Milestone 3: Coverage

2019-05-06 22:55
1a0f14a

For this milestone, you will write unit tests for a real, actively-used Python module called `humanize`[1].

Code to test

The code is in `cs164pub/coverage/humanize`[2], and you want to write your tests into the file `coverage/tests.py`[3]. Here is a starting point for that file:

```
import unittest
import humanize

class CommaTests(unittest.TestCase):
    def test_zeroes(self):
        self.assertEqual(humanize.intcomma(1048576), "1,048,576")

    # Returns non-numbers unchanged
    def test_non_number(self):
        self.assertEqual(humanize.intcomma("hello"), "hello")

if __name__ == "__main__":
    unittest.main()
```

Those tests are both testing the `humanize.intcomma` function, but there are many other functions in the module. Use the `README` on GitHub as a guide to how they should work.

[1] github.com/jmoiron/humanize

[2] gitlab.liu.edu/cs164s19/cs164pub/tree/master/coverage

[3] gitlab.liu.edu/cs164s19/cs164pub/blob/master/coverage/tests.py

Coverage tool

As we covered in class, here is a session that shows how to run the coverage tool:

```
$ cd ~/Desktop/cs164/coverage
$ coverage erase
$ coverage run tests.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
$ coverage report
Name                    Stmts   Miss  Cover
-----------------------------------------
```

```
humanize/__init__.py        6      0   100%
humanize/compat.py          4      1    75%
humanize/filesize.py       20     17    15%
humanize/i18n.py           32     17    47%
humanize/number.py         62     39    37%
humanize/time.py          114    102    11%
tests.py                    9      0   100%
----------------------------------------
TOTAL                     247    176    29%
```

You want to get that 29% number **as high as possible.** (It may be pretty difficult to get all the way to 100%, but try.)

To view the full report, run:

```
$ coverage html
```

And then open the file `coverage/htmlcov/index.html` in your browser. Click the file names in that report page to find the blocks that are highlighted in pink because they are *not executed*.

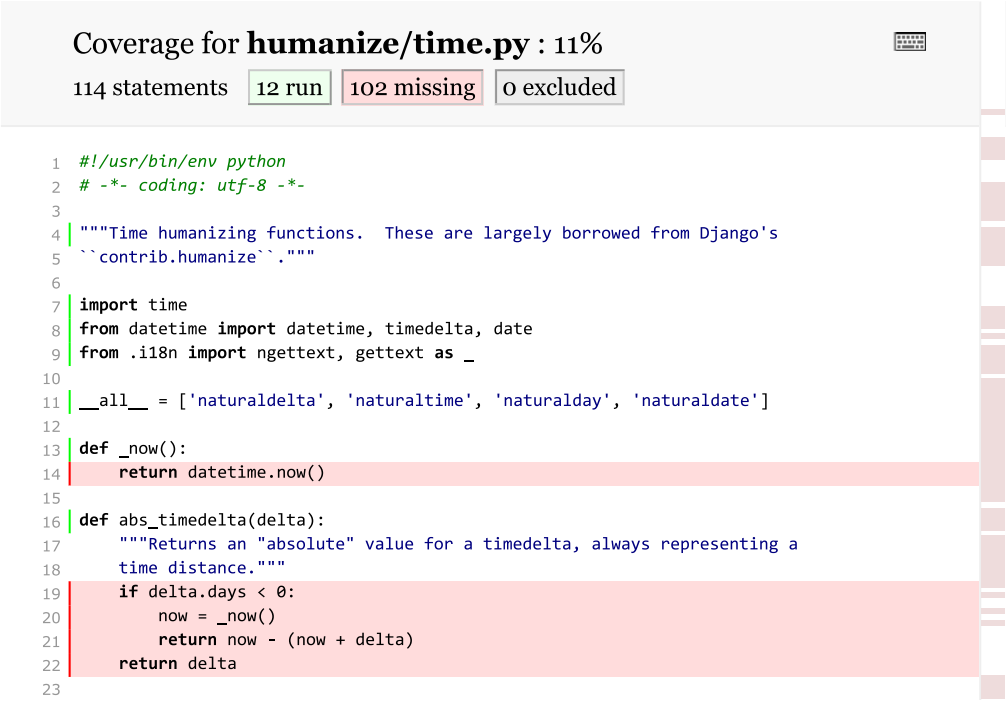

Figure 1: Sample coverage report for a particular Python source file

My results

In my solution, I got the coverage up to 98%:

```
Name                    Stmts   Miss  Cover
```

```
-----------------------------------------
humanize/__init__.py       6      0   100%
humanize/compat.py         4      1    75%
humanize/filesize.py      20      0   100%
humanize/i18n.py          32      0   100%
humanize/number.py        62      0   100%
humanize/time.py         114      5    96%
tests.py                 124      0   100%
-----------------------------------------
TOTAL                    362      6    98%
```