<table>
<thead>
<tr>
<th>Section</th>
<th>Subsections</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introduction</strong></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td><strong>1 Computational Thinking</strong></td>
<td>1.1 What do I know?</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>1.2 Using pseudocode</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>1.3 Exhaustive search algorithms</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>1.4 Sort algorithms</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>1.5 More search algorithms</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>1.6 The fastest route</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about computational thinking</td>
<td>32</td>
</tr>
<tr>
<td><strong>2 App Inventor</strong></td>
<td>2.1 Make an interface</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>2.2 Adding up</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>2.3 Work out the total score</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>2.4 Work out percentages</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>2.5 Text strings</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>2.6 Fixing run-time errors</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about App Inventor</td>
<td>60</td>
</tr>
<tr>
<td><strong>3 Data and the CPU</strong></td>
<td>3.1 Bits, bytes and base 2</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>3.2 Adding in binary</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>3.3 Measuring computer memory</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>3.4 Digital images</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>3.5 How computers store true colour</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>3.6 Image quality</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about data and the CPU</td>
<td>88</td>
</tr>
<tr>
<td><strong>4 Introducing Python</strong></td>
<td>4.1 Calculate</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>4.2 Repeat</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td>4.3 Exit condition</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>4.4 Make a list</td>
<td>104</td>
</tr>
<tr>
<td></td>
<td>4.5 Find the largest value in the list</td>
<td>108</td>
</tr>
<tr>
<td></td>
<td>4.6 Validate input</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about Python</td>
<td>116</td>
</tr>
<tr>
<td><strong>5 Information Technology</strong></td>
<td>5.1 Connecting to the Internet</td>
<td>120</td>
</tr>
<tr>
<td></td>
<td>5.2 Hardware and protocols for the Internet</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>5.3 Sending data</td>
<td>128</td>
</tr>
<tr>
<td></td>
<td>5.4 Searching the web</td>
<td>132</td>
</tr>
<tr>
<td></td>
<td>5.5 Narrowing your search</td>
<td>136</td>
</tr>
<tr>
<td></td>
<td>5.6 Following your route online</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about information technology</td>
<td>144</td>
</tr>
<tr>
<td><strong>6 Creative Communication</strong></td>
<td>6.1 What do I already know?</td>
<td>148</td>
</tr>
<tr>
<td></td>
<td>6.2 Plan your project</td>
<td>152</td>
</tr>
<tr>
<td></td>
<td>6.3 Adding structure to your web page</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>6.4 Structuring a web page</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>6.5 Creating HTML links</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td>6.6 Creating a graph using Microsoft Excel</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>Review What you have learned about creative communication</td>
<td>172</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td></td>
<td>174</td>
</tr>
</tbody>
</table>
Overview
In Matrix 1, you completed a simple task using App Inventor. You created an electronic ID card that was password protected. If you are not sure how to use App Inventor, look at Matrix 1, Chapter 2, App Inventor, before you go any further.
In this chapter you will use App Inventor to make an app for a sports journalist. The app will help a sports journalist to record key facts about a sports event.

Learning outcomes
By the end of this chapter you will know how to:

- design an interface to suit users
- make an app that records and counts events
- use arithmetic operators to change values
- initialise, set and get variables
- use the real and string data types
- find and fix run-time errors
- use conditional structures and logical tests.
Design an interface

On paper, design the interface for a mobile app. The interface will help the person who uses the app to record facts during a football match. You can make the interface as complex as you like.

Talk about...

What is your favourite sport? What are the key facts about this sport? What type of data would a reporter need to record when they report on a game or event in this sport?

FACT

Beware American spellings

There are some differences between American and British spelling. App Inventor was made in the USA so it uses American spellings. For example, it uses ‘Center’ instead of ‘Centre’.
2.1 Make an interface

Learning outcomes
In this chapter you will create a mobile phone app. This app will count goals scored. In this lesson you will create the interface for the app. You will use skills you learned in *Matrix 1*, Chapter 2, App Inventor. Read that chapter again if you need a reminder of how to use App Inventor.

When you have completed this lesson you will be able to:
- design an interface to suit users.

Learn about...
In this chapter you will make an app to count the number of goals at a football match. In this lesson you will make the interface for the app.

