Bubble Sort Algorithm Cross & Comparative Analysis (REPORT)

Project Information:

Coding the Bubble Sort Algorithm using Python & Javascript and making comparisons using both these languages for this task.

Algorithm Information:

Bubble sort is a simple sorting algorithm.
This sorting algorithm is a comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.
This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n2) where n is the number of items.

Objectives/Goals:

To build skills using computer programming data structures, data types, logic and other program specific properties.

COMPARISONS:

Javascript	Python
- JS arrays (Data Structure) - JS For Loop Knowledge - Counter Variable - JS Logic (Type Coercion) - JS if-else statements - JS Developer Console	- Python List (Data Structure) - Python For Loop Knowledge - Counter Variable - Python Logic (No type coercion) - Python if-else statements - Python Exception Handling Try/Except Clause - Python range function

- JavaScript const **keyword** differs in that assignments CANNOT be changed whereas the **let** keyword enables value to be re-assigned. So, once you declare a variable using the const keyword, you cannot re-assign a value to that variable. This does not mean that the variable is immutable. If you assign an object to a variable using the const keyword, you can mutate that object. You just can't re-assign that variable with a new value.

Challenges

- Python ran into issues regarding the indexing while using the **range function** which was used to compare the two adjacent elements. This was the main challenge faced using Python for this task. We had to build and implement a **TRY & EXCEPT** clause and finally resetting the index to return to the beginning of this list.
- We had to build and implement a counter variable into the algorithm to manually stop the algorithm meaning we manually set the number of iterations essentially this means we will manually set when the algorithm will STOP. We did not have this issue in Javascript as described above.

Review

- Python was difficult to successfully complete the sorting algorithm task (strict) however it was easier to interpret the process of building the algorithm on review & reflection. The Anaconda Navigator Jupyter Notebook was very useful for readability.
- JS was much easier to successfully complete the sorting algorithm task (loose). Primarily the use of the JS for loop which a counter variable is required for this code. This was the main feature which enabled the algorithm to automatically stop in comparison to Python, which is one the challenges that was faced using this language.
- Python code output analysis was definitely more efficient for this task to analyze the different iterations of the code. Due to the Jupyter Notebook setup this was an advantage.