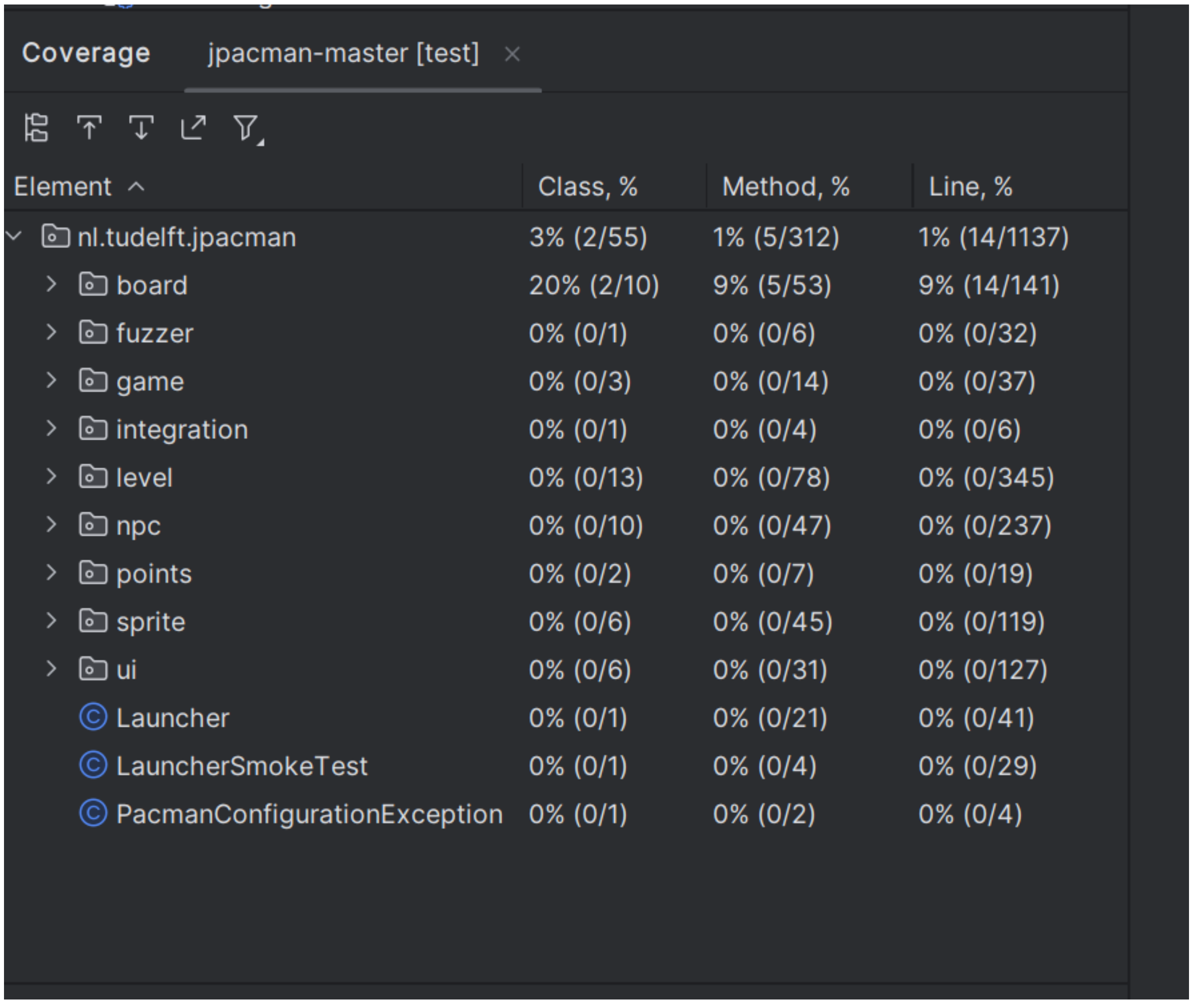
Ruben Langarica

Lab2: Testing

Task 1 – JPacman Test Coverage



Coverage   jpacman-master [test]

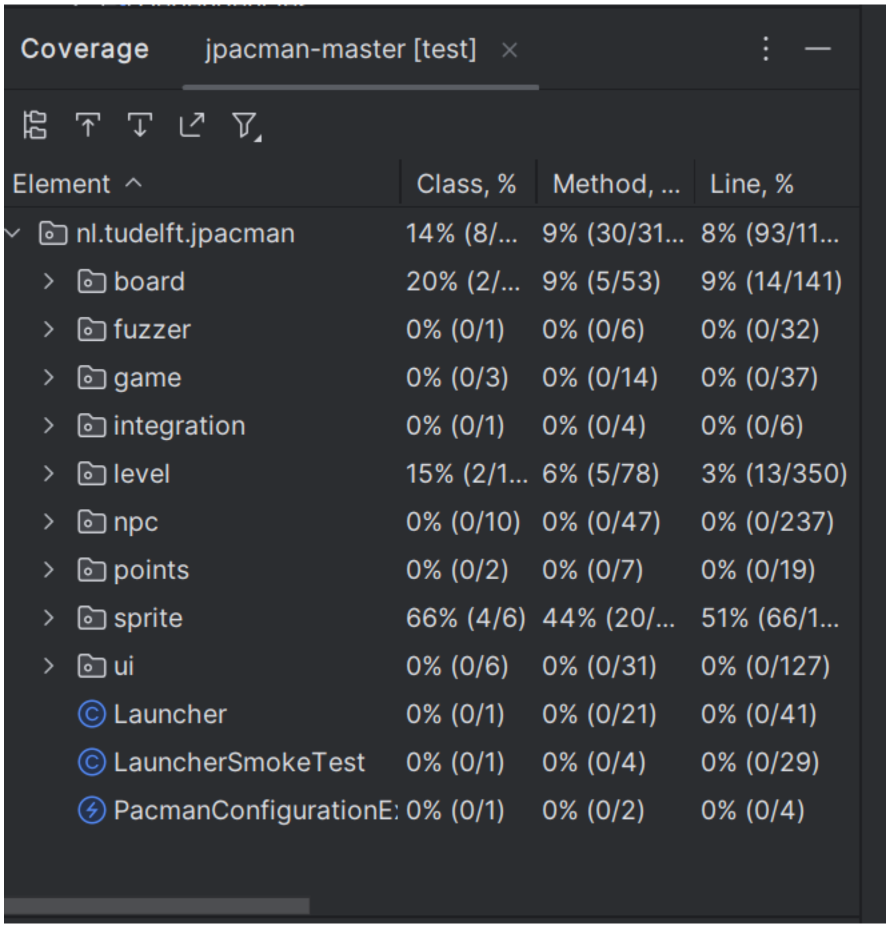
Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	3% (2/55)	1% (5/312)	1% (14/1137)
board	20% (2/10)	9% (5/53)	9% (14/141)
fuzzer	0% (0/1)	0% (0/6)	0% (0/32)
game	0% (0/3)	0% (0/14)	0% (0/37)
integration	0% (0/1)	0% (0/4)	0% (0/6)
level	0% (0/13)	0% (0/78)	0% (0/345)
npc	0% (0/10)	0% (0/47)	0% (0/237)
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)
PacmanConfigurationException	0% (0/1)	0% (0/2)	0% (0/4)

