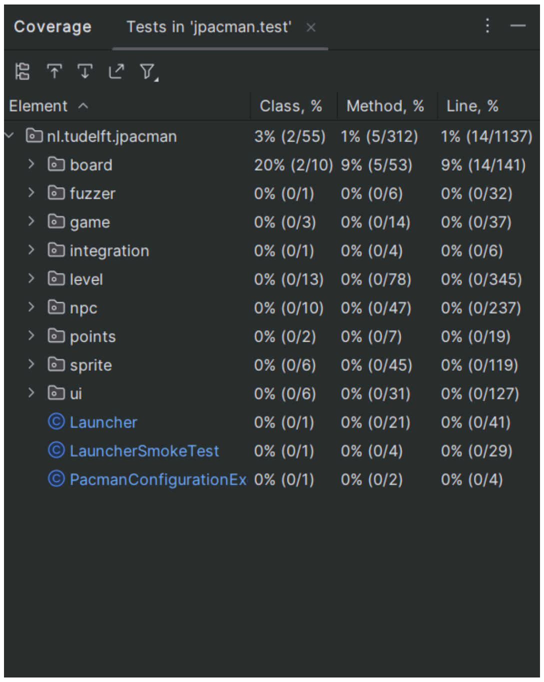
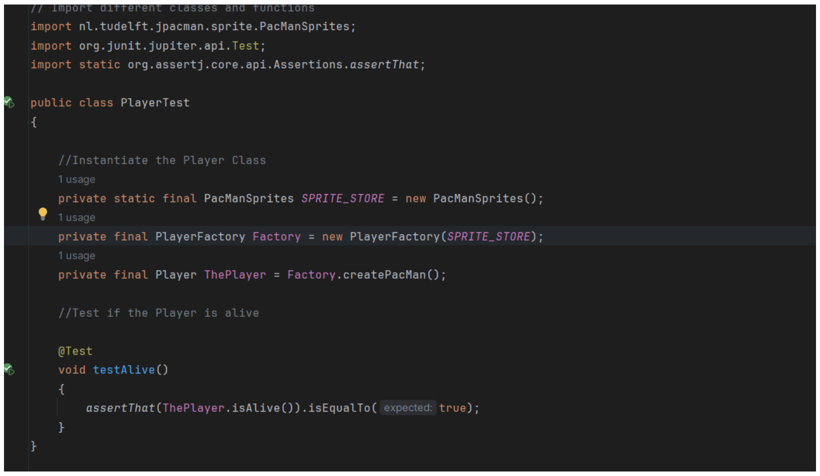
Forked jpacman: https://github.com/tnishamon/2DRogueLikeUnityGame/tree/jpacman

Task 1



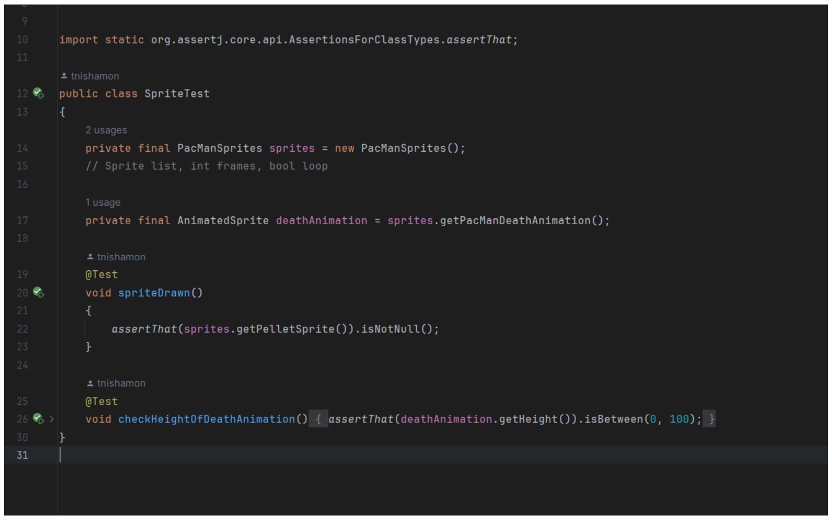
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	0% (0/13)	0% (0/78)	0% (0/345)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	0% (0/6)	0% (0/45)	0% (0/119)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationEx	0% (0/1)	0% (0/2)	0% (0/4)



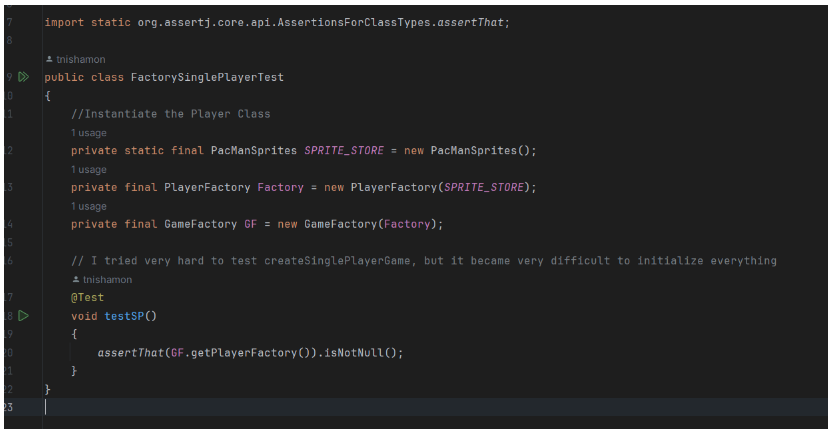
```java
// import different classes and functions
import nl.tudelft.jpacman.sprite.PacManSprites;
import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class PlayerTest
{

    //Instantiate the Player Class
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private final PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    1 usage
    private final Player ThePlayer = Factory.createPacMan();

    //Test if the Player is alive

    @Test
    void testAlive()
    {
        assertThat(ThePlayer.isAlive()).isEqualTo( expected: true);
    }
}
```

Is the coverage good enough?
No, the coverage only covers 3% of the entire code, and it only looks through the board. We probably do not need 100% coverage, but the more the better and at the moment it is very sparse.

