



# MIT App Inventor

**MIT**  
APP INVENTOR

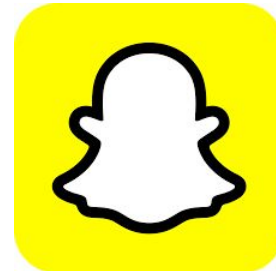
**“Your mobile device has quickly become the easiest portal into your digital self.”**  
– Phil Nickinson, Editor of Android Central



# Why App Development?

Think about different apps that you use everyday...

- Easy Access
- Notifications
- We all use mobile phones everyday
- Improves user engagement
- Many more



## MIT App Inventor - great for beginners!

- To create mobile apps for the android devices
- Works on android phones and tablets
- Drag-and-drop coding environment (blocks)
- You can actually publish the mobile app you created

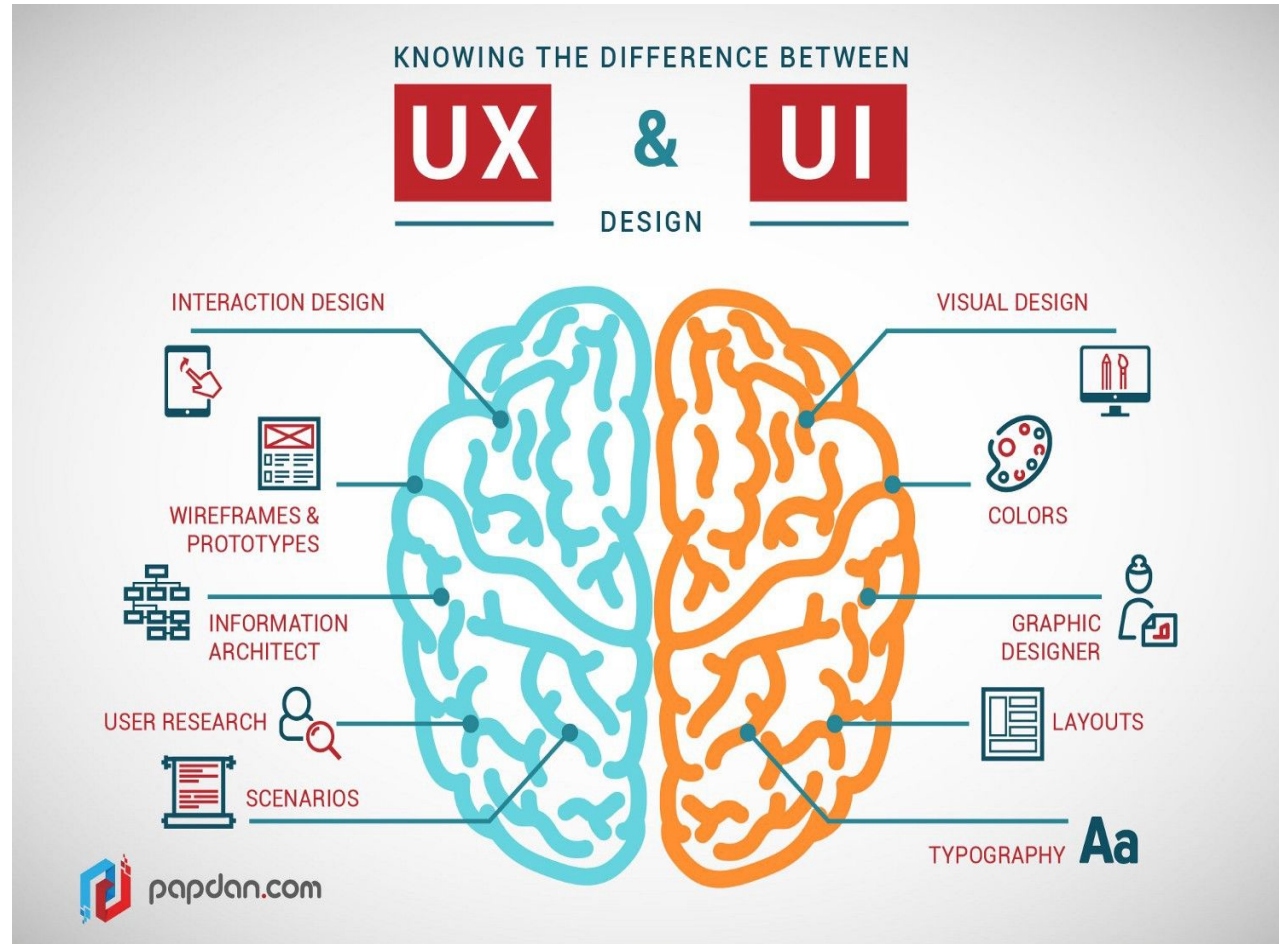




# UI & UX

**UI: User Interface**

**UX: User Experience**

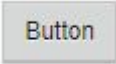



# UI Designs of App Inventor

---



## Screen components

- Buttons   
: components that users touch to perform some action in your app
- Labels  
: components used to show text, display text specified by the text property
- Textbox   
: users enter a text in a Textbox



## Screen components 2

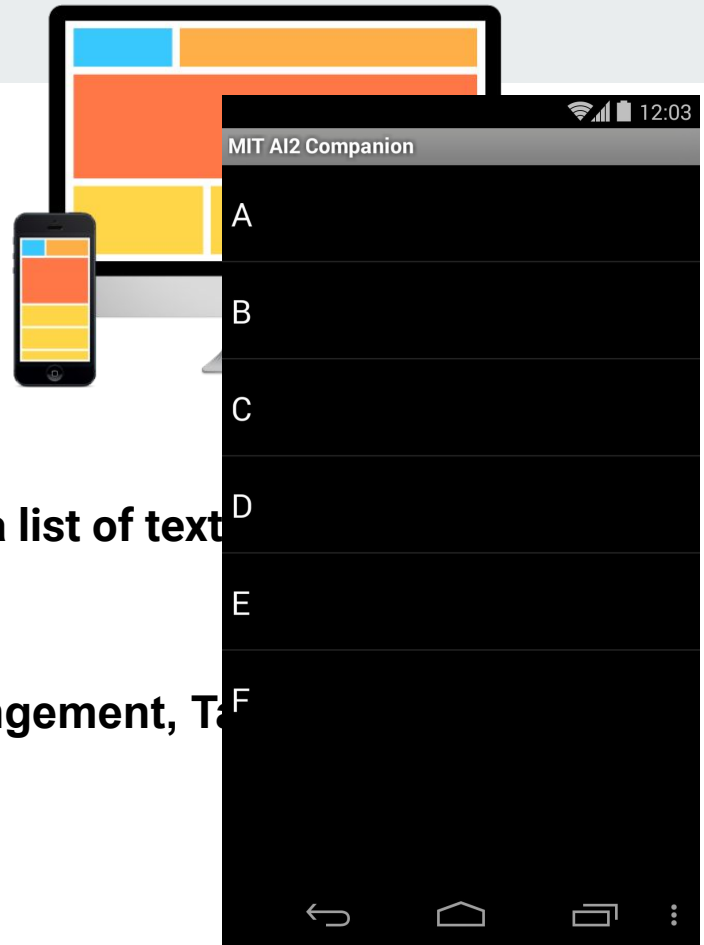
- Listpicker



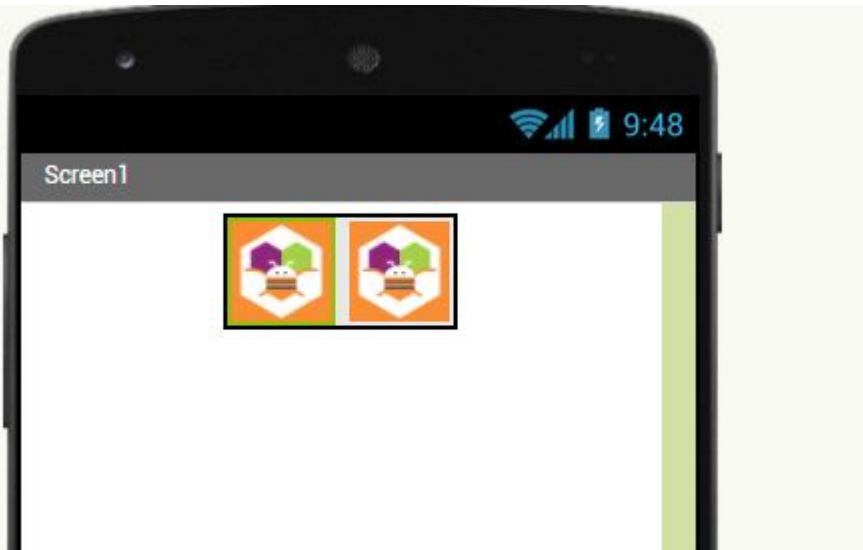
: a button that, when clicked on, displays a list of text to choose among

