

# What is MIT App Inventor?

see if it's a good fit for your project!

- Block-based educational programming environment for developing apps and projects

## Benefits:

- Visual, build projects quickly
- May have a smaller learning curve than traditional code
- Test in real time with an android device

## Challenges

- Can only collaborate with a shared account; simultaneous access might result in loss of work (it is easier to exchange the project by exporting the file and collaborate on a single device)
- Most projects require androids for testing



**VS**

```
1 response = requests.request("POST", "https://large-text-to-speech.p.rapidapi.com/tts",
2 data=json.dumps({"text": text}), headers=headers)
3 id = json.loads(response.text)['id']
4 eta = json.loads(response.text)['eta']
5 print(f'Waiting {eta} seconds for the job to finish...')
6 time.sleep(eta)
7 response = requests.request("GET", "https://large-text-to-speech.p.rapidapi.com/tts",
8 headers=headers, params={'id': id})
9 while "url" not in json.loads(response.text):
10 response = requests.request("GET", "https://large-text-to-speech.p.rapidapi.com/tts",
11 headers=headers, params={'id': id})
12 print(f'Waiting some more...')
13 time.sleep(3)
14 url = json.loads(response.text)['url']
15 response = requests.request("GET", url)
16 with open(filename, 'wb') as f:
17     f.write(response.content)
18 print(f'File saved to {filename} ! \nOr download here: {url}')
```

<https://ai2.appinventor.mit.edu/>

# A Brief Tutorial!

some beginner documentation

# Design Interface

## Components

Rename components, upload files in Media

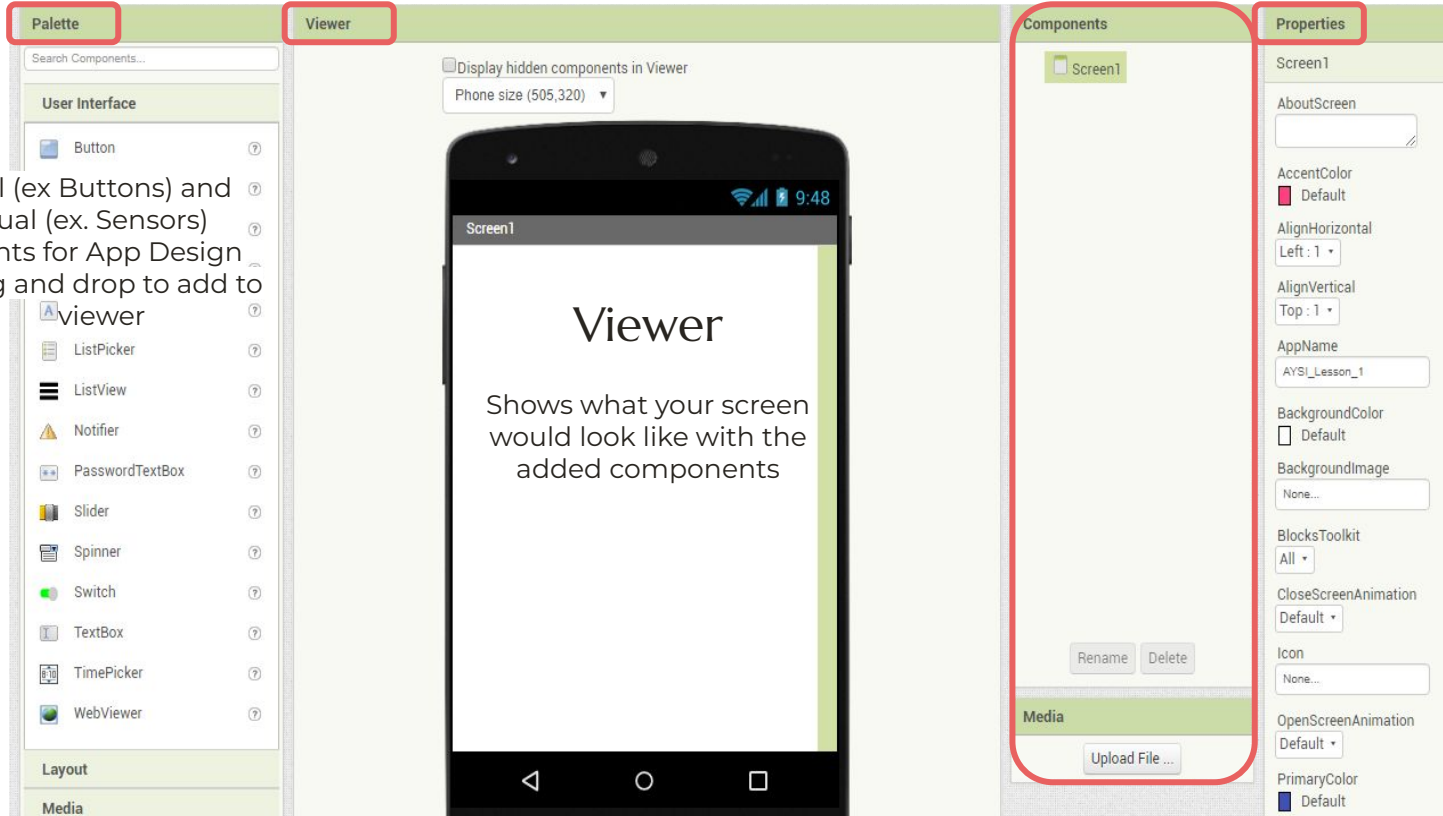
## Properties

Customize your components

## Palette

Get visual (ex Buttons) and non-visual (ex. Sensors) components for App Design

- Drag and drop to add to



# Blocks

Store code, can paste into other screens/projects

Built-in blocks are default and available for every app you make

Component blocks are specific to the components you add to your app

Click on a category or a component view the blocks corresponding to it, drag blocks into the viewer to use in code.

Shows errors in your code

Show Warnings

zoom

trash

**Blocks**

Built-in

- Control
- Logic
- Math
- Text
- Lists
- Dictionaries
- Colors
- Variables
- Procedures

Screen1

- lbl\_title
- btn\_calculate

Any component

Rename Delete

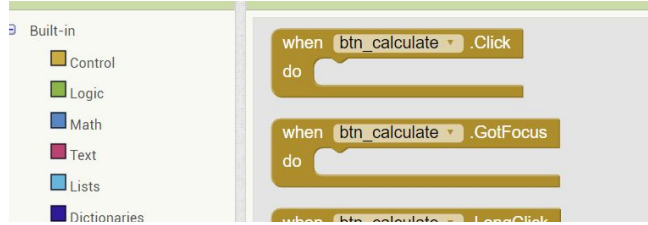
**Media**

Upload File ...

