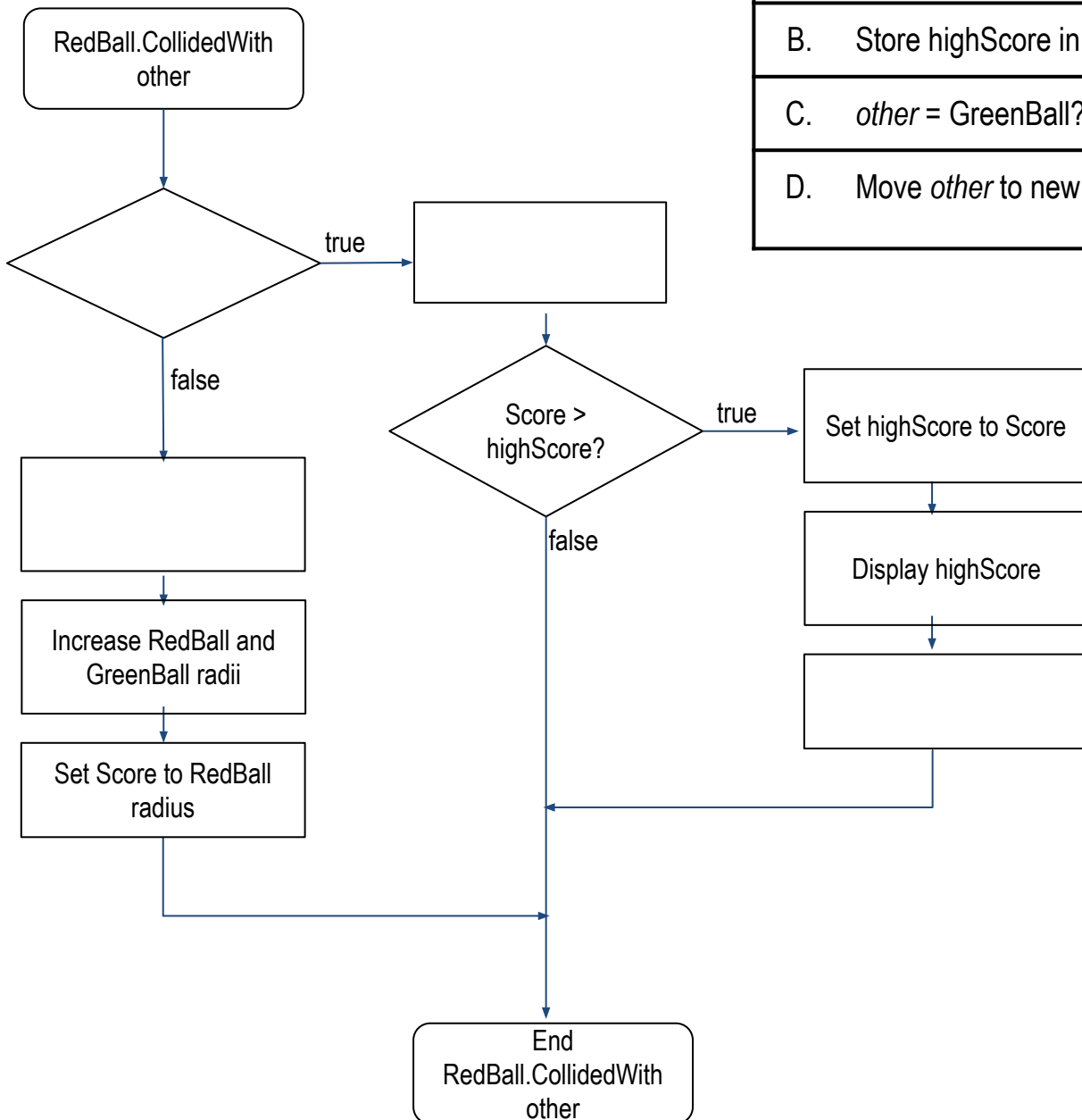# FOOD CHASE GAME: PART 4

START HERE

You will add a High Score for the game, using the final radius of the RedBall as a "score". The bigger the better, right?

1 With your partner, look at the following flowchart and fill in the missing blocks with the correct letter, according to this table.