- Screen layouts

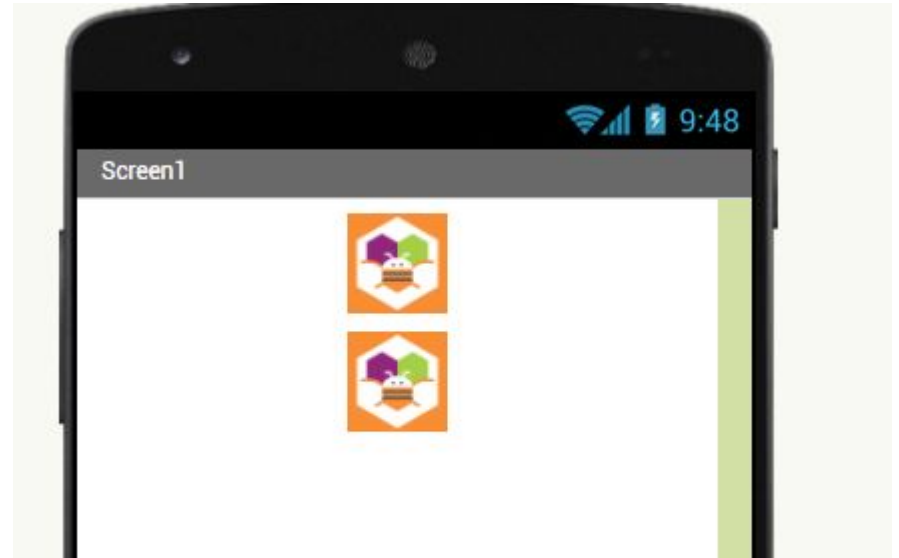
: **Horizontal Arrangement, Vertical Arrangement, Tabbed Arrangement, Scroll Arrangement**



