What is MIT App Inventor?

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

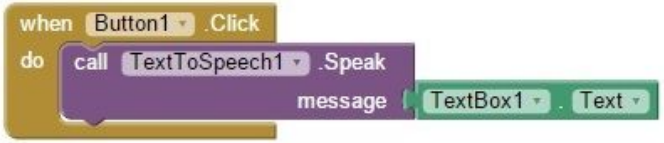
- Block-based educational programming environment for developing apps and projects

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

```
response = requests.request("POST", "https://large-text-to-speech.p.rapidapi.com/tts", data=json.dumps({"text": text}), headers=headers)
id = json.loads(response.text)['id']
eta = json.loads(response.text)['eta']
print(f'Waiting {eta} seconds for the job to finish...')
time.sleep(eta)
response = requests.request("GET", "https://large-text-to-speech.p.rapidapi.com/tts", headers=headers, params={'id': id})
while "url" not in json.loads(response.text):
    response = requests.request("GET", "https://large-text-to-speech.p.rapidapi.com/tts", headers=headers, params={'id': id})
    print(f'Waiting some more...')
    time.sleep(3)
url = json.loads(response.text)['url']
response = requests.request("GET", url)
with open(filename, 'wb') as f:
    f.write(response.content)
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

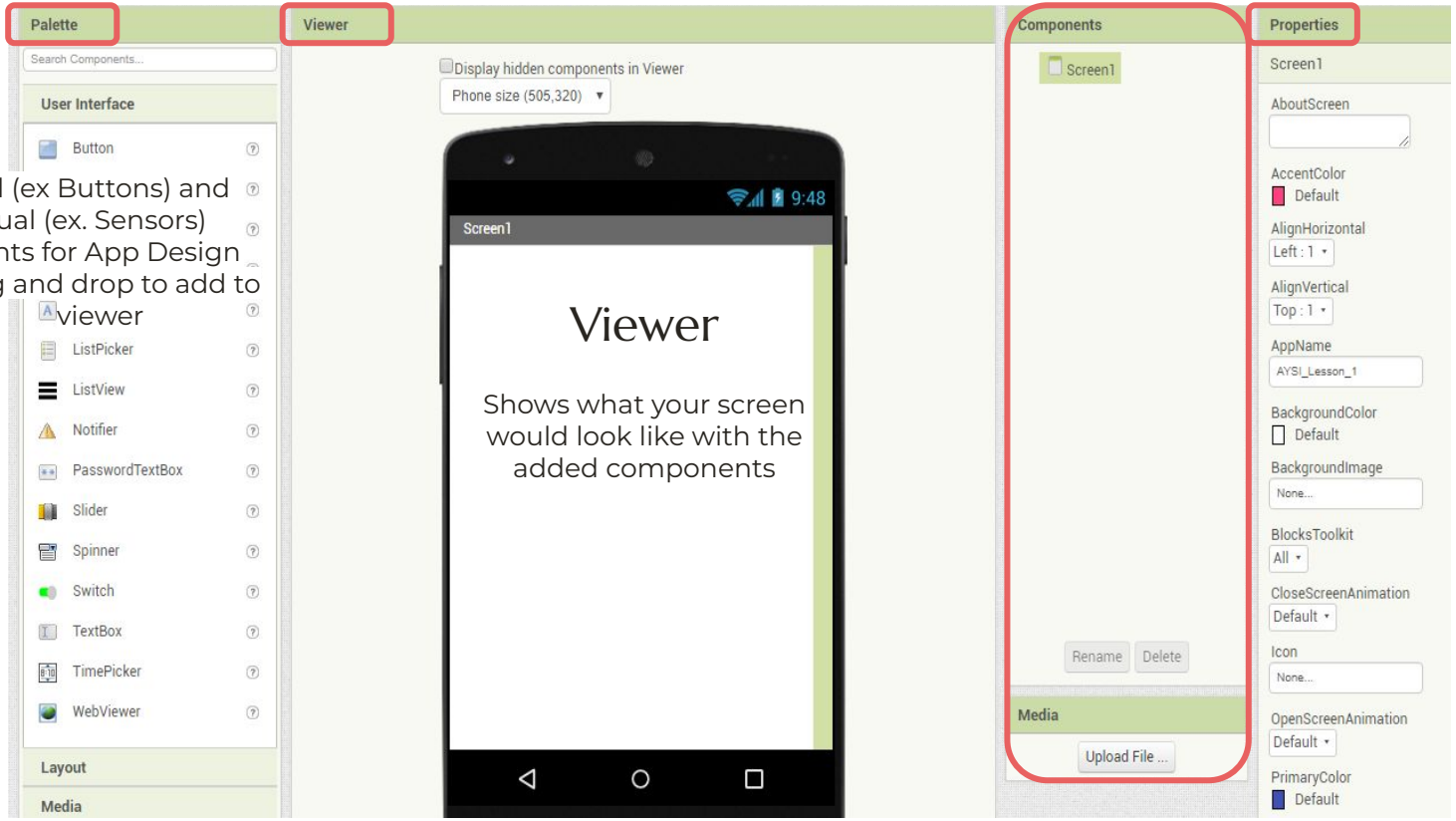
Properties

Customize your components

Palette

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

- Drag and drop to add to viewer



Blocks

Store code, can paste into other screens/projects

Blocks

Viewer

Built-in

Control

Logic

Math

Text

Lists

Dictionaries

Colors

Variables

Procedures

Screen1

lbl_title

btn_calculate

Any component

Rename Delete

Media

Upload File ...

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.

