# Tour Guide

# 1. Synopsis

In this unit, students will learn to use the Map component in App Inventor to create a tour guide app that showcases sites of interest in their community. Students will choose four landmarks in their community and create an app that can be used by visitors to locate and learn about those landmarks. Students will also use the TinyDB component, learning how to store data persistently in an app.

# 2. Learning Objectives

After completing this unit, students will be able to:

1. Design and code a location-based app, using the Maps component in App Inventor.
2. Create an app that uses multiple screens and passes values.
3. Demonstrate understanding of lists and indexes, utilizing multiple lists to manage data.
4. Use the TinyDB component to store data persistently on a device.

# 3. Mapping with the CSTA Standards

This table show the alignment of this unit with the intended learning outcomes to the CSTA CS Standards. The entries in the tables indicate the expected relevance of the unit to each outcome:

| | | |
|---|---|---|
| 2-DA-07 | Represent data using multiple encoding schemes.<br>[C] DA: Storage [P] Abstraction (all) | Persistent data used in TinyDB and locally in variables. |
| 2-DA-08 | Collect data using computational tools and transform the data to make it more useful and reliable.<br>[C] DA: Collection; Visualization & Transformation [P] Testing (6.3) | Landmark information (text and images) are included in a user-friendly app. |
| 2-AP-11 | Create clearly named variables that represent different data types and perform operations on their values.<br>[C] AP: Variables [P] Creating (5.1, 5.2) | Parallel lists are used to contain landmark information. |
| 2-AP-13 | Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.<br>[C] AP: Modularity [P] Computational Problems (3.2) | Three screens used to modularize parts of an app. |
| 2-AP-17 | Incorporate existing code, media, and libraries into original programs, and give attribution.<br>[C] AP: Program Development [P] Abstraction (4.2), Creating (5.2), Communicating (7.3) | Template is used with some user interface components included. |
| 2-AP-18 | Systematically test and refine programs using a range of test cases.<br>[C] AP: Program Development [P] Testing (6.1) | Testing and debugging done at all 3 stages of development. |

# 4. Learning Prerequisites

Students should have experience with the MIT App Inventor environment, be able to add UI components in the Designer, code an app, and test using MIT AI2 Companion or the emulator.

# 5. Lesson Plan ( 45 minutes x 5)

### Lesson 1

| Time | Activity |
|---|---|
| 10 min | **Introduction**<br>1. Ask students:<br>a. Have you used any maps-based apps?<br>b. What did you use them for?<br>c. Have you used an app to help you find out about places to see while you were travelling?<br>2. Explain to students that they will create an app that will highlight 4 different places in their community. If they prefer, they can choose a city/place they would like to visit.<br>3. Remind students they will work with a partner, using the Pair Programming model.<br>4. Demonstrate an example of the finished app (BostonTourGuide_complete.aia). |
| 30 min | **Geographic Coordinate System**<br>1. Explain latitude and longitude.<br>2. Demonstrate with www.latlong.net.<br>3. Have student groups fill out the Landmarks Worksheet with their partner. They should fill in title, description, and an image filename. The should also include the latitude and longitude for each landmark, using www.latlong.net. |

| | |
|---|---|
| 5 min | **Wrap-up**<br>Check with student groups to make sure they have all the necessary information for their chosen locations. |
| Homework | Complete Landmarks Worksheet. |

## Lesson 2

| **Time** | **Activity** |
|---|---|
| 10 min | **Maps and Markers Demonstration**<br>1. Check in with student groups to make sure they have their Landmarks worksheets completed.<br>2. Teacher demonstrates the Map and Marker components. Students will follow along with teacher at the board to add the components needed.<br>a. Show students how to add a Map component.<br>b. Demonstrate how to add Marker components.<br>c. Show properties of Map and Marker components. |
| 5 min | **Introduction to Using Multiple Screens**<br>1. Explain the use of multiple screens in this app.<br>2. Explain the **open another screen with start value block** as a way to send information to LocationScreen in the app. |
| 25 min | **Coding Screen1**<br>Students will follow the ***Tour Guide Student Guide: Screen1*** to complete adding all UI information in the Designer, and coding the Screen1 blocks. |
| 5 min | **Wrap-up**<br>1. Check with student groups to make sure they have completed the Screen1 portion of their app. |
| Homework | 1. (Required) Complete Screen1 of app.<br>2. (Optional) Continue with the Location Screen. |

## Lesson 3

| Time | Activity |
|---|---|
| 20 min | **Introduction to Multiple Screens and Lists**<br>1. Ensure students have completed all tasks from Lesson 1.<br>2. Explain new concepts and components covered in this lesson:<br>a. Multiple screens and passing values between screens<br>b. Lists (Locations, Descriptions, Pictures in LocationScreen)<br>i. Present **either** one of the following unplugged activities:<br>i. Unplugged Activity 1: Indexing in Lists<br>ii. Unplugged Activity 2 (file cabinet drawers) |
| 20 min | **Coding of LocationScreen**<br>Ask students to follow the ***Tour Guide Student Guide: LocationScreen***.<br>a. Complete LocationScreen coding.<br>b. Test and debug LocationScreen |
| 5 min | **Wrap-up and Review**<br>1. Check in with student groups to ensure they have completed the code blocks and tested the app to make sure it works..<br>2. Review main concepts/components covered in the tutorial. |

## Lesson 4

| Time | Activity |
|---|---|
| 15 min | **Introduction to New Components and Concepts**<br>1. Ensure students have completed all tasks from Lessons 1-3.<br>2. Explain new concepts and components covered in this lesson:<br>a. Camera blocks<br>b. TinyDB (already used in FoodChase) |
| 25 min | **Coding of GalleryScreen with Single Image** |

| | |
|---|---|
| | Ask students to follow the ***Tour Guide Student Guide: Taking Pictures***.<br>a. Complete GalleryScreen coding to save a single image.<br>b. Test and debug GalleryScreen. |
| 5 min | **Wrap-up and Review**<br>Check in with students to see that they are making progress on the app. |

## Lesson 5

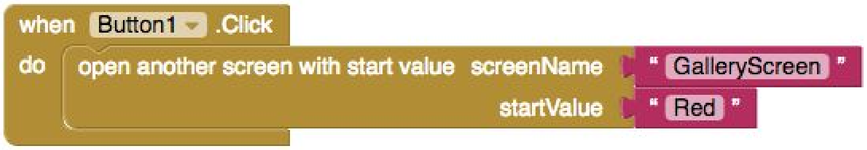
| Time | Activity |
|---|---|
| 10 min | **Review of Lists**<br>1. Ensure students have completed the GalleryScreen that allows a user to take a single picture and save it.<br>2. Explain to students they will add ability to save a list of images for a complete gallery and allows user to scroll through the images.<br>3. Do unplugged activity to demonstrate a list and how to index into a list to access the values. |
| 25 min | **Coding of Improved GalleryScreen**<br>Ask students to follow the ***Tour Guide Student Guide: Improved GalleryScreen***.<br>a. Complete GalleryScreen coding.<br>b. Test and debug GalleryScreen. |
| 10 min | **Wrap-up and Review**<br>1. Review lists, camera, TinyDB, Map components.<br>2. Ask students to complete multiple choice questions. |

# 6. Assessment

The following multiple choice questions is a check for understanding of key computational thinking concepts.

