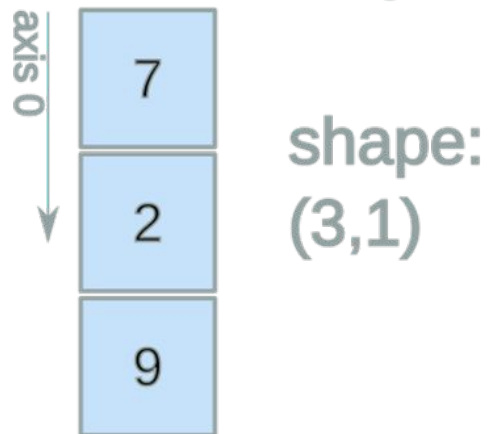


1D array

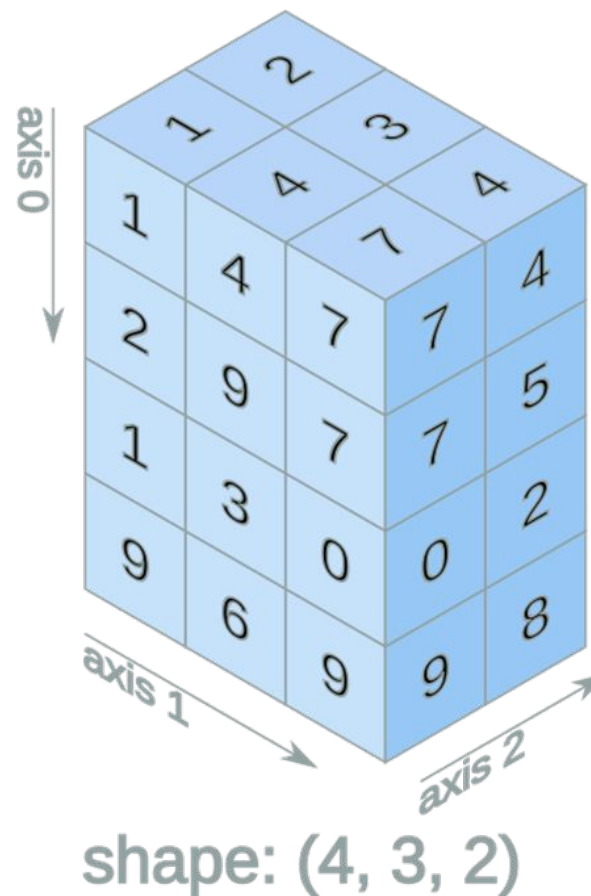


shape: (3,)

2D array



3D array



X: 0 1 2 3 4

Y: 0 1 2 3 4

0	0	0	0	0
1	0	0	0	0
2	0	0	1	0
3	0	1	1	1
4	1	1	1	1

4 columns

2	-5	-11	0
-9	4	6	13
4	7	12	-2

3 rows

Вывод двумерного списка

```
a = [[1, 2, 3], [4, 5, 6]]
```

```
print(a[0])
```

```
print(a[1])
```

```
[1, 2, 3]
```

```
[4, 5, 6]
```

Здесь первая строка списка `a[0]` является списком из чисел `[1, 2, 3]`.

То есть `a[0][0] == 1`,
значение `a[0][1] == 2`,

`a[0][2] == 3`,

`a[1][0] == 4`,

`a[1][1] == 5`,

`a[1][2] == 6`.

```
a = [[2, 4, 6, 8, 10],  
      [3, 6, 9, 12, 15],  
      [4, 8, 12, 16, 20]]
```

```
print(a)
```

```
[[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
```

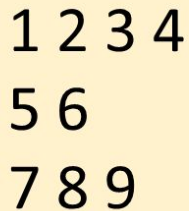
```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]  
for i in range(len(a)):  
    for j in range(len(a[i])):  
        print(a[i][j], end=' ')  
    print()
```

1 2 3 4

5 6

7 8 9


```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]  
for row in a:  
    for elem in row:  
        print(elem, end=' ')  
    print()
```



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

Переменная цикла `for` в Python может перебирать не только диапазон, создаваемый с помощью функции `range()`, но и вообще перебирать любые элементы любой последовательности.

