Software Engineering: Testing Tools

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - Tools related to testing
- Specifically:
  - Tools for test automation
  - Tools for statement/coverage testing

Motivation

- Motivation:
  - Testing is important
  - Testing will become more important???

Agenda

- **Tools for test automation**
- Tools for statement/coverage testing

Test Automation

Tools for test automation:

Language	Test Automation Tool
Python	***unittest*** (formerly ***PyUnit***)* ***PyTest***
Java	***JUnit***
C	***CUnit***
JavaScript (browser)	***Playwright**** ***Selenium***
JavaScript (Node.js)	***Jest****

* Described in the following slides

** See me if you want an example

Test Automation

- Python **module** testing
  - See **Fraction/**

```
$ cd Fraction
$ python fractionclient.py
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
frac1: 1/2
frac2: 3/4
frac1 does not equal frac2
frac1 is less than frac2
frac1 is less than or equal to frac2
-frac1: -1/2
frac1 + frac2: 5/4
frac1 - frac2: -1/4
frac1 * frac2: 3/8
frac1 / frac2: ⅔
$
```

Test Automation

- Python **module** testing (cont.)
  - See **Fraction/**
    - **euclid.py**
    - **fraction.py**
    - **fractionclient.py**

Test Automation

- Goal:
  - Test fraction.py
- Problem:
  - fractionclient.py is insufficient
- Solution…

Test Automation

- Python **module** testing (cont.)
  - See **Fraction/**
    - euclid.py
    - fraction.py
    - fractionclient.py
    - **testfraction.py**

Test Automation

```
$ cd Fraction
$ python testfraction.py
test_add (testfraction.TestFraction.test_add) ... ok
test_constructor (testfraction.TestFraction.test_constructor) ... ok
test_eq (testfraction.TestFraction.test_eq) ... ok
test_ge (testfraction.TestFraction.test_ge) ... ok
test_gt (testfraction.TestFraction.test_gt) ... ok
test_le (testfraction.TestFraction.test_le) ... ok
test_lt (testfraction.TestFraction.test_lt) ... ok
test_mul (testfraction.TestFraction.test_mul) ... ok
test_ne (testfraction.TestFraction.test_ne) ... ok
test_neg (testfraction.TestFraction.test_neg) ... ok
test_str (testfraction.TestFraction.test_str) ... ok
test_sub (testfraction.TestFraction.test_sub) ... ok
test_truediv (testfraction.TestFraction.test_truediv) ... ok

----------------------------------------------------------------------
Ran 13 tests in 0.002s

OK
$
```

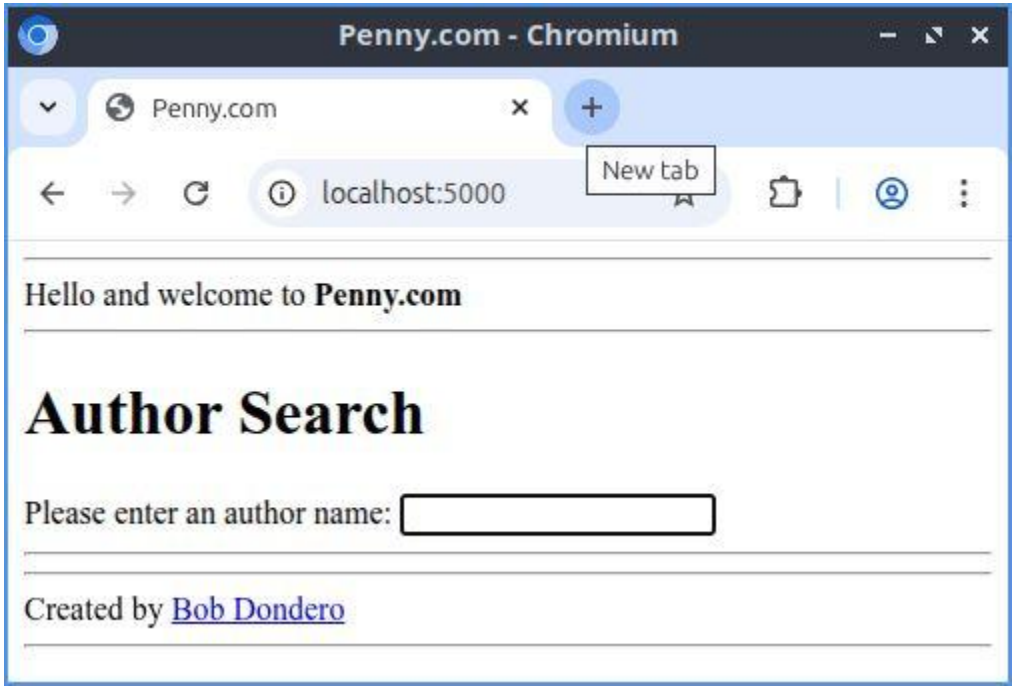
Test Automation

- Python **web app** testing
  - See **PennyTest** app

```
$ python runserver.py
 * Serving Flask app 'penny'
 * Debug mode: on
WARNING: This is a development server. Do not
use it in a production deployment. Use a
production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.29:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 457-747-552
...
```

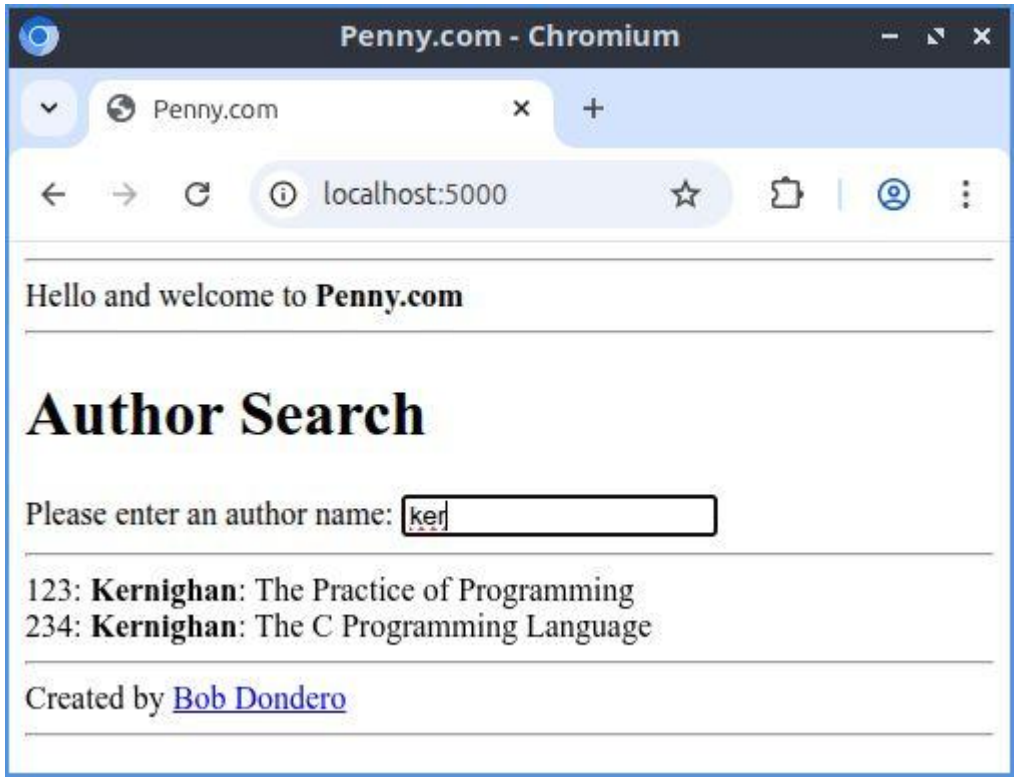
Test Automation

- Python **web app** testing
  - See **PennyTest** app (cont.)