An interface is used for input and output. A mobile phone has a touchscreen. The screen is used for both input and output.

- Input: the screen will have a button for the user to press to count each goal.
- Output: the screen will display the number of goals as a label.

The user
The person who will use the app is called the user. The programmer must make sure the interface meets the user’s needs. The programmer has to think of things like:

- what language the user speaks
- whether the user has good computer skills
- how the user will use the app.

This app will be used at a football match. The user will be busy watching the match. The match will be a crowded and noisy place, so the interface must be clear and bright. The interface must be easy to read.

Object properties
In this lesson you will add objects to the interface. The objects are buttons and labels. You will set the properties of the objects. Properties include colour, text and position. If you choose the right properties you will design a good user interface.

How to...
Start a new project on App Inventor and find the sections called Palette and Viewer. The Palette includes Layout boxes. You can use a Layout box to help you organise the objects on the interface.
Click the Layout section of the Palette. Find the box called: VerticalArrangement
Drag the box onto the Viewer

Set properties
Look in the Properties section of the screen. You will set the properties of the Layout box.

Click the property: AlignHorizontal
Choose Center from the drop-down menu. ‘Center’ is American spelling. App Inventor comes from the USA.
Click the property Width

Select the option Fill parent
You have changed the width so that the Layout box fills the whole width of the screen.
Add objects
You have added a Layout box to the interface.
Remember, the Palette section stores lots of different objects. You can drag objects onto the Viewer. That is how you make the interface.

Now you will add buttons and labels to the interface. The buttons and labels go inside the Layout box.

 aç Drag two labels onto the Layout box
 aç Drag two buttons onto the Layout box

You have set the alignment of the Layout box to Center. That means the objects will appear in the centre of the box.

Give names to the objects
You have added four objects to the interface.

Now you will set the name and the text property of each object. This table has some suggestions for names and text you might use.

If you can’t remember how to change the name and text of an object, look back at Matrix 1, Chapter 2, App Inventor.

<table>
<thead>
<tr>
<th>Object</th>
<th>Object Name</th>
<th>Text property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label1</td>
<td>HeadingLabel</td>
<td>Counter</td>
</tr>
<tr>
<td>Label2</td>
<td>Team1Label</td>
<td>0</td>
</tr>
<tr>
<td>Button1</td>
<td>Team1Button</td>
<td>Count + 1</td>
</tr>
<tr>
<td>Button2</td>
<td>ResetButton</td>
<td>Reset</td>
</tr>
</tbody>
</table>

Design the interface
Now you can design the interface. Make it bright and easy to read. Make changes to label and button properties, such as:

 aç font size, bold, typeface
 aç background and text colours
 aç shape and colour of buttons.

Here is an example of a finished design.

Improve the interface
If you have time you can improve the interface. In Matrix 1, Chapter 2, App Inventor, you learned how to upload an image. Use an uploaded image as the background image of the Layout box. This can produce a good interface design.

Perhaps you don’t want to make a football app? A student made an app to record a debate. She will press the button when she agrees with a comment. The app will count how many times she agrees.

Another student made an app to record every time a bird landed on her bird table. How would you design this interface?
Now you do it...

1. Decide what your app will be used for. For example, you might use it for football goals or for debating points.
2. Create an interface for your app.
3. Set the object properties to make the perfect interface design for your users.

If you have time...

At the moment the interface has a count button and a counter label.

When the app is working, the user will press the count button when a goal is scored. The counter label will show the number of goals.

Adapt the interface so there are TWO count buttons and TWO counter labels.

Use these names for the buttons and labels:

- Team1Button
- Team2Button
- Team1Label
- Team2Label.

Now the interface lets the user make a different goal count for each team in the match. This method will also work to count agreement in a two-person debate.

Test yourself...

1. An interface is used for two purposes: what are they?
2. Describe the different objects you added to the interface.
3. You changed object properties. What properties did you change?
4. Explain the features that make your interface easy to use.

Key words

**Interface:** An interface means all the features that let a user work with an app. The interface includes input and output.

**User:** The user is the person who will use the app. The user inputs data and looks at the output.
**Learning outcomes**

You have created an interface for your app. In this lesson you will add code to the interface. When you click the button on the interface, the counter will count up by one.

