

# Software Engineering: Testing Tools

Copyright © 2025 by  
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               | <i>unittest</i> (formerly <i>PyUnit</i> )*<br><i>PyTest</i> |
| Java                 | <i>JUnit</i>                                                |
| C                    | <i>CUnit</i>                                                |
| JavaScript (browser) | <i>Playwright</i> *<br><i>Selenium</i>                      |
| JavaScript (Node.js) | <i>Jest</i> **                                              |

\* 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: 2/3
$
```

# 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
...
```

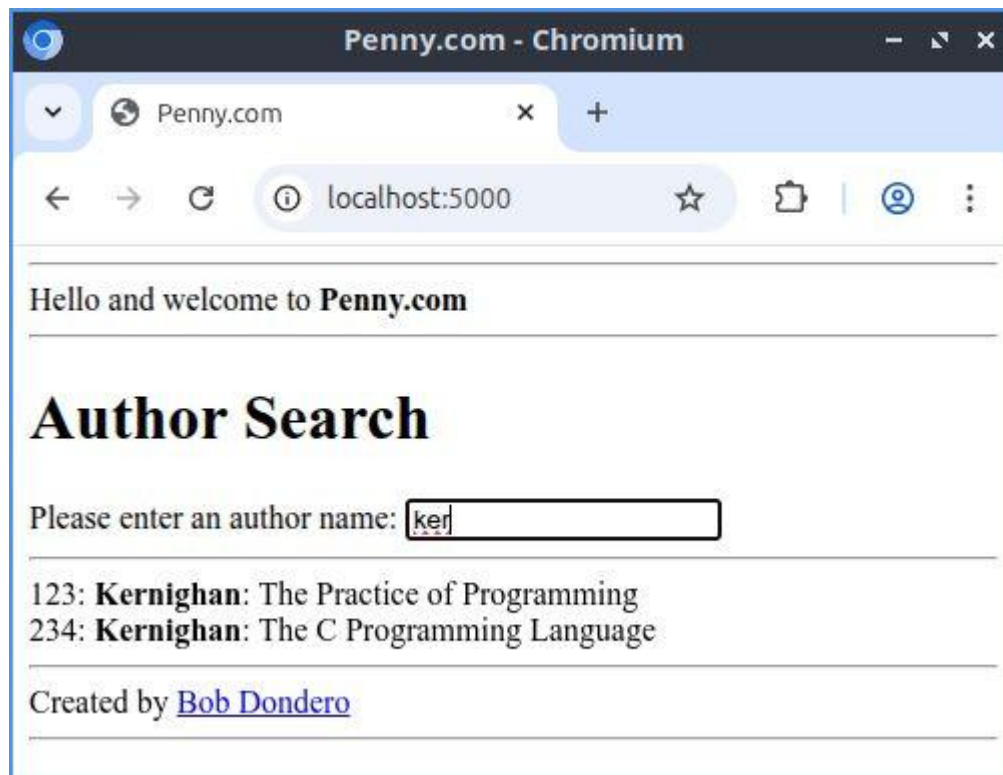
# Test Automation

- 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  
...
```

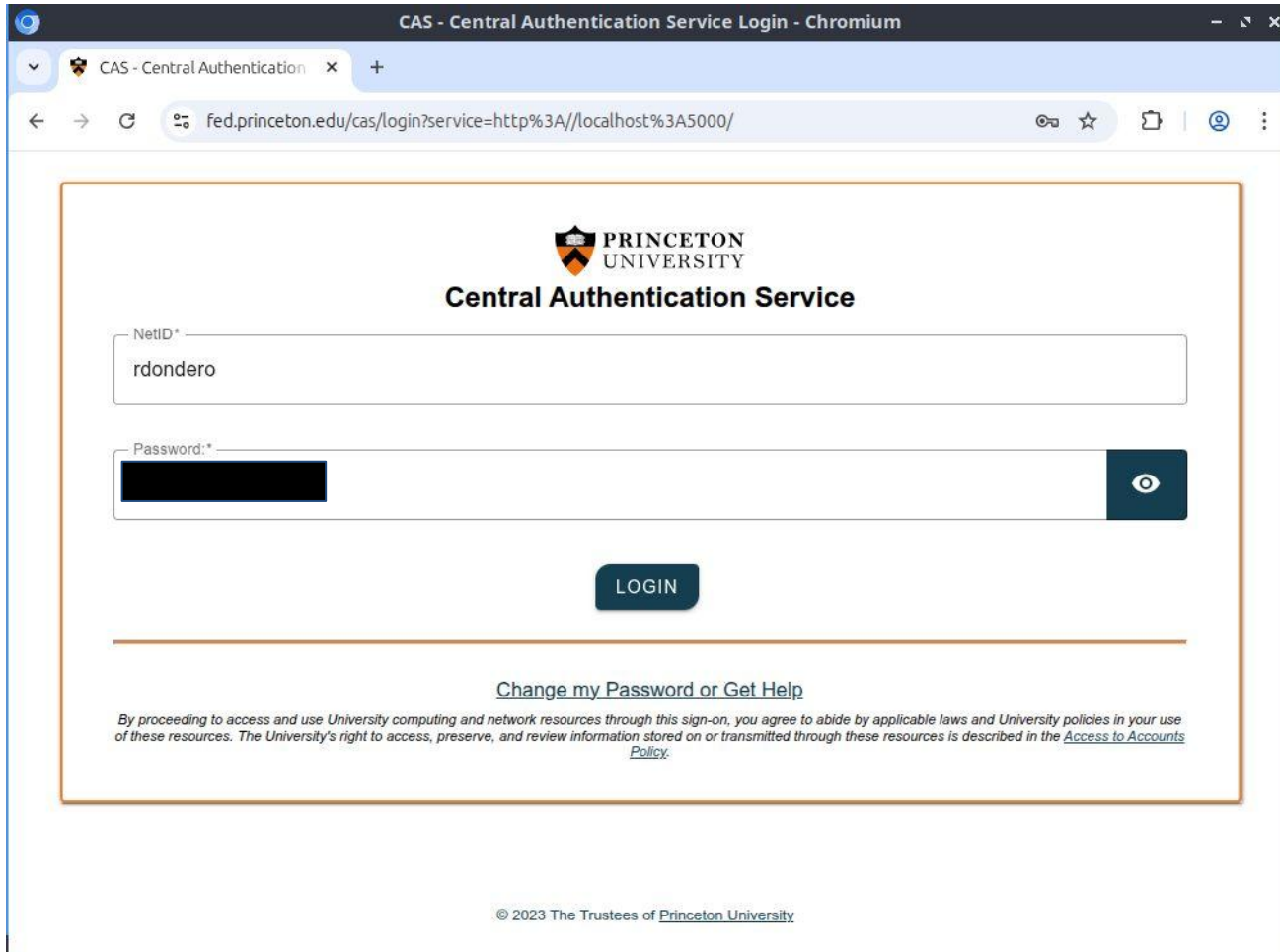
# Test Automation

## 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.):



The screenshot shows a web browser window titled "CAS - Central Authentication Service Login - Chromium". The address bar displays the URL "fed.princeton.edu/cas/login?service=http%3A//localhost%3A5000/". The main content area features the Princeton University logo and the text "Central Authentication Service". Below this, there are two input fields: "NetID\*" with the value "rdondero" and "Password:" with a masked password. A "LOGIN" button is positioned below the password field. At the bottom, there