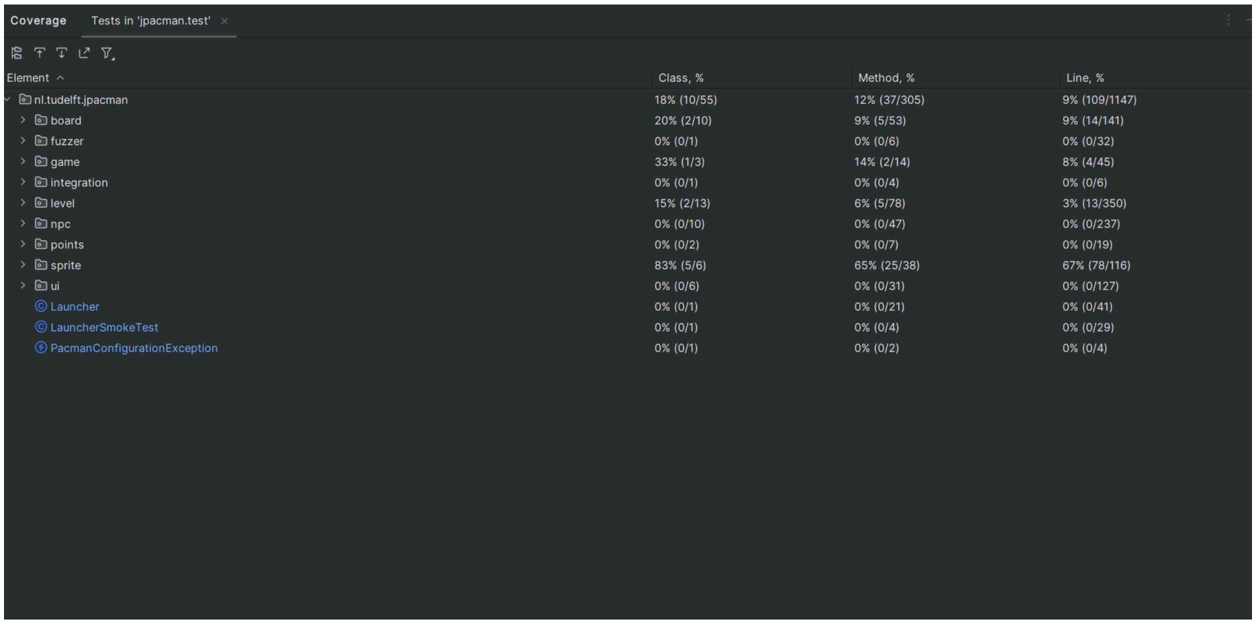
Task 2



```java

import static org.assertj.core.api.AssertionsForClassTypes.assertThat;

tnishamon
public class SpriteTest
{
    2 usages
    private final PacManSprites sprites = new PacManSprites();
    // Sprite list, int frames, bool loop

    1 usage
    private final AnimatedSprite deathAnimation = sprites.getPacManDeathAnimation();

    tnishamon
    @Test
    void spriteDrawn()
    {
        assertThat(sprites.getPelletSprite()).isNotNull();
    }

    tnishamon
    @Test
    void checkHeightOfDeathAnimation() { assertThat(deathAnimation.getHeight()).isBetween(0, 100); }
}
```



```java
import static org.assertj.core.api.AssertionsForClassTypes.assertThat;

tnishamon
public class FactorySinglePlayerTest
{
    //Instantiate the Player Class
    1 usage
    private static final PacManSprites SPRITE_STORE = new PacManSprites();
    1 usage
    private final PlayerFactory Factory = new PlayerFactory(SPRITE_STORE);
    1 usage
    private final GameFactory GF = new GameFactory(Factory);

    // I tried very hard to test createSinglePlayerGame, but it became very difficult to initialize everything
    tnishamon
    @Test
    void testSP()
    {
        assertThat(GF.getPlayerFactory()).isNotNull();
    }
}
```

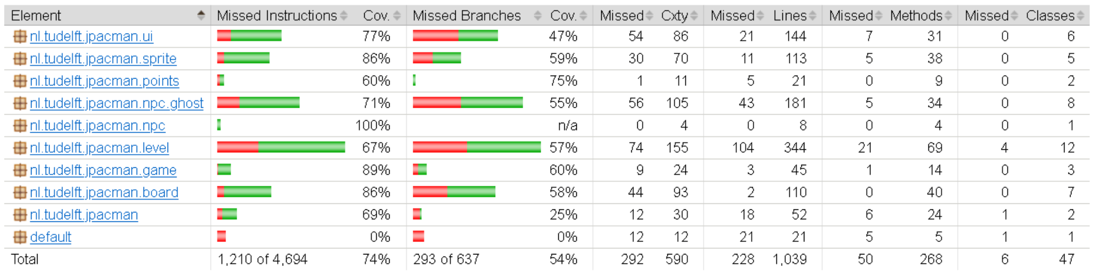


Coverage   Tests in 'jpacman.test'

Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	18% (10/55)	12% (37/305)	9% (109/1147)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	33% (1/3)	14% (2/14)	8% (4/45)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	15% (2/13)	6% (5/78)	3% (13/350)
npc	0% (0/10)	0% (0/47)	0% (0/237)
points	0% (0/2)	0% (0/7)	0% (0/19)
sprite	83% (5/6)	65% (25/38)	67% (78/116)
ui	0% (0/6)	0% (0/31)	0% (0/127)
Launcher	0% (0/1)	0% (0/21)	0% (0/41)
LauncherSmokeTest	0% (0/1)	0% (0/4)	0% (0/29)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Task 3

jpacman



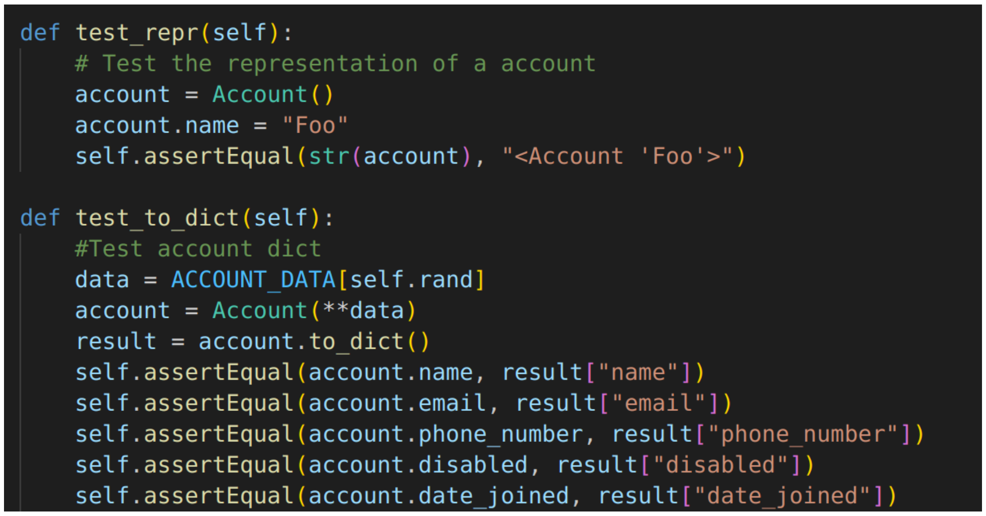
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
nl.tudelft.jpacman.level		67%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.game		89%		60%	9	24	3	45	1	14	0	3
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
default		0%		0%	12	12	21	21	5	5	1	1
Total	1,210 of 4,694	74%	293 of 637	54%	292	590	228	1,039	50	268	6	47

- Are the coverage results from JaCoCo similar to the ones you got from IntelliJ in the last task? Why so or why not?
- Did you find the source code visualization from JaCoCo on uncovered branches helpful?
- Which visualization did you prefer and why? IntelliJ's coverage window or JaCoCo's report?

