Генераторы.

Рекурсивное определение

```
def ones():
    yield 1
    yield from ones()
```

Как получить последовательность из двоек и троек?

Рекурсивное определение

```
def ones():
    yield 1
    yield from ones()

# например
def add_streams(st1, st2):
    yield from map(lambda x, y: x + y, st1, st2)

def take(itr, n):
    for i in range(n):
        yield next(itr)
```

Рекурсивное определение

```
print(list(take(add_streams(ones(), ones()), 10)))
# [2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

print(list(take(add_streams(
                    add_streams(
                        ones(),
                        ones()
                    ),
                    ones()),
                    10
                )))
# [3, 3, 3, 3, 3, 3, 3, 3, 3, 3]
```

Целые числа и числа Фибоначчи

```
def ints():
    yield 1
    yield from add_streams(ones(), ints())

print(take(ints(), 10))
#?

def fib():
    yield 1
    yield 1

    shifted_fib = fib()
    next(shifted_fib)
    yield from add_streams(fib(), shifted_fib)

print(take(fib(), 10))
#?
```

Проблема

```
print(list(take(ones(), 10000)))
# ----------------------------------------
# RecursionError   Traceback (most recent call last)
#       2     yield 1
# ----> 3     yield from ones()
#       4
#
# RecursionError: maximum recursion depth exceeded
```

Проблема

```
print(list(take(ones(), 10000)))
# ----------------------------------------
# RecursionError   Traceback (most recent call last)
#       2     yield 1
# ----> 3     yield from ones()
#       4
#
# RecursionError: maximum recursion depth exceeded
```

Генератор держит контекст — как минимум стекфрейм для исполнения генераторной функции. `yield from <gen>` указывает, что текущий генератор можно остановить и исполнять `<gen>`, но никто не удаляет текущий стекфрейм и количество созданных фреймов превышает лимиты.

Решение

```
def ones():
    yield 1
    yield ones()

def trampoline(it):
    while True:
        val = next(it, None)
        if val is None:
            return
        yield val
        it = next(it) # забываем ссылку на старый
        # генератор, давая возможность gc подобрать его
        # стекфрейм
```

Решение

```
print(
    len(
        list(
            take(
                trampoline(ones()),
                10_000_000
                )
            )
        )
    )
# 10000001
```

add_streams

```
# flat, flat -> argument of trampoline
def add_streams(st1, st2):
    v1 = next(st1)
    v2 = next(st2)
    yield v1 + v2
    yield add_streams(st1, st2)

def ints():
    yield 1
    yield add_streams(trampoline(ints()), trampoline(ones()))
```

add_streams

```
print(
    len(
        list(
            take(
                trampoline(ints()),
                10_000_000
                )
            )
        )
    )
# Traceback (most recent call last):
#   ....
#     v1 = next(it1)
#
# RecursionError: maximum recursion depth exceeded while
calling a Python object
```

ints

```
def ints(n=1):
    yield n
    yield ints(n + 1)

print(
    len(
        list(
            take(
                trampoline(ints()),
                10_000_000
                )
            )
        )
    )
# 10000001
```