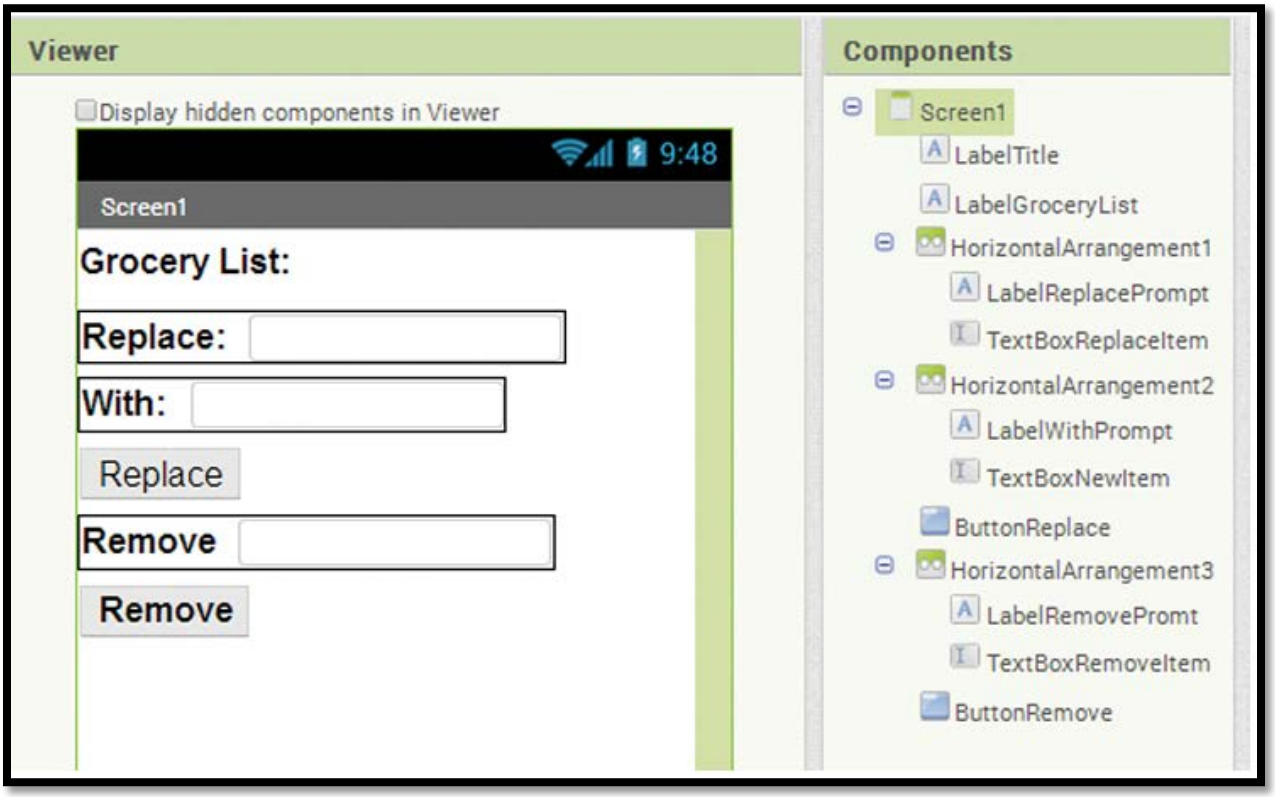
Replace and Remove Items App

In this tutorial we are going to write an app that will allow a user to remove or replace items in a grocery list. We will set up the list with several items and then give the user the ability to select an item to remove or replace. If the user chooses to replace an item, a TextBox will be available for the new value to be entered.