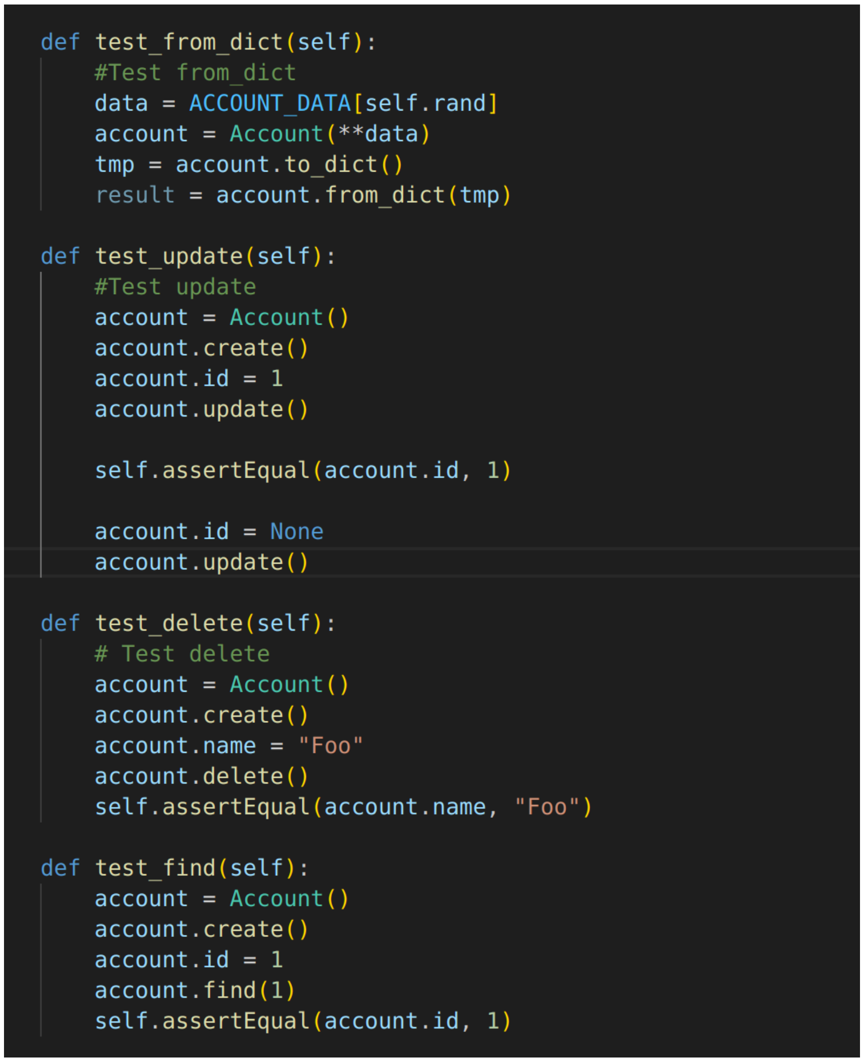
No, the coverage results from JaCoCo cover much more of the code than my own tests. I do most of my tests in sprite, but without testing EmptySprite, I overall have less coverage. There do seem to be a few select differences between JaCoCo and Intellij's output, like my output has 5 out of 6 classes found, but JaCoCo only found 5 classes. This is likely because my code counts the sprite interface as a class and we do not go through EmptySprite. If I did more tests in foundational Classes like NPC, board, or game my coverage might have been much greater. If I managed to implement createSinglePlayerGame like I had planned my coverage would be very similar since it needs to instantiate many classes, but I had trouble and made my task a little simpler by testing getPlayerFactory.

The source code visualization from JaCoCo was sort of helpful, it lets me see specifically where things are missed. Overall, I prefer the JaCoCo coverage for getting the extra information about everything, but I do think the IntelliJ coverage window has a place. If I had a large, finished project I think the JaCoCo coverage report is much better since it goes in depth. As I am working on a project and am coding things up, the IntelliJ coverage window is much more convenient and I think gives more than enough information.

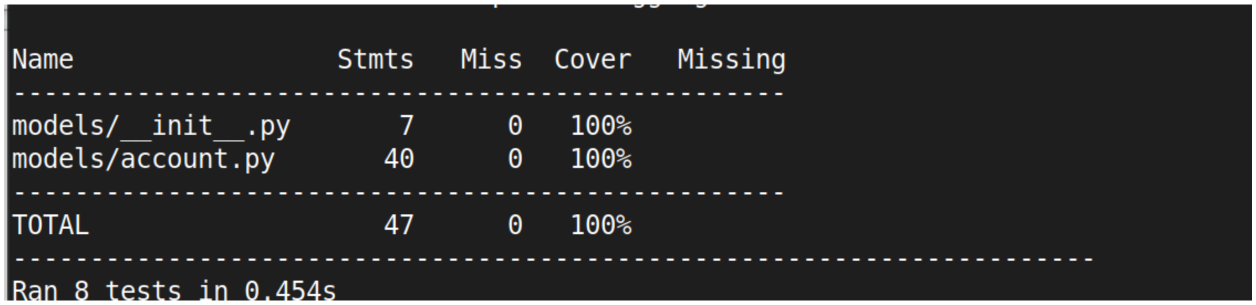
Task 4



```python
def test_repr(self):
    # Test the representation of a account
    account = Account()
    account.name = "Foo"
    self.assertEqual(str(account), "<Account 'Foo'>")

def test_to_dict(self):
    #Test account dict
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    result = account.to_dict()
    self.assertEqual(account.name, result["name"])
    self.assertEqual(account.email, result["email"])
    self.assertEqual(account.phone_number, result["phone_number"])
    self.assertEqual(account.disabled, result["disabled"])
    self.assertEqual(account.date_joined, result["date_joined"])
```

