Student name:

student id:

COIT20245 INTRODUCTION TO PROGRAMMING

Table of Contents

[Summary of Learnings: 2](#_Toc177143321)

[Strengths of the Implementation: 2](#_Toc177143322)

[Limitations of the Implementation: 2](#_Toc177143323)

[Potential Improvements: 3](#_Toc177143324)

# Summary of Learnings:

Through this assignment, I have gained a deeper understanding of designing and implementing a simple spreadsheet program that supports various data types and essential spreadsheet functionalities. The project was divided into three phases: basic operations and data structures, advanced functionalities, and user interface integration. This structured approach allowed me to incrementally build and test the spreadsheet program, ensuring that each phase was fully functional before moving on to the next.

I learned how to effectively manage data within a 2D list structure, which mimics a spreadsheet's grid layout. This required careful consideration of how to handle dynamic data entry, retrieval, and manipulation. Additionally, implementing core functions such as sum, average, min, max, sorting, and data type conversion provided valuable experience in performing operations on a grid of data. I also improved my skills in unit testing by validating each function individually, ensuring the program's reliability.

# Strengths of the Implementation:

1. **Modular Design:** The code is organized into clear, modular functions, each responsible for a specific operation. This separation of concerns makes the code easier to understand, maintain, and test.
2. **Comprehensive Functionality:** The program covers essential spreadsheet operations such as arithmetic calculations, sorting, and data type conversions. These operations are foundational to any spreadsheet application and are implemented efficiently.
3. **Interactive Command-Line Interface (CLI):** The CLI allows users to interact with the spreadsheet in a step-by-step manner. This interface simplifies testing and usage, especially for users unfamiliar with more complex interfaces.
4. **Error Handling and Type Conversion:** The program includes basic error handling and data type conversion functionalities. For instance, the convert\_to\_float function tries to convert cell values to floats and alerts the user if the conversion fails, ensuring data integrity.

# Limitations of the Implementation:

1. **Limited Data Structure Flexibility:** The current implementation uses a 2D list to represent the spreadsheet, which is straightforward but has limitations. For example, resizing the grid or handling sparse data could become inefficient as the spreadsheet grows in size.
2. **Basic Error Handling:** Although the program includes some error handling, it lacks robustness. For instance, operations like sorting or calculations do not fully account for edge cases, such as non-numeric data, which could lead to unexpected behavior or crashes.
3. **No Support for Complex Formulas:** While basic arithmetic operations are supported, the program does not allow for more complex formulas or functions within cells, limiting its functionality compared to modern spreadsheet applications.

# Potential Improvements:

1. **Enhanced Data Structure:** A more sophisticated data structure, such as a dictionary of lists or a sparse matrix, could improve performance, particularly for larger spreadsheets. This would allow for more efficient resizing, access, and manipulation of data.
2. **Improved Error Handling:** Implementing more comprehensive error handling would make the program more robust. For example, adding checks for valid data types during calculations and providing informative error messages would enhance usability.
3. **Support for Complex Formulas:** Introducing support for complex formulas and expressions would significantly increase the program's capabilities. This could involve parsing and evaluating expressions entered by the user.