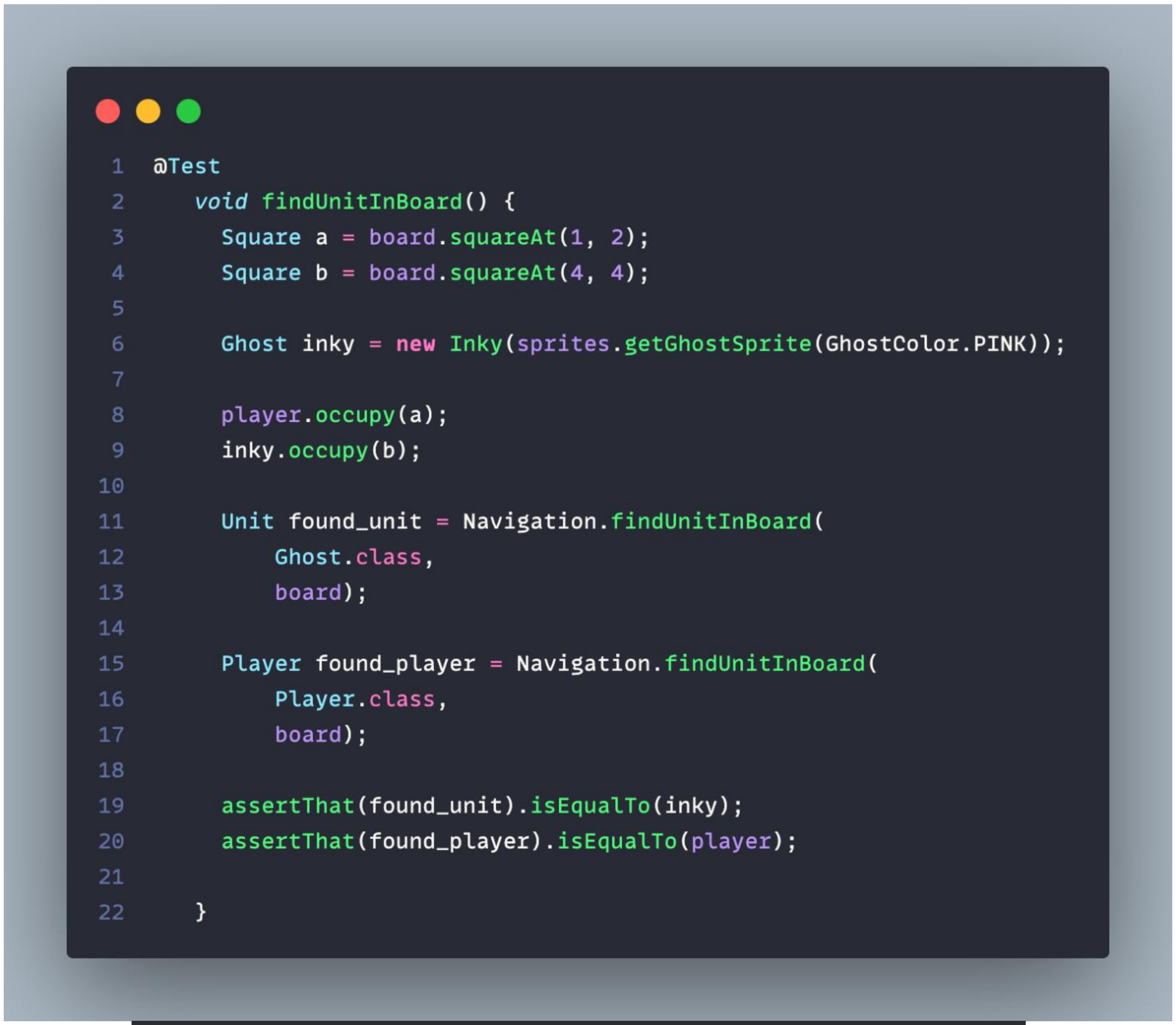
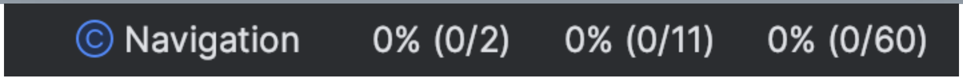
Brandon Cannon

https://github.com/heyitsmass/CS472/tree/jpacman_tests

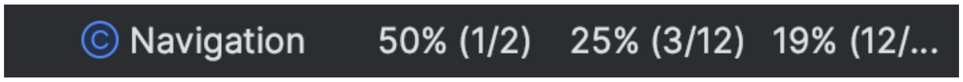
Task 2.1:



```java
@Test
  void findUnitInBoard() {
    Square a = board.squareAt(1, 2);
    Square b = board.squareAt(4, 4);

    Ghost inky = new Inky(sprites.getGhostSprite(GhostColor.PINK));

    player.occupy(a);
    inky.occupy(b);

    Unit found_unit = Navigation.findUnitInBoard(
        Ghost.class,
        board);

    Player found_player = Navigation.findUnitInBoard(
        Player.class,
        board);

    assertThat(found_unit).isEqualTo(inky);
    assertThat(found_player).isEqualTo(player);

  }
```

Before: 

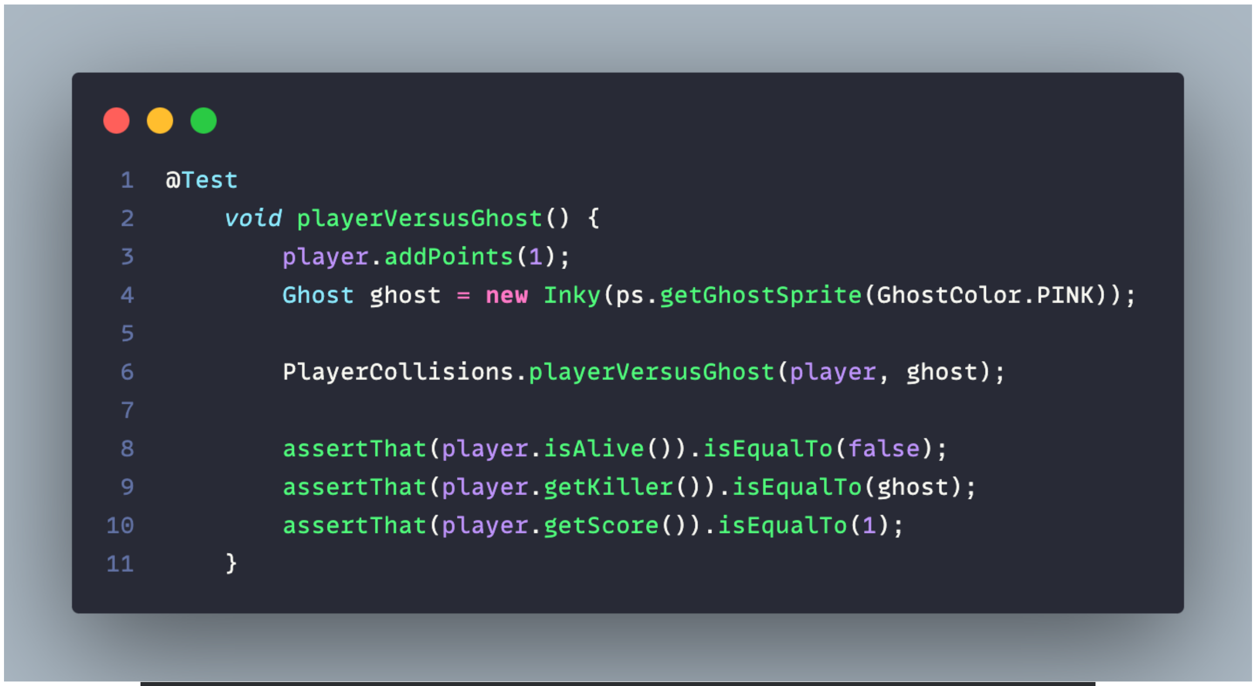
```
Navigation    0% (0/2)    0% (0/11)    0% (0/60)
```

After: 

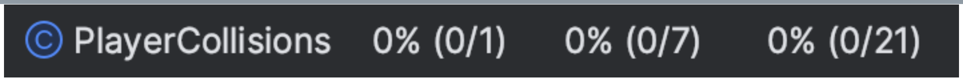
```
Navigation    50% (1/2)    25% (3/12)    19% (12/...
```

IntelliJ Seemingly doubled all-of its numbers as JaCoCo only reported 1 Class, 6 Methods and 49 lines. Compared to IntelliJ JaCoCo reported a 94% coverage percentage for the file overall. For this specific method it had a reported 93% coverage in JaCoCo.