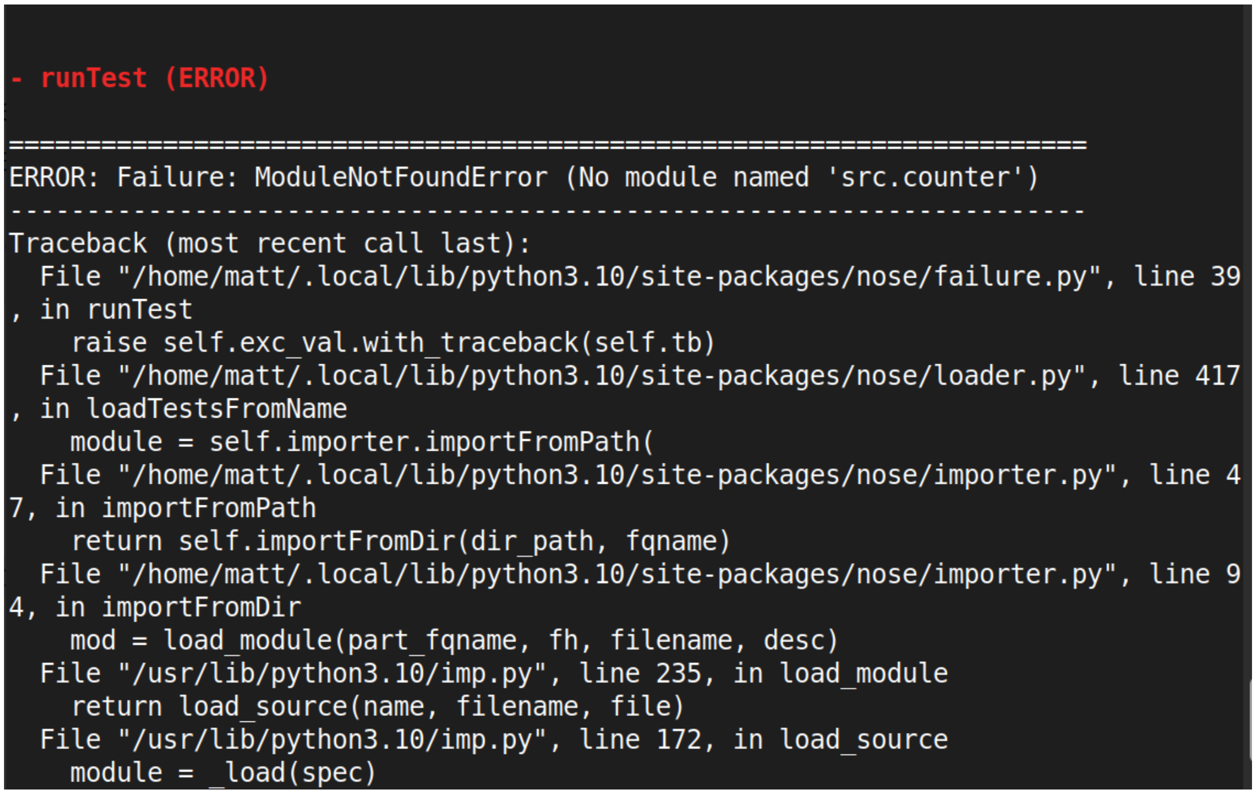


```python
def test_from_dict(self):
    #Test from_dict
    data = ACCOUNT_DATA[self.rand]
    account = Account(**data)
    tmp = account.to_dict()
    result = account.from_dict(tmp)

def test_update(self):
    #Test update
    account = Account()
    account.create()
    account.id = 1
    account.update()

    self.assertEqual(account.id, 1)

    account.id = None
    account.update()

def test_delete(self):
    # Test delete
    account = Account()
    account.create()
    account.name = "Foo"
    account.delete()
    self.assertEqual(account.name, "Foo")

def test_find(self):
    account = Account()
    account.create()
    account.id = 1
    account.find(1)
    self.assertEqual(account.id, 1)
```

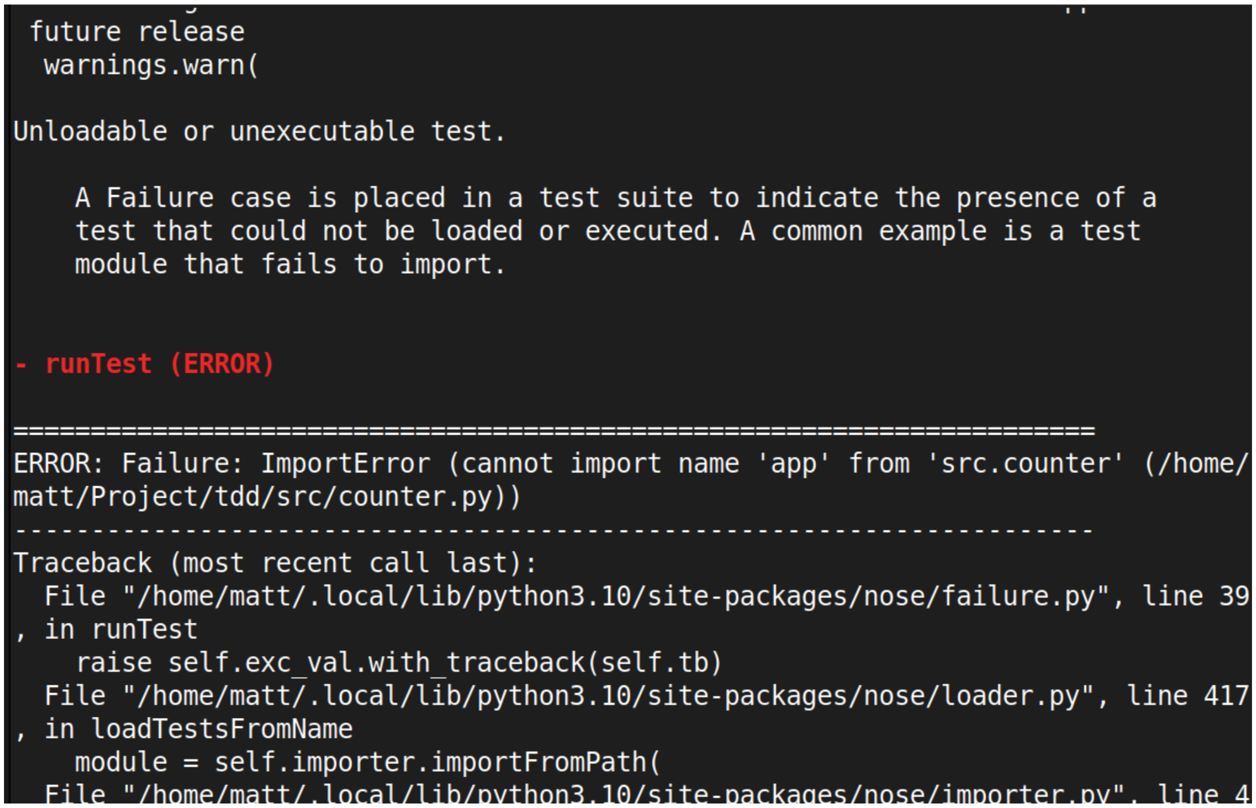


```
Name                   Stmts   Miss  Cover   Missing
-----------------------------------------------------
models/__init__.py         7      0   100%
models/account.py         40      0   100%
-----------------------------------------------------
TOTAL                     47      0   100%
----------------------------------------------------------------------
Ran 8 tests in 0.454s
```