With arrangement



Without arrangement





Horizontal  
Arrangement

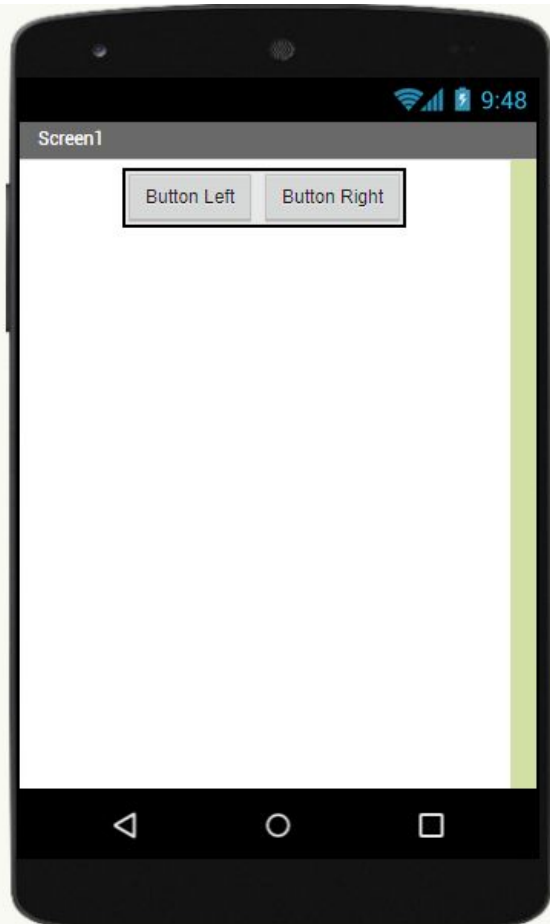
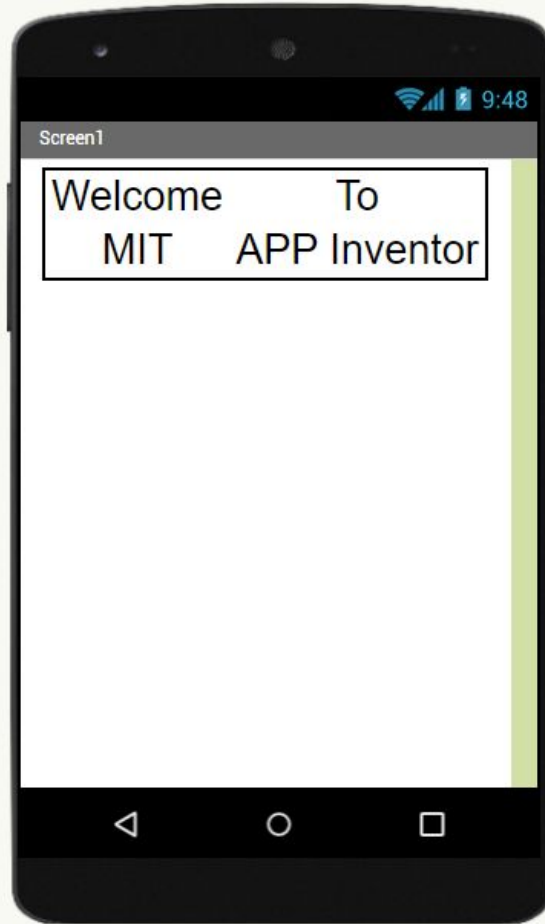
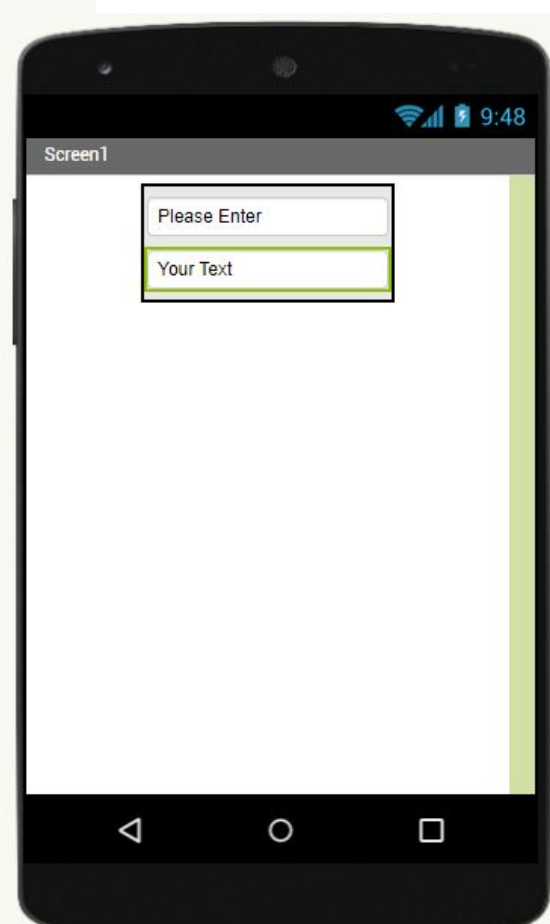


Table  
Arrangement



Vertical  
Arrangement






# Programming with App Inventor

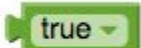






---



## Text

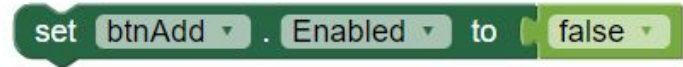
- **String** - a text element 
- **Join** - appends all of the items to make a single string 
- **Uppercase** - convert the string into all upper case 

## Logical Operators

- **True** 
- **False** 
- **Not** - returns false if the input is true, and returns true if the input is false 
- **Equal** - two numbers, text boxes, or lists are same 
- **Unequal** - two arguments are not equal 
- **And** - tests whether both conditions are true 
- **Or** - tests if one of the conditions is true 

## Setters

- To set the component to a certain value
- For example...
  - To disable a button
  - To clear the textbox



set btnAdd . Enabled to false



set txtItem . Text to " " (empty string)