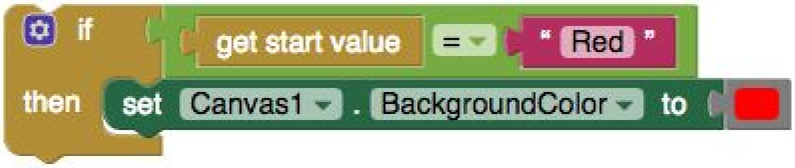
**Multiple Choice Questions**

1. A student makes an app with two screens, Screen1 and GalleryScreen. The student has the code in Screen1 to pass a start value to GalleryScreen.



```
when Button1 .Click
do  open another screen with start value  screenName  " GalleryScreen "
                                          startValue  " Red "
```

The student also has an if block in GalleryScreen to change the canvas color to red if the start value = "Red", but can't remember where to put the block. In which event block should it go?

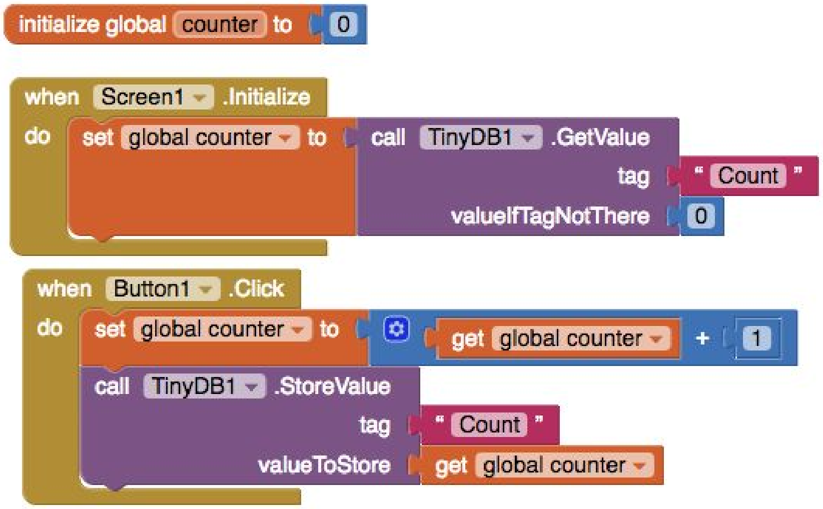


```
if    get start value  =  " Red "
then  set Canvas1 . BackgroundColor to 🟥
```

A. Button1.Click
B. Canvas1.Click
C. GalleryScreen.Click
D. GalleryScreen.Initialize

(Answer: D)

For Questions 2-3, use the following code blocks:



```
initialize global counter to 0

when Screen1 .Initialize
do  set global counter to  call TinyDB1 .GetValue
                                               tag  " Count "
                                valueIfTagNotThere  0

when Button1 .Click
do  set global counter to  get global counter + 1
    call TinyDB1 .StoreValue
                     tag  " Count "
            valueToStore  get global counter
```

2. When the app starts for the first time, what is the value of counter?

A. 0
B. 1
C. 2
D. 3

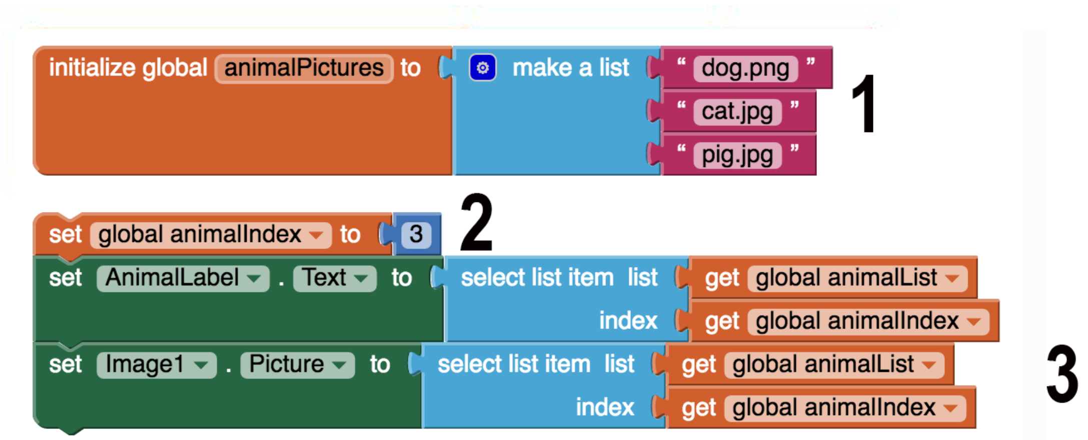
(Answer: A)

3. The user runs the app once, and clicks Button1 3 times, then closes the app. What is the value of counter when he opens the app again?

A. 0
B. 1
C. 2
D. 3

(Answer: D)

4. Anna uses the following blocks in her app. No picture appears. What is wrong?



```
initialize global animalPictures to  make a list  " dog.png "
                                                  " cat.jpg "
                                                  " pig.jpg "

set global animalIndex to 3
set AnimalLabel . Text to  select list item  list   get global animalList
                                             index  get global animalIndex
set Image1 . Picture to  select list item  list   get global animalList
                                           index  get global animalIndex
```

A. “pig.jpg” needs to be changed to “pig.png” in the block labelled 1.
B. “3” of animalIndex must be changed to “1” in the block labelled 2.
C. “animalList” must be changed to “animalPictures” in the block labelled 3.
D. A start value is needed.