Последовательностями в Python являются списки, строки, а также некоторые другие объекты.

Продemonстрируем, как выводить двумерный массив, используя это удобное свойство цикла `for`:

Создание вложенных списков

- Пусть даны два числа:
количество строк n и количество столбцов m .
- **Необходимо создать список размером $n \times m$,
заполненный нулями.**

```
n = 3
m = 4
a = [[0] * m] * n
print(a[0])
print(a[1])
print(a[2])
```

Это плохой
вариант

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

```
n = 3
```

```
m = 4
```

```
a = [[0] * m] * n
```

```
a[0][0] = 5
```

```
print(a[1][0])
```

```
print(a[0])
```

```
print(a[1])
```

```
print(a[2])
```

Это плохой
вариант

Глобальные
переменные

n 3

m 4

a

list (id=1):

0

list (id=2):

0

1

2

3

5

0

0

0

1

list (id=2):

0

1

2

3

5

0

0

0

2

list (id=2):

0

1

2

3

5

0

0

0

```
n = 3
m = 4
a = [0] * n
for i in range(n):
    a[i] = [0] * m
```

```
print(a[0])
print(a[1])
print(a[2])
```

Первый способ: сначала создадим список из n элементов (для начала просто из n нулей). Затем сделаем каждый элемент списка ссылкой на другой одномерный список из m элементов:

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

```
n = 3
m = 4
a = []
for i in range(n):
    a.append([0] * m)
```

```
print(a[0])
print(a[1])
print(a[2])
```

Другой (но похожий) способ:
создать пустой список, потом n
раз добавить в него новый
элемент, являющийся списком-
строкой:

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

Но еще проще воспользоваться генератором:
создать список из n элементов, каждый из которых
будет списком, состоящих из m нулей.

В этом случае каждый элемент создается независимо
от остальных (заново конструируется список $[0] * m$
для заполнения очередного элемента списка), а не
копируются ссылки на один и тот же список.

```
n = 3
```

```
m = 4
```

```
a = [[0] * m for i in range(n)]
```

```
print(a[0])
```

```
print(a[1])
```

```
print(a[2])
```

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

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

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

```
m = 3
```

```
n = 6
```

```
a = [[0 for x in range(n)] for x in range(m)]
```

```
print(a)
```

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

Создание двумерного списка N на M
из случайных чисел в диапазоне от 20 до 80

```
from random import randint
N=5
M=3
A = [ [0]*M for i in range(N) ]
for i in range(N):
    for j in range(M):
        A[i][j] = randint( 20, 80 )
print(A)
```

```
[[60, 79, 40], [31, 48, 35], [73, 32, 20], [57, 46, 61], [49, 62, 79]]
```


Ввод двумерного списка

```
# в первой строке ввода идёт количество строк массива
n = int(input())
a = []
# значения строки вводим через пробел
for i in range(n):
    a.append([int(j) for j in input().split()])
# смотрим полученный результат
print(a[0])
print(a[1])
```

Ввод

2

2 3 4

5 6 7

Вывод

[2, 3, 4]

[5, 6, 7]

в первой строке ввода идёт количество строк массива

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

смотрим полученный результат

```
print(a[0])  
print(a[1])
```

Ввод

2

5 8 6

8 9 6

Вывод

[5, 8, 6]

[8, 9, 6]

То же самое и при помощи генератора

в первой строке ввода идёт количество
строк массива

```
n = int(input())
```

```
a = [[int(j) for j in input().split()] for  
i in range(n)]
```

смотрим полученный результат

```
print(a[0])
```

```
print(a[1])
```

Ввод

2

8 9 8 7

3 6 5

Ввод

[8, 9, 8, 7]