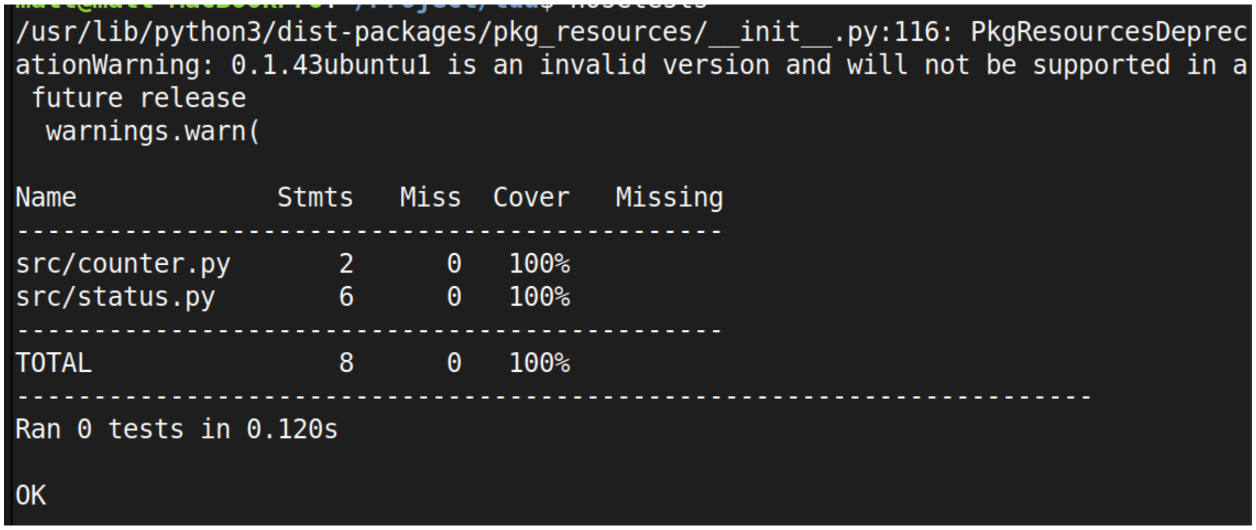
Task 5



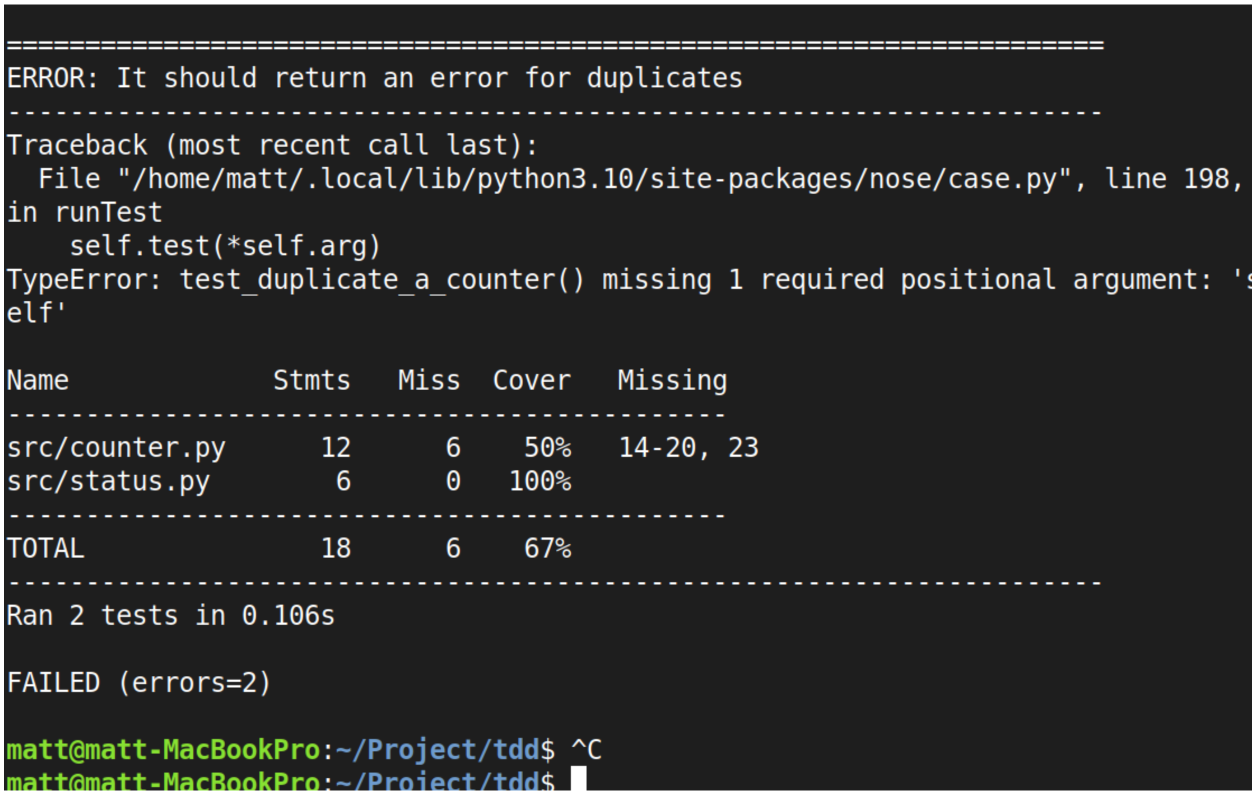
```
- runTest (ERROR)

======================================================================
ERROR: Failure: ModuleNotFoundError (No module named 'src.counter')
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/matt/.local/lib/python3.10/site-packages/nose/failure.py", line 39
, in runTest
    raise self.exc_val.with_traceback(self.tb)
  File "/home/matt/.local/lib/python3.10/site-packages/nose/loader.py", line 417
, in loadTestsFromName
    module = self.importer.importFromPath(
  File "/home/matt/.local/lib/python3.10/site-packages/nose/importer.py", line 4
7, in importFromPath
    return self.importFromDir(dir_path, fqname)
  File "/home/matt/.local/lib/python3.10/site-packages/nose/importer.py", line 9
4, in importFromDir
    mod = load_module(part_fqname, fh, filename, desc)
  File "/usr/lib/python3.10/imp.py", line 235, in load_module
    return load_source(name, filename, file)
  File "/usr/lib/python3.10/imp.py", line 172, in load_source
    module = _load(spec)
```



```
 future release
  warnings.warn(

Unloadable or unexecutable test.

    A Failure case is placed in a test suite to indicate the presence of a
    test that could not be loaded or executed. A common example is a test
    module that fails to import.


- runTest (ERROR)

======================================================================
ERROR: Failure: ImportError (cannot import name 'app' from 'src.counter' (/home/
matt/Project/tdd/src/counter.py))
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/matt/.local/lib/python3.10/site-packages/nose/failure.py", line 39
, in runTest
    raise self.exc_val.with_traceback(self.tb)
  File "/home/matt/.local/lib/python3.10/site-packages/nose/loader.py", line 417
, in loadTestsFromName
    module = self.importer.importFromPath(
  File "/home/matt/.local/lib/python3.10/site-packages/nose/importer.py", line 4
```

