


Задание_класс_multifilter

Итераторы и генераторы

Итераторы и генераторы

 <https://stepik.org/lesson/24464/step/4?auth=login&unit=6769>



Условие

Одним из самых часто используемых классов в Python является класс **filter**. Он принимает в конструкторе два аргумента **a** и **f** – последовательность и функцию, и позволяет проитерироваться только по таким элементам **x** из последовательности **a**, что **f(x)** равно **True**. Будем говорить, что в этом случае функция **f** допускает элемент **x**, а элемент **x** является допущенным.

В данной задаче мы просим вас реализовать класс **multifilter**, который будет выполнять ту же функцию, что и стандартный класс **filter**, но будет использовать не одну функцию, а несколько.

Решение о допуске элемента будет приниматься на основании того, сколько функций допускают этот элемент, и сколько не допускают. Обозначим эти количества за **pos** и **neg**.

Введем понятие *решающей функции* – это функция, которая принимает два аргумента – количества **pos** и **neg**, и возвращает **True**, если элемент допущен, и **False** иначе.

Рассмотрим процесс допуска подробнее на следующем примере.

a = [1, 2, 3] **f2(x) = x % 2 == 0** # возвращает **True**, если **x** делится на 2

f3(x) = x % 3 == 0

judge_any(pos, neg) = pos >= 1 # возвращает **True**, если хотя бы одна функция допускает элемент

В этом примере мы хотим отфильтровать последовательность **a** и оставить только те элементы, которые делятся на два или на три.

Функция **f2** допускает только элементы, делящиеся на два, а функция **f3** допускает только элементы, делящиеся на три. Решающая

функция допускает элемент в случае, если он был допущен хотя бы одной из функций **f2** или **f3**, то есть элементы, которые делятся на два или на три.

Возьмем первый элемент **x = 1**.

f2(x) равно **False**, т. е. функция **f2** не допускает элемент **x**.

f3(x) также равно **False**, т. е. функция **f3** также не допускает элемент **x**.

В этом случае **pos = 0**, так как ни одна функция не допускает **x**, и соответственно **neg = 2**.

judge_any(0, 2) равно **False**, значит мы не допускаем элемент **x = 1**.

Возьмем второй элемент **x = 2**.

f2(x) равно **True**

f3(x) равно **False**

pos = 1, neg = 1

judge_any(1, 1) равно **True**, значит мы допускаем элемент **x = 2**.

Аналогично для третьего элемента **x = 3**.

Таким образом, получили последовательность допущенных элементов **[2, 3]**.

Класс должен обладать следующей структурой:

```
class multifilter:
    def judge_half(pos, neg):
        # допускает элемент, если его допускает хотя бы половина функций (pos >= neg)

    def judge_any(pos, neg):
        # допускает элемент, если его допускает хотя бы одна функция (pos >= 1)

    def judge_all(pos, neg):
        # допускает элемент, если его допускают все функции (neg == 0)

    def __init__(self, iterable, *funcs, judge=judge_any):
        # iterable - исходная последовательность
        # funcs - допускающие функции
        # judge - решающая функция

    def __iter__(self):
        # возвращает итератор по результирующей последовательности
```

Пример использования

```
def mul2(x):
    return x % 2 == 0

def mul3(x):
    return x % 3 == 0

def mul5(x):
    return x % 5 == 0

a = [i for i in range(31)] # [0, 1, 2, ... , 30]

print(list(multifilter(a, mul2, mul3, mul5)))
# [0, 2, 3, 4, 5, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25, 26, 27, 28, 30]

print(list(multifilter(a, mul2, mul3, mul5, judge=multifilter.judge_half)))
# [0, 6, 10, 12, 15, 18, 20, 24, 30]

print(list(multifilter(a, mul2, mul3, mul5, judge=multifilter.judge_all)))
# [0, 30]
```

Мое решение