Урок 5. Генераторы и comprehensions. Множества

1. Написать генератор нечётных чисел от 1 до n (включительно), используя ключевое слово yield, например:

```
>>> odd_to_15 = odd_nums(15)
>>> next(odd_to_15)
1
>>> next(odd_to_15)
3
...
>>> next(odd_to_15)
15
>>> next(odd_to_15)
...StopIteration...
```

2. * (вместо 1) Решить задачу генерации нечётных чисел от 1 до n (включительно), не используя ключевое слово yield.

3. Есть два списка:

```
tutors = [
    'Иван', 'Анастасия', 'Петр', 'Сергей',
    'Дмитрий', 'Борис', 'Елена'
]
klasses = [
    '9А', '7В', '9Б', '9В', '8Б', '10А', '10Б', '9А'
]
```

Необходимо реализовать генератор, возвращающий кортежи вида `(<tutor>, <klass>)`, например:

```
('Иван', '9А')
('Анастасия', '7В')
...
```

Количество генерируемых кортежей не должно быть больше длины списка tutors. Если в списке klasses меньше элементов, чем в списке tutors, необходимо вывести последние кортежи в виде: `(<tutor>, None)`, например:

```
('Станислав', None)
```

```
### Доказать, что вы создали именно генератор. Проверить его работу вплоть до
истощения. Подумать, в каких ситуациях генератор даст эффект.

### 4. Представлен список чисел. Необходимо вывести те его элементы, значения которых
больше предыдущего, например:
```

src = [300, 2, 12, 44, 1, 1, 4, 10, 7, 1, 78, 123, 55]
result = [12, 44, 4, 10, 78, 123]
```
```

Подсказка: использовать возможности python, изученные на уроке.

5. Подумайте, как можно сделать оптимизацию кода по памяти, по скорости.

Представлен список чисел. Определить элементы списка, не имеющие повторений. Сформировать из этих элементов список с сохранением порядка их следования в исходном списке, например:

```
src = [2, 2, 2, 7, 23, 1, 44, 44, 3, 2, 10, 7, 4, 11]
result = [23, 1, 3, 10, 4, 11]
```

Подсказка: напишите сначала решение «в лоб». Потом подумайте об оптимизации.

Задачи со * предназначены для продвинутых учеников, которым мало сделать обычное задание.

** Дополнительные материалы

1. Лутц Марк. Изучаем Python.
2. Модуль datetime.