(Answer: C)

**Survey of learning attitudes**

In order to evaluate students' attitude, perception, and understanding towards coding, students are required to finish a 5-point scale survey below by putting a "✓" in the appropriate box.

| After completion of this unit, I think… | Disagree | Somewhat disagree | Neutral | Somewhat agree | Agree |
|---|---|---|---|---|---|
| Learning how to make apps makes me want to learn more about coding. | | | | | |
| I feel more connected to the technology around me when I make apps. | | | | | |
| I am excited to share this app with friends and family. | | | | | |

**Self Assessment on Collaboration**

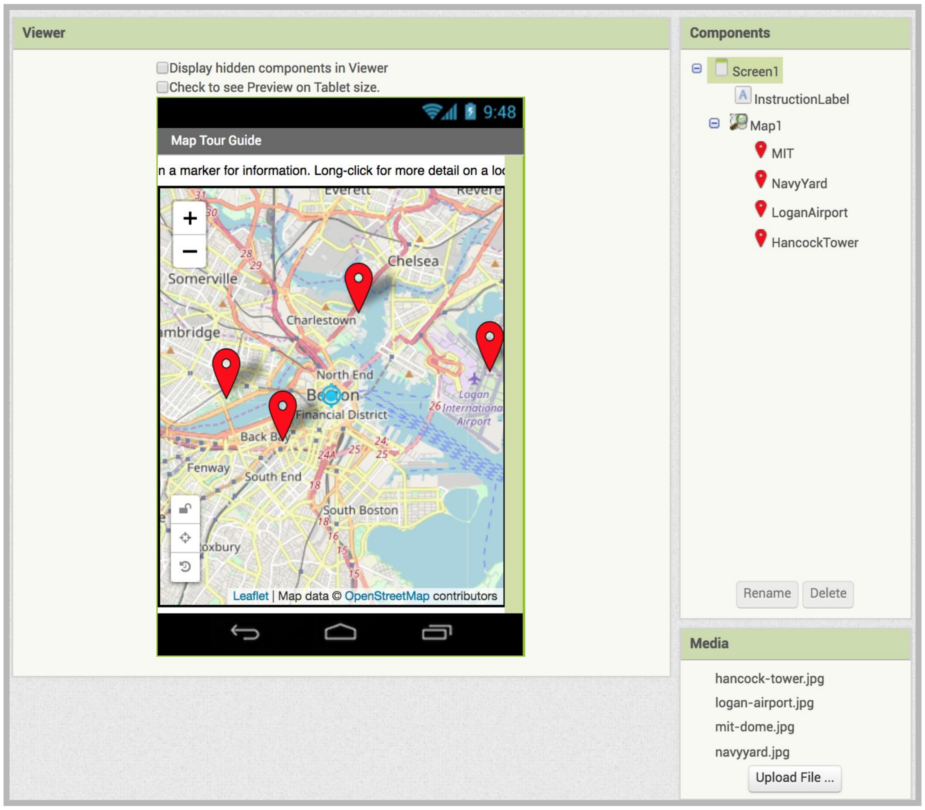
Ask students to reflect on how well they worked with their partner, and to answer the following questions honestly.

1. Did you like working with another person? Do you feel you were a good partner, and respected and encouraged your partner in the project?
2. What was your role as a partner in this project? What did you do and what did your partner do?

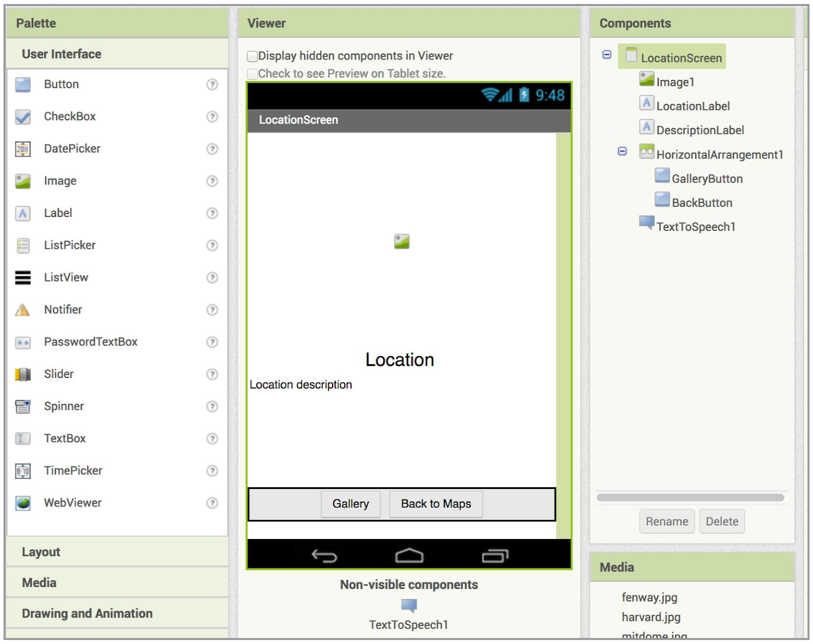
# 7. Screen Design and Code

### Designer

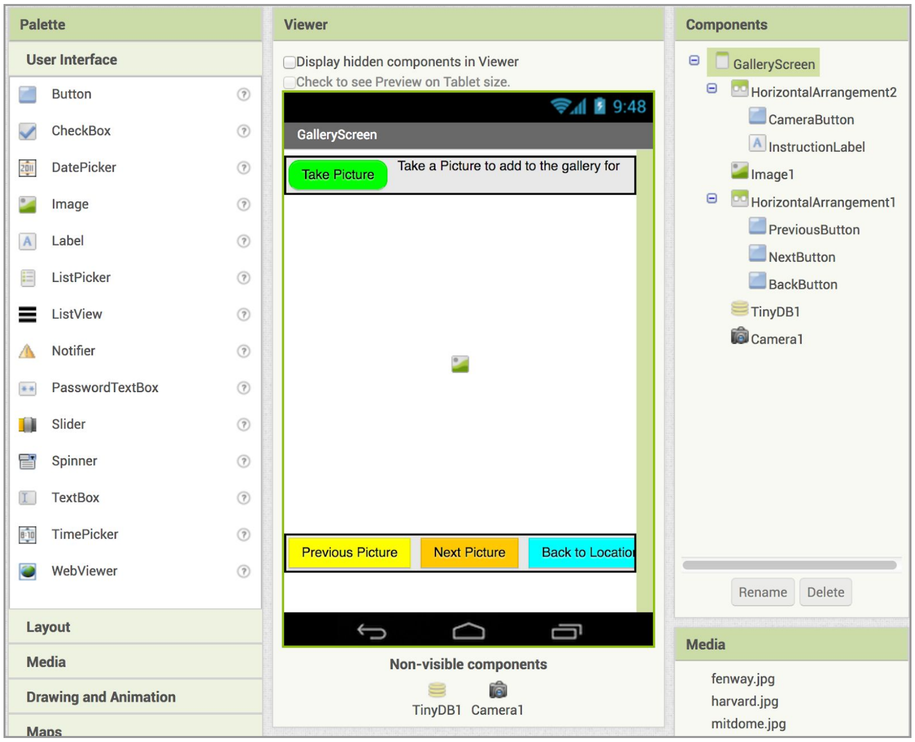
***Screen1*** (NOTE: This is an example screen for a Boston Tour Guide.)