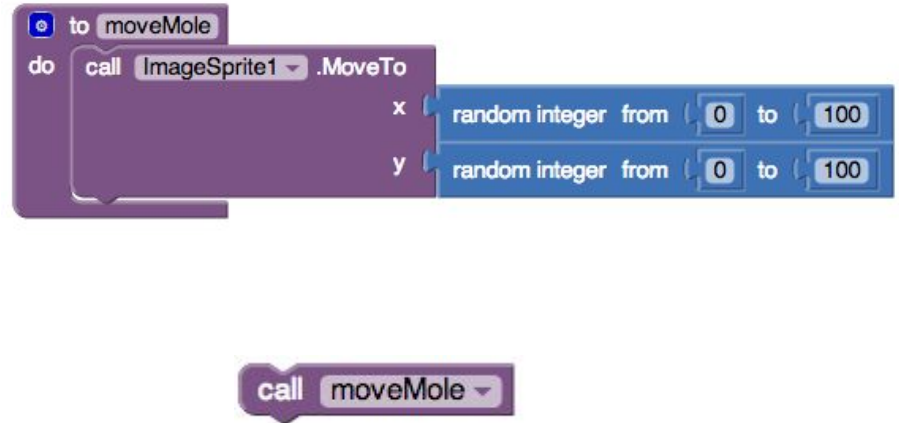
## If statement

- Necessary for **decision making** process
- **If-then:** performs actions if conditions are true. Otherwise, it ignores them.
- **If-then-else:** performs actions if conditions are true. Otherwise, it performs the actions of the “else”
- **If-then-else if-else:** performs actions if conditions are true. If “else if” conditions are met, it performs its actions. Otherwise, it performs the actions of the “else”



# Procedures

- A procedure is **a set of instructions that perform a specific task or tasks.**
- Codes can become repetitive and too long
- You can only set procedure once, and call it multiple times to avoid repetitive code blocks!



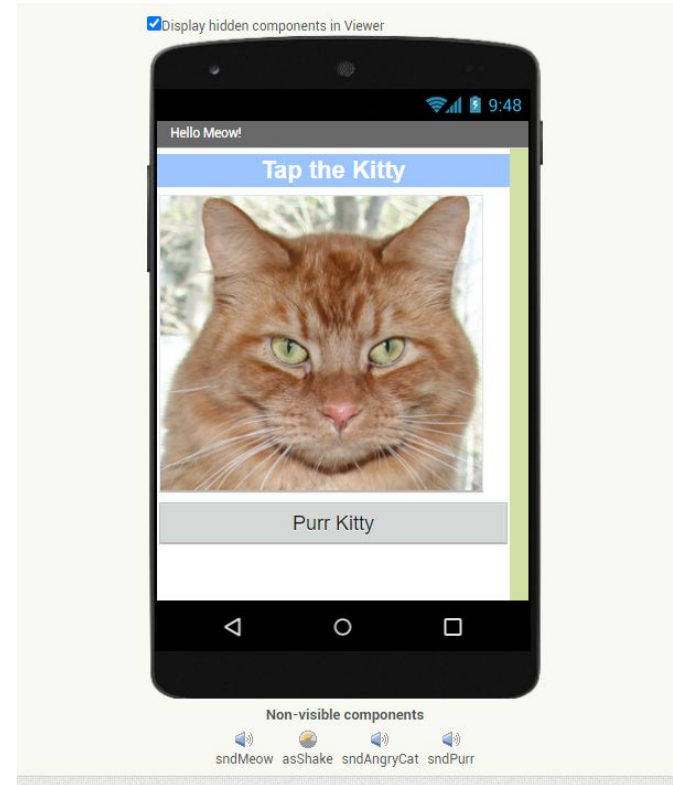
# Project 1: HelloMeow App

---



## Hello Meow App - Preview

1. Projects ☐ Import project (aia.) from my computer ☐ Choose the aia file
2. Go back to the Designer page and click Upload File for Media
3. Upload purr.mp3



# Hello Meow App - Let's observe

1.

```
when Screen1.Initialize
do
  set sndMeow.Source to "meow.mp3"
  set sndAngryCat.Source to "Angry-cat.mp3"
  set sndPurr.Source to "purr.mp3"
```

