Code 2 Challenge

Make a To-Do List

Description

Make a to-do list that allows users to enter in and remove items. Your app should remember the items each time your user opens the app. You may want to learn how to use a ListView before trying the challenge!

See our example app here: Technovation Challenge #2

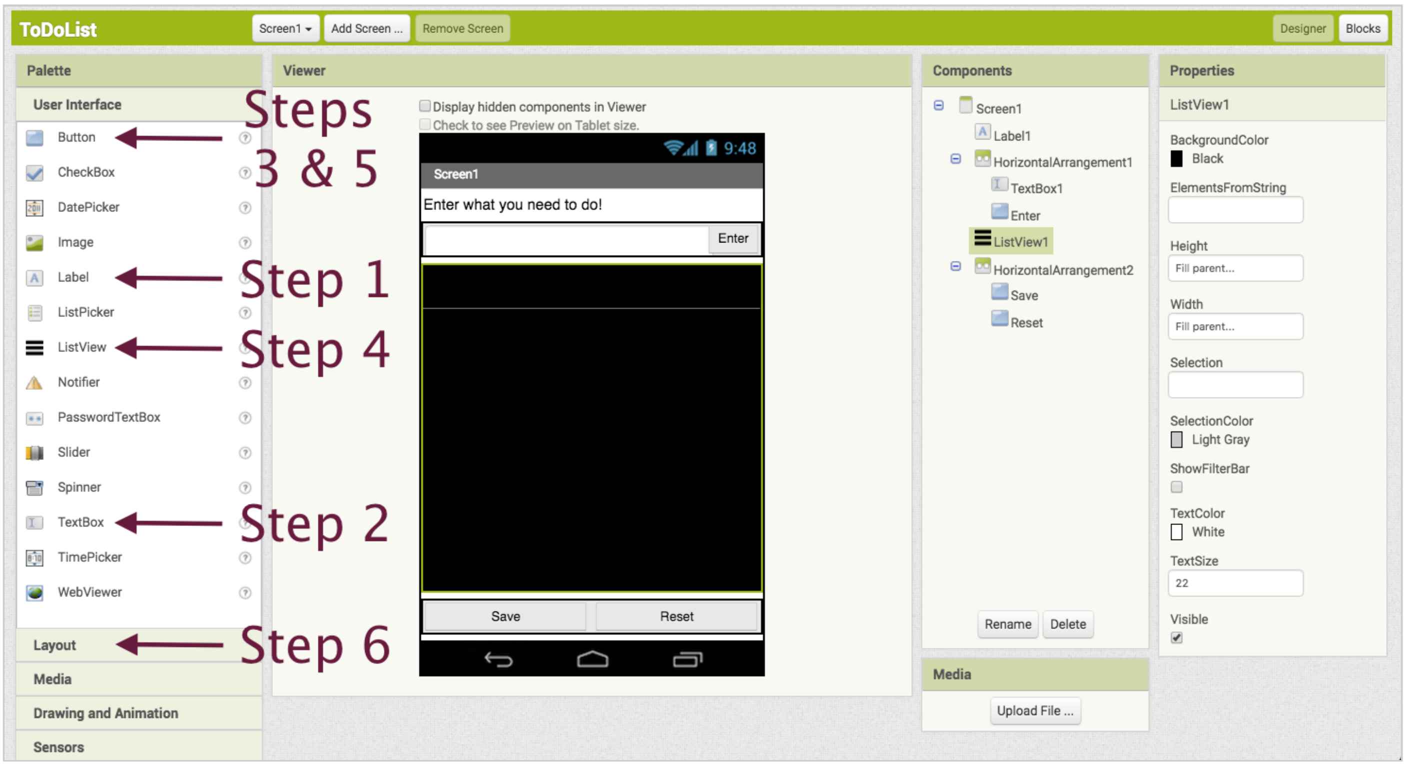
Try your hardest to do this challenge on your own before you read the instructions. Remember, these instructions are just one way the challenge can be solved!. Happy coding!