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

- make an app that records and counts events
- use arithmetic operators to change values.

**Learn about...**

App Inventor is an event-driven programming language. The code you write is always linked to an event. The event is a trigger. When the trigger happens, the code is carried out.

Before you start work, think about what the code needs to do. This is called a program plan.

1. The code is triggered when the user touches Team1Button
2. The value in Team1Label starts at 0. This number will go up by 1.

**Arithmetic operators**

In programming, calculations are carried out by arithmetic operators. The main arithmetic operators are:

- `+` plus
- `-` minus
- `×` multiply
- `/` divide.

App Inventor uses blocks to make program code. The dark blue Math blocks have arithmetic operators on them. You will use the plus sign in this lesson.

**FACT**

**Math versus Maths**

App Inventor was made in the USA so it uses American terms. For example, it uses ‘Math’ instead of ‘Maths’.
How to...

You are going to add code to your app. Open the Blocks screen. On the left of the Blocks screen is a menu. The coloured squares on the Blocks menu show the main types of block. In this lesson you will use the dark blue Math blocks.

Under the Blocks menu is a list of the objects from Screen1. There are blocks that go with each object. You will use these blocks in your code.

Start making code

At the moment, Team1Label shows 0. That is the Text property of Team1Label. You set the Text property to 0 when you made the interface.

Remember your program plan.

➔ When the user clicks Team1Button, the number in Team1Label will increase by 1.

Now you will make code to do that.

➔ Find Team1Button in the Blocks menu
➔ Select the block which says when Team1Button.Click
➔ Find Team1Label in the Blocks menu
➔ Select the block which says set Team1Label.Text to

The two blocks fit together like this.

This code means, “When the user clicks Team1Button, set the text of Team1Label to...”

There is one warning on the screen. The warning tells you there may be a syntax error in your program. There is a warning because the code is not finished yet. There is a gap for more blocks to be added.
Add one
On the left-hand side of the screen find the dark blue Math category.

- Find the block with the + (plus) sign
- Fit it into place as shown
The warning sign has not disappeared. The code is not finished yet. There are gaps to be filled.

Choose the values
The dark blue Math block with a plus sign is used to add together two values. There are two spaces in the block. You can put in blocks to show the two values. The two values are the number in Team1Label plus one.

- Find Team1Label on the left-hand side
- Select the block which says set Team1Label.Text to
- Put this block into the first gap
- Look in the dark blue Math blocks
- Find the block with 0 on it
- Put this block into the second gap and change the value to 1
The finished code looks like this:

This code means, “When the user clicks Team1Button, add one to Team1Label.” There are no warning signs because the code has no errors.

Reset button
The job of the Reset button is to return the value of Team1Label to 0. Here is what the code will look like. There is no need to use an arithmetic operator.

Now you do it...
- Make the code to add one to the counter when you click the button.
- Run the app to test that it works properly.
If you have time...

If you have not already done so, extend the interface so it has two buttons and two labels. Call the second label Team2Label. See Lesson 2.1 for details.

Now add code to both the buttons on your interface.

Here is a time-saving tip: Make the code for Team1Button. Right-click the code block and duplicate it. Use the drop-down menus to change to Team2Button and Team2Label.

Remember to add code so that the Reset button will set both labels to 0.

Test yourself...

1. What are the four main arithmetic operators?
2. What is meant by the text property of a label?
3. Clicking Team1Button is a trigger. What does ‘trigger’ mean?
4. Explain what happens in your program when the user clicks each button.

Key words

Arithmetic operator: An arithmetic operator is a symbol that makes the computer carry out a calculation.

Syntax error: A syntax error is a mistake in a computer language. Code with a syntax error cannot run. There is usually a warning message to tell you about syntax errors.

Trigger: A trigger is an event that makes the computer carry out some program code.
Learning outcomes

You have created an app. The app counts up the goals scored in a football match. In this lesson you will add an extra feature so that your app will count the goals for two teams and add them together to give a total.

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

- initialise a variable
- set the value of a variable
- get the value of a variable.

Learn about...