zoom

trash

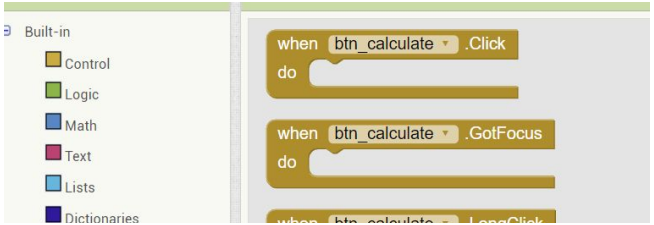
0 0

Show Warnings

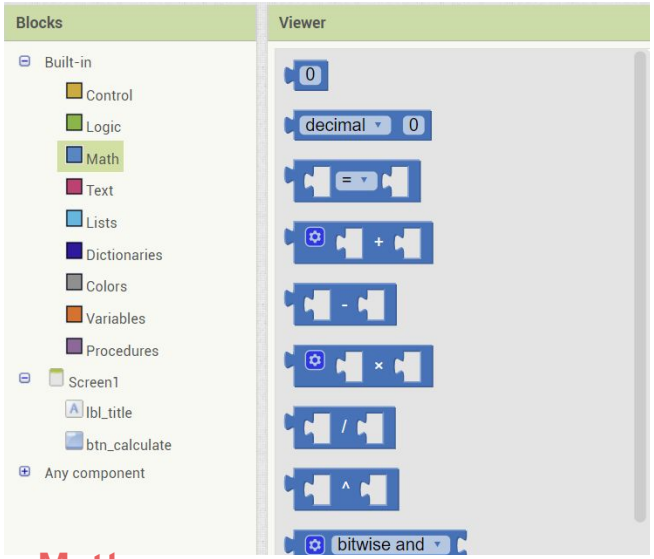
Shows errors in your code

DATA TYPES

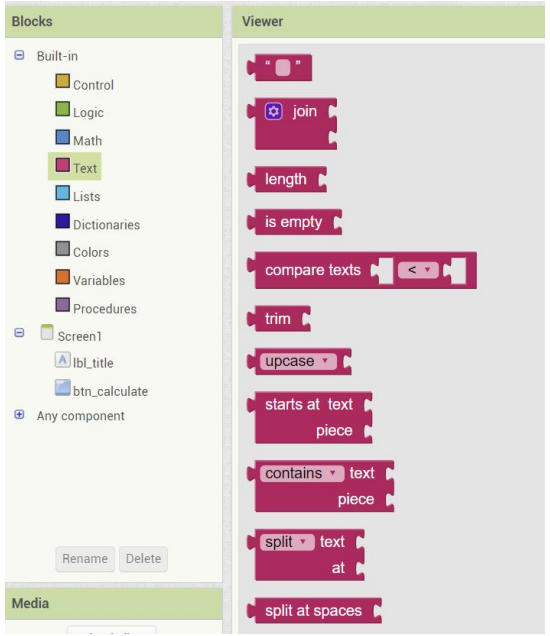
Event Handlers



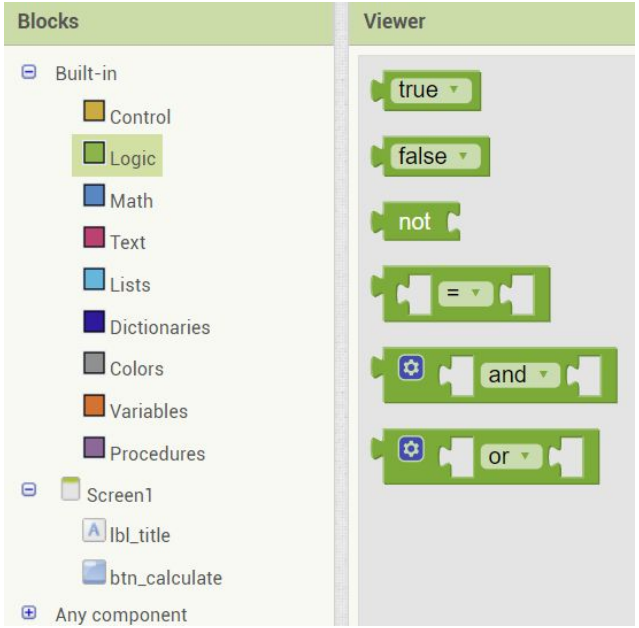
- Event handlers: “When ____” do something
- Other basic data types: Math (numbers), text (strings), logic (booleans)



Math



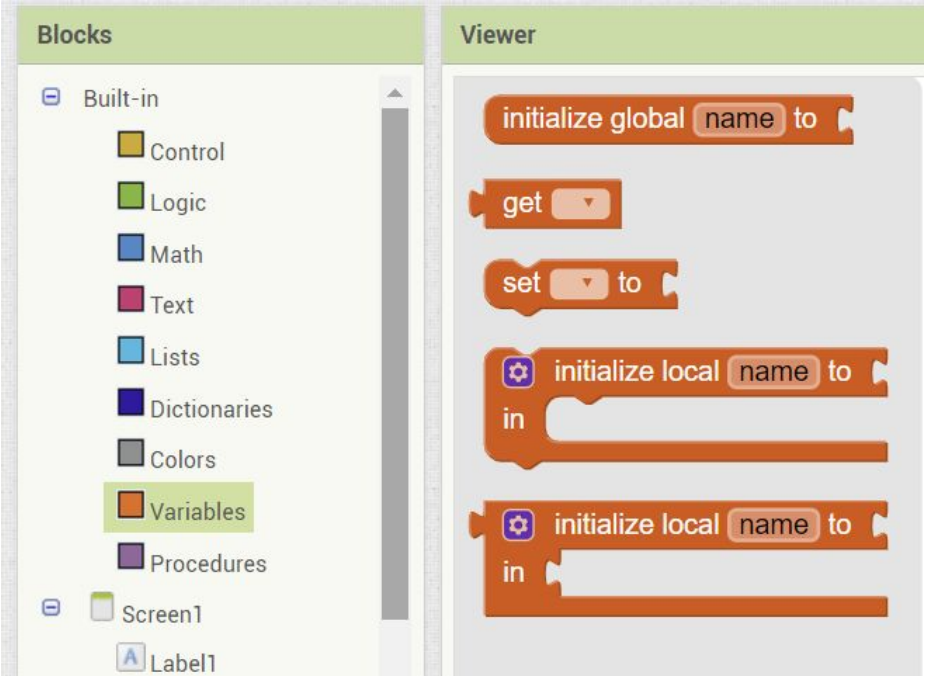
Strings



Logic

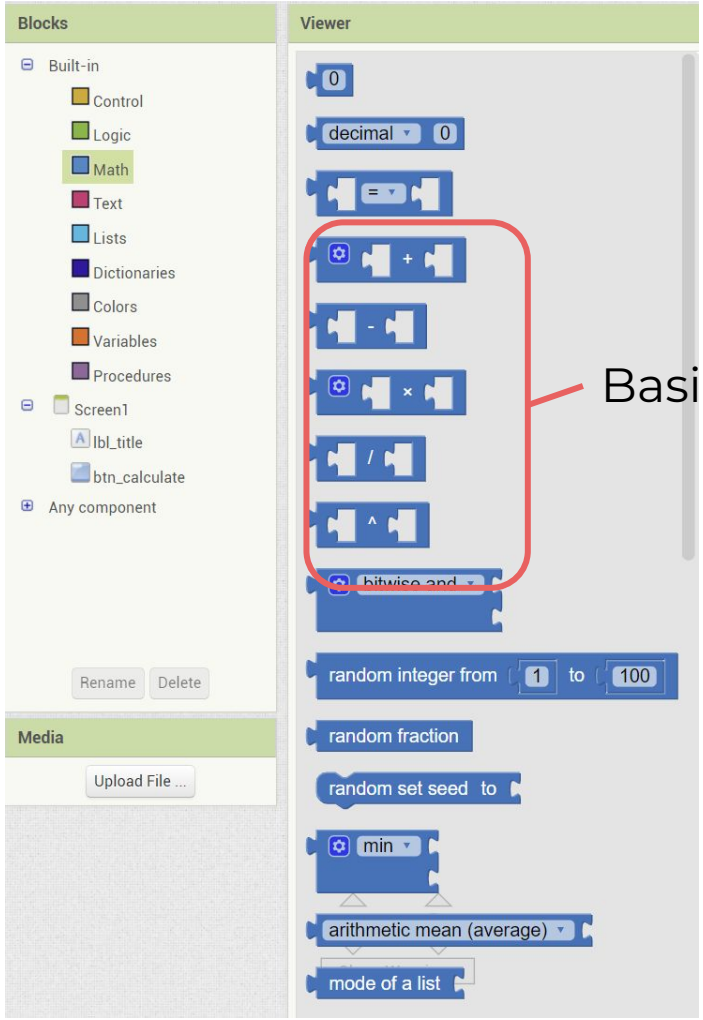
VARIABLES

