

# **VITYARTHI PROJECT**

**VELLORE INSTITUTE OF TECHNOLOGY, BHOPAL**



**Project Title:** Intelligent Waste Sorting Assistant (IWSA)

**Subject:** Introduction to Problem Solving

**Submitted To:** Pavithra Kannan

**Submitted by:** Chahak Gupta

**Registration Number:** 25BCE10138

# Introduction

Waste management is a critical environmental issue. Improper disposal of waste leads to pollution, health hazards, and inefficient recycling. Many people are unaware of which items are recyclable, organic, hazardous, or general waste.

The **Intelligent Waste Sorting Assistant (IWSA)** is a Python-based tool designed to help users quickly classify waste items and provide instructions for proper disposal. It serves as an educational and practical solution for households, schools, and environmental enthusiasts.

---

## Problem Statement

Improper waste segregation results in:

- Increased landfill pollution
- Contamination of recyclables
- Health hazards due to hazardous waste
- Inefficient recycling systems

There is a need for a simple, user-friendly solution that helps individuals classify waste items correctly and dispose of them responsibly.

**Solution:** IWSA provides keyword-based classification of waste items into categories and displays friendly disposal instructions.

---

## Functional Requirements

ID	Requirement	Description
FR1	Item Input	User can enter the name of a waste item.
FR2	Classification	Program classifies items as Recyclable, Organic, Hazardous, General, or Unknown.
FR3	Disposal Instructions	Displays clear instructions for each category.
FR4	Session History	Maintains a summary of all items sorted during a session.
FR5	Exit Functionality	User can exit program and view the session summary.

---

## Non-Functional Requirements

- **Performance:** Program responds immediately to user input.
- **Usability:** Friendly interface with clear prompts and instructions.
- **Scalability:** Keywords and instructions can be easily updated or extended.
- **Portability:** Runs on Windows, Mac, and Linux with Python installed.
- **Maintainability:** Modular code design for easy updates.

---

## System Architecture

### High-Level Architecture:

```
[User Input] --> [Waste Classification Module] --> [Instruction Module] --> [Display Module] --> [History Log]
```

- **User Input Module:** Accepts text input of waste items.
- **Classification Module:** Determines category using keyword matching.
- **Instruction Module:** Provides proper disposal guidance.
- **Display Module:** Shows results to the user.
- **History Module:** Stores session data for summary output.

---

## Design Diagrams

### Use Case Diagram

**Actors:** User, System

```
User --> Enter Waste Item
User --> View Disposal Instructions
User --> View Session Summary
System --> Classify Item
System --> Provide Instructions
System --> Maintain History
```

---

### Workflow Diagram

```
Start --> Input Waste Item --> Classify Item --> Display Instructions --> Save to History --> More Items? --> Exit --> Display Summary --> End
```

---

### Sequence Diagram

```
User -> System: Enter Item
System -> Classification Module: Determine Category
Classification Module -> Instruction Module: Get Instructions
Instruction Module -> System: Return Instructions
System -> User: Display Result
System -> History Module: Save Item
```

---

### Class/Component Diagram

## Classes/Components:

- Main – program driver
- ClassificationModule – determines category
- InstructionModule – returns disposal instructions
- HistoryModule – stores session data

Main

```
|-- ClassificationModule  
|-- InstructionModule  
|-- HistoryModule
```

---

## Design Decisions & Rationale

- **Keyword Matching:** Simple, fast, easy to maintain.
- **Modular Functions:** Each module handles a single responsibility.
- **CLI Interface:** Ensures portability and simplicity for beginners.
- **History Log:** Allows session summary for better tracking and testing.