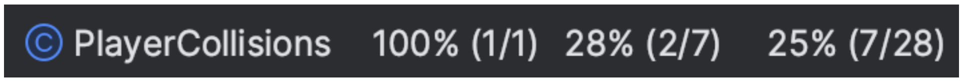
Brandon Cannon



```java
@Test
    void playerVersusGhost() {
        player.addPoints(1);
        Ghost ghost = new Inky(ps.getGhostSprite(GhostColor.PINK));

        PlayerCollisions.playerVersusGhost(player, ghost);

        assertThat(player.isAlive()).isEqualTo(false);
        assertThat(player.getKiller()).isEqualTo(ghost);
        assertThat(player.getScore()).isEqualTo(1);
    }
```

Before: 

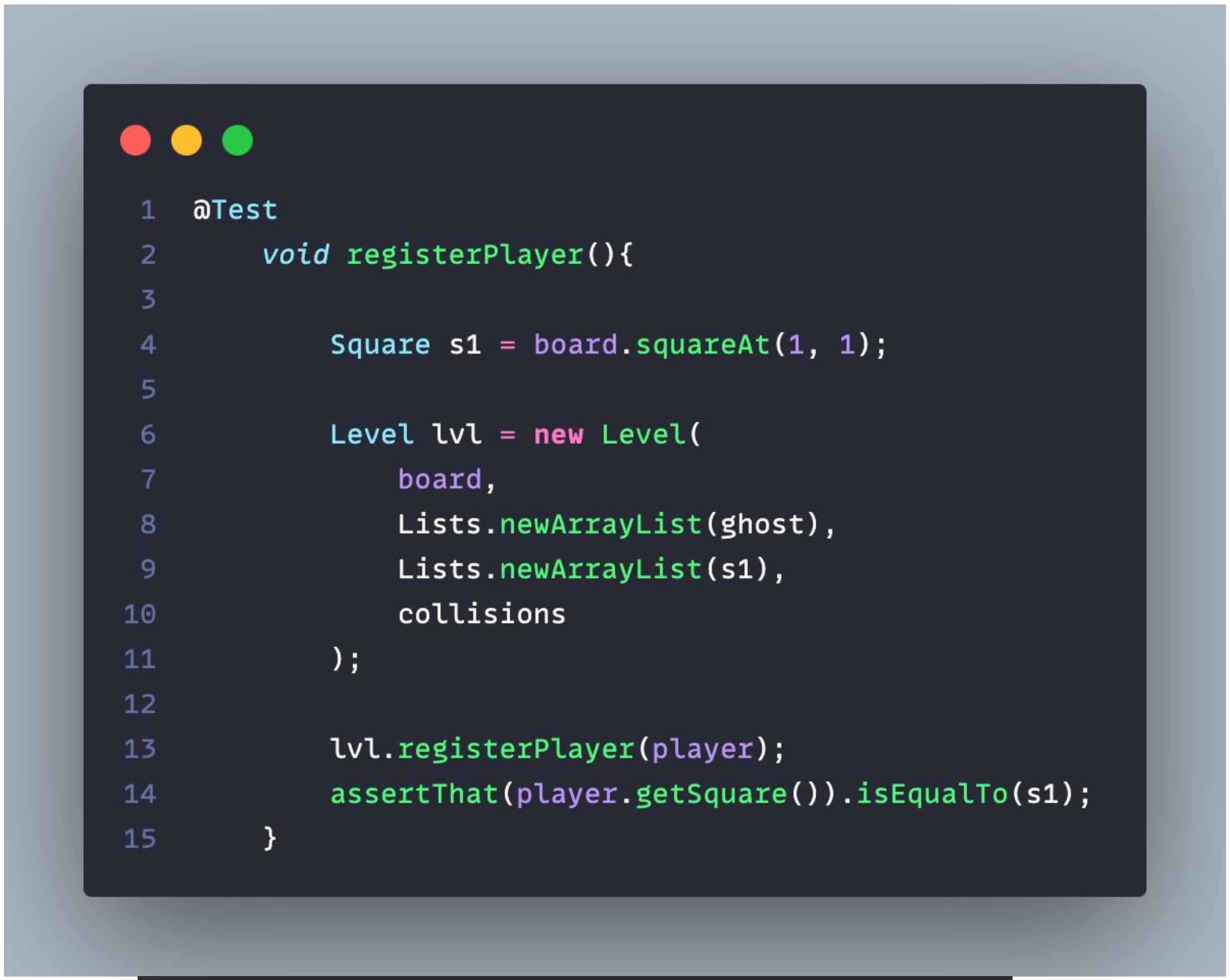
PlayerCollisions	0% (0/1)	0% (0/7)	0% (0/21)

After: 

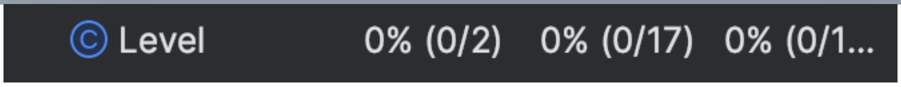
PlayerCollisions	100% (1/1)	28% (2/7)	25% (7/28)

JaCoCo reported the same number of classes, methods, and lines for this file however indicated only 1 method was missing tests and the file had a total of 75% coverage. The specific method was 100% covered in JaCoCo.

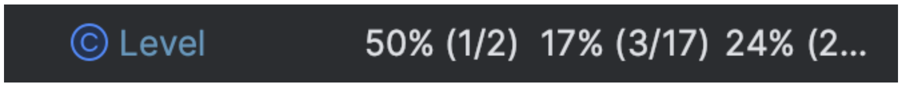
Brandon Cannon



```java
@Test
    void registerPlayer(){

        Square s1 = board.squareAt(1, 1);

        Level lvl = new Level(
            board,
            Lists.newArrayList(ghost),
            Lists.newArrayList(s1),
            collisions
        );

        lvl.registerPlayer(player);
        assertThat(player.getSquare()).isEqualTo(s1);
    }
```

Before: 

```
Level    0% (0/2)    0% (0/17)    0% (0/1...
```

After: 

```
Level    50% (1/2)   17% (3/17)   24% (2...
```

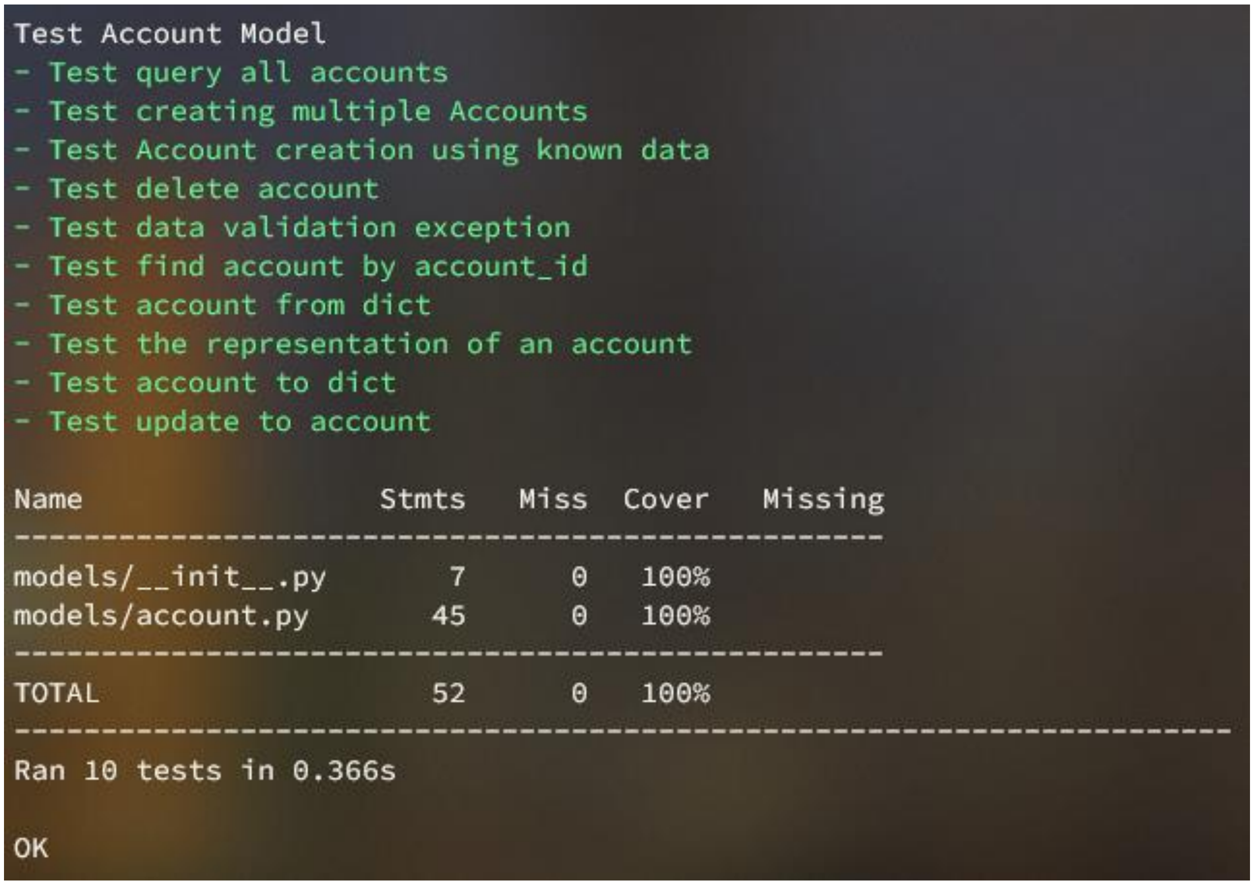
JaCoCo reported that there were 15 methods and only one untested compared to IntelliJ. IntelliJ similarly reported that only 24% of the lines were covered compared to JaCoCo reporting only 6 of 105 lines weren't tested. IntelliJ reported two classes total, only one of which was covered whereas JaCoCo reported only 1 class and was covered.

Task 3:

Brandon Cannon

Overall, the coverage results from JaCoCo were much different than IntelliJ. I'm not sure why other than assuming how each parses the test results. It could be because of the dynamic branch evaluation that checks the method directly and operates differently than IntelliJ. The source code visualization was helpful but initially confusing and tough to decipher as it's unclear if the highlighted regions are tested, covered, untested, errored, etc.… I preferred the IntelliJ visualization for quick evaluation where an in-depth perspective isn't necessary. However, for detailed analysis of uncovered branches, JaCoCo provides a more rich and deep interpretation of the test data.