Designing your screen

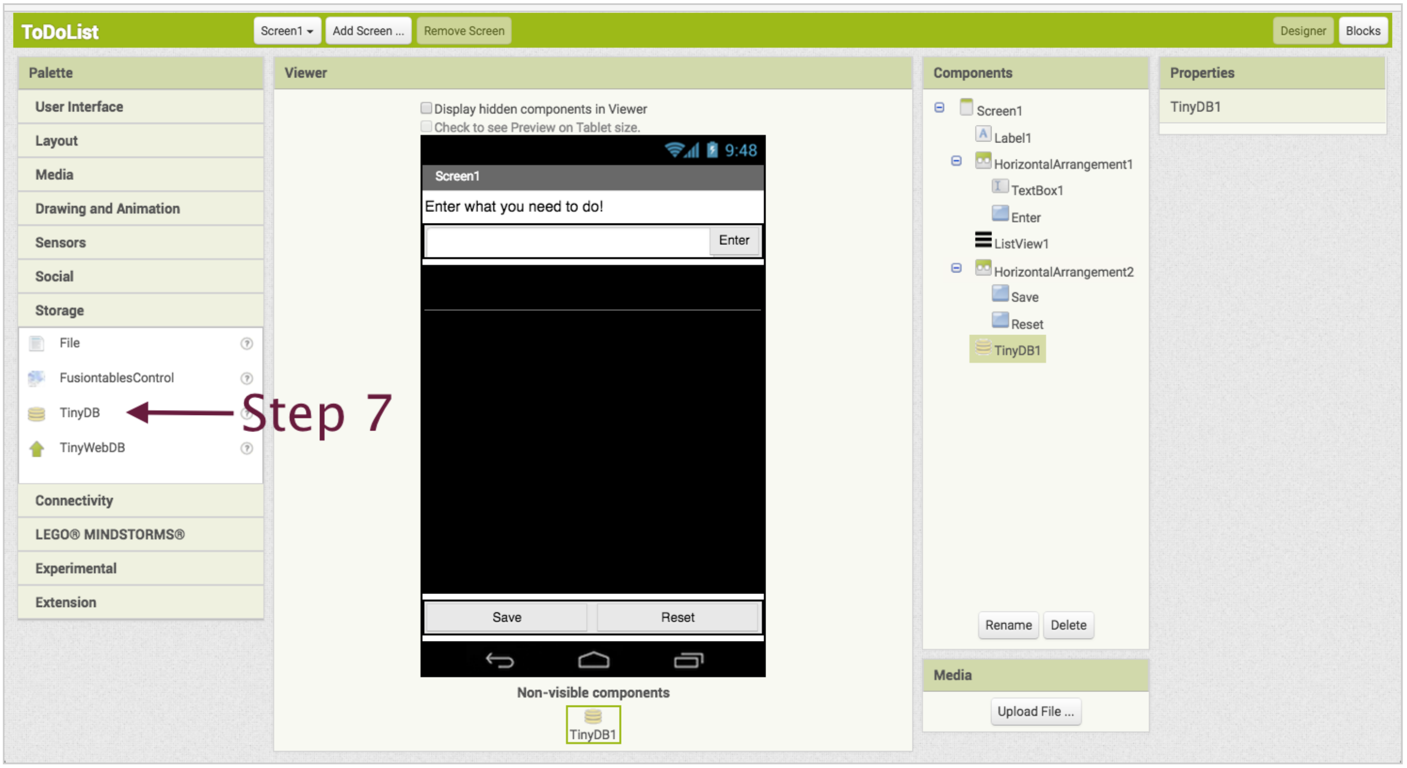
We created an app that allows the user to add things to the list, take things off the list, save the list, and reset the list. This required our screen to have three buttons: one to enter items into the list, one to save the user's items for the next time, and a reset button that will erase the list.