---

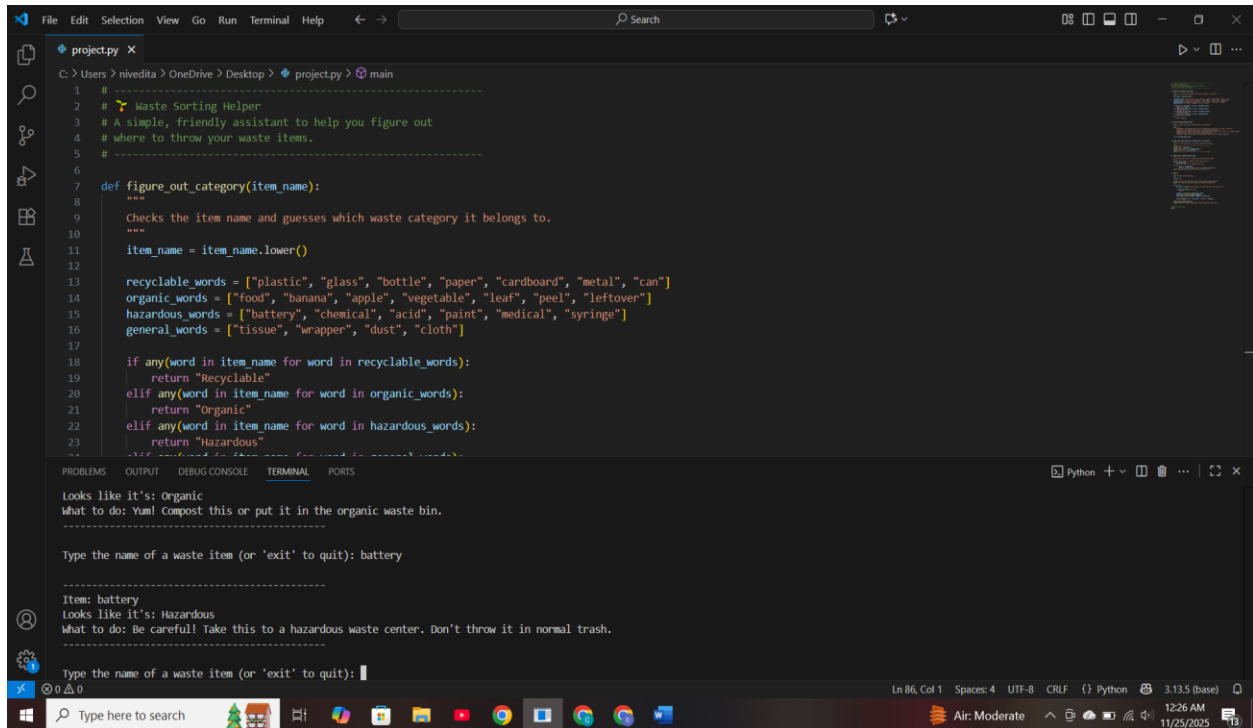
## Implementation Details

- **Language:** Python 3.x
- **Functions Used:** `figure_out_category()`, `how_to_dispose()`, `show_item_result()`, `show_overall_summary()`
- **Looping:** `while` loop for continuous user input
- **Data Structures:** Lists and dictionaries for keywords and instructions
- **Modularity:** Each functionality separated into functions for maintainability

## Example:

```
category = figure_out_category(user_input)  
instructions = how_to_dispose(category)  
show_item_result(user_input, category, instructions)
```

## Screenshots/Results



The screenshot shows a VS Code editor with a Python file named `project.py`. The script is a waste sorting helper that classifies items into categories: Recyclable, Organic, Hazardous, or General. The terminal output shows the program's execution flow for two inputs: 'Yum!' (classified as Organic) and 'battery' (classified as Hazardous).

```
1 # -----
2 # 🗑️ Waste Sorting Helper
3 # A simple, friendly assistant to help you figure out
4 # where to throw your waste items.
5 # -----
6
7 def figure_out_category(item_name):
8     """
9     Checks the item name and guesses which waste category it belongs to.
10    """
11    item_name = item_name.lower()
12
13    recyclable_words = ["plastic", "glass", "bottle", "paper", "cardboard", "metal", "can"]
14    organic_words = ["food", "banana", "apple", "vegetable", "leaf", "peel", "leftover"]
15    hazardous_words = ["battery", "chemical", "acid", "paint", "medical", "syringe"]
16    general_words = ["tissue", "wrapper", "dust", "cloth"]
17
18    if any(word in item_name for word in recyclable_words):
19        return "Recyclable"
20    elif any(word in item_name for word in organic_words):
21        return "Organic"
22    elif any(word in item_name for word in hazardous_words):
23        return "Hazardous"
24    else:
25        return "General"
```

Terminal Output:

```
Looks like it's: Organic
What to do: Yum! Compost this or put it in the organic waste bin.
-----
Type the name of a waste item (or 'exit' to quit): battery

Item: battery
Looks like it's: Hazardous
What to do: Be careful! Take this to a hazardous waste center. Don't throw it in normal trash.
-----
Type the name of a waste item (or 'exit' to quit):
```

## Testing Approach

**Objective:** Ensure the program correctly classifies waste items, provides accurate disposal instructions, and handles user input gracefully.

### Testing Methods:

- Unit Testing:**
  - Tested individual functions (`figure_out_category`, `how_to_dispose`) with various inputs.
  - Verified outputs match expected categories and instructions.
- Integration Testing:**
  - Tested the full program flow from user input to session summary.
  - Ensured all modules work together correctly.
- Boundary Testing:**
  - Tested unknown items to ensure the program handles them gracefully.
  - Checked case-insensitive inputs and plural forms.
- User Testing:**
  - Asked multiple users to enter real-life waste items.
  - Verified that instructions are clear and intuitive.

## Sample Test Cases:

Input	Expected Category	Expected Output
plastic bottle	Recyclable	Put this in your recycling bin. Make sure it's clean.
banana peel	Organic	Compost this or put it in the organic waste bin.
battery	Hazardous	Take to a hazardous waste center.
tissue	General Waste	Dispose in general waste bin.
unknown object	Unknown	Check guidelines manually.

---

## Challenges Faced

- **Keyword Limitations:** Some items are ambiguous (e.g., “plastic wrapper” vs “plastic bottle”).
- **Unknown Items:** Handling items not in keyword lists required clear fallback instructions.
- **User Input Variability:** Users may enter uppercase, lowercase, or misspelled items.
- **Scalability:** Adding new waste categories and items while keeping code maintainable.

## Solutions:

- Used **case-insensitive matching**.
  - Added **‘Unknown’ category** for unrecognized items.
  - Structured code modularly to allow easy keyword updates.
- 

## Learnings & Key Takeaways

- Learned **modular programming** and the importance of separating concerns.
  - Gained experience with **Python dictionaries, lists, and loops**.
  - Understood how to design **interactive CLI applications**.
  - Learned about **environmental awareness** and the importance of proper waste disposal.
  - Improved **problem-solving and debugging skills** through testing and refinement.
- 

## Future Enhancements

- **Graphical User Interface (GUI):** Use Tkinter or PyQt for a more interactive experience.
- **Voice Input/Output:** Allow users to speak the waste item and receive voice guidance.
- **Machine Learning Classification:** Use NLP to classify items intelligently beyond keyword matching.
- **Database Integration:** Store historical sorting data for analysis and reporting.

- **Mobile App Version:** Extend the project to Android/iOS for broader accessibility.
  - **Gamification:** Add rewards or points to encourage proper waste sorting.
- 

## References

1. Python Official Documentation – <https://docs.python.org/3/>
2. Environmental Protection Agency (EPA) – Waste Management Guidelines
3. Textbooks / Online Courses on Python Programming and Software Development