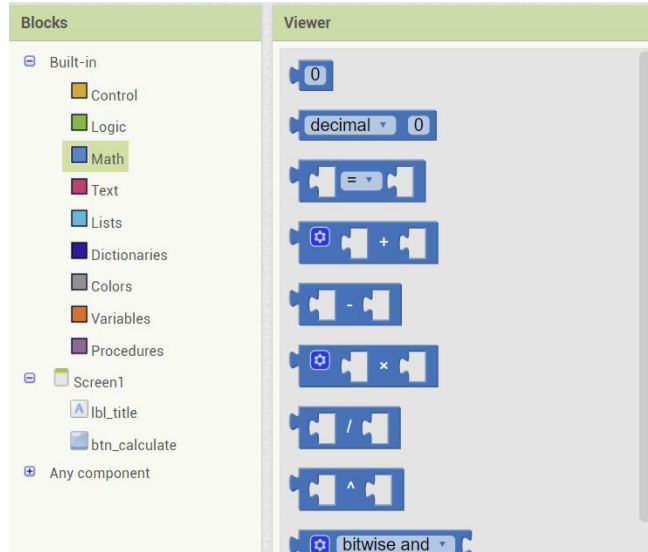
Viewer

# DATA TYPES

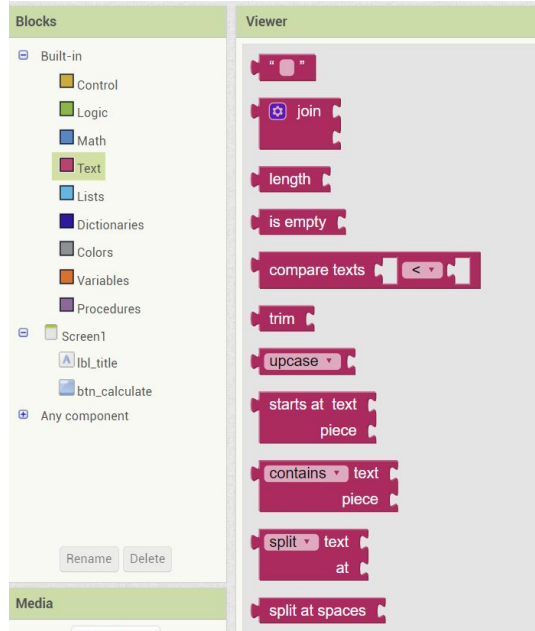
## Event Handlers



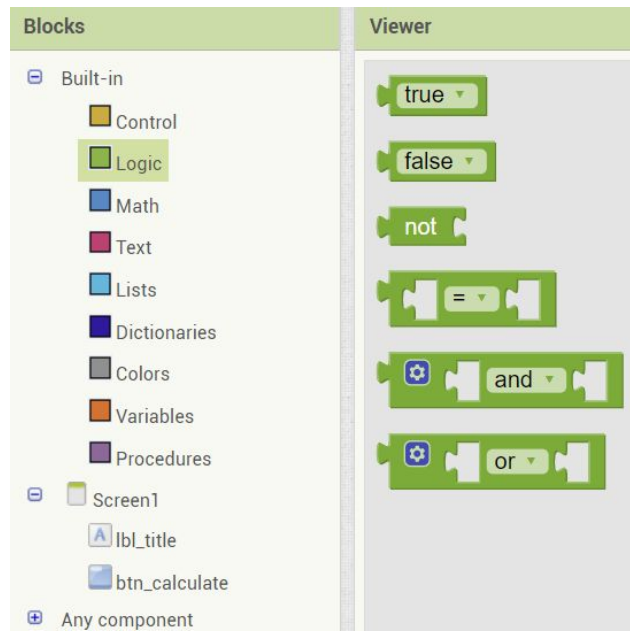
- Event handlers: “When \_\_\_\_” do something
- Other basic data types: Math (numbers), text (strings), logic (booleans)



## Math



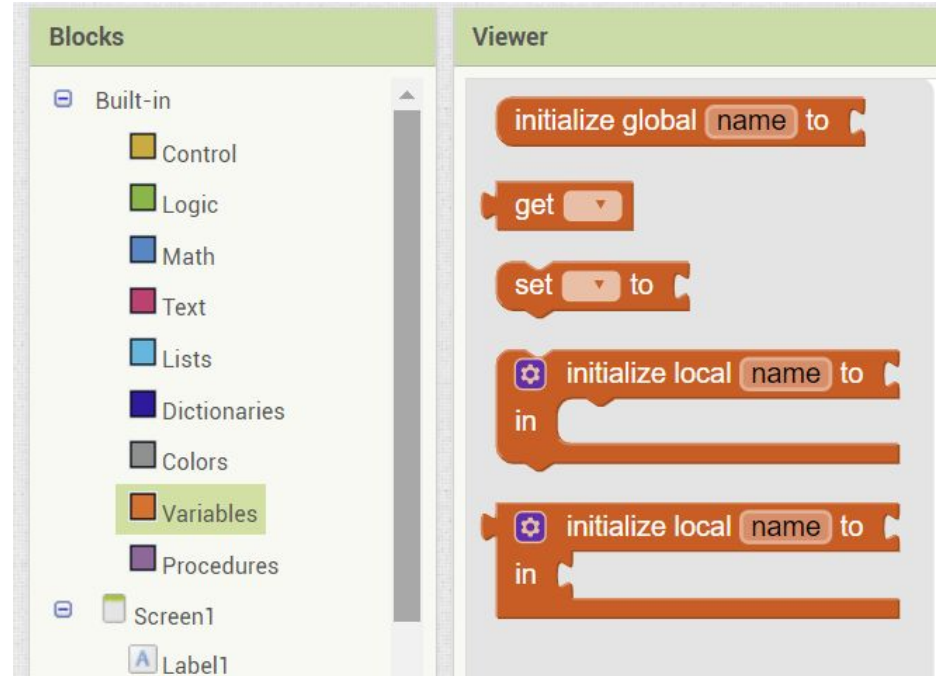
## Strings



## Logic

# VARIABLES

- Global variables: accessible within the whole program
- Local variables: accessible within the function/block it is defined in
- To initialize a variable: create a variable and set it to a starting value
- Get variable: retrieve the value of the variable
- Set variable: change the value of the variable



The image shows the Scratch interface with the 'Math' category selected in the 'Blocks' panel. The 'Viewer' panel displays a sequence of blocks: a '0' block, a 'decimal' block set to '0', an equals sign block, and a group of basic arithmetic blocks (addition, subtraction, multiplication, division, and exponentiation) which are highlighted by a red hand-drawn box. Below these are 'bitwise and', 'random integer from 1 to 100', 'random fraction', 'random set seed to', 'min', 'arithmetic mean (average)', and 'mode of a list' blocks. The 'Media' panel at the bottom has an 'Upload File ...' button.

Blocks

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Dictionaries
  - Colors
  - Variables
  - Procedures
- Screen1
  - lbl\_title
  - btn\_calculate
- Any component

Rename Delete

Media

Upload File ...