1. Add a label to your screen to tell your user what to do.
2. Add a textbox for your user to enter their to-do item into.
3. Add a button for the user to click once they entered in their to-do item.
    a. We renamed this button 'Enter' so we can remember what it does when we program our blocks!
4. Add ListView to your screen. This is how the user will see their to-dolist.
    a. We set our height and width to 'Fill Parent' but you should design your screen however you want!
    b. Click here to learn more about how to use a ListView.
5. Add a save button to your screen for saving the to-do list to a database and a reset button to erase the list.
    a. We renamed these buttons 'Save' and 'Reset'.

6. *Optional:* We added two horizontal arrangements to organize our layout, but you can organize your screen however you want!



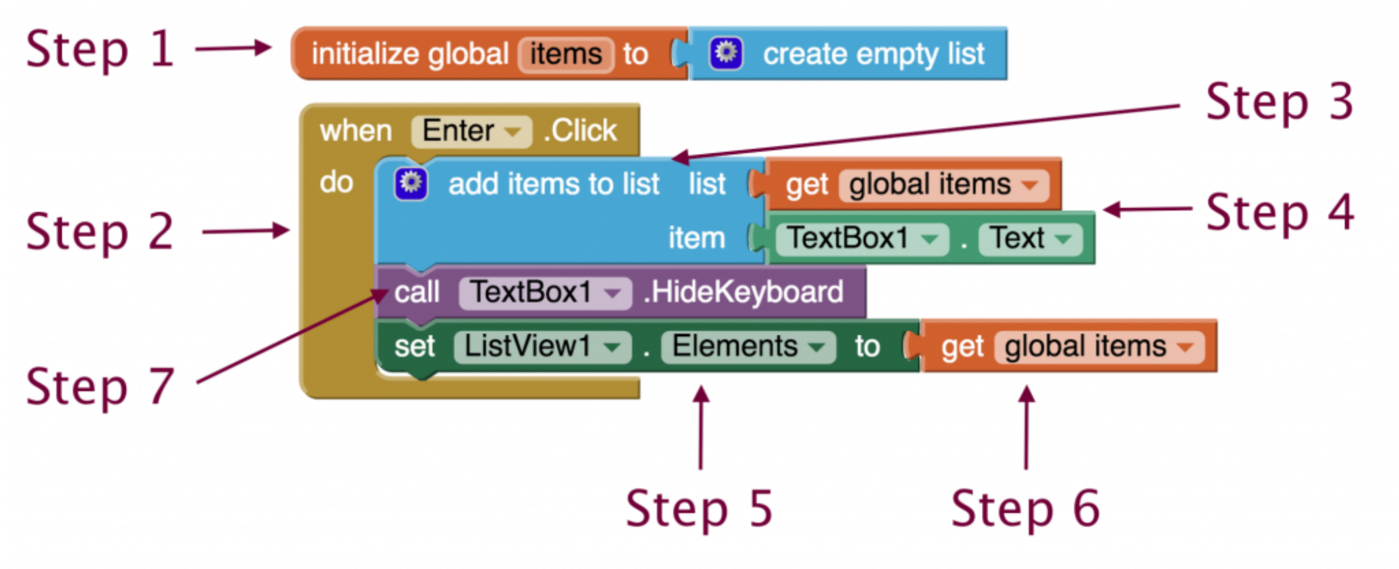
7. Add a TinyDB from the storage menu. When you drag it onto your screen, it will appear as a 'Non-visible component'.



Adding things to the to-do list

Our app to displays inputs that the user enters by storing the data in a list variable and then displaying it using ListView. We did this by creating a list variable called *items*. When our user hits the enter button, we add whatever is in textbox to the list *items* and display it in ListView.

1. Make a variable called *items* that will hold your to-do list data. Start the variable out as an empty list since the user hasn't entered any data yet.
2. Grab a **button.click** block for your enter button.
3. Get an **add items to list** block from **Lists**. Put it inside your **enter.click** event handler.
4. Put *items* for the list name and **TextBox1.Text** as the item to be added to the list.
5. Grab the **set ListView1.Elements to** block and put it underneath the **add items to list** block.
6. Add your *items* variable to the **set ListView1.Elements to** block.
7. *Optional:* To make our app easier to use we hid our keyboard when the user pressed enter. To do this, click on TextBox1 and grab the **TextBox1.HideKeyboard** block.