Step 1: Create the design with the components shown below:



1. Place a Label for the title Grocery List:. Change the Label's name to LabelTitle, its font size to 20, and its Text property to Grocery List:.
2. Place a Label beneath the title to display the grocery list, name it LabelGroceryList, change its font size to 20, and clear the Text property contents.
3. Add a HorizontalArrangement component and place a Label and TextBox inside it. Name the Label LabelReplacePrompt, change its font size to 20, and its Text property to Replace. Name the TextBox TextBoxReplaceItem, change its font size to 20, clear its Text property, and change the Hint property to Item number to replace.
4. Add another HorizontalArrangement component and place a Label and TextBox inside it. Name the Label LabelWithPrompt. Change its font size to 20 and its Text property to With:. Name the TextBox TextBoxNewItem, change its font size to 20, clear its Text property, and change the Hint property to Name of new item.
5. Add a Button, name it ButtonReplace, and change its Text property to Replace.
6. Add another HorizontalArrangement component and place a Label and TextBox inside it. Name the Label LabelRemovePrompt, change its font size to 20, and change

its Text property to Remove. Name the TextBox TextBoxRemoveItem, change its font size to 20, clear its Text property, and change the Hint property to Item number to remove.

7. Add a Button, name it ButtonRemove, and change its Text property to Remove.

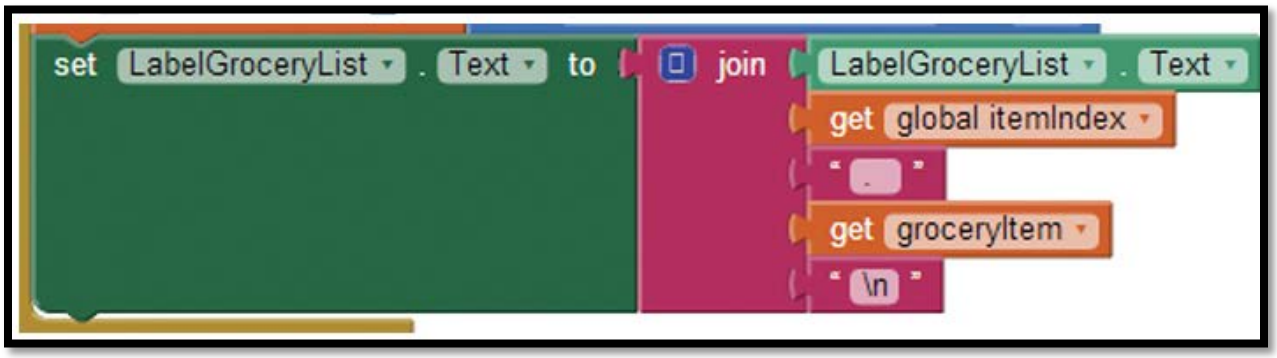
Step 2: Recall that we need a variable to hold our list of grocery items. Once our variable is created, we should use the make a list function block and fill it in with text items that represent grocery items. Because we are displaying the list numbered by the index of each item (1., 2., etc.) we also need a variable to hold the number, or index.

1. Use the following steps to create the blocks shown above:
    1. Go to the *Variables* drawer and select an initialize global name to block. Do this twice so that you have two in your workspace.
    2. Change the name of the first variable to groceryList (click on the word *name* and type groceryList on top of it).
    3. Change the name of the second variable to itemIndex.
    4. Go to the *Lists* drawer and select the make a list block. Plug that into the initialize global groceryList to block.
    5. Go to the *Text* drawer and select a text block. Repeat or copy and paste the block in the Blocks Editor until you have five text blocks. Change their names to your favorite grocery items. Plug the items into the make a list function block.
    6. Go to the *Math* drawer, click on the number block, change the value to zero, and plug it into the itemIndex variable block to initialize this variable to zero.

Step 3: In this step you will program the for each loop shown below, to iterate and display the list. When you complete this loop, you will plug it into a procedure block so it is called each time the user updates the grocery list by replacing or removing items.

