**Instructions On Running the Program:**

To run this Python script, you will need to have the psutil library. You can install the library by opening a terminal and running this command: pip install psutil. After installing, be sure that you have python3 by typing the command: python3 –version. Once you confirm that you have python3, using the terminal in the same directory as this Word document, type in this command in the terminal to run the program: python3 Module4\_Activity.py. After executing the script, the program should display Real-time CPU & Memory Usage until you press Ctrl+C to stop the program.

**Functions In Retrieving CPU & Memory Information:**

Using the psutil library, we can extract the information for the CPU and memory. Psutil is a powerful tool for retrieving information on system utilization and running processes.

psutil.cpu\_count(logical=True) is a function that gets the total number of logical CPU cores on a system. Setting the parameter logical to True includes hyper-threaded cores, giving the total number of processing threads available.

psutil.cpu\_percent(interval=1)is a function that calculates the current CPU usage as a percentage from a specified time interval. We set the interval parameter to 1 to account for the usage in one second.

psutil.virtual\_memory()is another function in psutil that returns statistics about memory usage as a named tuple. From this tuple, the script accesses .total, which returns the total amount of physical memory (RAM) in bytes. I set the script to convert it to gigabytes for readability. .percent returns the percentage of memory currently in use.

**CPU and Memory Information Displayed:**

The program provides a real-time snapshot of the system’s resource utilization, which displays the required metrics for this assignment.

CPU Count displays the total number of logical processor cores available.

Total RAM displays the total amount of physical memory (RAM) installed on a system, where we set the program to measure gigabytes.

CPU Usage Percentage displays the percentage of the CPU’s processing power that is currently being used by different running tasks.

Memory Percentage displays the percentage of total RAM that is currently being used.

The program displays a real-time bar graph for CPU and RAM usage. I grabbed this inspiration from a Linux terminal program called htop.

**Real-World Implications:**

It is essential to monitor the CPU and memory usage to maintain the system’s health and performance. There are many applications, such as video games and web browsers, that require an extensive amount of resources. Monitoring System Performance provides an immediate way of seeing the current workload that the computer is struggling with. It can help in optimizing the computer hardware system, or show the user that the computer system is not powerful enough for the application being performed.

Diagnosing Resource Bottlenecks can show that a component, like the CPU or memory, is limiting the overall performance of the system. For instance, if the CPU usage is greatly increased, it could mean there is a CPU bottleneck and RAM usage is low. In comparison, RAM can also be a bottleneck for the system if RAM usage is high and the system is forced to rely on slower storage for temporary data.

Optimizing Application Resource Usage is a great way for developers to determine how their application or deployment environment functions under different loads. If there is an unexpected increase in CPU or memory usage, it may indicate a that there is a performance issue within the application being tested, such as an inefficient algorithm or memory leak that needs to be addressed.

**Influence on Operating System Scheduling:**

CPU and memory usage are definite factors that influence how an operating system’s kernel handles scheduling algorithms and resource allocations. In terms of Physical Memory, the operating system manages physical memory by allocating it to processes in chunks. When a process needs more memory than what is physically available, the operating system uses virtual memory. This involves a swap that moves inactive data from RAM to a designated area on the hard drive, which frees up memory for active processes. This process can significantly slow down performance due to the speed difference between the RAM and secondary storage.

Logical memory is a reference to the memory space of a process, which is usually larger than the physical memory available. The operating system maps this logical memory to the physical memory. Processes that are memory-intensive or CPU-intensive can influence the kernel’s scheduler to give them more resources or pause other processes to prevent the hardware system from becoming unresponsive. The goal of the scheduler is to balance the needs of all the running processes to ensure the computer system's stability and responsiveness.

**Difficulties and Challenges:**

While creating this program, there were several challenges that I encountered and some considerations that I made. The first being, I wanted to do something like htop for some real-time monitoring with psutil. I did some research and found ANSI escape codes like \033[F to generate visual bars without flooding the terminal with new lines. The code would move the cursor up two lines and overwrite the previous output. This might not work with every terminal, though, in terms of display issues coming from it. The other challenge was implementing some error handling, like having a try-except block to gracefully handle the KeyboardInterrupt when the user presses Ctrl+C. Not having this would cause the program to terminate abruptly, which isn’t exactly user-friendly. Overall, I’m glad that I was able to do something more with psutil compared to the other programming activities.