Python Programming End of Term Project

Apply Python skills to design a functional program. Demonstrate mastery of core concepts including data structures, control flow, functions, OOP, and file handling.

Key Points

Objective

Apply Python skills to design a functional program. Demonstrate mastery of core concepts (data structures, control flow, functions, OOP, file handling).

Requirements

Individual project, Deadline: **Jan 9, 2026**.

Choose **Basic**, **Medium**, or **Expert** level based on your skill.

Assessment

Functionality (40%), **Code Quality (30%)**, **Creativity (20%)**, **Documentation (10%)**.

Submission

Email: tom.lotz@outlook.com, include **Name** and **Student ID**. Attach all relevant files.

Project Objective

The year-end project is designed to assess your ability to apply the Python programming skills you've learned throughout the semester. You will be tasked with designing and implementing a functional Python program that solves a real-world problem or fulfills a specific use case. The project will require you to demonstrate mastery of core Python concepts, including data structures, control flow, functions, object-oriented programming (OOP), and file handling.

The project allows for flexibility in terms of complexity. You can choose to focus on a basic implementation, develop a more complex system with added features, or challenge yourself with an advanced solution. Regardless of the level, your project should showcase your ability to write clean, well-organized, and modular Python code that is both efficient and easy to maintain.

Requirements

Individual Work

Each student has to submit their own project, no team work.

Deadline

January 9, 2026

Three Levels

The project has three levels of requirements: **Basic**, **Medium**, and **Expert**, allowing students to choose the complexity based on their current skill level.

- **Basic Level:** Covers essential Python concepts such as syntax, data types, and control flow. This level ensures a solid foundation for all students.
- **Medium Level:** For students comfortable with the basics, adding more advanced features like file handling and object-oriented programming to enhance your project.
- **Expert Level:** For students looking for a challenge, focusing on complex systems, external libraries, and advanced techniques.

Choose the level that matches your abilities, but feel free to start with a basic project and add more complexity as you improve.

Comprehensive Project Requirements by Level

Concept/Task	Basic	Medium	Expert
Python Syntax & Data Types	Use basic data types (integers, strings, booleans) and work with lists, dictionaries, and tuples.	Use nested data structures like lists of dictionaries or dictionaries of lists.	Work with complex data structures and external libraries like *pandas* for data manipulation or integration with APIs.
Control Flow & Loops	Implement basic conditional logic (if, else) and loops (for, while).	Use loops and conditions to create interactive programs. Implement flow control (break, continue).	Handle complex scenarios with multi-layered loops and conditional logic, such as nested loops and exception handling.
Functions	Define and use basic functions.	Organize code into well-defined functions, handle parameters and return values.	Modularize the program into reusable, testable functions. Include comprehensive error handling and validation.
File Handling	Not applicable.	Handle CSV or JSON files. Include basic file error handling.	Implement advanced file processing with structured formats (CSV, JSON) or integrate with a database for CRUD operations.
Error Handling	Basic error handling with try/except for general errors.	Implement custom exceptions and handle errors for user inputs.	Advanced exception handling, logging, and debugging techniques. Manage error recovery and ensure robustness in real-world use cases.

Comprehensive Project Requirements by Level

Concept/Task	Basic	Medium	Expert
Object-Oriented Programming (OOP)	Create simple classes with attributes and methods.	Design classes with inheritance or composition. Use classes to model real-world objects (e.g., users, tasks).	Build complex class systems with multiple levels of inheritance, polymorphism, and encapsulation. Use design patterns for better structure and scalability.
Modularization & Code Structure	Write simple, non-modular scripts.	Refactor code into functions to improve readability and maintainability.	Organize the code into multiple well-defined modules or packages. Ensure clear structure and proper separation of concerns.
User Input & Interaction	Handle user input through the console.	Build simple interactive programs with menus or basic command-line prompts.	Build complex user interfaces using libraries like Tkinter, or implement a GUI with interactive elements.
Advanced Features	Not applicable.	Optionally integrate libraries or APIs (e.g., requests for web interaction).	Integrate external libraries for added functionality (e.g., data visualization with matplotlib, web requests with requests, or databases).

Project Ideas

Here are some project suggestions based on different levels of difficulty. You can choose any project idea from this list, or come up with your own idea about something that is interesting to you.

Basic Ideas

- **Task Manager:** A simple app to add, remove, and view tasks.
- **Budget Tracker:** Track income and expenses, and display a summary.
- **Simple Library System:** Manage a list of books (add, remove, search).

Medium Ideas

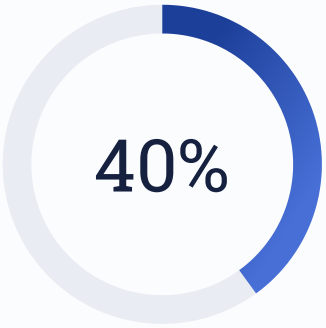
- **To-Do List with Categories:** Add, remove, and categorize tasks, store them in CSV files.
- **Contact Book:** Store, search, and delete contacts from a CSV file.
- **Student Grades Management:** Store grades and calculate averages or classifications.

Expert Ideas

- **Weather App:** Fetch weather data from an API and display it.
- **Personal Finance Tracker with Reports:** Track expenses and generate visual reports.
- **Inventory Management System:** Track products, sales, and stock levels, using a database for persistence.

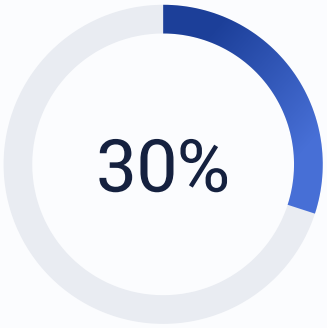
Assessment Criteria

Your project will be assessed based on the following:



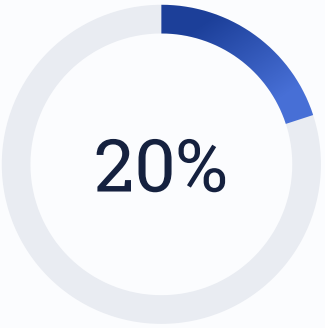
Functionality

Does the project meet the requirements and work as expected?



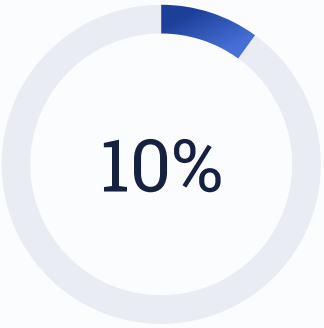
Code Quality

Is the code clean, modular, and well-commented?



Creativity & Complexity

How original or complex is the project? Does it incorporate advanced features?



Documentation

Is the project properly documented with clear instructions and explanations?

Expert projects may receive higher points for creativity and complexity due to their advanced techniques, but they will not automatically earn a higher overall score if the basic criteria are not met. The grading will focus on how well each project addresses the course objectives and demonstrates mastery of the concepts, regardless of the project's difficulty level.

If a basic project is executed very well, it can still receive the maximum score.

Submission Guidelines

01

Submit via Email

Submit your project via email to tom.lotz@outlook.com. Your email needs to mention your Name as on the course list, as well as your student ID number.

02

Include All Relevant Files

Include all relevant files: Python scripts, README.md, and any additional files (e.g., test cases, data files).

03

Document Your Project

Make sure your project is well-documented, including how to run it, its features, and any assumptions made.