Figure1: Default Test coverage of Jpacman

Is the coverage good enough?

No as there are significant amounts that went untested and leading to a low-test coverage.

Task 2 – Increasing Coverage on JPacman



Element	Class, %	Method, ...	Line, %
nl.tudelft.jpacman	14% (8/...	9% (30/31...	8% (93/11...
board	20% (2/...	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	15% (2/1...	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	66% (4/6)	44% (20/...	51% (66/1...
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)
PacmanConfigurationE:	0% (0/1)	0% (0/2)	0% (0/4)

Figure2: Test coverage with isAlive() implemented

Task 2.1 - 15 points

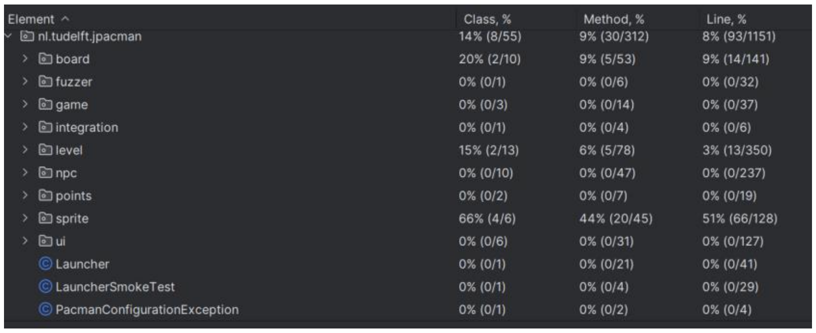
1. All NPC test

rc/main/java/nl/tudelft/jpacman/game/ghostFactory.createBlinky

rc/main/java/nl/tudelft/jpacman/game/ghostFactory.createInky

rc/main/java/nl/tudelft/jpacman/game/ghostFactory.createPinky

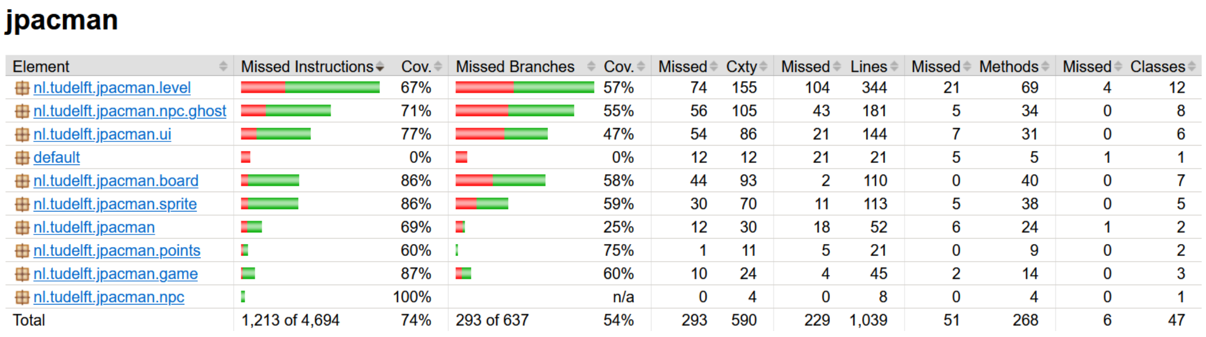
```
grid[x][y] = makeGhostSquare(ghosts, ghostFactory.createBlinky());
assertThat(grid[x][y]).isNotNull();

grid[x][y] = makeGhostSquare(ghosts, ghostFactory.createInky());
assertThat(grid[x][y]).isNotNull();

grid[x][y] = makeGhostSquare(ghosts, ghostFactory.createPinky());
assertThat(grid[x][y]).isNotNull();
```



Element	Class, %	Method, %	Line, %
nl.tudelft.jpacman	14% (8/55)	9% (30/312)	8% (93/1151)
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	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	66% (4/6)	44% (20/45)	51% (66/128)
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)