A.	Notify user game is over
B.	Store highScore in TinyDB
C.	*other* = GreenBall?
D.	Move *other* to new position

Flowchart blocks:
- RedBall.CollidedWith other
- (decision) → true → [ ]
- false → [ ]
- Increase RedBall and GreenBall radii
- Set Score to RedBall radius
- Score > highScore? → true → Set highScore to Score → Display highScore → [ ]
- false
- End RedBall.CollidedWith other

MIT APP INVENTOR

FOOD CHASE GAME: PART 4

ADDING COMPONENTS

You will add a High Score for the game, using the final radius of the RedBall as a "score". The bigger the better, right?

2 Go to the Designer. - - - - ▶

Designer | Blocks

3 At the top of the Viewer, add a **HorizontalArrangement** and add Labels inside it for Score and High Score. The following are the needed components.

Drawer	Component	Name	Property	Setting
Layout	HorizontalArrange-ment	HorizontalArrange-ment1	*Width* *AlignHorizontal*	"**Fill Parent**" "**Center**"
User Interface	Label (4)	ScoreLabel Score HighScoreLabel HighScore	*Text* *Text* *Text* *Text*	"**Score: **" "**0**" "**High Score: **" "**0**"
Storage	TinyDB	TinyDB1		

The screen should look like this: - - - ▶
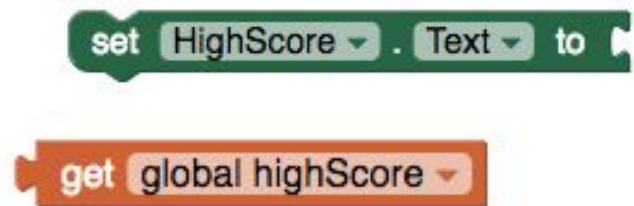
Score: 0   High Score: 0

MIT APP INVENTOR
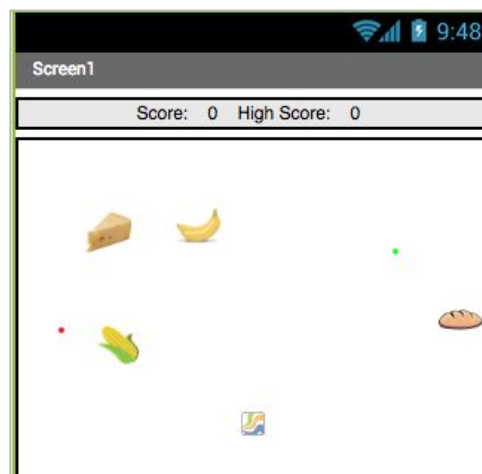
VARIABLES

4 From the Variables drawer, drag out a new block to initialize a global variable, called "highScore". Initialize it to zero.

initialize global highScore to ◄········ 0

5 In the **Screen1.Initialize** event block, add blocks to set the Label **HighScore**.**Text** to the value of **highScore**.

set HighScore . Text to

when Screen1 .Initialize
do call Restart
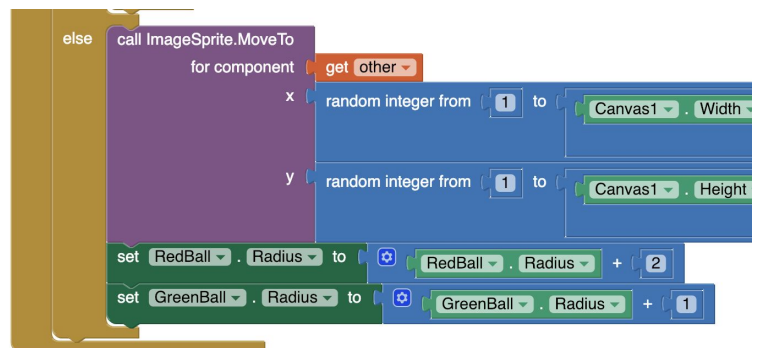
get global highScore

The **get** block *gets* the value of the variable. Use the dropdown to select highScore.

