**Списочные выражения**

**Пример 12.** Дан словарь. Перевернуть словарь, т.е. сделать значение словаря ключами, а ключи значениями.

dict\_abc = {'a': 1, 'b': 2, 'c': 3, 'd': 3}

dict\_123 = {v: k **for** k, v **in dict\_abc**.items()}

Обратите внимание, мы потеряли "с"! Так как значения были одинаковы, то когда они стали ключами, только последнее значение сохранилось.

**Пример 13.** Дан список чисел. Создать словарь, в котором ключами будут числа из списка, а значениями квадраты этих чисел.

a = [-2, -1, 0, 1, 2, 3, 4, 5]

d = {x: x \*\* 2 **for** x **in** a}

**Пример 14.** Дан список чисел. Создать строку из этих чисел через пробел.

a = [-2, -1, 0, 1, 2, 3, 4, 5]

s = ' '.join(str(x) **for** x **in** a)

Обратите внимание: элементы коллекции для объединения в строку должны быть строками!

**Вложенные циклы**

В списочном выражении можно использовать вложенные циклы.

**Пример 15.** Получить двумерный список из m строк и n столбцов, состоящий из нулей можно так

[[0] \* m **for** i **in range**(n)]

**Пример 16.** Но при этом внутренний список также можно создать при помощи списочного выражения

[[0 **for** j **in range**(n)] **for** i **in range**(m)]

Но если число 0 заменить на некоторое выражение, зависящее от i (номер строки) и j (номер столбца), то можно получить список, заполненный по некоторой формуле.

**Пример 17.** Создать таблицу умножения от 1 до 5.

table = [[i \* j **for** j **in** **range**(1, 6)] **for** i **in** **range**(1, 6)]

**Пример 18.** Создать тсписок, заполненный 0 и 1 в шахматном порядке.

table = [[(i + j) % 2 **for** j **in** **range**(1, 6)] **for** i **in** **range**(1, 5)]

**Пример 19.** Привести список из слов к одномерному списку из букв.

words = ['Я', 'изучаю', 'Python']

letters = [letter **for** word **in** words **for letter in** word]

Хоть генераторы списков и выглядят красивыми, когда небольшие, они начинают становиться громоздкими по мере роста «мерности» и сложности.

Так как мы пишем код для людей, желательно стремиться к его максимальной читабельности. В ситуациями с большими уровнями вложенности и длинными условиями не следует увлекаться списочными выражениями, а лучше выбрать другой инструмент.

Также,списочные выражениядаже теоретически не везде можно применить. Это касается вопроса замены циклов со множеством условий. Например, если бы задача с буквами разных алфавитов (Пример 9) стояла иначе (напротив каждой буквы указать язык, к которому она принадлежит: русский, английский или греческий), списочными выражениями воспользоваться не получилось бы.