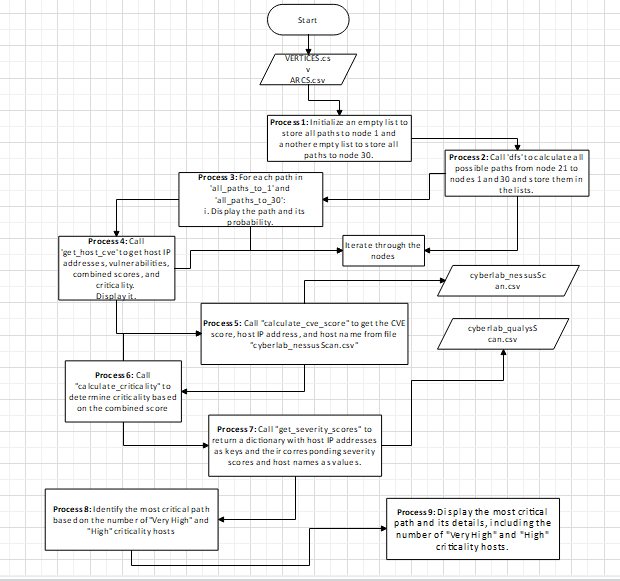
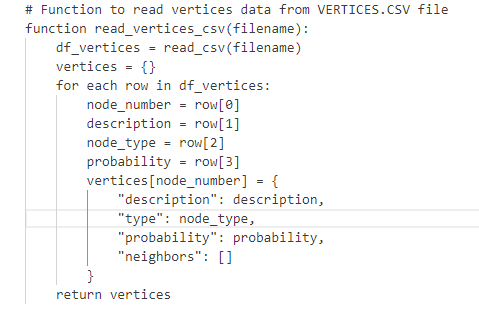
**Smart Grid Risk Assessment And Advanced Alert Notification System Report**

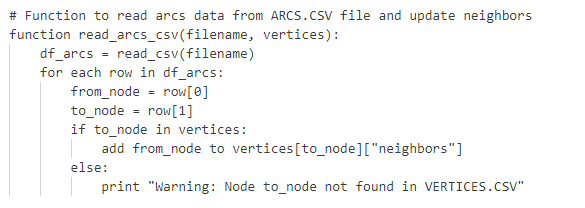
* ****

**Program Flowchart**

* **Process0 - Reading Data Files:**
  + The script starts by reading two CSV files: VERTICES.CSV and ARCS.CSV.
  + The read\_vertices\_csv() function reads the VERTICES.CSV file using pandas and stores the node information in a dictionary called vertices. Each node is represented by its number, and its details, such as description, type, probability, and neighbours, are stored as values in the dictionary.
  + The read\_arcs\_csv() function reads the ARCS.CSV file using pandas and updates the neighbour information for each node in the vertices dictionary. It establishes a directed graph relationship between nodes.



**Reading VERTICES.CSV**



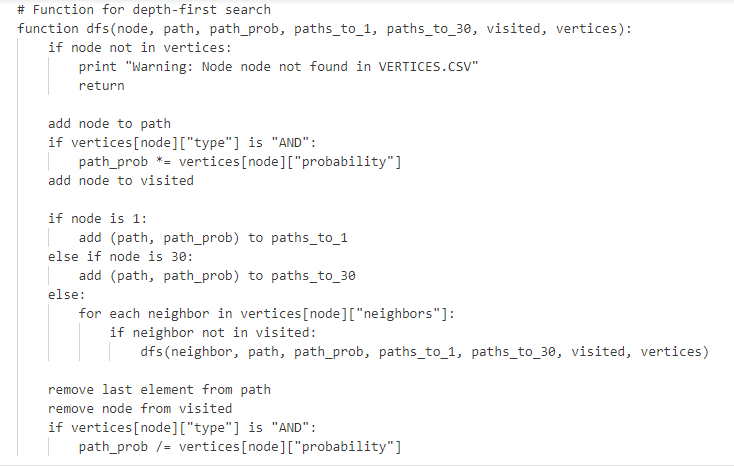
**Reading ARCS.CSV.**

* **Process 1 - Initialization Lists:**
  + Initialise an empty list to store all paths to node 1 and another empty list to store all paths to node 30.