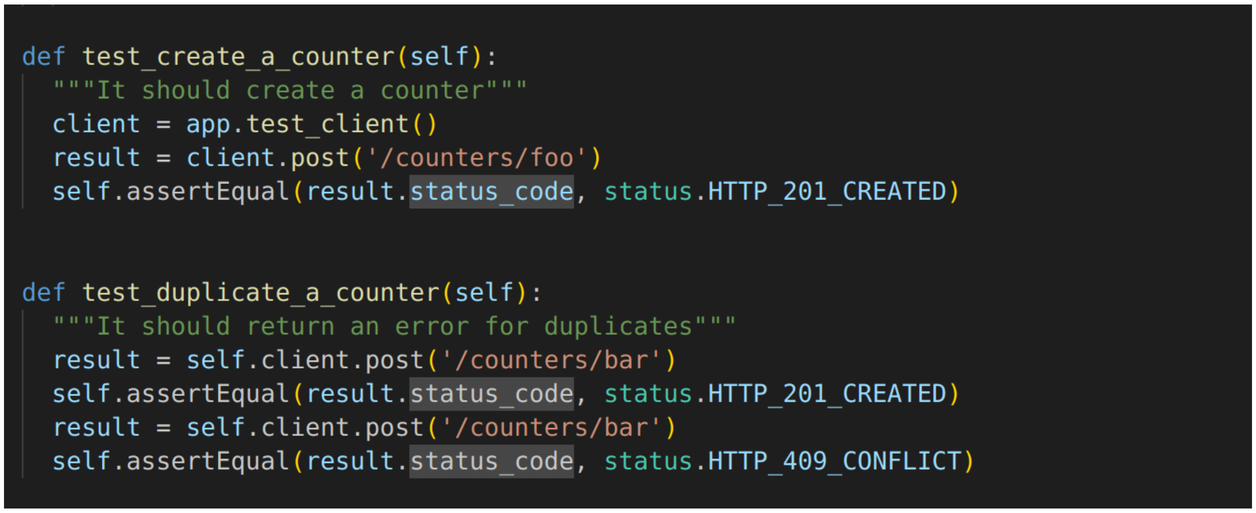


```
/usr/lib/python3/dist-packages/pkg_resources/__init__.py:116: PkgResourcesDeprec
ationWarning: 0.1.43ubuntu1 is an invalid version and will not be supported in a
 future release
  warnings.warn(

Name              Stmts   Miss  Cover   Missing
-----------------------------------------------
src/counter.py        2      0   100%
src/status.py         6      0   100%
-----------------------------------------------
TOTAL                 8      0   100%
----------------------------------------------------------------------
Ran 0 tests in 0.120s

OK
```

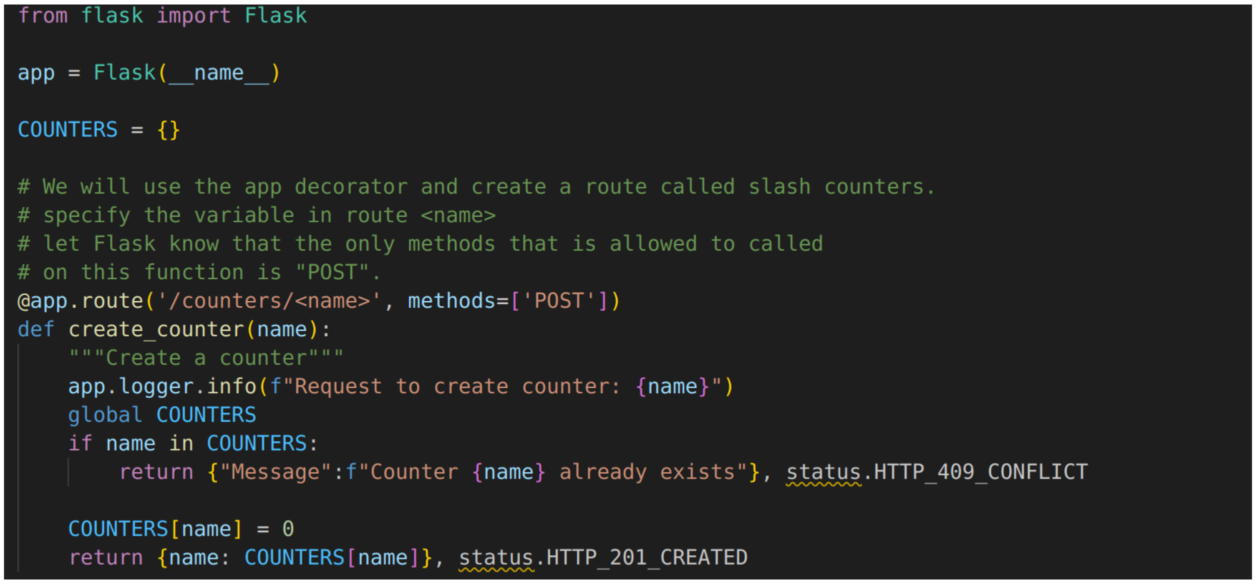


```
======================================================================
ERROR: It should return an error for duplicates
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/matt/.local/lib/python3.10/site-packages/nose/case.py", line 198,
in runTest
    self.test(*self.arg)
TypeError: test_duplicate_a_counter() missing 1 required positional argument: 's
elf'

Name               Stmts   Miss  Cover   Missing
----------------------------------------------
src/counter.py        12      6    50%   14-20, 23
src/status.py          6      0   100%
----------------------------------------------
TOTAL                 18      6    67%
----------------------------------------------------------------------
Ran 2 tests in 0.106s

FAILED (errors=2)

matt@matt-MacBookPro:~/Project/tdd$ ^C
matt@matt-MacBookPro:~/Project/tdd$
```

