Белорусско-Российский университет
Кафедра «Программное обеспечение информационных технологий»

Информатика. Программирование на Python

Тема: Python. Основы. Двумерные и многомерные массивы / списки

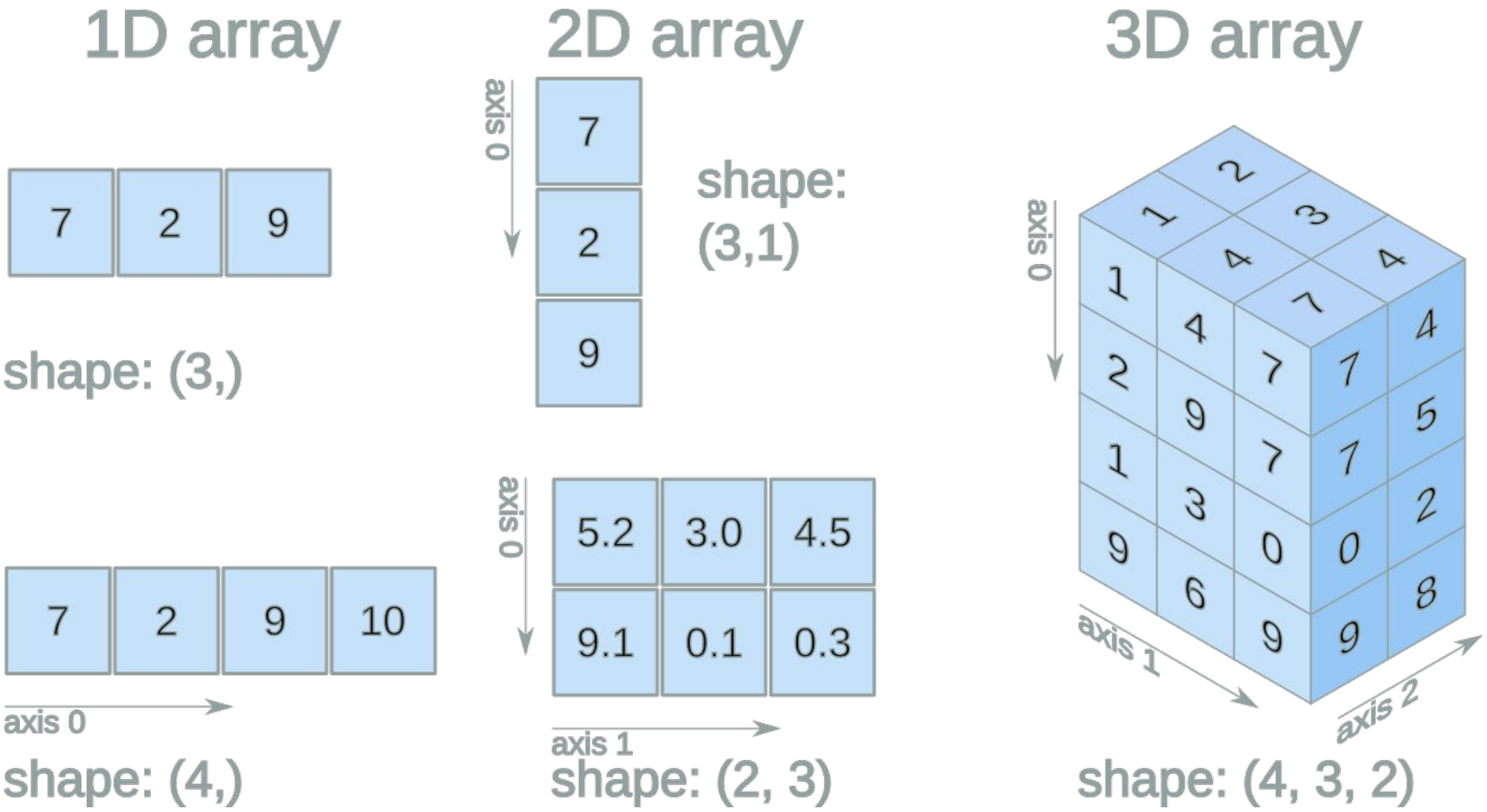
КУТУЗОВ Виктор Владимирович

Могилев, 2021

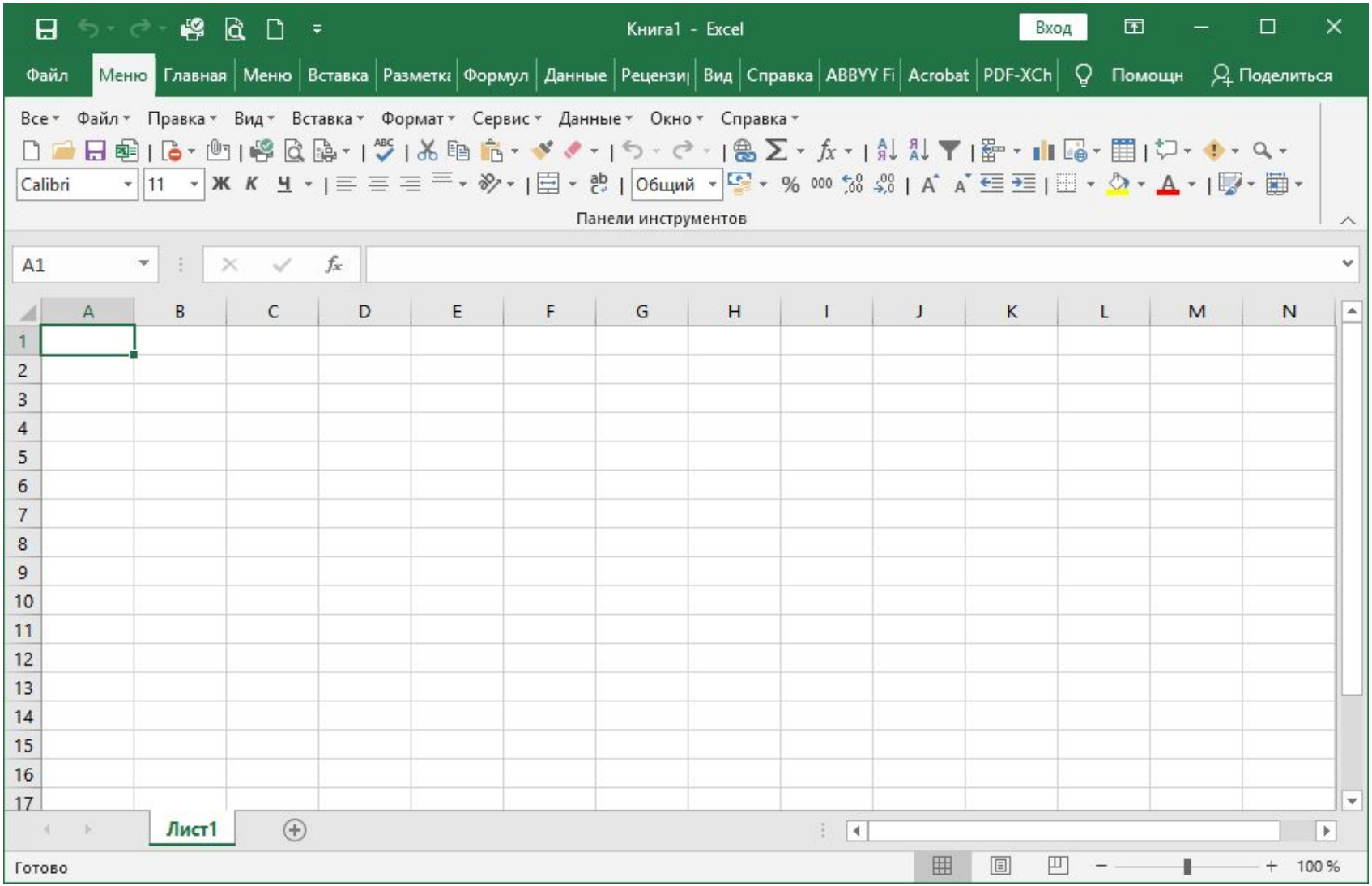
Массивы. Матрицы. Списки

- 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 <u>списки</u>**.