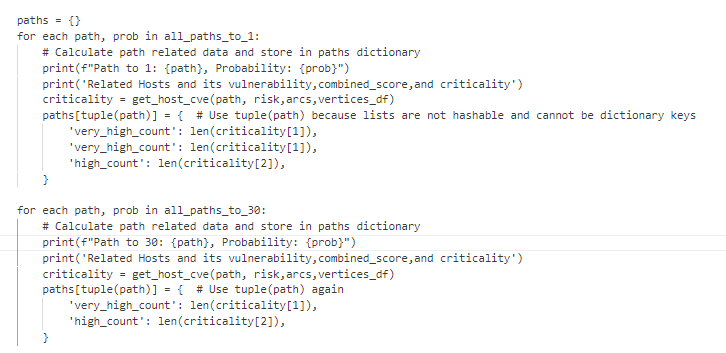
****

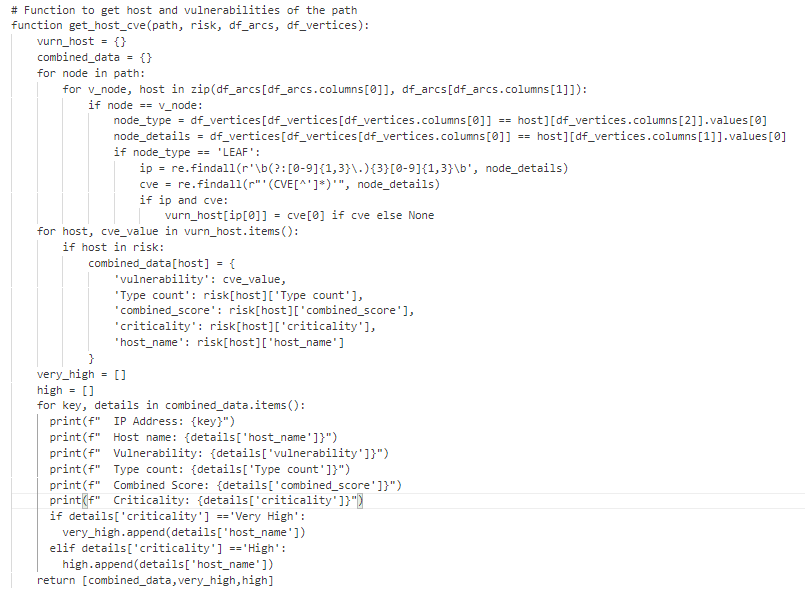
**Initialising Lists**

* **Process 2 - Depth-First Search (DFS):**
  + The script performs a depth-first search (DFS) to find all possible paths from a starting node (of type "AND") to two target nodes with numbers 1 and 30.
  + The dfs() function is a recursive implementation of the depth-first search algorithm. It explores all paths from the starting node while keeping track of visited nodes and probabilities along each path.
  + The function maintains two lists, all\_paths\_to\_1 and all\_paths\_to\_30, to store the paths to the target nodes and their probabilities.



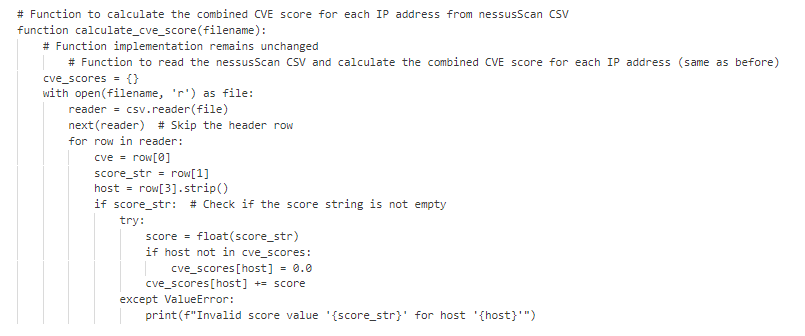
**DFS**

* **Process 3 - Displaying Paths:**
* For each path in 'all\_paths\_to\_1' and 'all\_paths\_to\_30',Display the path and its probability.
* 
* **Process 4 - Getting Hosts and Vulnerabilities for Paths:**
  + The get\_host\_cve() function is used to get the hosts and their vulnerabilities along a specific path. It extracts the relevant information from the df\_arcs and df\_vertices DataFrames based on the nodes in the path.



**get\_host\_cve()**

* **Process 5 - Calculating Vulnerabilities and Scores:**
  + The calculate\_cve\_score() function calculates the combined CVE (Common Vulnerabilities and Exposures) score for each IP address from the cyberlab\_nessusScan.csv file. It aggregates vulnerability scores associated with each IP address.



