

ENPM605: PYTHON APPLICATIONS FOR ROBOTICS

Assignment #1

v1.2

Lecturer: Z. Kootbally

School: University of Maryland

Semester/Year: Spring/2025



MARYLAND APPLIED
GRADUATE ENGINEERING

Table of Contents

🔗 Guidelines

🔗 Objectives & Skills Practiced

🔗 Package Structure

🔗 Deliverable

🔗 Task


🔗 Instructions

🔗 Tip

🔗 Example

🔗 Comments and Documentation

🔗 Grading Rubric

- ☐ **v1.2:** Removed the undo functionality from the grading rubric (slide 15).
- ☐ **v1.1:** Removed the undo functionality from slide 9. This may be a bit too advanced for newcomers.
 - ☐ The undo example was removed from  **assignment1_example.txt** and the file has been reuploaded.
- ☐ **v1.0:** Original version.

✱ Conventions

| | |
|---------------|---|
| bad practice |  |
| best practice |  |
| code syntax |  |
| example |  |
| exercise |  |
| file |  |
| folder |  |
| guideline |  |
| note |  |
| question |  |
| task |  |
| terminology |  |
| warning |  |
| web link | <u>link</u> |
| package |  |

| | |
|---------------|----------|
| ROS node | n |
| ROS topic | t |
| ROS message | m |
| ROS parameter | p |
| ROS frame | f |

✦ Guidelines





This assignment must be completed ***individually***. Ensure compliance with all specified guidelines, as failure to follow any part of these guidelines will lead to a grade of ***zero*** for the assignment.

- ☐ Do not reuse a package obtained from peers.
- ☐ Keep your work confidential and refrain from sharing it with peers.
 - ☒ If you use github, you have to make your repository private.
- ☐ While discussing assignment challenges is encouraged, refrain from exchanging solutions with peers.
- ☐ ***Do not use code generated by AI tools.***
 - ☐ ***The potential risks outweigh the benefits.***




✧ Objectives & Skills Practiced

- ☐ **Objectives:** The primary goal of this assignment is to strengthen your understanding of data structures and control flow in Python by creating an inventory management system. You will demonstrate your ability to:
 - ☐ Work with nested dictionaries to organize complex data.
 - ☐ Handle user input and implement data validation.
 - ☐ Process and filter data using dictionary operations.
 - ☐ Maintain data integrity through error checking.
 - ☐ Create basic reporting functionality.
 - ☐ Implement state tracking and data modification history.
- ☐ **Skills Practiced:** Nested dictionary manipulation, complex data structures, input validation, error handling, and data filtering.

✳ Package Structure

- ☐ Create a package named  ***RWA1_<last name>***¹ containing two files:  ***rwa1.py*** and  ***__init__.py***.
 - ☐ The package structure is shown below.
- ☐ Write the whole program in  ***rwa1.py***.

```

 RWA1_<last name>
├──  __init__.py
└──  rwa1.py
```

¹Replace ***<last name>*** with your actual last name.

✱ Deliverable

Compress (zip) the folder  **RWA1_<last name>** and upload it on Canvas by the due date.

- ☐ **Late submissions** will incur a penalty according to the guidelines specified in the syllabus, with exceptions for any valid reason.
 - ☐ Valid reasons include a doctor's note, proof of travel, or a note from a professor/MAGE.



Students with **special circumstances** may submit their assignments late without incurring any penalties. However, it is required that these students inform me in advance of their intention to submit their work past the deadline.



Task

Write a program that manages a store's product inventory using a nested dictionary structure. Each product has a name (`str`), price (`float`), quantity (`int`), category (`str`), and a list of suppliers (`list` of `str`). Implement a complete inventory management system.