Figure3: Coverage report after adding more test cases

Task 3 – JaCoCo Report on JPacman (10 points)



jpacman

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
nl.tudelft.jpacman.level		67%		57%	74	155	104	344	21	69	4	12
nl.tudelft.jpacman.npc.ghost		71%		55%	56	105	43	181	5	34	0	8
nl.tudelft.jpacman.ui		77%		47%	54	86	21	144	7	31	0	6
default		0%		0%	12	12	21	21	5	5	1	1
nl.tudelft.jpacman.board		86%		58%	44	93	2	110	0	40	0	7
nl.tudelft.jpacman.sprite		86%		59%	30	70	11	113	5	38	0	5
nl.tudelft.jpacman		69%		25%	12	30	18	52	6	24	1	2
nl.tudelft.jpacman.points		60%		75%	1	11	5	21	0	9	0	2
nl.tudelft.jpacman.game		87%		60%	10	24	4	45	2	14	0	3
nl.tudelft.jpacman.npc		100%		n/a	0	4	0	8	0	4	0	1
Total	1,213 of 4,694	74%	293 of 637	54%	293	590	229	1,039	51	268	6	47

Figure4: jacman coverage results

Are the coverage results from `JaCoCo` similar to the ones you got from `IntelliJ` in the last task? Why so or why not?

No as there is a different spread in the information presented as Jacoco shows branch coverage making results vary with conditionals used.

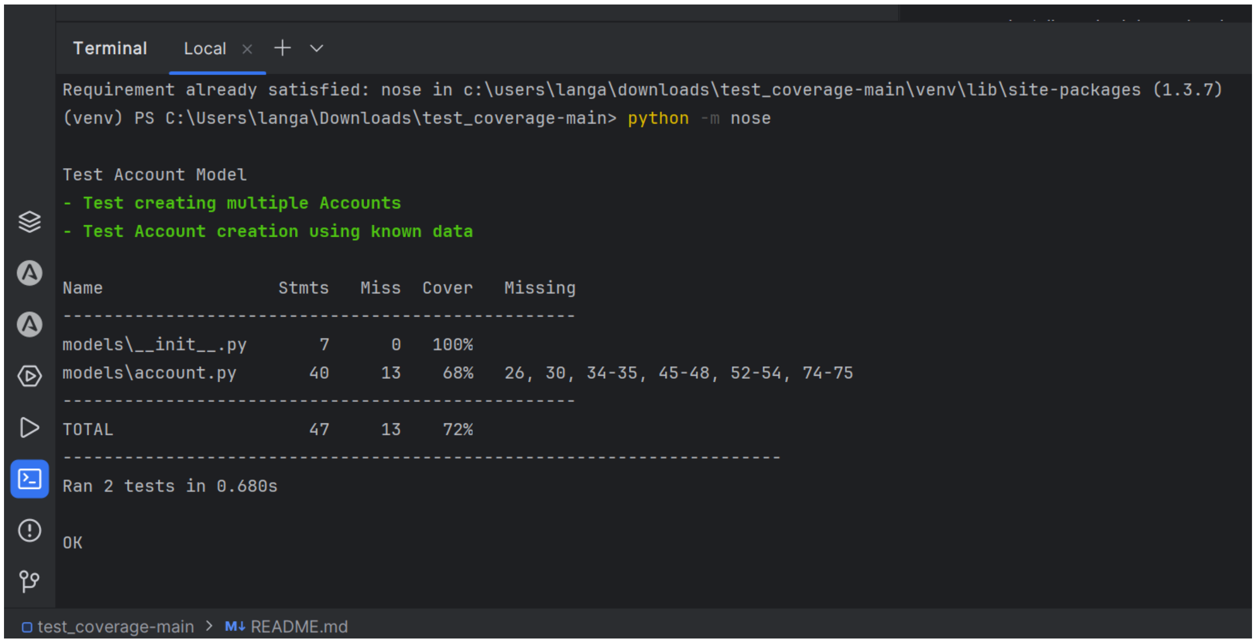
Did you find helpful the source code visualization from `JaCoCo` on uncovered branches?

Yes it help by showing which parts of the branches were not tested and tests can therefore be focused more for those.

Which visualization did you prefer and why? `IntelliJ`'s coverage window or `JaCoCo`'s report?