### *LocationScreen*

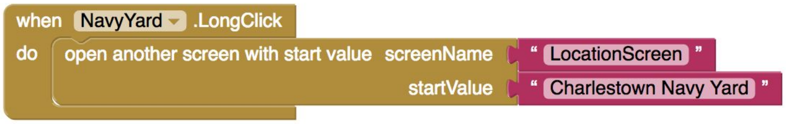


## *GalleryScreen*

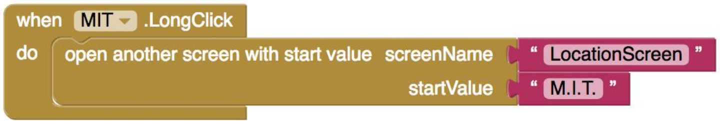


**Blocks** (NOTE: These are example blocks for markers in a Boston Tour Guide.)

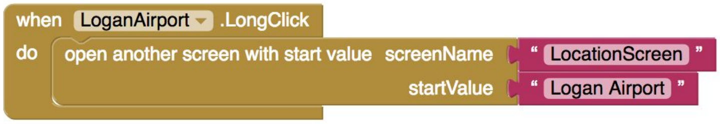
**Screen1 (Lesson 3):**



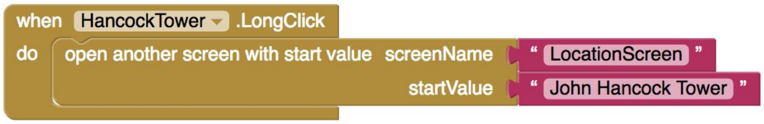
```
when NavyYard .LongClick
do  open another screen with start value  screenName  " LocationScreen "
                                          startValue  " Charlestown Navy Yard "
```



```
when MIT .LongClick
do  open another screen with start value  screenName  " LocationScreen "
                                          startValue  " M.I.T. "
```

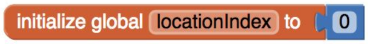


```
when LoganAirport .LongClick
do  open another screen with start value  screenName  " LocationScreen "
                                          startValue  " Logan Airport "
```

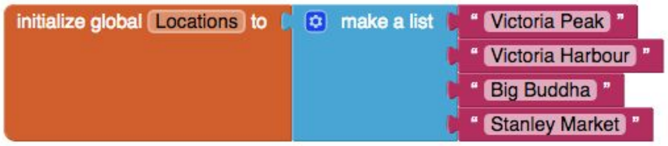


```
when HancockTower .LongClick
do  open another screen with start value  screenName  " LocationScreen "
                                          startValue  " John Hancock Tower "
```

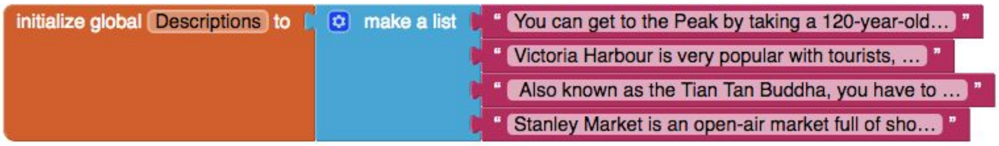
**LocationScreen (Lesson 3):**



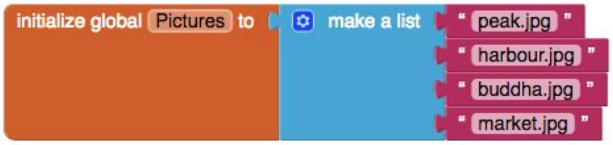
```
initialize global locationIndex to 0
```



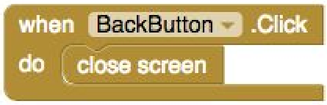
```
initialize global Locations to  make a list  " Victoria Peak "
                                             " Victoria Harbour "
                                             " Big Buddha "
                                             " Stanley Market "
```



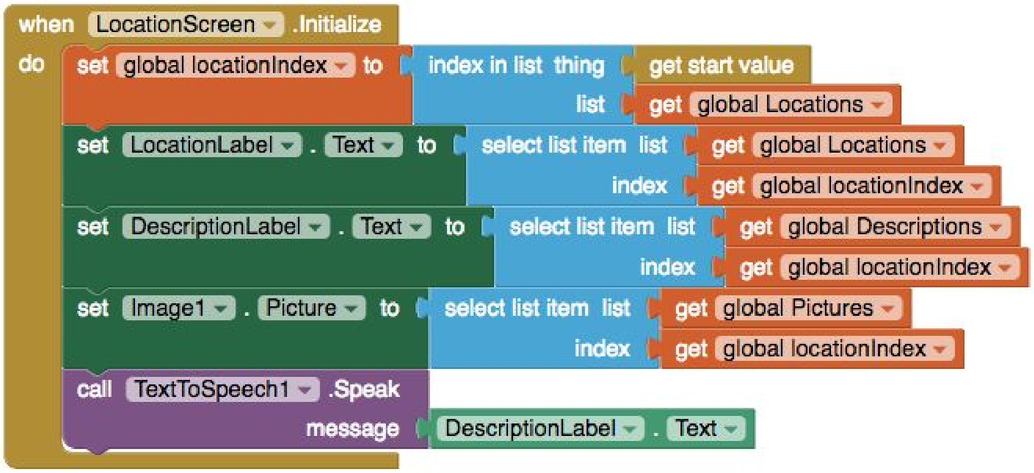
```
initialize global Descriptions to  make a list  " You can get to the Peak by taking a 120-year-old… "
                                                " Victoria Harbour is very popular with tourists, … "
                                                " Also known as the Tian Tan Buddha, you have to … "
                                                " Stanley Market is an open-air market full of sho… "
```



```
initialize global Pictures to  make a list  " peak.jpg "
                                            " harbour.jpg "
                                            " buddha.jpg "
                                            " market.jpg "
```

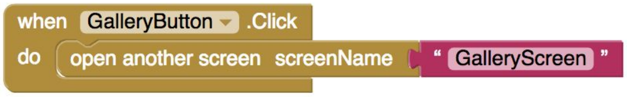


```
when BackButton .Click
do  close screen
```



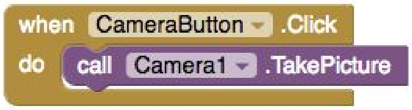
```
when LocationScreen .Initialize
do  set global locationIndex to  index in list  thing  get start value
                                                list   get global Locations
    set LocationLabel . Text to  select list item  list   get global Locations
                                                   index  get global locationIndex
    set DescriptionLabel . Text to  select list item  list   get global Descriptions
                                                      index  get global locationIndex
    set Image1 . Picture to  select list item  list   get global Pictures
                                               index  get global locationIndex
    call TextToSpeech1 .Speak
                     message  DescriptionLabel . Text
```

### LocationScreen (Lesson 4): added to blocks above

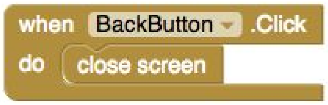


```
when GalleryButton .Click
do   open another screen  screenName  " GalleryScreen "
```

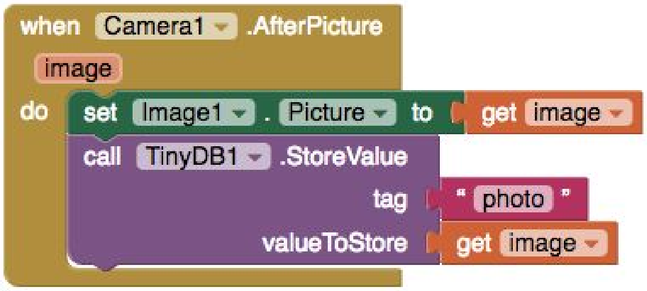
### GalleryScreen: (Lesson 4) added to blocks above



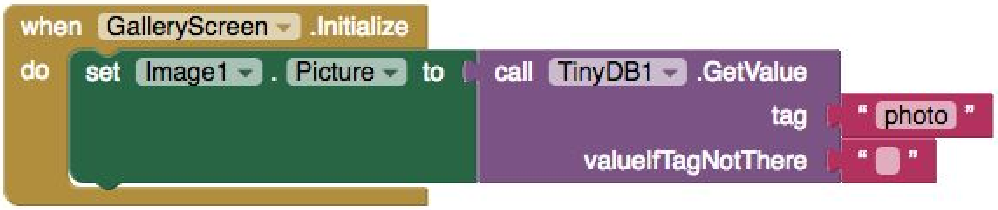
```
when CameraButton .Click
do   call Camera1 .TakePicture
```



```
when BackButton .Click
do   close screen
```

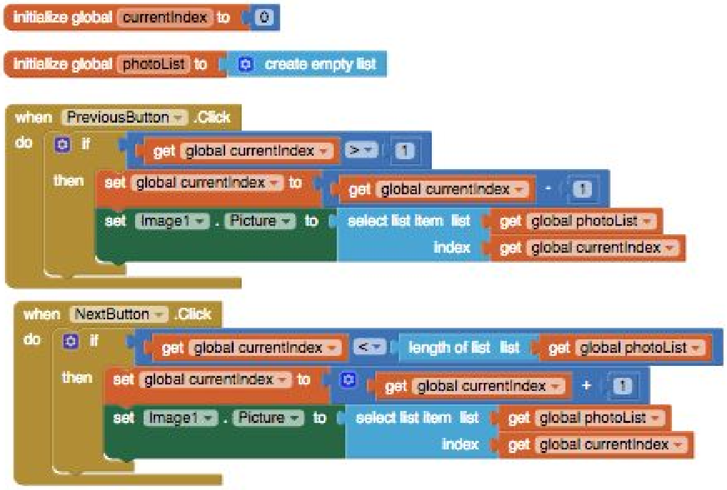


```
when Camera1 .AfterPicture
  image
do   set Image1 . Picture to  get image
     call TinyDB1 .StoreValue
                         tag  " photo "
                valueToStore  get image
```



```
when GallerySceen .Initialize
do   set Image1 . Picture to  call TinyDB1 .GetValue
                                               tag  " photo "
                                 valueIfTagNotThere  " "
```

### GalleryScreen: (Lesson 5) added to blocks above

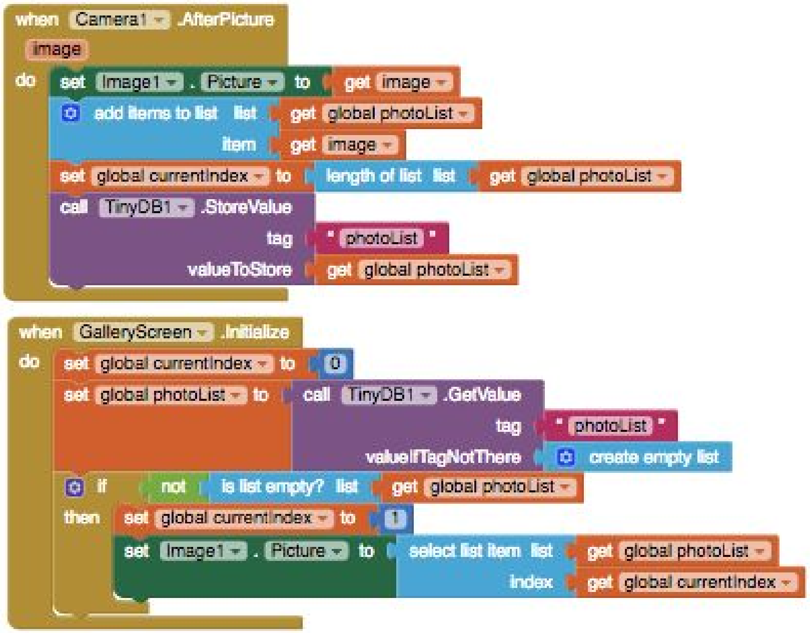


```
initialize global currentIndex to  0

initialize global photoList to  create empty list

when PreviousButton .Click
do   if    get global currentIndex  >  1
     then  set global currentIndex to   get global currentIndex  -  1
           set Image1 . Picture to  select list item  list  get global photoList
                                                     index  get global currentIndex

when NextButton .Click
do   if    get global currentIndex  <  length of list  list  get global photoList
     then  set global currentIndex to   get global currentIndex  +  1
           set Image1 . Picture to  select list item  list  get global photoList
                                                     index  get global currentIndex
```



```
when Camera1 .AfterPicture
  image
do  set Image1 . Picture to  get image
    add items to list  list  get global photoList
                       item  get image
    set global currentIndex to  length of list  list  get global photoList
    call TinyDB1 .StoreValue
                  tag  " photoList "
         valueToStore  get global photoList

when GalleryScreen .Initialize
do  set global currentIndex to  0
    set global photoList to  call TinyDB1 .GetValue
                                          tag  " photoList "
                             valueIfTagNotThere  create empty list
    if    not  is list empty?  list  get global photoList
    then  set global currentIndex to  1
          set Image1 . Picture to  select list item  list  get global photoList
                                                    index  get global currentIndex
```

# Tour Guide Teacher's Guide: Lesson 1

## Learning Objectives

At the end of this lesson, students should be able to:

1. Use latitude and longitude to mark a geographic location.
2. Take steps to plan a tour guide app, gathering visual and text content in preparation.