✧ Instructions

- ❑ Start with a pre-filled nested dictionary containing at least 5 products across 3 different categories, each with multiple suppliers.
- ❑ Continuously prompt the user for operations: add product (**a**), remove product (**r**), edit details (**e**), search (**s**), report (**t**), or quit (**q**).
 - ❑ **For adding:** collect all product details and validate that prices are positive, quantities are non-negative, and the category exists.
 - ❑ **For removing:** allow removal only if the product quantity is >0 to prevent inventory errors.
 - ❑ **For editing:** allow updating multiple fields at once and maintain data consistency (e.g., can't set negative quantities).
 - ❑ **For searching:** implement filtering by price range, category, or supplier name.
 - ❑ **For reporting:** display total inventory value, products low in stock (< 5 units), and items per category.
 - ❑ Track the timestamp of the last modification for each product. There is no need to store the timestamp in the dictionary.
 - ❑ ~~Implement undo functionality for the last operation.~~
 - ❑ Use exception handling to manage invalid inputs and operations.




Tip

- ☐ Use a **while** loop to interact with the user and keep looping until the user decides to quit.
- ☐ To retrieve user inputs from the terminal, see the module `readline` and the function `input()`.
 - ☐ Be careful, `input()` returns a `str` and sometimes you need `float` in your program. We have seen in class how to convert a `str` to `float`.




-
- ☐ `input()`
 - ☐ `readline`

✖ Example

An example of terminal outputs is provided in  ***assignment1_example.txt***



 ***assignment1_example.txt*** demonstrates the expected output format, including user prompts and data displays. Your program should follow a similar output structure. You may either use the suggested product categories (Electronics, Books, Food) or create your own categories.

✧ Comments

Include comments in your program so that the user (me) understands what is happening.

✧ Documentation

Docstring documentation is not required for this assignment, unless you decide to use functions.

✧ Core Functionality (8 points)

☐ Initial Dictionary Setup (2 points):

- ☐ Correct implementation of nested dictionary structure with at least 5 products.
- ☐ Products properly distributed across 3+ categories.
- ☐ All required fields present (name, price, quantity, category, suppliers).

☐ Basic Operations (6 points):

- ☐ Add product functionality with proper validation (1.5 points)
- ☐ Remove product functionality with quantity check (1.5 points)
- ☐ Edit product details with consistency checks (1.5 points)
- ☐ Search/filter functionality (1.5 points)

✳ Data Management & Validation (5 points) ---

☐ **Input Validation (2 points):**

- ☐ Proper validation of numeric inputs (price, quantity).
- ☐ Category existence verification.
- ☐ Data type checking and conversion.

☐ **Error Handling (2 points):**

- ☐ Appropriate use of `try-except` blocks.
- ☐ Meaningful error messages.
- ☐ Graceful handling of invalid inputs.

☐ **Data Consistency (1 point):**

- ☐ Maintaining data integrity throughout operations.
- ☐ Proper updating of related fields.

✧ Advanced Features (4 points)

- ☐ **Timestamp Tracking (2 points):**
 - ☐ Implementation of modification tracking.
 - ☐ Accurate timestamp updates.
- ☐ **Undo Functionality (2 points):**
 - ☐ Working undo for last operation.
 - ☐ Proper state management.
- ☐ **Reporting Features (2 points):**
 - ☐ Total inventory value calculation.
 - ☐ Low stock reporting (< 5 units).
 - ☐ Category-wise item listing.

✧ Code Quality (3 points)

- ☐ **Code Organization (1 point):**
 - ☐ Proper package structure.
 - ☐ Clear file organization.
- ☐ **Comments/Documentation (1 point):**
 - ☐ Clear comments explaining logic.
 - ☐ Function documentation where applicable.
 - ☐ Readable code formatting.
- ☐ **User Interface (1 point):**
 - ☐ Clear user prompts.
 - ☐ Consistent output formatting.
 - ☐ User-friendly interaction flow.

✱ Deductions

- ☐ Late submission: *As per syllabus guidelines.*
- ☐ Missing package structure: *-2 points.*
- ☐ Use of AI-generated code: Assignment grade of *zero.*
- ☐ Code sharing/plagiarism: Assignment grade of *zero.*
- ☐ Non-private GitHub repository: Assignment grade of *zero.*