**INTRODUCTION TO PROGRAMMING**

**Table of content**

[1. Introduction & Purpose of the Application 3](#_Toc184386843)

[2. Pseudocode 3](#_Toc184386844)

[3. Best Coding Practices & Standards 4](#_Toc184386845)

[4. User Guide & Output Screenshots 5](#_Toc184386846)

[5. Self-Reflection 10](#_Toc184386847)

[Conclusion 11](#_Toc184386848)

[Bibliography 12](#_Toc184386849)

# 1. Introduction & Purpose of the Application

This Hotel Management Application System is an easy console-based software that automatically allows the hotel to increase daily operations. Its main function is to mechanize routine tasks, by which hotel staff can do efficient management of room availability and customer bookings and thus billing processes. Users can easily add or remove rooms and view room details besides allocation of rooms to their customers according to their rooms' availability. The system enhances the efficiency of the hotel's operations by allowing the accomplishment of these tasks through an intuitive menu interface. Along with these core functionalities, it also supports file I/O operations that enable it to store and retrieve the details about the room allocation.

This means that all booking data persists even after closing the application, making it easier to track room availability and customer details over time. In addition, the application includes backup functionality to safeguard critical data (Finnie-Ansley *et al*. 2022). The copies of the room allocation file are created at regular intervals to protect against any losses in case the data goes wrong. The system further ensures reliability by implementing robust error and exception-handling techniques. This ensures that common problems, such as invalid inputs or file errors, are well handled, preventing crashes and generally improving the user experience. This results in a stable and efficient system that makes hotel management easier while also improving operational reliability.

# 2. Pseudocode

First, initialize a dictionary to hold room details. Display the hotel management menu and let the users select an option. Executing the appropriate function according to the user's input, and this process goes through an iteration until the user selects exit as the option.

To add a room, prompt the user for room details; check whether it already exists; and add it to the system. Delete room: Prompt room number, check whether the room exists; remove if the room exists. To view room details, iterate over all rooms displaying type, price, and allocation status.

Allocating a room requires prompting for the room number and customer name, checking its availability, and allocating it if possible. To display allocated rooms, identify rooms that are currently allocated and display their details along with customer names. Billing and de-allocation involve calculating the bill based on the number of nights stayed, displaying it, and marking the room as available for future bookings.

To save room allocation, write the details in a text file. Showing room allocation includes reading and displaying the contents of the file. The Backup makes a copy of the allocation file contents to a file with some timestamp, clearing out the original file after a backup.