Check Point #1

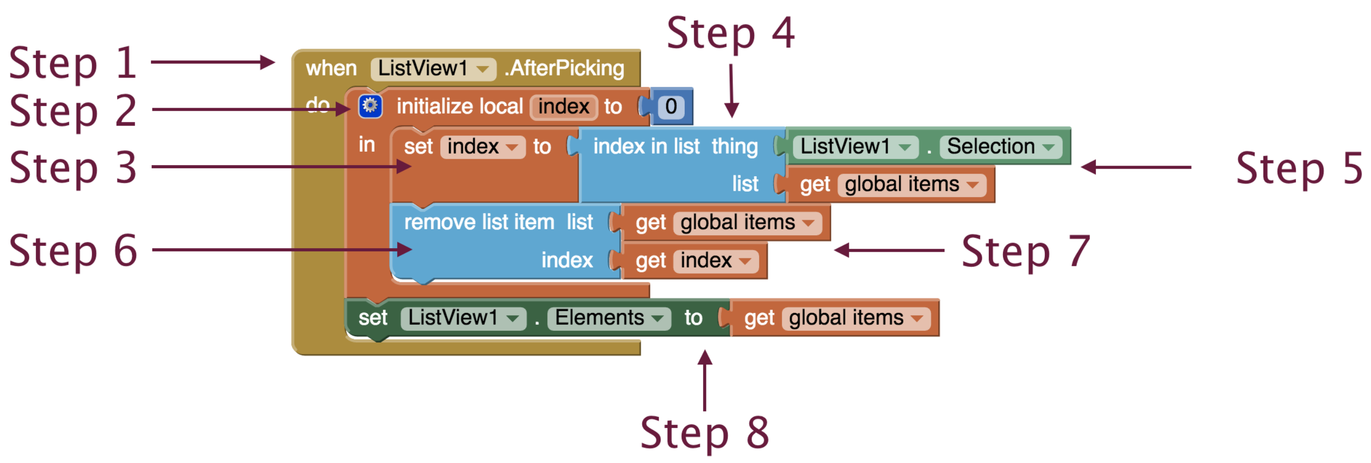
See if what you built so far is working! Hook up your phone or emulator and test your app. When you type things into the textbox and press enter they should appear below in the ListView. If your app is not working, go back and and look for errors to fix before moving on. Reach out to your mentor or teammates if you are stuck.

Deleting entries from the to-do list

Our users can delete something from the to-do list once they've completed the task or no longer want it there by clicking on it in ListView.

We did this by creating a variable called *index.* This is local variable since we only needed to use it in one place in the code. We set the variable *index* to be the index of what the user selected in the ListView. Then we used *index* to tell the app what to remove from the list *items*. After ListView is updated since the list *items* has changed.

1. Grab the **ListView1.AfterPicking** block.
2. Create a local variable called *index* and set it whatever you like to start out. We set our to zero.
3. Get the **set 'name' to** variable block and set it to the variable *index.*
4. Attach the block that says **index is list, thing, list**.
5. Assign **thing** to be **ListView1.Selection** and the **list** to be the *items* list.
6. Grab the **remove list item block** and put inside your local variable block.
7. Set your *items* variable to be the **list** and your *index* variable to be the **index**.
8. Place the **set ListView1.Elements to** block below your orange variable block and connect your *items* variable to it.



