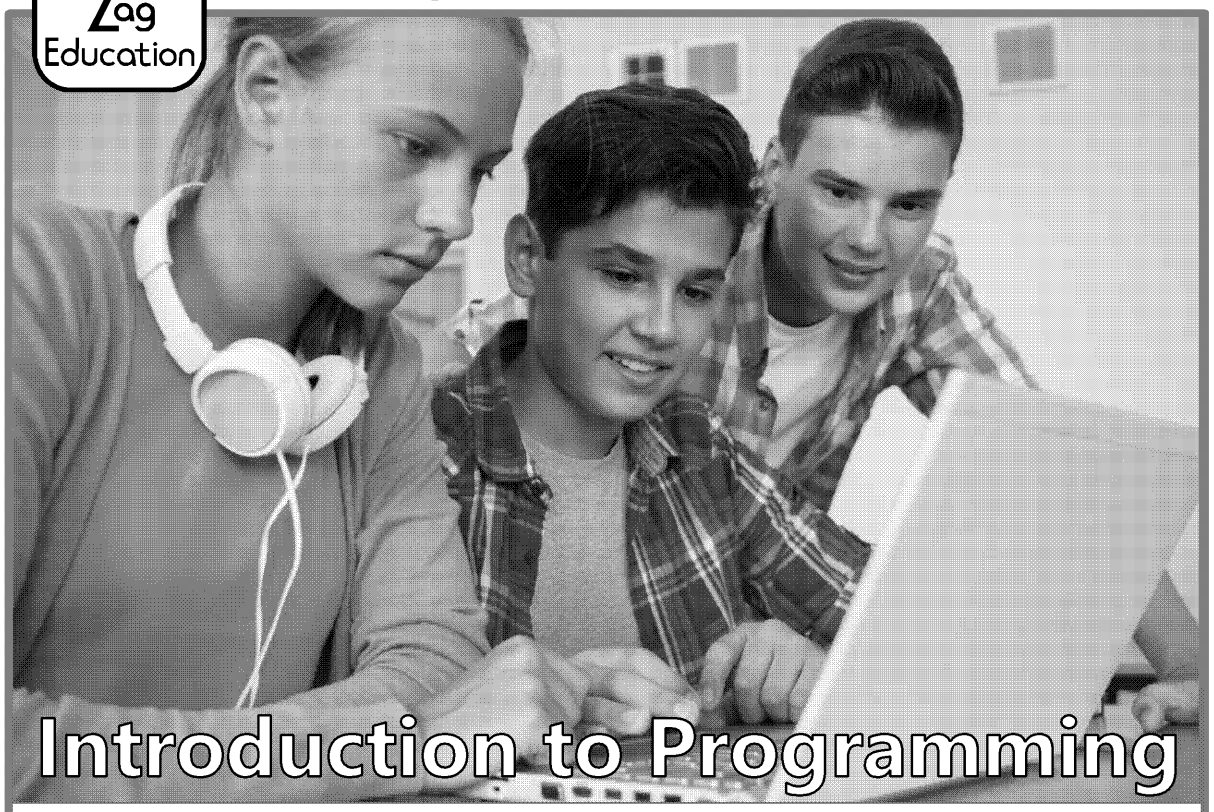
Zig Zag Education

Computing



Introduction to Programming

in Small Basic

CK12/ 7618

POD 7618

zigzageducation.co.uk

Contents

Teacher's Introduction

This resource includes a student booklet with seven lessons and a homework, along with matching PowerPoints. All Small Basic code that is used is also included so that the teacher can use it for demonstration.

The Booklet
The booklet is a combination of theoretical computer science knowledge as well as structured activities to test the ideas presented. The intention is for each student to have a personal copy of the booklet so that they can use it to complete the activities. This will serve as a good revision source. The booklet is written so that each student can work their way through it at their own pace.

Students are encouraged to write the Small Basic code and modify it where possible, wherever there is a ***CODE IT>>>*** icon.

The PowerPoint
The PowerPoint matches the sequence of lessons in the booklet. The same images, codes and labels are used so that students can see the link between what they are seeing on the display (from the PowerPoint) and what they are reading in the book.

The Small Basic Code
A zip folder containing Small Basic (.sb) files for each lesson can be downloaded from **http://zzed.uk/7618-files.**

Pedagogy
The PowerPoint and the booklet are very flexible and cater for a variety of teaching styles. The following methods can be used with these resources:

- The teacher introduces the main ideas (e.g. loops), demonstrates using the prepared Small Basic files, and allows students to program the ***CODE IT>>>*** and feed back what they have learnt.
- Students can program examples from the booklet then compare codes.
- Students can be involved in code review (students demonstrate and explain their code to other students, and feedback is given).
- Paired programming is also a useful tool.

