A bubble sort algorithm - Solution

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.

| 4: The range will start at 0 and end at 4 (5-1) exclusive of the end value. |
| --- |

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.

| 4: The condition passes < num\_items evaluates to 1 < 5 in the first instance, with passes incrementing by 1 after each iteration. |
| --- |

**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 |  | Maya |  |  |  |  |  |
| 8 |  |  | Dan | Dan | Vivian | Tobi | Areej |
| 9 |  |  | Dan | Maya | Vivian | Tobi | Areej |

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

| A temporary variable called temp is used to store the value at the current index before replacing the value with the next element in the list in line 8. Then in line 9 the temp value is stored in the next element so that the values have swapped places without losing the element at the current index. |
| --- |

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

| The value of passes would stay as 1 if line 12 is omitted. This means that the condition of the while loop will always be True as long as the items list contains one or more elements, resulting in an infinite loop. |
| --- |