is a link "Change my Password or Get Help" and a disclaimer: "By proceeding to access and use University computing and network resources through this sign-on, you agree to abide by applicable laws and University policies in your use of these resources. The University's right to access, preserve, and review information stored on or transmitted through these resources is described in the [Access to Accounts Policy](#)." The footer contains the copyright notice "© 2023 The Trustees of Princeton University".

CAS - Central Authentication Service Login - Chromium

CAS - Central Authentication

fed.princeton.edu/cas/login?service=http%3A//localhost%3A5000/

 **PRINCETON UNIVERSITY**

**Central Authentication Service**

NetID\*

Password:\*  

**LOGIN**

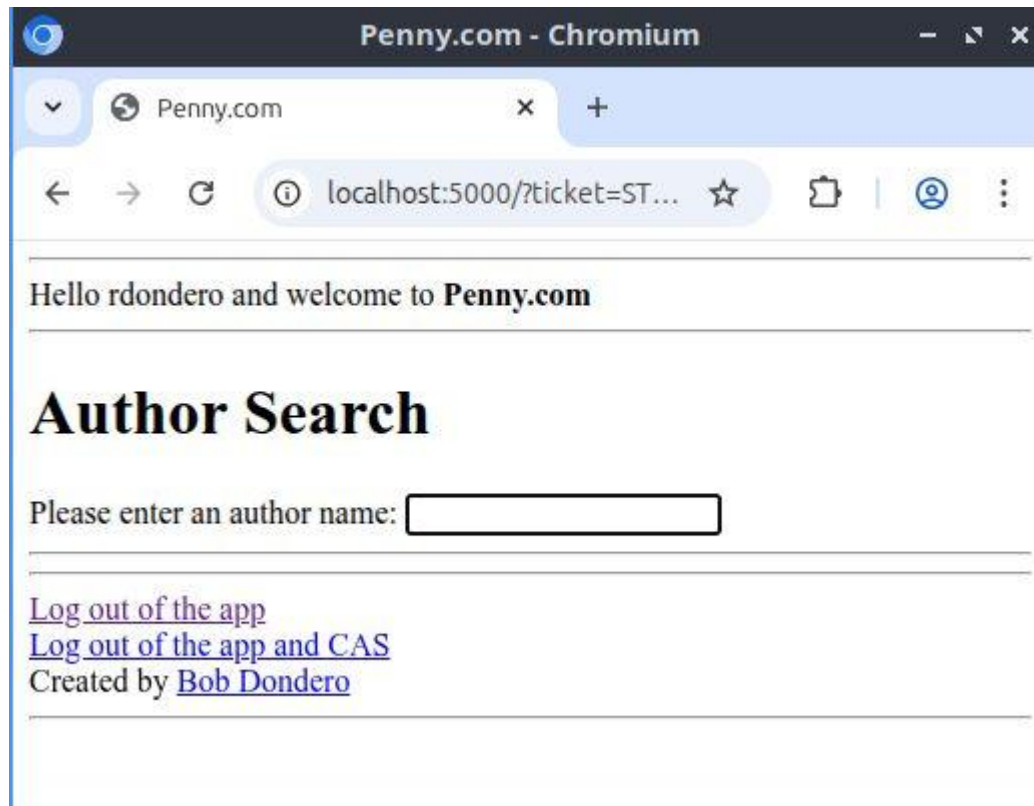
[Change my Password or Get Help](#)

By proceeding to access and use University computing and network resources through this sign-on, you agree to abide by applicable laws and University policies in your use of these resources. The University's right to access, preserve, and review information stored on or transmitted through these resources is described in the [Access to Accounts Policy](#).

© 2023 The Trustees of Princeton University

# Test Automation

**Solution:** The general idea (cont.):



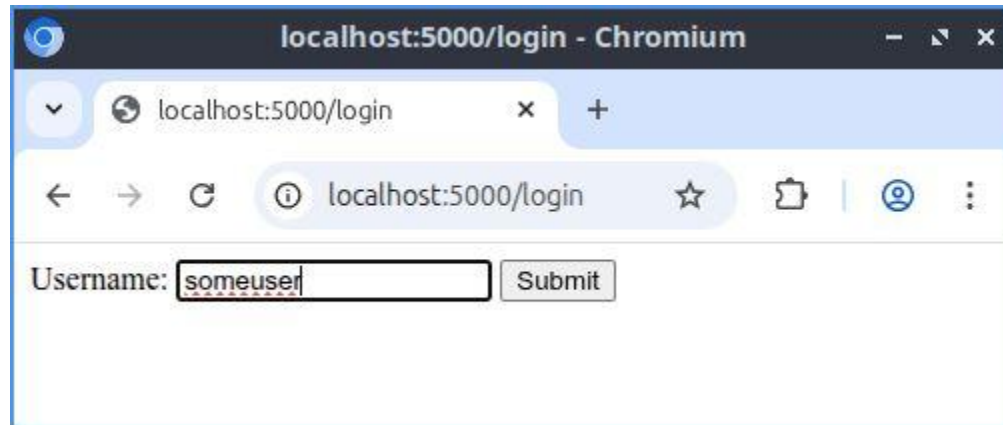
# Test Automation

## 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
...
```