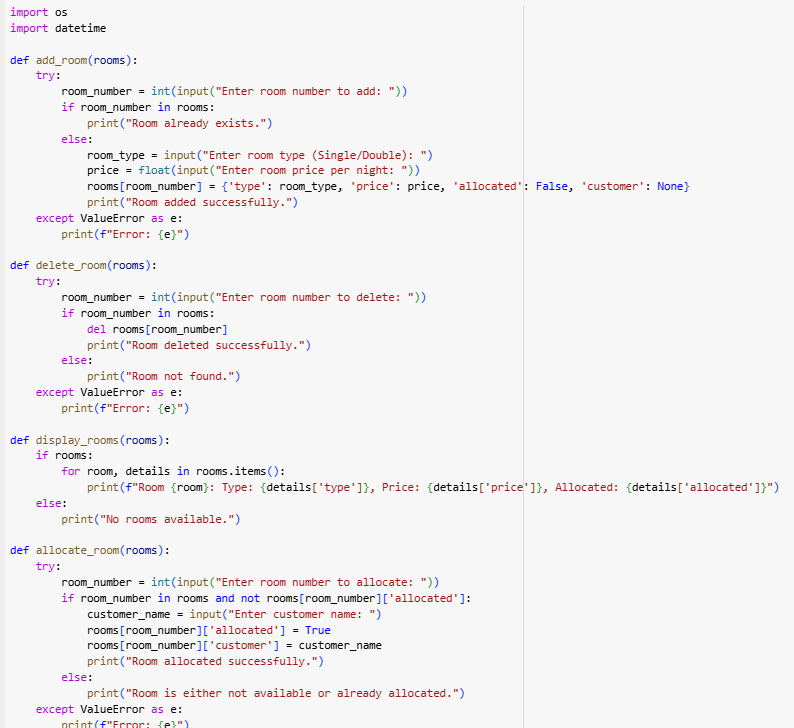
# 3. Best Coding Practices & Standards

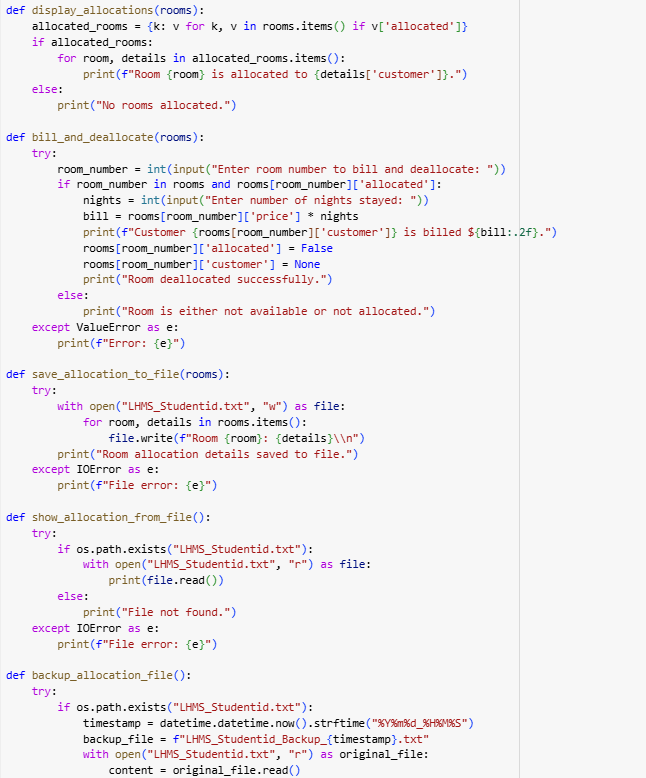
Comments are very important in the development of software because they explain the logic and structure of the code, hence making it easier for other people to understand and maintain. In this application, comments are used to ensure clarity. The script starts with a header comment that gives essential information, including the file name, author, purpose, and date of creation. This gives an overview of the application and its functionality. Inline comments are used throughout the code to explain specific logic, such as what happens at add, delete, and allocation of rooms. Inline comments help developers and anyone who may use the code again understand what the developer was trying to do on a line-by-line basis. Also, block comments are used to comment on the reasonableness of complicated segments in the program or overall functions for easy understanding of the flow and rationale related to individual design choices.

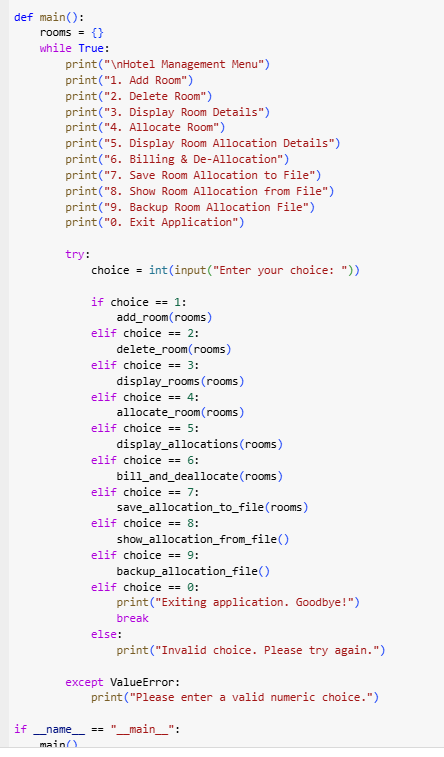
The proper naming convention is important for code readability and maintainability. Identifiers in this application use snake case consistently for variables and functions, like room details and add room, to make the names clear and descriptive so that developers can easily see what they do. Constants, however, are written in uppercase letters (like MAX\_ROOMS) to stand out from variables and functions. This naming convention follows Python’s best practices and makes the code more intuitive.

