Project Astrid: Logistics Robot for Moon Base Operations and Human-Robot Collaboration

OBJECT ORIENTATED PROGRAMMING

Table of Contents

Development

Overview

This project simulates the operation of a humanoid robot, designed for delivery tasks. The robot operates within a predefined environment, navigating a grid-like map to locate and reach destinations. The robot is designed to efficiently plans routes and interacts with its surroundings to successfully complete deliveries.

Features

Humanoid Robot Representation

The robot is designed following object-oriented programming principles to ensure modularity and expandability (Aniche, 2024). It simulates a functional delivery robot capable of decision-making and navigation.

Pathfinding with BFS

The robot uses the Breadth-First Search algorithm to calculate the shortest path as it is efficient in grid-based system (Cormen et al., 2001). This ensures efficient navigation across the environment, even when encountering obstacles or complex layouts.

Environment Simulation

The environment is modelled as a grid-based map.

Package Management

The robot handles different package types, including perishable, fragile and generic packages, each requiring specific handling.

Interactive User Interface

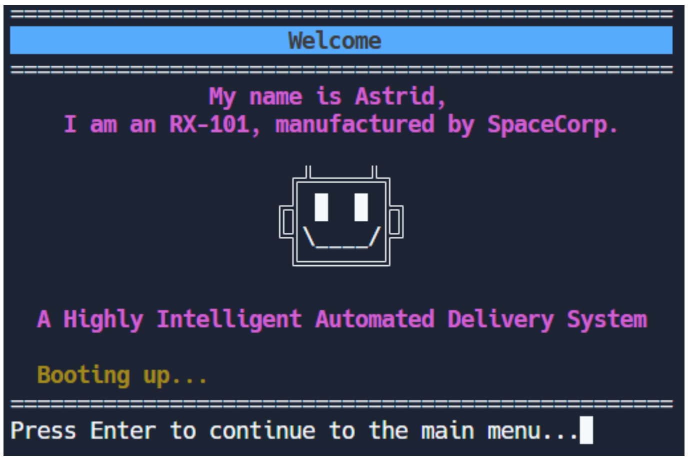
A command-line-based interface with a menu that allows the user to access the robot systems.

Testable and Modular Design

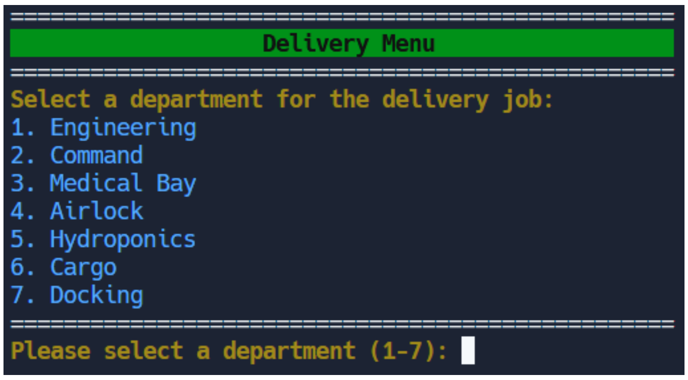
The codebase is designed with testability in mind. Unit tests validate core functionalities, ensuring reliability and robustness.

Adherence to PEP-8, PEP-257 and Best Practices

The implementation follows Python’s PEP-8 and PEP-257 style guide, ensuring readability and maintainability. (VanTol, No Date)



```
==================================================
                    Welcome
==================================================
              My name is Astrid,
   I am an RX-101, manufactured by SpaceCorp.

  A Highly Intelligent Automated Delivery System

  Booting up...
==================================================
Press Enter to continue to the main menu...
```



```
==================================================
                 Delivery Menu
==================================================
Select a department for the delivery job:
1. Engineering
2. Command
3. Medical Bay
4. Airlock
5. Hydroponics
6. Cargo
7. Docking
==================================================
Please select a department (1-7):
```