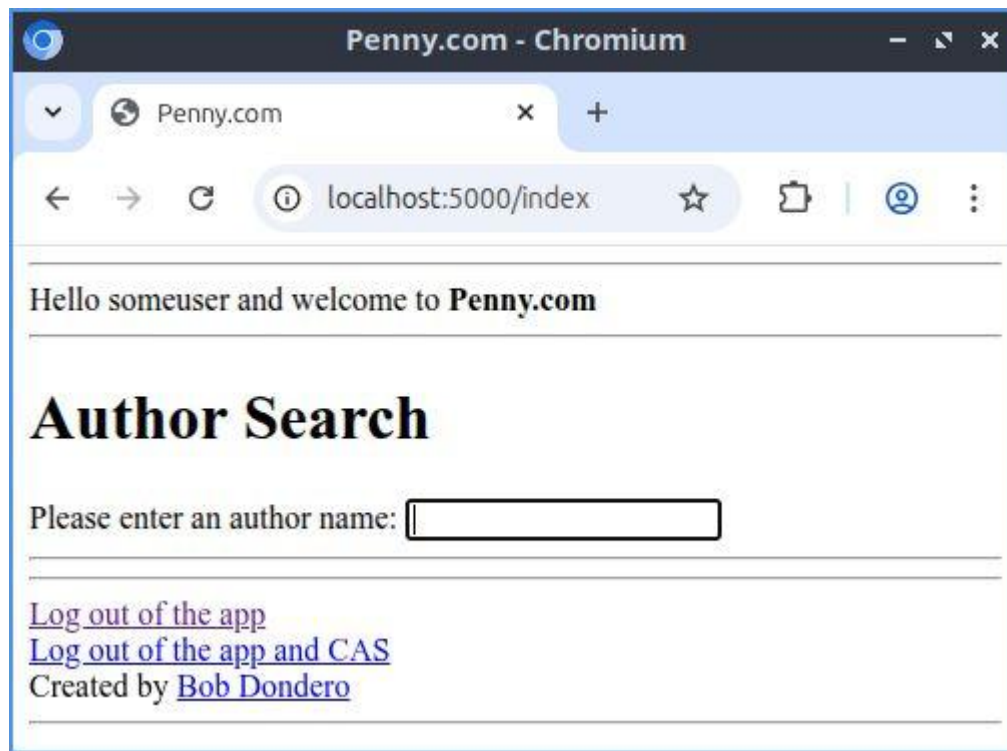
# Test Automation

**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**

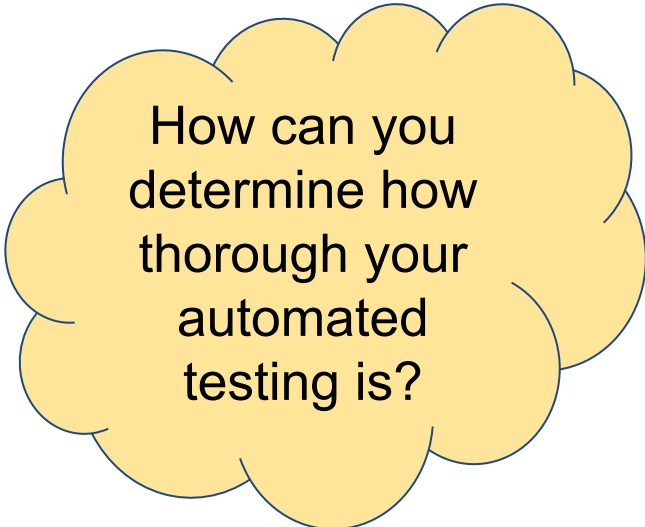
# Test Automation

- 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



How can you  
determine how  
thorough your  
automated  
testing is?

# Agenda

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

# Statement/Coverage Testing

Tools for statement/coverage testing:

