Но в Python 3.X он попадает в бесконечный цикл и терпит неудачу, потому что функция `map` из Python 3.X возвращает одноразовый итерируемый объект, а не список, как `map` из Python 2.X. В Python 3.X как только мы однократно выполним списковое включение внутри цикла, `iters` опустошается, но постоянно будет `True` (а `res` — `[]`). Чтобы код заработал в Python 3.X, необходимо использовать встроенную функцию `list` для создания объекта, который способен поддерживать множество итераций:

```
def myzip(*args):
    iters = list(map(iter, args))    # Допускает множество просмотров
    ...остальной код не меняется...
```

Запустите код самостоятельно и отследите его работу. Урок здесь в том, что помещение вызовов `map` внутрь `list` в Python 3.X предназначено не только для отображения!

Сводка по синтаксису включений

Внимание в главе было сосредоточено на списковых включениях и генераторах, но имейте в виду, что в Python 3.X и 2.7 доступны еще две формы выражений с включениями: включения множеств и словарей. Мы кратко упоминали о них в главах 5 и 8, но благодаря вновь обретенным знаниям включений и генераторов вы уже должны быть в состоянии понять суть таких расширений.

- Для множеств новая литеральная форма `{1, 3, 2}` эквивалентна `set([1, 3, 2])`, а новый синтаксис включений множеств `{f(x) for x in S if P(x)}` подобен генераторному выражению `set(f(x) for x in S if P(x))`, где `f(x)` — произвольное выражение.
- Для словарей новый синтаксис включений словарей `{key: val for (key, val) in zip(keys, vals)}` работает подобно форме `dict(zip(keys, vals))`, а `{x: f(x) for x in items}` похож на генераторное выражение `dict((x, f(x)) for x in items)`.

Ниже представлена сводка по всем альтернативным версиям включений в Python 3.X и 2.7. Последние две версии недоступны в Python 2.6 и более ранних выпусках:

```
>>> [x * x for x in range(10)]          # Списковое включение: строит список
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]    # Подобно list(генераторное выражение)

>>> (x * x for x in range(10))          # Генераторное выражение:
                                        #   производит элементы
<generator object at 0x009E7328>        # Круглые скобки часто необязательны

>>> {x * x for x in range(10)}          # Включение множества, Python 3.X и 2.7
{0, 1, 4, 81, 64, 9, 16, 49, 25, 36}    # {x, y} - тоже множество в этих версиях

>>> {x: x * x for x in range(10)}       # Включение словаря, Python 3.X и 2.7
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

Области видимости и переменные включений

Теперь, когда были показаны все формы включений, вспомним приведенный в главе 17 обзор локализации переменных циклов в этих выражениях. *Python 3.X* локализует переменные циклов во всех четырех формах — временные имена переменных циклов в генераторах и включениях множеств, словарей и списков являются локальными в отношении выражения. Они не конфликтуют с именами вне выражения, но там не доступны и работают не так, как оператор цикла `for`: