Alexander Moore
Methods for Teaching Computer Science
Portfolio Lesson #1 of 2
Professor Dyrland-Weaver
Summer 2021

## Lesson #1 of 2:  Value Swapping Lesson in the Context of Bubble Sort

**Bonus:**  Check out the original Bubble Sort program I created in Snap! for this lesson! https://snap.berkeley.edu/snap/snap.html#present:Username=fssahs&ProjectName=AlexMooreBubbleSortInSnap

**Intended Audience:**  12th Grade AP Computer Science Principles students in a selective arts school.  Students in the class have passed Algebra 2.

**Pedagogical Technique #1:**  Providing code skeleton
**Pedagogical Technique #2:**  Subgoal labeling with explanatory comments in code

**New York State K-12 Computer Science and Digital Fluency Learning Standards - 9-12:**

- 9-12.CT.7 Design or remix a program that utilizes a data structure to maintain changes to related pieces of data.
- Clarifying Statement The focus is on updating the elements or components within a named instance of a data structure, without changing the value associated with the name itself.

**Aim:** How can I use a temp variable to swap two values in a list in the context of a Bubble Sort?

**Warm up** Look at the code segment below in AP-style pseudocode. The programmer had intended to swap the values of variables a and b. Respond to the prompts below.

```
int a=5
int b=100
print (“a:”+ a)
print(“b:”+b)
a=b
b=a
print(“a:”+a)
print(“b:”+b)
```

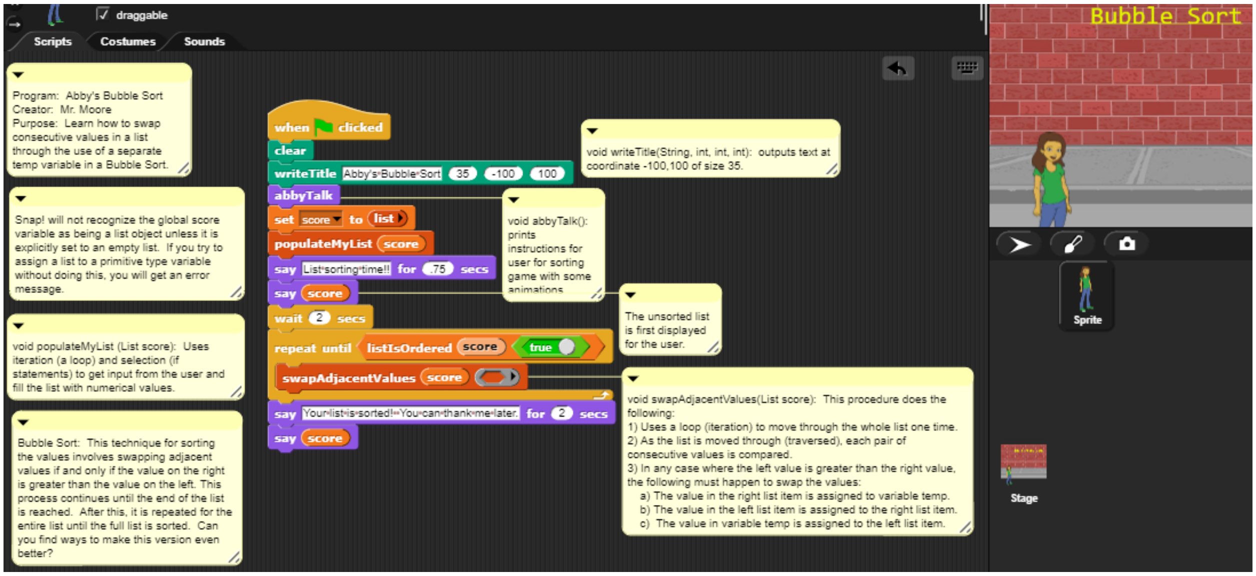
1) What is the difference between the l-value and the r-value in an assignment statement?
   *Cold Calling - Student Answer: The right side of an assignment is evaluated for value. This value must match with the data type (int, float, String, etc.) of the variable on the right side of an assignment or else there could be an error or unexpected results. The r-value is then stored in the memory address of the variable in the l-value.*
2) What is printed by the code segment above?
   *Cold Calling - Student Answer: a: 5 b: 100 a: 100 b: 100*
3) What is needed to switch the variables?
   *Cold Calling - Student Answer: We need a third variable - temp - to store the value of a variable so that it will not be overwritten during the swap.*

**Time: 8 minutes -** 5 minutes for students to work on the task above independently in Google Classroom, and 3 minutes for students to share answers through cold calling.