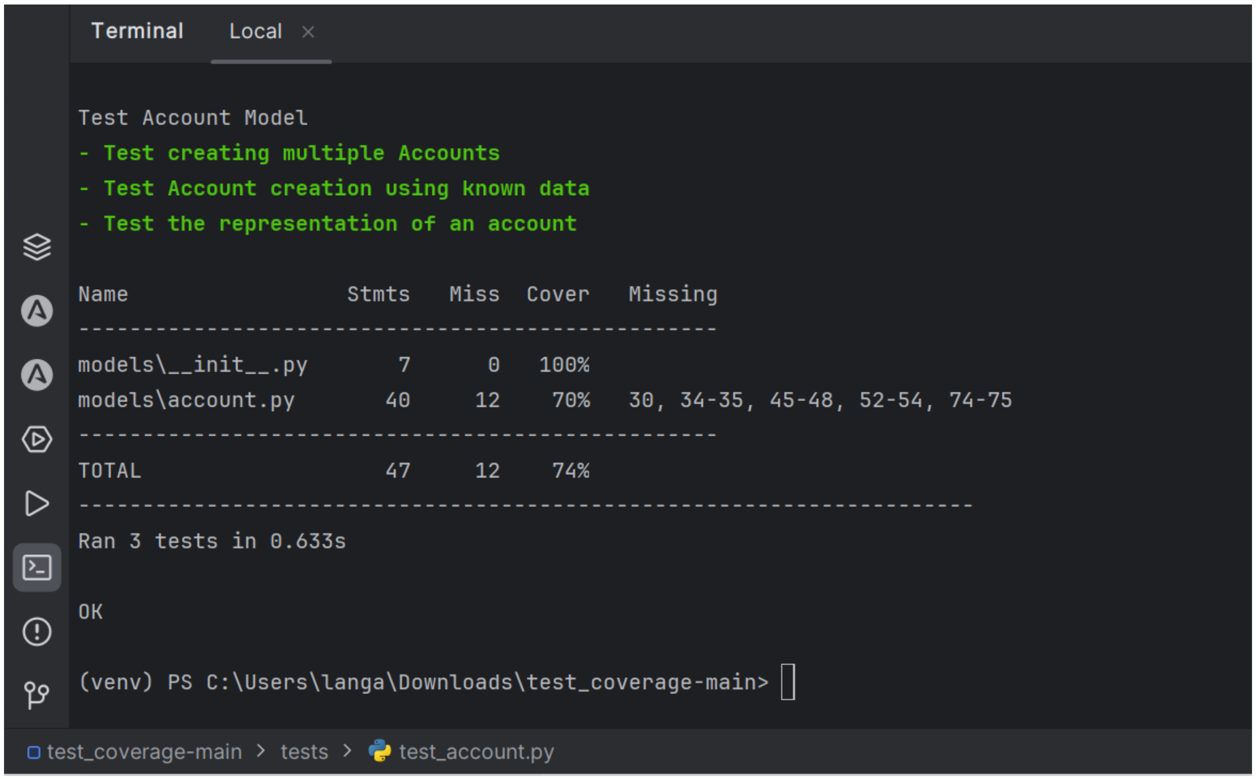
The Jacoco ended up being a better one as it is able to display the branches and in what categories they fall into that were missed or completed.

Task 4 – Working with Python Test Coverage



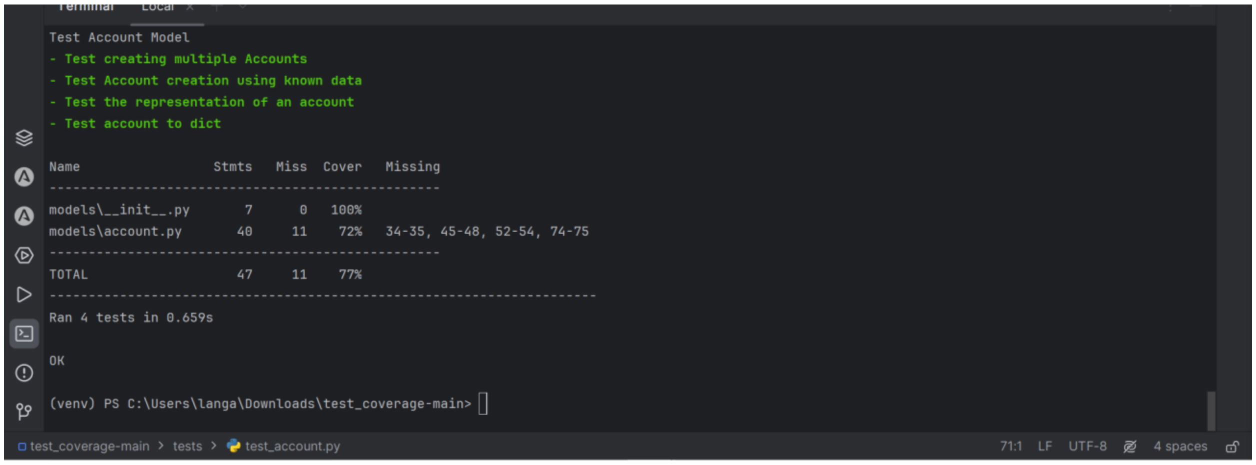
```
Requirement already satisfied: nose in c:\users\langa\downloads\test_coverage-main\venv\lib\site-packages (1.3.7)
(venv) PS C:\Users\langa\Downloads\test_coverage-main> python -m nose

Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40     13    68%   26, 30, 34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   47     13    72%
----------------------------------------------------------------------
Ran 2 tests in 0.680s

OK
```