- 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

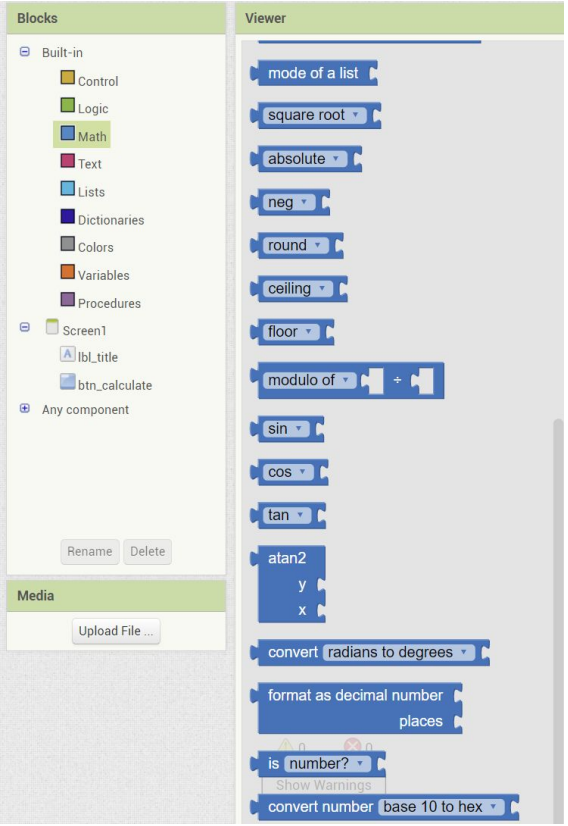


Math

Number data type used for calculations and counting

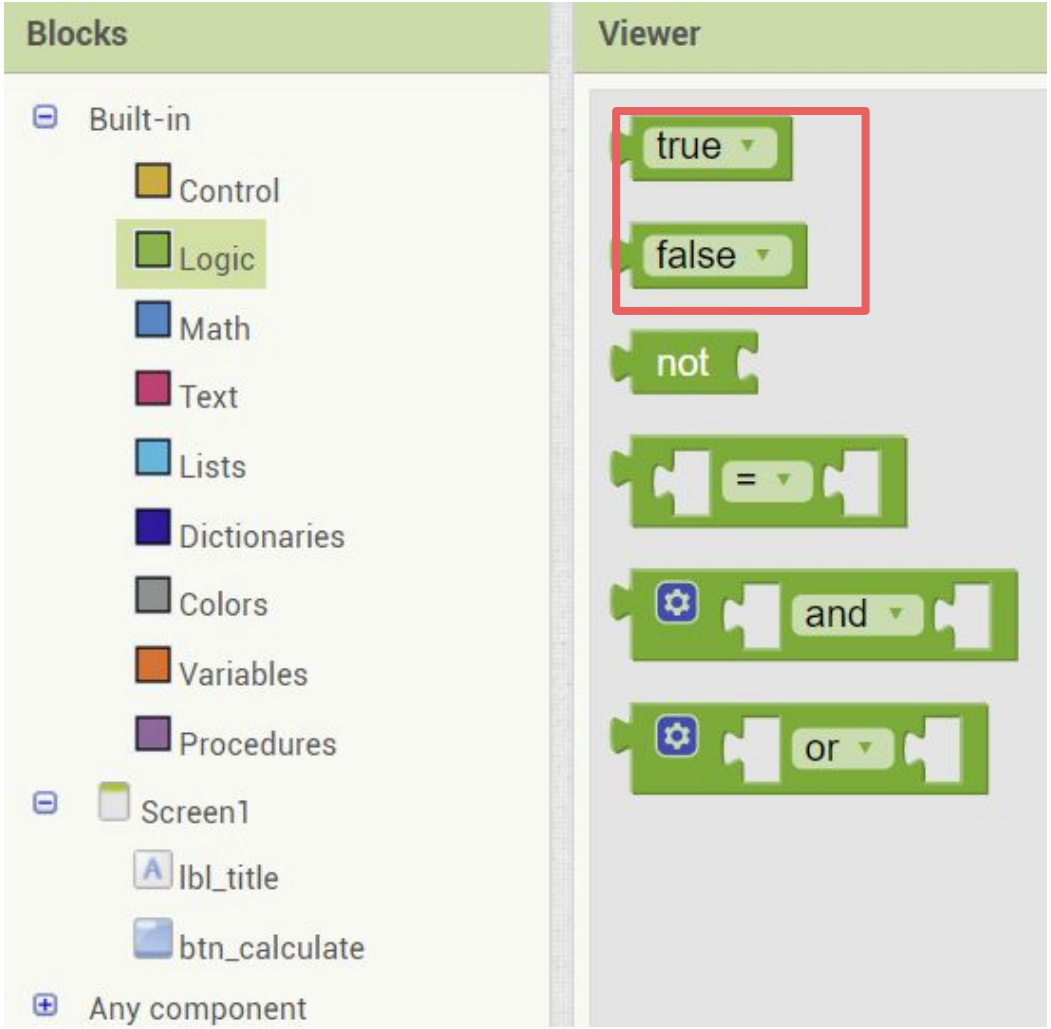


Basic operations

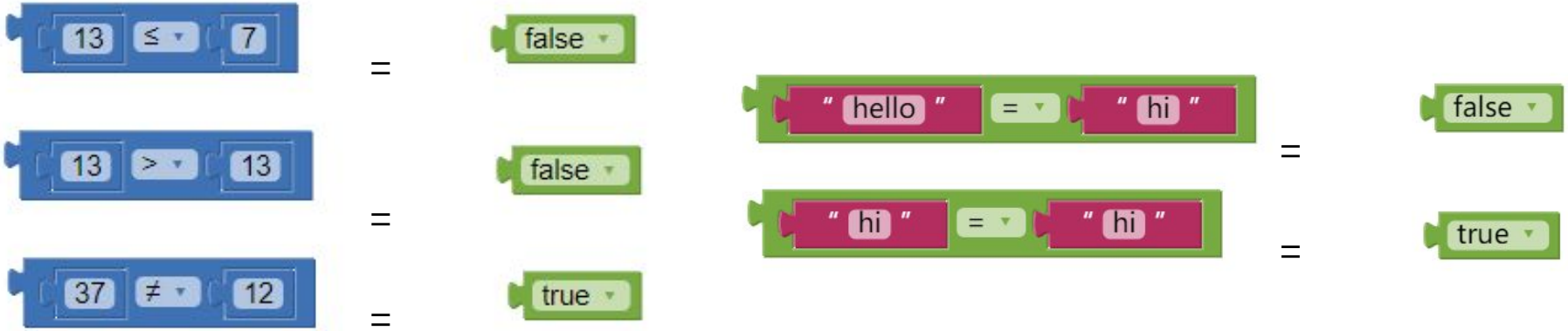


Logic (Booleans)

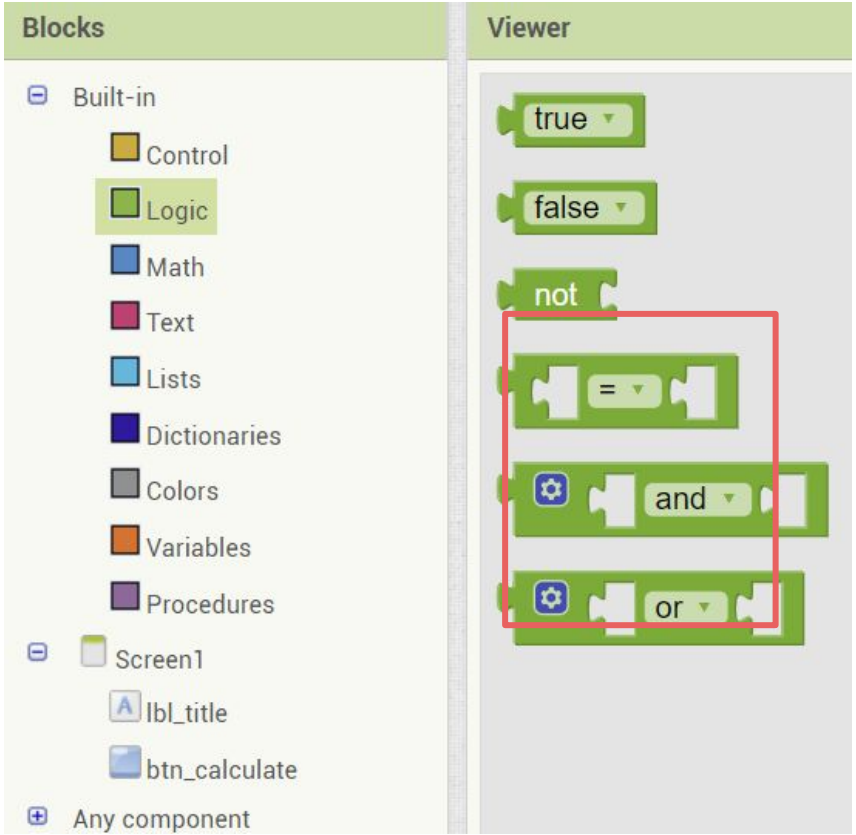
- values: true, false



Boolean Examples (Comparisons)



Logic Operators

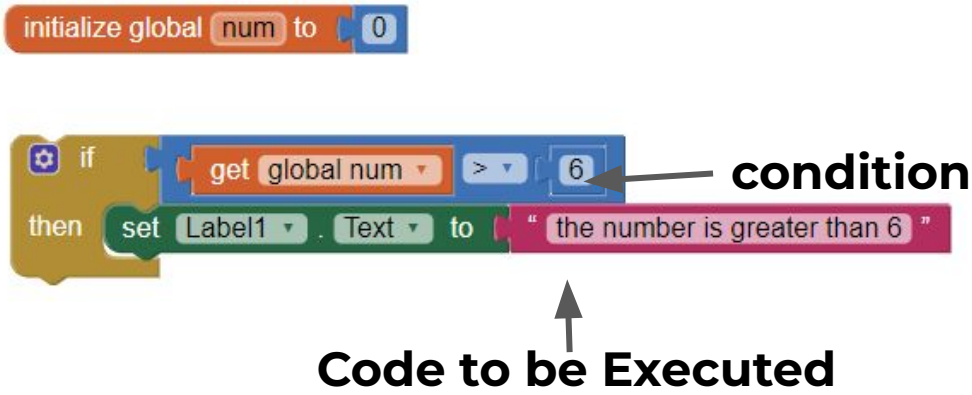


- **Not**: reverses the value
  - Not (true) => false
- **And**: considers both inputs