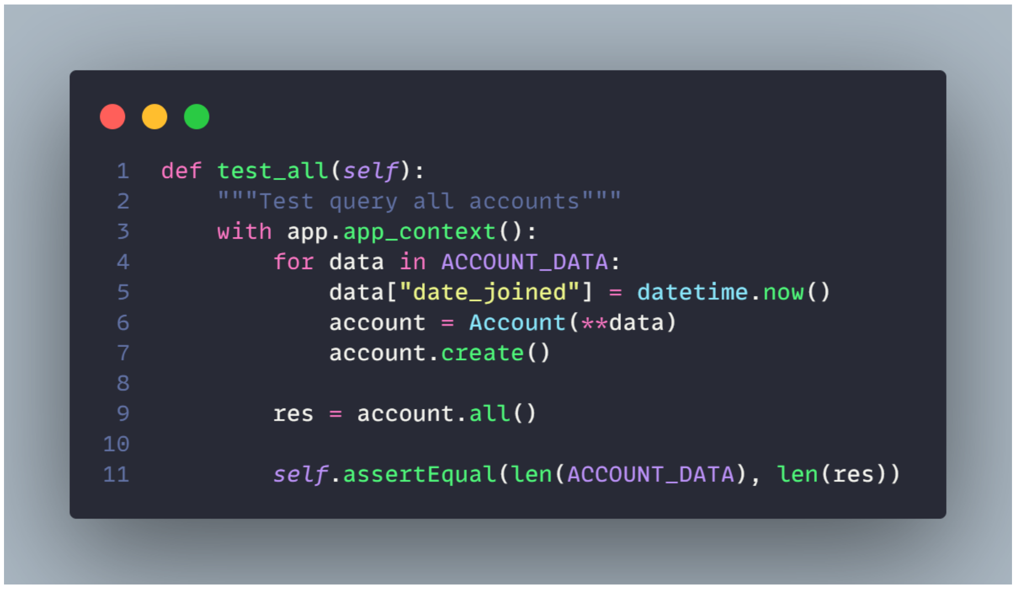
Task 4: *I've excluded the provided tests.*



```
Test Account Model
- Test query all accounts
- Test creating multiple Accounts
- Test Account creation using known data
- Test delete account
- Test data validation exception
- Test find account by account_id
- Test account from dict
- Test the representation of an account
- Test account to dict
- Test update to account

Name                     Stmts   Miss  Cover   Missing
------------------------------------------------------
models/__init__.py           7      0   100%
models/account.py           45      0   100%
------------------------------------------------------
TOTAL                       52      0   100%
----------------------------------------------------------------------
Ran 10 tests in 0.366s

OK
```

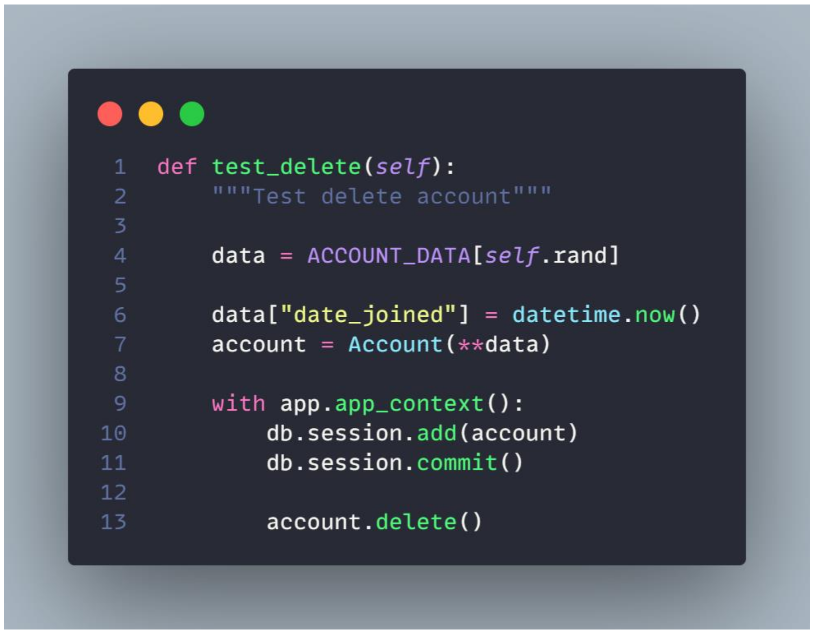
test_all()

Brandon Cannon



```python
def test_all(self):
    """Test query all accounts"""
    with app.app_context():
        for data in ACCOUNT_DATA:
            data["date_joined"] = datetime.now()
            account = Account(**data)
            account.create()

        res = account.all()

        self.assertEqual(len(ACCOUNT_DATA), len(res))
```

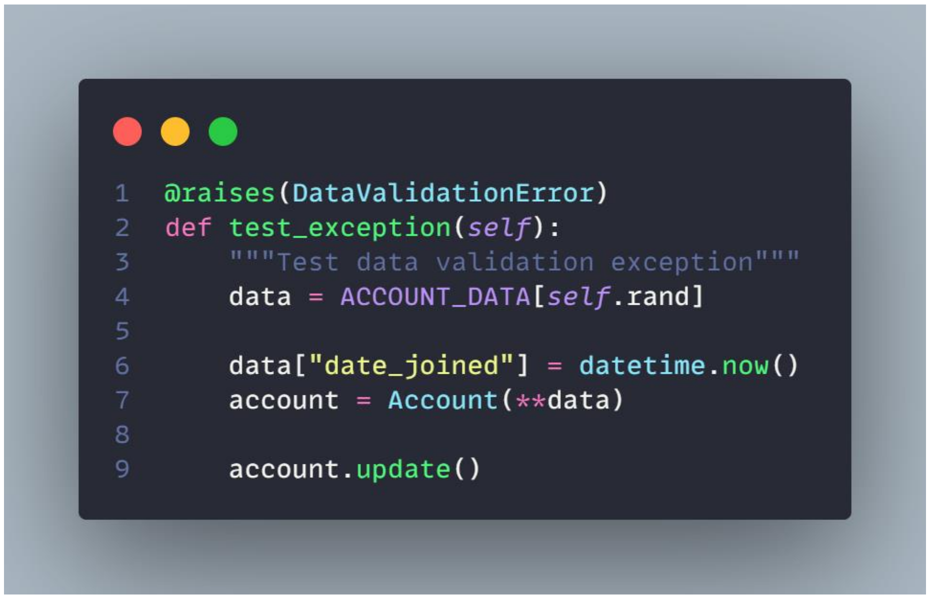
test_delete()



```python
def test_delete(self):
    """Test delete account"""

    data = ACCOUNT_DATA[self.rand]

    data["date_joined"] = datetime.now()
    account = Account(**data)

    with app.app_context():
        db.session.add(account)
        db.session.commit()

        account.delete()
```

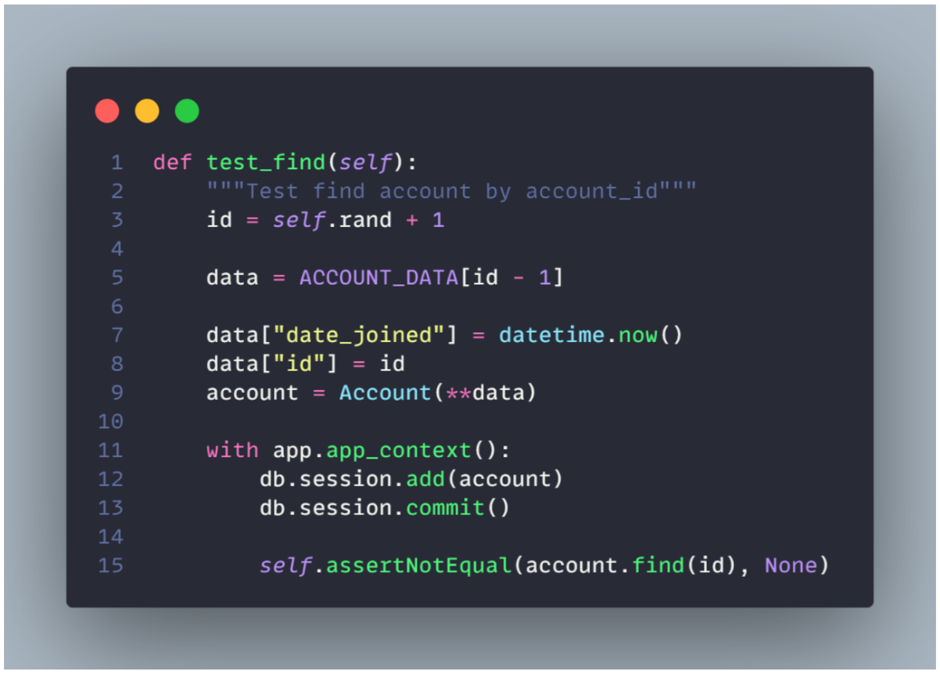
test_exception()

Brandon Cannon



```python
@raises(DataValidationError)
def test_exception(self):
    """Test data validation exception"""
    data = ACCOUNT_DATA[self.rand]

    data["date_joined"] = datetime.now()
    account = Account(**data)

    account.update()
```