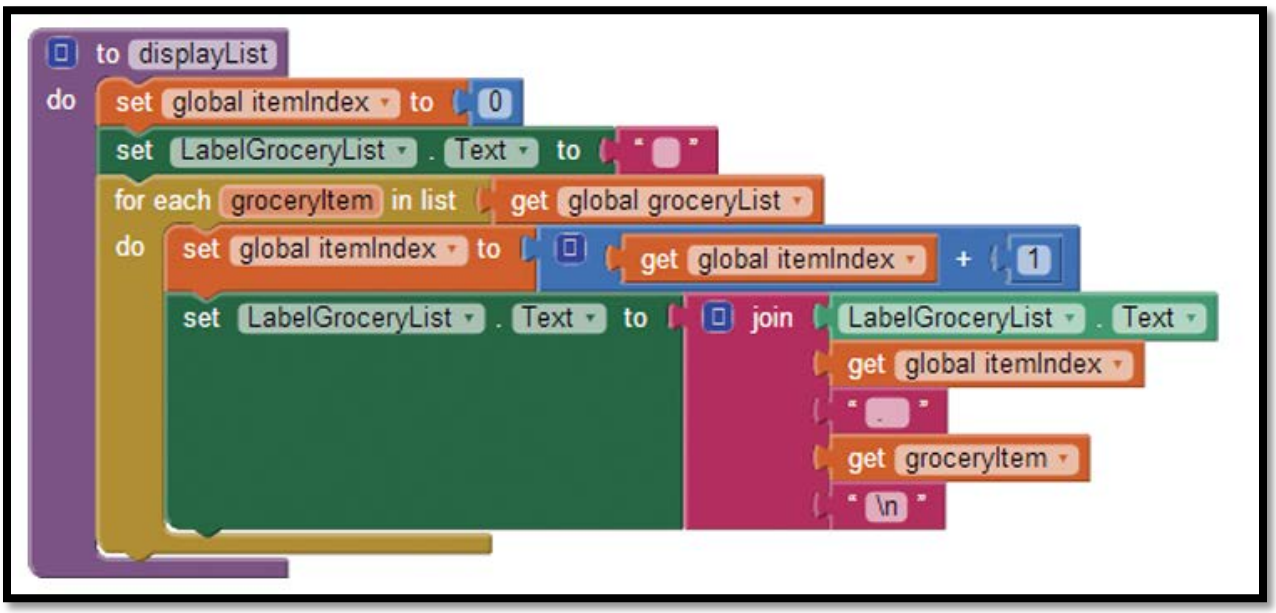
1. Go to the Control drawer and select a for each item in list block. Rename the item element that is named at the top of the loop to groceryItem (double click and type over the word item).
2. Hover over the name groceryList in the initialize global groceryList to block. Once you hover over the name, choose the get global groceryList block. That block holds the value, or contents, of the list. Plug it into the slot at the top of the loop.
3. Hover over the name itemIndex in the initialize global item Index to block. Once you hover over the name, choose the get global itemIndex block. Increment it by one by using a Math plus block (+), a Math number block, and the get global itemIndex block. Plug this block structure into the for each loop block.

4. Now, in the LabelGroceryList drawer, find the set Label GroceryList.Text to block and plug that in the for each loop under the itemIndex increment block. In the Text drawer, select the join block and plug in the following: LabelGroceryList.Text, get global itemIndex, a text block containing a period with a space, a get groceryItem block, and a text block containing the carriage return \n.



Step 4: Create a procedure called displayList to display the items in the list. By creating a procedure, we can call it multiple times after each update. The first steps in this procedure are to reset the itemIndex to 0 and the LabelGroceryList.Text to blank so that our list starts from scratch. Then we will plug in the for each loop from Step 3.