  - T and T ⇒ T
  - T and F ⇒ F
- **Or**: if one input is true, returns true
  - T or T ⇒ T
  - T or F ⇒ T
  - F or F ⇒ F

Conditional Statements

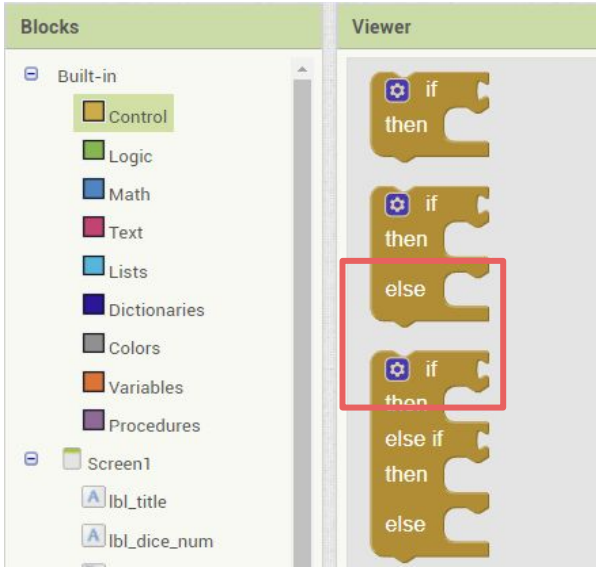
- **If statements** and **For loops**

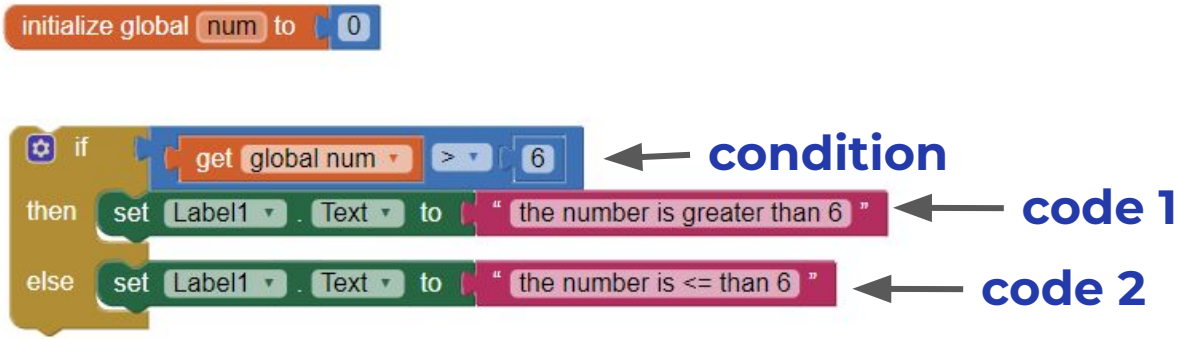
If Statements



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

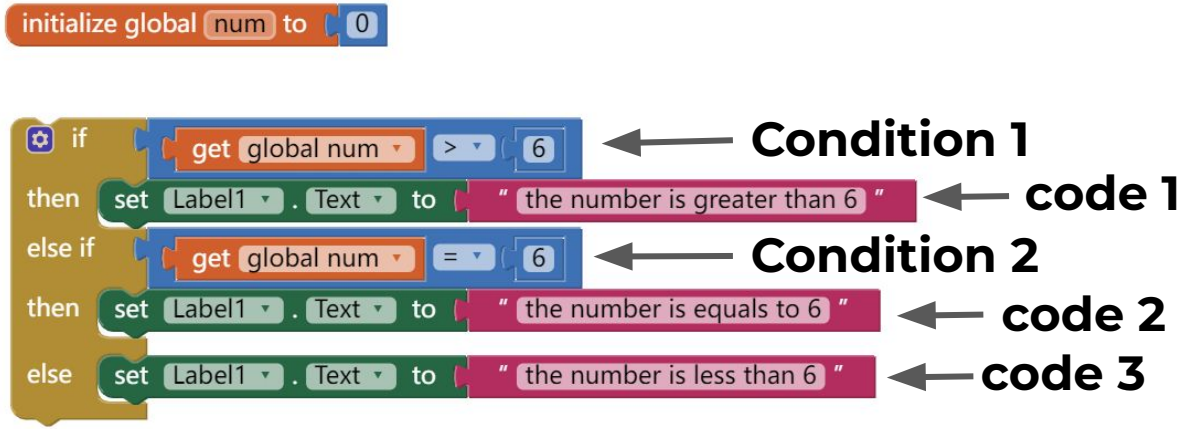
If-Else Statements





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

If-Else If- Else Statements



