Make a mobile app

Overview
Most phones and other mobile devices run apps. App is short for Application Software. An app is a set of instructions that control the computer. The instructions make the computer carry out a useful task. In this chapter you will make an app that displays an ID card on an Android phone.

Learning outcomes
By the end of this chapter you will know how to create simple applications software. You will use the App Inventor programming language, which runs on Android phones. You will know how to:
- design an interface with buttons, labels and text boxes
- make an event-driven program that reacts to user commands
- display text and images as output from a computer program
- describe syntax and error messages
- duplicate and delete code blocks
- upload and use multimedia content
- use logical tests and the if command to control the computer
- test and evaluate a program to see if it meets user needs.

Talk about...
Apps: There are millions of apps available to use on mobile phones. What apps do you know? What are the most useful apps? Are there any apps you would like that haven’t been invented yet?
Proof of ID: You often have to prove your identity, for example, when you go into the school building. What ways are there for a person to prove their identity? What are the advantages and disadvantages of the different methods?

Design an ID Card
In this chapter you will make an on-screen ID card that shows your name and image. Work in pairs or small groups to make a design for an ID card. What extra information would you show as well as name and photo?
Learning outcomes
In this chapter you will make an app for a phone. Your app will display your ID on the screen of a mobile phone. In this lesson you will start to design the screen.
When you have completed this lesson you will be able to:
- design an interface with buttons and labels.

Learn about...
In this chapter you will write a program using App Inventor. You will:
- design the screen of the app
- write program code that goes with the app.
You will make an app which displays your identity on the screen of your phone.

Interface
An interface is the part of a program that helps the human user. The interface lets you send a message to the computer. This is called input. The interface lets the computer send a message to you. This is called output.
A modern mobile phone has a computer inside it. The screen of the phone is used for input and output. The user touches the screen to control the computer. That is input. The computer shows words and images on the screen. That is output.

Interface design
You will make an interface that will have a button and a label. When you click the button, the label will show your name.

How to...
To get started, log in to the App Inventor website. Your teacher will show you how. At the top is a menu bar.
- Open the first menu, Projects, and select Start new project
- Give your first project a one-word name, such as: IDCard
The project will be linked to your personal login details. Nobody else will be able to see it.

Designer
Below the menu bar is the Designer screen. This is where you will design the interface for your app.

The Designer screen is split into four sections. Running from left to right, these are:
1. Palette
2. Viewer
3. Components
4. Properties
You will use all four sections to make the interface.

Palette
Look at the Palette. The Palette shows a list of different objects. You can use these objects to make an interface. You drag objects from the Palette onto the Viewer.
- Drag a Button and a Label onto the Viewer

Viewer
Look at the Viewer. This is where you see the interface design. You have put two objects onto the interface: a Button and a Label.
Components
Look at the Components section. The Components section lists the objects you have added to the interface.
The objects are called Button1 and Label1. You can change the names to IDButton and NameLabel. These names will help you remember what the objects are for.

- Select Button1
- Find the Rename button at the bottom of the Components list. Click the Rename button
- Type the new name for the object: IDButton
- Click OK
- Now change the name of Label1 to: NameLabel

Set object properties
The final section of the screen is called Properties. Every object has properties. A property is any feature of an object, like its colour or size. You can change the properties of an object, such as the text property. That means the text that is written on an object.

- Select the object IDButton
- Look down the Properties column until you find the Text property box
- Type the text: Click for ID
- Change the text property of NameLabel to: Name:

Now, when you look at the Viewer the objects show the text property you gave them.

Run the app
You can run the app you made using one of two methods:
- Use the App Inventor Emulator on your computer. You will see an image like a mobile phone screen on your computer.
- Use the App Inventor Companion. You will need a mobile device with a wireless connection. You will see the app on the screen on the mobile device.
Your teacher will tell you which to use. When you run the app you will see the interface you made.

The interface does not do anything yet. In Lesson 2.2 you will add code so that it works as an ID system.

Now you do it...
- Start App Inventor. Create an interface with a button and a label. Give the objects suitable names. Set the text properties of the objects.
- Run the app and check the interface you made. Correct any errors you see.

If you have time...
- There are other object properties, for example, background colour and font size. Change some of these properties. See if you can improve the appearance of the interface.
- If you click Screen1 you can change the colour of the screen itself.