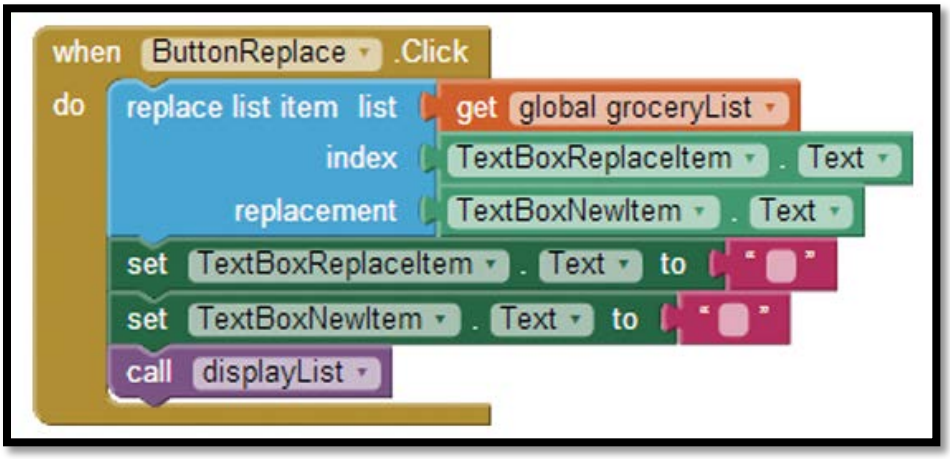
1. Go to the Procedures drawer and select a to procedure do block (you do not need a procedure with result). Double click on the name procedure and rename it to displayList.
2. Hover over the name itemIndex in the initialize global itemIndex to block to get a set global itemIndex block.
3. Plug a number block (from the Math drawer) set to 0 into the set global itemIndex block. Plug that into the procedure.
4. Next, pull out a set LabelGroceryList.Text to block from the LabelGroceryList drawer and an empty text block from the Text drawer. Plug those blocks together, and then plug the resulting block into the procedure.
5. Now, take the for each block from Step 4 and plug it in as shown in below.



Step 5: Now you will create the Screen1.Initialize event handler. Open the drawer Screen1 and select the Screen1.Initialize block. Next, open the Procedures drawer and select the call displayList block. Plug the procedure call block into the Screen1.Initialize event handler. The Blocks Editor workspace is now complete.

Run and test your app. As shown below, it should display the list and show the TextBoxes and Buttons. The Buttons will not work yet; we just want to make sure that the list is showing.

Step 6: Now let's make the Replace Button work by creating the blocks shown below. You will use the ButtonReplace.Click event handler and the replace list item block. You will also clear the two TextBoxes (to clear any existing user input) when the Replace Button is pressed. The last block in the ButtonReplace.Click event handler will be to call the displayList procedure so that the new list with the item removed is displayed.



Find the when ButtonReplace.Click do event handler in the ButtonReplace drawer and place it in the workspace.

1. Find the replace list item block in the *List* drawer; place it inside the ButtonReplace.Click event handler.
2. Plug in the get global groceryList block found by hovering over groceryList in its variable initialization block.
3. Plug the TextReplaceItem.Text block into the index slot; this is the index that the user typed into the TextBox. It can be found in the *TextBox ReplaceItem* drawer.
4. Plug the TextBoxNewItem.Text block into the replacement slot. This block is found in the *TextBoxNewItem* drawer.
5. Now find the set TextReplacementItem.Text to and the set Text BoxNewItem.Text to blocks and place them in the editor. From the *Text* drawer, pull out two empty text blocks. Plug the text blocks into the set blocks, and plug those in under the replace list item block.
6. Find the call displayList block in the *Procedures* drawer and place it as the last item in the Click event handler.

Step 7: Now you will program the Remove Button to remove an item from a list. As with the Replace button, you will need to reset the screen elements and call the displayList procedure to redisplay the list with the item removed.

1. Find the when ButtonRemove.Click do event handler in the Button Remove drawer and place it in the workspace.
2. Find the remove list item block in the List drawer and place it inside the ButtonRemove.Click event handler as shown below.

3. Now find the set TextRevmoveItem.Text to and place it in the workspace. From the Text drawer, pull out an empty text block. Plug the text block into the set TextRevmoveItem.Text to block and plug those in, under the remove list item block.
4. Find the call displayList block in the Procedures drawer and place it as the last item in the ButtonRemove.Click event handler.