## Lesson Outline

### Introduction (10 minutes)

Start by a short class discussion to relate this unit to other apps students may have used.

1. Ask students:
    a. Have you used any maps-based apps?
    b. What did you use them for?
    c. Have you used an app to help you find out about places while travelling?

2. Explain to students that they will work in pairs to create an app that will highlight four different places in and around their community, or a place they would like to visit. The students will make an app that shows the location of the places with markers on a map. Clicking on a location will display information about that location in a pop-up. Long-clicking will open another screen with more detailed information and a picture of the landmark. A third screen allows the user to take pictures for a gallery of images.

3. Remind students that they will work using the Pair Programming model in this unit.

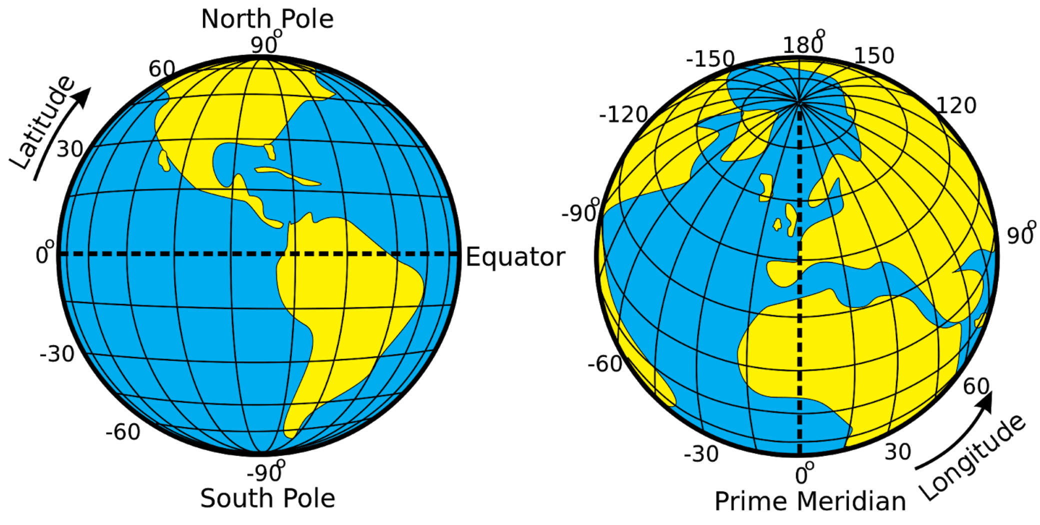
Remind them how the Pair Programming model works:

| DO | DON'T |
|---|---|
| ● Be respectful<br>● Talk to one another about the work<br>● Explain what you are doing<br>● Think ahead and make suggestions<br>● Switch roles often | ● Be a bossy navigator<br>● Grab the driver's mouse/keyboard |

4. Demonstrate the finished example app to students, showing all three screens. (BostonTourGuide_complete.aia)

### Geographic Coordinate System (30 minutes)

1. Explain to students that they will be using the geographic coordinate system, where latitude is the vertical marker for a location on the globe, and longitude is the horizontal marker.



Latitude is measured with 0º at the equator, the North Pole 90º, and the South Pole -90º. Longitude is measured with 0º at Prime Meridian, which is located in Greenwich, England. Clockwise around the globe is 0º to positive 180º, and counter-clockwise around the globe is 0º to negative 180º.

Each location on earth can then be marked by a location of latitude, longitude on the earth's surface.

2. Demonstrate using www.latlong.net.
    a. Type in a location (Boston, MA, or your city/town).
    b. Note the latitude and longitude that appears with the location at the bottom of the screen.
    c. Click left/right to see the change in longitude.
    d. Click up/down to see the change in latitude.

3. Have student groups fill out the *Landmarks Worksheet* with their partner. For each landmark, they must fill in the following:
    a. Location name
    b. Short description
    c. Longer paragraph (2-3 sentences) of information about the site
    d. Picture (digital picture) - this should be accessible online or on the student's computer, and they should have the exact filename listed on the worksheet.
    e. Latitude and longitude for each landmark, using www.latlong.net.

**Wrap-up (5 minutes)**

Check with student groups to make sure they have completed the Landmarks Worksheets.

**Homework**

Complete Landmarks Worksheet in preparation for next lesson, if not finished during class.

# Tour Guide Teacher's Guide: Lesson 2

## Learning Objectives

At the end of this lesson, students should be able to:

1. Use Map and Marker components in MIT App Inventor.
2. Programmatically open a second screen in an app
3. Work collaboratively using the Pair Programming model.

**Maps and Markers Demonstration (10 minutes)**

Introduce students to the **Map** and **Markers** components in MIT App Inventor by demonstrating the components at the board, and having students follow along at their computers.

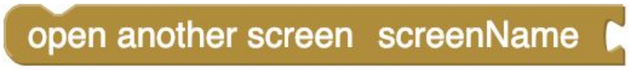
1. Check in with student groups to make sure they have their Landmarks worksheets completed.
2. Ask students to open the TourGuide_template.aia project in MIT App Inventor.
3. Drag a **Map** component onto the screen and change its *Width* and *Height* properties to "**Fill Parent**".
4. Drag out 4 **Marker** components onto the **Map** component.
5. Show students how to find the latitude, longitude for a city. As an example, Boston is at (42.360082, -71.058880) using http://www.latlong.net.
6. Set the *CenterFromString* property of the **Map** component to the latitude, longitude for the students' chosen cities.
7. Show students how changing the *ZoomLevel* property for the **Map** component will change the scale of the map. Setting the property to 12 will zoom out enough to see a good sized city area. Students can adjust larger or smaller, depending on where their landmarks are located.

8. Rename one **Marker** to one of the chosen landmarks. Remind students it is a good idea to name components to describe what they are being used for. For example, “LoganAirport”. No spaces are allowed in component names, so camel notation, with subsequent words capitalized, is suggested.
9. Show students how to change the latitude and longitude properties for a marker. For example, Logan Airport’s latitude is 42.365613, and its longitude is -71.009560.
10. Show students how to change the *Title* and *Description* properties for a marker.
11. Explain that checking the *EnableInfoBox* property for a **Marker** will pop up an information box for the marker when it is clicked. This feature will be used in **Screen1** of this app.

