Python III:
Списки

Списки (list) - это *упорядоченная* коллекция элементов, которые могут быть различных типов.

Списки могут быть *изменяемыми*, то есть вы можете добавлять, удалять и изменять элементы списка после его создания. Это отличает списки от других типов данных, таких как кортежи (tuple), которые являются неизменяемыми.

Списки в Python могут содержать элементы любого типа данных, включая другие списки. Создать список можно, используя квадратные скобки [ ] и разделяя элементы запятыми.

```
my_list = [1, 'two', 3.0, [4, 5, 6]]
```

Чтобы получить доступ к элементам списка, можно использовать индексы, начинающиеся с 0. Например, чтобы получить первый элемент списка my_list, можно написать my_list[0]. Если вы хотите получить последний элемент списка, вы можете использовать отрицательный индекс, например, my_list[-1].

Основные методы для изменения списков

append() добавляет элемент в конец списка

insert() добавляет элемент в список по указанному индексу

extend() добавляет элементы из другого списка в конец текущего списка

remove() удаляет первый элемент с указанным значением из списка. Если элемент не найден, возникает ошибка ValueError

pop() удаляет и возвращает элемент с указанным индексом из списка. Если индекс не указан, удаляется и возвращается последний элемент списка

Основные методы для работы со списками

count() возвращает количество элементов с указанным значением в списке

index() возвращает индекс первого элемента с указанным значением в списке. Если элемент не найден, возникает ошибка ValueError

sort() сортирует элементы списка в порядке возрастания. Метод не возвращает новый отсортированный список, а изменяет исходный список

Дополнительно о списках:

Изменение элементов списка: элементы списка можно изменять по индексу. Например:

```
lst = [0, 1, 2, 3, 4]
lst[2] = 'two'
print(lst) # [0, 1, 'two', 3, 4]
```

Удаление элементов списка: можно удалять элементы из списка с помощью метода remove() или оператора del. Например:

```
lst = [0, 1, 'two', 3, 4]
lst.remove('two')
del lst[0]
print(lst) # [1, 3, 4]
```

Длина списка: можно узнать длину списка с помощью функции len(). Например:

```
lst = [1, 2, 3, 4, 5]
print(len(lst)) # 5
```

Срезы списка: можно извлекать подсписки из списка с помощью срезов. Например:

```
lst = [0, 1, 2, 3, 4]
print(lst[1:3]) # [1, 2]
print(lst[:3]) # [0, 1, 2]
print(lst[3:]) # [3, 4]
```

Поиск элементов в списке: можно проверять, присутствует ли элемент в списке с помощью оператора in. Например:

```
lst = [1, 2, 3, 4, 5]
print(3 in lst) # True
print(6 in lst) # False
```

Копирование списка: можно создать копию списка с помощью метода copy() или оператора [:]. Например:

```
lst = [1, 2, 3, 4, 5]
lst_copy = lst.copy()
lst_copy[0] = 0
print(lst) # [1, 2, 3, 4, 5]
print(lst_copy) # [0, 2, 3, 4, 5]
```

Объединение списков: можно объединять два списка с помощью оператора +. Например:

```
lst1 = [1, 2, 3]
lst2 = [4, 5, 6]
lst3 = lst1 + lst2
print(lst3) # [1, 2, 3, 4, 5, 6]
```

Многомерные списки: списки могут содержать другие списки в качестве элементов, что позволяет создавать многомерные списки (матрицы, тензоры и т.д.). Например:

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(matrix[0][1]) # 2
```

Можно обратить порядок элементов в списке с помощью метода reverse(). Например:

```
lst = [1, 2, 3, 4, 5]
lst.reverse()
print(lst) # [5, 4, 3, 2, 1]
```

Срезы с шагом: можно выбирать элементы списка через заданный шаг с помощью срезов. Например:

```
lst = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(lst[::2]) # [0, 2, 4, 6, 8]
print(lst[1::2]) # [1, 3, 5, 7, 9]
```

Помимо len(), можно так же использовать sum(), max(), min(). Например:

```
lst = [1, 2, 3, 4, 5]
print(len(lst)) # 5
print(sum(lst)) # 15
print(max(lst)) # 5
print(min(lst)) # 1
```

Также списки можно умножать:

```
lst = [1, 2, 3]
new_lst = lst * 3
print(new_lst) # [1, 2, 3, 1, 2, 3, 1, 2, 3]
```