Automation of Order Documentation in E-Commerce Using Python

Rytis Bimbiras

DeepTech Entrepreneurship, 2025

Lecturers: Dr. Mindaugas Šarpis, Dr. Tadas Danielius

Submission date: 2025-04-06

1. INTRODUCTION

As part of the Artificial Intelligence and Data Analytics course, students were assigned a task to identify a problem in their real-world environment that could be automated using programming. The chosen challenge was the manual process of preparing order-related documents and communication for an e-commerce business. This process, though repetitive and time-consuming, is essential for day-to-day operations. Automating it reduces human error, increases efficiency, and enhances scalability.

The solution was implemented using Python, focusing on the application of fundamental programming practices. The following sections present the overall structure of the program, the reasoning behind key design choices, and a detailed explanation of how best practices in Python programming were applied to ensure clarity, maintainability, and extensibility.

2. PROBLEM OVERVIEW

Small and medium-sized online businesses often rely on manual workflows to handle orders, generating invoices, preparing shipping labels, resizing product images, and drafting confirmation emails. These processes, though straightforward, consume a significant amount of time and are prone to inconsistency or errors. As order volumes increase, such workflows become unsustainable and may affect customer satisfaction and internal efficiency.

Recognizing this issue, the goal was to automate the preparation of all necessary order documents, ensuring a more reliable and scalable approach.

3. SYSTEM DESIGN AND IMPLEMENTATION

3.1 Modular Programming

The system is built using a modular programming approach. Each distinct task such as PDF generation, image processing, data formatting, or Word document creation—is encapsulated within its own function. This modular structure enhances readability, simplifies debugging, and makes future updates more manageable. It also enables individual components to be reused or extended without affecting the core program.

3.2 Use of Informative Naming

One of the key strengths of the program is its use of clear, descriptive functions and variable names. Functions such as *generate_invoice_pdf*, *create_shipping_label*, and *resize_image* directly indicate their purpose. This improves the readability of the code and makes it easier to follow the logic, especially for future developers or collaborators.

3.3 Automation Workflow

The program operates in a logical sequence:

- Reads an Excel spreadsheet containing order data.
- Validates and formats necessary fields such as customer phone numbers, VAT codes, and addresses.
- For each order:
    - Creates a specific folder for attachments.
    - Generates a structured Word document summarizing order content.
    - Downloads and resizes associated product images.
    - Generates invoice PDFs and shipping labels automatically.
- Ensures that processed orders are marked to prevent duplication.

This automated workflow minimizes human interaction while ensuring consistency in document generation.

3.4 Python Best Practices

The program was developed by following key Python programming principles:

- **Function-Based Structure:** Tasks are divided into well-scoped functions that each serve a single purpose.
- **Error Handling:** Try-except blocks are used to manage exceptions (e.g. missing images or invalid data) without interrupting the entire process.
- **Data Validation:** Input data is carefully checked before use to prevent processing errors.
- **Code Reusability:** Helper functions are used across multiple parts of the program to avoid code duplication.
- **Consistent Style:** Adheres to the PEP 8 style guide, with consistent indentation, spacing, and naming conventions throughout the codebase.

These practices not only ensured the code ran correctly but also laid a foundation for long-term maintainability and further development.

4. FUTURE EXPANSION POSSIBILITIES

While the current version successfully automates order documentation, the structure of the program supports several logical directions for future development. Potential areas of expansion include:

4.1 Advanced Data Analytics

The system could be extended to analyze historical sales data, generate visual reports on product performance, identify seasonal trends, and provide decision support for inventory management.

4.2 Machine Learning Integration

Predictive models could be introduced to forecast future sales, detect anomalies in order patterns, or provide personalized product recommendations to customers based on past behavior.

4.3 Automated Customer Communication

The system could be integrated with email services to automatically send confirmation emails, invoices, and delivery updates to customers. This would reduce manual communication and improve the overall user experience.

5. CONCLUSION

The project presented a practical solution to a real-world challenge through the automation of order documentation in an e-commerce setting. The program was designed and developed using Python, applying best practices such as modular function design, informative naming, data validation, and structured error handling. These choices not only ensured reliable performance but also created a codebase that is clean, readable, and ready for future expansion.

This work highlights how structured programming and automation can significantly improve operational efficiency, even for individuals beginning their programming journey. It also demonstrates the potential for further development using analytics, machine learning, and web technologies to enhance business performance and intelligence.