Test Automation

- Python **web app** testing
  - See **PennyTest** app (cont)



Test Automation

- Python **web app** testing
  - See **PennyTest** app
    - **runserver.py**
    - **penny.sql**, penny.sqlite
    - **database.py**
    - **penny.py**
    - **index.html**

Test Automation

- Python **web app** testing
  - See **PennyTest** app
    - runserver.py
    - penny.sql, penny.sqlite
    - database.py
    - penny.py
    - index.html
    - **testdatabase.py**

Test Automation

```
$ cd PennyTest
$ python testdatabase.py
test_abc (testdatabase.TestDatabase.test_abc) ... ok
test_ern (testdatabase.TestDatabase.test_ern) ... ok
test_ker (testdatabase.TestDatabase.test_ker) ... ok
test_sed (testdatabase.TestDatabase.test_sed) ... ok

----------------------------------------------------------------------
-
Ran 4 tests in 0.002s

OK
$
```

Test Automation

- Python **web app** testing
  - See **PennyTest** app
    - runserver.py
    - penny.sql, penny.sqlite
    - database.py
    - penny.py
    - index.html
    - testdatabase.py
    - **testpenny.py**
      - Like assignments: uses Playwright
      - Unlike assignments: uses unittest, checks results

Test Automation

```
$ cd PennyTest
$ python runserver.py
* Serving Flask app 'penny'
 * Debug mode: on
WARNING: This is a development server. Do not use
it in a production deployment. Use a production
```

```
$ cd PennyTest
$ python testpenny.py
test_abc (testpenny.PennyTestCase.test_abc) ... ok
test_ker (testpenny.PennyTestCase.test_ker) ... ok
test_ker_bad_spaces (testpenny.PennyTestCase.test_ker_bad_spaces) ... ok
test_ker_spaces (testpenny.PennyTestCase.test_ker_spaces) ... ok
test_sed (testpenny.PennyTestCase.test_sed) ... ok
test_spaces (testpenny.PennyTestCase.test_spaces) ... ok

----------------------------------------------------------------------
Ran 6 tests in 13.266s

OK
$
```

Test Automation

- **Problem**:
  - What if the web app does authentication?
    - Via CAS, EntraID, Google, …
  - How can your testing code use app endpoints that are protected?

Test Automation

Solution: The general idea:

authstub.py

```
…
# Implement an "authentication" mechanism
# that simply asks the user for a
# username.
…
```

penny.py

```
…
TESTING = os.getenv('TESTING', 'no')
if TESTING == 'no':
    import auth
else:
    import authstub as auth
…
```

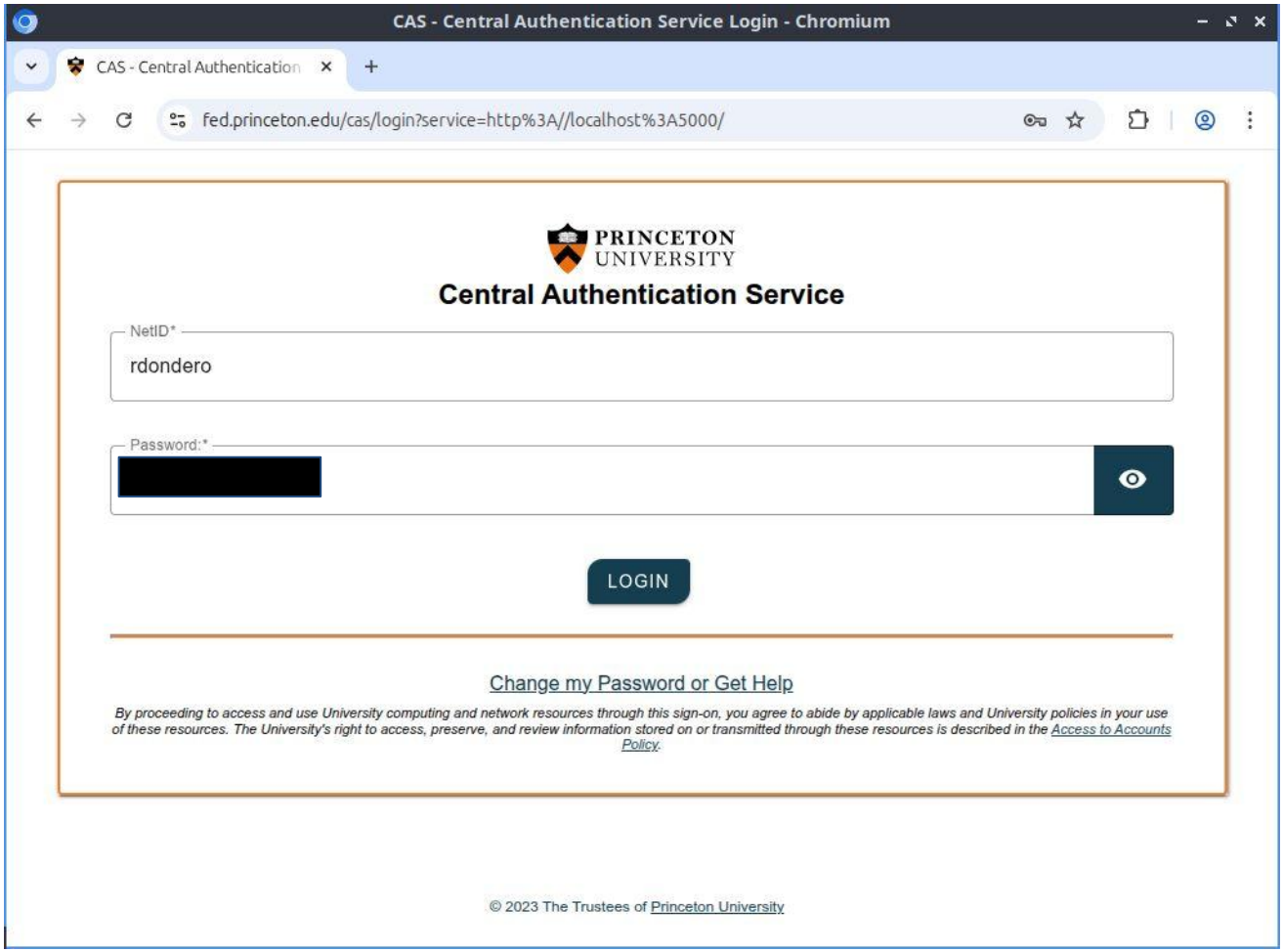
Test Automation

Solution: The general idea (cont.):

```
$ export TESTING="no"
$ python runserver.py
 * Serving Flask app 'top'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.29:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 457-747-552
...
```