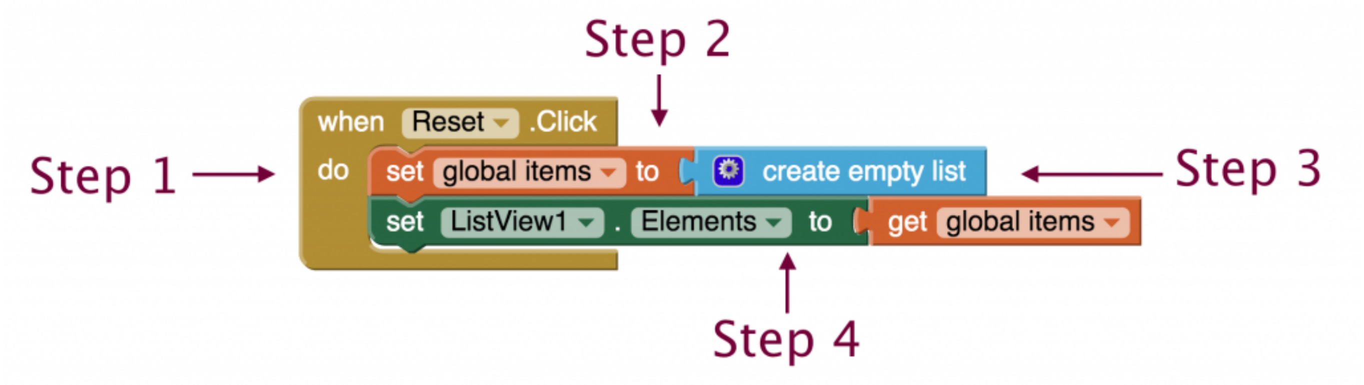
Check Point #2

Time to see if your app is working! Hook up your phone or emulator and test it out. When you use your app you should now be able to touch on the items you have entered into your to-do list to delete them. If your app is not working go back and check your code. Try looking at some ListView tutorials or reaching out to your mentor or teammates if you are stuck!

Programming the reset button

To make our app easier to use, we added a reset button that clears all entries out of the to-do list. We programmed our reset button to change the list *items* back to an empty list when the user hits it. Then, we updated ListView to display the new empty contents of the list *items*.

1. Grab **when button.click** block for your reset button.
2. Get the **set 'name' to** variable block and select your *items* list
3. Attach a **create empty list** block.
4. Grab the **set ListView1.Elements to** block and attach the get *items* variable block.


Step 2
Step 1
Step 3
Step 4
when Reset .Click
do set global items to create empty list
set ListView1 . Elements to get global items

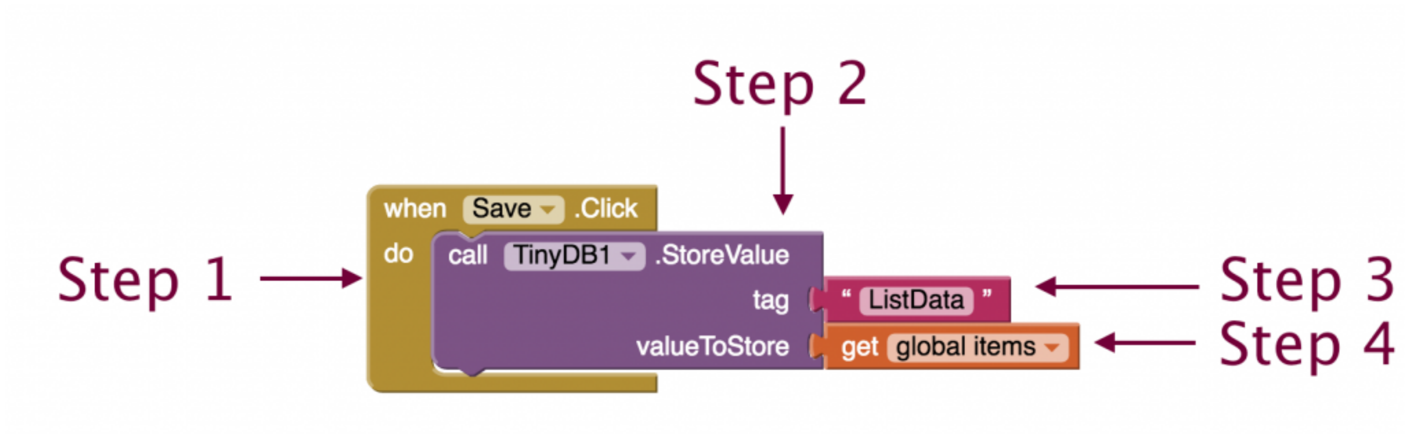
Check Point #3

Make sure your reset button works! Hook up your phone or emulator and test it out. After you put entires into your to-do list you should be able to hit reset and to clear the list. Reach out to your mentor or teammates if you are stuck.