- **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

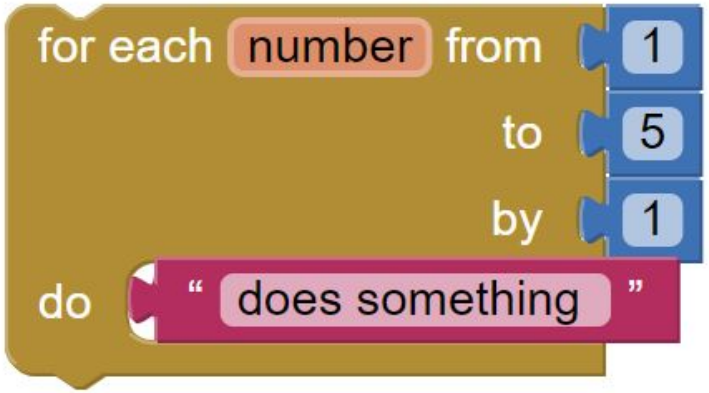
- If Age over 18, you are an adult
- Else if Age under 13, you are a child
- Else you are a teenager

- 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



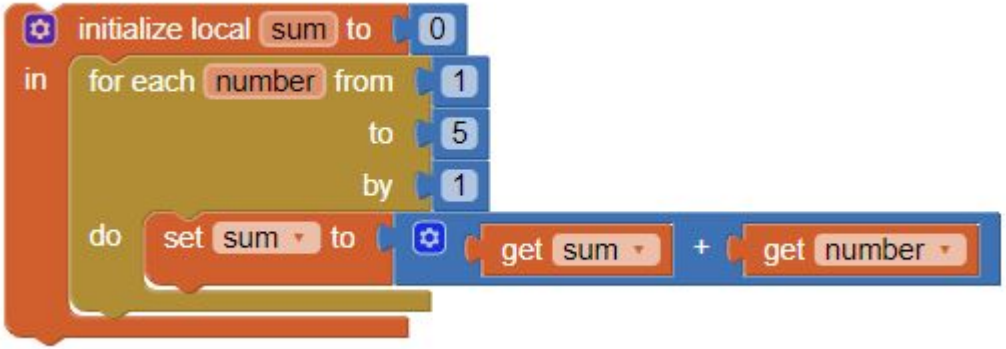
starting value

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+....+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

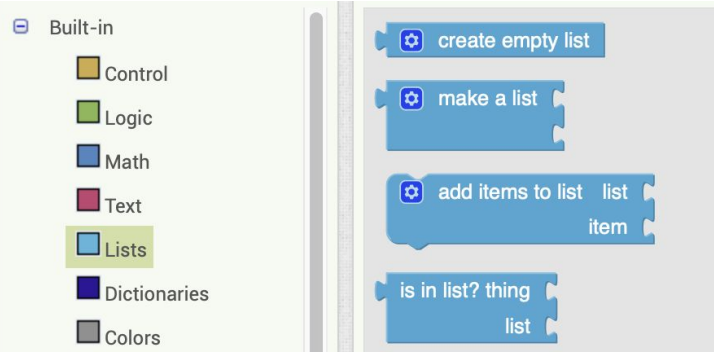
For each item in cart
- Add [item price] to total price paid

