Многомерные массивы

Многомерные массивы структуры данных, которые содержат элементы, расположенные в двух или более измерениях. Работа с вложенными списками в Python позволяет создавать и манипулировать многомерными массивами

Вложенные списки представляют собой списки, содержащие другие списки в качестве элементов. Каждый внешний список может быть рассмотрен как строка или столбец многомерного массива

Примеры операций с вложенными списками для работы с многомерными массивами

01 Создание многомерного массива с использованием вложенных списков

```
multidimensional_list = [[1, 2, 3], [4, 5, 6]]
```

В этом примере multidimensional_list представляет собой многомерный массив размером 2x3, созданный с помощью вложенных списков

02 Доступ к элементам многомерного массива

```
print(multidimensional_list[0][0])  # Выводит 1
print(multidimensional_list[1][2])  # Выводит 6
```

В этом примере осуществляется доступ к элементам многомерного массива по индексам внешнего и внутреннего списка

03 Итерация по многомерному массиву

```
for row in multidimensional_list:
    for item in row:
        print(item)
```

В этом примере используется вложенный цикл для итерации по каждому элементу во внутреннем списке каждой строки многомерного массива

04 Манипуляции с многомерным массивом

```
multidimensional_list[0][0] = 10  # Изменение значения элемента
multidimensional_list.append([7, 8, 9])  # Добавление новой строки в конец
multidimensional_list.pop(1)  # Удаление строки по индексу
```

В этом примере показаны некоторые операции по изменению многомерного массива, такие как изменение значения элемента, добавление новой строки и удаление строки

05 Получение размера многомерного массива

```
rows = len(multidimensional_list)  # Количество строк
cols = len(multidimensional_list[0])  # Количество столбцов
```

В этом примере используется функция len() для определения количества элементов во внешнем списке, чтобы получить количество строк многомерного массива. Затем с помощью функции len() и доступа по индексу к первому внутреннему списку определяется количество столбцов

06 Объединение вложенных списков в один плоский список

```
flat_list = [item for sublist in multidimensional_list for item in
sublist]  # Разворачивание многомерного массива в одномерный списком
```

В этом примере используется конструкция list comprehension для прохода по каждому внутреннему списку и каждому элементу в нем. При этом элементы добавляются в новый список flat_list, чтобы получить одномерный список из всех элементов многомерного массива

07 Создание пустого многомерного массива

```
empty_list = [[None] * num_cols for _ in range(num_rows)]
# Создание пустого многомерного массива заданного размера
```

В этом примере используется вложенный list comprehension для создания списка указанного размера с пустыми значениями (None в данном случае). При этом создается новый список empty_list, содержащий num_rows внешних списков. Каждый внешний список имеет num_cols пустых значений.

08 Копирование многомерного массива

```
import copy
copy_list = copy.deepcopy(multidimensional_list)
# Глубокое копирование многомерного массива
```

В этом примере используется функция deepcopy() из модуля copy, чтобы создать глубокую копию многомерного массива. Глубокое копирование гарантирует, что создается полностью независимая копия массива, а не только ссылка на него

---

09 Транспонирование многомерного массива

```
transposed_list = list(map(list, zip(*multidimensional_list)))
# Транспонирование многомерного массива
```

Используется функция zip() в сочетании с оператором распаковки * для группировки элементов столбцов в кортежи. Затем с помощью map() преобразуется каждый кортеж в список. И наконец, результат передается в функцию list() для преобразования в список.
В результате получается транспонированный многомерный массив

---

10 Поиск значения в многомерном массиве

```
value = 5
found = any(value in sublist for sublist in
multidimensional_list)  # Проверка, содержится ли значение в
многомерном массиве
```

Используется генераторное выражение с оператором any() для проверки, содержится ли заданное значение в любом из внутренних списков многомерного массива. Функция any() возвращает True, если в генераторном выражении найдено хотя бы одно совпадение, и False, если ни одного совпадения не найдено

Это лишь несколько примеров операций и методов, которые могут быть полезны при работе с вложенными списками в Python для создания и манипуляции с многомерными массивами. Все эти методы могут быть довольно гибкими и могут быть дополнены другими операциями в зависимости от конкретных потребностей вашего кода

Работа с вложенными списками предоставляет базовый, но гибкий способ работы с многомерными массивами в Python. В языке Python многомерные массивы также могут быть представлены с использованием встроенного модуля array или библиотеки NumPy

Итог

В данном теме мы рассмотрели способы работы с многомерными массивами с использованием вложенных списков. Вложенные списки позволяют создавать, изменять, итерировать и манипулировать многомерными массивами. Они обеспечивают базовый, но гибкий способ работы с многомерными данными в Python