# Урок 13.01. List comprehensions. Условия, модуль random.

Как мы уже убедились, python предоставляет множество методов для более простого, а значит, более читаемого и понятного кода. И в сообществе утвердился неологизм pythonic («пайтоник»), что отражает дзен python, или простату, краткость и логичность. И сегодня мы рассмотрим одну из таких конструкций применимой к спискам.

Допустим если нам нужно заполнить список квадратами чисел от 0 до 10, то такая программа будет выглядеть вот так:

squares = []

for x in range(10):

squares.append(x \*\* 2)

print(squares)

Но подобные задачи повсеместно встречаются, и над этим программисты много думали, чтобы запись получилась краткой и функциональной. Их рассуждения были примерно следующие:

- хочется заполнять список при его создании;

- если мой список есть какая-то математическая последовательность, то хотелось бы написать формулу как вычислять эту последовательность; и последнее

- нужно указать в каком диапазоне создается моя последовательность.

Так как python язык программирования на английском языке, то и команды у нас будут на английском, однако все команды мы уже знаем. Давайте воссоздадим такую последовательность действий.

sequence = [x \*\* 2 for x in range(10)]

print(sequence)

И вот наша программа готова. Но как же эта программа работает?

На самом деле всё что стоит у нас между квадратными скобками состоит из трех частей:

- первая часть x \*\* 2 называет *выражение* (*expresion*), эта часть отвечает за то, чем мы будем заполнять список;

- вторая часть for x in называют *переменная цикла* (*member*), эта часть отвечает за переменную в итерируемом объекте;

- и третья часть range(10) – *функция, строка, список* … (*iterable*), т.е. это любой объект, который может возвращать свои элементы по одному.

Этот способ создания список наиболее распространен среди разработчиков и называется list comprehensions.

И закрепим это задачей «Цены»

У нас есть список цен на пять товаров с точностью до копейки. Так как экономика даёт о себе знать, мы спрогнозировали, что через год придётся повышать цены на X процентов, а ещё через один год — ещё на Y процентов.

Напишите программу, которая получает на вход список цен на товары (вещественные числа, список генерируется также с помощью list comprehensions) и выводит в одну строку общую сумму стоимости товаров за каждый год.

def get\_higher\_price(percent, price):

return round(price \* (1 + percent / 100), 2)

prices\_now = [1.09, 23.56, 57.84, 4.56, 6.78]

# запросим у пользователя на сколько происходит повышение цен

first\_percent = int(input('Повышение на первый год:'))

second\_percent = int(input('Повышение на второй год:'))

# теперь сформируем новые списки

prices\_first = [get\_higher\_price(first\_percent, i\_price) for i\_price in prices\_now]

prices\_second = [get\_higher\_price(second\_percent, i\_price) for i\_price in prices\_first]

print(f'Сумма цен за каждый год: {round(sum(prices\_now), 2)}, {round(sum(prices\_first), 2)}, {round(sum(prices\_second), 2)}')

И вот наша задача. Работает.

Такой способ заполнения не является единственным.

И изменим нашу задачу с получением списка квадратов чисел: теперь нам нужно получить только квадраты нечетных чисел и при этом range трогать нельзя.

sequence\_odds = [x \*\* 2 for x in range(10) if x % 2 != 0]

print(sequence\_odds)

Вот такая короткая запись.

Если нам нужно создать список нечетных числе в квадрат, а четных в кубы, то программа будет выглядеть вот так:

sequence\_cubes = [(x \*\* 2 if x % 2 != 0 else x \*\* 3) for x in range(10)]

print(sequence\_cubes)

Таким образом, можно генерировать списки в одну строку.

И рассмотрим такую задачу «Отряды»

Мы продолжаем пробовать себя в качестве разработчика игр. Теперь нужно написать небольшую логику поведения некоторых отрядов, а также их урон. Есть два отряда, в каждом по 10 монстров. В первом отряде у каждого монстра урон абсолютно случайный и колеблется от 50 до 80, а во втором — от 30 до 60. Оба отряда вместе напали на третий, также из 10 юнитов. Юнит третьего отряда погибает, если сумма урона от двух монстров больше 100.

Нужно написать программу, которая генерирует случайные значения в первых двух списках в заданных диапазонах, а также генерирует список, состоящий из фраз «Погиб» или «Выжил». Выведите все списки на экран.

Пример:

Урон первого отряда: [77, 75, 76, 77, 76, 73, 57, 67, 76, 52]

Урон второго отряда: [53, 51, 31, 60, 49, 37, 31, 60, 37, 47]

Состояние третьего отряда: ['Погиб', 'Погиб', 'Погиб', 'Погиб', 'Погиб', 'Погиб', 'Выжил', 'Погиб', 'Погиб', 'Выжил']

Тут нам поможет новый модуль random. Он позволяет получить случайные числа.

import random

first\_squad = [random.randint(50, 80) for \_ in range(10)]

second\_squad = [random.randint(30, 60) for \_ in range(10)]

min\_len\_of\_squad = min(len(first\_squad), len(second\_squad))

third\_squad = [("Погиб" if first\_squad[i] + second\_squad[i] > 100

else "Выжил")

for i in range(min\_len\_of\_squad)]

print("Урон первого отряда:", first\_squad)

print("Урон второго отряда:", second\_squad)

print("Состояние третьего отряда:", third\_squad)

Тут мы не заморачивались с вводом чисел и сэкономили еще время. Такой способ помогает программистам отладить программу и экономить немного времени.