не поддерживали понятие дополнения результатов, когда длины последовательностей отличаются, как делает версия map в Python 2.X с аргументом None:

```
C:code> c:\python27\python
>>> map(None, [1, 2, 3], [2, 3, 4, 5])
[(1, 2), (2, 3), (3, 4), (None, 5)]
>>> map(None, 'abc', 'xyz123')
[('a', 'x'), ('b', 'y'), ('c', 'z'), (None, '1'), (None, '2'), (None, '3')]
```

Используя итерационные инструменты, мы можем написать код аналогов, которые эмулируют и усечение zip, и дополнение map из Python 2.X — как оказалось, их код почти одинаков:

```
# Аналоги zip(seqs...) и map(None, seqs...) из Python 2.X
def myzip(*seqs):
    seqs = [list(S) for S in seqs]
    res = []
    while all(seqs):
        res.append(tuple(S.pop(0) for S in seqs))
    return res

def mymapPad(*seqs, pad=None):
    seqs = [list(S) for S in seqs]
    res = []
    while any(seqs):
        res.append(tuple((S.pop(0) if S else pad) for S in seqs))
    return res

S1, S2 = 'abc', 'xyz123'
print(myzip(S1, S2))
print(mymapPad(S1, S2))
print(mymapPad(S1, S2, pad=99))
```

Обе написанные функции работают с *итерируемым* объектом любого типа, поскольку они прогоняют свои аргументы через встроенную функцию list, чтобы инициировать генерацию результатов (например, вдобавок к последовательностям наподобие строк в качестве аргументов подошли бы файлы). Обратите внимание на применение встроенных функций all и any — они возвращают True, если соответственно все или любые элементы в итерируемом объекте являются True (или непустыми, что эквивалентно). Указанные встроенные функции используются для останова цикла, когда любой или все перечисленные аргументы становятся пустыми после удалений.

Также следует отметить применение аргумента pad *с передачей только по ключевому слову* из Python 3.X. В отличие от функции map в Python 2.X наша версия позволит указывать любой дополняющий объект (если вы используете Python 2.X, тогда взамен для поддержки такой возможности применяйте форму ****ключевые_аргументы*; за деталями обращайтесь в главу 18). После запуска этих функций выводятся следующие результаты вызова zip и двух вызовов дополняющих map:

```
[('a', 'x'), ('b', 'y'), ('c', 'z')]
[('a', 'x'), ('b', 'y'), ('c', 'z'), (None, '1'), (None, '2'), (None, '3')]
[('a', 'x'), ('b', 'y'), ('c', 'z'), (99, '1'), (99, '2'), (99, '3')]
```

Такие функции не поддаются переводу в списковые включения из-за того, что их циклы слишком специфичны. Тем не менее, хотя аналоги zip и map в текущий момент строят и возвращают результирующие списки, столь же легко посредством yield превратить их в *генераторы*, так что каждая функция станет возвращать по одной порции своего результирующего набора за раз. Результаты будут такими же, как раньше, но