Test yourself...
1. What is the difference between input and output?
2. Every object in App Inventor has properties. List three properties that an object might have.
3. The interface you made has two objects. Which object is used for input and which for output?
4. You changed the names of two objects. What makes a good name for an object?

**App Inventor**
App Inventor is a programming language that is available free on the Internet. App Inventor is provided by a university called the Massachusetts Institute of Technology (MIT). Anyone on the Internet is allowed to use App Inventor. You use App Inventor to make apps for mobile phones.

**FACT**
**App**: App is short for Application Software. An app is a set of instructions that control the computer. Apps are created by programmers using a programming language.

**Input**: Input means signals and information sent from the user to the computer.

**Output**: Output means signals and information sent from the computer to the user.

**Property**: A property of an object is any of its features, such as its colour or size.

**Text property**: The text property of an object means the words written on it.

**Naming objects**
The name of an object must be one word with no spaces. The name of an object should explain what it is and what it does. The button on this app will show your ID. The button could be named IDButton.
Learning outcomes

You have made an app interface with two objects on it. Now you will write program code. The code will make the app carry out actions in response to user input.

When you have completed this lesson you will be able to:
- make an event-driven program that reacts to user commands
- display text as output from a computer program.

Learn about...

In this lesson you will write program code. The code will tell the computer to show your name on the screen. You will make the code by fitting blocks together.

The code is carried out when the user clicks the button called IDButton. An event that makes the computer carry out an action is called a trigger. Clicking IDButton is a trigger.

How to...

Make sure your app is open in the web browser. Look at the top right of the screen. There are two buttons. These buttons let you swap between two different views.

1. Designer: This screen is where you make the interface.
2. Blocks: This screen is where you make the program code.

Select the Blocks button

Look at the Blocks screen. The Blocks screen has two sections: Blocks and Viewer. You choose blocks from the Blocks section and then drag blocks onto the Viewer section. You can build up the code by fitting blocks together.

- IDButton
  You can make code that is triggered when the user clicks IDButton.
  Look at the Blocks section. You will see a list of different types of block. At the bottom of the list find IDButton.

- Click IDButton
  When you select IDButton you will see a range of blocks. These blocks are all linked to IDButton.

- Select the top block called: when IDButton.Click
- Position this block on the Viewer
  Your screen now looks like this.

The block has a space in it that you can put blocks into. These blocks tell the computer what to do. When the user clicks IDButton, the computer will do these actions.
Set NameLabel text
When the user clicks IDButton their name will appear in NameLabel. This means you need a block that will set the text property of NameLabel.

- **Look in the Blocks section**
- **Click NameLabel**
  You will see a range of blocks on the screen. The blocks are all linked to NameLabel.
- **Scroll down through the list to find the block which lets you set the text property of NameLabel**
- **Drag this block onto the Viewer**
  It will fit exactly into the space in the other block. Fitting the blocks together is like making a jigsaw.

These blocks mean, “When the user clicks IDButton, set the text property of NameLabel.” There is a warning symbol at the bottom left of the blocks screen. The warning symbol appears because your task is not finished yet. Now you must tell the computer what text to show in NameLabel.

**Add new text**
Look in the Blocks section. Find the pink Text category. Click this to see all the Text blocks. The block you need is at the top of the list of Text blocks. The block has an empty space where you can type any text.

- **Drag the empty Text block onto the Viewer**
- **Fit it into the NameLabel block**
- **Put your name in the empty space**
In this example we are using the name Pharaoh Tutankhamun. Use your own name or any name you like. The program blocks now look like this:

These blocks mean, “When the user clicks IDButton, set the text property of NameLabel to Pharaoh Tutankhamun.” The program is finished.

**Run the app**
Run the app you made to see what it looks like. Remember there are two ways to do this. You can use the App Inventor Emulator on your computer, or you can use the App Inventor Companion on a wireless device. Click the IDButton and your name will be displayed on the screen.

**Now you do it...**
1. Use the Blocks screen to create a simple program. The program will show your name when you click IDButton.
2. Run the app to check that it works and correct any errors.

**If you have time...**
You can add extra blocks inside: when IDButton.Click. When the user clicks IDButton, these extra commands will be carried out.