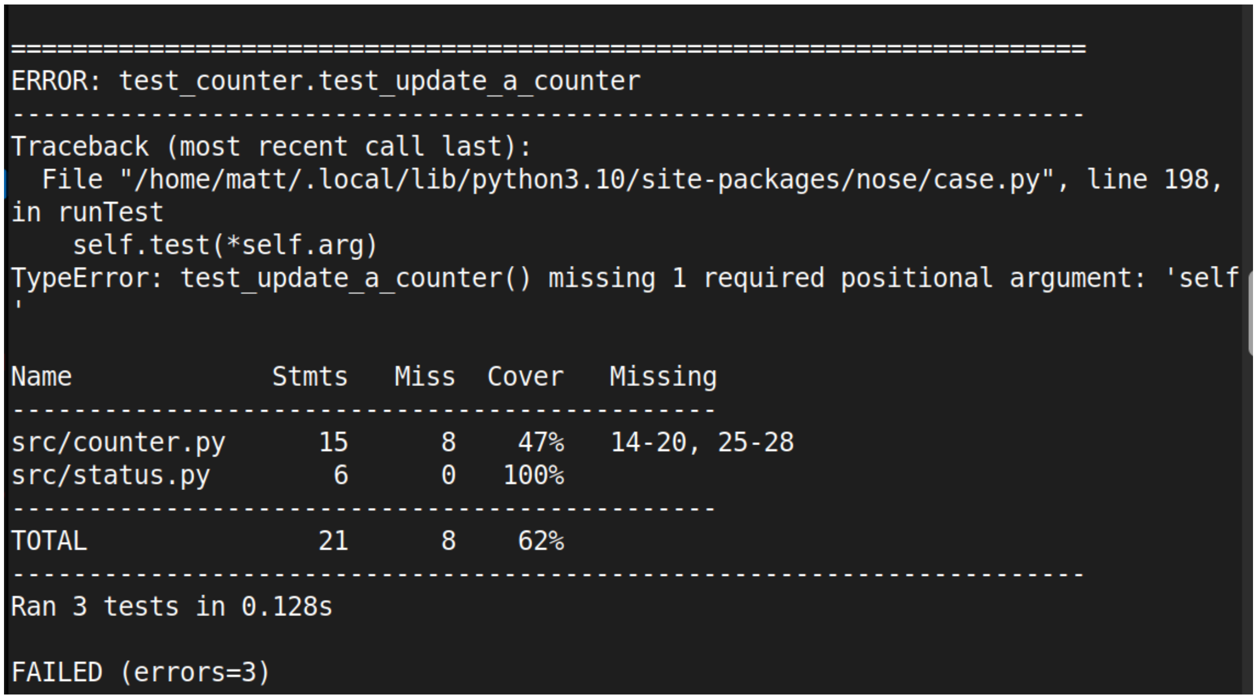
These are the results no matter what I updated. Something with nosetests must be wrong and I have been troubleshooting it for a while. It worked for Task 4 so I am unsure what the problem is.



```python
def test_create_a_counter(self):
    """It should create a counter"""
    client = app.test_client()
    result = client.post('/counters/foo')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)


def test_duplicate_a_counter(self):
    """It should return an error for duplicates"""
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    result = self.client.post('/counters/bar')
    self.assertEqual(result.status_code, status.HTTP_409_CONFLICT)
```

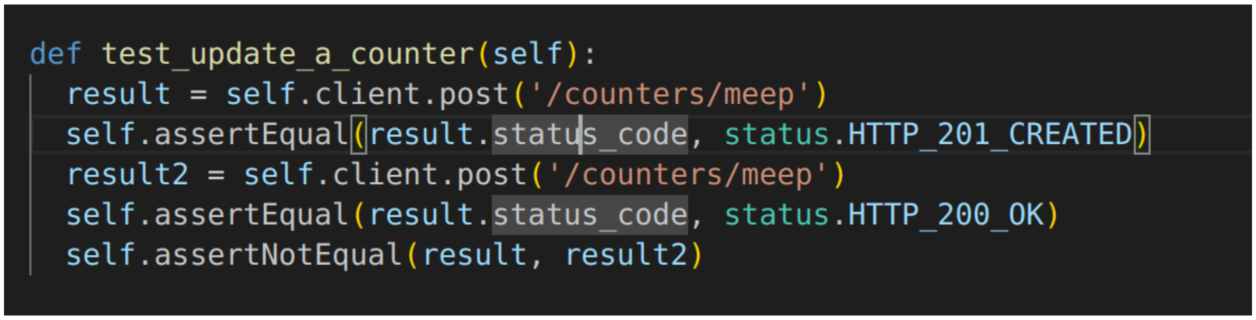


```python
from flask import Flask

app = Flask(__name__)

COUNTERS = {}

# We will use the app decorator and create a route called slash counters.
# specify the variable in route <name>
# let Flask know that the only methods that is allowed to called
# on this function is "POST".
@app.route('/counters/<name>', methods=['POST'])
def create_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to create counter: {name}")
    global COUNTERS
    if name in COUNTERS:
        return {"Message":f"Counter {name} already exists"}, status.HTTP_409_CONFLICT

    COUNTERS[name] = 0
    return {name: COUNTERS[name]}, status.HTTP_201_CREATED
```

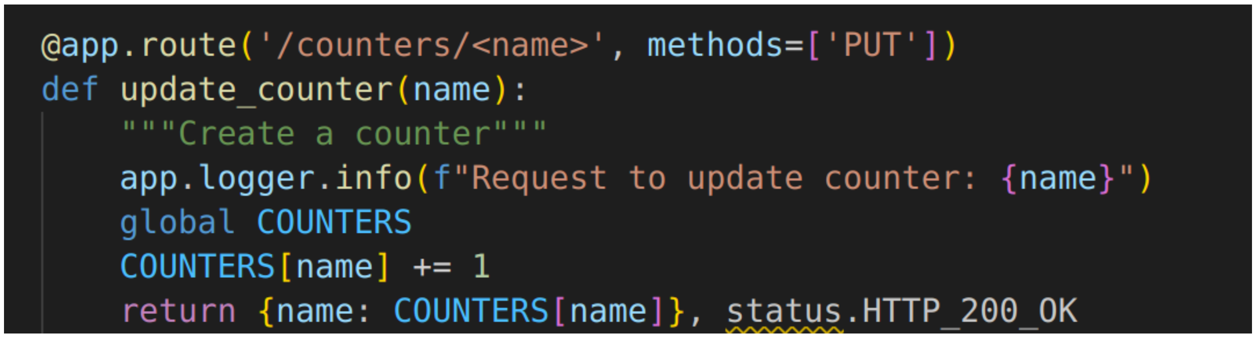


```
======================================================================
ERROR: test_counter.test_update_a_counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/matt/.local/lib/python3.10/site-packages/nose/case.py", line 198,
in runTest
    self.test(*self.arg)
TypeError: test_update_a_counter() missing 1 required positional argument: 'self
'

Name                Stmts   Miss  Cover   Missing
-------------------------------------------------
src/counter.py         15      8    47%   14-20, 25-28
src/status.py           6      0   100%
-------------------------------------------------
TOTAL                  21      8    62%
----------------------------------------------------------------------
Ran 3 tests in 0.128s

FAILED (errors=3)
```