For each friend in class
- Generate a custom gift box

For each item on shelf
- +1 to inventory count

Lists

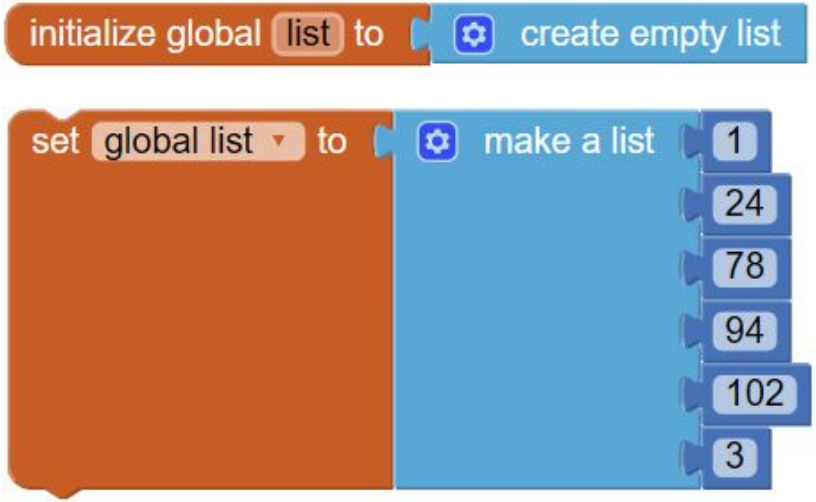
- **are a data structure**
- Items in a list are separated by commas and have indexes (position in list)
  - In App Inventor, the first element in a list is index 1
    - Note: in other programming languages it is sometimes 0
  - Useful when addressing specific element in a list



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