2.

```
when btnCat.Click
do
  call sndMeow.Play
  call sndMeow.Vibrate
  milliseconds 500
  set btnPurr.Visible to true
```

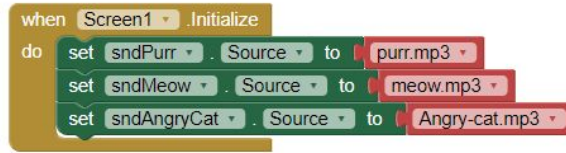
```
when asShake.Shaking
do
  call sndPurr.Stop
  call sndAngryCat.Play
```

```
when btnPurr.Click
do
  call sndPurr.Play
```



## Hello Meow App - UI design

1. Change Screen to: alignment Horizontal - center
2. Add a button
  - Set its text to: “Purr Kitty”
  - Name it: BtnPurr
  - Width - fill parent
3. Import the purr sound
  - Drag in a Sound component
  - Rename it to sndPurr



## Hello Meow App - time to code

- When Screen 1 initializes...set sndPurr's source to purr.mp3
- Play sound when button is pressed: When btnPurr is clicked...call sndpurr to play
- We don't want our cat to purr when we are petting them: When btnCat is clicked...call sndPurr to stop, call sndMeow to play, call sndMeow to vibrate for 500 milisecs, and set btnPurr to visible



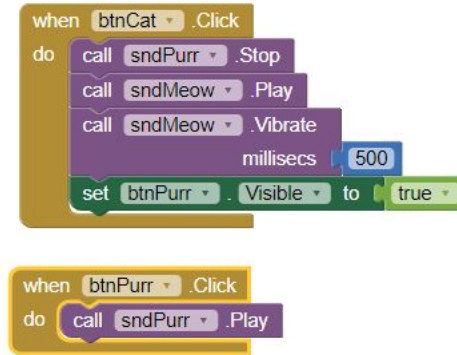
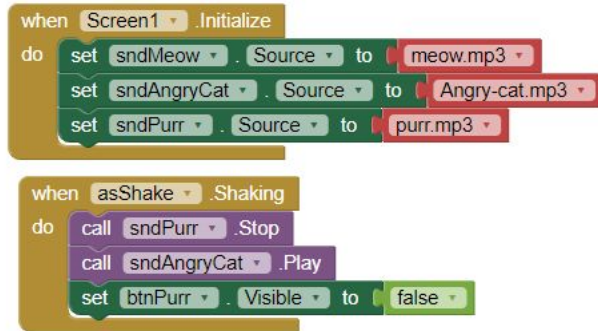
## Hello Meow App - time to code

- When asShake is shaking, call sndPurr to stop, and call sndAngryCat to play
- Set btnPurr to invisible



# Hello Meow App








Preview: Connect - AI companion - scan the QR code or enter the word code



# Project 2: explore different features of App Inventor






---

## Math Blocks

- Number 
- Unequal 
- Equal 
- $>$  
- $\geq$  
- $<$  
- $\leq$  



## Math Blocks

- Addition (+) 
- Subtraction (-) 
- Multiplication (\*) 
- Division(/) 
- Picks the random integer from the given range of numbers 

# Variables

- Variables are **data values that can change when the user is asked a question**
- We can set our variables to numbers, words, lists, and many more!
- **Initialize global name to** - creates a new global variable by setting it to a certain value
- **Get** - gets created variables
- **Set to** - gives a created variable a new value
- **Initialize local name to** - creates a new local variable that only works in a certain procedure





## Concept Cards

- Speech recognition/text to speech
- Make a video
- Take a picture
- Bounce Spire/Ball off Edge
- Movement on a Timer
- Fling movement
- Drawing
- Creating your own color
- Random numbers
- Drag a sprite
- Collision detection

---

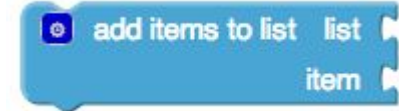
# Questions?