- Extend the program interface to include more labels. These can display extra facts such as ‘Date of birth’ or ‘School name’. Give these objects suitable names.
- Add extra code blocks inside: when IDButton.Click. These extra blocks will change the text in the new labels that you added. When the user clicks IDButton they will see the new text in these labels.

**Test yourself...**
1. Why do you think App Inventor is sometimes called a **visual programming** language?
2. The code you made is triggered when an event happens. What event triggers the action?
3. Explain the purpose of the Designer screen and the Blocks screen in App Inventor. What is the difference between them?
4. When the trigger happens, what action is carried out by the computer?

**Key words**

**Event-driven**: Computer software can be ‘event driven’. This means the actions of the software are triggered by events. Until the event happens the software won’t do anything. App Inventor is an event-driven programming language.

**Visual programming**: Visual programming means that you make the program by fitting shapes together. App Inventor is a visual programming language.

**Trigger**: A trigger is an event which causes the computer to carry out an instruction.
Learning outcomes

You made a simple interface. You added program code. In this lesson you will add an extra button to the interface, plus code to make it work. This is a Reset button. When the user presses the Reset button the text of NameLabel will blank out.

When you have completed this lesson you will be able to:
- describe syntax and error messages
- duplicate and delete code blocks.

Learn about...

Every programming language has rules. The rules are the syntax of the language. If you write an instruction with the wrong syntax it will not work at all. The computer will not be able to understand the instruction. This is called a syntax error.

Every programmer makes errors sometimes. If you make errors it does not mean you are bad at programming. The key to good programming is to spot the errors, then fix them. If you can do this you can be a good programmer.

Error messages

The computer may spot a syntax error in your program. The computer will show an error message or warning. The message should tell you where the error is and how you can fix it. Look out for error messages as you complete this task.

How to...

You have learned how to add buttons to the interface. Now you will add an extra button. The new button is called a Reset button. When the user clicks this button the text of NameLabel will go back to blank.

- Add a new button to the interface. Name the button: ResetButton. Make the text property: Reset
- Go to the Blocks screen. Look in the menu on the left. Find ResetButton and click it. You will see many blocks that belong with ResetButton.
  - Find the block that says: when ResetButton.Click
  - Drag this block onto the Designer section of the screen. This block will be triggered when the user clicks ResetButton.

If you can’t remember how to do these things, look back at Lesson 2.1 or 2.2.

Duplicate blocks

In Lesson 2.2 you made a block of code that set the text of NameLabel. Now, you can make a copy of this block of code. You can make a small change to the copy so that it sets the text to a blank.

- Look at the code blocks you have already made. Find the block that sets the text of NameLabel
- Right-click this block. That means click the block with the right button of the mouse
- A menu appears. Choose Duplicate from the menu

You have made an exact copy of the program block.

- Drag the duplicate block down to the ResetButton block. The duplicate block will fit exactly into the new place
- Delete your name so the Text block is empty
- The blocks mean, “When the user clicks ResetButton, set the text property of NameLabel to blank” The program is ready.
Delete blocks
Can you see the rubbish bin at the bottom right of the screen? Sometimes you may accidentally make a block you don’t want. You can drag any unwanted block into the rubbish bin. That will delete the block from the screen.

Syntax errors
Commands in App Inventor are held as blocks. The blocks look like jigsaw pieces. If you try to connect the blocks in the wrong order, then they will not join up. Here is an example. These blocks are the wrong way round. They won’t fit together.

Below these blocks you can see a warning message. There is an exclamation mark in a yellow triangle. A number 2 next to it tells you there are two warnings.

- Put the blocks the wrong way round to see this error
- Click Show Warnings

Exclamation marks appear on the blocks that are the wrong way round. If you click an exclamation mark you will see a message explaining the problem.

- Now you do it...
  - Create a Reset button for the app. Be aware of warning messages and syntax errors as you make the code.
  - Run your app to check it works.

- If you have time...
  - Change the properties of the Reset button, such as colour and size.
  - Change the code so that when you press the Reset button the text STAND BY is displayed.

- Test yourself...
  1. What is program syntax?
  2. “A good programmer never makes errors.” Discuss this statement – is it true?
  3. Explain how error messages and warning messages may help a programmer do their job.
  4. Explain how duplicating or copying a block of code may help a programmer do their job.