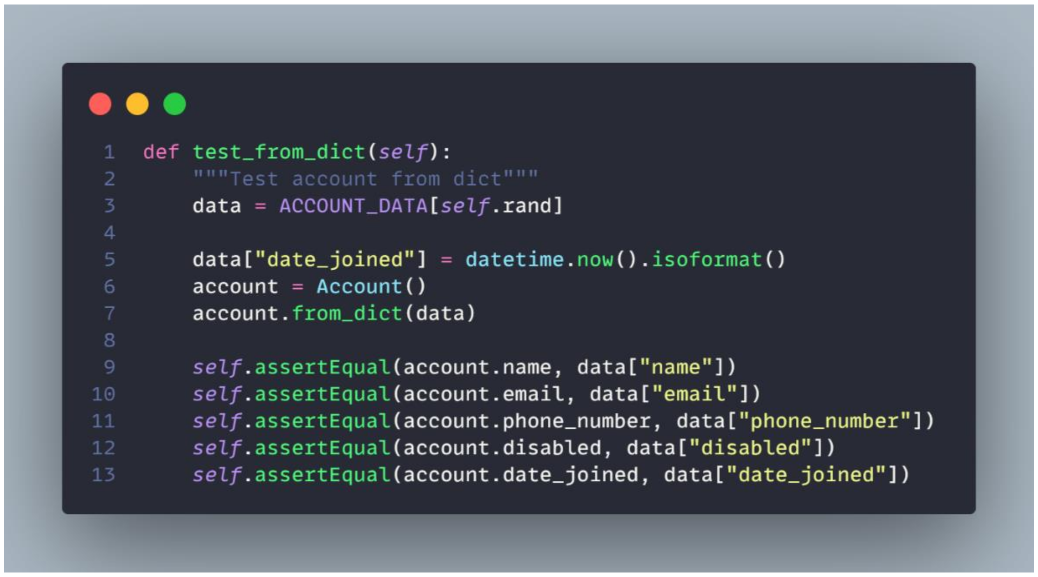
test_find()



```python
def test_find(self):
    """Test find account by account_id"""
    id = self.rand + 1

    data = ACCOUNT_DATA[id - 1]

    data["date_joined"] = datetime.now()
    data["id"] = id
    account = Account(**data)

    with app.app_context():
        db.session.add(account)
        db.session.commit()

        self.assertNotEqual(account.find(id), None)
```

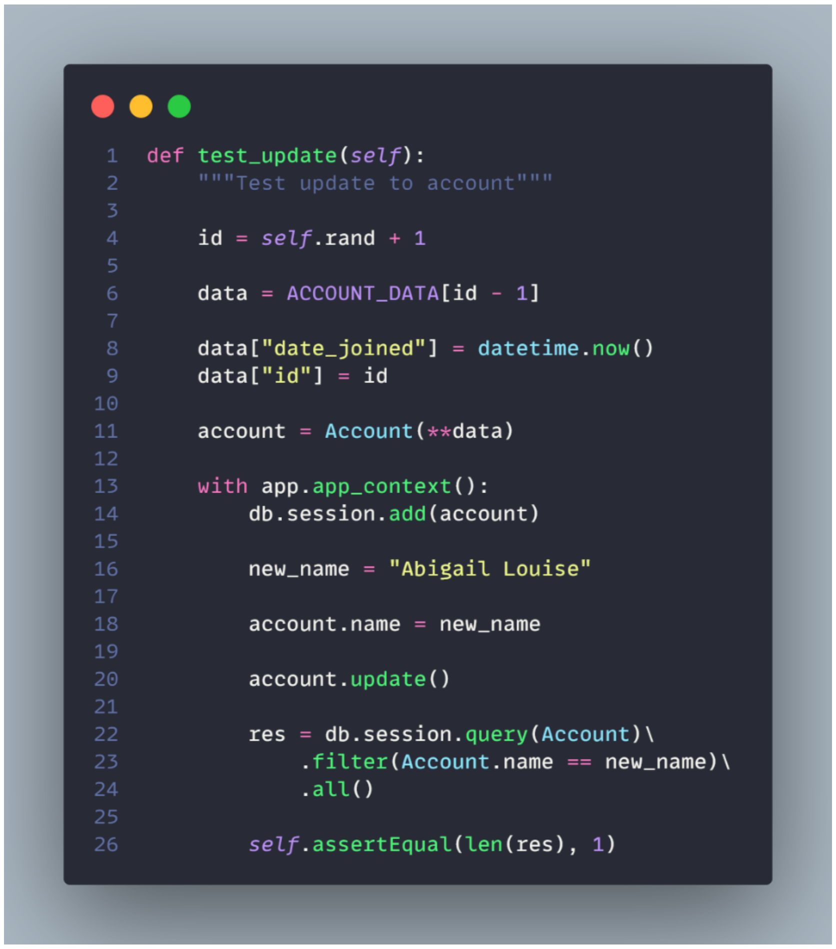
test_from_dict()

Brandon Cannon



```python
def test_from_dict(self):
    """Test account from dict"""
    data = ACCOUNT_DATA[self.rand]

    data["date_joined"] = datetime.now().isoformat()
    account = Account()
    account.from_dict(data)

    self.assertEqual(account.name, data["name"])
    self.assertEqual(account.email, data["email"])
    self.assertEqual(account.phone_number, data["phone_number"])
    self.assertEqual(account.disabled, data["disabled"])
    self.assertEqual(account.date_joined, data["date_joined"])
```

test_update()

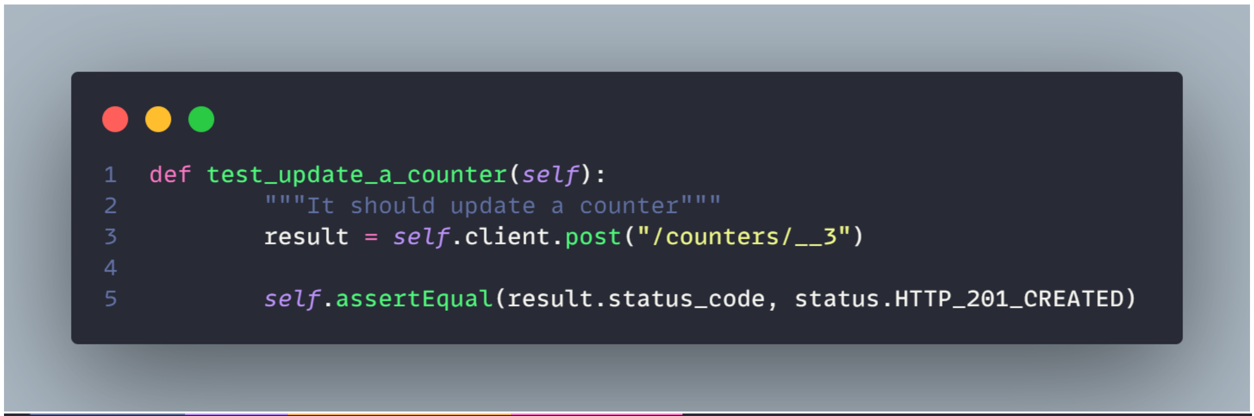


```python
def test_update(self):
    """Test update to account"""

    id = self.rand + 1

    data = ACCOUNT_DATA[id - 1]

    data["date_joined"] = datetime.now()
    data["id"] = id

    account = Account(**data)

    with app.app_context():
        db.session.add(account)

        new_name = "Abigail Louise"

        account.name = new_name

        account.update()

        res = db.session.query(Account)\
            .filter(Account.name == new_name)\
            .all()

        self.assertEqual(len(res), 1)
```

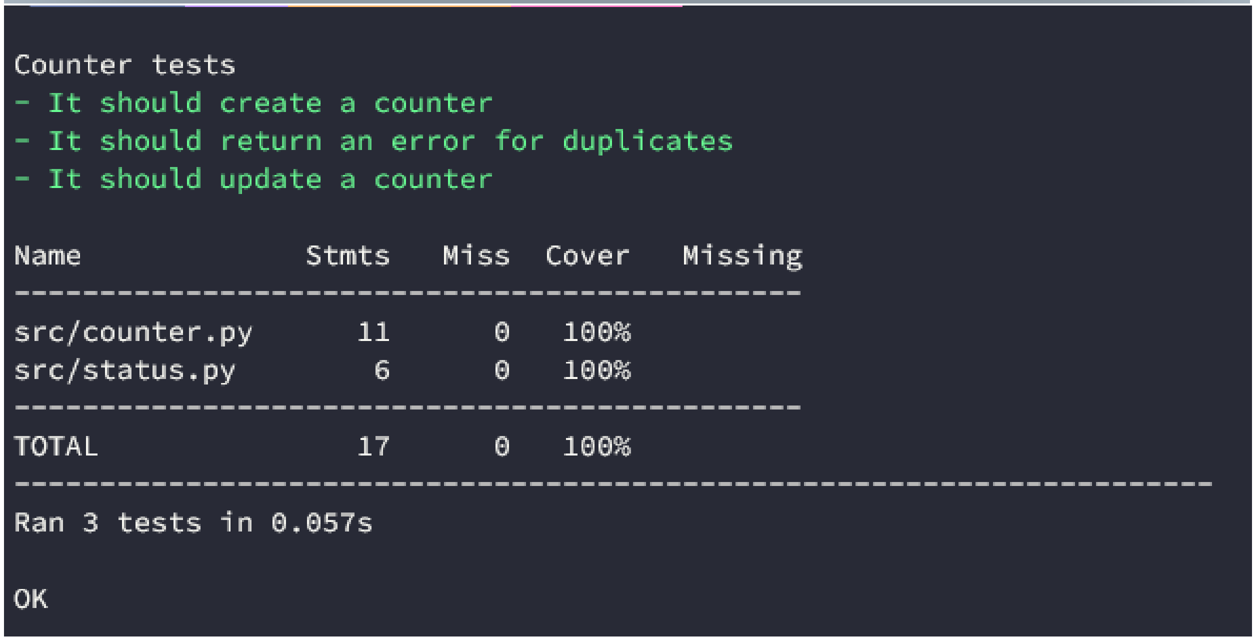
Brandon Cannon

Task 5:

I did these steps out of order before realizing they needed to be inserted chronologically. I tried to reconstruct them as best I could, but they may be out of order.