Figure5: Python Coverage test at 72%



```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test the representation of an account

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40     12    70%   30, 34-35, 45-48, 52-54, 74-75
--------------------------------------------------
TOTAL                   47     12    74%
----------------------------------------------------------------------
Ran 3 tests in 0.633s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

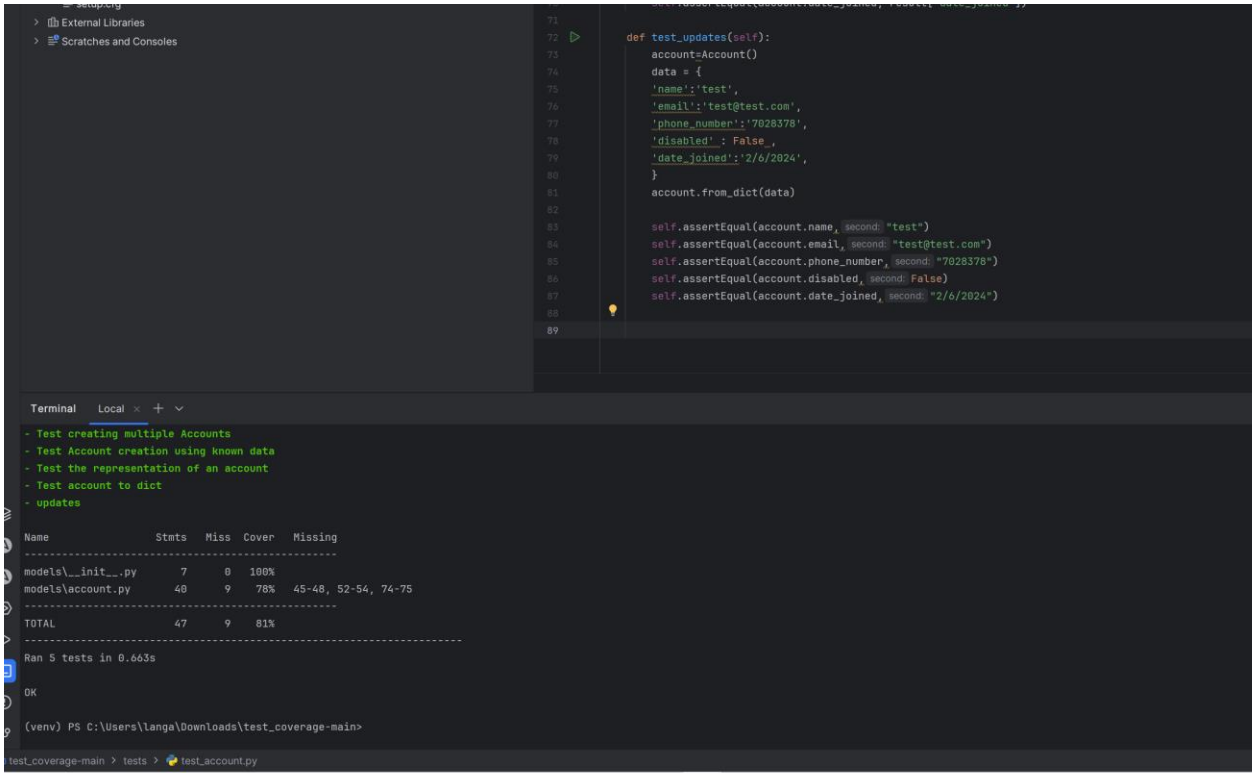
Figure6: Test Coverage 74% after adding test_repr()



```
Test Account Model
- Test creating multiple Accounts
- Test Account creation using known data
- Test the representation of an account
- Test account to dict

Name                   Stmts   Miss  Cover   Missing
----------------------------------------------------
models\__init__.py         7      0   100%
models\account.py         40     11    72%   34-35, 45-48, 52-54, 74-75
----------------------------------------------------
TOTAL                     47     11    77%
----------------------------------------------------------------------
Ran 4 tests in 0.659s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

Figure7: Test Coverage 77% after adding test_to_dict()

