Сценарий измерения времени

Чтобы измерить скорость итерационных инструментов (наша исходная цель), мы запустим следующий сценарий – в нем применяется модуль `timer`, который был написан для измерения относительных скоростей изученных ранее методик построения списков:

```
# Файл timeseqs.py
"Проверка относительной скорости итерационных альтернатив."

import sys, timer                          # Импортирование функций timer
reps = 10000
repslist = list(range(reps))               # Вынесение наружу, список
                                           #   в Python 2.X/3.X

def forLoop():
    res = []
    for x in repslist:
        res.append(abs(x))
    return res

def listComp():
    return [abs(x) for x in repslist]

def mapCall():
    return list(map(abs, repslist))        # Использовать list() только
                                           #   в Python 3.X!
  # return map(abs, repslist)

def genExpr():
    return list(abs(x) for x in repslist)  # list() требуется для инициирования
                                           #   выпуска результатов

def genFunc():
    def gen():
        for x in repslist:
            yield abs(x)
    return list(gen())   # list() требуется для инициирования выпуска результатов

print(sys.version)
for test in (forLoop, listComp, mapCall, genExpr, genFunc):
    (bestof, (total, result)) = timer.bestoftotal(5, 1000, test)
    print ('%-9s: %.5f => [%s...%s]' %
           (test.__name__, bestof, result[0], result[-1]))
```

В сценарии проверяются пять альтернативных способов создания списков результатов. Как видно, сообщаемые им показания времени отражают порядка 10 миллионов шагов для каждой из пяти тестируемых функций – каждая строит список из 10 000 элементов 1000 раз. Процесс повторяется 5 раз, чтобы получить лучшее время для каждой из 5 тестируемых функций, суммарно давая колоссальное количество в 250 миллионов шагов сценария (впечатляюще, но вполне реально на большинстве компьютеров в наши дни).

Обратите внимание на то, что мы должны прогнать результаты генераторного выражения и функции через вызов встроенного метода `list`, заставляя их выдавать все свои значения. Если этого не сделать, тогда в Python 2.X и 3.X мы произвели бы просто генераторы, не делающие какую-либо реальную работу. В Python 3.X мы обязаны делать то же самое для результата `map`, т.к. теперь он также является итерируемым объектом. Для Python 2.X вызов `list` вокруг `map` потребуется удалить вручную, чтобы