Viewer

0

decimal 0

=

+

-

×

÷

^

bitwise and

random integer from 1 to 100

random fraction

random set seed to

min

arithmetic mean (average)

mode of a list

# Math

Number data type used for calculations and counting

Basic operations

The image shows the Scratch interface with the 'Math' category selected in the 'Blocks' panel. The 'Viewer' panel displays a sequence of advanced math blocks: 'mode of a list', 'square root', 'absolute', 'neg', 'round', 'ceiling', 'floor', 'modulo of' (with an addition block), 'sin', 'cos', 'tan', 'atan2' (with 'y' and 'x' inputs), 'convert radians to degrees', 'format as decimal number' (with 'places' input), 'is number?' (with 'Show Warnings' button), and 'convert number base 10 to hex'. The 'Media' panel at the bottom has an 'Upload File ...' button.

Blocks

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Dictionaries
  - Colors
  - Variables
  - Procedures
- Screen1
  - lbl\_title
  - btn\_calculate
- Any component

Rename Delete

Media

Upload File ...

Viewer

mode of a list

square root

absolute

neg

round

ceiling

floor

modulo of +

sin

cos

tan

atan2

y

x

convert radians to degrees

format as decimal number

places

is number?

Show Warnings

convert number base 10 to hex

# Logic (Booleans)

- values: true, false

The image displays the Scratch interface, divided into two main sections: 'Blocks' and 'Viewer'.

**Blocks Section:**

- Built-in:**
  - Control:** Represented by a tan icon.
  - Logic:** Represented by a green icon and is currently selected.
  - Math:** Represented by a blue icon.
  - Text:** Represented by a pink icon.
  - Lists:** Represented by a light blue icon.
  - Dictionaries:** Represented by a dark blue icon.
  - Colors:** Represented by a grey icon.
  - Variables:** Represented by an orange icon.
  - Procedures:** Represented by a purple icon.
- Screen1:**
  - lbl\_title:** Represented by a light blue icon.
  - btn\_calculate:** Represented by a blue icon.
- Any component:** Represented by a blue icon with a plus sign.

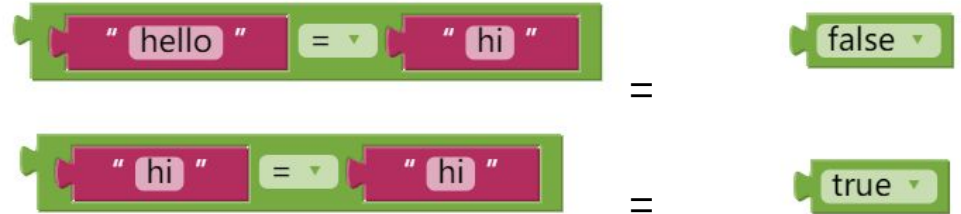
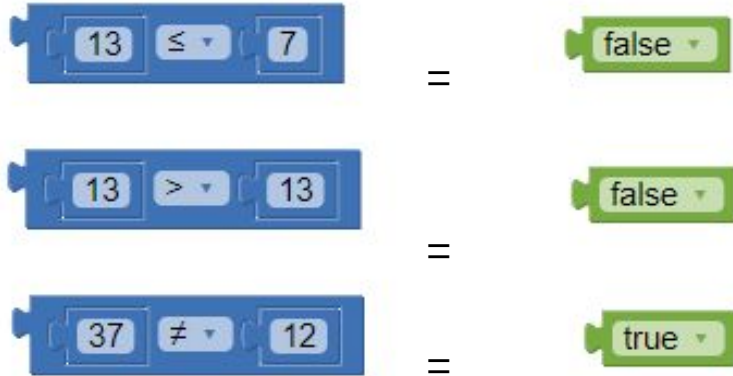
**Viewer Section:**

The Viewer area shows a stack of Logic blocks:

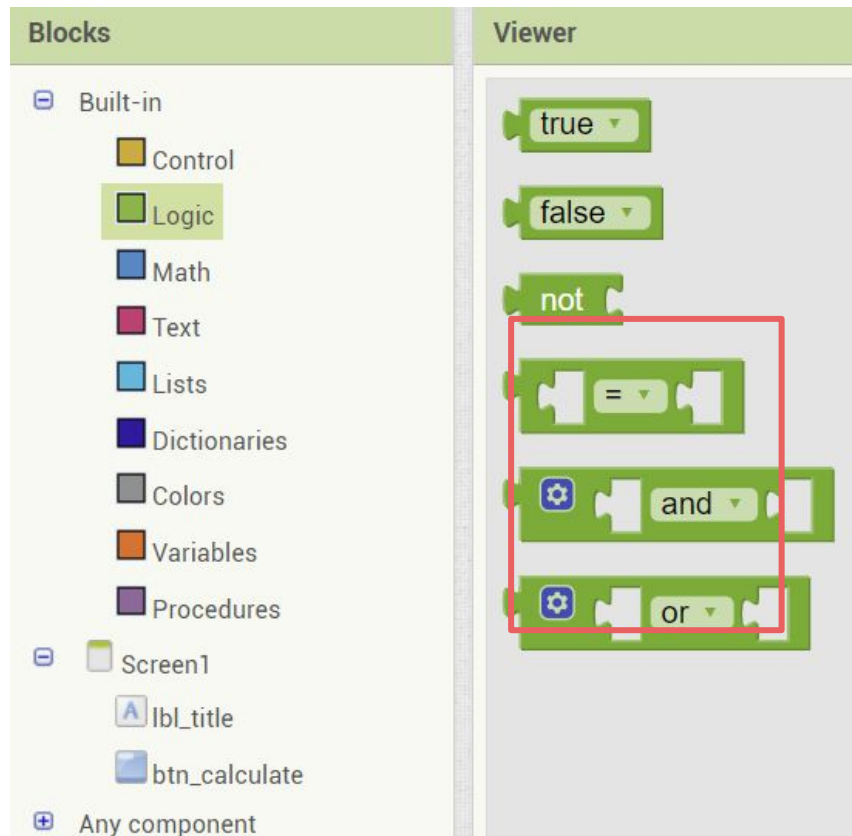
- A red rectangle highlights the first two blocks: a 'true' block and a 'false' block, both with dropdown arrows.
- Below them is a 'not' block.
- Next is an equals sign (=) block.
- Then an 'and' block with a settings gear icon.
- Finally, an 'or' block with a settings gear icon.