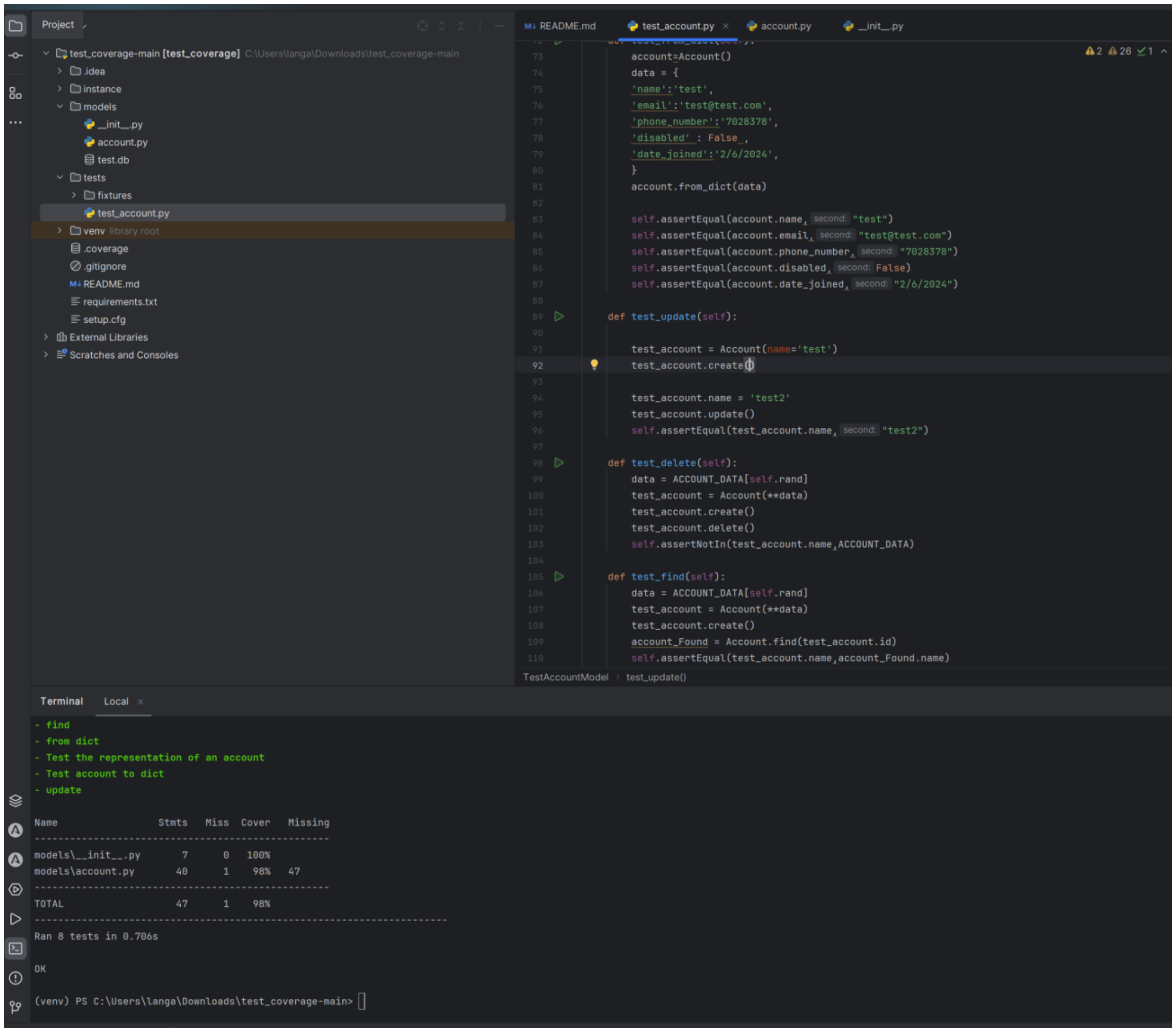
This test result ended up doing 1% more then what is stated in the lab



```
    def test_updates(self):
        account=Account()
        data = {
        'name':'test',
        'email':'test@test.com',
        'phone_number':'7028378',
        'disabled' : False ,
        'date_joined':'2/6/2024',
        }
        account.from_dict(data)

        self.assertEqual(account.name, second: "test")
        self.assertEqual(account.email, second: "test@test.com")
        self.assertEqual(account.phone_number, second: "7028378")
        self.assertEqual(account.disabled, second: False)
        self.assertEqual(account.date_joined, second: "2/6/2024")
```

```
- Test creating multiple Accounts
- Test Account creation using known data
- Test the representation of an account
- Test account to dict
- updates

Name                   Stmts   Miss  Cover   Missing
----------------------------------------------------
models\__init__.py         7      0   100%
models\account.py         40      9    78%   45-48, 52-54, 74-75
----------------------------------------------------
TOTAL                     47      9    81%
----------------------------------------------------------------------
Ran 5 tests in 0.663s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

Figure 8: Test increase after Testing of 1 case implemented.

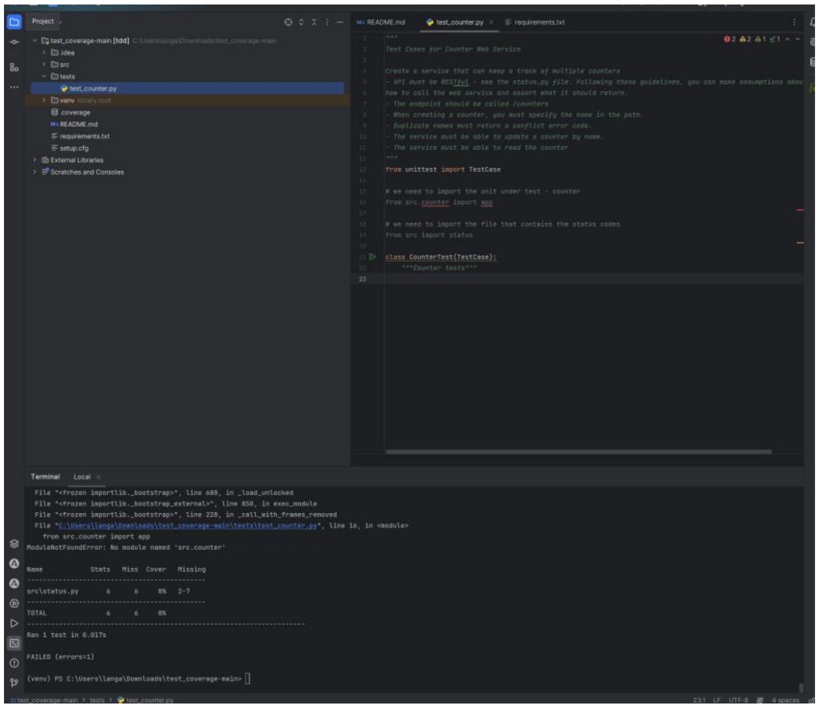


```
test_coverage-main [test_coverage] C:\Users\langa\Downloads\test_coverage-main
  .idea
  instance
  models
    __init__.py
    account.py
    test.db
  tests
    fixtures
    test_account.py
  venv library root
  .coverage
  .gitignore
  README.md
  requirements.txt
  setup.cfg
External Libraries
Scratches and Consoles
```

```python
        account=Account()
        data = {
        'name':'test',
        'email':'test@test.com',
        'phone_number':'7028378',
        'disabled' : False ,
        'date_joined':'2/6/2024',
        }
        account.from_dict(data)

        self.assertEqual(account.name, "test")
        self.assertEqual(account.email, "test@test.com")
        self.assertEqual(account.phone_number, "7028378")
        self.assertEqual(account.disabled, False)
        self.assertEqual(account.date_joined, "2/6/2024")

    def test_update(self):

        test_account = Account(name='test')
        test_account.create()

        test_account.name = 'test2'
        test_account.update()
        self.assertEqual(test_account.name, "test2")

    def test_delete(self):
        data = ACCOUNT_DATA[self.rand]
        test_account = Account(**data)
        test_account.create()
        test_account.delete()
        self.assertNotIn(test_account.name,ACCOUNT_DATA)

    def test_find(self):
        data = ACCOUNT_DATA[self.rand]
        test_account = Account(**data)
        test_account.create()
        account_Found = Account.find(test_account.id)
        self.assertEqual(test_account.name,account_Found.name)
```

```
- find
- from dict
- Test the representation of an account
- Test account to dict
- update

Name                 Stmts   Miss  Cover   Missing
--------------------------------------------------
models\__init__.py       7      0   100%
models\account.py       40      1    98%   47
--------------------------------------------------
TOTAL                   47      1    98%
----------------------------------------------------------------------
Ran 8 tests in 0.706s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