Program plan

If you have not done so already, you can expand your app so that it counts the goals for two different teams. You can add a new button to the interface. When the user clicks this button:

- the computer will add the scores together to give the total match score
- the computer will display the total score.

Variables

A variable is a location in computer memory. The memory location is given a name. You can set the value of the memory location. You can get the stored value from the memory location.

Variables can be local or global. What does this mean?

- **Local variables** can only be used inside a single block of code.
- **Global variables** can be used anywhere in your code.

In this chapter you will use global variables.

Initialise a variable

Making a variable is called declaring or initialising the variable. In App Inventor ‘initialize’ and ‘initializing’ are spelled with a ‘z’. App Inventor uses American spellings. When you initialise a variable you:

- choose a name for the variable
- choose whether it is a local or global variable
- set a starting value for the variable.

In this lesson you will initialise a global variable called **TotalScore**. The variable will store the total match score.
How to...

Prepare the interface

Make sure your interface is set up to record scores for two teams. In Lessons 2.1 and 2.2 this was included as the ‘If you have time...’ activity. If you have not done these activities yet, your teacher will help you to complete the task.

When you have done this, look at your interface:

- Add a new button to the interface
- Rename it StatsButton
- Set the Text property to Show total
- Add a new label to the interface
- Rename the new label TotalLabel
- Set the Text property to 0

If you can’t remember how to do this, look back at Matrix 1, Chapter 2, App Inventor, for help.

In the ‘Goal Counter’ example you can see we have made the Stats button and the Reset button very wide and set the colours to green and yellow. You can use any design you like.

Variable blocks

Look at the list of blocks. One of the blocks in the list is called Variables. When you select the orange Variables item, you will see these blocks.

- initialize: make the variable, give it a name, and set the starting value. Remember that the App Inventor block, initialize global, is spelled with a ‘z’. App Inventor uses American spellings.
- set: put a new value into the variable.
- get: get the value from the variable and use it in the program.

Initialise variable

You will make a variable to store the total score. At the start of the match the total is zero.

- Drag the orange initialize global block onto the Viewer
- Type the name TotalScore in the block
- Add the dark blue Math block 0

This code initialises the variable TotalScore and sets the starting value to 0. An initialise block is carried out when you start the app.
**set variable**
The computer will add the scores for Team 1 and Team 2. The result is the total score. The computer will add the scores when the user clicks StatsButton. You will now set the value of the variable TotalScore.

- Find the block when StatsButton.Click
- Find the orange set Variables block and put it inside this block

If you click the small arrow, you will see a drop-down menu of variables. You have only made one variable so far.

- Choose the variable TotalScore

TotalScore is calculated by adding together the scores of Team1 and Team2.

- Find the dark blue Math block with the + operator. The Math block has two gaps.
- Add it to the set Variables block
- Put the text values from Team1Label and Team2Label into the gaps as shown

Here is the completed code block.

**get variable**
You have set the value of TotalScore. Now you will get the value of TotalScore. You will display the value on the interface. You will set the text of TotalLabel.

- Find the block that sets the Text value of TotalLabel
- Drag this block into the Viewer. The block fits into the code underneath the set block you just made.
The code looks like this:

This code sets the text value of TotalLabel. Now you will set it to display the variable TotalScore.

To get the value of the variable you use the get block.

- Drag the orange get Variables block onto the Viewer
- Choose the variable name from the drop-down menu

Here is the complete code you made this lesson:

You have initialised a variable, and used set and get to work with the variable.

**Now you do it...**

- Make all the code to add together the goals scored by the two teams and give the total.
- Run the app to check that it works. The app should show the total score of the game by adding the scores of the two teams.

**If you have time...**

- Add code to the Reset button so that it sets the value of TotalLabel to 0.

**Test yourself...**

1. How would you pick a good name for a variable?
2. What is the difference between a global variable and a local variable?
3. What happens when you initialise a variable?
4. What happens when you set the value of a variable?
5. What happens when you get the value of a variable?

**Key words**

**Global variable**: A global variable is a variable that can be used anywhere in your code.

**Initialise**: Initialise means to create a variable, giving it a name and a starting value.

**Local variable**: A local variable is a variable that can only be used inside a single block of code.

**Variable**: A variable is a location in computer memory which is given a name.