Programming the TinyDB

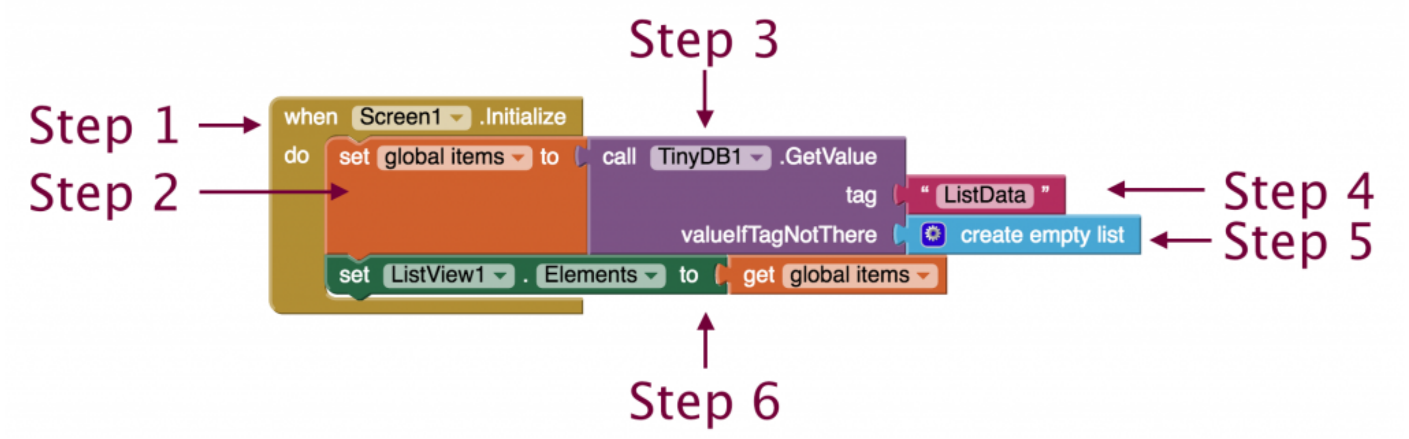
When our user closes the app, the to-do list is saved for the next time. We programmed our “save” button to call our database (TinyDB) and save the *items* list. The database remembers the user’s to-do list exactly how it is at that moment and keeps it there for the next time.

1. Grab the when **when button.Click** block for your save button.
2. Add the **call TinyDB1.StoreValue** block.
3. Add a **tag** name.
4. Add your *items* variable to **valueToStore**.



Our app checks the database when it starts up to see if there is any previously saved to-do lists. To do this, we used the **when Screen1.Initialize** block, which will run when Screen1 starts up. We set our list *items* to whatever is in the database under the tag name "ListData". If there is no stored data, we told our app to create an empty list, or a blank to-do list. After that the app display whatever is in the *items* variable which is now either stored data or an empty list.

1. Grab the **when Screen1.Initialize** block.
2. Grab a **set ‘variable name’ to** block and set the variable to be your *items* list.
3. Call the database using the **call TinyDB1.GetValue** block.
4. Enter the tag name you used to save the list for the **tag**.
5. Put the **create empty list** block for **valueIfTagNotThere**.
6. Grab the **set ListView1.Elements to** block and attach it to your *items* variable.



Final Check Point

Time to see if your app is working! Boot up your app and put some entries into your to-do list. Hit your save button and close out of the AI companion app on your phone or emulator. Open the app back up and see if your entries are there. If they aren't, go back and make sure you don't have any errors in your code. Make sure you tag names match exactly including all capital letters. If everything is working you've completed the challenge! Congratulations!

Still stuck? Download our source code. Here are **instructions** on how to download and use source codes.

DOWNLOAD THE SOURCE CODE

English