Title: Modifying Real-Time Data in the 'toppler64' Game using Python Script

This Python script is crafted to modify the live data of the 'toppler64' game, specifically targeting the player's lives count. The script leverages Python's os module and ptrace.debugger library to interact with the operating system and manipulate the game's process memory.

Outline of the Python Script

The script consists of several helper functions, each designed to perform a specific task, and a main block of code that ties these functions together to achieve the desired goal of modifying the player's lives count.

Helper Functions

get_pid_by_name: This function takes a process name as an argument and scans all running processes for a match. It uses the psutil.process_iter function to iterate over all processes, and if a matching name is found, it returns the corresponding Process ID (PID). If no match is found, it returns None.

read_memory: This function accepts a PID, a memory address, and a length as arguments. It creates a ptrace debugger, adds the specified process to the debugger, reads the specified number of bytes from the specified memory address in the process memory, and then quits the debugger. The read data is returned.

write_memory: This function takes a PID, a memory address, and some data as arguments. Like read_memory, it creates a ptrace debugger, adds the specified process to the debugger, and then writes the data to the specified memory address in the process memory. Afterward, it quits the debugger.

write_value: This function accepts a PID, a memory address, and a lives count as arguments. It converts the lives count into bytes (considering it as a signed integer) and calls write_memory to write these bytes to the specified memory address in the specified process's memory.

Main Code Block

The main code block starts by retrieving the PID of the 'toppler64' process via the get_pid_by_name function. If no process with the specified name is found, it prints an error message and the script execution ends. If the 'toppler64' process is found, the script modifies the lives count by calling the write_value function. This function is passed the PID, the memory address of the lives count (which has been determined through separate analysis), and the new lives count value, which in this case is set to 100.

Summary

In conclusion, this Python script demonstrates how to interact with a running process at the memory level to modify its state, specifically, changing the lives count of a game. It is important to use such techniques responsibly and ethically, as they could potentially be misused for malicious purposes.