Chapter 5.1: Searching and Sorting Algorithms

NAME:

Learning Outcomes

- **Understand why searching and sorting is important in our programs**
- **Understand and explain how a Bubble Sort works**

Why Search and Sort?

Searching and sorting are some of the most common tasks we need to do in our programs. For example.

- In a game we will need to sort the leaderboard into an order.
- Google search results need to sort the results with the most relevant first.
- Online shopping you search for a specific item using the menus or keywords. You can sort the products by various criteria for example, Price, Relevance, Distance From Home.

Key Terms

- **Searching:** Looking through a file to see if a particular piece of data is present.
- **Sorting:** Putting items of data into a precise order, for example alphabetical or numerical.
- **Ascending:** Small to large eg **1, 2, 3, 4** or **a, b, c, d**
- **Descending:** Large to small eg **4, 3, 2, 1** or **d, c, b, a**

Sorting Algorithms

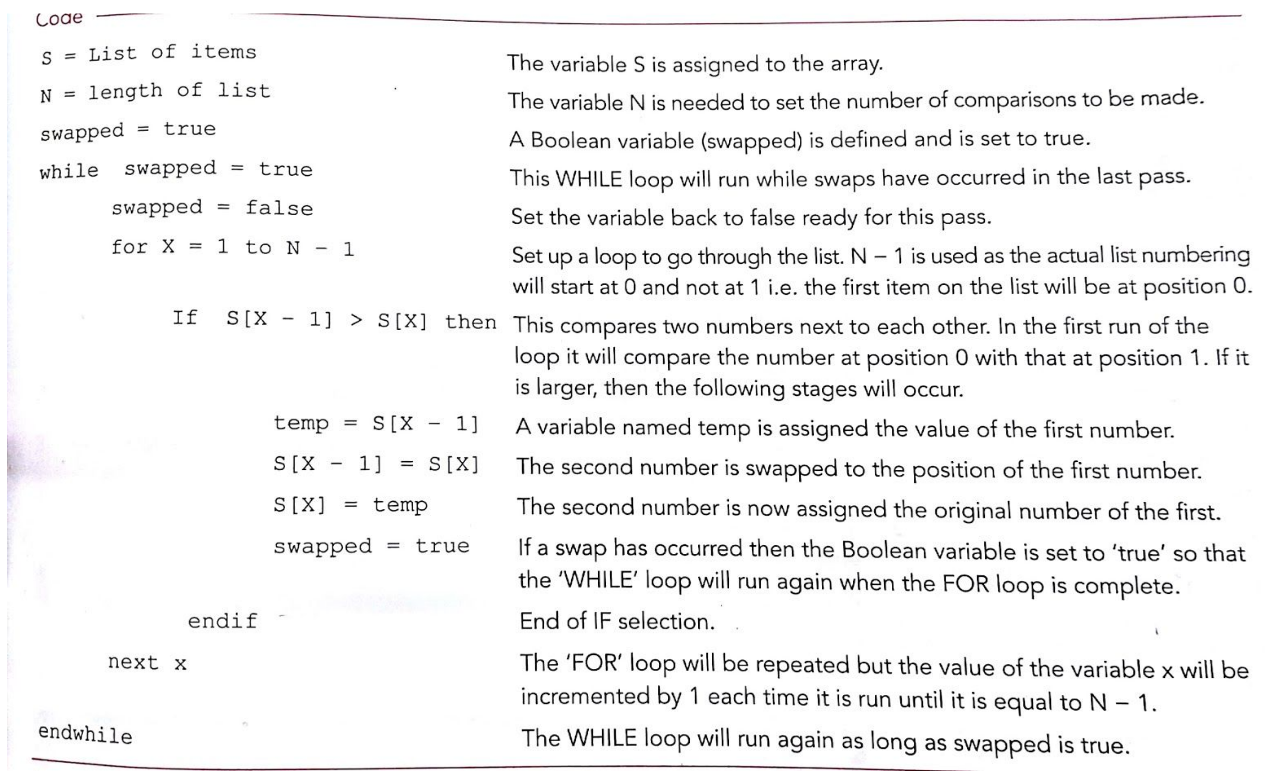
These compare the data with each other so that the correct order can be found. In a business there could be millions of data to compare, so sorting algorithms need to be efficient as possible because they could cause a bottleneck which means that the rest of the program may not run until the sorting task is complete.

Bubble Sort

Create a program that performs a Bubble Sort on a random list of numbers, letters of words. Create or upload your program to Repl.it. Paste your link below. Underneath the link in the space provided write a brief explanation of the Bubble Sort.

The algorithm is below.

