A bubble sort algorithm

Task .

An implementation of a bubble sort in Python is shown in **Figure 1**. Read through the code to familiarise yourself with it - don’t worry if you don’t understand all of it yet.

| 1  2  3  4  5  6  7  8  9  10 | def bubble\_sort(items):  # Initialise the variables  num\_items = len(items)  passes = 1  # Repeat while the maximum numbers of passes has not been made  while passes < num\_items:  # Repeat for each pair of items  for current in range(num\_items - 1):  # Compare the item at the current position with the next item  if items[current] > items[current+1]:  # Swap the out-of-order items  temp = items[current]  items[current] = items[current+1]  items[current+1] = temp  # Increase the number of passes by 1  passes = passes + 1 |
| --- | --- |

**Figure 1**

The following questions will be based on executing the algorithm in **Figure 1** when items is the list: **[“Maya”, “Dan”, “Vivian”, “Tobi”, “Areeji”]**

Examine line 5 and **state** how many times the inner loop is performed on the list above i.e. how many pairs of items does every single pass examine.

|  |
| --- |

Examine line 4 and **state** how many times the outer loop is performed on the list above i.e. how many passes does the algorithm make.

|  |
| --- |

**Complete** the trace table below only for lines 7-9 of the algorithm. The first line in the trace table contains the values for the current variable and the items list.

|  |  |  | items | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Line | current | temp | [0] | [1] | [2] | [3] | [4] |
|  | 0 | - | Maya | Dan | Vivian | Tobi | Areej |
| 7 |  |  |  |  |  |  |  |
| 8 |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |

**Explain** the purpose of lines 7-9 in the bubble sort algorithm in **Figure 1**.

|  |
| --- |

What happens when line 10 is omitted from the algorithm in **Figure 1**?

|  |
| --- |