

# Homework 4: Code Testing

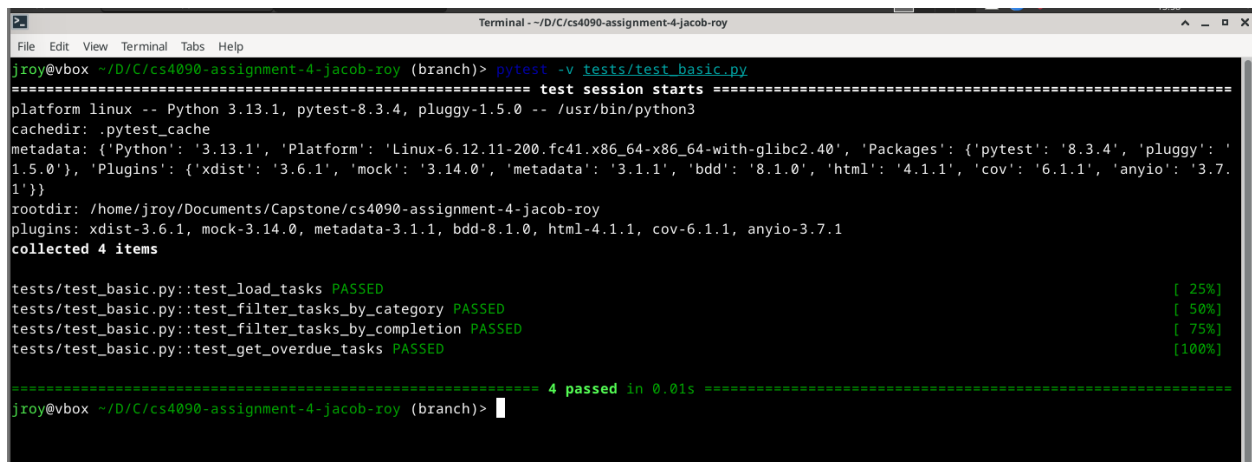
Jacob Roy

## Unit Testing:

For unit testing, I like to create tests for the more complicated functions first making sure that any branches in the function are gone through before moving on to the less complicated functions.

This means that, for this assignment, I started with the `load_tasks` function. Using temporary test files, I built the test to check the standard behavior and two alternate behaviors that occur when either the targeted file is missing or corrupted. I then created tests for the different filter functions; using a test list, I checked that the returned list was the expected size and that the contents of the list contained the expected values. Lastly, I created a test for the `get_overdue_tasks` function, which behaviorally is similar to the filter functions so has a similar test, checking the returned list's size and values.

With these tests to help find bugs in the functionality, we can run them and observe the results. Looking at the figure below, it can be seen that every test that was ran passed, indicating that the test functions are working as intended.

A terminal window titled "Terminal - ~/D/C/cs4090-assignment-4-jacob-roy" showing the execution of a pytest command. The user runs `pytest -v tests/test_basic.py`. The output displays the test session starting with platform and plugin information, followed by the collection of 4 items. The test results show four tests passing: `test_load_tasks` (25%), `test_filter_tasks_by_category` (50%), `test_filter_tasks_by_completion` (75%), and `test_get_overdue_tasks` (100%). The final summary indicates "4 passed in 0.01s".

```
Terminal - ~/D/C/cs4090-assignment-4-jacob-roy
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)> pytest -v tests/test_basic.py
===== test session starts =====
platform linux -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
metadata: {'Python': '3.13.1', 'Platform': 'Linux-6.12.11-200.fc41.x86_64-x86_64-with-glibc2.40', 'Packages': {'pytest': '8.3.4', 'pluggy': '1.5.0'}, 'Plugins': {'xdist': '3.6.1', 'mock': '3.14.0', 'metadata': '3.1.1', 'bdd': '8.1.0', 'html': '4.1.1', 'cov': '6.1.1', 'anyio': '3.7.1'}}
rootdir: /home/jroy/Documents/Capstone/cs4090-assignment-4-jacob-roy
plugins: xdist-3.6.1, mock-3.14.0, metadata-3.1.1, bdd-8.1.0, html-4.1.1, cov-6.1.1, anyio-3.7.1
collected 4 items

tests/test_basic.py::test_load_tasks PASSED [ 25%]
tests/test_basic.py::test_filter_tasks_by_category PASSED [ 50%]
tests/test_basic.py::test_filter_tasks_by_completion PASSED [ 75%]
tests/test_basic.py::test_get_overdue_tasks PASSED [100%]

===== 4 passed in 0.01s =====
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)>
```