Code Structure

The project is organised into modules:

environment.py	interface.py	package.py
robot.py	utils.py	main.py

robot.py

Representing the humanoid robot:

- Initialisation of robot attributes.
- Package handling and delivery mechanics.
- Interaction with the environment.

environment.py

Models the robot's operational environment:

- Generating a grid-based map with predefined locations.
- Simulating barriers and paths within the environment.
- Providing the find_path method to calculate routes using the BFS algorithm.
- Visualising the environment through a string-based map display.

interface.py

Provides a command-line user interface for interacting with the robot:

- Displaying the robot's status and map.
- Accepting user input to define delivery destinations.

package.py

Provides specialised subclasses Perishable and Fragile. Using inheritance and polymorphism the code is easier to maintain and scalable (w3, no date):

- Unique package identification.
- Delivery speed simulation.

utils.py

Contains tools use in the system:

- Clear screen.
- Unique number generator.

main.py

Used to start the system:

- Instantiates a robot and environment class.
- Runs the interface.

Testing

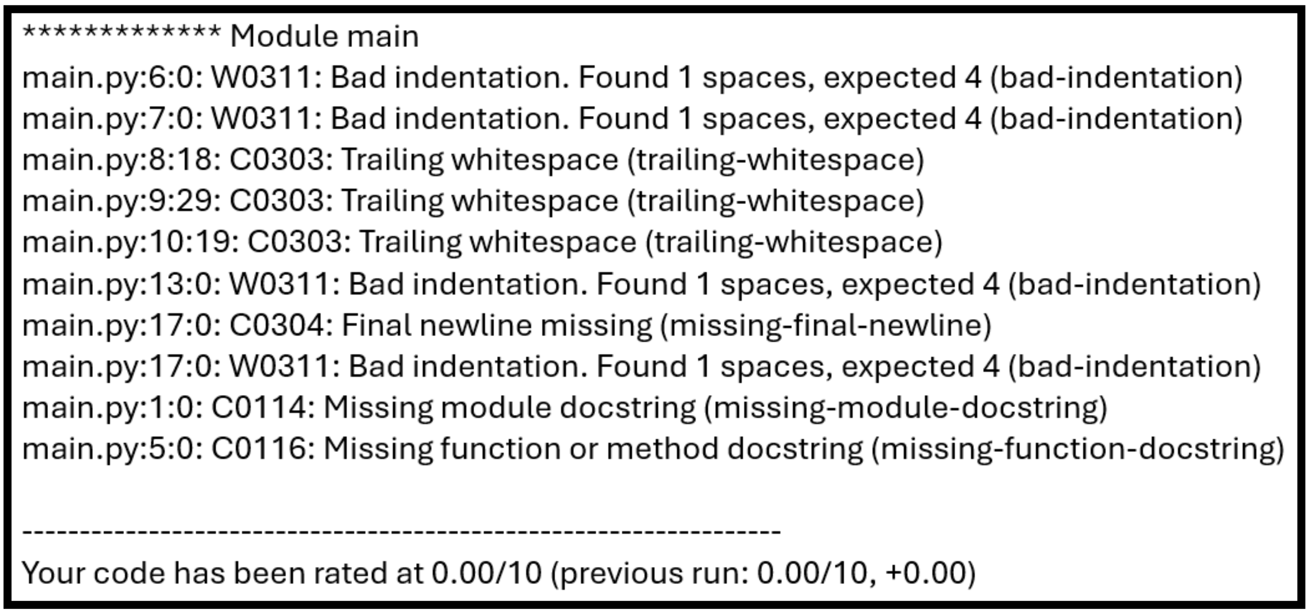
Testing was conducted using pylint and pytest to provide unit and integration testing.