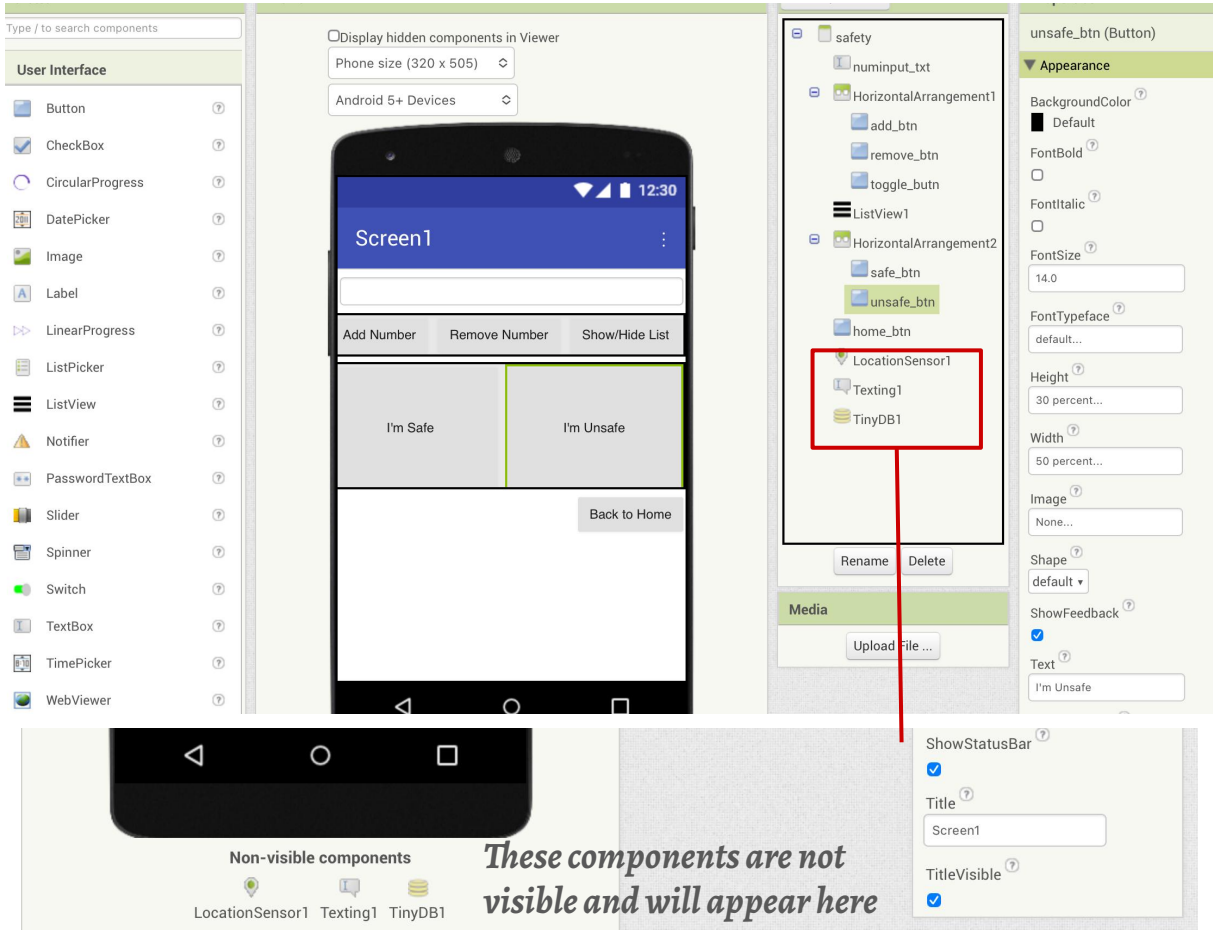
https://appinventor.mit.edu/explore/ai-with-mit-app-inventor

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



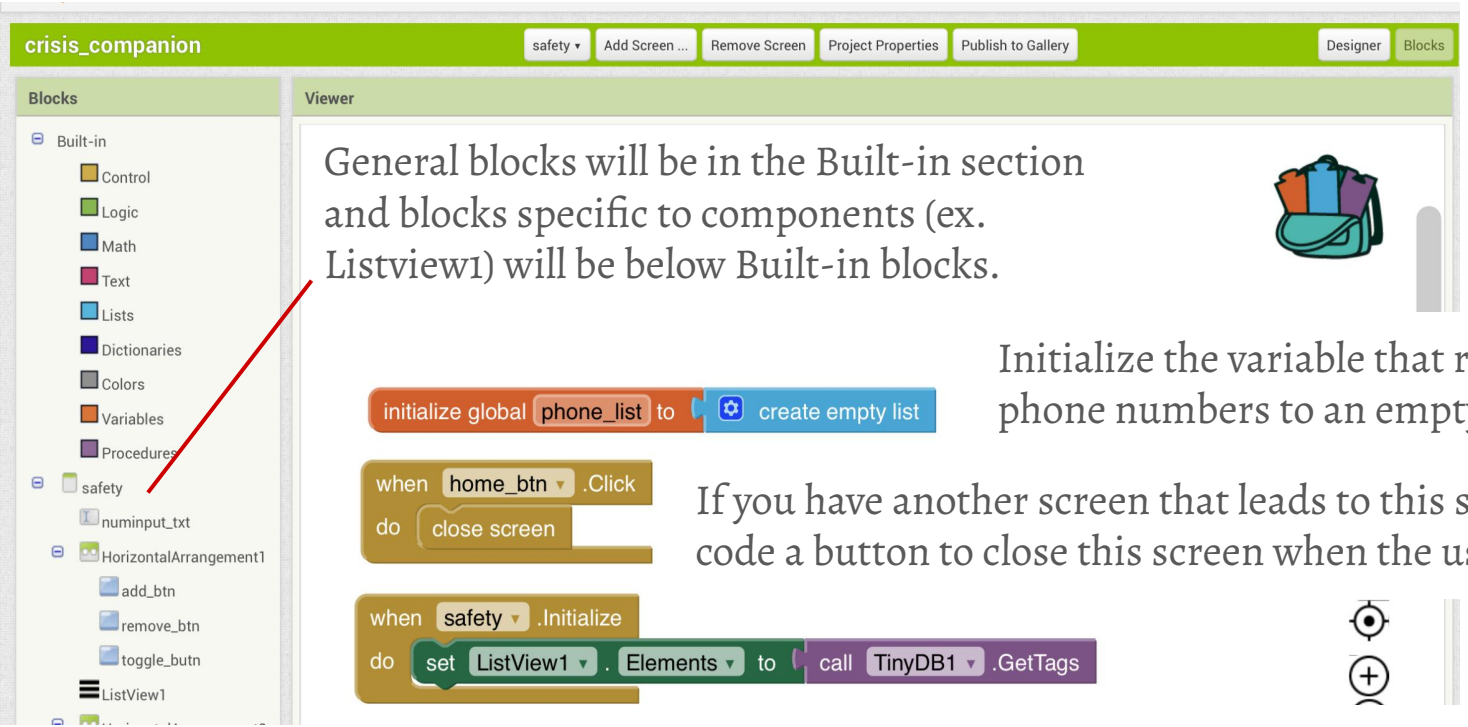
Add components from the palette on the left by dragging and dropping into the viewer

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

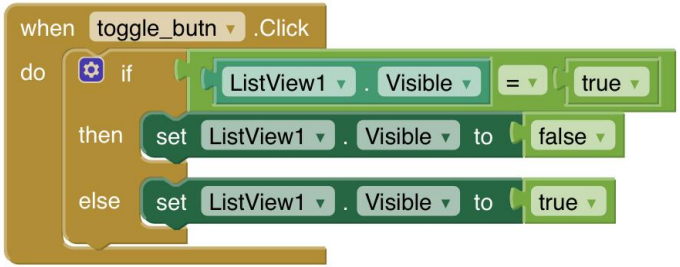


General blocks will be in the Built-in section and blocks specific to components (ex. Listview1) will be below Built-in blocks.

