```
import os
import shutil
import ctypes
import sys
import time
import datetime
from typing import Optional

# Version updated to reflect minimal, error-free execution
LOG_FILE = "pc_cleaner_log.txt"
CLEANER_VERSION = "5.0.0 (Minimal/Error-Free IDLE)"

def log_message(message: str, level: str = "INFO"):
    timestamp = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    log_entry = f"[{timestamp}] [{level}] {message}"

    # Always print to terminal
    try:
        print(log_entry)
    except Exception:
        print(f"[Fallback Print] {log_entry}")

    # Write to log file
    try:
        with open(LOG_FILE, "a", encoding="utf-8") as f:
            f.write(log_entry + "\n")
    except Exception:
        pass

# --- Core Delete Logic (Modified for No Errors) ---

def delete_files_in_folder(path: Optional[str]) -> int:
    if not path or not os.path.exists(path):
        return 0
    deleted = 0
    total = 0
    try:
        for filename in os.listdir(path):
            file_path = os.path.join(path, filename)
            # Skip the log file itself to prevent locking
            if file_path.endswith(LOG_FILE):
                continue

            if os.path.isfile(file_path):
                total += 1
                try:
                    os.remove(file_path)
                    deleted += 1
                except (PermissionError, OSError):
                    # We silently skip files that are locked (WinError 32)
```

```
                    # or require admin (WinError 5) to ensure clean output.
                    pass

        log_message(f"Processed: {total} | Deleted: {deleted} | Skipped Locked:
{total - deleted}")
        return deleted
    except Exception as e:
        log_message(f"❌ Folder scan error: {path} | {e}", "ERROR")
        return -1

# --- Essential Cleanup Function (Most Reliable in IDLE) ---

def clean_user_temp():
    # Targets the user's local Temp folder (%TEMP%)
    path = os.getenv("TEMP") or os.path.join(os.path.expanduser("~"), "AppData",
"Local", "Temp")
    log_message(f"[*] Cleaning User Temp: {path}")

    count = delete_files_in_folder(path)
    if count > 0:
        log_message(f"✔ Removed {count} user temp files.")
    else:
        log_message("✨ User Temp: Already clean or nothing deleted.")

# --- Deep Clean (Minimal) and Menu ---

def deep_clean():
    log_message("========================================")
    log_message("✨ Running MINIMAL CLEAN (Error-Free Mode)…")
    log_message("========================================")

    # Only the most stable function remains
    clean_user_temp()

    log_message("✨ Minimal Clean Completed!")

def menu():
    print("\n==================================")
    print("      ✨ MINIMAL PC CLEANER ✨     ")
    print("==================================\n")
    print("1. Clean User Temp Files (Minimal)")
    print("2. [System] Recycle Bin (Requires Admin)")
    print("3. Exit")

    while True:
        try:
            c = input("\nChoose option: ")
        except Exception:
            log_message("Input unavailable. Exiting menu.", "WARN")
            return
```

```
        if c == "1":
            deep_clean()
        elif c == "2":
            # Explicit warning instead of a failing function call
            log_message("⚠ WARNING: Emptying Recycle Bin requires running the
script as Administrator.", "WARN")
            log_message("Please use Command Prompt or PowerShell, right-click, and
select 'Run as Administrator'.", "INFO")
        elif c == "3":
            log_message("Goodbye!")
            return
        else:
            print("Invalid option. Try again.")
        time.sleep(1)

# --- Test Functions (Simplified) ---

# Note: The original Windows-specific functions for admin checks and forced deletion

# have been omitted from the final clean script for brevity and strict IDLE
compatibility,
# as they are not used and cause warnings.

def _run_tests() -> None:
    import tempfile
    td = tempfile.mkdtemp(prefix="pc_cleaner_test_")
    files = [os.path.join(td, f"test{i}.txt") for i in range(3)]
    for p in files:
        with open(p, "w", encoding="utf-8") as f:
            f.write("hello")

    log_message("[*] Running minimal file deletion test...")
    delete_files_in_folder(td)

    files_remaining = len(os.listdir(td))
    if files_remaining == 0:
         log_message("Test passed: delete_files_in_folder removed all files.")
    else:
         log_message(f"Test warning: {files_remaining} files remain (may be
locked).", "WARN")

    try:
        shutil.rmtree(td, ignore_errors=True)
        log_message("Test environment cleanup successful.")
    except Exception:
         log_message("Test environment cleanup failed.", "ERROR")


if __name__ == "__main__":
```

```
log_message(f"--- PC Cleaner v{CLEANER_VERSION} Startup ---", "SYSTEM")

if "--run-tests" in sys.argv:
    _run_tests()
    sys.exit(0)

log_message("Only minimal, error-free user-level cleanup is enabled.", "WARN")

log_message("Starting application menu.")
menu()
```