1. START at the beginning of the list of values.
2. Compare the first and second values. IF they are in order do nothing, IF NOT swap them.
3. Compare the second and third values. IF they are in order do nothing, IF NOT swap them.
4. Compare the third and fourth values. IF they are in order do nothing, IF NOT swap them.
5. Continue UNTIL the end of the list.
6. This is the first pass.
7. Continue steps 1 to 5 UNTIL no more swaps are made. (The list must now be in order)



Code

`S = List of items`	The variable S is assigned to the array.
`N = length of list`	The variable N is needed to set the number of comparisons to be made.
`swapped = true`	A Boolean variable (swapped) is defined and is set to true.
`while swapped = true`	This WHILE loop will run while swaps have occurred in the last pass.
`    swapped = false`	Set the variable back to false ready for this pass.
`    for X = 1 to N - 1`	Set up a loop to go through the list. N – 1 is used as the actual list numbering will start at 0 and not at 1 i.e. the first item on the list will be at position 0.
`        If S[X - 1] > S[X] then`	This compares two numbers next to each other. In the first run of the loop it will compare the number at position 0 with that at position 1. If it is larger, then the following stages will occur.
`            temp = S[X - 1]`	A variable named temp is assigned the value of the first number.
`            S[X - 1] = S[X]`	The second number is swapped to the position of the first number.
`            S[X] = temp`	The second number is now assigned the original number of the first.
`            swapped = true`	If a swap has occurred then the Boolean variable is set to 'true' so that the 'WHILE' loop will run again when the FOR loop is complete.
`        endif`	End of IF selection.
`    next x`	The 'FOR' loop will be repeated but the value of the variable x will be incremented by 1 each time it is run until it is equal to N – 1.
`endwhile`	The WHILE loop will run again as long as swapped is true.

TASK 1

Produce a table like the example above showing how the list **20, 15, 3, 13, 9, 2, 6** would be sorted in order.

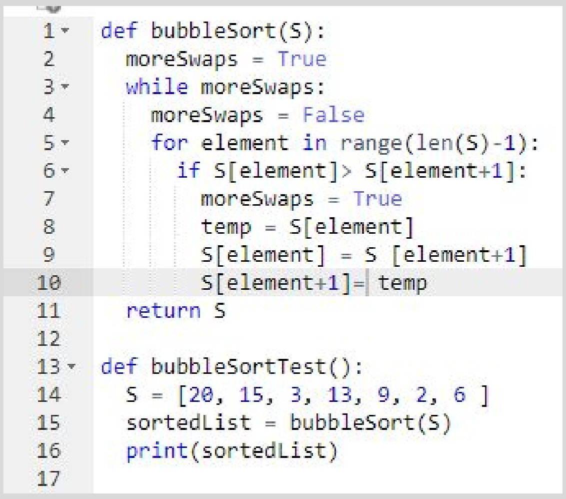
Passes through list	Position 1	Position 2	Position 3	Position 4	Position 5	Position 6	Position 7
0	20	15	3	13	9	2	6
1	15	3	13	9	2	6	20
2	3	13	9	2	6	15	20
3	3	9	2	6	13	15	20
4	3	2	6	7	13	15	20
5	2	3	6	7	13	15	20

TASK 2

Code the example above

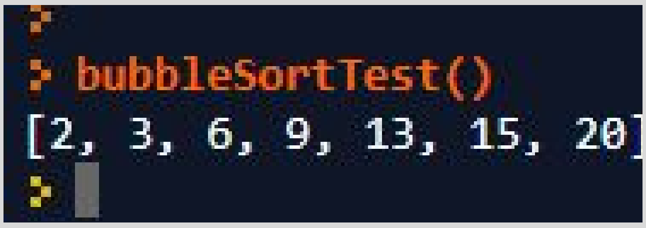
CODE to sort the list 20, 15, 3, 13, 9, 2, 6

Chapter 5.1: Searching and Sorting Algorithms



```python
def bubbleSort(S):
  moreSwaps = True
  while moreSwaps:
    moreSwaps = False
    for element in range(len(S)-1):
      if S[element]> S[element+1]:
        moreSwaps = True
        temp = S[element]
        S[element] = S [element+1]
        S[element+1]= temp
  return S

def bubbleSortTest():
  S = [20, 15, 3, 13, 9, 2, 6 ]
  sortedList = bubbleSort(S)
  print(sortedList)
```



```
>
> bubbleSortTest()
[2, 3, 6, 9, 13, 15, 20]
>
```

Description of the lines of code used in your Bubble Sort
Well it basically does what Bubble Sort does, it swaps the numbers depending if one's higher or lower. At the start you can see I'm making a loop that will repeat the Bubble Sort for all the elements in 'S'. I had to use element - 1 because the first element in an array is referred to as 0.

Extra Help
https://vimeo.com/104864168 (python School)
https://pythonschool.net/data-structures-algorithms/bubble-sort/