


Вложенные циклы. Примеры задач с вложенными циклами

На предыдущих занятиях мы с вами в деталях познакомились с работой операторов циклов `for` и `while`. На этом занятии сделаем следующий важный шаг и узнаем, как реализуются и работают вложенные циклы.

Само это название уже говорит, что один оператор цикла можно вложить в другой. Это могут быть два цикла `for` или два цикла `while` или смешанные варианты. Давайте вначале разберемся, как работают эти конструкции. Принцип у всех один, поэтому, для простоты, я возьму два цикла `for` (один вложен в другой). Эти циклы будут просто пробегать диапазоны чисел, первый от 1 до 3, а второй – от 1 до 5:

```
for i in range(1, 4):  
    for j in range(1, 6):  
        print(f"i = {i}, j = {j}")  
    print()
```



1	1	2	3	4	5
2	1	2	3	4	5
3	1	2	3	4	5

Во втором вложенном цикле мы будем выводить значения `i` и `j` в строку без перехода на новую строку. А после завершения работы вложенного цикла вызовем функцию `print()`, как раз, для перевода курсора на новую строку. В результате выполнения этой программы, мы получим таблицу значений переменных `i` и `j`.

Почему получились именно такие значения? Вначале у нас счетчик `i` принимает значение 1, а счетчик `j` пробегает числа от 1 до 5, в итоге получаем первую строку. После завершения вложенного цикла, срабатывает функция `print()` и курсор переходит на новую строку. После этого переходим ко второй итерации первого цикла и `i = 2`. Счетчик `j` снова проходит значения от 1 до 5 и получаем вторую строку. На следующей итерации первого цикла `i = 3`, `j` проходит от 1 до 5 и получаем третью строку. То есть, у нас вложенный цикл `for` трижды запускался заново и каждый раз `j` изменялось от 1 до 5. Это и есть принцип работы вложенного цикла – на каждой итерации он отработывает снова и снова, пока не завершится первый цикл.

Теперь второй вопрос – зачем все это нужно? Давайте представим, что у нас есть вложенный (двумерный) список: `a = [[1, 2, 3, 4], [2, 3, 4, 5],`

(О таких списках мы с вами уже говорили и вам здесь все должно быть понятно). Так вот, если мы будем перебирать его элементы с помощью одного оператора цикла for:

```
for row in a:  
    print(row, type(row))
```

То переменная row будет ссылаться сначала на первый вложенный список, затем, на второй и потом на третий. Но, так как row ссылается на список, то есть, на итерируемый объект, то нам ничто не мешает перебрать его элементы с помощью второго, вложенного цикла for:

```
for row in a:  
    for x in row:  
        print(x, type(x), end=' ')  
    print()
```

Как видите, теперь в консоль выводятся числа типа int, то есть, мы обращаемся непосредственно к элементам этого двумерного списка.

Ну, хорошо, а все-таки, зачем это может быть нужно? Например, так можно выполнить сложение значений из двух одинаковых двумерных списков:

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

И сформировать на их основе третий список: `c = []` следующим образом:

```
for i, row in enumerate(a):  
    r = []  
    for j, x in enumerate(row):  
        r.append(x + b[i][j])  
  
    c.append(r)
```

1	2	3	4		1	1	1	1	
2	3	4	5	+	2	2	2	2	=
3	4	5	6		3	3	3	3	

```
print(c)
```

Я здесь воспользовался еще одной уже знакомой нам функцией `enumerate()`, которая возвращает индекс и значение текущего элемента. Это удобно для реализации данной программы. Внутри первого цикла мы каждый раз создаем новый пустой список и с помощью метода `append()` добавляем в его конец новый элемент как сумму значений из списков `a` и `b`. Полученную строку (список `r`) мы, затем, добавляем в основной список `c`. Так вычисляется сумма значений элементов двух одинаковых списков `a` и `b`.

Как видите, для реализации данной программы нам потребовался вложенный оператор цикла `for`. И это лишь один маленький пример. Другой пример, пусть у нас имеется текст, представленный в виде списка:

```
t = ["- Скажи-ка, дядя, ведь не  
"Я Python выучил с каналом"  
"Балакирев что раздавал?",  
"Ведь были ж заданья боевые,  
"Да, говорят, еще какие!",  
"Недаром помнит вся Россия  
"Как мы рубили их тогда!"  
]
```

```
for i, line in enumerate(t):  
    while line.count(' '):  
        line = line.replace(' ', '')  
  
    t[i] = line  
  
print(t)
```

Здесь в строках присутствуют два и более пробелов. Наша задача удалить их и оставить только один. Сделаем это с помощью вложенных циклов. В первом цикле for будем перебирать строки –элементы списка, а во втором (вложенном) цикле while удалять лишние пробелы:

В качестве условия цикла мы здесь вызываем метод count(), который подсчитывает число фрагментов из двух пробелов подряд. Как только их станет 0 – это будет означать False и цикл завершится. Преобразованная строка становится новым i-м элементом списка и в конце результат выводим в консоль.

Следующий пример. Предположим, вначале мы формируем вложенный список размером M x N элементов. Причем, M, N вводим с клавиатуры. Вначале сформируем список, состоящий из всех нулей:

```
M, N = list(map(int, input().split()))  
  
zeros = []  
for i in range(M):  
    zeros.append([0]*N)  
  
print(zeros)
```

А после этого все элементы заменим на единицы, используя вложенные циклы:

```
for i in range(M):  
    for j in range(N):  
        zeros[i][j] = 1
```

Как видите, в целом, все достаточно просто.

И последний пример. Пусть у нас имеется квадратный список (размерности совпадают):

```
A = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 16]]
```

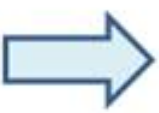
Необходимо поменять строки на столбцы и получить новое представление этого же списка. Для этого достаточно поменять

местами элементы, стоящие выше главной диагонали с элементами, стоящими ниже главной диагонали. То есть, у нас счетчик i будет меняться от 0 до 3, а счетчик j от $i+1$ до 3. Затем, соответствующие элементы будем менять между собой:

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

```
for r in A:
    for x in r:
        print(x, end='\t')
    print()
```

		j			
		0	1	2	3
i->	0	1	2	3	4
	1	5	6	7	8
	2	9	10	11	12
	3	13	14	15	16



1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

$$A[i][j], A[j][i] = A[j][i], A[i][j]$$

Как видите, у нас получилось нужно преобразование. В математике это называется транспонированием матрицы.