Key words
Right-click: Click with the right button of your mouse instead of the usual left button.
Syntax: The rules of a programming language are called syntax.
Syntax error: A syntax error is when the code breaks the rules of the programming language.
Learning outcomes
You have created a simple app which displays your identity as text. In this lesson you will adapt the app so it also shows a photo of your face. This will help confirm your identity.
When you have completed this lesson you will be able to:
- upload and use multimedia content.

Learn about...
Modern apps use multimedia content. That means images, sound and video. Multimedia content makes an app more interesting and enjoyable for the user. There are many ways to make or get multimedia content. For example, you can take a photo with a modern mobile phone. You can also record sound and video clips.

Prepare an image
Take a photo of yourself with a mobile phone, or ask a friend to do it. Transfer the photo to your computer. You could also find an image file on the Internet and make a copy on your computer.

How to...
When you upload the image file, you will first upload the image ready to use. Look at the Designer screen of App Inventor. At the bottom of the Components section is a small area for Media files.
- Click the button that says Upload File...
- Then click Browse and find the image file you want to use (perhaps a photo of you)
- Click OK to upload it

Add an image object to the screen
Now you will add an image object to the interface. This object will display an image on the interface.
- Look in the Palette. Find the object called Image and drag it onto the screen
The image object appears on the interface. The image object looks like a little green block. You won’t see the image yet as this is just a marker to show you where the image will be. We have put it above the Reset button. The image object is called Image1.

Set object properties
You can change the properties of Image1. First link it to your picture.
- Select the image object
- Look at the Properties section of the screen. Find the property called Picture
- Click this property. You will see a list of uploaded images. Pick the name of your picture
Make Image1 invisible for now.
- Find the property called Visible. The Visible property is a box with a tick
- Untick this box
Now the Properties section of the screen looks like this:
Add code blocks
You made Image1 invisible. Now you can add code that makes Image1 visible. The image will become visible when the user clicks IDButton.

- Open the Blocks screen
- Select the component Image1
- Find the block that says: set Image1.Visible

Put this block into the IDButton code. The block will fit exactly, like a jigsaw piece.

Can you see the red warning symbol on this block? That tells you the task is not finished yet.

- Look in the Blocks section. Find the green Logic category. Click to see Logic blocks
- The first block is called ‘true’. Drag this block onto the screen
- Fit the blocks together as shown here

These blocks mean, “When the user clicks IDButton, make Object1 visible.” The program is ready.

Run the app
- Run the app by clicking IDButton
- You should see your picture on the screen

Now you do it...
- Make changes to the ID app so that it displays your picture.

If you have time...
Edit the Reset button so that when the user clicks this button, the picture becomes invisible.
- Duplicate the block that makes Image1 visible.
- Fit the duplicate into the ResetButton block.
- Change the value from true to false. There is a little menu you can use.

Test yourself...
1. What is the event that makes the image visible?
2. What is the advantage of showing an image on an ID card?
3. What else could you include in an ID card?
4. You added an image object to the interface. What is the purpose of this object?
5. You changed two properties of the image object. Why did you have to change these properties?

Multimedia
Multimedia content can be stored using many different formats. The name of the multimedia file includes a three-letter code that tells you the format. JPEG and GIF are suitable image formats.
WAV and MP3 are suitable audio formats.

FACT
 Multimedia: Multimedia content means images, sound and video.
Upload: When a file is uploaded, it is copied onto a computer that is connected to the Internet. The file can be accessed and used over the Internet.
Learning outcomes

You have created a simple app that displays your name and photo when you press a button. In this lesson you will adapt the app by adding a password. The ID card app will only work if you know the right password. When you have completed this lesson you will be able to:
- design an interface with text boxes
- use logical tests and the if command to control the computer.

Learn about...

In this lesson you will change the program. The user will enter a password. The app will test to see if the password is right. If it is right, the app will show the name and photo.

Logical test

This is an example of selection. You learned about the idea of selection in Chapter 1, Computational Thinking. Selection is an important part of programming. In this lesson you will use selection with App Inventor.

A programmer often wants to make a computer carry out a test and then make a choice. For example, a programmer wanted to write a program to control a self-driving car. Here are some things the program had to do:

- If the driver enters the right passcode, the car will start.
- If the sensor detects an object ahead, the car will steer around it.
- If the speed is too great, the car will slow down.