# Boolean Examples (Comparisons)



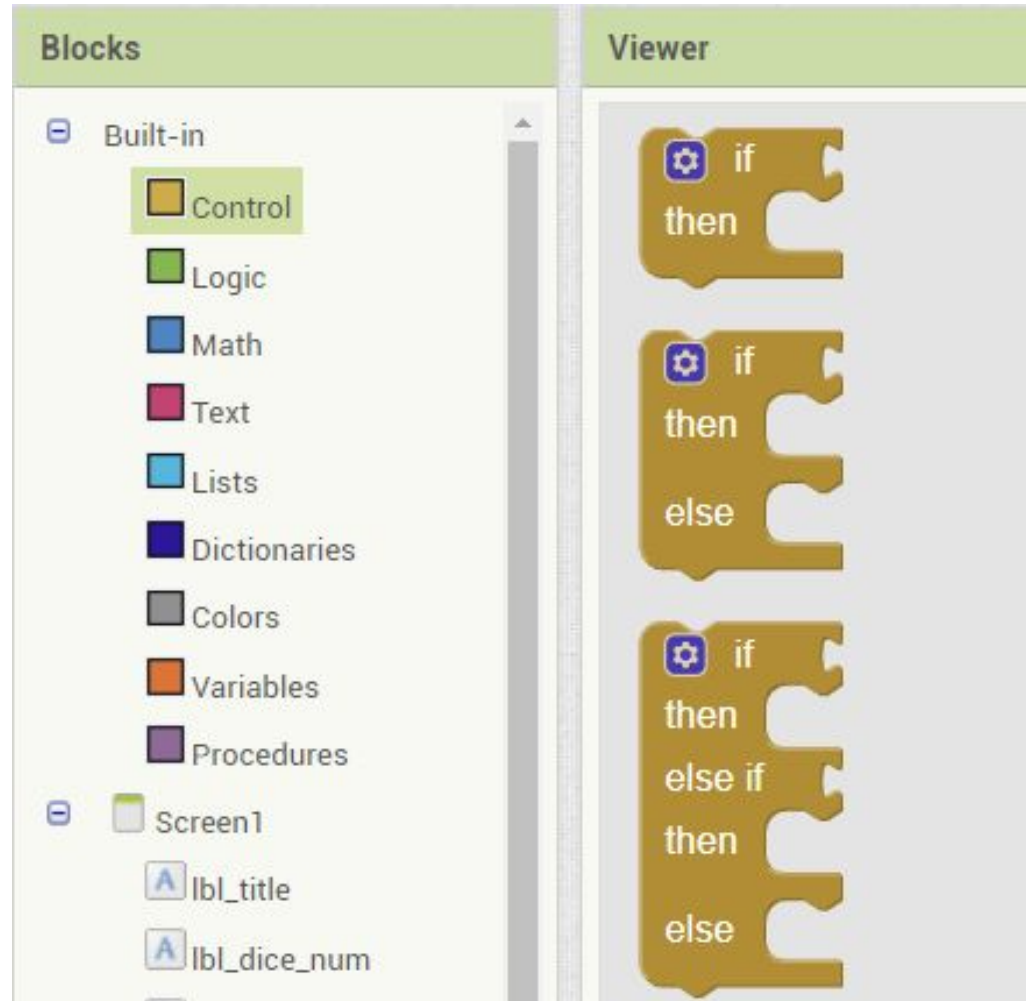
# Logic Operators



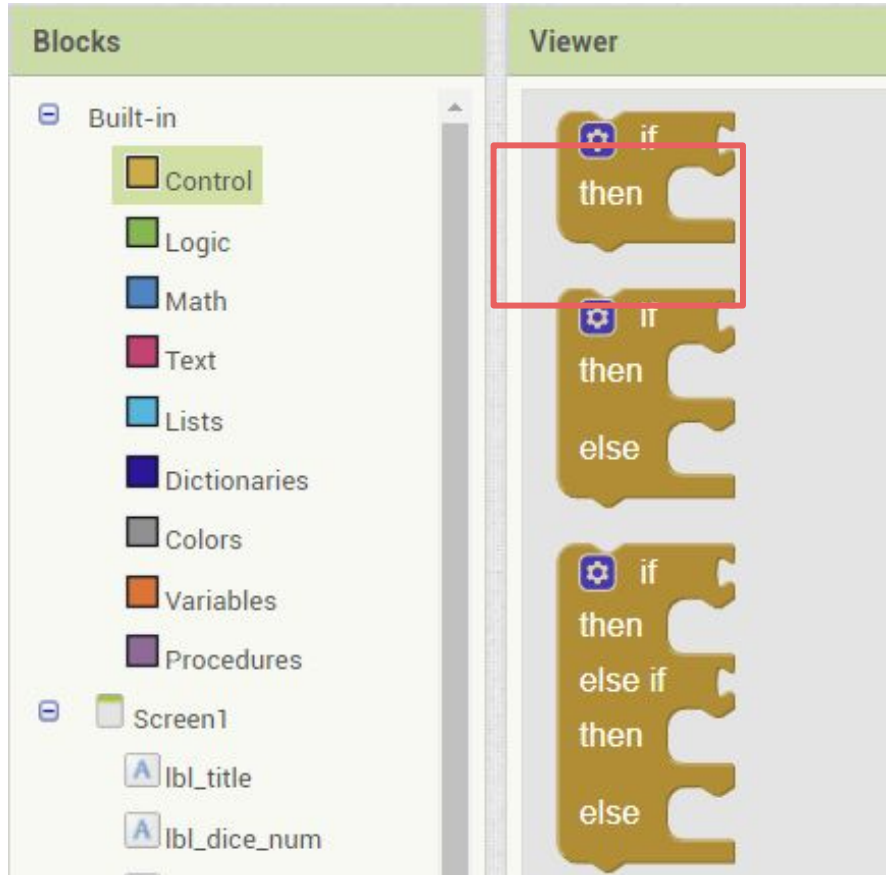
- **Not:** reverses the value
  - Not (true)  $\Rightarrow$  false
- **And:** considers both inputs
  - T and T  $\Rightarrow$  T
  - T and F  $\Rightarrow$  F
- **Or:** if one input is true, returns true
  - T or T  $\Rightarrow$  T
  - T or F  $\Rightarrow$  T
  - F or F  $\Rightarrow$  F