Several issues related to consistency and missing comments were detected. Some of these issues were due to the historical use of tabs for indentation. These were corrected and replicated for the other files. The order of the imports was also incorrect as the system modules should come before the modules.

pylint also highlighted redundant else statements in the code, which could be simplified for better clarity and performance. Additionally, it pointed out unnecessary inheritance within the Package class. The inheritance was retained intentionally to allow for potential future expansion.

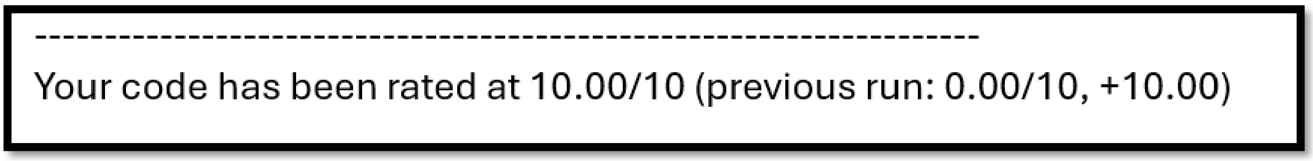
pylint

```python
from environment import Environment
from robot import Robot


def main():
  luna_9 = Environment()
  astrid = Robot(
      name = "Astrid",
      manufacturer = "SpaceCorp",
      model = "RX-101",
      environment = luna_9
  )
  astrid.start_up()


if __name__ == "__main__":
  main()
```



```
************* Module main
main.py:6:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
main.py:7:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
main.py:8:18: C0303: Trailing whitespace (trailing-whitespace)
main.py:9:29: C0303: Trailing whitespace (trailing-whitespace)
main.py:10:19: C0303: Trailing whitespace (trailing-whitespace)
main.py:13:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
main.py:17:0: C0304: Final newline missing (missing-final-newline)
main.py:17:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
main.py:1:0: C0114: Missing module docstring (missing-module-docstring)
main.py:5:0: C0116: Missing function or method docstring (missing-function-docstring)

------------------------------------------------------------------
Your code has been rated at 0.00/10 (previous run: 0.00/10, +0.00)
```

```python
"""
This script initialises and runs the humanoid robot simulation.

The program creates an environment and a robot instance, then starts the
robot's operations to perform tasks in the simulated environment.

Classes:
    Environment: Represents the environment the robot operates in.
    Robot: Represents the humanoid robot and its operations.

Functions:
    main(): Initialises the environment and robot and starts the robot's operations.
"""

from environment import Environment
from robot import Robot


def main():
    """
    Main function to initialise the environment and the robot,
    and start the robot's operations.
    """
    # Create the environment
    luna_9 = Environment()

    # Create the robot instance
    astrid = Robot(
        name="Astrid",
        manufacturer="SpaceCorp",
        model="RX-101",
        environment=luna_9
    )

    # Start the robot
    astrid.start_up()


if __name__ == "__main__":
    main()
```



```
------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 0.00/10, +10.00)
```

```
************* Module interface
interface.py:81:0: C0301: Line too long (111/100) (line-too-long)
interface.py:147:0: C0301: Line too long (112/100) (line-too-long)
interface.py:51:22: W1401: Anomalous backslash in string: '\_'. String constant might be missing an r
prefix. (anomalous-backslash-in-string)
interface.py:1:0: C0114: Missing module docstring (missing-module-docstring)
interface.py:82:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the
code inside it (no-else-return)
interface.py:134:16: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the
code inside it (no-else-return)
interface.py:2:0: C0411: standard import "sys" should be placed before first party import
"utils.clear_console"  (wrong-import-order)
-------------------------------------------------------------------
Your code has been rated at 9.03/10 (previous run: 0.00/10, +9.03)


--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.86/10, +0.14)
```

```
************* Module environment
environment.py:29:0: C0303: Trailing whitespace (trailing-whitespace)
environment.py:46:0: C0303: Trailing whitespace (trailing-whitespace)
environment.py:71:76: C0303: Trailing whitespace (trailing-whitespace)
environment.py:1:0: C0114: Missing module docstring (missing-module-docstring)
environment.py:1:0: C0115: Missing class docstring (missing-class-docstring)
environment.py:18:4: C0116: Missing function or method docstring (missing-function-docstring)
environment.py:21:4: C0116: Missing function or method docstring (missing-function-docstring)
environment.py:24:4: C0116: Missing function or method docstring (missing-function-docstring)
environment.py:53:4: C0116: Missing function or method docstring (missing-function-docstring)
-------------------------------------------------------------------
Your code has been rated at 8.00/10 (previous run: 0.00/10, +8.00)


--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.78/10, +0.22)
```