Each example starts with the word if. Then there is a logical test. A logical test means a test which only has two possible answers:
- true
- false.

For example, either there is an object ahead (the test is true) or there isn’t (the test is false).

Equals

The logical test you will make in this lesson uses the equals sign. The test is true if the two values on either side of the equals sign are the same. There are other types of logical test and you will learn about them in later chapters.

How to...

You can add a new object to the interface. The new object is a text box. That is a space on the interface where the user can type input. The user enters the password into this box.

Look in the Palette. You will see there is a special kind of text box called a PasswordTextBox. The PasswordTextBox is a text box that shows *** instead of the letters you type. No one can read your password over your shoulder.

- Drag the Password text box onto the interface

Make a logical test

Now you can make the computer test the password. Remember you must start with the word if. There is a special block for that:

- Open the Blocks screen
- Find the yellow Control blocks
- The first block in this section is the 'if' block. Drag it onto the Viewer

Now you need to add a logical test. The test will use the equals sign (=). The equals sign compares two values. If the two values are equal the test is true.

- Find the green Logic blocks
- One of the blocks in that section is the equals block
- Slot the equals block into the 'if' block. The equals block fits exactly

Now you have to tell the computer what two values to compare. There are two gaps in the equals block. You will put two different values into those two gaps. If the two values match then the test is true.
On one side put the password text. That is the text entered by the user into the password box:

- Look at the list of blocks on the left of the screen
- Click the object PasswordTextBox1 to find the blocks that go with this object
- Find the block that says: PasswordTextBox1.Text
- Drag that block into the gap in the equals box

On the other side type the right password.

- Find the pink Text blocks
- Drag an empty text box into the other gap in the equals box
- Type the word in the empty text box that you want to be the password

The finished logical test will look something like this:

Let us take a second to read through what these blocks mean. “When the user clicks the ID button, if the password is right, then show the name and photo.” That is the right instruction.

- Run the app to check that the program works

**Now you do it...**

- Add password protection to your app by adding a password box to the interface. Use the if block to make the computer check the password text. You can choose any password you like.

**If you have time...**

- A short extra task to test your skills: Add a new block to the Reset button. When the user clicks the Reset button the password box will reset to blank.
- A longer task if you have made good progress: Add an extra button to the app. If the user clicks on the new button a different name and face will appear on the screen. Now the ID app can be used by two people. You will have to do a lot of work to make this happen, but it is good practice at using your programming skills.

**Test yourself...**

1. When you type letters into a password text box, the letters do not appear on the screen. Instead you see this symbol *. Why is that?
2. A logical test only has two possible answers. What are they?
3. In this lesson, there are three examples of tests that might be needed by software that runs a self-drive car. Think of some more tests that might be needed by this software.
4. Describe the logical test you used in this lesson. When is the test true?

**FACT**

*Common passwords*

Some passwords are very common. Lots of people use these passwords. The most common passwords are ‘123456’, ‘qwerty’ and ‘password’. Do not use passwords like these. That makes it easy for other people to guess your password.
Learning outcomes

You have made a simple app which shows an ID card on the screen of the computer.

When you have completed this lesson you will be able to:

- test and evaluate a program to see if it meets user needs.

Learn about...

Every app is made for a purpose.

- Sometimes a programmer works for a client. The client pays the programmer. The client tells the programmer what the app should do.
- Sometimes the programmer makes an app to sell to customers. The customers will buy the app if it is useful to them.

In both cases the app has to be useful. The app has to do what the client or the customers want. Otherwise, no one will pay for the app.

When you have made an app you must evaluate it. This means checking that the app does what the client wants. If the app passes the evaluation, it is ready for the client.

The output of the program is what the client will see when they run the app. When we evaluate a program we evaluate the output. The output should match what the client wants.

Testing

The programmer has to make sure an app produces the right outputs. The programmer will test the app. Testing involves input and output:

- Type input into the app. The input is called test data.
- See what the output is. The output is called test results.

If the test results do not match client requirements then the programmer must make corrections. Then the programmer will test the app again. When the test results match client requirements, the app is ready. The programmer can give the app to the client.

The software development process

This diagram represents the process of developing software.