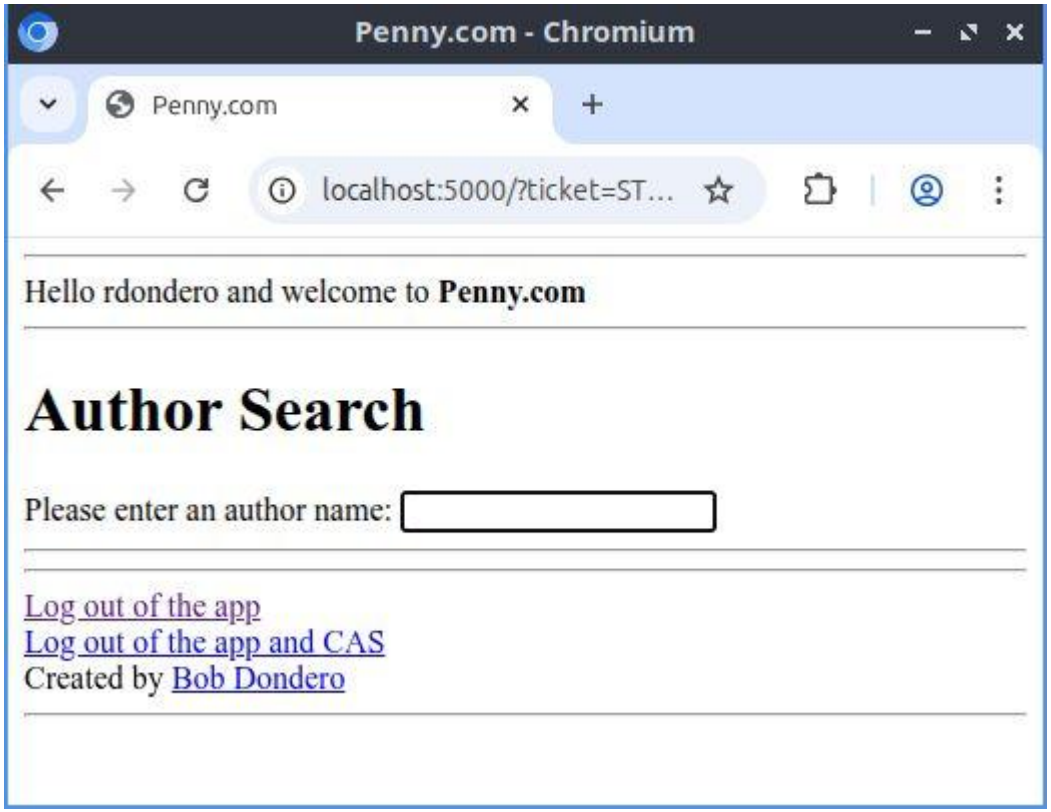
Test Automation

Solution: The general idea (cont.):



Test Automation

Solution: The general idea (cont.):



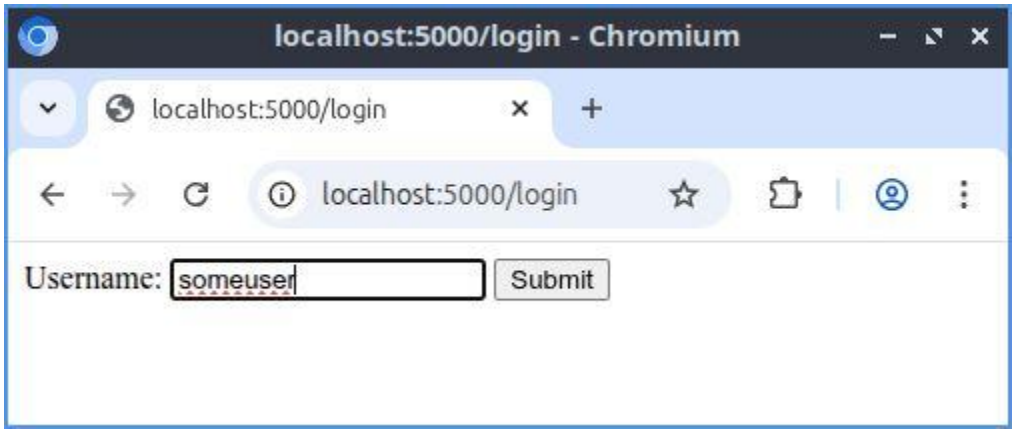
Test Automation

Solution: The general idea (cont.):

```
$ export TESTING="yes"
$ python runserver.py
 * Serving Flask app 'top'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.29:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 457-747-552
...
```

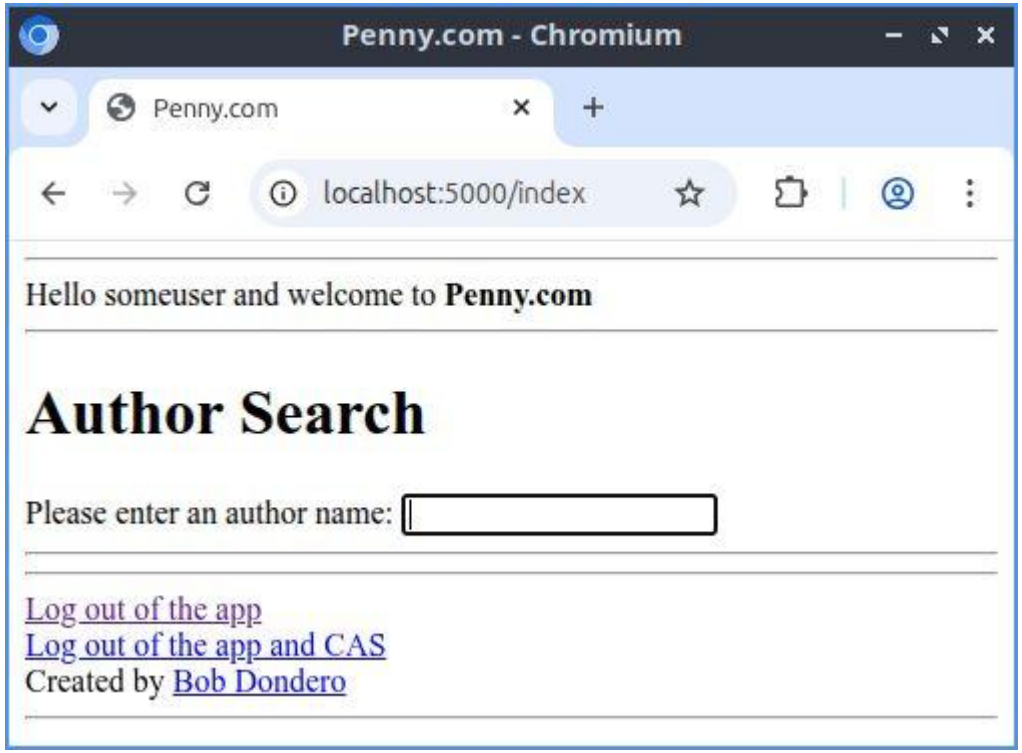
Test Automation

Solution: The general idea (cont.):