SETTING THE SCORE

6 Update the **Restart** procedure to also reset the **Score**.

to Restart
do call Food1 .MoveTo
x randor
y randor

0

set Score . Text to
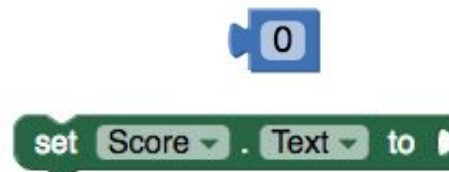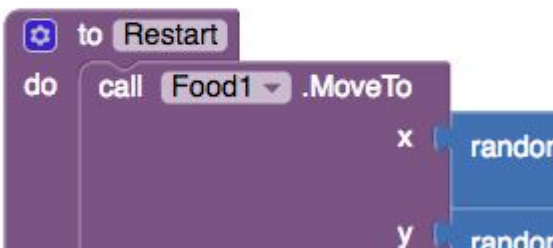
7 In the **RedBall.CollidedWith** block, add blocks to set the **Score** to be the same as the *Radius* of the **RedBall**.

set Score . Text to

RedBall . Radius

else call ImageSprite.MoveTo
for component get other
x random integer from 1 to Canvas1 . Width
y random integer from 1 to Canvas1 . Height
set RedBall . Radius to RedBall . Radius + 2
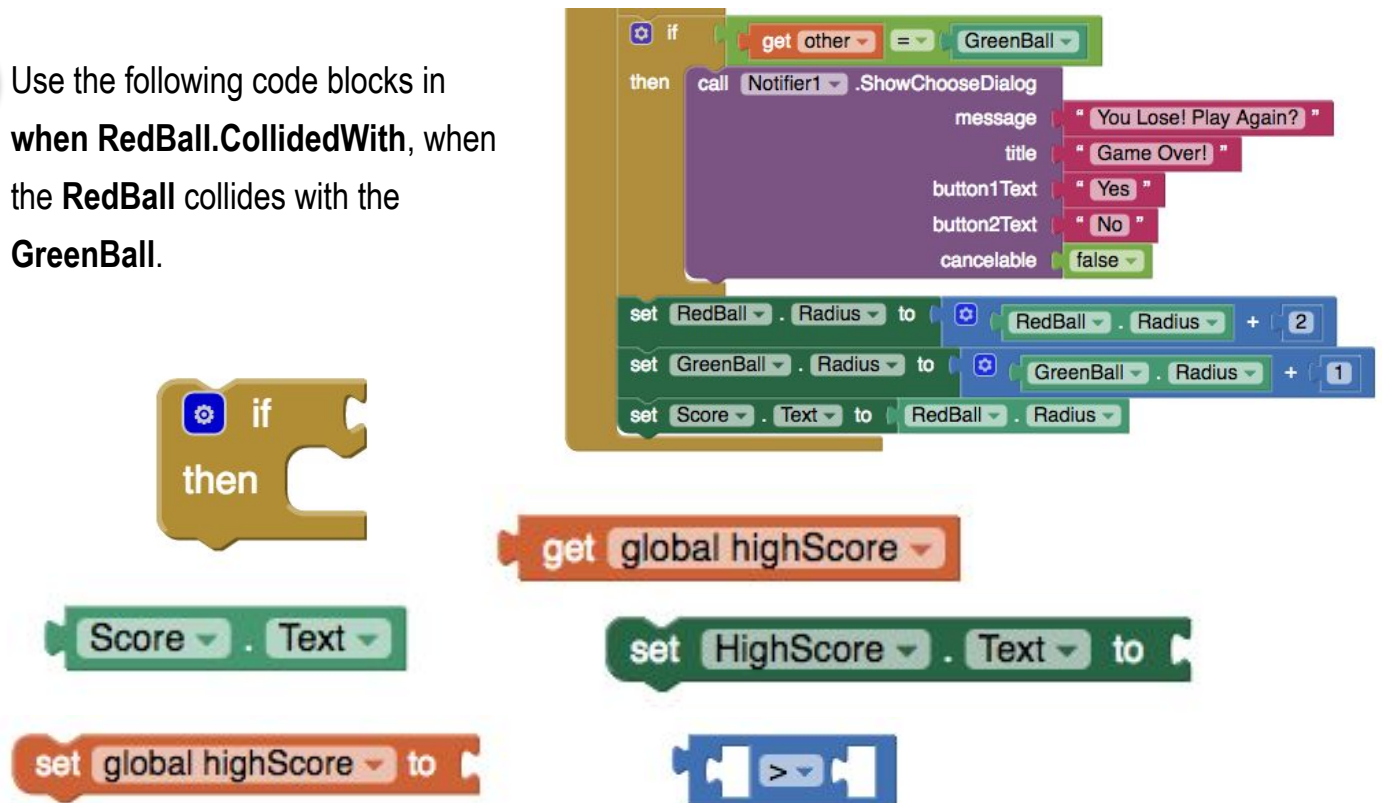set GreenBall . Radius to GreenBall . Radius + 1

3

MIT APP INVENTOR

SETTING THE HIGH SCORE

When the game ends, test if the current Score is higher than the current High Score. If it is, then the current Score becomes the new High Score.