# Advanced Programming with App Inventor

---

# Lists

- A list of values
- Used to avoid the repetitive use of elements
- Lists can simply organize all the items or components under one list
- **Index** - indicates a certain item of a list
- **Create empty list** - creates a new empty list
- **Make a list** - creates a list which you can add elements
- **Add items to list** - appends a new item to the created list
- **Select list item** - selects the item at the given index

A blue Scratch block with a gear icon on the left and the text "create empty list".A blue Scratch block with a gear icon on the left and the text "make a list".A blue Scratch block with a gear icon on the left, the text "add items to list", a dropdown menu showing "list", and an input field with the text "item".A blue Scratch block with the text "select list item", a dropdown menu showing "list", and an input field with the text "index".



# Lists

- **Length of list** - returns the number of items in the list
- **Is list empty?** - true if there are no items in the list
- **Index in list**- selects the item at the given index



length of list list



is list empty? list



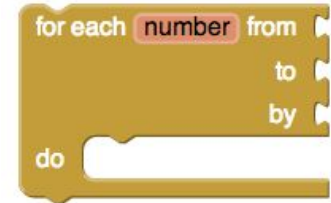
index in list thing

list

# Loops

**: A sequence of instructions that is continually repeated until a certain condition is reached**

- Runs the blocks in the do section for each number in the range starting from *from* and ending at *to*. Use the given variable name, **number**, to refer to the current value. You can change the name **number** to something else if you wish.
- Runs the blocks in the do section for each item in the list. Use the given variable name, **item**, to refer to the current list item. You can change the name **item** to something else if you wish.
- Tests the condition. If true, performs the action given in -do , then tests again. When test is false, the block ends and the action given in -do is no longer performed. It is the infinite loop





# Storage

- Stores the data, which enables the app to restore the data when the user opens an app for the second time
- Call TinyDB1 to store value
- Set “Tag” to a string and set “valueToStore” to a global variable that must be stored
- When the screen initializes, make sure that the stored global variable can restore its value with same tag. Set a new value for “valueIfTagNotThere”



# Advanced projects to try

---



## Note Taker APP

- You can type your text into the textbox. This text will be added to the notes when you click the button “Add Note”
- “Remove Note” should enable you to select a note that you want to remove
- “Clear Note” should clear all existing notes
- All data should be stored

initialize global notes to create empty list

```
when Screen1.Initialize
do
  set global notes to call TinyDB1.GetValue
  tag "notes"
  valueIfTagNotThere create empty list
  if is list empty? list get global notes
  then set lstRemoveButton.Enabled to false
  else call outputList
```

```
to outputList
do
  set lblNotesOutput.Text to join items using separator "\n"
  list get global notes
  call TinyDB1.StoreValue
  tag "notes"
  valueToStore get global notes
```

```
when lstRemoveButton.AfterPicking
do
  remove list item list get global notes
  index lstRemoveButton.SelectionIndex
  call outputList
  if is list empty? list get global notes
  then set lstRemoveButton.Enabled to false
```

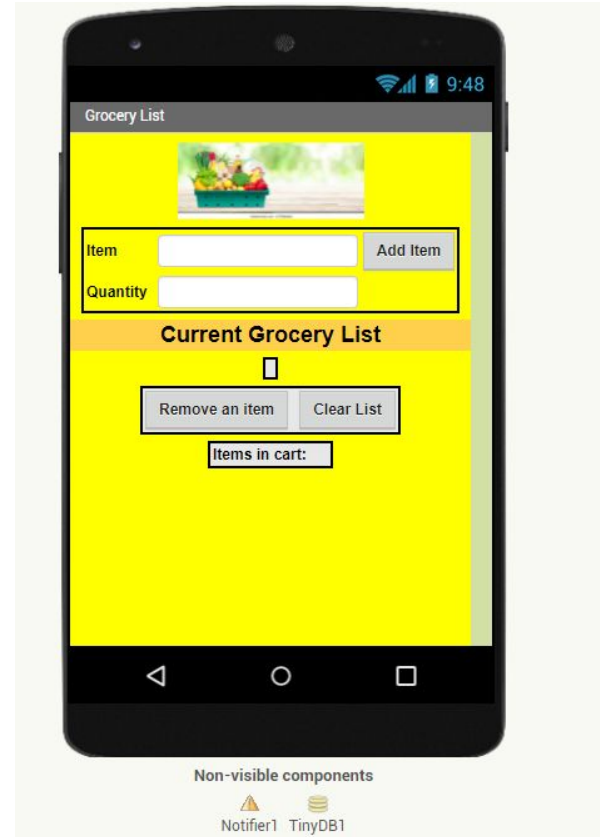
```
when btnAddNote.Click
do
  if not is empty txtEntry.Text
  then
    add items to list list get global notes
    item txtEntry.Text
    set txtEntry.Text to ""
    set lstRemoveButton.Enabled to true
    call outputList
```

```
when lstRemoveButton.BeforePicking
do
  set lstRemoveButton.Elements to get global notes
```

```
when btnClearAll.Click
do
  set global notes to create empty list
  set lstRemoveButton.Enabled to false
  call outputList
```