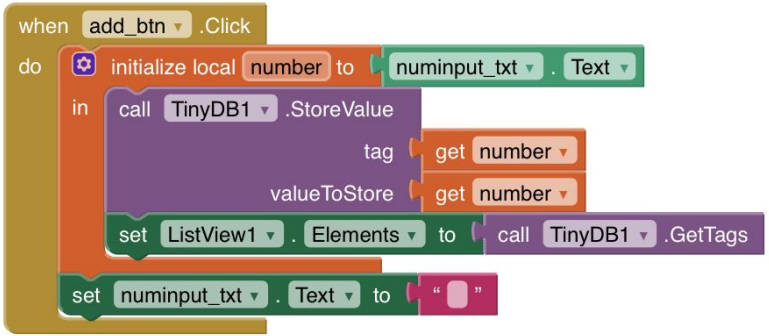
Initialize the variable that represents the list of phone numbers to an empty list

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

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.

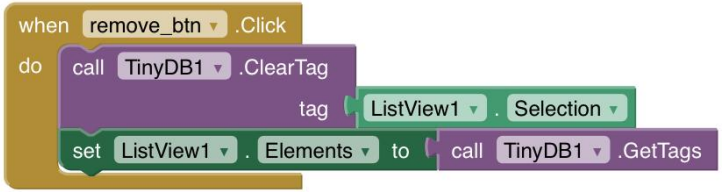


Sometimes having the list in sight is cumbersome. Maybe we can allow the user to show and hide it with these logic statements.



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



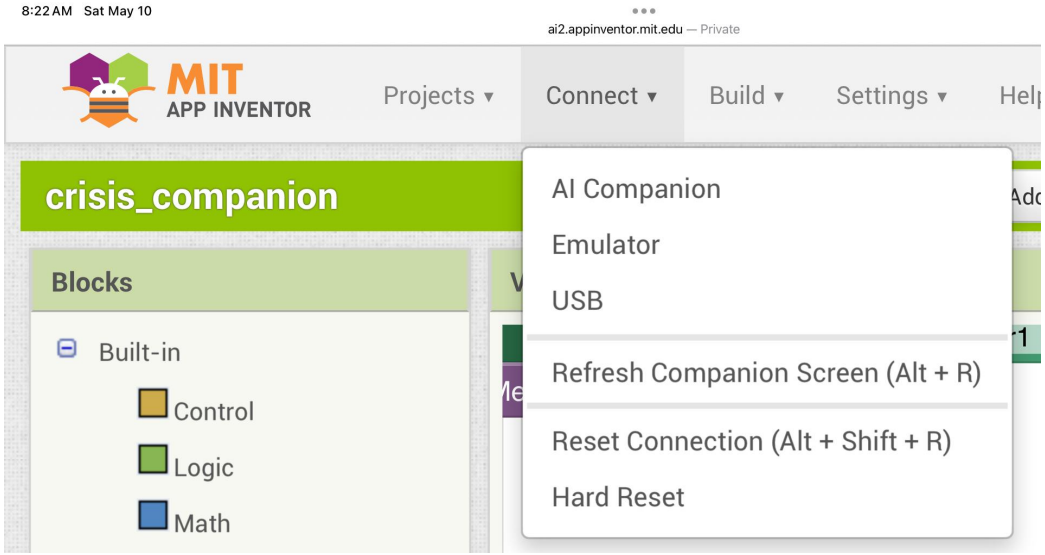
When the remove button is clicked, the tag corresponding to the phone number is cleared from TinyDB and the ListView elements are reset to reflect that change.

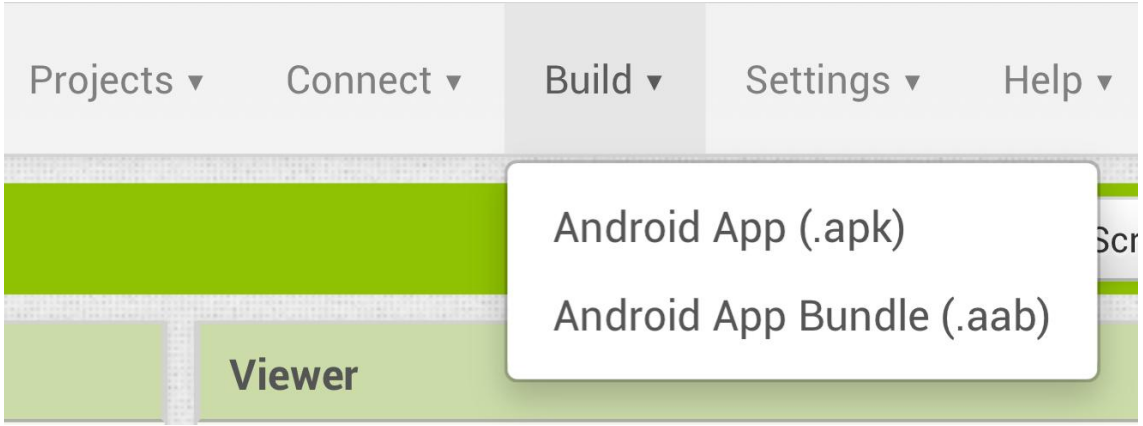
When the user clicks the button saying that they are safe, we set our global variable, the list of phones, to the numbers stored in TinyDB. Then, for each phone number in the list, we call the texting component to send our address *from the Location Sensor) directly to the contact.

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.





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?)