8 Use the following code blocks in **when RedBall.CollidedWith**, when the **RedBall** collides with the **GreenBall**.
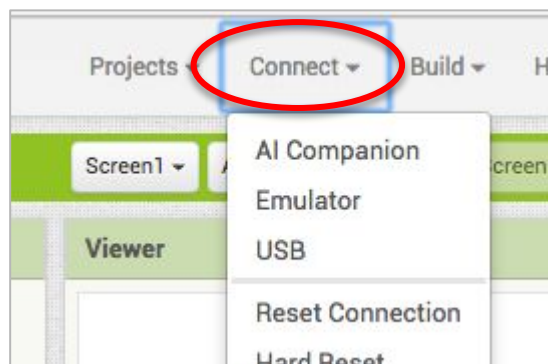
TESTING

9 Test with the MIT AI2 Companion.
- ○ Try playing the game and see if the high score changes.
- ○ Now close MIT AI2 Companion and run it again. Does the high score display correctly?

The high score gets set back to 0 because variables are not **persistent**, which means their values get erased when a program or app closes. You will use a new component, called **TinyDB**, to save the high score on the device, so it can be saved between different occasions of playing the game.

ABOUT TINYDB

Below is an explanation of what TinyDB is and how it works.

TinyDB is a component that stores "persistent" data--so it saves data even after you close the app for the next time you use it. It stands for "Tiny Database".
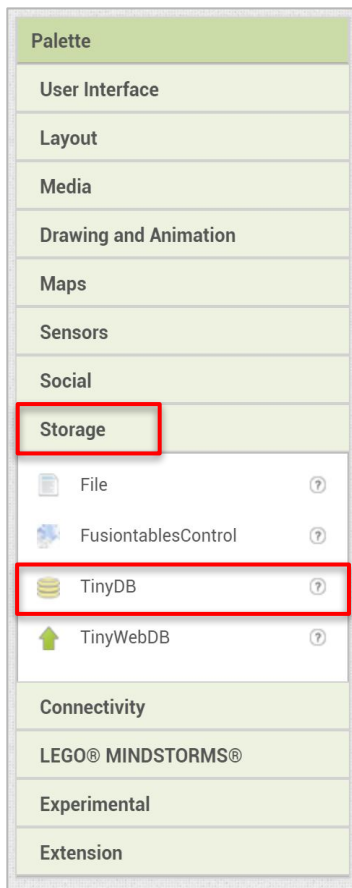
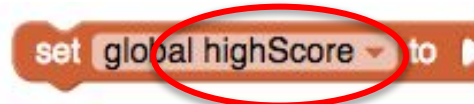You will use TinyDB to store the high score for the Food Chase app.

TinyDB has two main functions: **StoreValue** and **GetValue**.

❏ **StoreValue** stores a value, replacing whatever was in the database before. The name of the value is **tag** and the new value is **valueToStore**.