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

\* 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

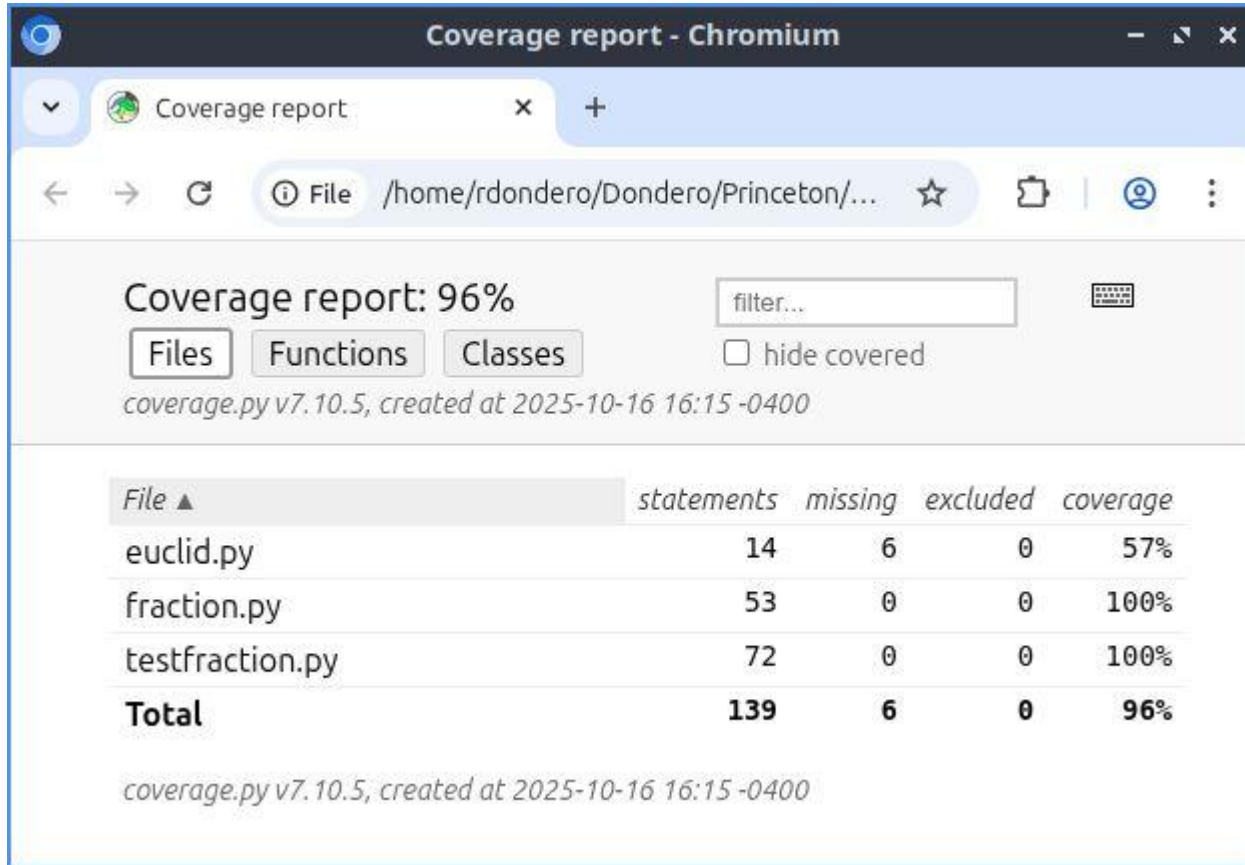
OK
$
```

# 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](http://localhost/htmlcov/index.html)...

# Statement/Coverage Testing



Coverage report: 96%

filter...

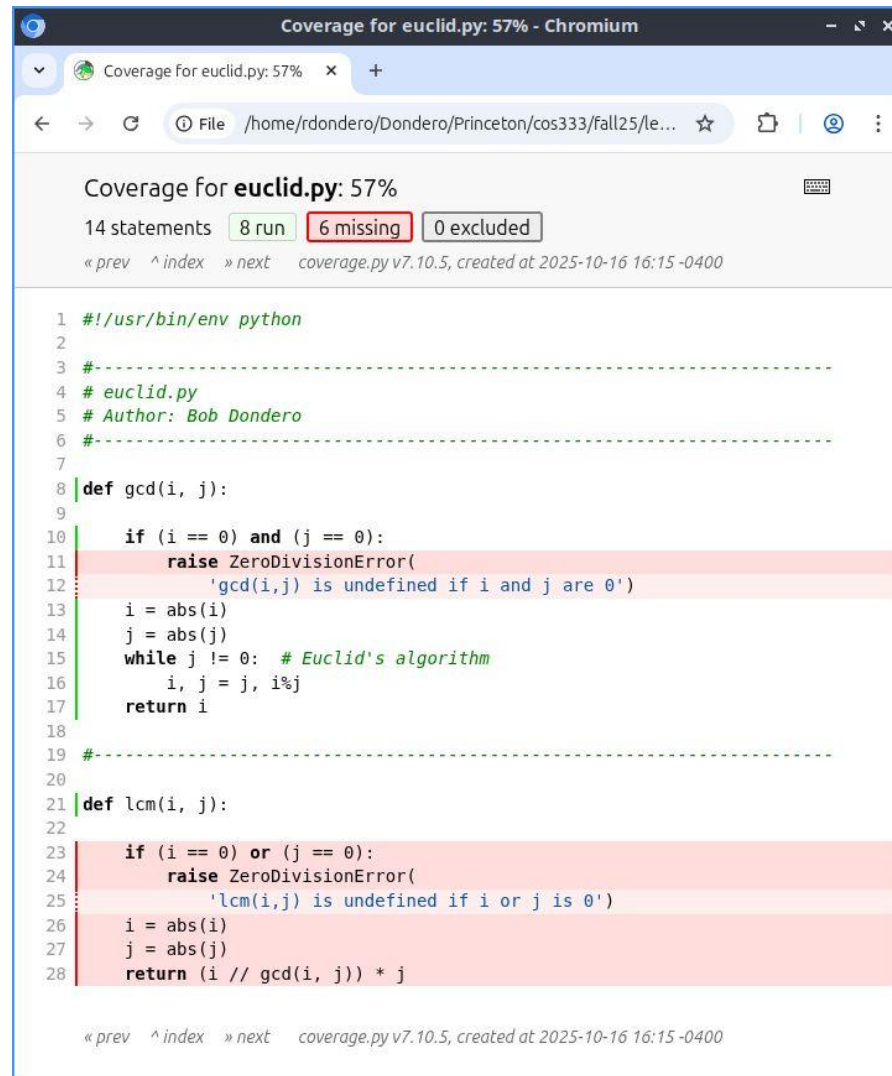
☐ hide covered

coverage.py v7.10.5, created at 2025-10-16 16:15 -0400

| File ▲          | statements | missing  | excluded | coverage   |
|-----------------|------------|----------|----------|------------|
| euclid.py       | 14         | 6        | 0        | 57%        |
| fraction.py     | 53         | 0        | 0        | 100%       |
| testfraction.py | 72         | 0        | 0        | 100%       |
| <b>Total</b>    | <b>139</b> | <b>6</b> | <b>0</b> | <b>96%</b> |

coverage.py v7.10.5, created at 2025-10-16 16:15 -0400

# Statement/Coverage Testing



Coverage for euclid.py: 57% - Chromium

Coverage for euclid.py: 57%

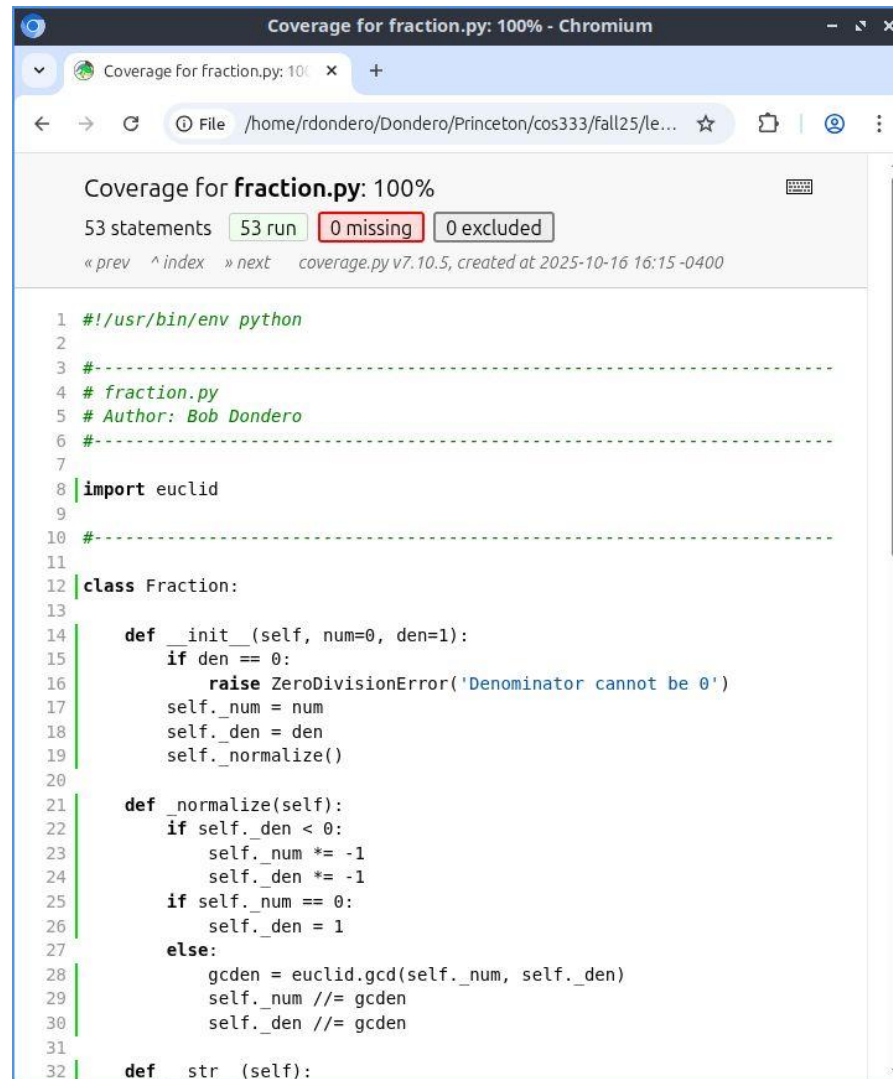
14 statements 8 run 6 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-10-16 16:15 -0400