# Conditional Statements

- If statements and For loops



# If Statements



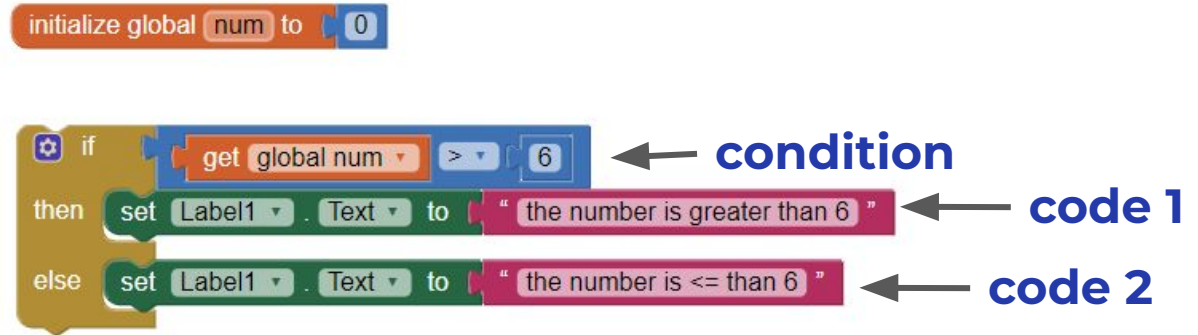
initialize global num to 0



**Code to be Executed**

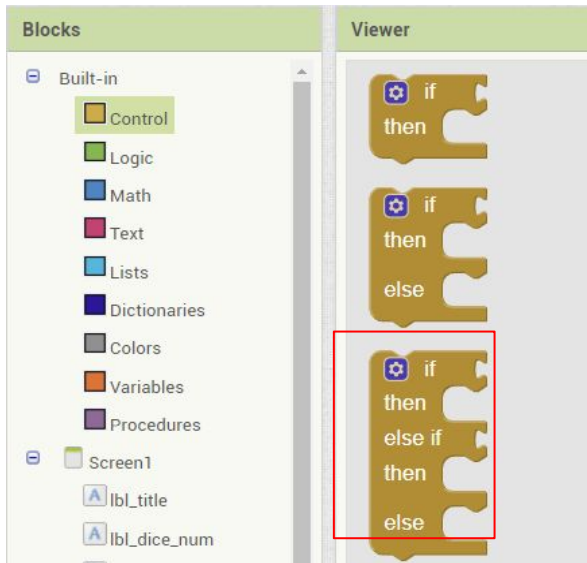
- **If** [condition] is true, **then** execute the code

# If-Else Statements

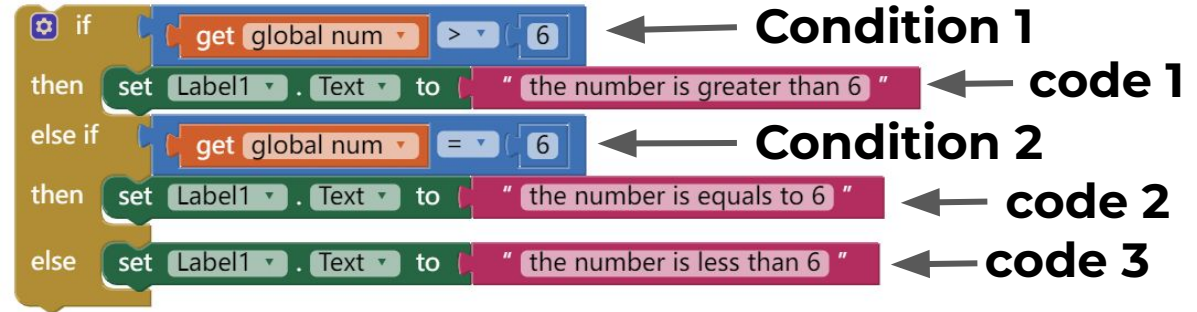


- **If** [condition] is true, **then** execute code 1
- **Else**, execute code 2

# If-Else If- Else Statements



initialize global num to 0



- **If** [Condition 1] is true, **then** execute the code 1
  - **Else if** [Condition 2] is true, **then** execute code 2
  - **Else**, execute code 3
- \*can add more else-if's if needed

# If-Else If- Else Statement Examples

- If Age over 18, you are an adult
  - Else if Age under 13, you are a child
  - Else you are a teenager
- 
- If battery over 80%, battery is high
  - Else if battery under 20%, battery is low
  - Else battery is medium

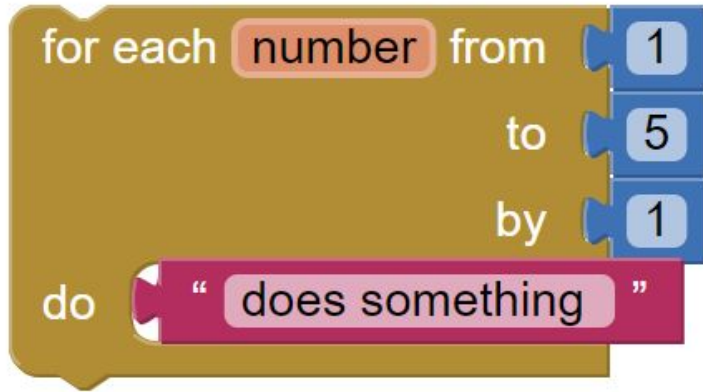
# What are Loops?

- Loop: repeats code until a specific condition is met
- main types are **for-loop** and **while loop**





# For-Loop



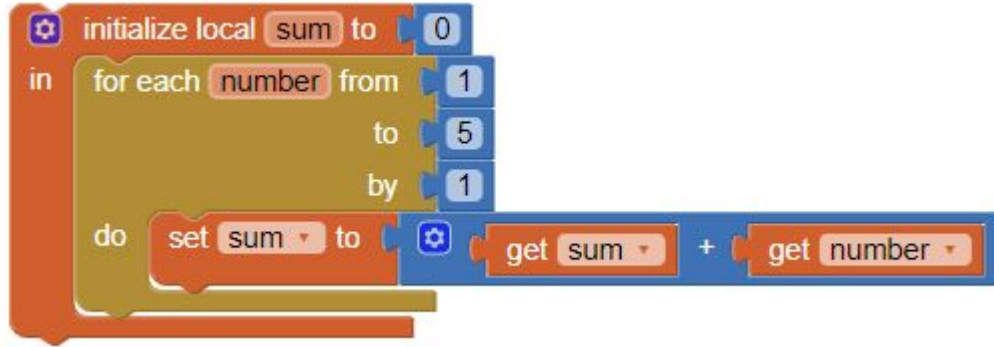
**starting value**

**ending value**

**increment:** value “**number**” increases by after each iteration

- Repeats code for a specific amount of times
  - When “**number**” reaches ending value, loop stops

# For-Loop Example: Calculate Sum



- The above method calculates  $1+2+\dots+5$

	Number = 1	Number = 2	Number = 3	Number = 4	Number = 5
Sum = 0	Sum = 0 + 1 = 1	Sum = 1 + 2 = 3	Sum = 3 + 3 = 6	Sum 6 + 4 = 10	Sum = 10 + 5 = 15