Test Automation

Solution: The general idea (cont.):



Test Automation

- Python **authenticating web app** testing
  - See **PennyTestCas** app
    - runserver.py
    - penny.sql, penny.sqlite
    - **index.html**, **loggedout.html**
    - database.py
    - **top.py**, **penny.py**
    - **auth.py**, **authstub.py**
    - testdatabase.py
    - **testpenny.py**

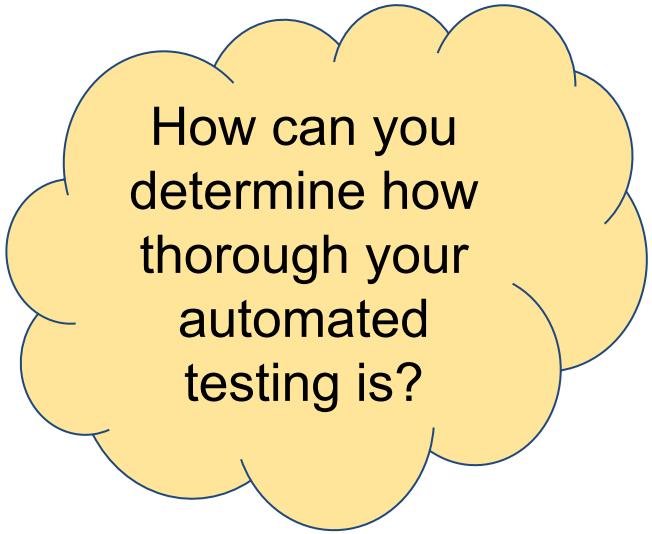
Test Automation

- Python **authenticating web app** testing
  - See **PennyTestGoogle**
    - runserver.py
    - penny.sql, penny.sqlite
    - cert.pem, key.pem
    - **index.html**, **loggedout.html**
    - database.py
    - **top.py**, **penny.py**
    - **auth.py**, **authstub.py**
    - testdatabase.py
    - **testpenny.py**

Test Automation

- Python **authenticating web app** testing
  - See **PennyTestEntra** app
    - runserver.py
    - penny.sql, penny.sqlite
    - index.html, loggedout.html
    - database.py
    - **top.py**, **penny.py**
    - **authstub.py**
    - testdatabase.py
    - **testpenny.py**

Test Automation



Agenda

- Tools for test automation
- **Tools for statement/coverage testing**

Statement/Coverage Testing

Tools for statement/coverage testing:

Language	Statement Testing Tool
Python	***coverage****
Java	***JaCoCo***
C	***gcov***
JavaScript (Node.js)	***nyc/istanbul**** ***Jest/Istanbul****
JavaScript (browser)	***Chrome Coverage***

* Described in the following slides

** See me if you want an example

Statement/Coverage Testing

- Python **module** testing
  - Recall **Fraction/**
    - euclid.py
    - fraction.py
    - fractionclient.py
    - testfraction.py

Statement/Coverage Testing

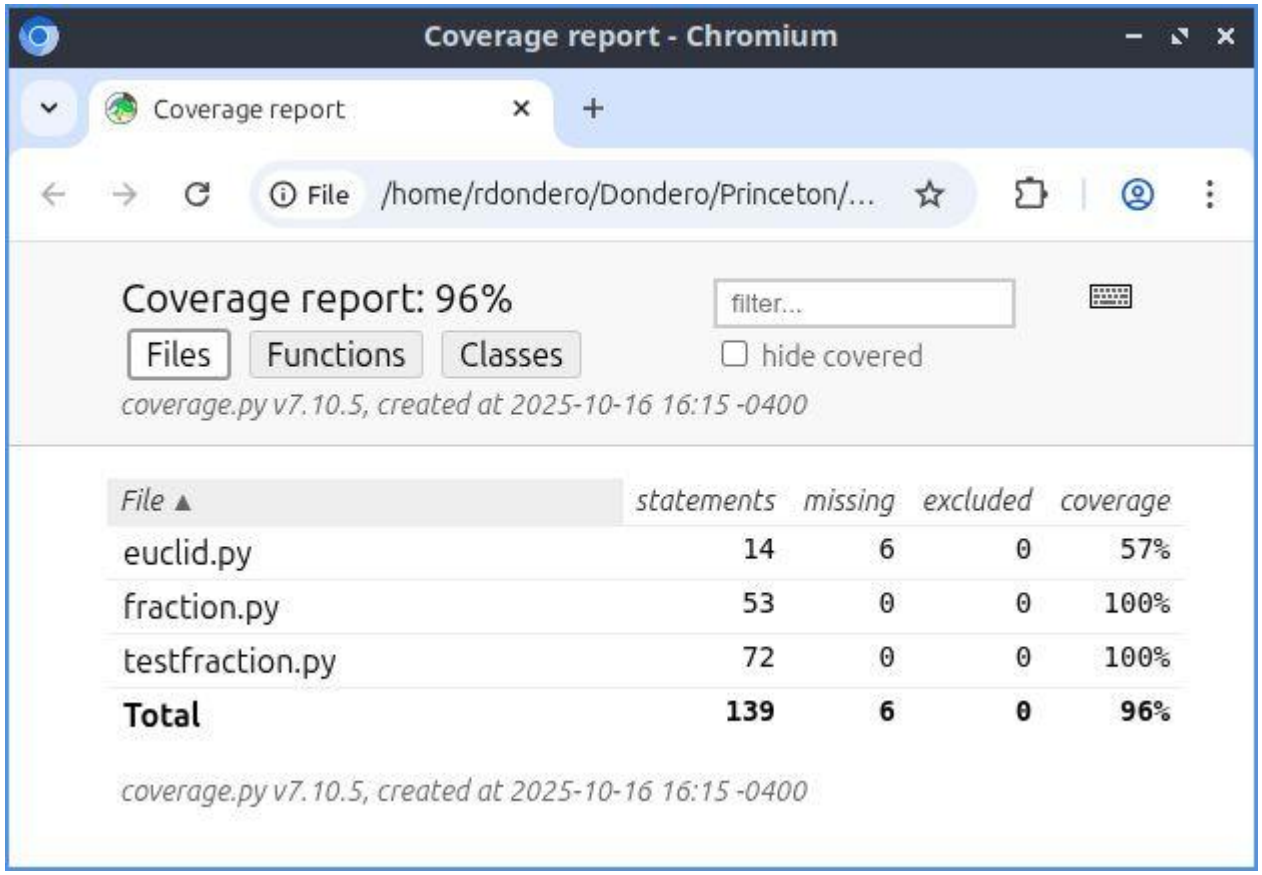
```
$ cd Fraction
$ python -m coverage run -p testfraction.py
test_add (testfraction.TestFraction.test_add) ... ok
test_constructor (testfraction.TestFraction.test_constructor) ... ok
test_eq (testfraction.TestFraction.test_eq) ... ok
test_ge (testfraction.TestFraction.test_ge) ... ok
test_gt (testfraction.TestFraction.test_gt) ... ok
test_hash (testfraction.TestFraction.test_hash) ... ok
test_le (testfraction.TestFraction.test_le) ... ok
test_lt (testfraction.TestFraction.test_lt) ... ok
test_mul (testfraction.TestFraction.test_mul) ... ok
test_ne (testfraction.TestFraction.test_ne) ... ok
test_neg (testfraction.TestFraction.test_neg) ... ok
test_str (testfraction.TestFraction.test_str) ... ok
test_sub (testfraction.TestFraction.test_sub) ... ok
test_truediv (testfraction.TestFraction.test_truediv) ... ok

----------------------------------------------------------------------
Ran 14 tests in 0.002s

OK
$
```