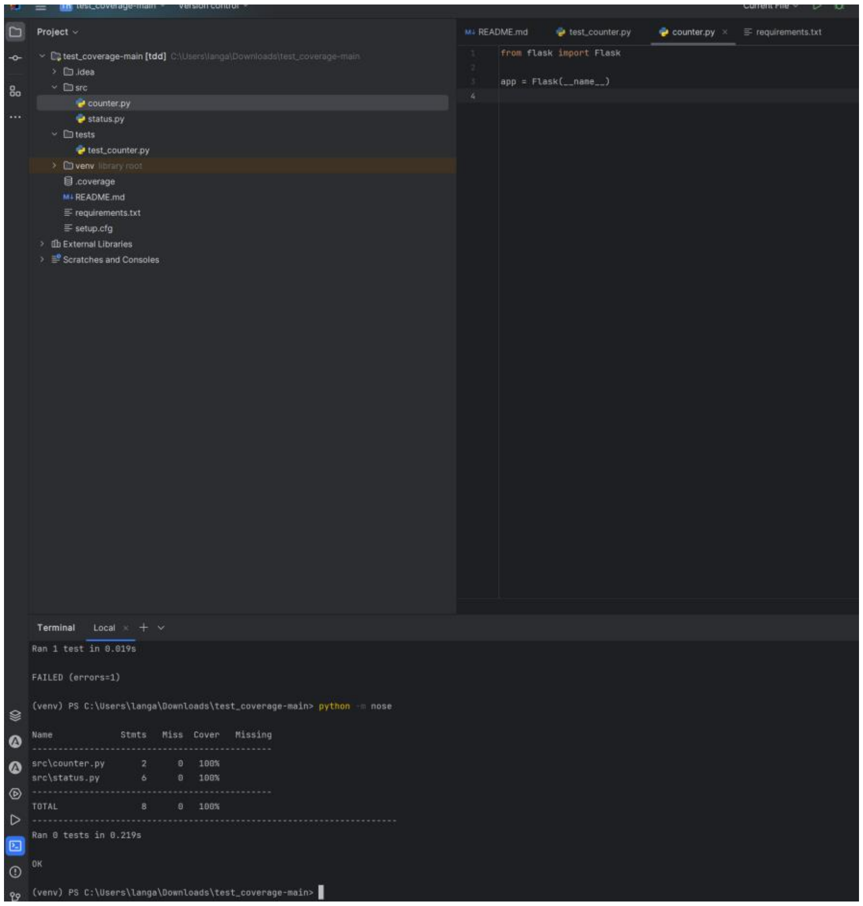
Task 5 - TDD



```
"""
Test Cases for Counter Web Service

Create a service that can keep a track of multiple counters
- API must be RESTful - see the status.py file. Following these guidelines, you can make assumptions abou
how to call the web service and assert what it should return.
- The endpoint should be called /counters
- When creating a counter, you must specify the name in the path.
- Duplicate names must return a conflict error code.
- The service must be able to update a counter by name.
- The service must be able to read the counter
"""
from unittest import TestCase

# we need to import the unit under test - counter
from src.counter import app

# we need to import the file that contains the status codes
from src import status

class CounterTest(TestCase):
    """Counter tests"""
```

```
  File "<frozen importlib._bootstrap>", line 688, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 850, in exec_module
  File "<frozen importlib._bootstrap>", line 228, in _call_with_frames_removed
  File "C:\Users\langa\Downloads\test_coverage-main\tests\test_counter.py", line 16, in <module>
    from src.counter import app
ModuleNotFoundError: No module named 'src.counter'

Name            Stmts   Miss  Cover   Missing
---------------------------------------------
src\status.py       6      6     0%   2-7
---------------------------------------------
TOTAL               6      6     0%
----------------------------------------------------------------------
Ran 1 test in 0.017s

FAILED (errors=1)

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

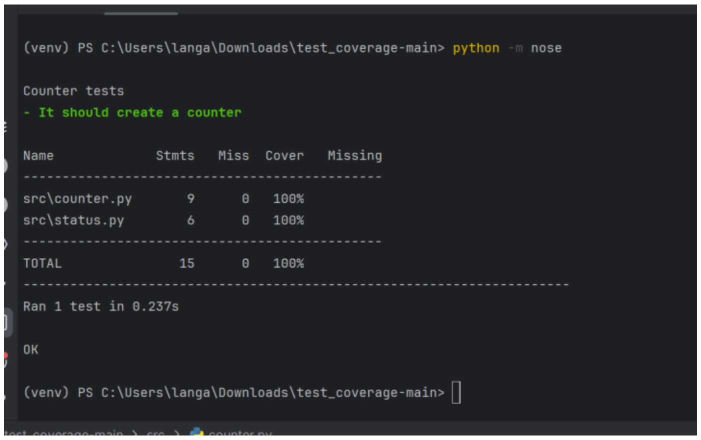
Figure10: Error in TDD