```
1  #!/usr/bin/env python
2
3  #-----
4  # euclid.py
5  # Author: Bob Dondero
6  #-----
7
8  def gcd(i, j):
9
10     if (i == 0) and (j == 0):
11         raise ZeroDivisionError(
12             'gcd(i,j) is undefined if i and j are 0')
13     i = abs(i)
14     j = abs(j)
15     while j != 0: # Euclid's algorithm
16         i, j = j, i%j
17     return i
18
19 #-----
20
21 def lcm(i, j):
22
23     if (i == 0) or (j == 0):
24         raise ZeroDivisionError(
25             'lcm(i,j) is undefined if i or j is 0')
26     i = abs(i)
27     j = abs(j)
28     return (i // gcd(i, j)) * j
```

« prev ^ index » next coverage.py v7.10.5, created at 2025-10-16 16:15 -0400

# Statement/Coverage Testing



Coverage for fraction.py: 100% - Chromium

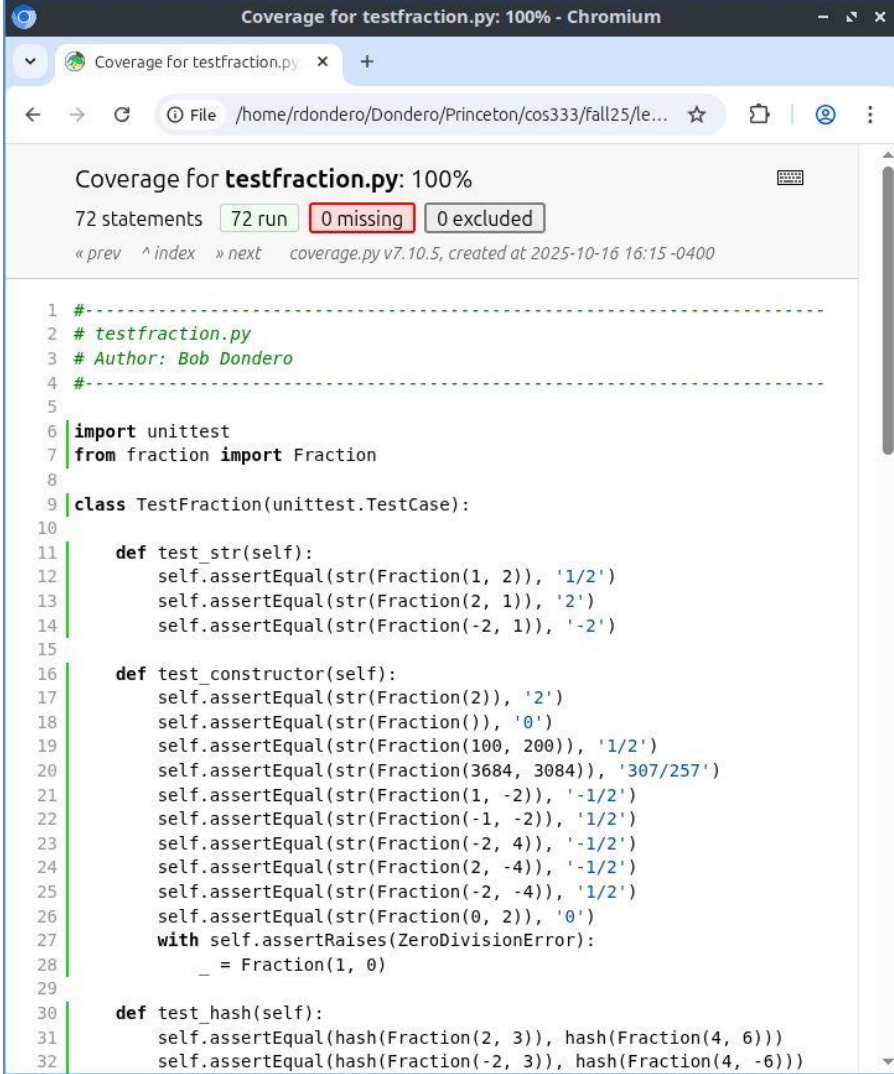
Coverage for fraction.py: 100%

53 statements 53 run 0 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-10-16 16:15-0400