Statement/Coverage Testing

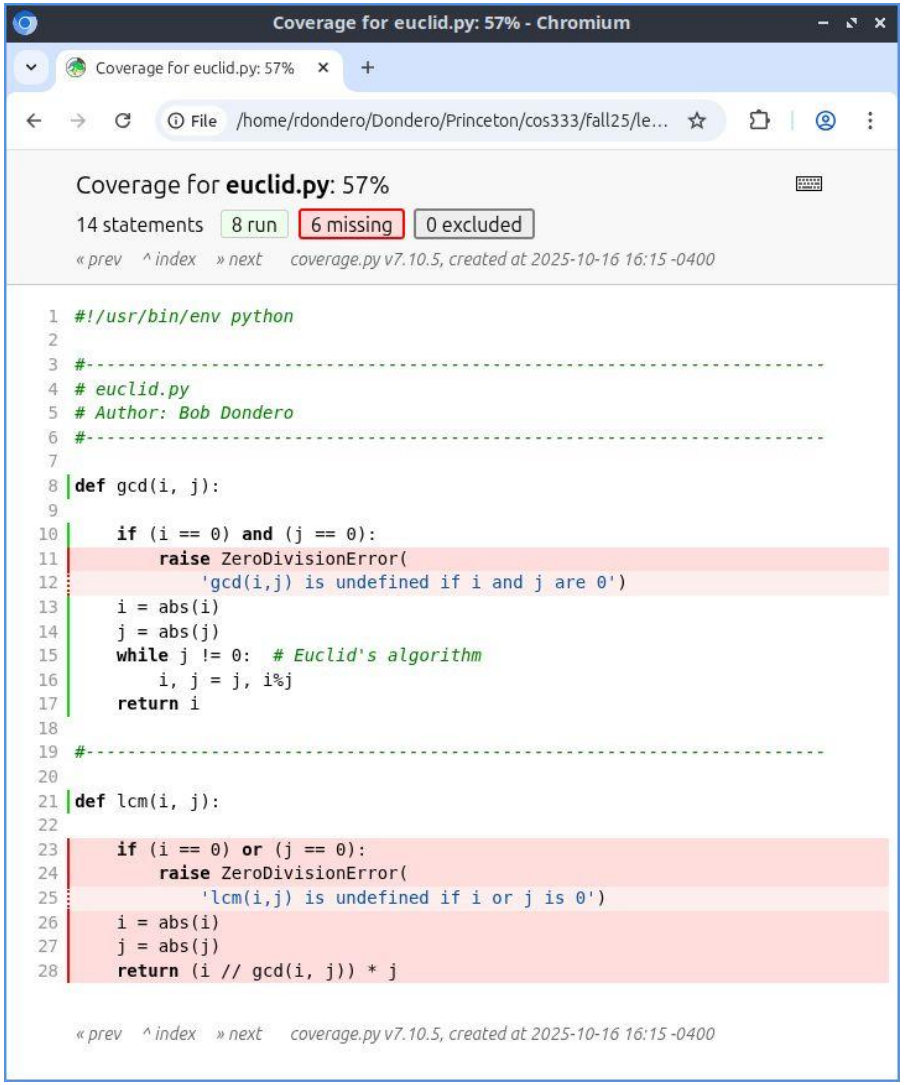
```
...
$ python -m coverage combine
Combined data file .coverage.rdondero-latitudee6430s.8287.XSCZzRMx
$ python -m coverage html
Wrote HTML report to htmlcov/index.html
$
```