tag is like a variable name

valueToStore is like a value of a variable

❏ **GetValue** fetches a value from the database that was stored before, by its **tag**. If there's no value stored, then it returns **valueIfTagNotThere**.
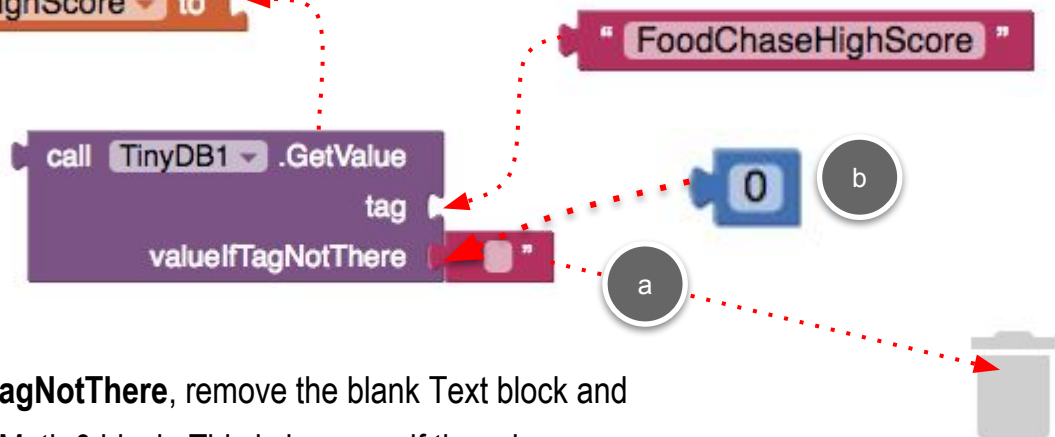
SAVING THE HIGH SCORE BETWEEN GAMES

Each time you save something to **TinyDB**, you save it with a tag. You'll use "**FoodChaseHighScore**" for your tag.

10 Set highScore in **Screen1.Initialize** to the value *Got* from **TinyDB1**.

A **tag** is just like a name, like the name of a variable

11 For the **valueIfTagNotThere**, remove the blank Text block and replace it with a Math **0** block. This is because if there is no high score yet, you'll use **0**.
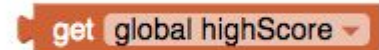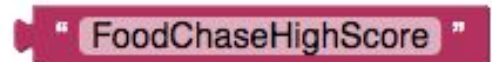
UPDATING THE HIGH SCORE

Last thing to do is to store the value of the high score whenever you get a new one.

12   Add the **TinyDB1.StoreValue** block to the **RedBall.CollidedWith** event to store the new high score.
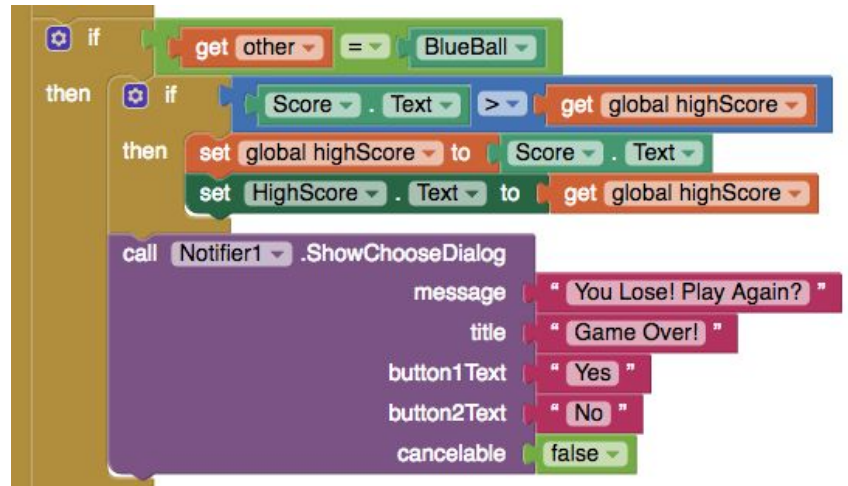
Get the correct sequence of blocks! Set the variable *before* you store it in **TinyDB.**

