Assignment: Horrible Code Activity
Deliverables: good_code.py, bad_code.py, and this PDF.

Program description
Both versions implement the same simple terminal calculator. Users choose an operator (+, -, *, /) or quit. The
program asks for two numbers, performs the operation, and prints the result.

Principles demonstrated
• **KISS** (Keep It Simple, Stupid): simple control flow and small functions.
• **DRY** (Don't Repeat Yourself): shared parsing and a single operator map instead of repeated blocks.
• **Single Responsibility**: each function does one job (parse input vs. compute).
• **Separation of Concerns**: business logic (math) is separated from the CLI loop.
• **Clean Code**: clear names, consistent formatting, and predictable error handling.

Good version (good_code.py): how it follows best practices
• Uses four tiny math functions (add/subtract/multiply/divide).
• Uses one dictionary (OPS) to map an operator to the correct function (DRY).
• Input validation is handled in dedicated functions (get_number, get_operator).
• divide() raises a clear error for division by zero; CLI catches and prints it.
• Main code is protected by if __name__ == "__main__": (safe imports).

Bad version (bad_code.py): how it violates best practices
• Everything is inside one large function (no single responsibility).
• Repeated try/except blocks for every operation (not DRY).
• Poor naming (doStuff, x, a, b, r) and vague messages ("??", "no").
• Silent/incorrect behavior: division by zero prints ans=0 instead of an error.
• Runs immediately on import (no main guard), causing unwanted side effects.

How to run
python good_code.py
python bad_code.py

Note: The two files intentionally produce similar outputs, but the "bad" version is written to be harder to maintain, extend, and debug.