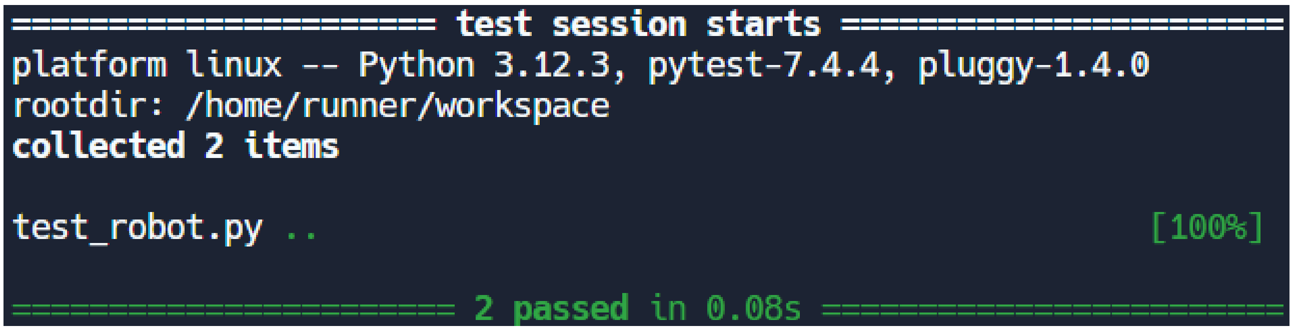
```
************* Module utils
utils.py:5:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
utils.py:6:0: W0311: Bad indentation. Found 2 spaces, expected 8 (bad-indentation)
utils.py:7:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
utils.py:8:0: W0311: Bad indentation. Found 2 spaces, expected 8 (bad-indentation)
utils.py:12:0: C0304: Final newline missing (missing-final-newline)
utils.py:12:0: W0311: Bad indentation. Found 1 spaces, expected 4 (bad-indentation)
utils.py:1:0: C0114: Missing module docstring (missing-module-docstring)
utils.py:4:0: C0116: Missing function or method docstring (missing-function-docstring)
utils.py:11:0: C0116: Missing function or method docstring (missing-function-docstring)
--------------------------------------------------------------------
Your code has been rated at 0.00/10


--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 0.00/10, +10.00)
```

```
************* Module robot
robot.py:50:0: C0303: Trailing whitespace (trailing-whitespace)
robot.py:58:17: C0303: Trailing whitespace (trailing-whitespace)
robot.py:62:0: C0303: Trailing whitespace (trailing-whitespace)
robot.py:68:70: C0303: Trailing whitespace (trailing-whitespace)
robot.py:69:51: C0303: Trailing whitespace (trailing-whitespace)
robot.py:71:42: C0303: Trailing whitespace (trailing-whitespace)
robot.py:1:0: C0114: Missing module docstring (missing-module-docstring)
robot.py:6:0: C0115: Missing class docstring (missing-class-docstring)
robot.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)
robot.py:21:4: C0116: Missing function or method docstring (missing-function-docstring)
robot.py:25:4: C0116: Missing function or method docstring (missing-function-docstring)
robot.py:34:4: C0116: Missing function or method docstring (missing-function-docstring)
robot.py:48:4: C0116: Missing function or method docstring (missing-function-docstring)
robot.py:4:0: C0411: standard import "time" should be placed before first party imports
"interface.Interface", "package.Package", "utils.generate_id"  (wrong-import-order)
-------------------------------------------------------------------
Your code has been rated at 7.41/10 (previous run: 0.00/10, +7.41)


--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 0.00/10, +10.00)
```