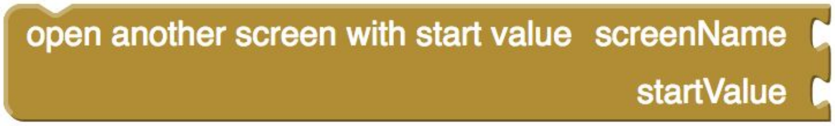
**Introduction to Using Multiple Screens (5 minutes)**

In this app, there are 3 screens (Screen1, LocationScreen, and GalleryScreen). The app will open the other screens at different points in the running of the app.

1. The **open another screen** and **open another screen with start value** blocks are found in the Control drawer.

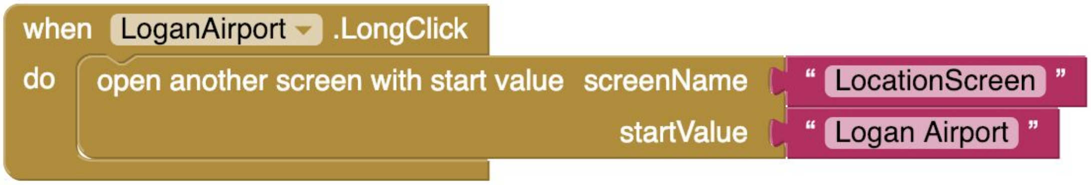


```
open another screen  screenName
```



```
open another screen with start value  screenName
                                      startValue
```

2. The *screenName* (a Text block) must match the name of the screen to be opened.
3. This app uses the open another screen with start value block. When a user long clicks on a Marker, the Location name of the Marker will be sent to the *LocationScreen* so it knows which Location to load and display.



```
when LoganAirport .LongClick
do  open another screen with start value  screenName  " LocationScreen "
                                          startValue  " Logan Airport "
```

**Coding Screen1 (25 minutes)**

1. Remind students they are to work using Pair Programming, so each partner in a group takes a turn (10 minutes) as the driver who codes, while the second person is a navigator who assists the driver (without touching the keyboard or mouse).

2. Ask students to update the properties for their 4 markers: Title, Description, Latitude, Longitude, according to the information on the Landmarks Worksheet. They may follow these directions in the ***Tour Guide Student Guide: Screen1***. They should also make sure EnableInfoBox is checked for all four markers.

3. Students can follow the ***Tour Guide Student Guide: Part 1*** to complete the coding for Screen1 in the Blocks Editor.

**Wrap-up (5 minutes)**

Check with student groups to make sure they have added all 4 markers and completed the coding for Screen1.

**Homework**

1. (Required) Complete Screen1 functionality for the app.
2. (Optional) Continue coding the Location Screen.

# Tour Guide Teacher’s Guide: Lesson 3

## Learning Objectives

At the end of this lesson, students should be able to:

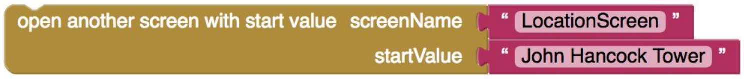
1. Write code to open a second screen in an app and to pass a start value to that screen.
2. Demonstrate understanding of lists as a way to store multiple pieces of related information.
3. Use parallel lists to organize and retrieve related information in an easily accessible way.
4. Collaborate using the Pair Programming model.

## Lesson Outline

### Introduction to Lists (20 minutes)

1. Check with students to see that they have completed **Screen1** of the app.

2. Review how LocationScreen is opened.
   a. The “open another screen with start value” block allows you to open another screen and pass it a “start value”. The “start value” is a variable that can be accessed in the second screen.



```
open another screen with start value  screenName  " LocationScreen "
                                      startValue  " John Hancock Tower "
```

   b. This app will pass the location name of the clicked marker between screens as a start value. That code was completed in Part 1.

3. Ask students if they recall using variables to store information in the Food Chase app. Show code blocks from the Food Chase app, to remind students how the the high score was stored in a variable. Explain to students that as they try to store more and more information, it helps to organize their variables into something called a list.

4. Depending on time and comfort level, choose either (or both) of the following unplugged activities to do with the students.

   a. Students act out elements of a list:

      (a) Have five students come to the blackboard and line up. (If space is limited, three or four will work).

      (b) Write the numbers 1-5 above each student's head on the blackboard to be the "index", which tells us where in the list they are.

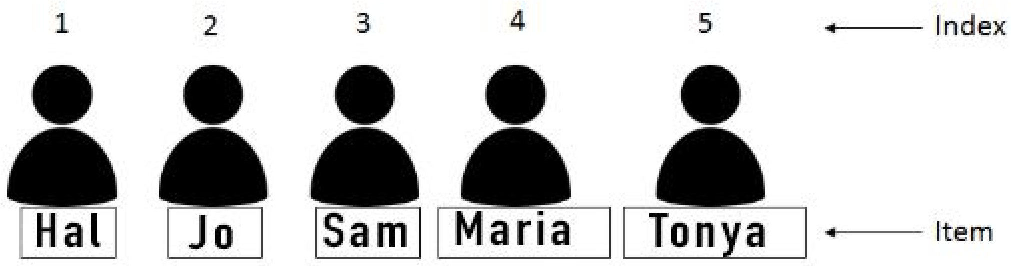
      (c) Hand each student a large piece of paper with their name on it.

      (d) Explain that the five students are simulating a list containing the names of students in the classroom.

      (e) When the teacher calls the number (index) written on the blackboard, the corresponding student will say their name.

      (f) Explain that just like variables, lists can hold other values, like numbers.

      (g) Hand each student a paper with a random number on it (better if they are different than the index), but still call on students by their index. You can also have each student hold a picture, and show it when their index is called (since you can store images in lists as well).



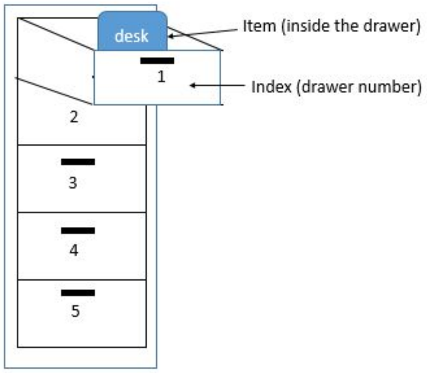
b. Use a file cabinet as an analogy for a list:

(a) Tape numbers 1-5 (or however many drawers available) on a file cabinet in the classroom.

(b) Inside each drawer, put a piece of paper with a word, number, (or even a picture)

(c) Demonstrate opening drawer 1 to get the value inside the drawer.

(d) Explain the analogy of a list to the set of drawers.