## Bug Report:

### Task ID Generation Not Reliable When Tasks Are Deleted

#### Environment

- Python with Streamlit
- File-based task storage via JSON (tasks.json)

#### Preconditions

- Multiple tasks have been added
- At least one task has been deleted

#### Steps to Reproduce

1. Add three tasks via the sidebar form.
2. Delete the second task.
3. Add a new task.

#### Expected Result

- The new task should get a unique ID not previously used (e.g., ID 4 after deleting ID 2).

#### Actual Result

- The new task receives ID  $\text{len}(\text{tasks}) + 1$ , which could **reuse** a previously deleted ID (e.g., reuse ID 3 if only two tasks are left). This causes **duplicate IDs**, leading to UI bugs in Streamlit (e.g., incorrect key behavior in buttons).

#### Bug Type

- **Logical bug** in ID generation

### **Affected Code**

```
"id": len(tasks) + 1, # Problematic: not guaranteed to be unique
```

### **Fix Recommendation**

Use the provided `generate_unique_id(tasks)` function (already defined but unused):

```
from tasks import generate_unique_id
```

# Replace:

```
"id": len(tasks) + 1,
```

# With:

```
"id": generate_unique_id(tasks),
```

Task Apply filters not using available function

### **Environment**

- Python with Streamlit
- File-based task storage via JSON (tasks.json)

### **Preconditions**

- Multiple tasks have been added

### **Steps to Reproduce**

1. Add three tasks via the sidebar form.
2. Complete one of the tasks.
3. Uncheck show completed.

### **Expected Result**

- Only Incomplete tasks appear

### **Actual Result**

- Only incomplete tasks appear

### **Bug Type**

- **Integration bug** in Filter Application

### **Affected Code**

```
filtered_tasks = [task for task in filtered_tasks if not task["completed"]] # Problematic: not making use of available methods
```

## Fix Recommendation

Use the provided `filter_tasks_by_completion(tasks, completed)` function (already defined but unused):

```
from tasks import filter_tasks_by_completion
```

# Replace:

```
filtered_tasks = [task for task in filtered_tasks if not task["completed"]]
```

# With:

```
filtered_tasks = filter_tasks_by_completion(filtered_tasks, False)
```

## Pytest Features:

Pytest has many features that assist with testing. The ones that I used for this assignment were fixtures, parameterization, and coverage. Using a `pytest.fixture` let me create a single `sample_tasks` list that would be used in multiple tests, saving me from needing to redundantly create new test list for each test. Using parameterization let me test multiple values against the same test, once again preventing redundancy. Coverage allowed me to know how much of my code was being tested, so that I can ensure that everything I want tested gets tested.

Looking at the pytest using these features shows how they can make a difference. As can be seen in the figure below, parameterization allows for three test cases to generate ten different tests. This allows for the simple testing of many possible inputs so that the behavior of the method can be more readily confirmed.

```
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch) (4)> pytest -v tests/test_advanced.py
===== test session starts =====
platform linux -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
metadata: {'Python': '3.13.1', 'Platform': 'Linux-6.12.11-200.fc41.x86_64-x86_64-with-glibc2.40', 'Packages': {'pytest': '8.3.4', 'pluggy': '1.5.0'}, 'Plugins': {'xdist': '3.6.1', 'mock': '3.14.0', 'metadata': '3.1.1', 'bdd': '8.1.0', 'html': '4.1.1', 'cov': '6.1.1', 'anyio': '3.7.1'}}
rootdir: /home/jroy/Documents/Capstone/cs4090-assignment-4-jacob-roy
plugins: xdist-3.6.1, mock-3.14.0, metadata-3.1.1, bdd-8.1.0, html-4.1.1, cov-6.1.1, anyio-3.7.1
collected 10 items

tests/test_advanced.py::test_generate_unique_id[task_list0-1] PASSED [ 10%]
tests/test_advanced.py::test_generate_unique_id[task_list1-3] PASSED [ 20%]
tests/test_advanced.py::test_generate_unique_id[task_list2-11] PASSED [ 30%]
tests/test_advanced.py::test_filter_tasks_by_priority[High-2] PASSED [ 40%]
tests/test_advanced.py::test_filter_tasks_by_priority[Medium-1] PASSED [ 50%]
tests/test_advanced.py::test_filter_tasks_by_priority[Low-0] PASSED [ 60%]
tests/test_advanced.py::test_search_tasks[gym-expected_ids0] PASSED [ 70%]
tests/test_advanced.py::test_search_tasks[novel-expected_ids1] PASSED [ 80%]
tests/test_advanced.py::test_search_tasks[milk-expected_ids2] PASSED [ 90%]
tests/test_advanced.py::test_search_tasks[xyz-expected_ids3] PASSED [100%]

===== 10 passed in 0.01s =====
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)>
```