```
1  #!/usr/bin/env python
2
3  #-----
4  # fraction.py
5  # Author: Bob Dondero
6  #-----
7
8  import euclid
9
10 #-----
11
12 class Fraction:
13
14     def __init__(self, num=0, den=1):
15         if den == 0:
16             raise ZeroDivisionError('Denominator cannot be 0')
17         self._num = num
18         self._den = den
19         self._normalize()
20
21     def _normalize(self):
22         if self._den < 0:
23             self._num *= -1
24             self._den *= -1
25         if self._num == 0:
26             self._den = 1
27         else:
28             gcden = euclid.gcd(self._num, self._den)
29             self._num //= gcden
30             self._den //= gcden
31
32     def __str__(self):
```

# Statement/Coverage Testing



Coverage for testfraction.py: 100% - Chromium

Coverage for testfraction.py x +

File /home/rdontero/Dondero/Princeton/cos333/fall25/le... ☆

Coverage for **testfraction.py**: 100%

72 statements 72 run 0 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-10-16 16:15 -0400

```
1 #-----
2 # testfraction.py
3 # Author: Bob Dondero
4 #-----
5
6 import unittest
7 from fraction import Fraction
8
9 class TestFraction(unittest.TestCase):
10
11     def test_str(self):
12         self.assertEqual(str(Fraction(1, 2)), '1/2')
13         self.assertEqual(str(Fraction(2, 1)), '2')
14         self.assertEqual(str(Fraction(-2, 1)), '-2')
15
16     def test_constructor(self):
17         self.assertEqual(str(Fraction(2)), '2')
18         self.assertEqual(str(Fraction()), '0')
19         self.assertEqual(str(Fraction(100, 200)), '1/2')
20         self.assertEqual(str(Fraction(3684, 3084)), '307/257')
21         self.assertEqual(str(Fraction(1, -2)), '-1/2')
22         self.assertEqual(str(Fraction(-1, -2)), '1/2')
23         self.assertEqual(str(Fraction(-2, 4)), '-1/2')
24         self.assertEqual(str(Fraction(2, -4)), '-1/2')
25         self.assertEqual(str(Fraction(-2, -4)), '1/2')
26         self.assertEqual(str(Fraction(0, 2)), '0')
27         with self.assertRaises(ZeroDivisionError):
28             _ = Fraction(1, 0)
29
30     def test_hash(self):
31         self.assertEqual(hash(Fraction(2, 3)), hash(Fraction(4, 6)))
32         self.assertEqual(hash(Fraction(-2, 3)), hash(Fraction(4, -6)))
```

# 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

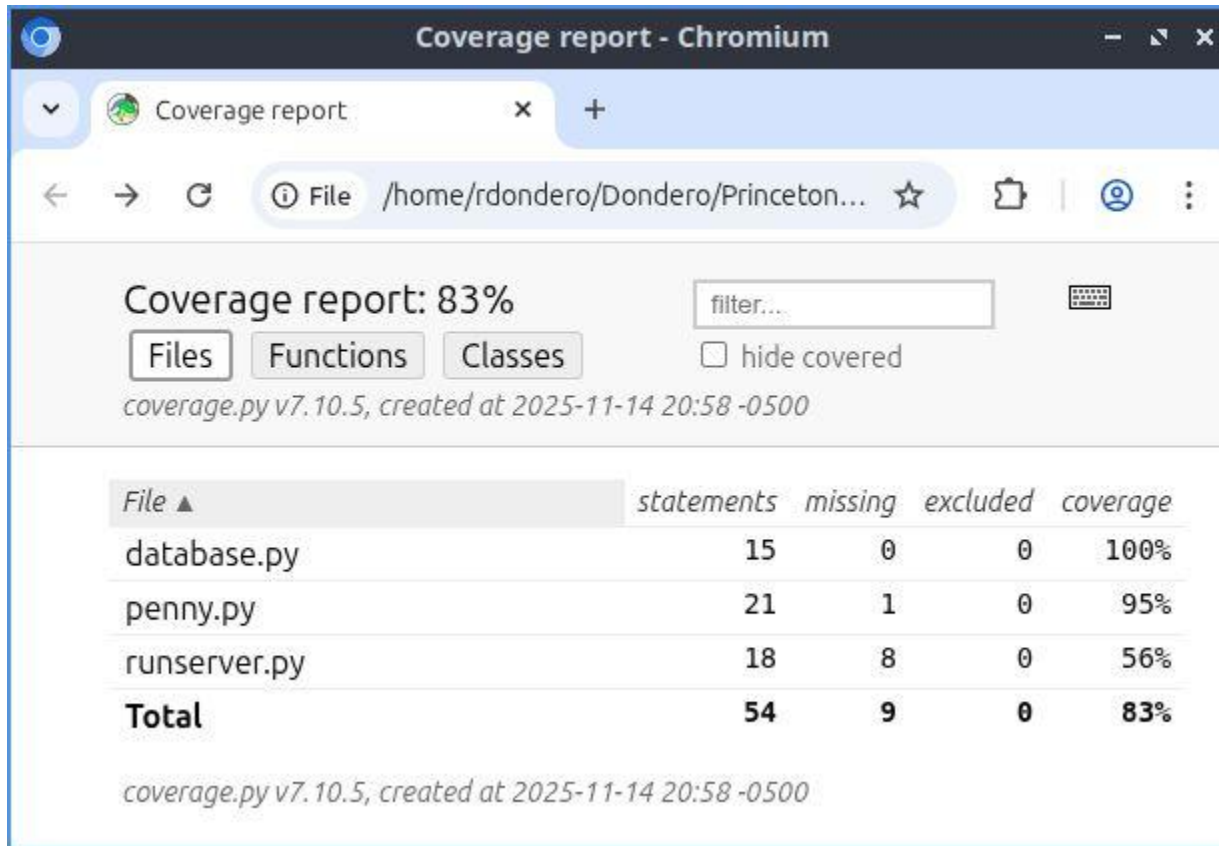
```
...
127.0.0.1 - - [14/Nov/2025 20:51:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:51] "GET /searchresults?author=abc HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2025 20:51:54] "GET /searchresults?author=ker HTTP/1.1" 200 -
127.0.0.1 - - [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](http://htmlcov/index.html)...