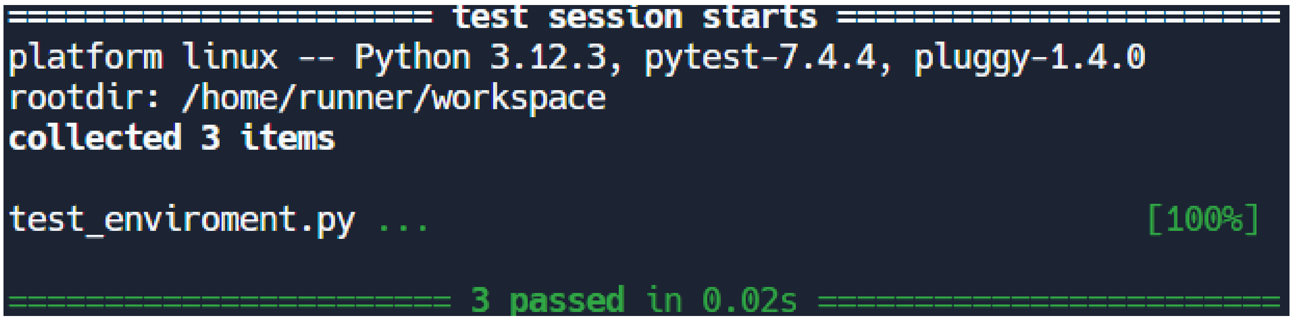
```
************* Module package
package.py:7:0: C0325: Unnecessary parens after 'return' keyword (superfluous-parens)
package.py:23:0: W0311: Bad indentation. Found 12 spaces, expected 8 (bad-indentation)
package.py:25:22: C0303: Trailing whitespace (trailing-whitespace)
package.py:25:0: W0311: Bad indentation. Found 12 spaces, expected 8 (bad-indentation)
package.py:26:0: C0303: Trailing whitespace (trailing-whitespace)
package.py:27:0: C0303: Trailing whitespace (trailing-whitespace)
package.py:30:0: W0311: Bad indentation. Found 12 spaces, expected 8 (bad-indentation)
package.py:32:0: W0311: Bad indentation. Found 12 spaces, expected 8 (bad-indentation)
package.py:1:0: C0114: Missing module docstring (missing-module-docstring)
package.py:1:0: C0115: Missing class docstring (missing-class-docstring)
package.py:10:4: C0116: Missing function or method docstring (missing-function-docstring)
package.py:14:4: C0116: Missing function or method docstring (missing-function-docstring)
package.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)
package.py:21:0: C0115: Missing class docstring (missing-class-docstring)
package.py:22:4: W0246: Useless parent or super() delegation in method '__init__' (useless-parent-
delegation)
package.py:28:0: C0115: Missing class docstring (missing-class-docstring)
package.py:29:4: W0246: Useless parent or super() delegation in method '__init__' (useless-parent-
delegation)
-----------------------------------
Your code has been rated at 2.27/10


************* Module package
package.py:81:4: W0246: Useless parent or super() delegation in method '__init__' (useless-parent-
delegation)
package.py:109:4: W0246: Useless parent or super() delegation in method '__init__' (useless-parent-
delegation)

-----------------------------------------------------------------
Your code has been rated at 9.09/10 (previous run: 9.13/10, -0.04)
```