# For-Loop Examples in Real Life

For each item in cart

- Add [item price] to total price paid

For each friend in class

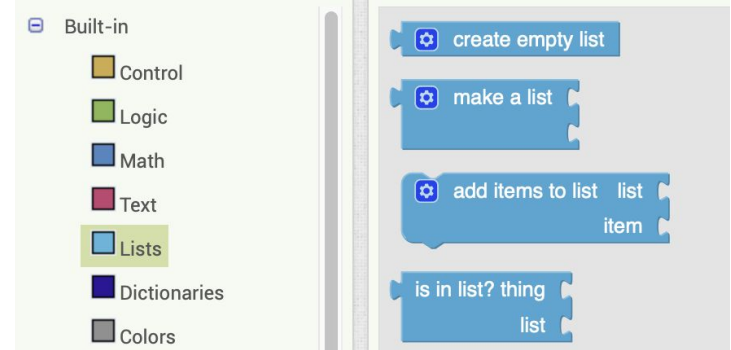
- Generate a custom gift box

For each item on shelf

- +1 to inventory count

# Lists

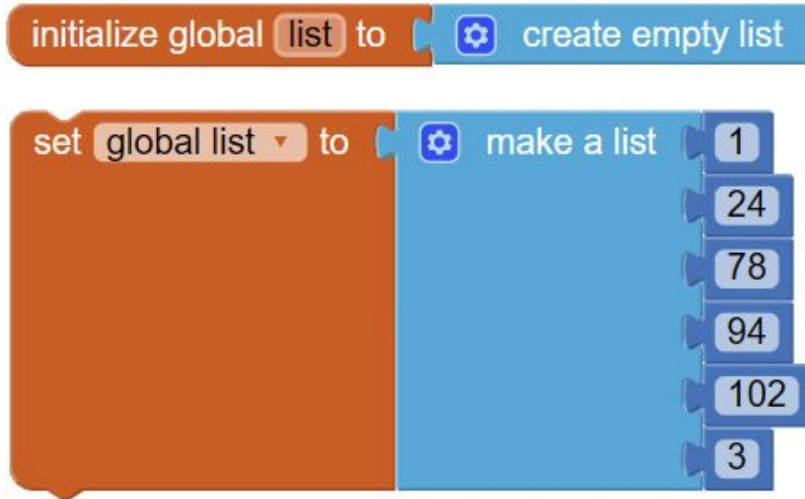
- **are a data structure**
- Items in a list are separated by commas and have indexes (position in list)
  - In App Inventor, the first element in a list is index 1
    - Note: in other programming languages it is sometimes 0
  - Useful when addressing specific element in a list



**data structure: special way  
of storing data**

# List Example

1, 24, 78, 94, 102, 3



Returned value: 102



Returned value: 1



What value will be returned?

Ans: 78

# Resources

# Tutorials by App Inventor

Doing a few quick ones may help familiarize and show you useful resources for your project!

<https://appinventor.mit.edu/explore/ai2/tutorials>

<https://appinventor.mit.edu/explore/app-building-guides>

<https://appinventor.mit.edu/explore/ai-with-mit-app-inventor>

Some of these apps might also teach you how to use Artificial Intelligence in your projects, or import extensions to elevate your project.

# **Starter Kit Project Walkthrough**



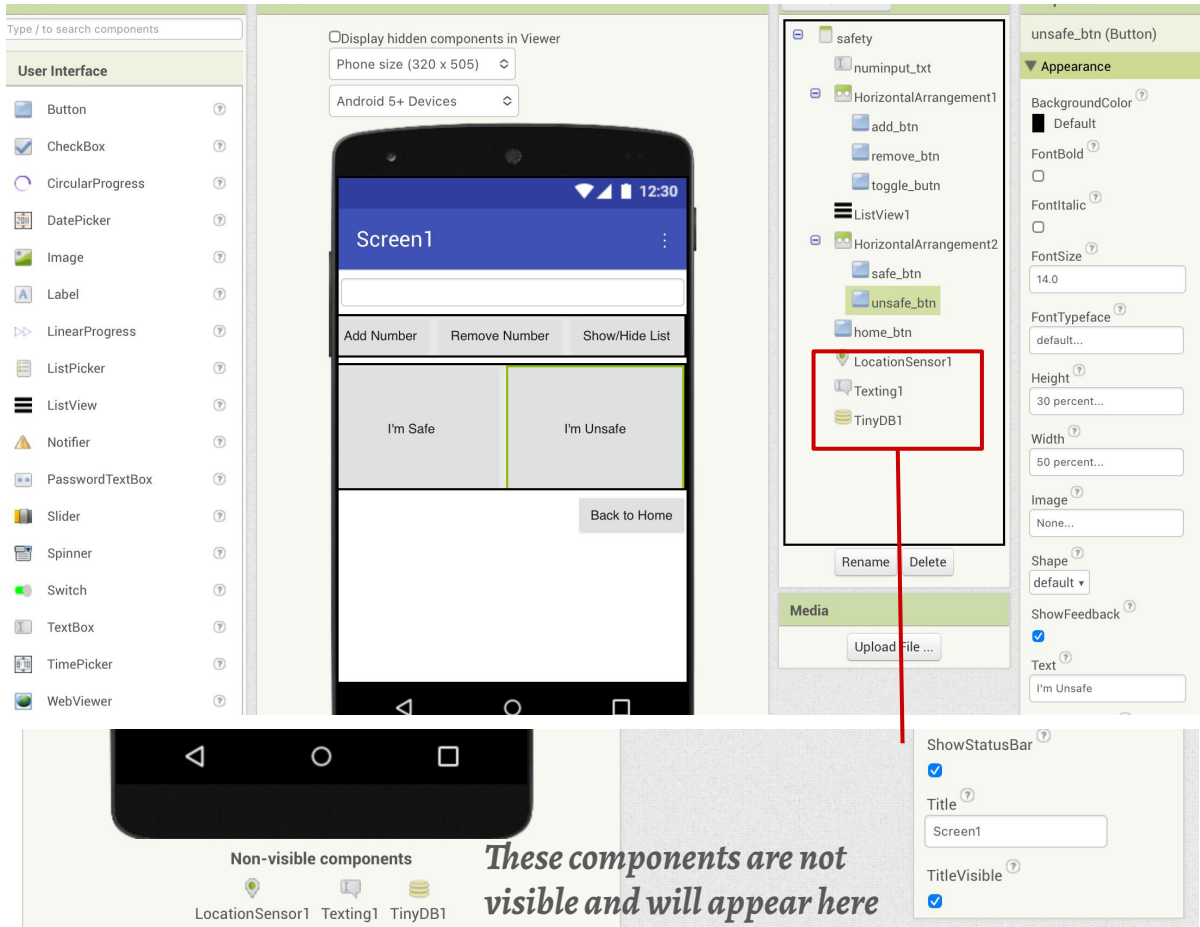
This example project is a crisis companion that stores a user's list of emergency contact phone numbers and reports the location and safety status of the user easily.

# Step 1: User Interface

Add components from the palette on the left by dragging and dropping into the viewer