Then browse to htmlcov/index.html…

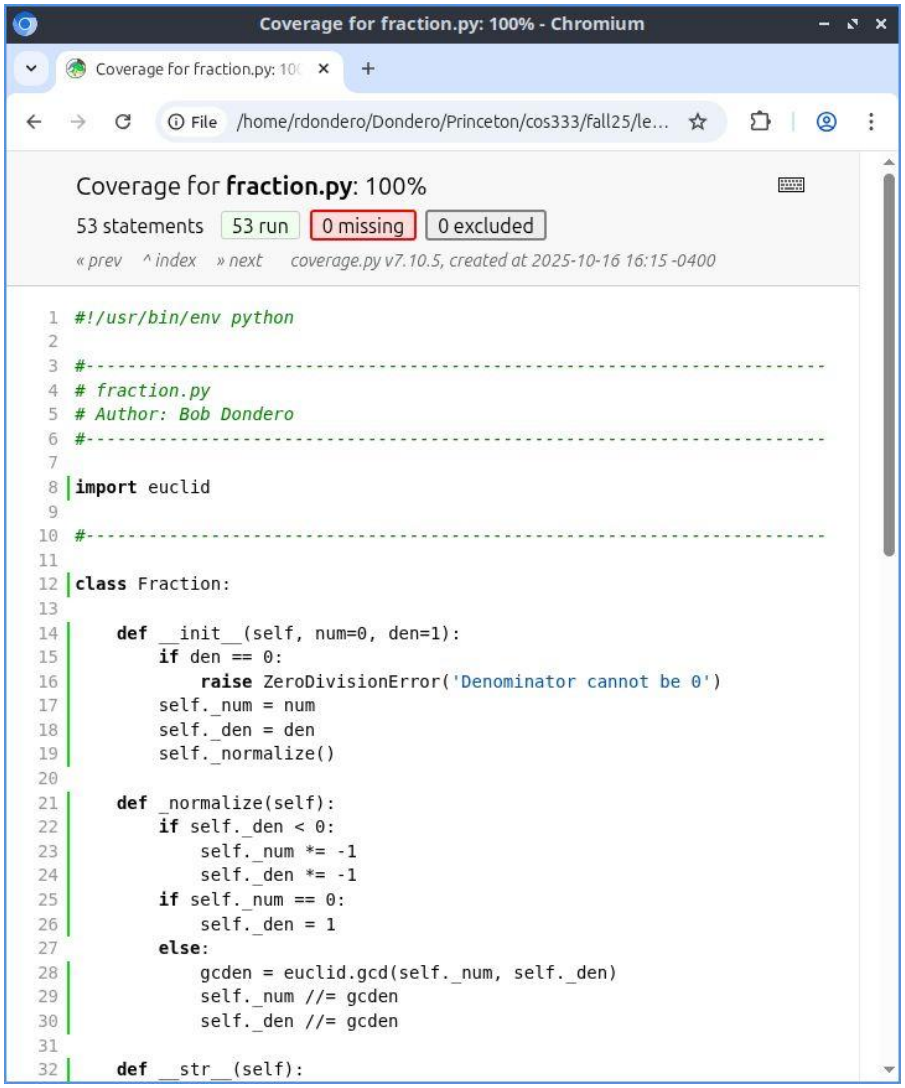
Statement/Coverage Testing



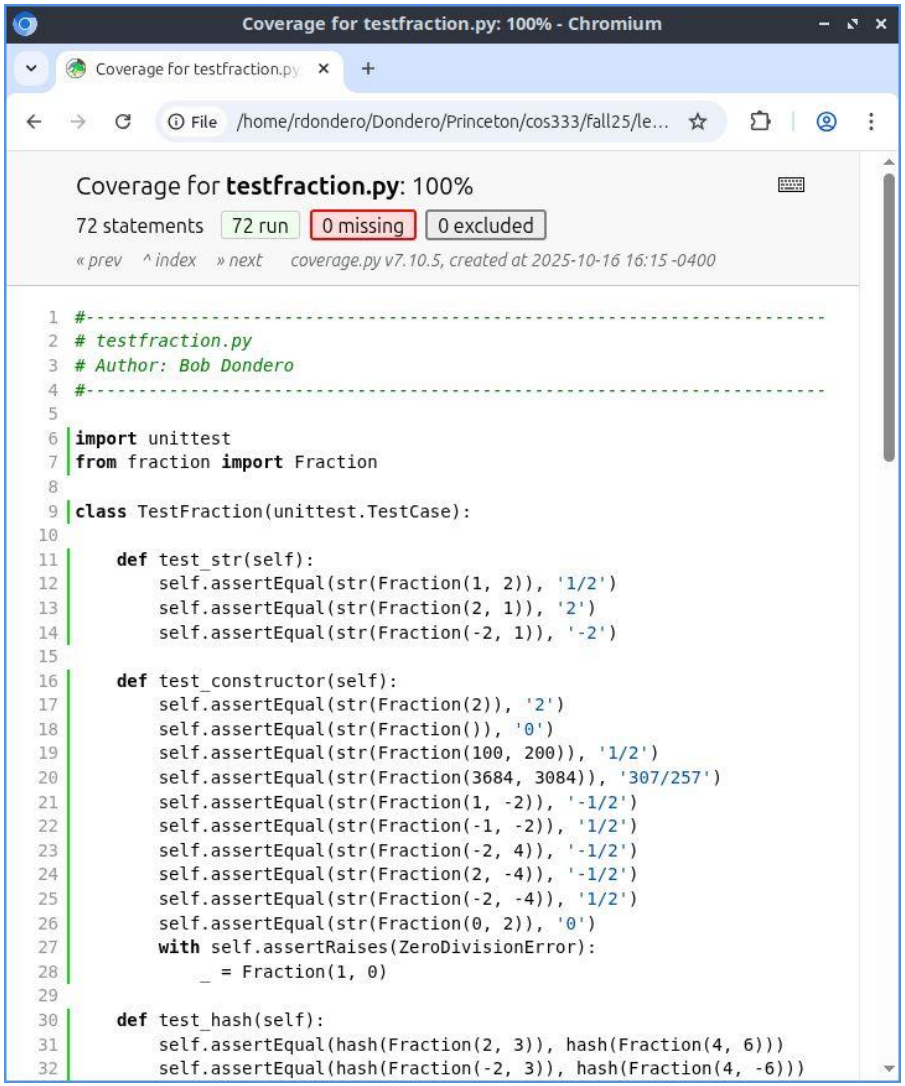
Statement/Coverage Testing



Statement/Coverage Testing



Statement/Coverage Testing



Statement/Coverage Testing

- Python **web app** testing
  - Recall **PennyTest** app
    - runserver.py
    - penny.sql, penny.sqlite
    - database.py
    - penny.py
    - index.html
    - testdatabase.py
    - testpenny.py

Statement/Coverage Testing

```
$ python -m coverage run -p runserver.py
 * Serving Flask app 'penny'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server
instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.29:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
```

```
$ python testpenny.py
test_abc (testpenny.PennyTestCase.test_abc) ... ok
test_ker (testpenny.PennyTestCase.test_ker) ... ok
test_ker_bad_spaces (testpenny.PennyTestCase.test_ker_bad_spaces) ... ok
test_ker_spaces (testpenny.PennyTestCase.test_ker_spaces) ... ok
test_sed (testpenny.PennyTestCase.test_sed) ... ok
test_spaces (testpenny.PennyTestCase.test_spaces) ... ok

----------------------------------------------------------------------
Ran 6 tests in 13.278s

OK
$
```