pytest

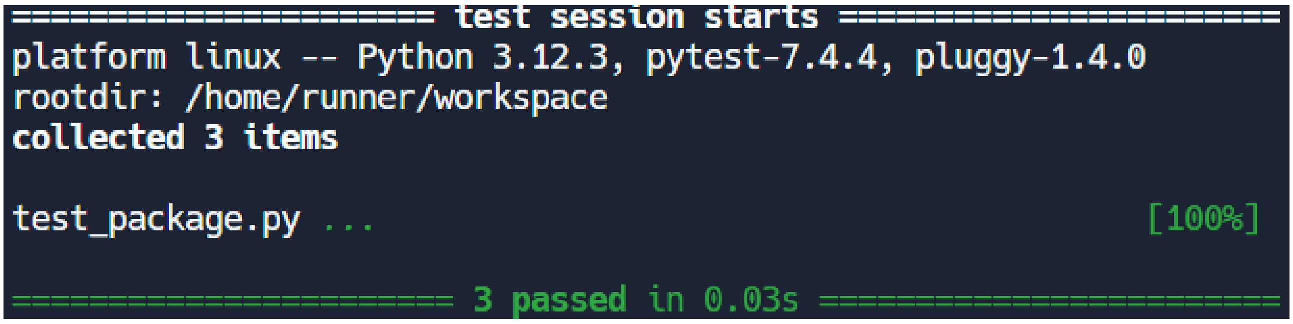
Test scripts written to be used with pytest the test cases are included in the testing folder. pytest was chosen for its simplicity and flexibility, particularly its ability to mock inputs using monkeypatch.



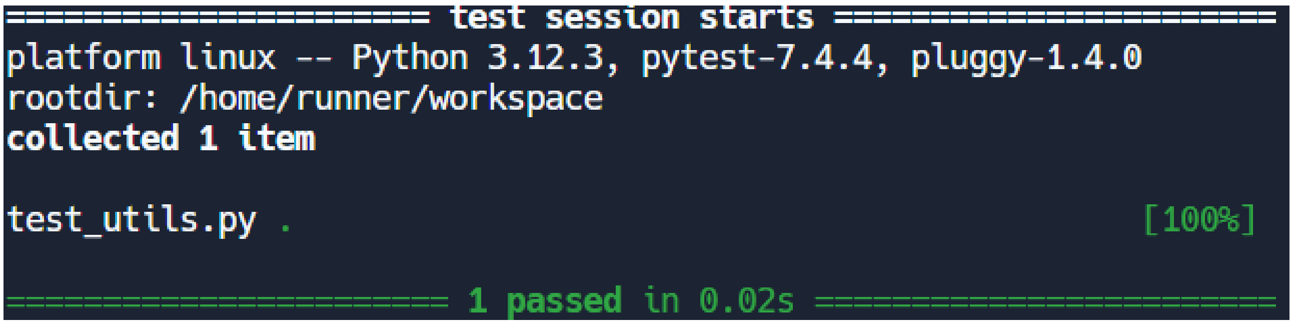
```
======================== test session starts ========================
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/runner/workspace
collected 2 items

test_robot.py ..                                              [100%]

========================= 2 passed in 0.08s =========================
```



```
======================== test session starts ========================
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/runner/workspace
collected 3 items

test_enviroment.py ...                                        [100%]

========================= 3 passed in 0.02s =========================
```



```
======================== test session starts ========================
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/runner/workspace
collected 3 items

test_package.py ...                                           [100%]

========================= 3 passed in 0.03s =========================
```



```
======================== test session starts ========================
platform linux -- Python 3.12.3, pytest-7.4.4, pluggy-1.4.0
rootdir: /home/runner/workspace
collected 1 item

test_utils.py .                                               [100%]

========================= 1 passed in 0.02s =========================
```

Instruction

Running the Simulation

Launch the main script:

python main.py

Follow the on-screen prompts to:

View the current map.

Select destinations.

Initiate and observe robot movements.

Customising the Environment

You can customise the robot's environment and behaviour by modifying the configuration parameters in the environment.py file:

Map Layout: Adjust the grid or locations.

