STRUCTURAL TESTING WORKSHOP
SOFTWARE ENGINEERING II

Student:
Mario César Chalén Carvajal

Student ID:
201914264

Instructor's Name:
Mónica Villavicencio Cabezas, PhD

II PAO 2024

FIEC – ESPOL

Introduction

In this workshop, we will perform acceptance testing to a command-line application built in Python. The application is for managing tasks and allows basic operations such as adding new tasks, listing all tasks, marking tasks as completed, and clearing the entire list. To add persistence the script uses a json file to keep the data. This project also incorporates behavior-driven development (BDD) with Behave to test the application's features, ensuring it meets predefined user requirements effectively.

Development

Repository: https://github.com/MarioECU/acceptance-test-workshop

Part 1

Tool configuration evidence

```
mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ python3 -m venv venv
mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ l
total 0
drwxr-xr-x. 1 mchalen mchalen 56 Jan  9 15:21 venv
mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ source venv/bin/activate
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ pip install behave
Collecting behave
  Using cached behave-1.2.6-py2.py3-none-any.whl.metadata (6.4 kB)
Collecting parse>=1.8.2 (from behave)
  Using cached parse-1.20.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting parse-type>=0.4.2 (from behave)
  Using cached parse_type-0.6.4-py2.py3-none-any.whl.metadata (12 kB)
Collecting six>=1.11 (from behave)
  Downloading six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Using cached behave-1.2.6-py2.py3-none-any.whl (136 kB)
Using cached parse-1.20.2-py2.py3-none-any.whl (20 kB)
Using cached parse_type-0.6.4-py2.py3-none-any.whl (27 kB)
Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: parse, six, parse-type, behave
Successfully installed behave-1.2.6 parse-1.20.2 parse-type-0.6.4 six-1.17.0

[notice] A new release of pip is available: 23.3.2 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ l
total 0
drwxr-xr-x. 1 mchalen mchalen 56 Jan  9 15:21 venv
```

Code quality tools result

Flake8: For formatting and adherence to Python Standards.

Racon: For analyzing code complexity.

```
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ radon cc main.py -a
main.py
    F 84:0 main - B
    F 53:0 mark_task_completed - A
    F 41:0 list_tasks - A
    F 8:0 load_tasks - A
    F 71:0 clear_tasks - A
    F 17:0 save_tasks - A
    F 23:0 add_task - A

7 blocks (classes, functions, methods) analyzed.
Average complexity: A (2.857142857142857)
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ flake8 main.py
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$
```

Part 2 - Test run evidence
4 scenarios provided by the instructions

```
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ behave
Feature: To-Do List Manager # features/todo_list.feature:1

  Scenario: Add a task to the to-do list               # features/todo_list.feature:4
    Given the to-do list is empty                      # features/steps/todo_list_steps.py:19 0.001s
    When the user adds a task "Buy groceries"          # features/steps/todo_list_steps.py:37 0.001s
    Then the to-do list should contain "Buy groceries" # features/steps/todo_list_steps.py:69 0.000s

  Scenario: List all tasks in the to-do list  # features/todo_list.feature:9
    Given the to-do list contains tasks       # features/steps/todo_list_steps.py:23 0.001s
      | Task          |
      | Buy groceries |
      | Pay bills     |
    When the user lists all tasks             # features/steps/todo_list_steps.py:50 0.000s
    Then the output should contain            # features/steps/todo_list_steps.py:74 0.000s
      """
      Tasks:
      - Buy groceries
      - Pay bills
      """
      Assertion Failed: Expected: ['Tasks:', '- Buy groceries', '- Pay bills'], Got: ['- Buy groceries', '- Pay bills']


  Scenario: Mark a task as completed                                  # features/todo_list.feature:22
    Given the to-do list contains tasks                               # features/steps/todo_list_steps.py:23 0.000s
      | Task          | Status  |
      | Buy groceries | Pending |
    When the user marks task "Buy groceries" as completed             # features/steps/todo_list_steps.py:55 0.001s
    Then the to-do list should show task "Buy groceries" as completed # features/steps/todo_list_steps.py:81 0.000s

  Scenario: Clear the entire to-do list  # features/todo_list.feature:29
    Given the to-do list contains tasks  # features/steps/todo_list_steps.py:23 0.000s
      | Task          |
      | Buy groceries |
      | Pay bills     |
    When the user clears the to-do list  # features/steps/todo_list_steps.py:64 0.000s
    Then the to-do list should be empty  # features/steps/todo_list_steps.py:88 0.000s
```

