Rahul Rajagopalan
6.170

Assignment 2 Testing Strategy

For most of the assignment, I tried to use test-driven-development. That is, I wrote tests first, ran them (so they failed), then wrote code that would make the tests pass, repeating this loop until the component of code in question was complete. I was generally able to do this for the parts of the code that were easy to test – mostly pure functions.

I generally attempted to test any code whose implementation was likely to have bugs – likely enough that the benefit of writing tests outweighted its drawbacks. This usually involved string processing or reading metadata (since this is my first time using IPTCInfo and I wanted to verify that I was doing it right). Extremely simple code, such as getter functions, don't have tests.

Testing code that used external libraries was a bit more complicated. For example, consider the code in JpegPicture that tries to read metadata from a JPEG image. Using an actual JPEG for the test has some flaws; it's slower and harder to set up. For these components, I replaced the actual IPTCInfo object with a stub object that returned data in the same format as an actual object. To do this, I used the dependency injection pattern, passing in the IPTCInfo object to the JpegPicture instead of instantiating it in the constructor. In this way, most code was testable.

The primary exception to this was the presentation of the final output. It's not really possible for the computer to know what looks good to a human. Therefore the outputted HTML files were manually inspected in a browser.

All the test code is in the tests/ directory under photogallery/. Most tests do not hit the filesystem or otherwise have side effects. A few tests, the ones under photogallery/tests/integration do in face perfomr IO (in some cases this was unavoidable, such as for copier.py). These tests may run slower.

The method of running the tests is a bit different from just invoking the script. This is because they are in a package and use intra-package imports. You can invoke them by cd'ing to src/ and then invoking

```
python -m photogallery.tests.test_name_without_py_extension
```

to run the tests in the corresponding module. More details about this are in the README file in the tests/ directory.