

Теоретический материал

Теоретический материал по теме "Двумерные массивы"

Сайт: [Дистанционная подготовка](#)

Курс: Д. П. Кириенко. Программирование на языке Python (школа 179 г. Москвы)

Book: Теоретический материал

Printed by: maung myo

Date: Воскресенье 4 Март 2018, 01:27

Table of Contents

[Обработка и вывод вложенных списков](#)

[Создание вложенных списков](#)

[Ввод двумерного массива](#)

[Пример обработки двумерного массива](#)

[Вложенные генераторы двумерных массивов](#)

Обработка и вывод вложенных списков

Часто в задачах приходится хранить прямоугольные таблицы с данными. Такие таблицы называются матрицами или двумерными массивами. В языке программирования Питон таблицу можно представить в виде списка строк, каждый элемент которого является в свою очередь списком, например, чисел. Например, создать числовую таблицу из двух строк и трех столбцов можно так:

```
A = [ [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.

Для обработки и вывода списка как правило используется два вложенных цикла. Первый цикл по номеру строки, второй цикл по элементам внутри строки. Например, вывести двумерный числовой список на экран построчно, разделяя числа пробелами внутри одной строки, можно так:

```
for i in range(len(A)):
    for j in range(len(A[i])):
        print(A[i][j], end = ' ')
    print()
```

То же самое, но циклы не по индексу, а по значениям списка:

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

Естественно для вывода одной строки можно воспользоваться методом join:

```
for row in A:
    print(' '.join(list(map(str, row))))
```

Используем два вложенных цикла для подсчета суммы всех чисел в списке:

```
S = 0
for i in range(len(A)):
    for j in range(len(A[i])):
        S += A[i][j]
```

Или то же самое с циклом не по индексу, а по значениям строк:

```
S = 0
for row in A:
    for elem in row:
        S += elem
```

Создание вложенных списков

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

Очевидное решение оказывается неверным:

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

В этом легко убедиться, если присвоить элементу $A[0][0]$ значение 1, а потом вывести значение другого элемента $A[1][0]$ — оно тоже будет равно 1! Дело в том, что $[0] * m$ возвращает ссылку на список из m нулей. Но последующее повторение этого элемента создает список из n элементов, которые являются ссылкой на один и тот же список (точно так же, как выполнение операции $B = A$ для списков не создает новый список), поэтому все строки результирующего списка на самом деле являются одной и той же строкой.

Таким образом, двумерный список нельзя создавать при помощи операции повторения одной строки. Что же делать?

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:

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

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

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

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

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

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.

Ввод двумерного массива

Пусть программа получает на вход двумерный массив, в виде n строк, каждая из которых содержит m чисел, разделенных пробелами. Как их считать? Например, так:

```
A = []
for i in range(n):
    A.append(list(map(int, input().split())))
```

Или, без использования сложных вложенных вызовов функций:

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

Можно сделать то же самое и при помощи генератора:

```
A = [ list(map(int, input().split())) for i in range(n)]
```

Пример обработки двумерного массива

Пусть дан квадратный массив из n строк и n столбцов. Необходимо элементам, находящимся на главной диагонали, проходящей из левого верхнего угла в правый нижний (то есть тем элементам $A[i][j]$, для которых $i=j$) присвоить значение 1, элементам, находящимся выше главной диагонали – значение 0, элементам, находящимся ниже главной диагонали – значение 2. То есть получить такой массив (пример для $n=4$):

```
1 0 0 0
2 1 0 0
2 2 1 0
2 2 2 1
```

Рассмотрим несколько способов решения этой задачи. Элементы, которые лежат выше главной диагонали – это элементы $A[i][j]$, для которых $i < j$, а для элементов ниже главной диагонали $i > j$. Таким образом, мы можем сравнивать значения i и j и по ним определять значение $A[i][j]$. Получаем следующий алгоритм:

```
for i in range(n):
    for j in range(n):
        if i < j:
            A[i][j] = 0
        elif i > j:
            A[i][j] = 2
        else:
            A[i][j] = 1
```

Данный алгоритм плох, поскольку выполняет одну или две инструкции `if` для обработки каждого элемента. Если мы усложним алгоритм, то мы сможем обойтись вообще без условных инструкций.

Сначала заполним главную диагональ, для чего нам понадобится один цикл:

```
for i in range(n):
    A[i][i] = 1
```

Затем заполним значением 0 все элементы выше главной диагонали, для чего нам понадобится в каждой из строк с номером i присвоить значение элементам $A[i][j]$ для $j=i+1, \dots, n-1$. Здесь нам понадобятся вложенные циклы:

```
for i in range(n):
    for j in range(i + 1, n):
        A[i][j] = 0
```

Аналогично присваиваем значение 2 элементам $A[i][j]$ для $j=0, \dots, i-1$:

```
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
```

Можно также внешние циклы объединить в один и получить еще одно, более компактное решение:

```
for i in range(n):  
    for j in range(0, i):  
        A[i][j] = 2  
    A[i][i] = 1  
    for j in range(i + 1, n):  
        A[i][j] = 0
```

А вот такое решение использует операцию повторения списков для построения очередной строки списка. i -я строка списка состоит из i чисел 2, затем идет одно число 1, затем идет $n-i-1$ число 0:

```
for i in range(n):  
    A[i] = [2] * i + [1] + [0] * (n - i - 1)
```

А можно заменить цикл на генератор:

```
A = [ [2] * i + [1] + [0] * (n - i - 1) for i in range(n)]
```

Вложенные генераторы двумерных массивов

Для создания двумерных массивов можно использовать вложенные генераторы, разместив генератор списка, являющегося строкой, внутри генератора для строк. Например, сделать список из n строк и m столбцов при помощи генератора, создающего список из n элементов, каждый элемент которого является списком из m нулей:

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

Но при этом внутренний список также можно создать при помощи, например, такого генератора: `[0 for j in range(m)]`. Вложив один генератор в другой получим вложенные генераторы:

```
[ [0 for j in range(m)] for i in range(n)]
```

Но если число 0 заменить на некоторое выражение, зависящее от i (номер строки) и j (номер столбца), то можно получить список, заполненный по некоторой формуле.

В этом листке вам нужно будет придумать генераторы для заданных двумерных массивов.

Например, пусть нужно задать следующий массив (для удобства добавлены дополнительные пробелы между элементами):

```
0 0 0 0 0 0
0 1 2 3 4 5
0 2 4 6 8 10
0 3 6 9 12 15
0 4 8 12 16 20
```

В этом массиве $n = 5$ строк, $m = 6$ столбцов, и элемент в строке i и столбце j вычисляется по формуле: $A[i][j] = i * j$.

Для создания такого массива можно использовать генератор:

```
[[ i * j for j in range(m)] for i in range(n)]
```