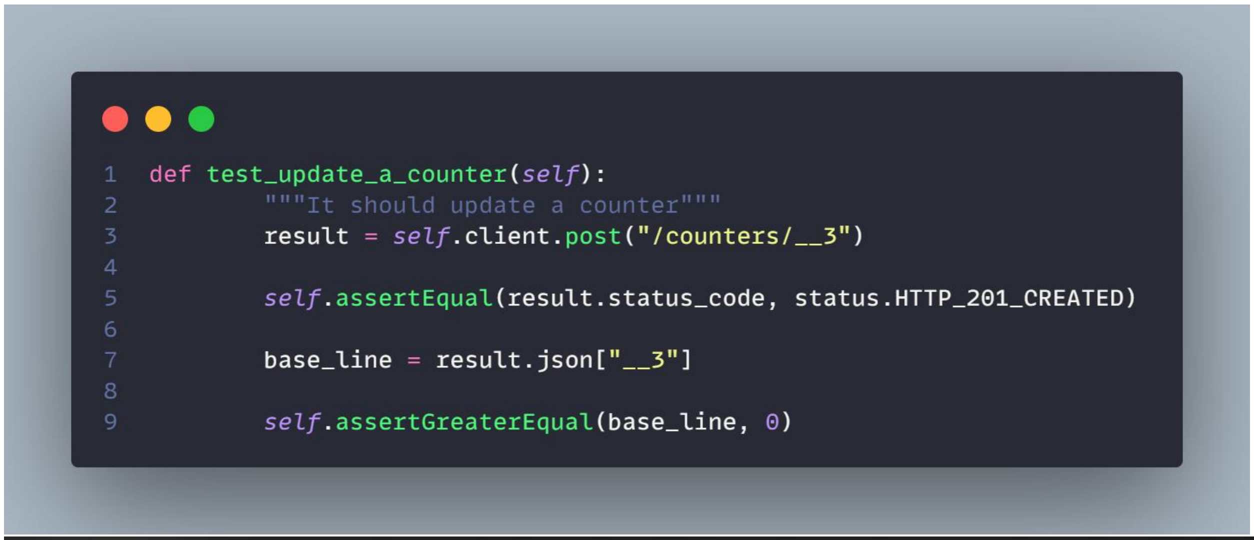


```python
def test_update_a_counter(self):
        """It should update a counter"""
        result = self.client.post("/counters/__3")

        self.assertEqual(result.status_code, status.HTTP_201_CREATED)
```

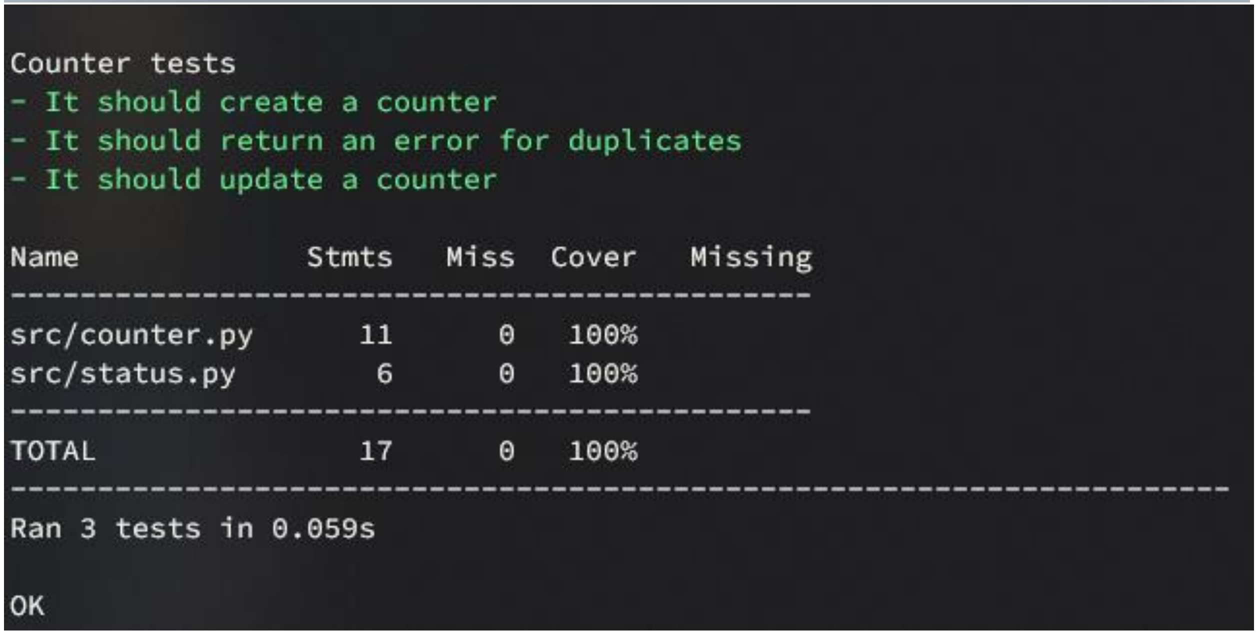


```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
src/counter.py         11      0   100%
src/status.py           6      0   100%
-------------------------------------------------
TOTAL                  17      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.057s

OK
```

Brandon Cannon

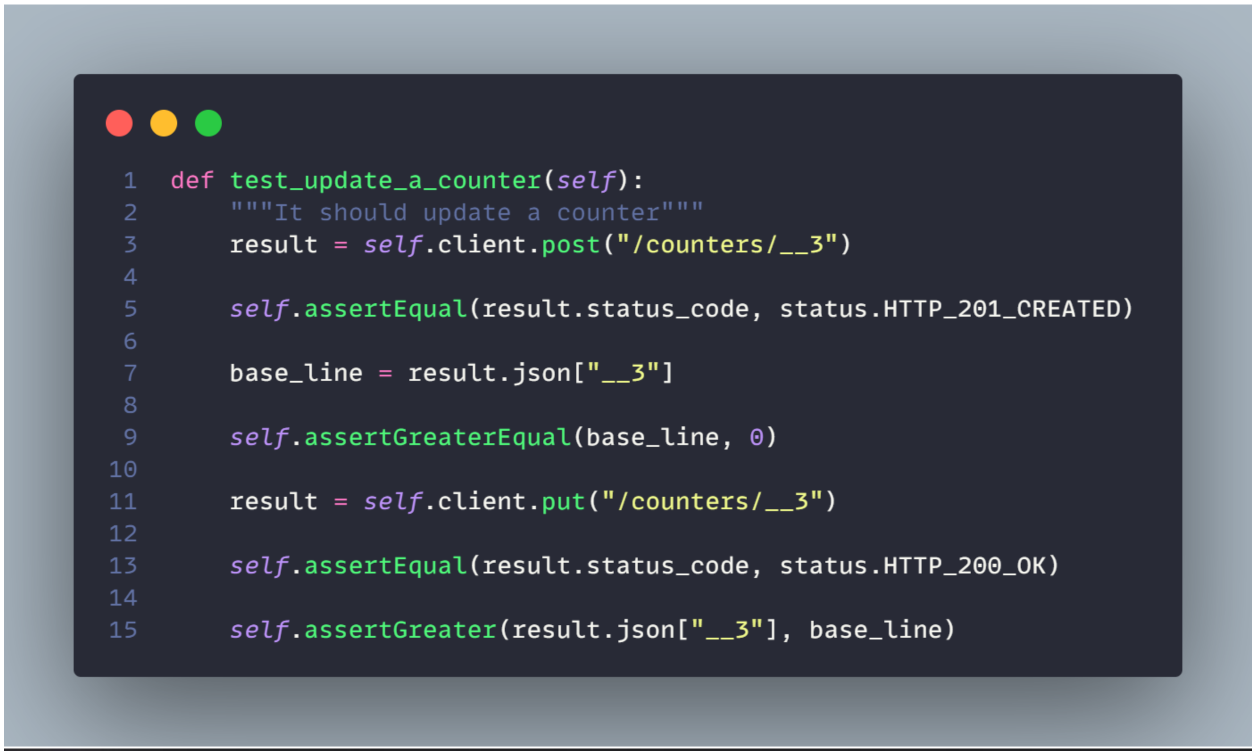


```
def test_update_a_counter(self):
        """It should update a counter"""
        result = self.client.post("/counters/__3")

        self.assertEqual(result.status_code, status.HTTP_201_CREATED)

        base_line = result.json["__3"]

        self.assertGreaterEqual(base_line, 0)
```

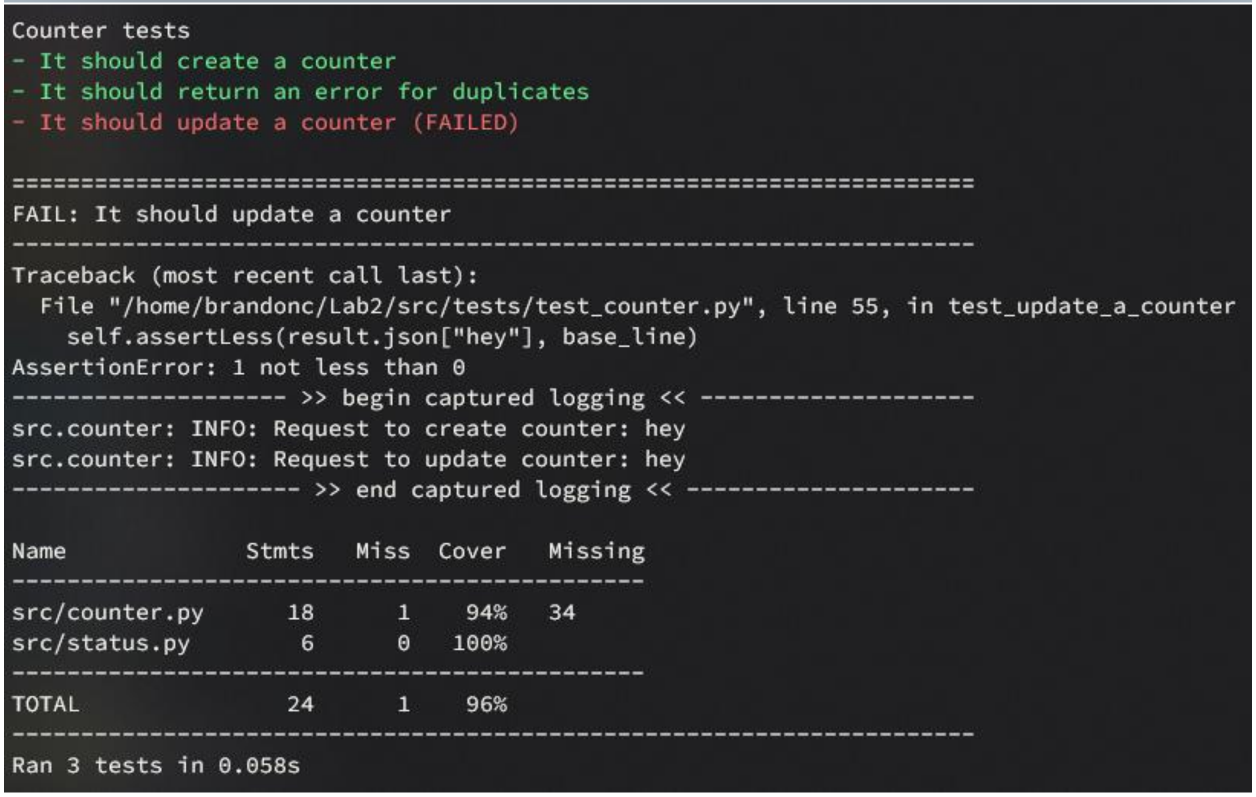


```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter

Name                 Stmts   Miss  Cover   Missing
----------------------------------------------
src/counter.py          11      0   100%
src/status.py            6      0   100%
----------------------------------------------
TOTAL                   17      0   100%
----------------------------------------------------------------------
Ran 3 tests in 0.059s

OK
```