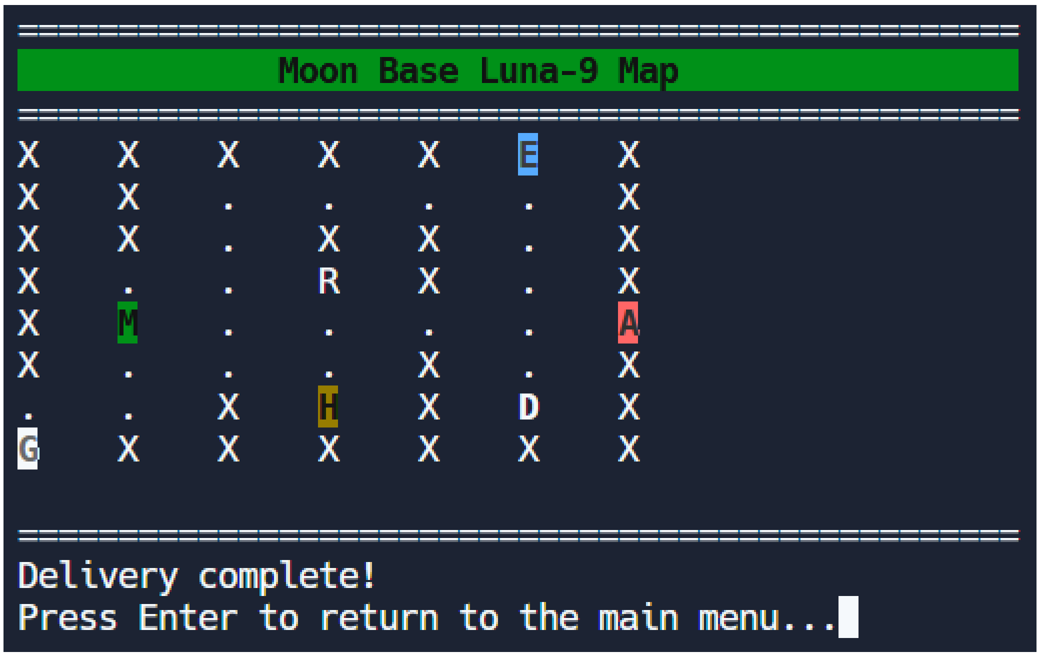
Destination Names: Change or add new points of interest.

Evaluation

The program provides an intuitive interface for users to interact with the robot. It supports tasks like package deliveries and can easily be extended to handle future tasks such as maintenance or repairs.

The robot's basic functionality includes navigation and delivery using BFS. Future improvements could include adding environmental factors and switching to A* for better efficiency.

The implementation successfully followed the design (Brayshaw, T. 2024), using OOP principles for maintainable and extensible code.



```
==================================================
                 Moon Base Luna-9 Map
==================================================
X    X    X    X    X    E    X
X    X    .    .    .    .    X
X    X    .    X    X    .    X
X    .    .    R    X    .    X
X    M    .    .    .    .    A
X    .    .    .    X    .    X
.    .    X    H    X    D    X
G    X    X    X    X    X    X

==================================================
Delivery complete!
Press Enter to return to the main menu...
```

References

Aniche, M. (2024) *Simple Object-Oriented Design : Create Clean, Maintainable Applications. 1st ed*. New York, NY: Manning Publications.

Brayshaw, T. (2024) *Project Astrid Design. OOP: Object Oriented Programming.* Report submitted to the University of Essex Online.

Cormen, Thomas H. Leiserson, Charles Eric. Rivest, Ronald L. Stein, Clifford. (2001) *Introduction to algorithms*. Cambridge, Mass.: MIT Press.

VanTol, A. (no date) *Python Code Quality: Tools & Best Practices*. Available at: https://realpython.com/python-code-quality/ (Accessed: January 18, 2025).

W3 Schools (no date) *Python Inheritance*. Available at: https://www.w3schools.com/python/python_inheritance.asp (Accessed: January 18, 2025).