In the figure below, the coverage of both the basic and advanced test can be seen as 94%, meaning that 94% of the code in `task.py` has been tested and passed the test.

```
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)> pytest --cov=src tests/test_basic.py tests/test_advanced.py
===== test session starts =====
platform linux -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0
rootdir: /home/jroy/Documents/Capstone/cs4090-assignment-4-jacob-roy
plugins: xdist-3.6.1, mock-3.14.0, metadata-3.1.1, bdd-8.1.0, html-4.1.1, cov-6.1.1, anyio-3.7.1
collected 14 items

tests/test_basic.py ..... [ 28%]
tests/test_advanced.py ..... [100%]

===== tests coverage =====
_____ coverage: platform linux, python 3.13.1-final-0 _____

Name           Stmts   Miss  Cover
-----
src/app.py       85     85     0%
src/tasks.py     32      2    94%
TOTAL            117     87    26%

===== 14 passed in 0.05s =====
```

## Test-Driven Development:

For Test-Driven Development (TDD), the unit tests are developed first, and the functions are built so that they can minimally pass the test. This allows developers to avoid bloat when it comes to designing the functions. For this assignment, I came up with three new function ideas: `sort_task_by_due_date`, `duplicate_task`, and `reminder`. These functions are intended to add functionality to the application in the form of sorting task by date, duplicating a task given its id, and returning the ids of task due on a given date.