Statement/Coverage Testing

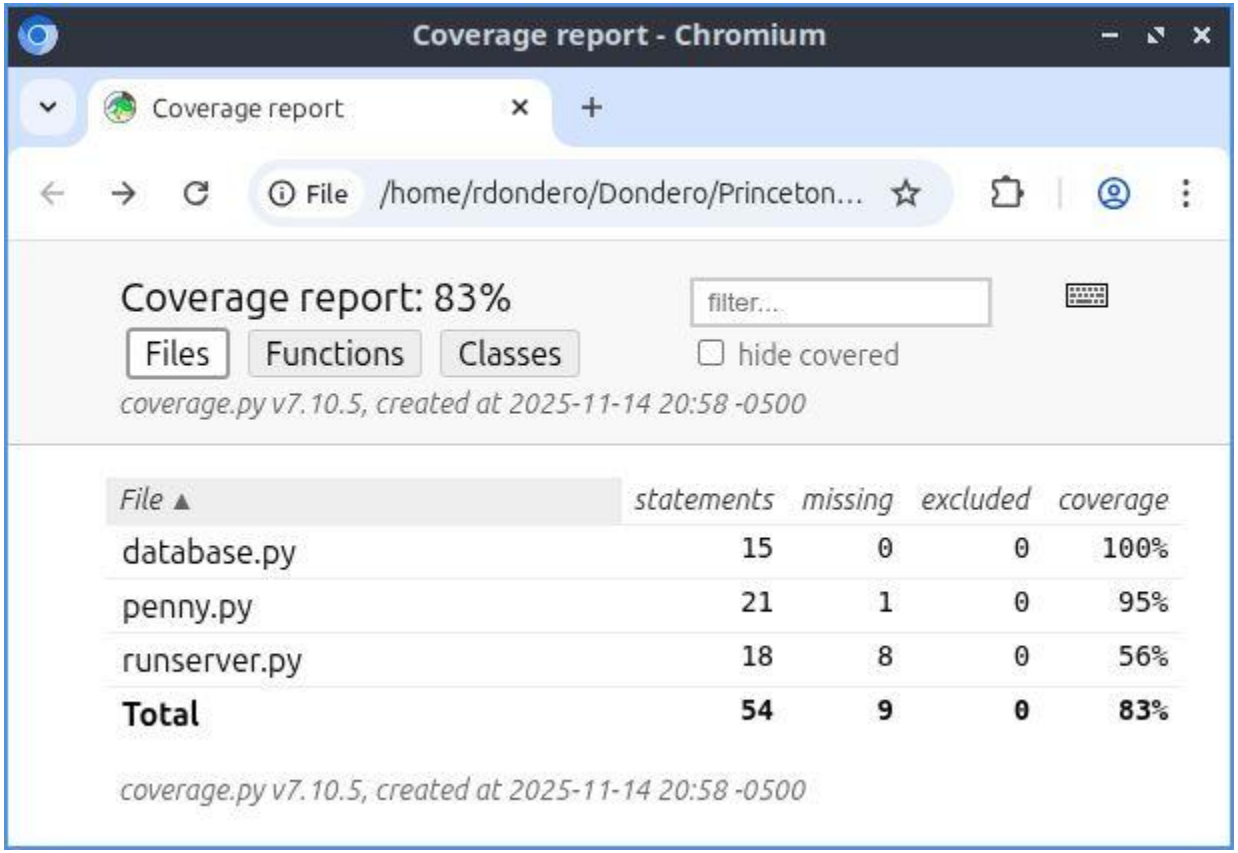
```
...
127.0.0.1 - - [14/Nov/2025 20:51:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:51] "GET /searchresults?author=abc HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:54] "GET /searchresults?author=ker HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:55] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:56] "GET /searchresults?author=%20ker%20nigh%20
HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:57] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:58] "GET /searchresults?author=%20kernigh%20
HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:52:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:52:00] "GET /searchresults?author=sed HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:52:02] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:52:02] "GET /searchresults?author=%20%20%20 HTTP/1.1"
200 -
^C
...
```

Statement/Coverage Testing

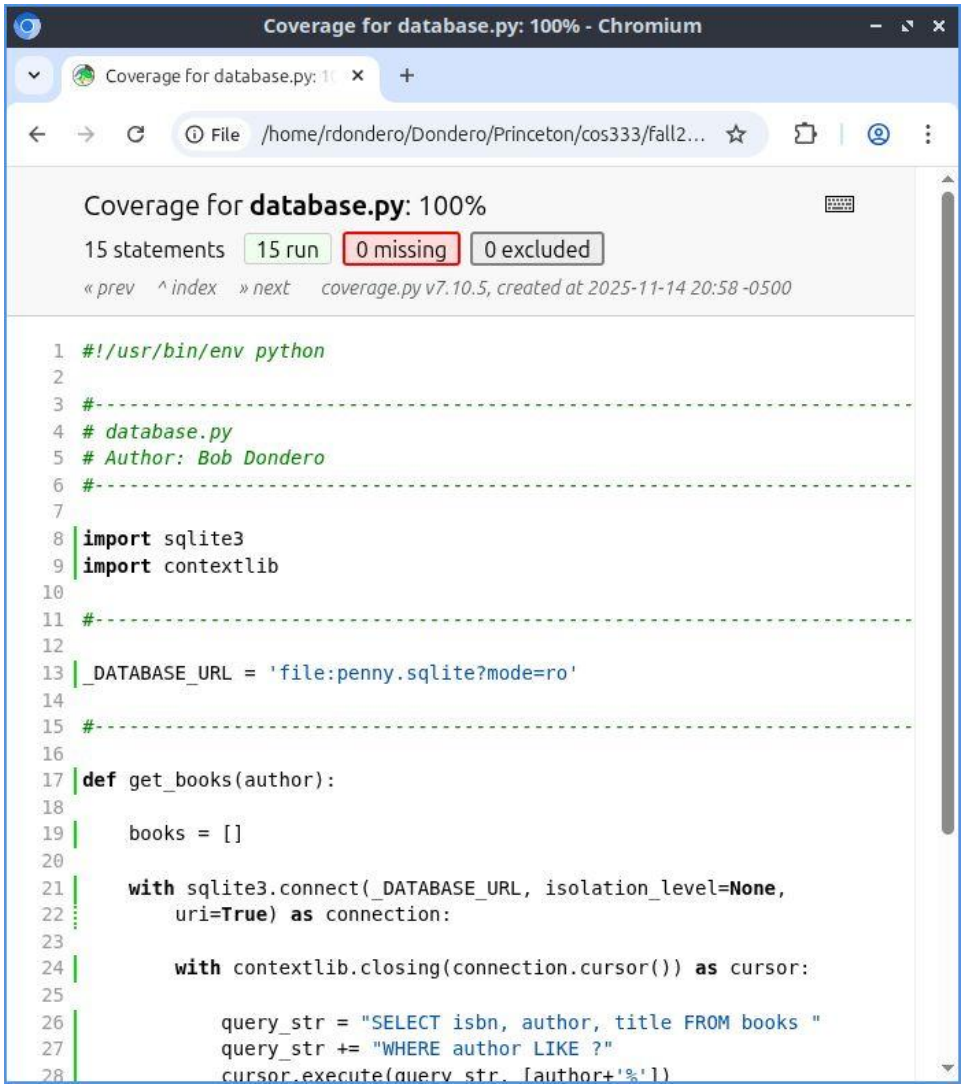
```
...
$ python -m coverage combine
Combined data file .coverage.rdondero-latitudee6430s.19007.XRrRksyx
Combined data file .coverage.rdondero-latitudee6430s.19008.XyCMLoLx
$ python -m coverage html
Wrote HTML report to htmlcov/index.html
$
```

