**Итераторы и генераторы**

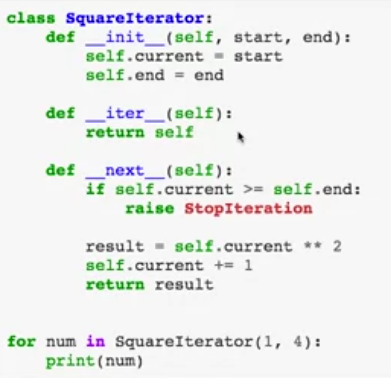
**Итератор** — это специальный объект, который позволяет поочередно перебирать элементы.

Итерируемый объект — это любой объект, от которого встроенная функция iter() может получить итератор. Для извлечения следующего элемента из итератора используется функция next(). Например:

Itr = iter(my\_list)

print(next(itr))

Чтобы создать свой итератор, нужно в классе реализовать методы \_\_next\_\_() и *\_\_iter\_\_().*



Коллекции не являются итераторами, но позволяют его создать. Большинство функций, которые работают с итераторами, умеют работать и с коллекциями.

В Python цикл for делает следующее:

1. По объекту создает итератор.
2. Получает из итератора объекты по одному и каждый раз передает полученный объект в выполняемый блок кода.

**Генераторы**

Генераторы позволяют значительно упростить работу по конструированию итераторов. Бывают выражения-генераторы и функции-генераторы.

Выражения вида

i for I in L if i>10

размещенные внутри круглых скобок, называются выражениями-генераторами. Это выражение возвращает итератор.

Особенности выражений-генераторов:

* Генератор нельзя писать без скобок — это синтаксическая ошибка
* При передаче в функцию дополнительные **скобки** необязательны
* Нельзя получить длину функцией len()
* Нельзя распечатать элементы функцией print()
* После прохождения по выражению-генератору оно остается пустым
* Выражение-генератор может быть бесконечным
* К выражению-генератору не применимы срезы!

Функция-генератор возвращает объект-генератор. Объект-генератор реализует интерфейс итератора, соответственно с этим объектом можно работать, как с любым другим итерируемым объектом.

Вместо ключевого слова return в генераторе используется yield. При каждом вызове next выполнение в функции начинается с того места где было завершено в последний раз и продолжается до следующего yield

Свой вариант range:

def cool\_range(start, stop, inc):

x = start

while x < stop:

yield x

x += inc

for n in cool\_range(1, 5, 0.5):

print(n)

print(list(cool\_range(0, 2, 0.5)))

*Задания для самостоятельной работы:*

1. Вычислите, сколько слов в заданном списке имеют длину 2, не создавая список этих слов (в одну строку).

2. Напишите функцию-генератор n чисел Фибоначчи. Выведите первые 10 чисел на экран.

3. Матрица задана как список строк, а каждая строка – список значений. Например, A = [[1,2,3,4], [11,12,13,14], [21,22,23,24]]. Вычислите транспонированную матрицу. Указание: используйте функцию zip().

4. Напишите программу, которая печатает «True», если в матрице есть нули, и «False», если их нет.