---

**Lesson Content**

The code for my original Snap! Bubble Sort program is shared with students with comments. This provides context for the swapping algorithm students will complete. The code skeleton is provided here, and subgoals are labeled. In this instance, students are only completing the variable swapping portion of the Bubble Sort program. They are given a skeleton with supporting comments, and they can even make use of an additional support list if desired. The goal of the initial teacher-led component is to provide context and explanation to get students started, but this is largely a student-driven lesson. See materials and further explanation below:



Program: Abby's Bubble Sort
Creator: Mr. Moore
Purpose: Learn how to swap consecutive values in a list through the use of a separate temp variable in a Bubble Sort.

Snap! will not recognize the global score variable as being a list object unless it is explicitly set to an empty list. If you try to assign a list to a primitive type variable without doing this, you will get an error message.

void populateMyList (List score): Uses iteration (a loop) and selection (if statements) to get input from the user and fill the list with numerical values.

Bubble Sort: This technique for sorting the values involves swapping adjacent values if and only if the value on the right is greater than the value on the left. This process continues until the end of the list is reached. After this, it is repeated for the entire list until the full list is sorted. Can you find ways to make this version even better?

```
when clicked
clear
writeTitle Abby's Bubble Sort 35 -100 100
abbyTalk
set score to list
populateMyList score
say List sorting time!! for .75 secs
say score
wait 2 secs
repeat until listIsOrdered score true
    swapAdjacentValues score
say Your list is sorted!  You can thank me later. for 2 secs
say score
```

void writeTitle(String, int, int, int): outputs text at coordinate -100,100 of size 35.

void abbyTalk(): prints instructions for user for sorting game with some animations

The unsorted list is first displayed for the user.

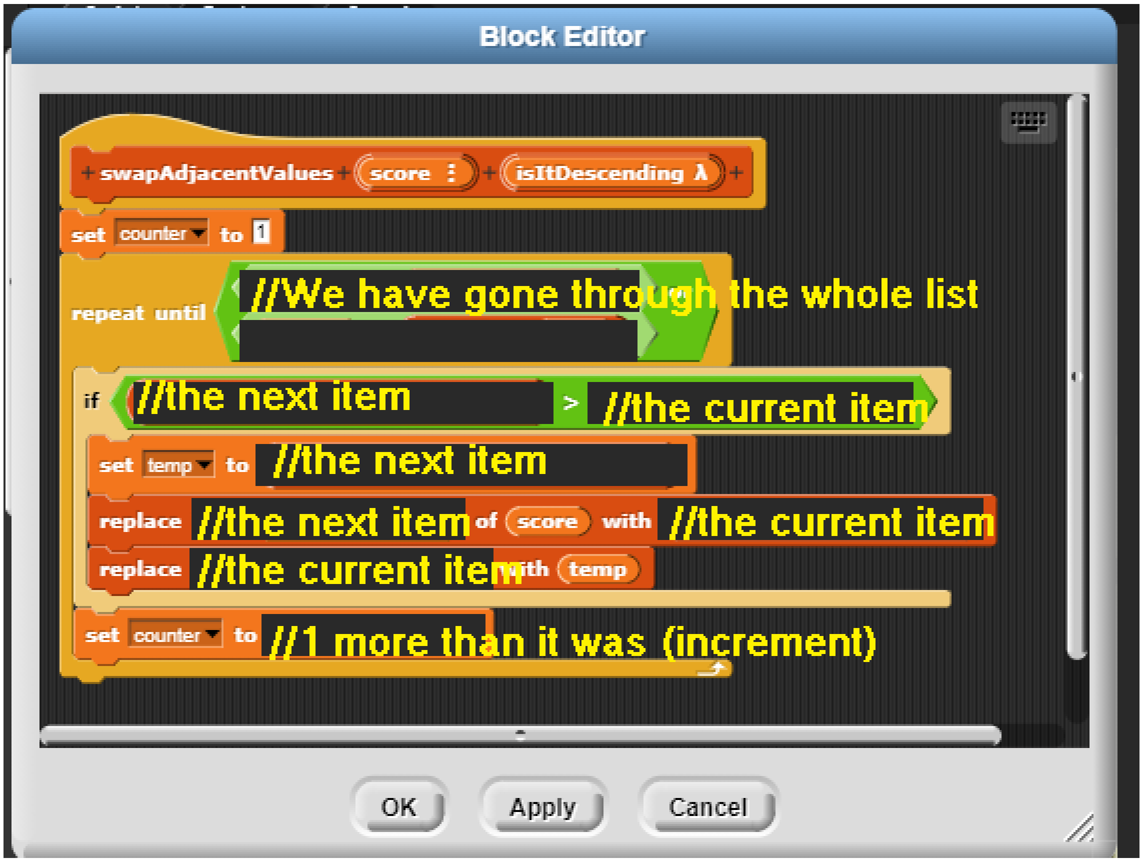
void swapAdjacentValues(List score): This procedure does the following:
1) Uses a loop (iteration) to move through the whole list one time.
2) As the list is moved through (traversed), each pair of consecutive values is compared.
3) In any case where the left value is greater than the right value, the following must happen to swap the values:
a) The value in the right list item is assigned to variable temp.
b) The value in the left list item is assigned to the right list item.
c) The value in variable temp is assigned to the left list item.