Then browse to htmlcov/index.html…

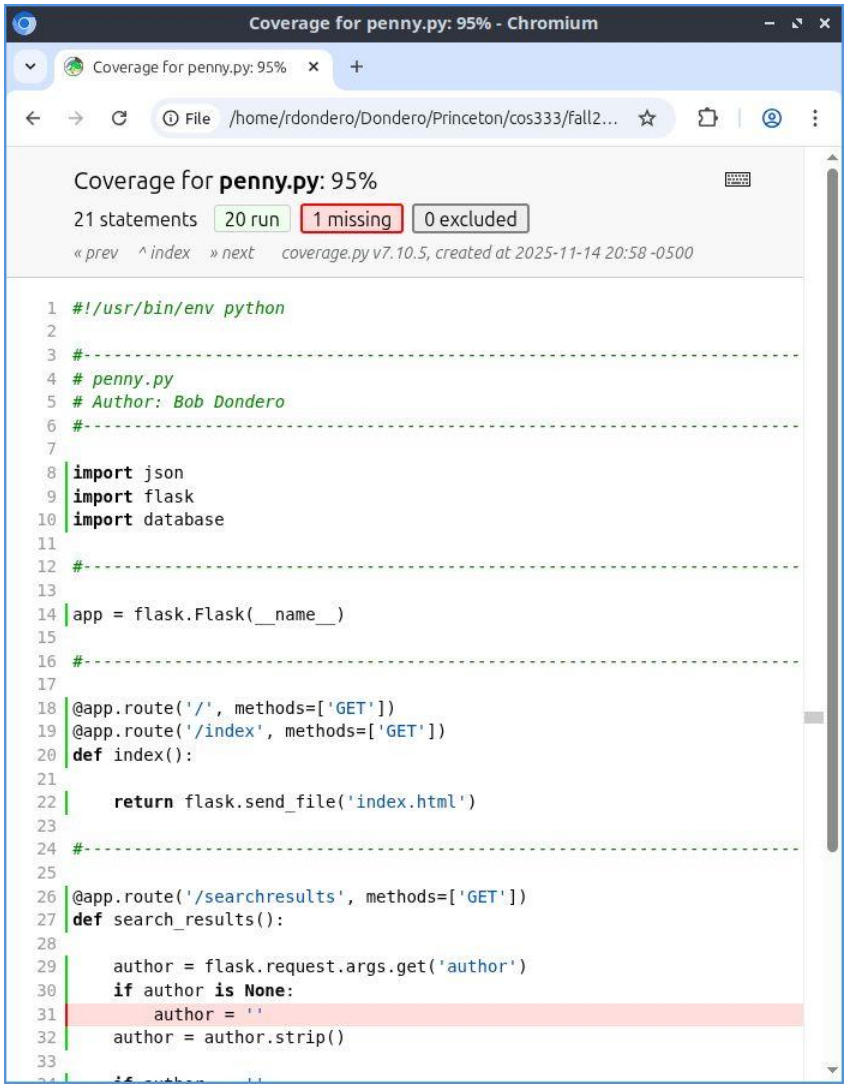
Statement/Coverage Testing



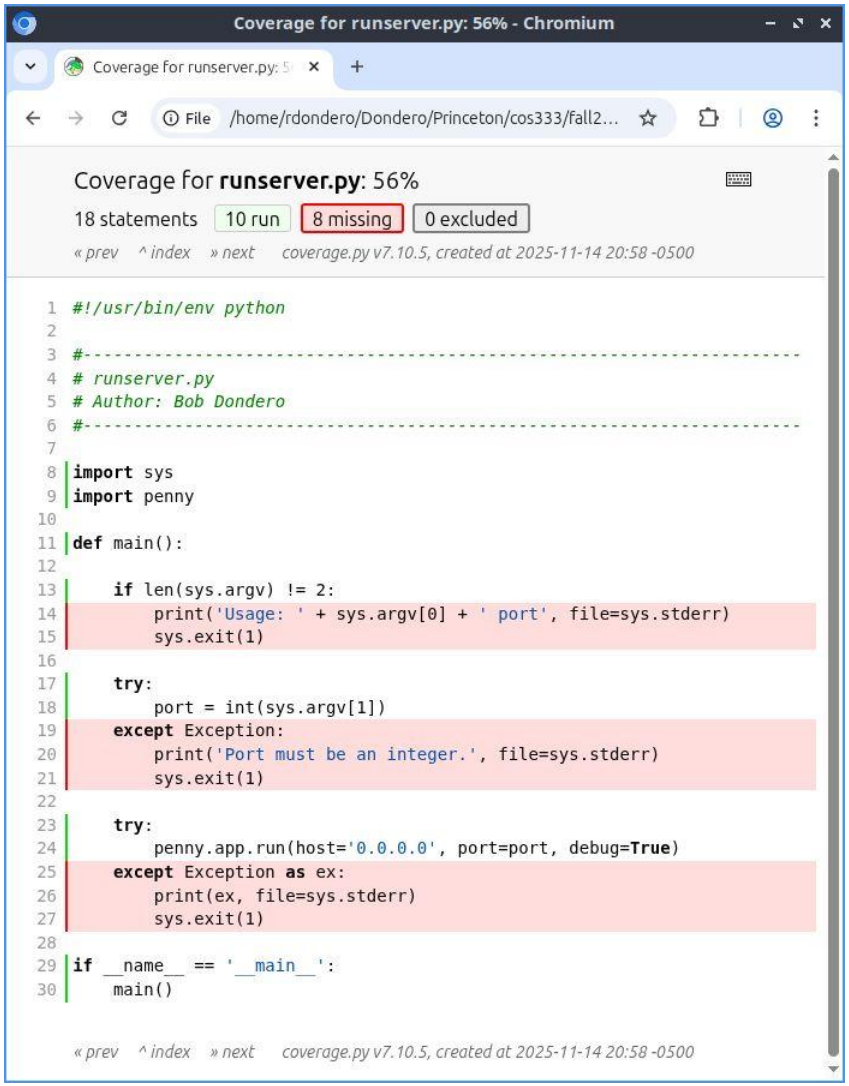
Statement/Coverage Testing



Statement/Coverage Testing



Statement/Coverage Testing



Statement/Coverage Testing

- **Problem**
  - How can you do statement/coverage testing on your app if it's written in JavaScript?
- **Solution**
  - See me for an example use of **nyc** and **Jest**

Statement/Coverage Testing

- Notes:
  - Statement/coverage testing can be use **without** test automation
  - Statement/coverage testing can be used to test coverage of **multiple** runs of a program

Statement/Coverage Testing

- Project concern?
  - Statement/coverage testing: **Yes**
    - But not a burden
    - We'll expect reasonable coverage
      - But not 100% coverage
  - Test automation: **No**
    - But we'll award a little extra credit for proof of concept

Summary

- We have covered:
  - Tools related to testing
- Specifically:
  - Tools for test automation
  - Tools for statement/coverage testing