Running 2to3

We're going to migrate the `chardet` module from Python 2 to Python 3. Python 3 comes with a utility script called `2to3`, which takes your actual Python 2 source code as input and auto-converts as much as it can to Python 3. In some cases this is easy — a function was renamed or moved to a different module — but in other cases it can get pretty complex. To get a sense of all that it can do, refer to the appendix, Porting code to Python 3 with `2to3`. In this chapter, we'll start by running `2to3` on the `chardet` package, but as you'll see, there will still be a lot of work to do after the automated tools have performed their magic.

The main `chardet` package is split across several different files, all in the same directory. The `2to3` script makes it easy to convert multiple files at once: just pass a directory as a command line argument, and `2to3` will convert each of the files in turn.

```
C:\home\chardet> python c:\Python30\Tools\Scripts\2to3.py -w chardet\
RefactoringTool: Skipping implicit fixer: buffer
RefactoringTool: Skipping implicit fixer: idioms
RefactoringTool: Skipping implicit fixer: set_literal
RefactoringTool: Skipping implicit fixer: ws_comma
--- chardet\__init__.py (original)
+++ chardet\__init__.py (refactored)
@@ -18,7 +18,7 @@
 __version__ = "1.0.1"

 def detect(aBuf):
-    import universaldetector
+    from . import universaldetector
     u = universaldetector.UniversalDetector()
     u.reset()
     u.feed(aBuf)
--- chardet\big5prober.py (original)
+++ chardet\big5prober.py (refactored)
@@ -25,10 +25,10 @@
 # 02110-1301  USA
 ######################### END LICENSE BLOCK #########################

-from mbcharsetprober import MultiByteCharSetProber
-from codingstatemachine import CodingStateMachine
+from .mbcharsetprober import MultiByteCharSetProber
+from .codingstatemachine import CodingStateMachine
 from chardistribution import Big5DistributionAnalysis
```

```
-from chardistribution import Big5DistributionAnalysis
-from mbcssm import Big5SMModel
+from .mbcharsetprober import MultiByteCharSetProber

+from .codingstatemachine import CodingStateMachine
+from .chardistribution import Big5DistributionAnalysis
+from .mbcssm import Big5SMModel

 class Big5Prober(MultiByteCharSetProber):
     def __init__(self):
--- chardet\chardistribution.py (original)
+++ chardet\chardistribution.py (refactored)
@@ -25,12 +25,12 @@
 # 02110-1301  USA
 ######################### END LICENSE BLOCK #########################

-import constants
-from euctwfreq import EUCTWCharToFreqOrder, EUCTW_TABLE_SIZE, EUCTW_TYPICAL_DISTRIBUTION_RAT
-from euckrfreq import EUCKRCharToFreqOrder, EUCKR_TABLE_SIZE, EUCKR_TYPICAL_DISTRIBUTION_RAT
-from gb2312freq import GB2312CharToFreqOrder, GB2312_TABLE_SIZE, GB2312_TYPICAL_DISTRIBUTION
-from big5freq import Big5CharToFreqOrder, BIG5_TABLE_SIZE, BIG5_TYPICAL_DISTRIBUTION_RATIO
-from jisfreq import JISCharToFreqOrder, JIS_TABLE_SIZE, JIS_TYPICAL_DISTRIBUTION_RATIO
+from . import constants
+from .euctwfreq import EUCTWCharToFreqOrder, EUCTW_TABLE_SIZE, EUCTW_TYPICAL_DISTRIBUTION_RA
+from .euckrfreq import EUCKRCharToFreqOrder, EUCKR_TABLE_SIZE, EUCKR_TYPICAL_DISTRIBUTION_RA
+from .gb2312freq import GB2312CharToFreqOrder, GB2312_TABLE_SIZE, GB2312_TYPICAL_DISTRIBUTIO
+from .big5freq import Big5CharToFreqOrder, BIG5_TABLE_SIZE, BIG5_TYPICAL_DISTRIBUTION_RATIO
+from .jisfreq import JISCharToFreqOrder, JIS_TABLE_SIZE, JIS_TYPICAL_DISTRIBUTION_RATIO

 ENOUGH_DATA_THRESHOLD = 1024
 SURE_YES = 0.99
.
.
. (it goes on like this for a while)
.
.
RefactoringTool: Files that were modified:
RefactoringTool: chardet\__init__.py
RefactoringTool: chardet\big5prober.py
RefactoringTool: chardet\chardistribution.py
RefactoringTool: chardet\charsetgroupprober.py
RefactoringTool: chardet\codingstatemachine.py
RefactoringTool: chardet\constants.py
RefactoringTool: chardet\escprober.py
RefactoringTool: chardet\escsm.py
RefactoringTool: chardet\eucjpprober.py
RefactoringTool: chardet\euckrprober.py
RefactoringTool: chardet\euctwprober.py
RefactoringTool: chardet\gb2312prober.py
RefactoringTool: chardet\hebrewprober.py
RefactoringTool: chardet\jpcntx.py
RefactoringTool: chardet\langbulgarianmodel.py
RefactoringTool: chardet\langcyrillicmodel.py
RefactoringTool: chardet\langgreekmodel.py
RefactoringTool: chardet\langhebrewmodel.py
RefactoringTool: chardet\langhungarianmodel.py
RefactoringTool: chardet\langthaimodel.py
RefactoringTool: chardet\latin1prober.py
RefactoringTool: chardet\mbcharsetprober.py
RefactoringTool: chardet\mbcsgroupprober.py
RefactoringTool: chardet\mbcssm.py
RefactoringTool: chardet\sbcharsetprober.py
RefactoringTool: chardet\sbcsgroupprober.py
```

```
RefactoringTool: chardet\sjisprober.py
RefactoringTool: chardet\universaldetector.py
RefactoringTool: chardet\utf8prober.py
```

Now run the `2to3` script on the testing harness, `test.py`.

```
C:\home\chardet> python c:\Python30\Tools\Scripts\2to3.py -w test.py
RefactoringTool: Skipping implicit fixer: buffer
RefactoringTool: Skipping implicit fixer: idioms
RefactoringTool: Skipping implicit fixer: set_literal
RefactoringTool: Skipping implicit fixer: ws_comma
--- test.py (original)
+++ test.py (refactored)
@@ -4,7 +4,7 @@
 count = 0
 u = UniversalDetector()
 for f in glob.glob(sys.argv[1]):
-    print f.ljust(60),
+    print(f.ljust(60), end=' ')
     u.reset()
     for line in file(f, 'rb'):
         u.feed(line)
@@ -12,8 +12,8 @@
     u.close()
     result = u.result
     if result['encoding']:
-        print result['encoding'], 'with confidence', result['confidence']
+        print(result['encoding'], 'with confidence', result['confidence'])
     else:
-        print '******** no result'
+        print('******** no result')
     count += 1
-print count, 'tests'
+print(count, 'tests')
RefactoringTool: Files that were modified:
RefactoringTool: test.py
```

Well, that wasn't so hard. Just a few imports and print statements to convert. Speaking of which, what *was* the problem with all those import statements? To answer that, you need to understand how the `chardet` module is split into multiple files.