/*
Comment 1: Program: Abby's Bubble Sort
Creator: Mr. Moore
Purpose: Learn how to swap consecutive values in a list through the use of a separate temp variable in a Bubble Sort.
*/
/*
Comment 2: Bubble Sort: This technique for sorting the values involves swapping adjacent values if and only if the value on the right is greater than the value on the left. This process continues until the end of the list is reached. After this, it is repeated for the entire list until the full list is sorted. Can you find ways to make this version even better?
*/
/*
Comment 3: Snap! will not recognize the global score variable as being a list object unless it is explicitly set to an empty list. If you try to assign a list to a primitive type variable without doing this, you will get an error message.
*/
/*
Comment 4: void populateMyList (List score): Uses iteration (a loop) and selection (if statements) to get input from the user and fill the list with numerical values.
*/
/*
Comment 5: void writeTitle(String, int, int, int): outputs text at coordinate -100,100 of size 35.
*/
/*
Comment 6: void abbyTalk(): prints instructions for user for sorting game with some animations
*/
/*
Comment 7: The unsorted list is first displayed for the user.

*/
/*
Comment 8:  void swapAdjacentValues(List score):  This procedure does the following:
1) Uses a loop (iteration) to move through the whole list one time.
2) As the list is moved through (traversed), each pair of consecutive values is compared.
3) In any case where the left value is greater than the right value, the following must happen to swap the values:
    a) The value in the right list item is assigned to variable temp.
    b) The value in the left list item is assigned to the right list item.
    c)  The value in variable temp is assigned to the left list item.
*/
**Time:  5 minutes**