# Statement/Coverage Testing



Coverage report: 83%

filter...

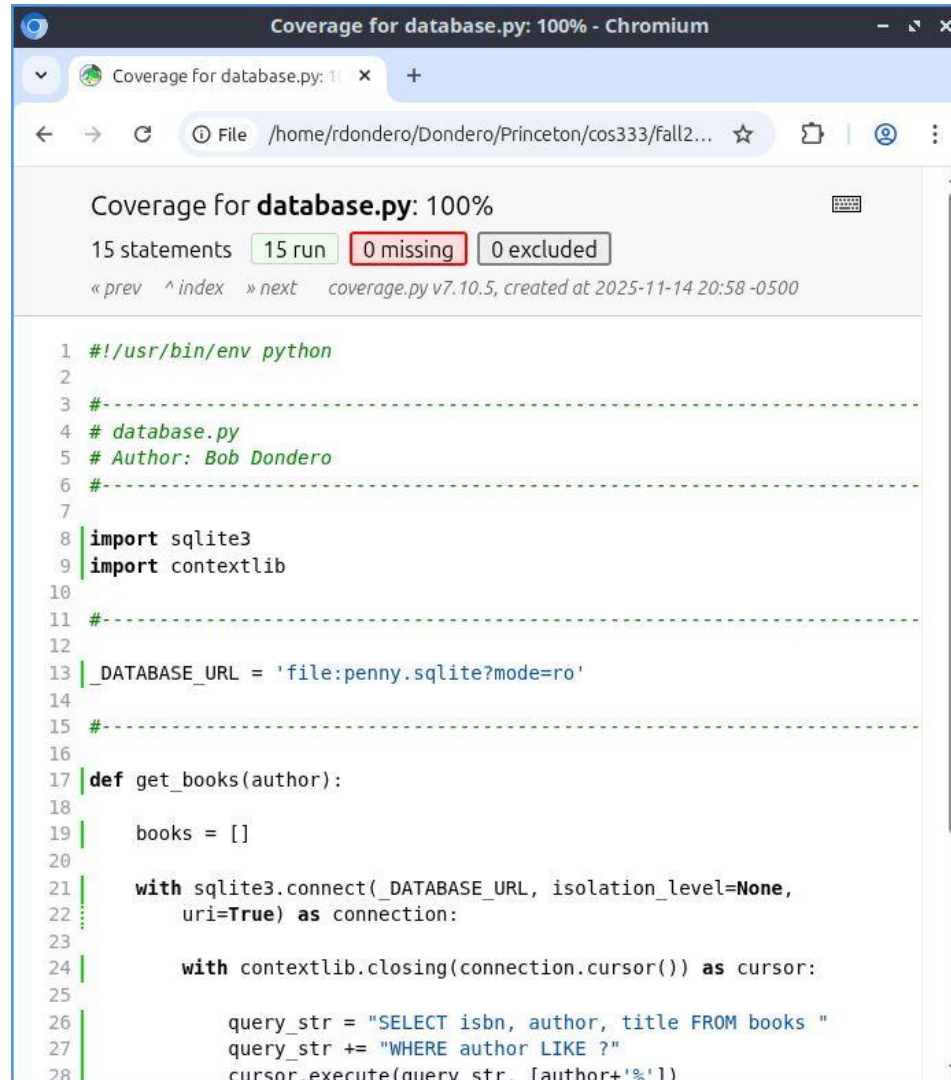
☐ hide covered

coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

| File ▲       | statements | missing  | excluded | coverage   |
|--------------|------------|----------|----------|------------|
| database.py  | 15         | 0        | 0        | 100%       |
| penny.py     | 21         | 1        | 0        | 95%        |
| runserver.py | 18         | 8        | 0        | 56%        |
| <b>Total</b> | <b>54</b>  | <b>9</b> | <b>0</b> | <b>83%</b> |

coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

# Statement/Coverage Testing



Coverage for database.py: 100% - Chromium

Coverage for database.py: 100%

15 statements 15 run 0 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

```
1  #!/usr/bin/env python
2
3  #-----
4  # database.py
5  # Author: Bob Dondero
6  #-----
7
8  import sqlite3
9  import contextlib
10
11 #-----
12
13 _DATABASE_URL = 'file:penny.sqlite?mode=ro'
14
15 #-----
16
17 def get_books(author):
18     books = []
19
20     with sqlite3.connect(_DATABASE_URL, isolation_level=None,
21         uri=True) as connection:
22         with contextlib.closing(connection.cursor()) as cursor:
23
24             query_str = "SELECT isbn, author, title FROM books "
25             query_str += "WHERE author LIKE ?"
26             cursor.execute(query_str, [author+'%'])
```

# Statement/Coverage Testing



Coverage for penny.py: 95% - Chromium

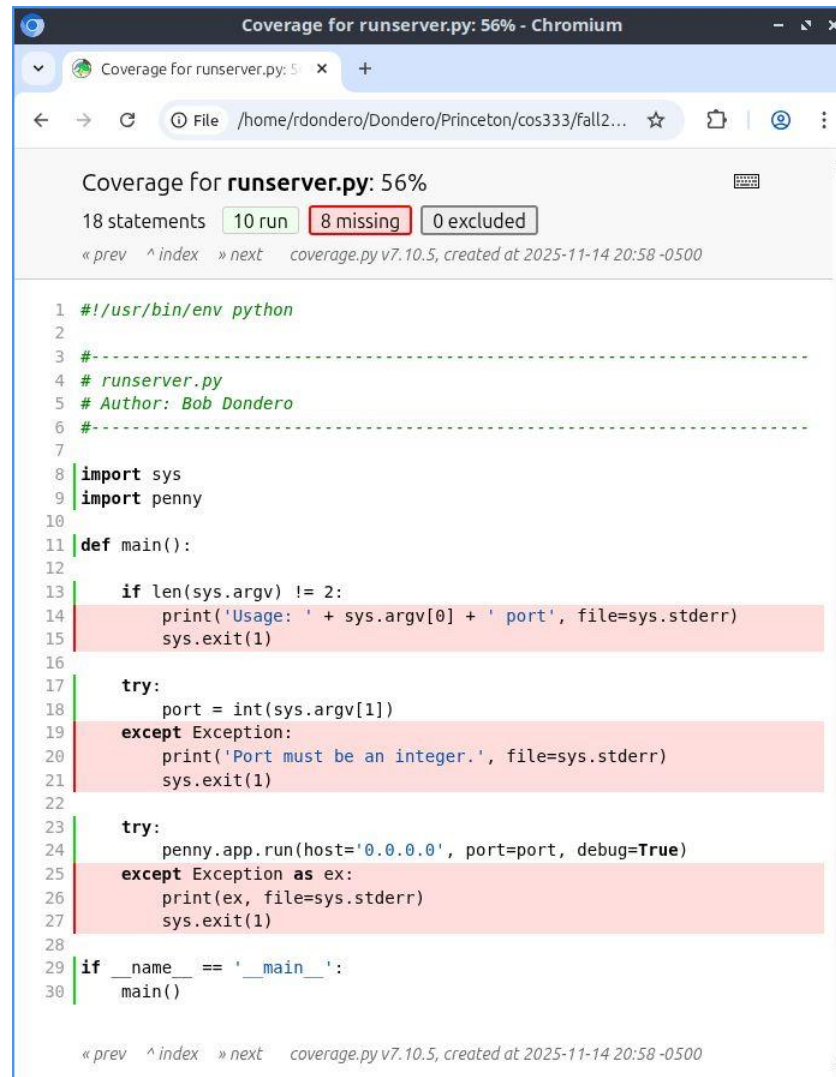
Coverage for penny.py: 95%

21 statements 20 run 1 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

```
1  #!/usr/bin/env python
2
3  #-----
4  # penny.py
5  # Author: Bob Dondero
6  #-----
7
8  import json
9  import flask
10 import database
11
12 #-----
13
14 app = flask.Flask(__name__)
15
16 #-----
17
18 @app.route('/', methods=['GET'])
19 @app.route('/index', methods=['GET'])
20 def index():
21
22     return flask.send_file('index.html')
23
24 #-----
25
26 @app.route('/searchresults', methods=['GET'])
27 def search_results():
28
29     author = flask.request.args.get('author')
30     if author is None:
31         author = ''
32     author = author.strip()
33
34     if author == '':
```

# Statement/Coverage Testing



Coverage for runserver.py: 56% - Chromium

Coverage for runserver.py: 56%

18 statements 10 run 8 missing 0 excluded

« prev ^ index » next coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

```
1  #!/usr/bin/env python
2
3  # -----
4  # runserver.py
5  # Author: Bob Dondero
6  # -----
7
8  import sys
9  import penny
10
11 def main():
12
13     if len(sys.argv) != 2:
14         print('Usage: ' + sys.argv[0] + ' port', file=sys.stderr)
15         sys.exit(1)
16
17     try:
18         port = int(sys.argv[1])
19     except Exception:
20         print('Port must be an integer.', file=sys.stderr)
21         sys.exit(1)
22
23     try:
24         penny.app.run(host='0.0.0.0', port=port, debug=True)
25     except Exception as ex:
26         print(ex, file=sys.stderr)
27         sys.exit(1)
28
29 if __name__ == '__main__':
30     main()
```

« prev ^ index » next coverage.py v7.10.5, created at 2025-11-14 20:58 -0500

# 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