Список в Python -- это встроенный тип (класс) данных, представляющий собой одну из разновидностей структур данных. Структуру данных можно представить как сложную единицу, объединяющую в себе группу более простых. Каждая разновидность структур данных имеет свои особенности.

Отличительной особенностью списка является то, что он -- изменяемая последовательность элементов произвольных типов.

Давайте же объявим список a, содержащий в себе числа от 1 до 3 и строку «fox». Это делается очень просто:

a=[1, 2, 3, 'fox']

Помимо этого можно объявить пустой список или же с помощью конструктора класса:

s=[]

s=list(),

или же повторить один элемент несколько раз:

b=[0]*5

Пример выше вернет список из пяти нулей.

Часто говорят, что массив – это таблица. Однако пока мы рассматривали только таблицы, одна из размерностей которых равна единице. Это так называемые одномерные массивы. Однако возможны двух и даже трёхмерные массивы. Давайте рассмотрим примеры их объявлений.

На основании примера выше, есть желание объявить его как

b = [[0]*10]*10.

Однако при таком объявлении создаётся десять указателей на массив из 10 нулей (один и тот же). Таким образом, если что-то изменить в одной из строк, поменяются элементы этого столбца во всех строках.

Корректное объявление многомерных массивов неразрывно связано с методами списков. Методами называют функции, выполняющие задачи для определенного класса. Вспомним некоторые из них:

- **append(item)**: добавляет элемент item в конец списка
- **insert(index, item)**: добавляет элемент item в список по индексу index
- **remove(item)**: удаляет элемент item. Удаляется только первое вхождение элемента. Если элемент не найден, генерирует исключение ValueError
- **clear()**: удаление всех элементов из списка
- **index(item)**: возвращает индекс элемента item. Если элемент не найден, генерирует исключение ValueError
- **pop([index])**: удаляет и возвращает элемент по индексу index. Если индекс не передан, то просто удаляет последний элемент.
- **count(item)**: возвращает количество вхождений элемента item в список
- **sort([key])**: сортирует элементы. По умолчанию сортирует по возрастанию. Но с помощью параметра key мы можем передать функцию сортировки.

- **reverse()**: расставляет все элементы в списке в обратном порядке

Кроме того, Python предоставляет ряд встроенных функций для работы со списками:

- **len(list)**: возвращает длину списка
- **sorted(list, [key])**: возвращает отсортированный список
- **min(list)**: возвращает наименьший элемент списка
- **max(list)**: возвращает наибольший элемент списка

Вернёмся к многомерным спискам. Очевидное решение оказалось неправильным, однако всё ещё возможно создать вложенные списки таким образом:

```
a = [[1, 2, 3, 4], [5, 6, 7], ['fox', 8, 9]]
```

или так:

```
n = int(input())
m = int(input())
a = [0] * n
for i in range(n):
    a[i] = [0] * m
```

Помимо этого, можно воспользоваться так называемыми генераторами списков:

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

Или же добавлять в цикле несколько пустых списков и работать непосредственно с ними, используя append.