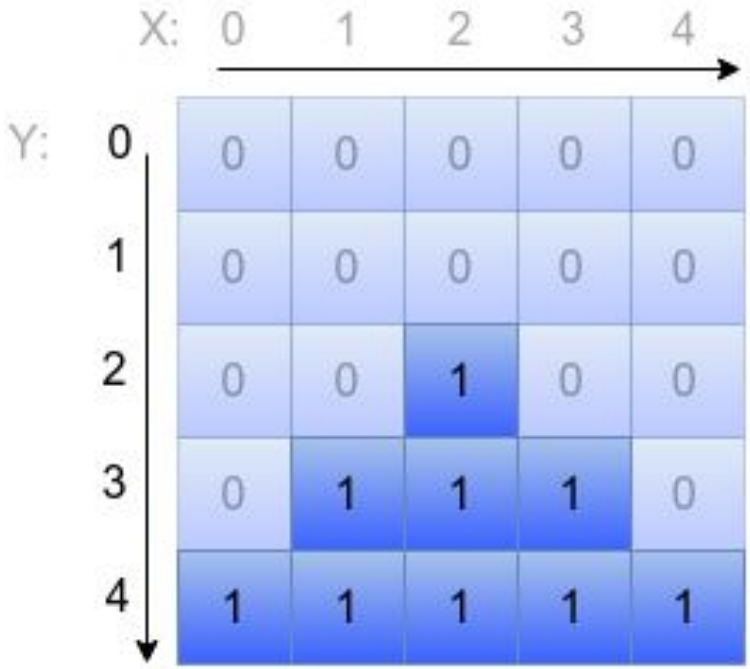
Массивы / Матрицы / Списки

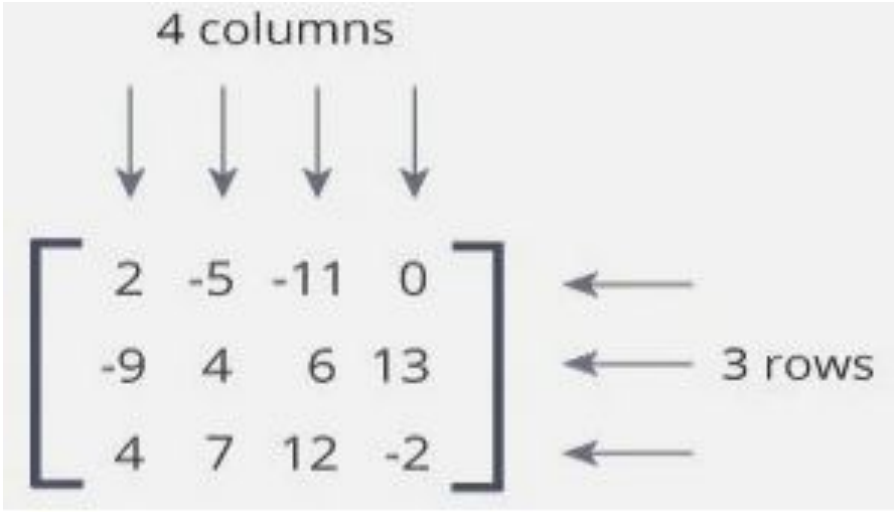


Пример – Microsoft Excel



Массивы / Матрицы / Списки





Двумерный список – Номера элементов

$$
\begin{array}{ccccccccc}
L_{00} & L_{01} & L_{02} & L_{03} & L_{04} & L_{05} & \ldots & L_{0j} \\
L_{10} & L_{11} & L_{12} & L_{13} & L_{14} & L_{15} & \ldots & L_{1j} \\
L_{20} & L_{21} & L_{22} & L_{23} & L_{24} & L_{25} & \ldots & L_{2j} \\
L_{30} & L_{31} & L_{32} & L_{33} & L_{34} & L_{35} & \ldots & L_{3j} \\
L_{40} & L_{41} & L_{42} & L_{43} & L_{44} & L_{45} & \ldots & L_{4j} \\
L_{50} & L_{51} & L_{52} & L_{53} & L_{54} & L_{55} & \ldots & L_{5j} \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
L_{i0} & L_{i1} & L_{i2} & L_{i3} & L_{i4} & L_{i5} & \ldots & L_{ij}
\end{array}
$$

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.