```
from flask import Flask

app = Flask(__name__)
```

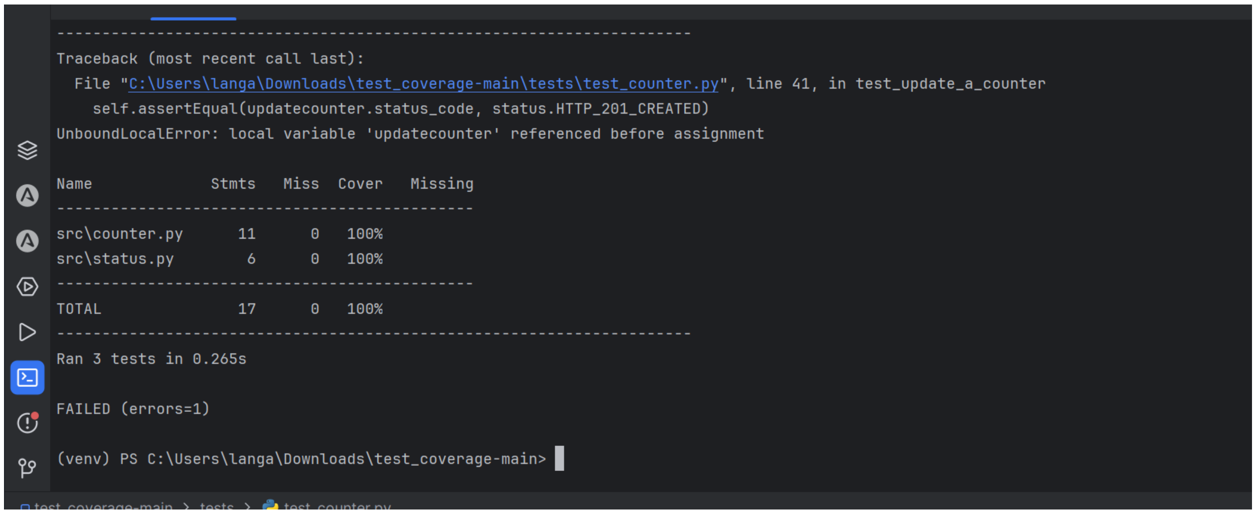
```
Ran 1 test in 0.019s

FAILED (errors=1)

(venv) PS C:\Users\langa\Downloads\test_coverage-main> python -m nose

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 0.219s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

Figure11: counter.py test with no error



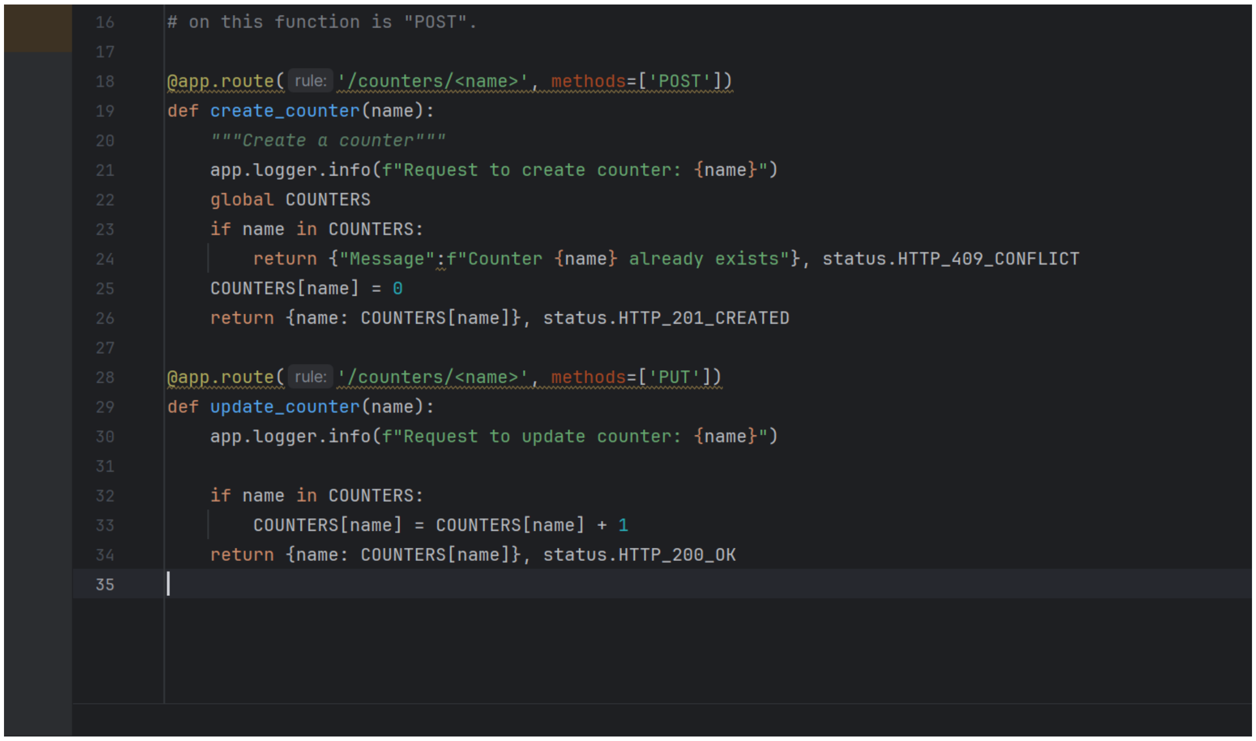
```
(venv) PS C:\Users\langa\Downloads\test_coverage-main> python -m nose

Counter tests
- It should create a counter

Name            Stmts   Miss  Cover   Missing
----------------------------------------------
src\counter.py      9      0   100%
src\status.py       6      0   100%
----------------------------------------------
TOTAL              15      0   100%
----------------------------------------------------------------------
Ran 1 test in 0.237s

OK

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

Figure12: Green to proceed.



```
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\langa\Downloads\test_coverage-main\tests\test_counter.py", line 41, in test_update_a_counter
    self.assertEqual(updatecounter.status_code, status.HTTP_201_CREATED)
UnboundLocalError: local variable 'updatecounter' referenced before assignment

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 0.265s

FAILED (errors=1)

(venv) PS C:\Users\langa\Downloads\test_coverage-main>
```

Figure 13: In red after adding `test_update_a_counter(self)`



```
# on this function is "POST".

@app.route( rule: '/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

@app.route( rule: '/counters/<name>', methods=['PUT'])
def update_counter(name):
    app.logger.info(f"Request to update counter: {name}")

    if name in COUNTERS:
        COUNTERS[name] = COUNTERS[name] + 1
    return {name: COUNTERS[name]}, status.HTTP_200_OK

```