Brandon Cannon

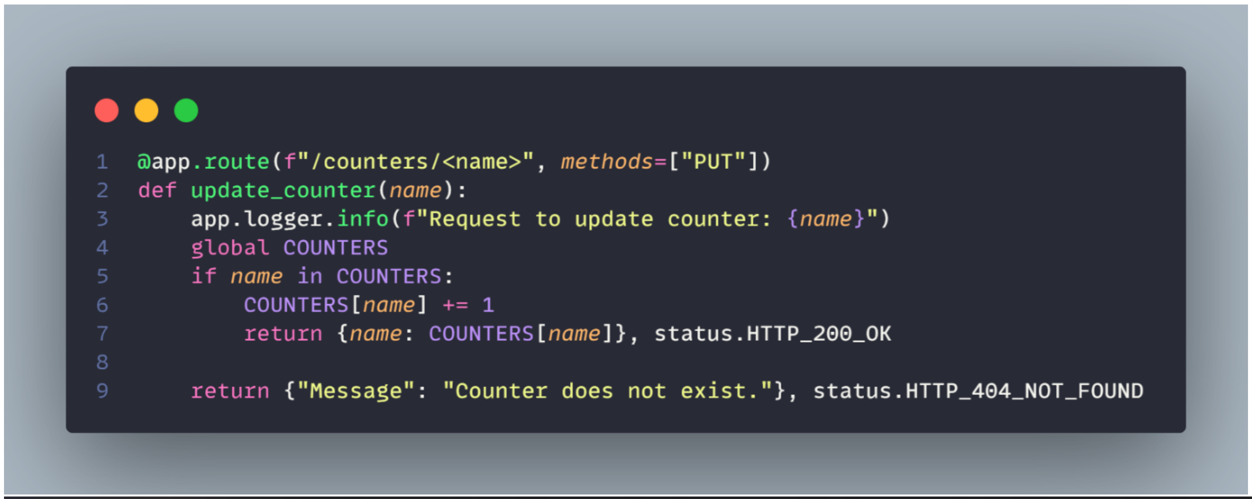


```python
def test_update_a_counter(self):
    """It should update a counter"""
    result = self.client.post("/counters/__3")

    self.assertEqual(result.status_code, status.HTTP_201_CREATED)

    base_line = result.json["__3"]

    self.assertGreaterEqual(base_line, 0)

    result = self.client.put("/counters/__3")

    self.assertEqual(result.status_code, status.HTTP_200_OK)

    self.assertGreater(result.json["__3"], base_line)
```

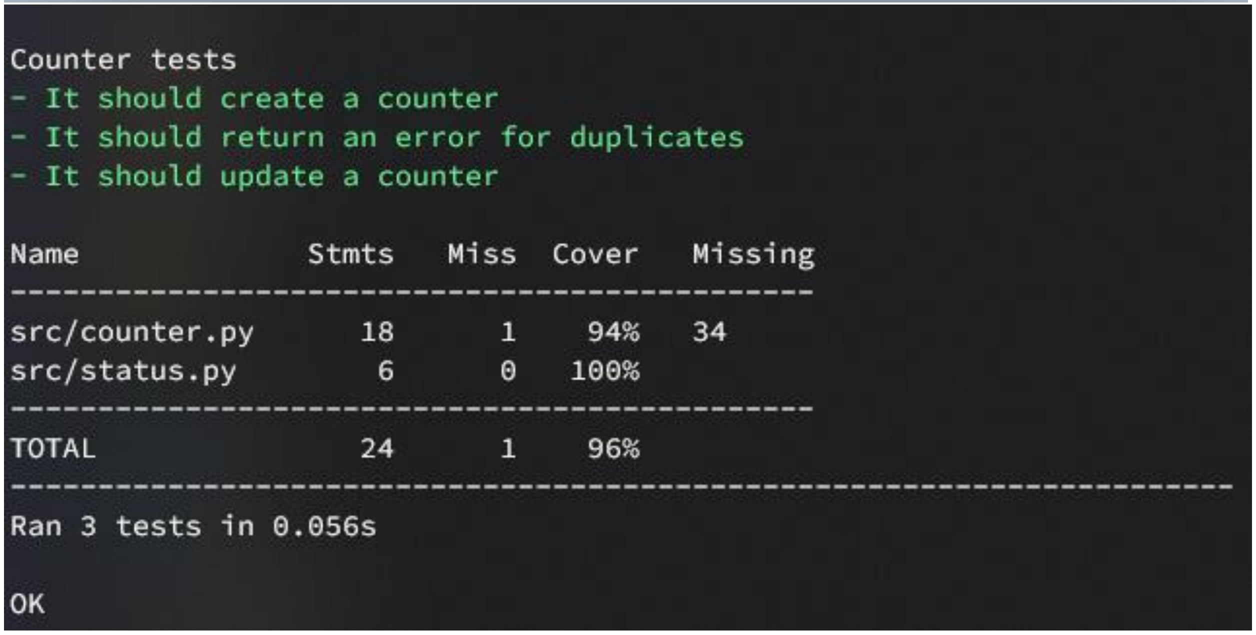


```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter (FAILED)

======================================================================
FAIL: It should update a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/brandonc/Lab2/src/tests/test_counter.py", line 55, in test_update_a_counter
    self.assertLess(result.json["hey"], base_line)
AssertionError: 1 not less than 0
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create counter: hey
src.counter: INFO: Request to update counter: hey
--------------------- >> end captured logging << ---------------------

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
src/counter.py          18      1    94%   34
src/status.py            6      0   100%
--------------------------------------------------
TOTAL                   24      1    96%
----------------------------------------------------------------------
Ran 3 tests in 0.058s
```

Brandon Cannon

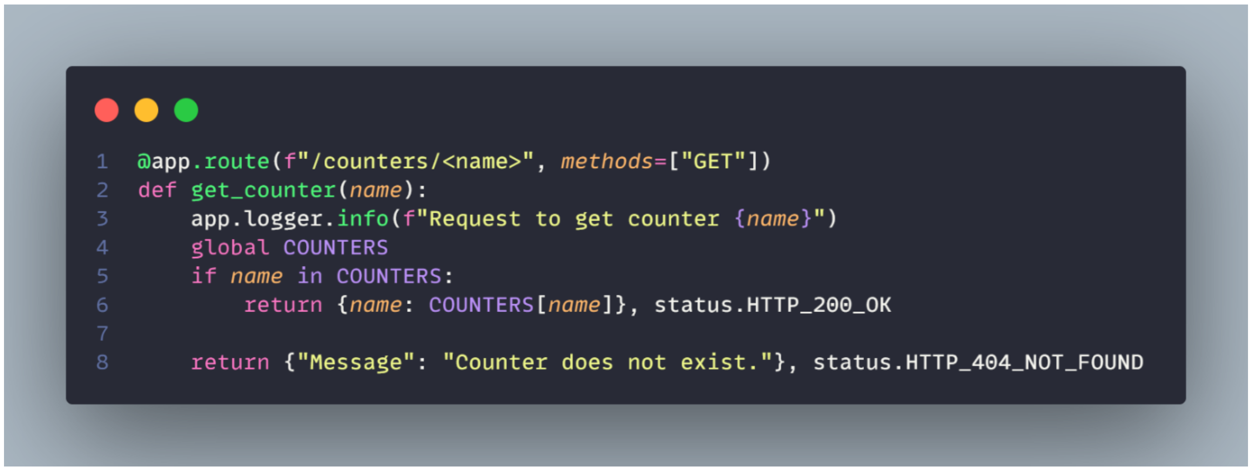


```python
@app.route(f"/counters/<name>", methods=["PUT"])
def update_counter(name):
    app.logger.info(f"Request to update counter: {name}")
    global COUNTERS
    if name in COUNTERS:
        COUNTERS[name] += 1
        return {name: COUNTERS[name]}, status.HTTP_200_OK

    return {"Message": "Counter does not exist."}, status.HTTP_404_NOT_FOUND
```



```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should update a counter

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
src/counter.py         18      1    94%   34
src/status.py           6      0   100%
-------------------------------------------------
TOTAL                  24      1    96%
----------------------------------------------------------------------
Ran 3 tests in 0.056s

OK
```

