Багатомірні масиви. Рекурсія.

Створення багатомірного масиву та його індексація

Масиви в Python являють собою набір посилань на елементи, які збережені окремо. Тому в масиві можуть бути будь-які типи даних одночасно. За тим самим принципом у масиві можна залишити посилання на інший масив, при тому не важливо, буде це список, кортеж або словник.

При вкладенні масивів один в одний масив, у котрий вкладають, буде називатися *головним* або *батьківським*. А масив всередені батьківського – *вкладеним* або *дочірнім*. Такі назви працюють на будь-якому рівні вкладеності, відносно своїх батьківских масивів.

```
>>> arr1 = [1,2,3]
>>> arr2 = (2,4,7,'k')
>>> main_arr = [arr1, 5,6,arr2,'q']
>>> print(main_arr)
[[1, 2, 3], 5, 6, (2, 4, 7, 'k'), 'q']
```

Таким чином, ми можемо взяти масив за його індексом у головному масиві.

```
>>> print(main_arr[3])
(2, 4, 7, 'k')
```

Якщо нам потрібно взяти конкретний елемент зі вкладеного масиву, то після вказання індексу дочірнього масиву вказуємо індекс елементу, що нам потрібен, у нових квадратних дужках.

```
>>> print(main_arr[3][2])
7
>>> print(arr2[2])
7
```

Приведені в прикладі вирази є рівносильними, так як кортеж arr2 вкладений в main_arr на 4-ту позицію (має індекс 3).

Якщо всередені вкладеного масиву є ще один масив, ми можемо достати з нього елемент, вказавши ще один індекс в окремих квадратних дужках. Така дія розповсюджується на нескінченну кількість вкладених масивів.

Будь-які масиви, в тому числі і багатомірні, підтримують зворотню індексацію:

```
>>> print(main_arr[-2][-2])
7
```

Дії з багатомірними масивами

Для перебору масивів доцільніше використовувати цикл for.

Якщо просто перебрати масив як одномірний, то програма поверне вкладені масиви.

```
>>> arr = [[1,2,3,4],[5,6],(7,8,9)]
>>> for i in arr:
        print(i)


[1, 2, 3, 4]
[5, 6]
(7, 8, 9)
```

Але можливо всередені одного циклу зробити ще один, щоб він робив перебор вкладеного масиву. Такий цикл аналогічно називається *вкладеним*.

```
>>> for i in arr:
        if type(i) == list or type(i) == tuple:
                for k in i:
                        print(k)
        else:
                print(i)


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

Данний приклад виводить кожен елемент дочірніх масивів. Також у ньому робиться перевірка на випадок, якщо у батьківському списку окрім масивів є неітеруємий елемент. У випадку, якщо цикл for отримає такий елемент, відбудеться помилка TypeError.

Рекурсія

Також перебор та вивод елементів можна виконати рекурсивною функцією.

Рекурсивна функція – це функція, яка викликає саму себе в процессі виконання. При створенні рекурсивної функції потрібно бути уважним, бо, як і у випадку з циклом while, можна випадково створити функцію з нескінченними викликами. Тому потрібно написати *базовий випадок.*

Базовий випадок рекурсивної функції – це кінцевий випадок, коли рекурсивна функція не визиває саму себе.

Прикладом рекурсивної функції для перебору багатомірного масиву і виводу його елементів є:

```
def recurs(mas):
    for i in mas:
        if type(i)==list or type(i)==tuple:
            recurs(i)
        else:
            print(i)

masive_4 = [1, [2,[3,4],5], [6,[7,[8,[9,[10,11],12]]],13], 14]
recurs(masive_4)
```

Вивод:

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

Контрольні питання:

1. Як розглядається кожен елемент у масиві в Python?
2. Що собою представляє багатовимірний масив?
3. Як перебрати двовимірний масив за допомогою for?
4. Що таке рекурсивна функція? Базовий випадок?