2 new scenarios

```
  Scenario: Update the priority of a task                                    # features/todo_list.feature:38
    Given the to-do list contains tasks                                      # features/steps/todo_list_steps.py:23 0.001s
      | Task          | Description | Due Date   | Priority | Status  |
      | Buy groceries | Food items  | 2025-01-15 | Low      | Pending |
    When the user updates the priority of task "Buy groceries" to "High"     # features/steps/todo_list_steps.py:95 0.001s
    Then the to-do list should show task "Buy groceries" with priority "High" # features/steps/todo_list_steps.py:111 0.000s

  Scenario: Delete a specific task from the to-do list  # features/todo_list.feature:45
    Given the to-do list contains tasks                 # features/steps/todo_list_steps.py:23 0.000s
      | Task          | Description | Due Date   | Priority | Status  |
      | Buy groceries | Food items  | 2025-01-15 | Low      | Pending |
      | Pay bills     | Electricity | 2025-01-20 | Medium   | Pending |
    When the user deletes task "Pay bills"              # features/steps/todo_list_steps.py:104 0.000s
    Then the to-do list should not contain "Pay bills"  # features/steps/todo_list_steps.py:118 0.000s
```

Part 3 - Tests/code correction evidence

For the second scenario, it was a bit confusing to process the output of the list_tasks function but after some research I finally found a way to correctly evaluate the output and got it passed:

```
(venv) mchalen@fedora:~/Documents/ESPOL/SoftwareEngineeringII/AcceptanceTestWorkshop$ behave
Feature: To-Do List Manager # features/todo_list.feature:1

  Scenario: Add a task to the to-do list               # features/todo_list.feature:4
    Given the to-do list is empty                     # features/steps/todo_list_steps.py:11 0.001s
    When the user adds a task "Buy groceries"         # features/steps/todo_list_steps.py:29 0.001s
    Then the to-do list should contain "Buy groceries" # features/steps/todo_list_steps.py:69 0.000s

  Scenario: List all tasks in the to-do list  # features/todo_list.feature:9
    Given the to-do list contains tasks       # features/steps/todo_list_steps.py:15 0.001s
      | Task          |
      | Buy groceries |
      | Pay bills     |
    When the user lists all tasks             # features/steps/todo_list_steps.py:42 0.001s
    Then the output should contain            # features/steps/todo_list_steps.py:74 0.000s
      """
      Tasks:
      - Buy groceries
      - Pay bills
      """

  Scenario: Mark a task as completed                                # features/todo_list.feature:22
    Given the to-do list contains tasks                             # features/steps/todo_list_steps.py:15 0.000s
      | Task          | Status  |
      | Buy groceries | Pending |
    When the user marks task "Buy groceries" as completed           # features/steps/todo_list_steps.py:55 0.000s
    Then the to-do list should show task "Buy groceries" as completed # features/steps/todo_list_steps.py:82 0.000s

  Scenario: Clear the entire to-do list  # features/todo_list.feature:29
    Given the to-do list contains tasks  # features/steps/todo_list_steps.py:15 0.000s
      | Task          |
      | Buy groceries |
      | Pay bills     |
    When the user clears the to-do list  # features/steps/todo_list_steps.py:64 0.000s
    Then the to-do list should be empty  # features/steps/todo_list_steps.py:89 0.000s

  Scenario: Update the priority of a task                                  # features/todo_list.feature:38
    Given the to-do list contains tasks                                    # features/steps/todo_list_steps.py:15 0.001s
      | Task          | Description | Due Date   | Priority | Status  |
      | Buy groceries | Food items  | 2025-01-15 | Low      | Pending |
    When the user updates the priority of task "Buy groceries" to "High"   # features/steps/todo_list_steps.py:96 0.001s
    Then the to-do list should show task "Buy groceries" with priority "High" # features/steps/todo_list_steps.py:112 0.000s

  Scenario: Delete a specific task from the to-do list  # features/todo_list.feature:45
    Given the to-do list contains tasks                 # features/steps/todo_list_steps.py:15 0.001s
      | Task          | Description | Due Date   | Priority | Status  |
      | Buy groceries | Food items  | 2025-01-15 | Low      | Pending |
      | Pay bills     | Electricity | 2025-01-20 | Medium   | Pending |
    When the user deletes task "Pay bills"              # features/steps/todo_list_steps.py:105 0.001s
    Then the to-do list should not contain "Pay bills"  # features/steps/todo_list_steps.py:119 0.000s

1 feature passed, 0 failed, 0 skipped
6 scenarios passed, 0 failed, 0 skipped
18 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.008s
```

Conclusion

The To-Do List Manager successfully demonstrates the practical implementation of a task management system. Its functionality covers the essential features users need to track and organize tasks, while BDD testing ensures a user-focused and reliable experience. This project highlights the importance of modular programming, data persistence, and automated testing in building dependable software.

Recommendations

To enhance the To-Do List Manager, it is recommended to implement additional features such as task filtering by priority or due date, notifications for overdue tasks, and the ability to edit existing tasks. To improve acceptance testing for the To-Do List Manager, it is recommended to expand test coverage by including edge cases and negative scenarios, such as invalid inputs and data corruption. Automated execution of Behave tests in CI/CD pipelines will ensure continuous validation during development.

References

1. Behave Documentation. Retrieved from https://behave.readthedocs.io/
2. PEP 8 – Style Guide for Python Code. Python Software Foundation. Retrieved from https://peps.python.org/pep-0008/