Exercise 1: Arithmetic Expression Calculator

AISE501 – AI in Software Engineering I

Dr. Florian Herzog

Spring Semester 2026

1 Use Case

A user enters an arithmetic expression as a text string, for example `"3 + 5 * 2"`. The program evaluates the expression and prints the result.

The calculator must:

- Support the four basic operations: `+`, `-`, `*`, `/`
- Respect standard operator precedence (`*` and `/` bind more tightly than `+` and `-`)
- Support parentheses for grouping, e.g. `"(4 + 6) * 2"`
- Support decimal numbers, e.g. `"3.5 + 2.5"`
- Handle errors gracefully (division by zero, invalid characters, empty input)
- **Not** use Python's built-in `eval()` function

2 Example Input / Output

Input Expression	**Expected Output**
`3 + 5`	`8`
`10 - 2 * 3`	`4`
`(4 + 6) * 2`	`20`
`100 / (5 * 2)`	`10`
`3.5 + 2.5 * 4`	`13.5`
`(1 + 2) * (3 + 4)`	`21`
`(empty)`	Error message
`10 / 0`	Error message
`abc + 1`	Error message

3 Exercise

Two implementations are provided:

1. `calculator_bad.py` – A working but poorly written version that violates many clean code and PEP 8 principles.
2. `calculator_good.py` – A clean, well-structured version following PEP 8 and clean code best practices.

Tasks

1. Run both programs and verify they produce the same results.
2. Read the bad version and list all clean code / PEP 8 violations you can find.
3. For each violation, explain which principle is broken and why it makes the code harder to read or maintain.
4. Compare your list with the good version to see how each issue was resolved.

Violations to Look For

- Unused imports
- Missing or misleading comments and docstrings
- Poor variable and function names (abbreviations, single letters)
- Inconsistent indentation and spacing
- Multiple statements on one line (semicolons)
- Missing whitespace around operators
- No proper error handling (bare `except`, printing instead of raising)
- Magic numbers and unclear logic flow
- Missing `if __name__ == "__main__"` guard
- No type clarity in function signatures