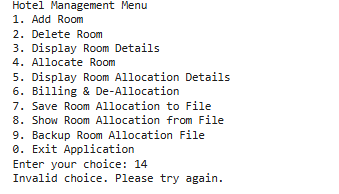
The code is consistently indented with four spaces per level, which follows a Python convention. Proper indentation makes it easier to read and makes the structure of the program visually clear. Additionally, it's logical, group-oriented, and modular for an application, which means maintaining it becomes much easier, as does scaling. Each function includes a docstring describing the purpose, parameters, and return values, giving a clear understanding of what each function does. For instance, the allocate room function has an elaborate docstring that explains how to use it and the expected inputs. Lastly, version control is implemented using GitHub where the project is hosted. This will allow the ability to track changes over time, manage collaborative contributions, and ensure the integrity of the codebase through commit history and change tracking.

# 4. User Guide & Output Screenshots









To use the Hotel Management Application, open the program in a Python environment, for example, PyCharm, VS Code, or in the terminal. The application begins with a menu of options. Each of the options relates to one particular operation that can be carried out: adding new rooms to the system, deleting existing rooms, showing details about rooms, allocating rooms to customers, and preparing bills for already served customers. Users can interact with the application by entering the corresponding menu number for the desired operation.

The application has advanced file-handling features to persist and back up data. Users can save the room allocation details to a text file, retrieve and view these details later, and create backups to ensure that data integrity is maintained. The backup feature appends the content of the allocation file to a separate backup file with a timestamp and clears the original file for the next session.

By using menu prompts, the users can complete hotel management operations easily. The user-friendly interface assists the user in walking through the operation step-by-step. Sample runs of the application are performed, showing capabilities, such as adding rooms, booking accommodation, saving the allocation data, and backing up data. Such sample runs display the effectiveness of the system in its capacity to support the normal, day-to-day activities of the hotel.

# 5. Self-Reflection

Developing the Hotel Management Application System required a focus on best coding practices to ensure the system's reliability, maintainability, and robustness. The primary consideration was to handle potential errors and exceptions gracefully so that the application would not crash under unexpected circumstances. This was achieved by implementing exception handling for common issues such as invalid input, file handling errors, and attempts to access non-existing rooms. By using error-handling techniques, the program ensures smooth operation even when the user provides incorrect or incomplete data.

File handling is a critical part of the application because it stores and retrieves room allocation data. One of the major enhancements was to ensure that files are closed after every I/O operation. This is to prevent data corruption and ensure that all changes are saved. For example, while saving room allocation details or creating backups, the application closes files right after the operation, so no actual file access issues or data loss can occur.

In doing so, the readability of an application is improved as well as maintainability through the segregation of the code into many, logically organized functions. Through this modular approach, codes are not only made easily readable and understandable but can also be reused easily in other parts of the systems. The add room and delete room operations allocate a room to students which was able to perform specific operations including that of add room, delete room, and allocate room functionality.

Testing was an important part of the design process. The application was tested using some edge cases, such as a room that is already booked being attempted to be allocated or the deletion of a room that does not exist. This allowed the identification of logical errors and corrected so that the system performed the exact way it was expected to in every respect (Sarsa *et al*. 2020). Thus, the final application developed would be robust and user-friendly, offering an efficient hotel operation solution.

# Conclusion

In conclusion, the Hotel Management Application System successfully automates the core operations of a hotel by improving efficiency in room availability, bookings, and billing. The application uses file handling to ensure data persistence and backup, which is essential for long-term operation. With the proper use of exception handling, the system gracefully handles errors and ensures a smooth user experience even in cases of unexpected inputs or system failure. Adopting best practices in coding, such as commenting, proper naming conventions, and modular design, promotes the maintainability and scalability of the application. It also makes a user-friendly interface and reliable functionality useful for staff working in hotels. Tests carried out during the testing phase proved the application's soundness so that all developed features worked properly. Indeed, this project exemplifies the value of using structured development approaches toward the creation of stable, efficient, and reliable hotel management solutions.

# Bibliography

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022, February). The robots are coming: Exploring the implications of openai codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference* (pp. 10-19). Doi: <https://dl.acm.org/doi/pdf/10.1145/3511861.3511863>

Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022, August). Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1* (pp. 27-43). Doi: <https://dl.acm.org/doi/pdf/10.1145/3501385.3543957>