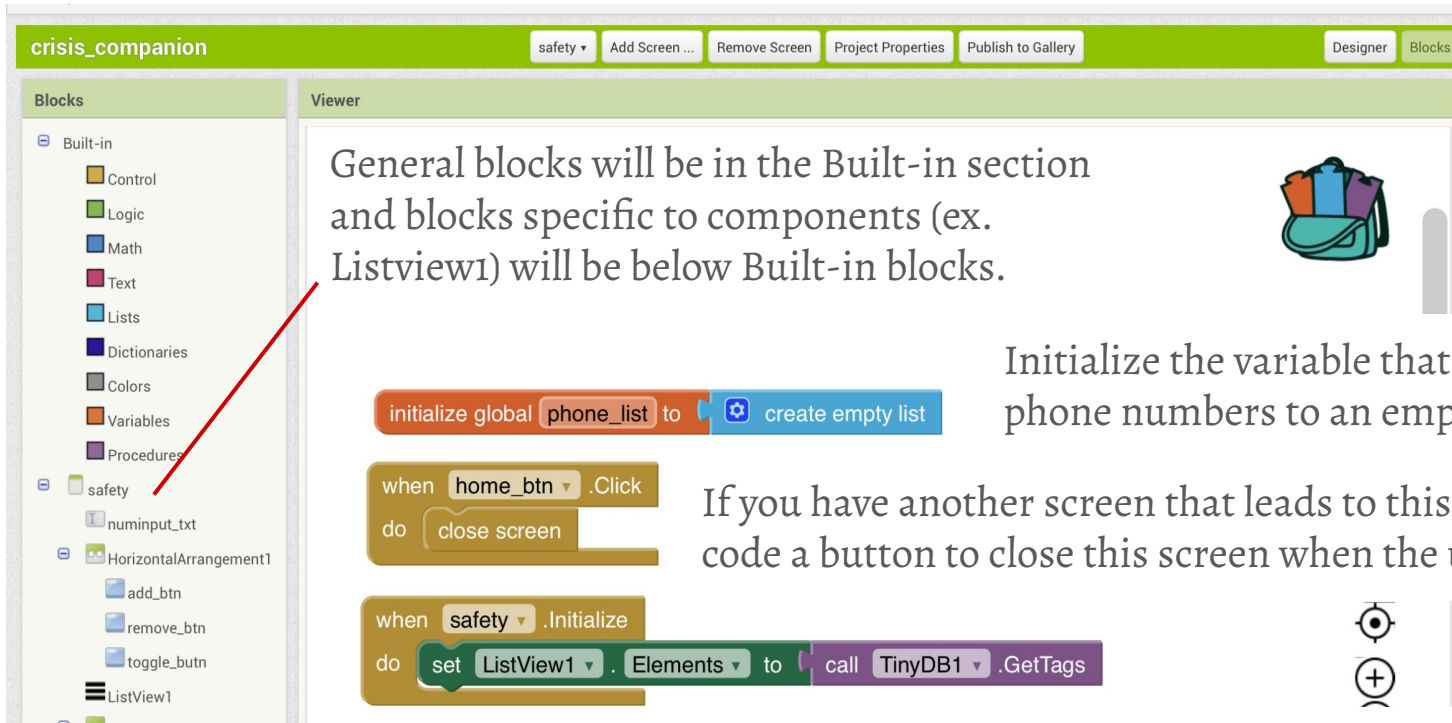
Adjust appearance (height, width, alignment, font, color, etc.) to your preference. You can also rename components for organization and easier coding later.

**Note:** Arrangements allow you to group objects horizontally, vertically, or in tables



# Step 2: Block Code

switch from designer to blocks



**crisis\_companion** safety Add Screen ... Remove Screen Project Properties Publish to Gallery Designer Blocks

**Blocks**

- Built-in
  - Control
  - Logic
  - Math
  - Text
  - Lists
  - Dictionaries
  - Colors
  - Variables
  - Procedures
- safety
  - numinput\_txt
  - HorizontalArrangement1
    - add\_btn
    - remove\_btn
    - toggle\_btn
  - ListView1

**Viewer**

General blocks will be in the Built-in section and blocks specific to components (ex. Listview1) will be below Built-in blocks.

Initialize the variable that represents the list of phone numbers to an empty list

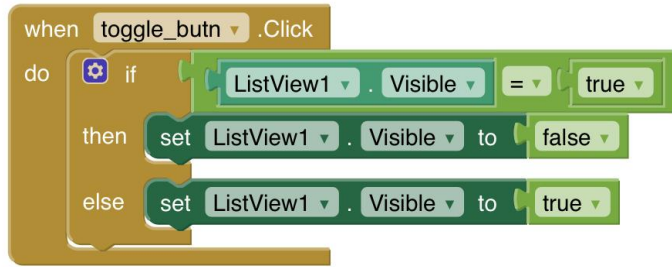
```
initialize global phone_list to create empty list
```

If you have another screen that leads to this screen you can make code a button to close this screen when the user wishes to

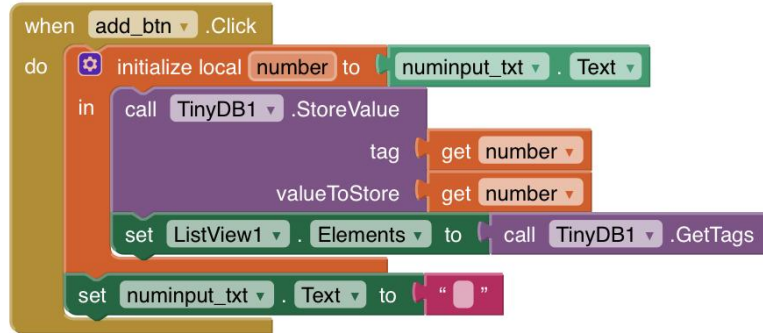
```
when home_btn.Click do close screen
```

when safety.Initialize do set ListView1.Elements to call TinyDB1.GetTags

TinyDB is a local storage. We will store phone numbers here so the user does not have to retype them every time the app is opened.



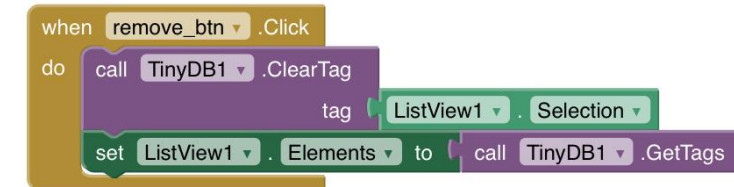
Sometimes having the list in sight is cumbersome. Maybe we can allow the user to show and hide it with these logic statements.



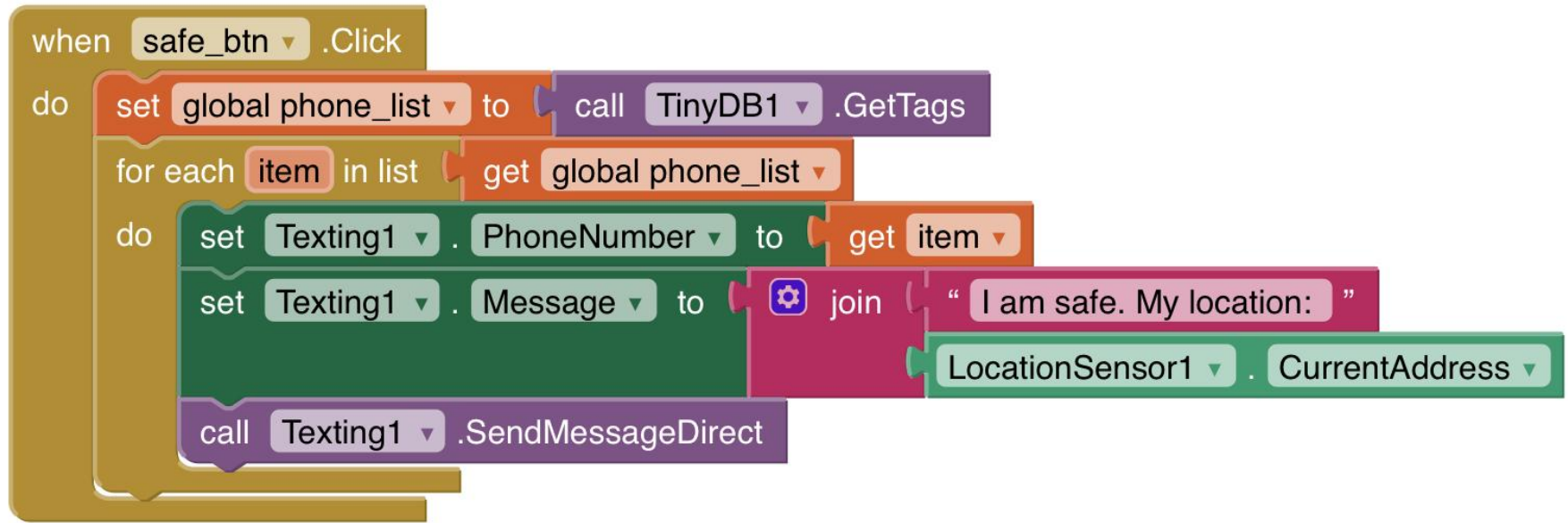
This block of code says that when the “add” button is clicked, we store the phone number that the user inputted into our TinyDB under the tag and value of the number. Then we set the ListView Elements to display the numbers.



