**Списочные выражения**

Для создания списков можно использовать цикл

**Пример 1.** Создадим список из десяти нулей

a = []

**for** i **in** **range**(10

 a.**append**(0)

**print**(a)

В Python есть более простой и компактный способ для создания такого списка. Можно использовать умножение списка на число

a = [0] \* 10

Для создания списков по более сложным правилам можно также использовать цикл

**Пример 2.** Создадим список кубов чисел от 0 до 9

a = []

**for** i **in** **range**(10

 a.**append**(i \*\* 3)

**print**(a)

В Python есть механизм для создания списков из неповторяющихся элементов. Такой механизм называется — **списочное выражение** (list comprehension).

Предыдущий код можно записать следующим образом:

a = [i \*\* 3 **for** i **in range**(10)]

Общий вид списочного выражения следующий:

[выражение for переменная in последовательность]

где переменная – имя некоторой переменной,

последовательность – последовательность значений, которые она принимает (список, строка или объект, полученный при помощи функции range),

выражение – некоторое выражение, как правило, зависящее от использованной в списочном выражении переменной, которым будут заполнены элементы списка.

Списковые выражения считаются более «пайтонообразными», нежели [циклы](https://smartiqa.ru/courses/python/lesson-3#forwhile) или функция **map()** для генерации списков. В других популярных языках программирования вы не встретите такой конструкции.

Их можно использовать не только в качестве «создателей» списков, но и для фильтрации данных. Зачастую (но не всегда) списочные выраженияпроще понимать и читать в чужом коде (да и в своем, когда вы вернетесь к нему через некоторое время).

Вам не обязательно создавать пустой список, цикл, чтобы сформировать новую последовательность. Все это делается в одном единственном выражении.

Рассмотрим несколько примеров использования списочных выражений

**Пример 3.** Создать список, состоящий из n нулей

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

**Пример 4.** Создать список, состоящий из квадратов чисел от 1 до n

a= [i \* i **for** i **in range**(1, n + 1)]

**Пример 5.** Создать список, заполненный случайными числами от 1 до 9

**from** math **import** randint

a = [randint(1, 9) **for** i **in range**(n)]

**Пример 6.** Создать список из строк, считанных со стандартного ввода: сначала нужно ввести число элементов списка, потом — заданное количество строк

a = [**input**() **for** i **in** **range**(**int**(**input**()))]

**Условия в списочном выражении**

В списочных выражениях можно использовать условный оператор.

**Пример 7.** Требуется создать список четных чисел от 0 до 20

evens = [i **for** i **in range**(21) **if** i % 2 == 0]

**Пример 8.** Получить список из чисел, которые делятся на 11 без остатка

a = [num **for** num **in** numbers **if** num % 11 == 0]

Если требуется не фильтрация данных по какому-то критерию, а изменение типа операции над элементами последовательности, условия могут использоваться в начале генератора списков. В отличие от предыдущего типа условий, здесь оно может дополняться вариантом **else** (но **elif** и тут невозможен).

**Пример 9.** Дается строка, в которой могут присутствовать буквы любых алфавитов. Нужно составить новый список, где напротив каждой буквы будет отмечено, является ли она английской или нет.

**from** string **import** ascii\_letters

letters = **input**()

is\_eng = [f'{letter}-ДА' **if** letter **in** ascii\_letters **else** f'{letter}-НЕТ' **for** letter **in** letters]

**Пример 10.** Из данного списка получить новый по следующим требованиям. Из данного списка нужно отобрать только четные числа, при этом отрицательные возвести в куб, а положительные в квадрат.

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

new = [x\*\*3 **if** x < 0 **else** x\*\*2 **for** x **in** list\_a **if** x % 2 == 0]

Подобные конструкции позволяют создавать не только списки, но и множества, словари

**Пример 11.** Из списка чисел создать множество модулей этих чисел.

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

my\_set= {**abs**(i) **for** i **in** list\_a}