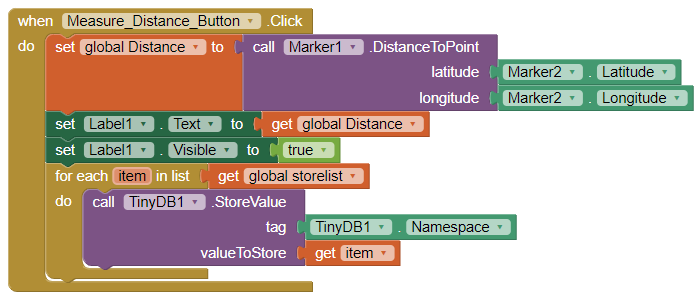
2.a.)

The programming language used is MIT App Inventor. The purpose of the program is to allow users to map out a specific course in their local area to run or jog in, be able to measure that distance, and share it with the recipients they want. The video illustrates the following features in this order: the ability to switch between screens, the ability to open/close the drop-down menu, the ability to create markers, the ability to touch and move markers, the ability to measure the distance between the markers regardless of position, the ability to hide markers, and the scroll ability in the drop-down menu.

2.b.)

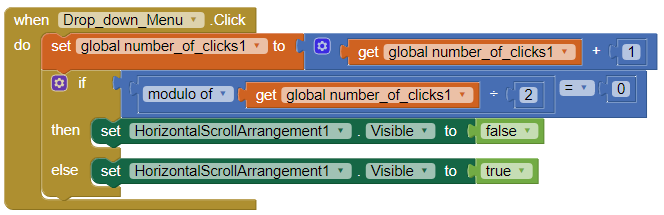
While creating my program independently from start to finish, I would focus on one task or feature and code it; then, I would pair it with a device and see if the new feature in the program works as intended. If it did not, I would go back and figure out how to fix it, if it did, I would move on to the next important task. I implemented buttons to help navigate the map such as one button to create markers. A difficulty at first is that I didn’t know how to make it so that the markers are created into existence by the user. Eventually I made it so that the markers were created beforehand and invisible; when the user “creates” the markers they just become visible, offering the illusion that they were created. By the end of the program I had many functional buttons to help the user navigate the program and its features. With these many buttons, I saw the opportunity to use a sort of drop-down menu to help organize things and minimize space; it gave the program a more clean, sleek, and professional feel.

2.c.)



One algorithm records the most recent distance from Marker1 to Marker2 by getting and comparing their latitude and longitudes; then it changes the text on a label to the new distance variable and makes it visible to inform the user of the distance. The other algorithm goes through a list of values created by measuring the distance (such as the latitude and longitude of Marker1, Marker2, and the distance itself) and stores them in TinyDB1 to be called upon later. When the user tries to share this information in another screen to their recipients, they use these stored values in TinyDB1.

2.d.)



This abstraction basically dictates whether the drop\_down\_menu is opened or closed when the user clicks on it. Initially the number\_of\_clicks1 is set to 0. When the button for the drop\_down\_menu is clicked, it adds 1 to the number\_of\_clicks1. If number\_of\_clicks1 is not divisible by 2, then the drop\_down\_menu becomes visible, it’s opened. If the number\_of\_clicks1 is divisble by 2, then the drop\_down\_menu becomes invisible, it’s closed. The number\_of\_clicks1 is initially set to 0 so that the first time the user clicks on it, it’s an odd number and opens; the next time the user clicks on it, it should become an even number and close. There should be a cycle and every time the user wants to open it, it should be odd, and close it, it should be even. To the user they’re just opening and closing a menu.