5. Explain to students that a List is a data structure, which consists of “indices” and “items”. Stress that the index is how you access a particular item in the list.

6. Explain that lists allow us to make the blocks simpler. Instead of using several variables, a single list can hold several values. Ask students to imagine making an app with 100 different words. Making 100 variables becomes too difficult to manage. Lists help organize many values under one name.

7. Explain and demonstrate using parallel lists for related information (using the table below). Students will use 3 lists in LocationScreen - Locations, Descriptions, and Pictures. Each list will contain the information for the 4 markers. The important thing is that they are all in the same order, e.g. MIT information are all the first element in each list. So when the start value is passed to LocationScreen, it will tell the app which information to pull out of each list, based on the same index.

| Locations | | Descriptions | Pictures |
|---|---|---|---|
| 1 | MIT | The Massachusetts Institute of Technology (MIT) is a private research university in Cambridge, Massachusetts. | mit-dome.jpg |
| 2 | Navy Yard | Charlestown Naval Shipyard, was one of the oldest shipbuilding facilities in the United States Navy. | navyyard.jpg |
| 3 | Logan Airport | Logan Airport is an international airport located in the East Boston neighborhood of Boston, Massachusetts | logan-airport.jpg |
| 4 | Hancock Tower | The John Hancock Tower, on the southeast corner of Copley Square, is a 60-story, 790 ft (240 m) skyscraper. | hancock-tower.jpg |

**Coding of LocationScreen (20 minutes)**

Ask students to work using the Pair Programming model and follow ***Tour Guide Student Guide: LocationScreen*** to complete the LocationScreen of the app.

1. Remind students of the Pair Programming model, and to take turns as driver.
2. Remind students to test and debug the app as they go. Student pairs should code the LocationScreen, then test to make sure that screen works correctly.

**Wrap-Up and Review (5 minutes)**

1. Check in with student groups to ensure they have finished coding the LocationScreen and tested it to make sure it works.
2. Review main concepts/components covered in this lesson:
    a. Multiple screens
    b. Lists and indexes

# Tour Guide Teacher's Guide: Lesson 4

## Learning Objectives

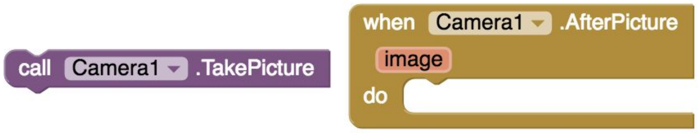
At the end of this lesson, students should be able to:

1. Use the Camera component to take a picture save the picture within an app.
2. Demonstrate understanding of the TinyDB component to persistently save image data within an app for later retrieval.
3. Work collaboratively using the Pair Programming model.

## Lesson Outline

### Introduction to New Components and Concepts (15 minutes)

1. Show students the **Camera** component in App Inventor.
    a. The **Camera** is included in the template, but show students that they can find it under the Media Drawer in the Designer.
    b. In the Blocks Editor, show students the **Camera** blocks they will use.



```
call Camera1 .TakePicture

when Camera1 .AfterPicture
  image
do
```

i. **Camera.TakePicture** switches to the device camera app and allows users to take a picture.

ii. **Camera.AfterPicture** is an event triggered after the user takes the picture and clicks the checkmark to accept the picture. The image parameter contains the image just taken.

2. Students will use **TinyDB** to store the image data. It will actually just save the file information for the pictures taken with the app. You can remind students of the persistence feature of TinyDB vs. variables.

| | **Variable** | **TinyDB** |
|---|---|---|
| **Persistence** | Anything stored in a variable is erased when the app closes | Anything stored in TinyDB with a tag can be retrieved at any point, even after the app is closed and reopened |
| **Storing data** | **set name** to value | **TinyDB.StoreValue** (tag, value) |
| **Retrieving stored data** | **get name** | **TinyDB.GetValue**(tag, valueIfTagNotThere) |

**Coding of GalleryScreen of App (25 minutes)**

Ask students to work using the Pair Programming model and follow ***Tour Guide Student Guide: GalleryScreen*** to complete the first version of the **GalleryScreen**.

1. Remind students of the Pair Programming model, and taking turns as driver.
2. Remind students to test and debug the app as they go. Student pairs should make sure that the first two screens (Screen1 and LocationScreen) work correctly before moving onto the **GalleryScreen**.
3. Students can then move on to code the third screen, the **GalleryScreen**. They should code the screen, then test to make sure that they can take a picture in the gallery, and that it appears when they shut down the app and restart it.

**Wrap-Up and Review (5 minutes)**

1. Check in with student groups to ensure they have completed the code blocks and tested the app to make sure it works.

# Tour Guide Teacher's Guide: Lesson 5

## Learning Objectives

At the end of this lesson, students should be able to:

1. Demonstrate understanding of the TinyDB component to persistently save image data within an app for later retrieval.
2. Use lists to store multiple items in TinyDB.
3. Demonstrate understanding of indexes to traverse through a list.
4. Work collaboratively using the Pair Programming model.

## Lesson Outline

**Review of Lists (10 minutes)**

1. Ensure students have completed the **GalleryScreen** that allows a user to take a single picture and save it.
2. Explain to students they will add ability to save a list of images for a complete gallery and allows user to scroll through the images.
   a. Ask students how they would retrieve the third item in a list. (using index 3)
   b. Ask students if they didn't know how long their list is, how can they access the last item in a list. (use **length of list** as the index - students may not immediately get this).
3. Unplugged Activity:
   a. Ask 4 or 5 students to come to the front of the class to stand in a row, and hand them each a picture.

b. Have a different student come to the front of the class to be the "index". Have pieces of paper with 1,2,3,4,5 ready. Ask students how they can look through the list of pictures starting at the beginning. The "index" must be set to 1, so hand the student the paper with "1" on it. Then 2, then 3, etc. So, the index must be increased by 1 after each picture is viewed. Exchange the correct value paper to the student so they hold up what the index value is at each step.

c. What happens when they get to the end of the list of pictures? Index must be set back to 1 to start over.

d. Now have the "index" move backwards through the list. What happens to the index? It is decreased by 1 each time.

e. Explain to students that this is what they will do, following the student guide. They will use an index to allow users to look at each picture, and will increase or decrease the index depending on whether they press the Next or Previous buttons.

**Coding of Improved GalleryScreen (25 minutes)**

Ask students to follow the ***Tour Guide Student Guide: Improved GalleryScreen***.

c. Complete GalleryScreen coding.

d. Test and debug GalleryScreen.

**Wrap-up and Review (10 minutes)**

1. Review lists, camera, TinyDB, Map components.
2. Ask students to complete multiple choice questions, survey of learning attitudes, and self assessment on collaboration.