[3, 6, 5]

1. Подсчитайте сумму всех чисел в двумерном списке 3x3
2. Создайте список 4x5 заполненный единицами

Методы в многомерных списках

Добавление подсписка

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15],  
      [4, 8, 12, 16, 20]]
```

```
print(a)
```

```
a.append([10, 10, 10, 10, 10])
```

```
print(a)
```

```
[[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
```

```
[[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20], [10, 10, 10, 10, 10]]
```

`extension ()`: добавьте элементы списка (или любого итерируемого) в конец текущего списка

Расширение списка

```
a = [[2, 4, 6, 8, 9], [3, 6, 9, 12, 15],  
      [4, 8, 12, 16, 20]]
```

```
print(a)
```

```
a[0].extend([10, 10, 10, 10])
```

```
print(a)
```

```
[[2, 4, 6, 8, 9], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
```

```
[[2, 4, 6, 8, 9, 10, 10, 10, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
```

reverse (): обратный порядок в списке

```
# Сторнирование подписка
```

```
a = [[2, 4, 6, 8, 10], [3, 6, 9, 12, 15],  
      [4, 8, 12, 16, 20]]
```

```
print(a)
```

```
a[2].reverse()
```

```
print(a)
```

```
[[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [4, 8, 12, 16, 20]]
```

```
[[2, 4, 6, 8, 10], [3, 6, 9, 12, 15], [20, 16, 12, 8, 4]]
```

Сумма значений в строках

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

[6, 15, 24]

```
result = list(map(sum, array))  
print(result)
```

Функция map - самая простая из встроенных в Python, map() применяет указанную функцию к каждому элементу в качестве итератора.

Транспонирование матрицы*

```
my_list=[[1,2], [3,4], [5,6]]  
for row in my_list:  
    print(row)  
print("Транспонирование матрицы")  
  
#list(map(list, zip(*my_list)))  
my_list_T=map(list, zip(*my_list))  
list(my_list_T)
```

[1, 2]

[3, 4]

[5, 6]

Транспонирование матрицы

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

Транспонирование матрицы — это процесс замены строк на столбцы и наоборот. Это полезная операция при работе с матрицами, особенно в задачах линейной алгебры и обработки изображений.

```
# Транспонирование матрицы
```

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

```
transposed_matrix = [[row[i] for row in matrix] for i in range(len(matrix[0]))]
```

```
print(transposed_matrix)
```

```
# Вывод: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

Задачи

1. Дополните приведенный ниже код так, чтобы список

```
list1= [[1, 7, 8], [9, 7, 102], [102, 106, 105], [100, 99, 98, 103], [1, 2, 3]]
```

имел вид:

```
list1 = [[8, 7, 1], [102, 7, 9], [105, 106, 102], [103, 98, 99, 100], [3, 2, 1]]
```

2. На вход программе подается число n . Напишите программу, которая создает и выводит построчно список, состоящий из n списков $[[1, 2, \dots, n], [1, 2, \dots, n], \dots, [1, 2, \dots, n]]$.
3. На вход программе подается число n . Напишите программу, которая создает и выводит построчно вложенный список, состоящий из n списков $[[1], [1, 2], [1, 2, 3], \dots, [1, 2, \dots, n]]$.

Задачи

3. На вход программе подаются два натуральных числа n и m , каждое на отдельной строке – количество строк и столбцов в матрице. Далее вводятся сами элементы матрицы – слова, каждое на отдельной строке; подряд идут элементы сначала первой строки, затем второй, и так далее.

Напишите программу, которая сначала считывает элементы матрицы один за другим, затем выводит их в виде матрицы.

4. Следом квадратной матрицы называется сумма элементов главной диагонали. Напишите программу, которая выводит след заданной квадратной матрицы.

5. Напишите программу, которая выводит количество элементов квадратной матрицы в каждой строке, больших среднего арифметического элементов данной строки.