For Task 1, the code segment is a function that takes in the path of the security log, which in this case I chose a linux security log, and then it goes through the file and looks for where the file has the phrase "authentication failure" and then adds that into the anomaly log that was created. Then it creates a report as a file called summary_logs.txt, that first says the total amount of suspicious activity, then writes each log within the anomaly log into the summary_logs.txt file. There were no libraries imported. It detected any failed login attempts and wrote it to a new file called summary_logs.txt.

For Task 2, the code segment is a function that looks at the current cpu and memory usage with an interval of 5 seconds and then outputs what the percentage is for both of them, then it opens a file called performance_data.txt and then puts that information onto there as well. Then, the metrics being used here for problems or alerts is if either the cpu usage or memory usage is greater than 85%, in which it outputs an alert saying that the percentage is too high. It also opens a file called performance_alert.txt in which it puts that information onto there as well. Since this is a program to monitor performance, there is an infinite loop since the performance should be checked constantly, and would need manual interruption to end the program. The psutil library was imported in order to obtain the cpu and memory percentage.

For Task 3, the code segment is a function that takes in the subject and the body of the email that the user wishes to send, and it creates an email with a specific starting email address and a destination email address, in this case a group member's two emails, and then it logs in to the starting email and sends the message. This function is a partner function with Task 2, as the performance_monitoring function calls on the send_alert function whenever an alert (in this case CPU and memory > 85) is needed. The psutil, smtplib, and EmailMessage libraries were imported in order to receive the performance percentages and the emails. Whenever the CPU or memory was higher than the threshold, an email was sent depending on what the alert was to our group member’s email.

For Task 4, the code segment imports the nmap module and then creates an initialization of an Nmap PortScanner, called nm and then initializes a target, called target which can either be the target IP or the hostname depending on what you have. Then a scan is done by using the scan function that takes in the previously defined target, with ports 22-80. Then it prints all the hosts of the machine with the state it's in, and then prints the protocol, port, and state to conduct an effective scan. The nmap library was imported in order to conduct the vulnerability scans. The results of this script showed the hosts on the network, and the state that they were in. It also showed the protocols, ports, and the states of those on the network.