(continued from previous page)

```
else:
    from importlib2 import abc
```

The best solution, though, is to do no version detection at all and instead rely on feature detection. That avoids any potential issues of getting the version detection wrong and helps keep you future-compatible:

```
try:
    from importlib import abc
except ImportError:
    from importlib2 import abc
```

2.6 Prevent compatibility regressions

Once you have fully translated your code to be compatible with Python 3, you will want to make sure your code doesn't regress and stop working under Python 3. This is especially true if you have a dependency which is blocking you from actually running under Python 3 at the moment.

To help with staying compatible, any new modules you create should have at least the following block of code at the top of it:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
```

You can also run Python 2 with the `-3` flag to be warned about various compatibility issues your code triggers during execution. If you turn warnings into errors with `-Werror` then you can make sure that you don't accidentally miss a warning.

You can also use the Pylint project and its `--py3k` flag to lint your code to receive warnings when your code begins to deviate from Python 3 compatibility. This also prevents you from having to run Modernize or Futurize over your code regularly to catch compatibility regressions. This does require you only support Python 2.7 and Python 3.4 or newer as that is Pylint's minimum Python version support.

2.7 Check which dependencies block your transition

After you have made your code compatible with Python 3 you should begin to care about whether your dependencies have also been ported. The caniusepython3 project was created to help you determine which projects – directly or indirectly – are blocking you from supporting Python 3. There is both a command-line tool as well as a web interface at https://caniusepython3.com.

The project also provides code which you can integrate into your test suite so that you will have a failing test when you no longer have dependencies blocking you from using Python 3. This allows you to avoid having to manually check your dependencies and to be notified quickly when you can start running on Python 3.

2.8 Update your `setup.py` file to denote Python 3 compatibility

Once your code works under Python 3, you should update the classifiers in your `setup.py` to contain `Programming Language :: Python :: 3` and to not specify sole Python 2 support. This will tell anyone using your code that you support Python 2 **and** 3. Ideally you will also want to add classifiers for each major/minor version of Python you now support.