**calculate\_cve\_score()**

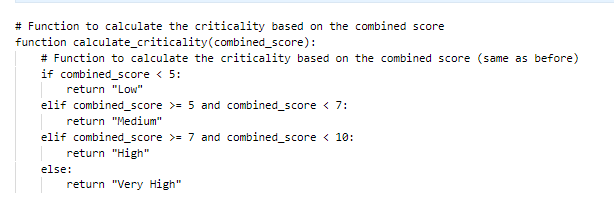
* The script reads additional CSV file: cyberlab\_nessusScan.csv



**Reading cyberlab\_nessusScan.csv**

**Process 6 - Calculating Criticality**

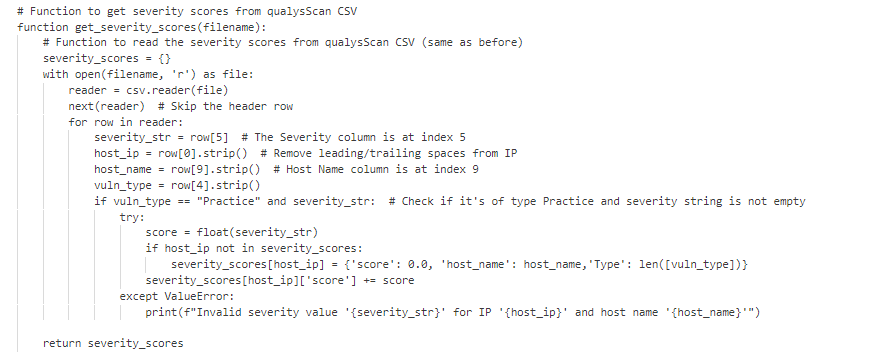
* Call "calculate\_criticality" to determine criticality based on the combined score



**Calculating Criticality**

**Process 7 - Get Severity Scores**

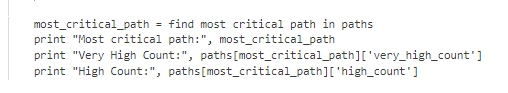
* + The get\_severity\_scores() function retrieves severity scores for practice-type vulnerabilities from the cyberlab\_qualysScan.csv file.



**get\_severity\_scores()**

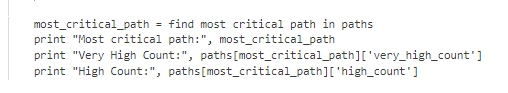
**Process 8 - Finding the Most Critical Path**

* + The script evaluates all paths to find the most critical one based on the counts of "Very High" and "High" criticality nodes along each path. The results are stored in the paths dictionary.
  + The script prints the most critical path, the count of "Very High" criticality nodes, and the count of "High" criticality nodes.



**Finding the Most Critical Path**

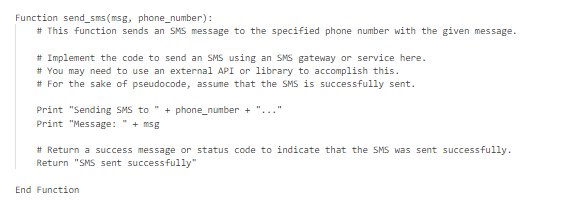
* **Process 9 - Printing Results:**
  + The script prints the most critical path and its probability to node 1 and node 30, if available.



* **Printing Results**

**Process 10 - Sending SMS Alerts**

* The send\_sms() function:
  + This function represents the basic functionality to send an SMS message to a given phone number with a specific message.
  + In the actual implementation, you would need to use an SMS gateway or service to send the SMS. The pseudocode assumes that the SMS sending is successful for the sake of simplicity.



**send\_sms()**

* The send\_sms\_to\_specific\_persons(details) function:
  + This function is responsible for determining which person(s) to send the SMS based on the criticality and hostname of the device (details).
  + It first defines phone numbers for different recipients, such as IT Security Officers, System Admin, and Operator.
  + The function then uses conditional statements (if, elif, else) to check the criticality and hostname to decide which SMS messages to send.
  + If the hostname is in ['SCADA1', 'SCADA2', 'workstation', 'webserver']:
    - If the criticality is 'Very High', it sends an SMS to IT Security Officers with a specific message.
    - If the criticality is 'High', it sends an SMS to the System Admin with a different message.
    - If the criticality is 'Medium', it sends an SMS to the System Admin with a different message.
    - If the criticality is 'Low', it sends an SMS to the System Admin with a different message.



**First Case**

* + If the hostname is in ['PLC1', 'PLC2']:
    - If the criticality is 'Very High', it sends an SMS to both the Operator and IT Security Officers with specific messages.
    - If the criticality is 'High', it sends an SMS to both the Operator and IT Security Officers with specific messages.
    - If the criticality is 'Medium', it sends an SMS to the Operator with a different message.
    - If the criticality is 'Low', it sends an SMS to the Operator with a different message.



**Second Case**

The pseudocode outlines the logic for determining which recipients to send SMS messages to based on the criticality and hostname, and it demonstrates the format of the messages to be sent. You would need to integrate this logic into your actual code and adapt it to work with an SMS service to send the SMS messages effectively.

Overall, the script aims to analyse a graph structure, represented by nodes and edges in the CSV files, and perform a depth-first search to find the most critical path based on combined vulnerability scores and severity scores. It provides insights into the risk associated with different paths in the graph, as well as information about hosts, vulnerabilities, and criticality levels.