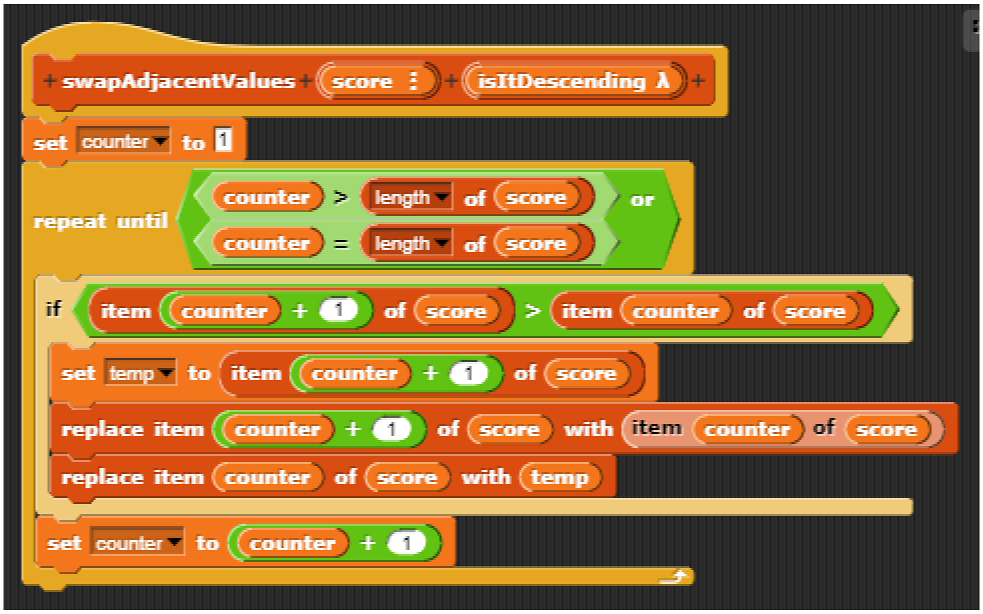
---

### Lesson Activity



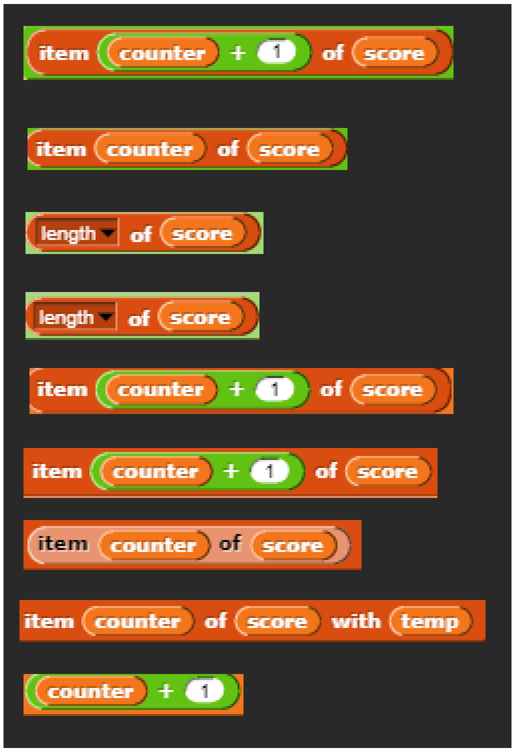
```
+ swapAdjacentValues + (score ⋮) + (isItDescending λ) +
set counter to 1
repeat until //We have gone through the whole list
    if //the next item > //the current item
        set temp to //the next item
        replace //the next item of score with //the current item
        replace //the current item with temp
    set counter to //1 more than it was (increment)
```

Solution:  This swapAdjacentvalues procedure will exchange consecutive values of a list when the first is greater than the second.  Note that this will continue to be called until the list is fully sorted, which is ascertained by the procedure Boolean listIsSorted (List score).



```
+ swapAdjacentValues + score ⋮ + isItDescending λ +
set counter to 1
repeat until < counter > length of score  or  counter = length of score >
    if < item ( counter + 1 ) of score > item counter of score >
        set temp to item ( counter + 1 ) of score
        replace item ( counter + 1 ) of score with item counter of score
        replace item counter of score with temp
    set counter to ( counter + 1 )
```

Students will be given access to this hint sheet, but they will be encouraged not to consult it unless they get stuck.  It contains the answers that can be plugged into the blanks to make a working procedure, but the order of the answers is not communicated here.



```
item (counter + 1) of score
item counter of score
length of score
length of score
item (counter + 1) of score
item (counter + 1) of score
item counter of score
item counter of score with temp
counter + 1
```

**Time: 25 minutes**

**Closing -** *Teacher comments: Today we reviewed the basic idea of assignment. You are putting the r-value into the variable on the left. This becomes more important when you get to work with more advanced assignments in languages like C/C++ and try to make sense of pointers to memory addresses. Variable assignment relates to our Aim for the day because we were trying to swap values of variables, but it was much more annoying than we had expected: We actually had to create a third, temp variable to exchange variables. One algorithm that uses variable swapping is called Bubble Sort. We worked with our teacher's implementation of Bubble Sort in Snap!, and you worked with a partner to code the variable swapping component. Please take a moment to talk with a partner about any questions you may still have - [pause for 30 seconds] - O.K., what does not make sense about this?*

**Time: 5 minutes**