Horrible Code Activity
Author: Ahmad Idris
Files included: bad_calculator.py, good_calculator.py
Program: Menu based calculator (add, subtract, multiply, divide, power, average)

Best-Practice Principles Selected
1) DRY
2) Single Responsibility
3) Separation of Concerns

Bad Version (bad_calculator.py): What It Violates
The bad one does so deliberately by copy/pasting the input, printing the results and. structure of the branch on all operations. The key functionality is breached in Single Responsibility. (doStuff) takes into consideration: the menu, user input, validation (or lack thereof), calculations, formatting and control flow. It is a contravention of Separation of Concerns in that business logic and UI are intertwined and make use of global variables. Other deliberate issues are non-specific names of functions/variables, magic numbers (division by 0.00001), recursion rather than a clean loop, irregular style, and little or no error handling.

Good Version (good_calculator.py): How It Follows Best Practices
The good one is a DRY version which stores operations in a single dictionary and uses a single shared flow to promptly take in the numbers, invoke the desired operation, and send out the results. This eliminates the same code used on each menu option. It is also based on Single Responsibility by dividing tasks into small focused functions: math functions only calculate and provide values (add, subtract), promptfloat only checks the input (numeric), printmenu only provides the menu and run_calculator is the control loop and interaction with the user. It is based upon Separation of Concerns and the separation of calculation logic (pure functions) and the UI. The CLI layer serves input/output as well as errors (division by zero) and does not clutter the math functions with such matters.