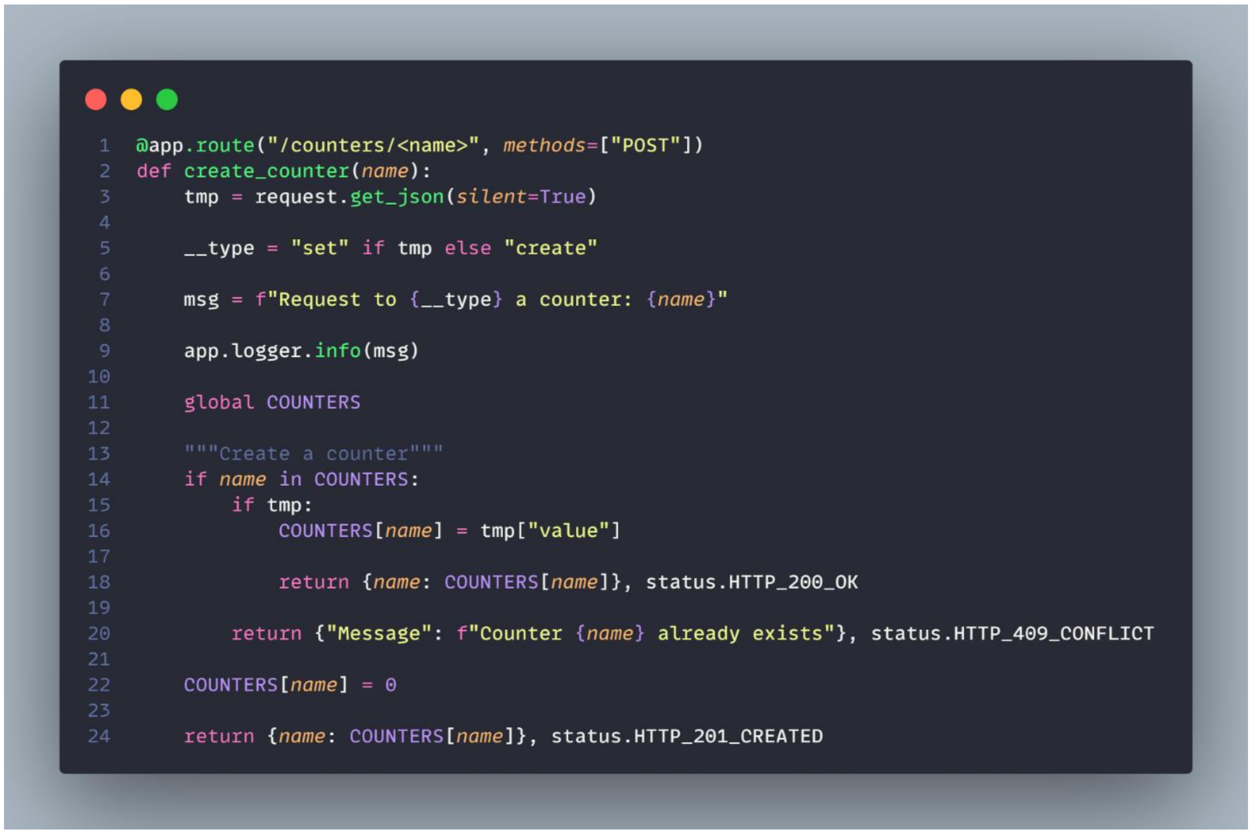
Construct the GET Method



```python
@app.route(f"/counters/<name>", methods=["GET"])
def get_counter(name):
    app.logger.info(f"Request to get counter {name}")
    global COUNTERS
    if name in COUNTERS:
        return {name: COUNTERS[name]}, status.HTTP_200_OK

    return {"Message": "Counter does not exist."}, status.HTTP_404_NOT_FOUND
```

Brandon Cannon

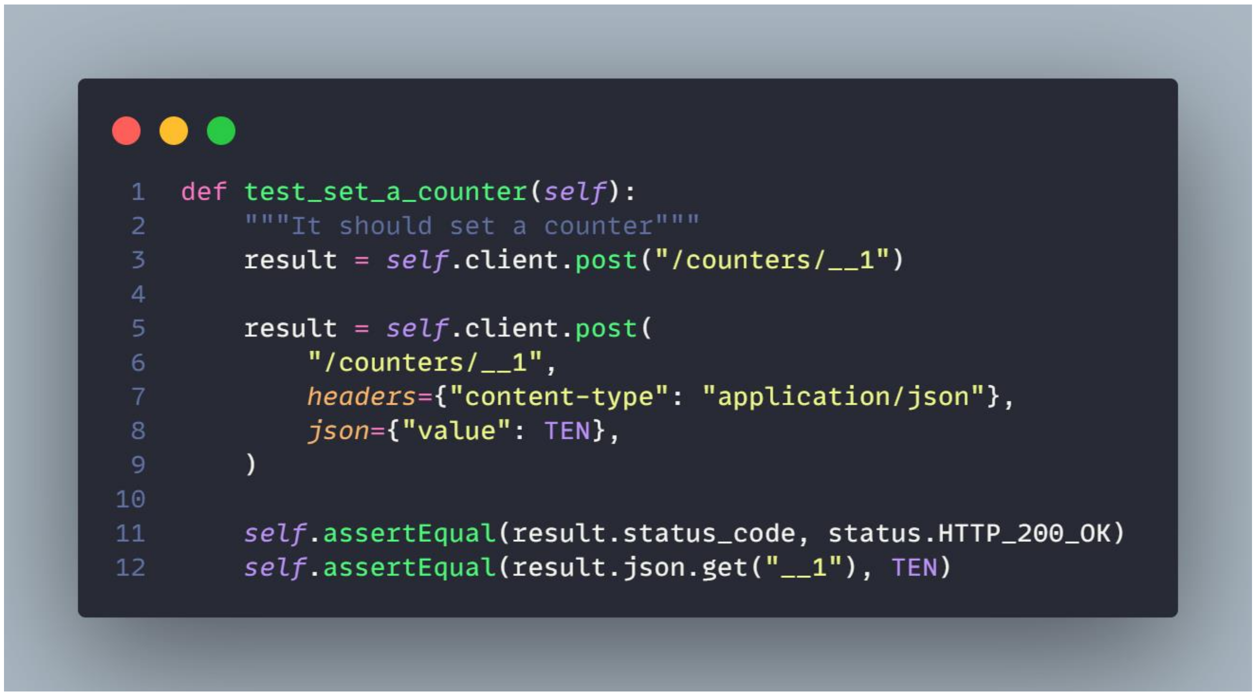
I created a method to set the counter to make it slightly more interesting and reworked the original POST method slightly. **The function now checks for any JSON sent in the body and modifies the response message for the logger, updates the value if one is provided or creates a new one. Otherwise, a duplicate error is returned.**



```python
@app.route("/counters/<name>", methods=["POST"])
def create_counter(name):
    tmp = request.get_json(silent=True)

    __type = "set" if tmp else "create"

    msg = f"Request to {__type} a counter: {name}"

    app.logger.info(msg)

    global COUNTERS

    """Create a counter"""
    if name in COUNTERS:
        if tmp:
            COUNTERS[name] = tmp["value"]

            return {name: COUNTERS[name]}, status.HTTP_200_OK

        return {"Message": f"Counter {name} already exists"}, status.HTTP_409_CONFLICT

    COUNTERS[name] = 0

    return {name: COUNTERS[name]}, status.HTTP_201_CREATED
```