# Grocery List App!

- The app should enable the users to enter the name and the quantity of the item
- The app should notify the user when the click “Add Item” if the textbox is empty or if the user entered text as the quantity
- “Items in cart” should show the sum of all quantities
- “Remove an item” and “clear list” should properly function
- All data must be stored.



initialize global items to create empty list

initialize global quantity to create empty list

initialize global total to 0

```
when btnAdd.Click
do
  if is empty txtItem.Text
  then
    call Notifier1.ShowAlert
    notice Add items
  else if is empty txtQuantity.Text
  then
    call Notifier1.ShowAlert
    notice Add quantity
  else if not is number? txtQuantity.Text
  then
    call Notifier1.ShowAlert
    notice Enter a number
  else
    add items to list list get global items
    item txtItem.Text
    add items to list list get global quantity
    item txtQuantity.Text
    set global total to get global total + txtQuantity.Text
    call outputList
    set txtItem.Text to
    set txtQuantity.Text to
```

```
when btnRemove.BeforePicking
do
  set btnRemove.Elements to get global items
```

```
when btnClear.Click
do
  set global items to create empty list
  set global quantity to create empty list
  set global total to 0
  call outputList
  if is list empty? list get global items
  then
    set btnRemove.Enabled to false
```

```
when btnRemove.AfterPicking
do
  remove list item list get global items
  index btnRemove.SelectionIndex
  remove list item list get global quantity
  index btnRemove.SelectionIndex
  set global total to 0
  for each item in list get global quantity
  do
    set global total to get global total + get item
  call outputList
  if is list empty? list get global items
  then
    set btnRemove.Enabled to false
```

```
when Screen1.Initialize
do
  set global items to call TinyDB1.GetValue
  tag Items
  valueIfTagNotThere create empty list
  set global quantity to call TinyDB1.GetValue
  tag Quantity
  valueIfTagNotThere create empty list
  set global total to call TinyDB1.GetValue
  tag Total
  valueIfTagNotThere create empty list
  call outputList
  if is list empty? list get global items
  then
    call outputList
    set btnRemove.Enabled to false
```

```
to outputList
do
  set lblTotal.Text to get global total
  set btnRemove.Enabled to true
  set lblOutput.Text to
  for each index from 1
  to length of list list get global items
  by 1
  do
    set lblOutput.Text to join
    select list item list get global items
    index get index
    * *
    select list item list get global quantity
    index get index
    * *
  call TinyDB1.StoreValue
  tag Items
  valueToStore get global items
  call TinyDB1.StoreValue
  tag Quantity
  valueToStore get global quantity
  call TinyDB1.StoreValue
  tag Total
  valueToStore get global total
```

---

# Questions?

# Thank You

More resources can be found on next slide

Happy programming with MIT App Inventor!







## Resources

- App Inventor Website: <http://ai2.appinventor.mit.edu/>
- Concept Cards Link:  
<https://appinventor.mit.edu/explore/resources/beginner-app-inventor-concept-cards>
- How to connect your phone or tablet:  
<https://appinventor.mit.edu/explore/ai2/setup-device-wifi.html>
- Control Blocks: <http://ai2.appinventor.mit.edu/reference/blocks/control.html>
- Documentation: <https://appinventor.mit.edu/explore/library>
- Other beginner-friendly projects:  
<https://appinventor.mit.edu/explore/hour-of-code>