Global vs. Local Variables:

* Global variables are defined outside functions and accessible everywhere.
* Local variables are defined inside functions and exist only during function execution.
* Example: x = 3 (global), def foo(): y = 5 (local).

Function Scoping:

* Functions have their own local scope.
* Variables declared inside functions are not accessible outside.
* Use the global keyword to modify global variables inside a function.
* Example: global x allows modifying x inside a function.

Function Calls & Stack Behavior:

* A new stack frame is created when a function is called.
* Function calls another function, stack grows.
* When the function returns, stack shrinks.

Memory Model:

* Global variables are stored in the global frame.
* Local variables exist in function stack frames.

Nested Function Calls:

* Functions execute in a last-in, first-out (LIFO) order.
* Execution follows a structured sequence when calling nested functions.

Magic Numbers & Best Practices:

* Magic numbers are unexplained constants that reduce readability.
* Fix: Use named constants like TAX\_RATE = 1.15 instead of 100 \* 1.15.

Avoiding Bad Practices:

* Do not modify global variables in functions; return modified values instead.
* Pass global variables as arguments instead of using global.
* Avoid multiple functions modifying the same global variable.