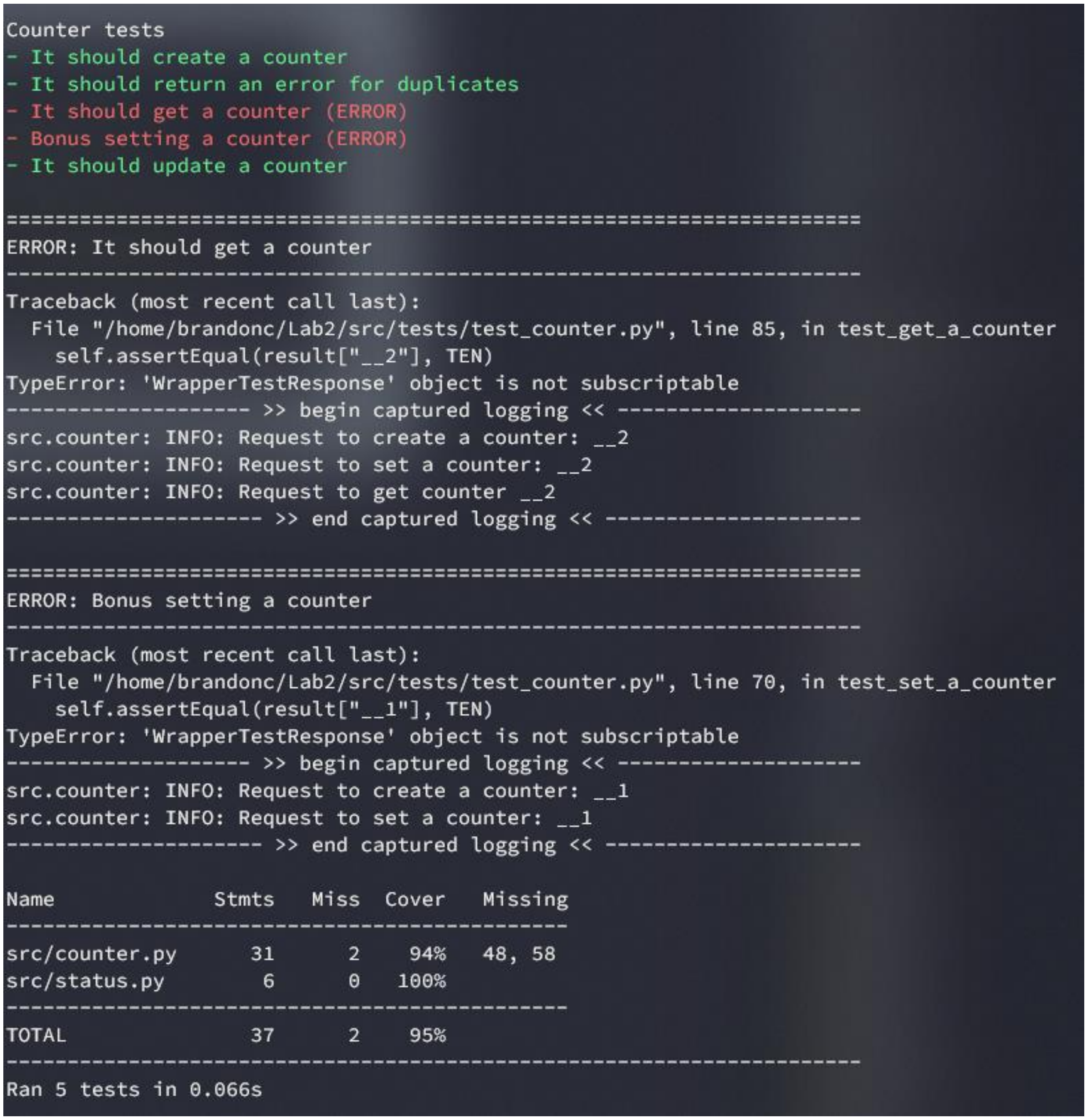
Brandon Cannon

Create the tests for the set method



```python
def test_set_a_counter(self):
    """It should set a counter"""
    result = self.client.post("/counters/__1")

    result = self.client.post(
        "/counters/__1",
        headers={"content-type": "application/json"},
        json={"value": TEN},
    )

    self.assertEqual(result.status_code, status.HTTP_200_OK)
    self.assertEqual(result.json.get("__1"), TEN)
```

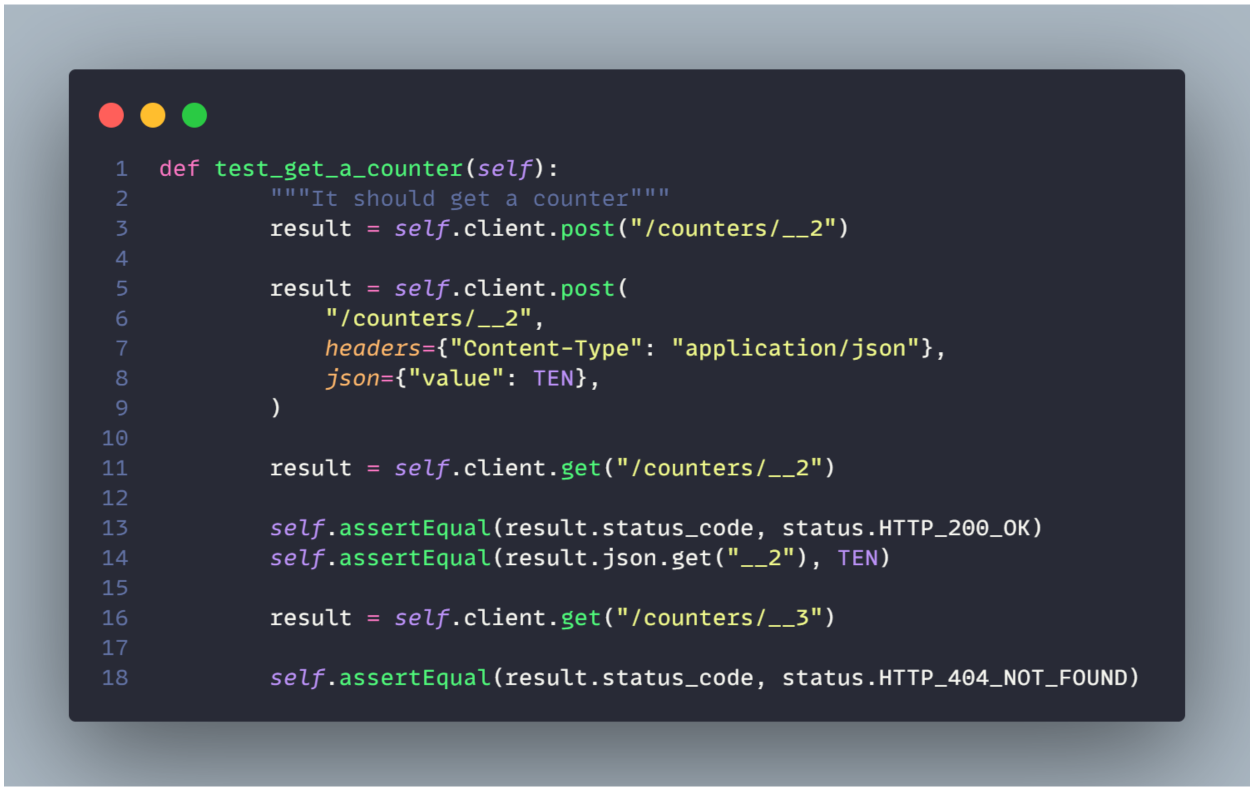
Brandon Cannon



```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should get a counter (ERROR)
- Bonus setting a counter (ERROR)
- It should update a counter

======================================================================
ERROR: It should get a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/brandonc/Lab2/src/tests/test_counter.py", line 85, in test_get_a_counter
    self.assertEqual(result["__2"], TEN)
TypeError: 'WrapperTestResponse' object is not subscriptable
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create a counter: __2
src.counter: INFO: Request to set a counter: __2
src.counter: INFO: Request to get counter __2
--------------------- >> end captured logging << ---------------------

======================================================================
ERROR: Bonus setting a counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/brandonc/Lab2/src/tests/test_counter.py", line 70, in test_set_a_counter
    self.assertEqual(result["__1"], TEN)
TypeError: 'WrapperTestResponse' object is not subscriptable
-------------------- >> begin captured logging << --------------------
src.counter: INFO: Request to create a counter: __1
src.counter: INFO: Request to set a counter: __1
--------------------- >> end captured logging << ---------------------

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       31      2    94%   48, 58
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                37      2    95%
----------------------------------------------------------------------
Ran 5 tests in 0.066s
```

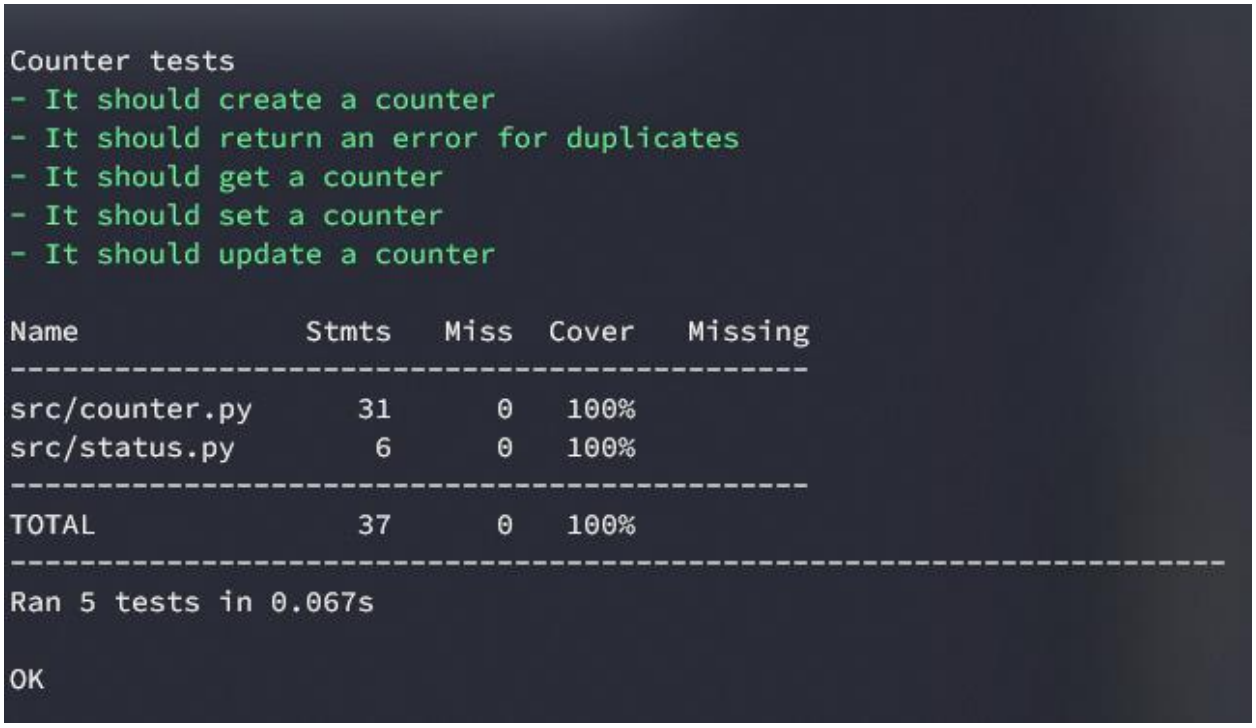
Fix the bugs and create the GET method

Brandon Cannon



```
def test_get_a_counter(self):
        """It should get a counter"""
        result = self.client.post("/counters/__2")

        result = self.client.post(
            "/counters/__2",
            headers={"Content-Type": "application/json"},
            json={"value": TEN},
        )

        result = self.client.get("/counters/__2")

        self.assertEqual(result.status_code, status.HTTP_200_OK)
        self.assertEqual(result.json.get("__2"), TEN)

        result = self.client.get("/counters/__3")

        self.assertEqual(result.status_code, status.HTTP_404_NOT_FOUND)
```

Run the tests.



```
Counter tests
- It should create a counter
- It should return an error for duplicates
- It should get a counter
- It should set a counter
- It should update a counter

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py       31      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                37      0   100%
----------------------------------------------------------------------
Ran 5 tests in 0.067s

OK
```