13   Test again! Now your high score should display correctly, even if you close the app and open it again!

**Choose Ways to
Extend Your App**

Here are a few features you could add if you want to expand your app

Add sounds! One for eating food and another for losing game

Make the GreenBall move faster as time goes by
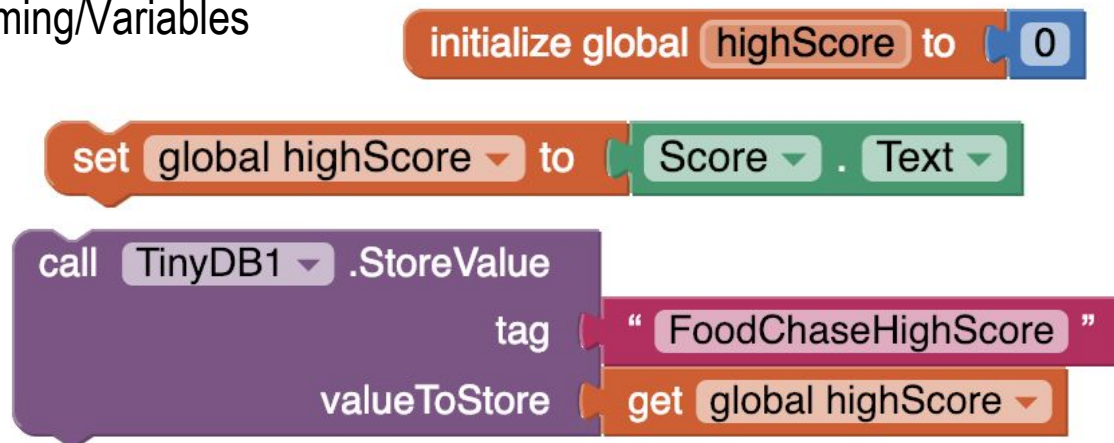
Make the Food Sprites move too

What other ideas do you have?
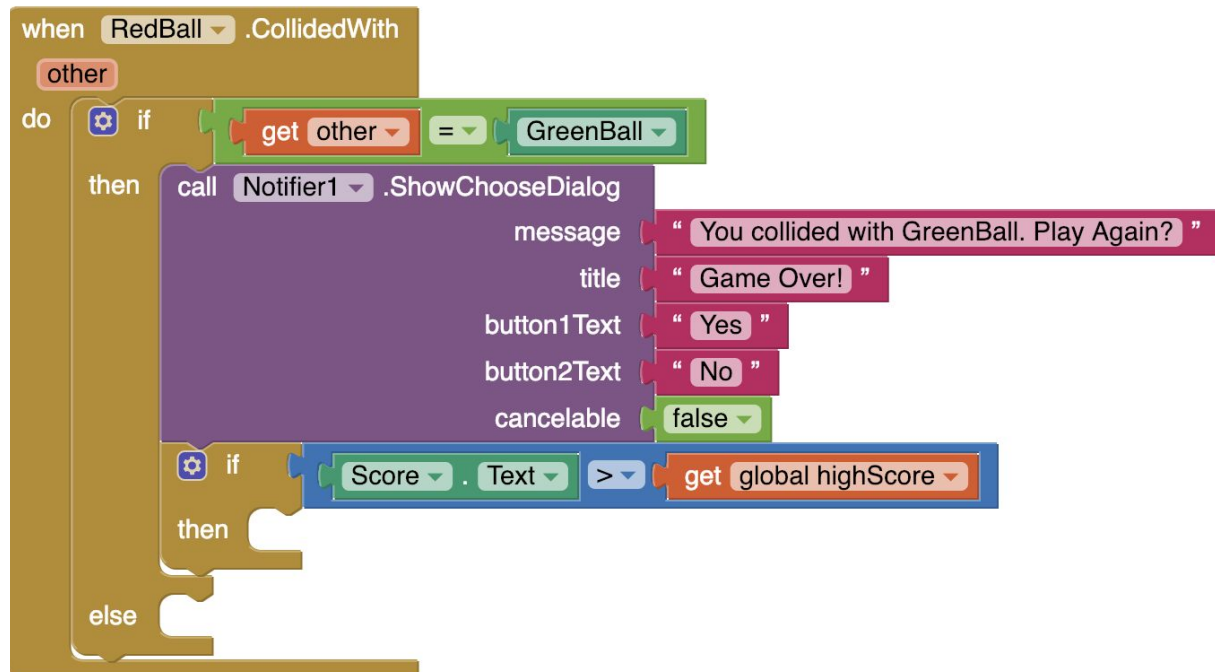
COMPUTATIONAL THINKING CONCEPTS and PRACTICES

The following are the Computational Thinking Concepts and Practices used in Part 4.

Food Chase Game
1. Naming/Variables
2. Conditionals

COMPUTATIONAL THINKING CONCEPTS and PRACTICES (continued)

The following are the Computational Thinking Concepts and Practices used in Part 4.

Food Chase Game
3. Manipulation of data and elementary data structures ![call TinyDB1.StoreValue tag "FoodChaseHighScore" valueToStore get global highScore] ![set global highScore to call TinyDB1.GetValue tag "FoodChaseHighScore" valueIfTagNotThere 0]
4.   Testing   and   Debugging ❏   Test again! Now your high score should display correctly, even if you close the app and open it again!