Вывод двумерного списка

```
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]]
```

Вывод двумерного списка

```
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
```

Для обработки и вывода списка, как правило, используют два вложенных цикла. Первый цикл перебирает номер строки, второй цикл бежит по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки

Вывод двумерного списка

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

Вывод двумерного списка

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

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

Для вывода одной строки можно воспользоваться методом join()

Вывод двумерного списка

```
# Программа Python для демонстрации печати
# строки полного многомерного списка
# по строке.

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

for record in a:
    print(record)
```

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

Создание вложенных списков

Создание двумерного списка n x m

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

Это плохой вариант

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

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

Создание двумерного списка n x m

```
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])
```

Обратите внимание на номер id у списков. Если у двух списков id совпадает, то это на самом деле один и тот же список в памяти.

Это плохой вариант

Создание двумерного списка n x m

```
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]
```

Создание двумерного списка n x m

```
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]
```

Создание двумерного списка n x m

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]
```

Создание двумерного списка 5 x 5

```
a1 = []
for j in range(5):
    a2 = []
    for i in range(5):
        a2.append(0)
    a1.append(a2)
# смотрим полученный результат
print(a1[0])
print(a1[1])
print(a1[2])
print(a1[3])
print(a1[4])
```

```
[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]
```

Создание двумерного списка 3 x 6

```
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

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]

Ввод двумерного списка

```
# 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]
```

Ввод двумерного списка

```
# 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]

Пример обработки двумерного списка

Подсчет суммы всех чисел в списке

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for i in range(len(a)):
    for j in range(len(a[i])):
        s += a[i][j]
print(s)
```

45

Подсчет суммы всех чисел в списке

```
a = [[1, 2, 3, 4], [5, 6], [7, 8, 9]]
s = 0
for row in a:
    for elem in row:
        s += elem
print(s)
```

45

То же самое с циклом не по индексу, а по значениям строк

Подсчет суммы всех чисел в списке

```
a = [[34, 24, 68, 63], [64, 66, 56, 79]
, [48, 58, 35, 39]]
N = 3
M = 4
s = 0
for i in range(N):
    for j in range(M):
        s += a[i][j]
print(s)
```

634

Вариант с указанием количества строк и столбцов с перебором в цикле

Методы в многомерных списках

append (): добавляет элемент в конец списка

```
# 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 подсписка

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

[6, 15, 24]

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]]
```

Белорусско-Российский университет
Кафедра «Программное обеспечение информационных технологий»

Информатика. Программирование на Python
Тема: Python. Основы. Двумерные и многомерные массивы / списки.

Благодарю за внимание

КУТУЗОВ Виктор Владимирович

Список использованных источников

1. Python
   https://www.python.org/
2. Google Colaboratory
   https://colab.research.google.com/
3. Python: двумерные и многомерные массивы
   https://otus.ru/nest/post/1441/
4. §9 Списки (Lists). Цикл for по коллекции. Генераторы. Двумерные списки. Срезы
   http://inf-w.ru/?page_id=4808
5. ПитонТьютор. Занятие 9. Двумерные массивы
   https://pythontutor.ru/lessons/2d_arrays/
6. Многомерные списки в Python
   http://espressocode.top/multi-dimensional-lists-in-python/
7. Конспект по Python. Двумерные массивы
   http://www.239.ru/sites/default/files/userdata/konspekt16._dvumernye_massivy_0.pdf
8. Функциональное программирование в Python: lambda, zip, filter, map reduce
   http://pythonicway.com/python-functinal-programming