When the user selects an item in the list, the option to remove it appears



When the remove button is clicked, the tag corresponding to the phone number is cleared from TinyDB and the ListView elements are reset to reflect that change.



When the user clicks the button saying that they are safe, we set our global variable, the list of phones, to the numbers stored in TinyDB. Then, for each phone number in the list, we call the texting component to send our address \*from the Location Sensor) directly to the contact.



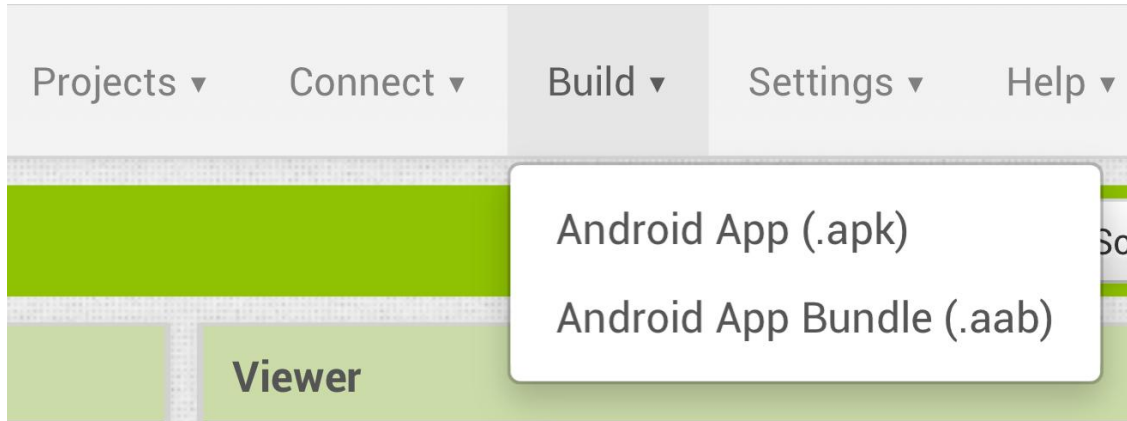
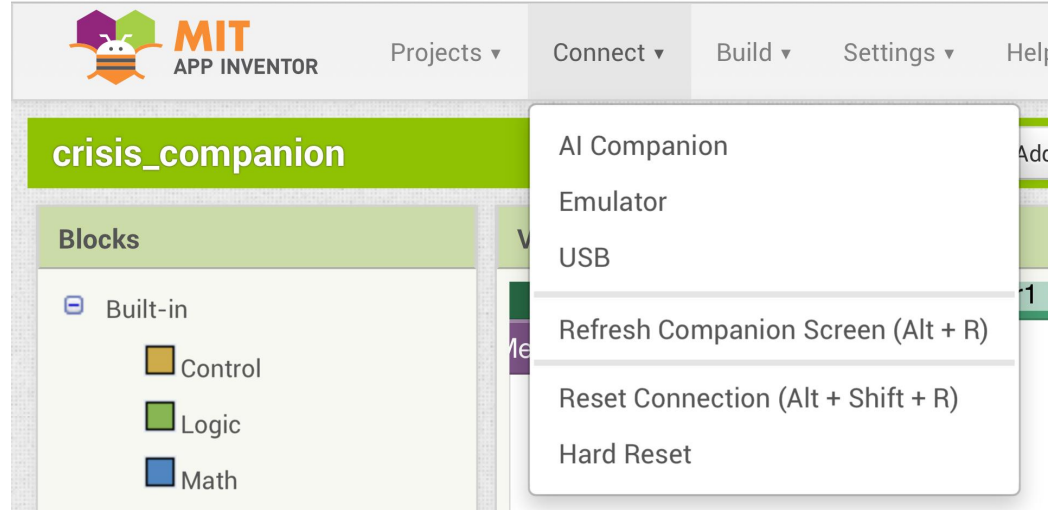
When the user clicks the button that they are unsafe, the same process happens but we change the content of the message.

# Testing and Downloading

To test: download the MIT App inventor app on your phone and connect to the AI Companion. Certain apps with extensions will only work on android devices. Also, because we used direct texting due to the nature of the app, some device permissions might block this function.

8:22 AM Sat May 10

ai2.appinventor.mit.edu — Private



To download: If you have an android device or can borrow one, you can download your app onto your device with a .apk file. Once this app is downloaded, you can allow permissions for direct texting and location.

# Some Considerations

- Projects always have room for improvement! For example: in the Properties, we can make the text box only have numerical text.
  - Here's another example: maybe we also want the list to refuse to add the text input when the input is blank, or doesn't fit a normal phone number format to avoid error. We can do this with an if statement.
- MIT App Inventor is an educational program that is great for making prototypes, not polished commercial projects. In real use, projects might have a lot more considerations like privacy of user information, etc. Your project does not have to be perfect and ready for commercial use, just try to make your broad idea creative and display it to the best of your abilities.



# How Can You Build Upon This App?

- Improve UI/UX
  - There are endless ways to improve the User Interface and User Experience. You can use tools like Figma to create a more appealing interface or use components like the contact picker for easier adding of emergency contacts
- Implement speech recognizer and text to speech components for easier use while commuting, for example, and accessibility
- Perhaps the user accidentally pressed the button. Is there a way you can give the user a countdown timer to cancel before sending the message?
- This screen might be one of multiple in your project. Connect this screen to a Home Screen that has other functions. Possible ideas for other screens:
  - Screen that displays emergency and disaster tips, hotlines, basic information guide
  - A digital alarm screen (can you make the app play a loud sound or do something to attract attention when the user needs? Could this be a useful feature in real life?)