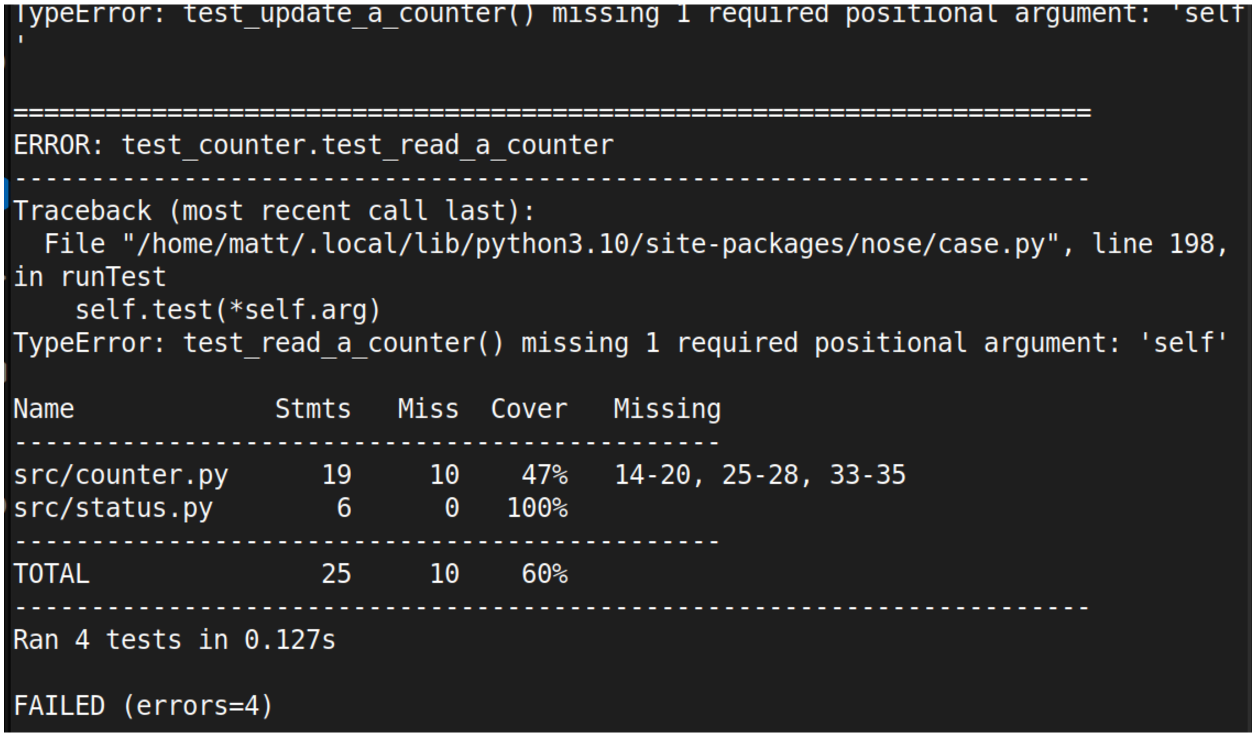
Below is the code from the above tests



```python
def test_update_a_counter(self):
  result = self.client.post('/counters/meep')
  self.assertEqual(result.status_code, status.HTTP_201_CREATED)
  result2 = self.client.post('/counters/meep')
  self.assertEqual(result.status_code, status.HTTP_200_OK)
  self.assertNotEqual(result, result2)
```

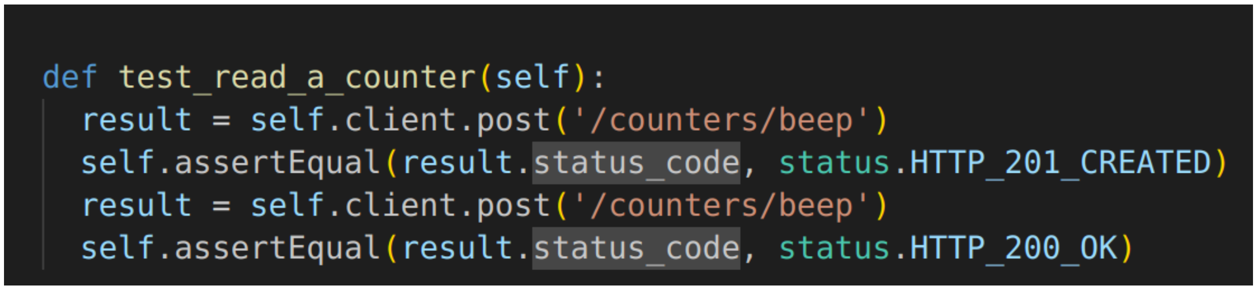


```python
@app.route('/counters/<name>', methods=['PUT'])
def update_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to update counter: {name}")
    global COUNTERS
    COUNTERS[name] += 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```

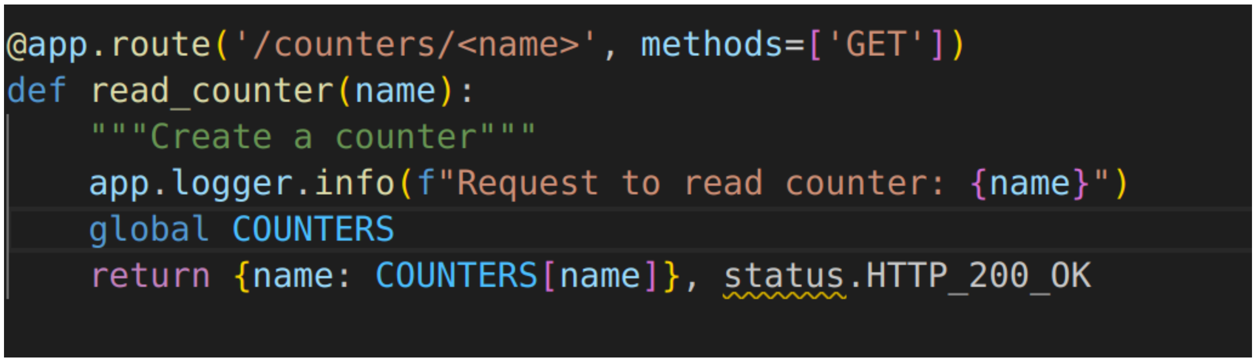


```
TypeError: test_update_a_counter() missing 1 required positional argument: 'self
'

======================================================================
ERROR: test_counter.test_read_a_counter
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/matt/.local/lib/python3.10/site-packages/nose/case.py", line 198,
in runTest
    self.test(*self.arg)
TypeError: test_read_a_counter() missing 1 required positional argument: 'self'

Name                 Stmts   Miss  Cover   Missing
----------------------------------------------------
src/counter.py          19     10    47%   14-20, 25-28, 33-35
src/status.py            6      0   100%
----------------------------------------------------
TOTAL                   25     10    60%
----------------------------------------------------------------------
Ran 4 tests in 0.127s

FAILED (errors=4)
```

Below is the code from the above tests



```
  def test_read_a_counter(self):
    result = self.client.post('/counters/beep')
    self.assertEqual(result.status_code, status.HTTP_201_CREATED)
    result = self.client.post('/counters/beep')
    self.assertEqual(result.status_code, status.HTTP_200_OK)
```



```
@app.route('/counters/<name>', methods=['GET'])
def read_counter(name):
    """Create a counter"""
    app.logger.info(f"Request to read counter: {name}")
    global COUNTERS
    return {name: COUNTERS[name]}, status.HTTP_200_OK
```