<table>
<thead>
<tr>
<th>Get client requirements</th>
<th>Make the app</th>
<th>Test the app</th>
<th>Compare test results to requirements</th>
<th>If results match requirements, app is ready</th>
</tr>
</thead>
</table>

Extension and improvement

If the app meets client requirements it is completed and no more work is needed. But sometimes the programmer will think of other improvements. They will add extra features to the app to make it even better.

How to...

There are two buttons on the app: IDButton and ResetButton. There is one other input on the app: the password. You can test all these inputs. Try these tests:

- Type the right password and click IDButton
- Click ResetButton
- Type the wrong password and click IDButton
- Don’t type a password at all and click IDButton

Here is the test plan. ‘Test Data’ is what we will type. ‘Expected Result’ is what we should see if the app is working right. The last two columns are blank for now.

<table>
<thead>
<tr>
<th>Test</th>
<th>Test Data</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDButton and right password</td>
<td>KingBC</td>
<td>Name and photo appear</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Click ResetButton</td>
<td>Name and photo disappear</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IDButton and wrong password</td>
<td>12345</td>
<td>No name or photo</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>IDButton and no password</td>
<td></td>
<td>No name or photo</td>
<td></td>
</tr>
</tbody>
</table>

- Open a word-processed document and make this table
Carry out tests

Now you will carry out the tests.

- Run the app
- Carry out the first test. Input the test data
- Record the output in the Actual Result column
- Repeat for all the tests

Analyse test results

Finally, compare the actual results and the expected results. If they match, the program is working correctly. If they do not match, the program must be improved. This is called analysis of test results.

- Write the analysis of test results in the final column of the table

<table>
<thead>
<tr>
<th>Test</th>
<th>Test Data</th>
<th>Expected Result</th>
<th>Actual Result</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IDButton and right password</td>
<td>KingBC Name and photo appear</td>
<td>Name and photo appear</td>
<td>Passes the test</td>
</tr>
<tr>
<td>2</td>
<td>Click ResetButton</td>
<td>Name and photo disappear</td>
<td>Name and photo disappear</td>
<td>Passes the test</td>
</tr>
<tr>
<td>3</td>
<td>IDButton and wrong password</td>
<td>12345 No name or photo</td>
<td>No name or photo</td>
<td>Passes the test</td>
</tr>
<tr>
<td>4</td>
<td>IDButton and no password</td>
<td>No name or photo</td>
<td>No name or photo</td>
<td>Passes the test</td>
</tr>
</tbody>
</table>

If the actual and expected results do not match, you have found an error in the program. You must correct the error.

Finished program

Here is the completed program with all blocks in place.
1.1 What is computational thinking?

Overview
In this chapter you created a simple app for a mobile phone. The app displays an ID card. The card includes your name and your photo. The ID card is password protected.

You have learned how to:
- design an interface with buttons, labels and text boxes
- make an event-driven program that reacts to user commands
- display text and images as output from a computer program
- describe syntax and error messages
- duplicate and delete code blocks
- upload and use multimedia content
- use logical tests and the if command to control the computer
- test and evaluate a program to see if it meets user needs.

Test questions
Answer these questions to check how well you have learned this topic.

1. A touch-screen interface is made up of objects. Describe three objects you can add to a touch-screen interface.
2. The objects on an interface have properties. Describe three properties of interface objects.
3. What is a client?
4. Explain why you must know the client requirements before you begin making an app.
5. App Inventor is an event-driven visual programming language. Explain what that means.
6. What is a logical test?
7. Explain how the equals sign is used in a logical test.

The next three questions relate to this image. The image shows a block of program code. The code is part of a science quiz.

8. What is the logical test in this block of code?
9. What happens if the logical test is true?
10. What event triggers this block of code?

Assessment activities
A programmer made an app that shows pictures of famous landmarks. Here is the interface she made:

Here is part of the code which makes this app work:

This is what the app looks like when it is running:

Starting activity
Make the interface and code that you see on this page. You will need to:
- add buttons, labels and images to the interface
- give names to the objects and change the text
- find an image of Machu Picchu and upload it
- make the code that goes with the Machu Picchu button
- copy what you see on this page.

Intermediate activity
Add code to all buttons on the interface. If the user clicks on any button they will see an image of that landmark and its name on the screen.

Extension activity
Add password protection and a Reset button to the ‘Famous Landmarks’ app.
Test the app and record test results.
If you have time, add extra features to the app.