# Урок 12.02. Методы extend и count

В реальных программах, конечно же, часто приходится работать не с одним списком. И давайте познакомимся с еще несколькими методами, которые могут улучшить вашу программу, и каких ошибок вы можете избежать.

И вспомним нашу предыдущую программу про «Рейтинг фильмов».

И давайте представим такую ситуацию: На нашем сайте уже два пользователя Анна и Борис. И вот Борис зашел на сайт, посмотрел список фильмов Анны и понял, что тоже посмотрел все эти фильмы и хотел бы их добавить себе. Давайте реализуем программу, которая добавляет все объекты одного списка в другой список.

my\_list = ['Дюна 2', 'Матрица', 'Игра', 'Изгой']

your\_list = ['Начало', 'Выживший', 'Король Лев']

my\_list.append(your\_list)

print(my\_list)

Запустим. Вроде всё работает, но что-то не так: у нас тут какое-то еще квадратные скобки. Если приглядеться, то мы увидим, что это список внутри другого списка, и получается что последний объект нашего списка my\_list это целый список. И в этом мы можем убедиться, если выведем последний элемент нашего списка.

my\_list = ['Дюна 2', 'Матрица', 'Игра', 'Изгой']

your\_list = ['Начало', 'Выживший', 'Король Лев']

my\_list.append(your\_list)

print(my\_list[-1])

Так, понятно, но как же всё же перенести наши фильмы? И как всегда мы можем решить эту задачу через циклы (сделать цикл по списку your\_list и каждый элемент этого списка добавлять в наш список my\_list). И тут нам поможет новый метод списков это extend, который переводится как «расширение».

my\_list = ['Дюна 2', 'Матрица', 'Игра', 'Изгой']

your\_list = ['Начало', 'Выживший', 'Король Лев']

my\_list.extend(your\_list)

print(my\_list)

Вот так, всё просто.

Тут может возникнуть вопрос: чем отличается append от extend, если они делают одно и то же – добавляют элементы в список. Однако между ними есть разница. И ваша задача попробовать добавить один фильм с помощью метода extend и посмотреть, что из этого получилось. Еще один способ для соединения двух списков это использовать запись my\_list = my\_list + your\_list. Но так никто не делает, потому что скорость работы такой записи значительно ниже, чем у метода. Поэтому в реальной разработке вы не встретите объединение списков через сложение.

И рассмотрим новую задачу «Пакеты».

При работе с сервером мы кодируем сообщение и отправляем его в виде пакетов информации. Их количество равно N. Допустим, каждый пакет содержит четыре числа, каждое из которых равно нулю или единице. Эти числа называются битами. Иногда в кодировке сообщения встречаются ошибки, и в пакете эта ошибка обозначается числом -1. Если таких ошибок не больше одной, то этот пакет мы целиком добавляем в список для декодирования, а иначе отбрасываем.

Напишите программу, которая будет обрабатывать полученные пакеты и выведет на экран итоговое сообщение для декодирования, а также количество ошибок в нём и количество необработанных пакетов.

pack = []

decode = []

bad\_packs = 0

packs\_amt = int(input('Количество пакетов: '))

for i\_pack\_num in range(packs\_amt):

 print(f'\nПакет номер {i\_pack\_num + 1}')

 for i\_bit in range(4):

 print(f'{i\_bit + 1} бит: ', end=' ')

 num = int(input())

 pack.append(num)

 if pack.count(-1) <= 1:

 decode.extend(pack)

 else:

 print('Получен плохой пакет')

 bad\_packs += 1

 pack = []

print(f'\nПолученное сообщение: {decode}')

print(f'Количество ошибок в сообщении {decode.count(-1)}')

print(f'Количество потерянных сообщений: {bad\_packs}')

Запустим программу и протестируем её. Всё работает.

Отлично, вот мы с вами и познакомились с еще одним свойством count.