I started by writing the tests that define these functions' functionality. Then, I ran the test to confirm that they were functional. As expected, every test failed as the functions they are testing did not yet exist. This can be seen in the figure below.

```
===== FAILURES =====
test_sort_tasks_by_due_date

sample_tasks = [{'due_date': '2025-05-10', 'id': 1}, {'due_date': '2025-05-15', 'id': 2}, {'due_date': '2025-04-20', 'id': 3}]

def test_sort_tasks_by_due_date(sample_tasks):
    # test list with tasks
    sorted_tasks = [
        {"id": 3, "due_date": "2025-04-20"},
        {"id": 1, "due_date": "2025-05-10"},
        {"id": 2, "due_date": "2025-05-15"}
    ]

> result = tasks.sort_tasks_by_due_date(sample_tasks)
E   AttributeError: module 'tasks' has no attribute 'sort_tasks_by_due_date'

tests/test_tdd.py:26: AttributeError

test_duplicate_task

sample_tasks = [{'due_date': '2025-05-10', 'id': 1}, {'due_date': '2025-05-15', 'id': 2}, {'due_date': '2025-04-20', 'id': 3}]

def test_duplicate_task(sample_tasks):
    # test id in list
    duplicated_tasks = [
        {"id": 1, "due_date": "2025-05-10"},
        {"id": 2, "due_date": "2025-05-15"},
        {"id": 3, "due_date": "2025-04-20"},
        {"id": 4, "due_date": "2025-05-10"}
    ]

> result = tasks.duplicate_task(sample_tasks, id=1)
E   AttributeError: module 'tasks' has no attribute 'duplicate_task'

tests/test_tdd.py:46: AttributeError

test_reminder

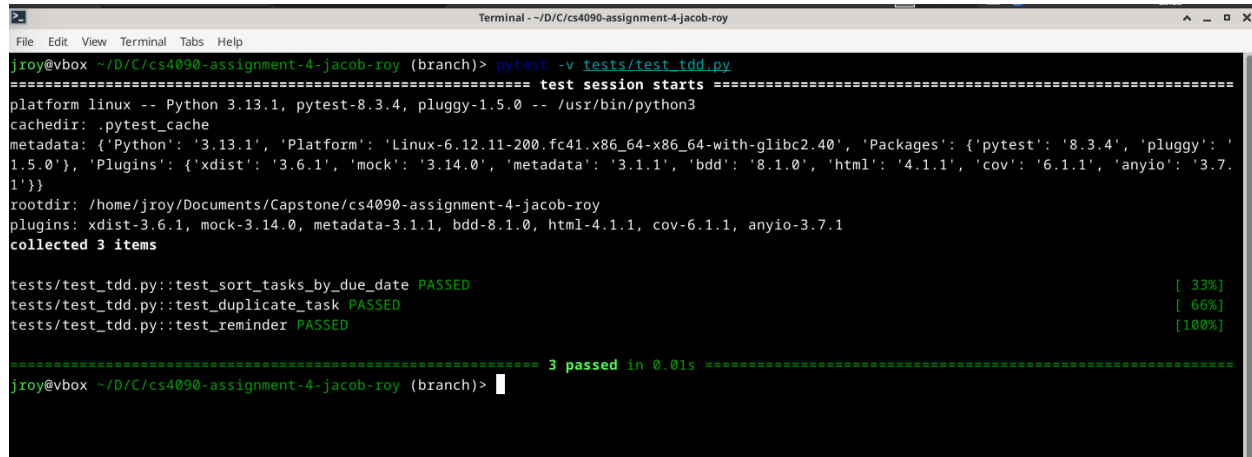
sample_tasks = [{'due_date': '2025-05-10', 'id': 1}, {'due_date': '2025-05-15', 'id': 2}, {'due_date': '2025-04-20', 'id': 3}]

def test_reminder(sample_tasks):
    # test date in list
> assert tasks.reminder(sample_tasks, date="2025-04-20") == [3]
E   AttributeError: module 'tasks' has no attribute 'reminder'

tests/test_tdd.py:59: AttributeError

===== short test summary info =====
FAILED tests/test_tdd.py::test_sort_tasks_by_due_date - AttributeError: module 'tasks' has no attribute 'sort_tasks_by_due_date'
FAILED tests/test_tdd.py::test_duplicate_task - AttributeError: module 'tasks' has no attribute 'duplicate_task'
FAILED tests/test_tdd.py::test_reminder - AttributeError: module 'tasks' has no attribute 'reminder'
3 failed in 0.04s
```

Then, I implemented the functions in a way that I hoped would minimally pass the already created test. After implementing them, I ran the test again and, as the figure below shows, passed the test. This means that I successfully made the functions in the way that I envisioned them and didn't need to go back and do refactoring.

A terminal window titled "Terminal - ~/D/C/cs4090-assignment-4-jacob-roy" showing a successful pytest execution. The user runs "pytest -v tests/test\_tdd.py". The output includes platform information (Linux, Python 3.13.1, pytest-8.3.4), metadata, and the results of three tests: "test\_sort\_tasks\_by\_due\_date PASSED", "test\_duplicate\_task PASSED", and "test\_reminder PASSED". The session concludes with "3 passed in 0.01s".

```
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)> pytest -v tests/test_tdd.py
===== test session starts =====
platform linux -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- /usr/bin/python3
cachedir: .pytest_cache
metadata: {'Python': '3.13.1', 'Platform': 'Linux-6.12.11-200.fc41.x86_64-x86_64-with-glibc2.40', 'Packages': {'pytest': '8.3.4', 'pluggy': '1.5.0'}, 'Plugins': {'xdist': '3.6.1', 'mock': '3.14.0', 'metadata': '3.1.1', 'bdd': '8.1.0', 'html': '4.1.1', 'cov': '6.1.1', 'anyio': '3.7.1'}}
rootdir: /home/jroy/Documents/Capstone/cs4090-assignment-4-jacob-roy
plugins: xdist-3.6.1, mock-3.14.0, metadata-3.1.1, bdd-8.1.0, html-4.1.1, cov-6.1.1, anyio-3.7.1
collected 3 items

tests/test_tdd.py::test_sort_tasks_by_due_date PASSED [ 33%]
tests/test_tdd.py::test_duplicate_task PASSED [ 66%]
tests/test_tdd.py::test_reminder PASSED [100%]

===== 3 passed in 0.01s =====
jroy@vbox ~/D/C/cs4090-assignment-4-jacob-roy (branch)>
```