**Двумерные списки (list)**

Элементы списка в Python могут быть произвольными структурами данных a=[1,2,”one”,”two”, [7,8,9]] Первые два элемента числа, два следующих строки, а последний элемент сам является списком. Можно найти len(a[4])=3. Можно сделать списки большой вложенности. Обычно используют двумерные списки – таблицы.

Для представления двумерных массивов в Python используют вложенные списки.

Представим таблицу, как список, где элементами являются строки таблицы.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | 1 | 2 | 3 |
| 1 | 4 | 5 | 6 |
| 2 | 7 | 8 | 9 |
| 3 | 10 | 11 | 12 |

a=[[1,2,3], [4,5,6],[7,8,9],[10,11,12]]

В Python в общем случае строчки могут быть разной длины. Таблица может оказаться не квадратной или прямоугольной, а изломанной.

**Основные операции**

Для обхода таблицы используют вложенные циклы. Одним циклом мы идем по строчкам 0,1,2,… Обрабатывая каждую строчку мы идем по всем столбцам.

**Поиск наибольшего элемента**

1 вариант

m = a[0][0]

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

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

 **if** a[i][j] > m:

 m = a[i][j]

2 вариант

m = a[0][0]

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

 **for** j **in range**(**len**(row)):

 **if** row[j] > m:

 m = row[j]

3 вариант

m = a[0][0]

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

 **for** el **in** row:

 **if** el > m:

 m = el

**Вывод двумерного списка**

print(a)

Список будет выведен примерно в таком же виде, как он задается в программе.

1 вариант

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

 **for** el **in** row:

 **print**(el, end= " ")

 **print**()

2 вариант

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

 **print**(" ".**join**(**map**(**str**, row)))

3 вариант С выравниванием по столбцам по правому краю

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

 **for** el **in** row:

 **print**(**str**(el).**rjust**(3," "),end=" ")

 **print**()

**rjust** – выравнивание по правому краю

**ljust** – выравнивание по левому краю

**Создание двумерных списков**

3 4

1 1 1 1

2 2 2 2

3 3 3 3

1 вариант

m, n = **map**(**int**, **input**().**split**())

a = [0] \* m

**for** i **in** **range**(m):

 a[i] = **list**(**map**(**int**, **input**().**split**()))

2 вариант

m, n = **map**(**int**, **input**().**split**())

a = []

**for** i **in** **range**(m):

 a.**append**(**list**(**map**(**int**, **input**().**split**())))

3 вариант при помощи генератора

m, n = **map**(**int**, **input**().**split**())

a = **[list**(**map**(**int**, **input**().**split**())) **for** i **in** range(m)]

**Создать массив 100×100 из нулей**

1 вариант

a = []

**for** i **in** **range**(m):

 a.**append**([0] \* n)

2 вариант

a = [0] \* m

**for** i **in** **range**(m):

 a[i] = [0] \* n

3 вариант

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

Пусть необходимо создать пустой список — скажем, длиной 10 — и заполнить его нулями (не может же он быть совсем пустой). Это легко: [0]\*10. А если нужен список из 10 таких списков — двумерный список? Могло бы показаться, что сработает a=[[0]\*10]\*10, но это не так. (Попробуйте понять, почему. Подсказка: создайте такой список, измените в нём один элемент и посмотрите, что получилось.)

**«Глубокое» копирование:**

**import copy**

**D = copy.deepcopy(C)**

**Алгоритмы обработки двумерных списков**

**Нахождение суммы элементов двумерного списка в каждой строки**

**1 способ.**

Можно к элементам списка обращаться по индексам.

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

 s = 0

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

 s += a[i][j]

 **print**(s)

**2 способ.**

Можно в циклах перебирать непосредственно строчки и элементы двумерного массива

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

 s = 0

 **for** el **in row**:

 s += el

 **print**(s)

**Нахождение суммы элементов двумерного списка в каждом столбце**

В данном случае к элементам списка придется обращаться по индексу.

**for** j **in range**(**len**(a[0])):

 s = 0

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

 s += a[i][j]

 **print**(s)

**Нахождение суммы элементов двумерного списка, стоящих на главной диагонали**

**1 способ.**

Условие того, что элемент находится на главной диагонали, заключается в том, что индексы элемента равны. Т.е. можно среди всех элементов массива выбрать те, которые стоят на главной диагонали и прибавить их к сумме.

s = 0

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

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

 **if** i == j:

 s += a[i][j]

**print**(s)

**2 способ.**

Для того чтобы обработать все элементы главной диагонали достаточно одного цикла с параметром.

s = 0

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

 s += a[i][i]

**print**(s)

**Нахождение суммы элементов двумерного списка, стоящих на побочной диагонали**

Побочная диагональ – это линия, соединяющая правый верхний и левый нижний угол в квадратной матрице.

Для того чтобы обработать все элементы побочной диагонали также достаточно одного цикла с параметром.

s = 0

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

 s += a[i][n – 1 - i]

**print**(s)

**Нахождение суммы элементов двумерного списка, стоящих над (правее) главной диагональю**

**1 способ.**

Условие того, что элемент находится над главной диагональю, заключается в том, что у данного элемента индекс строки меньше индекса столбца. Т.е. можно среди всех элементов массива выбрать те, которые стоят над главной диагональю и прибавить их к сумме.

s = 0

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

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

 **if** i < j:

 s += a[i][j]

**print**(s)

**2 способ.**

Для того чтобы обработать все элементы над главной диагональю можно записать следующие два вложенных цикла:

s = 0

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

 **for** j **in range**(i + 1, **len**(a[i])):

 s += a[i][j]

**print**(s)

**Нахождение суммы элементов двумерного списка, стоящих под (левее) главной диагональю**

**1 способ.**

Условие того, что элемент находится под главной диагональю, заключается в том, что у данного элемента индекс строки больше индекса столбца. Т.е. можно среди всех элементов массива выбрать те, которые стоят под главной диагональю и прибавить их к сумме.

s = 0

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

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

 **if** i > j:

 s += a[i][j]

**print**(s)

**2 способ.**

Для того чтобы обработать все элементы под главной диагональю можно записать следующие два вложенных цикла:

s = 0

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

 **for** j **in range**(i):

 s += a[i][j]

**print**(s)