The .sb File Format
Small Basic saves its source code in a .sb file format. This is same file format that MIT Scratch saves its program with. If students double-click on the file, it might open MIT Scratch, if this program is installed on the computer. The solution is to open Microsoft Small Basic IDE then open the file by clicking on 'Open' and selecting the location in which the file is saved.

KS3 Computin

Introduction to Programming in

Name: .....

Form: .....

Contents

Zig Zag Education

L1. Introduction to Small B

In this lesson you will understand:

- **the Small Basic environment**
- **how to use variables and data types in Small Basic**

The Small Basic Environment

This booklet teaches programming using the **Microsoft Small Basic** environment in this unit. This is called an **integrated development** The IDE allows us to write, test and execute your Small Basic program

Getting Small Basic Home

Go to **http://smallbasic.com/** Click download, then install the software on your computer. Note that Small Basic requires the Windows operating system

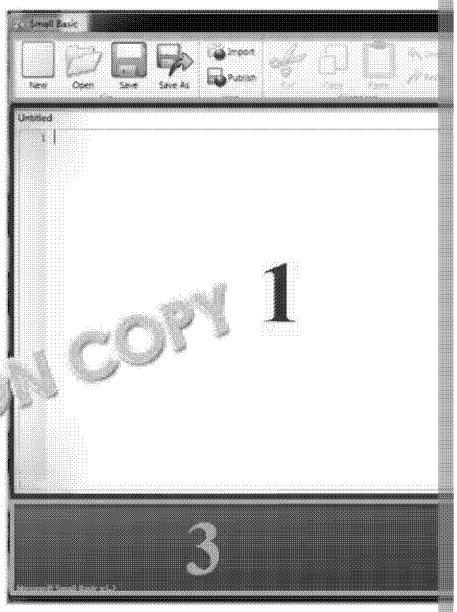
Key Definitions

An **algorithm** is a precise set of instructions for solving a specific problem

A **computer program** is a set of instructions that the computer can understand. A computer program is an algorithm written in a particular programming language. This is the programming language that we use.

Recall
What programming languages have you used before? List two other

The Small Basic Environment
This is where you will write all of your Small Basic programs. The different sections are labelled on the following page.



1. **The editor:** This is where you will write your Small Basic programs. The program code for previously written Small Basic programs can also be opened here.

   You can work on more than one Small Basic program at a time. Each program will be opened as a separate editor.

   The program that you are currently working on is called the *active*

2. **The toolbar:** The toolbar is used to issue commands to the active programming environment. This is where we will create and run

3. **The surface:** This is where the editors are displayed.

Test and Run a Program

Type the following code:

CODE IT >>>

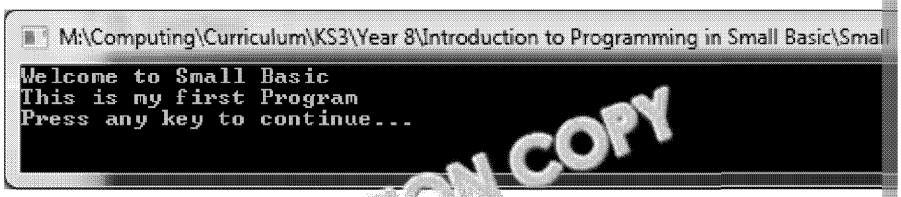
```
TextWindow.WriteLine("Welcome to Small Basic")
TextWindow.WriteLine("This is my first Program")
```

Save your program in your **Introduction to Programming in Small Ba**

Your program should look like this:

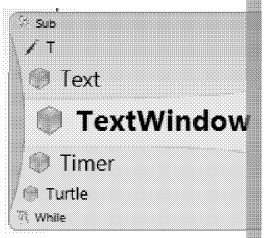
Run the program

Run the program by clicking the **Run** button on the toolbar, or by pre
F5, on the keyboard. You should see this output:



Intelli[illegible] Making It Easy

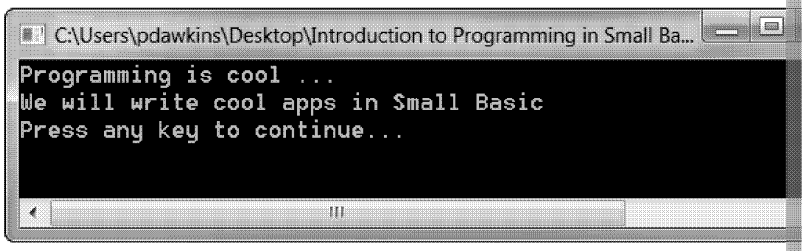
As yo[illegible] in the editor you perhaps notice a pop-up menu with options displaying possible commands. This is called 'IntelliSense'. You can use the up/down arrow key to select the command. You will also notice that there is an explanation of the highlighted command.



An IntelliSense list contains a list of commands that you can type.

Activity 1

1.1 Write a program in Small Basic that prints the following to the



1.2 Write a program in Small Basic to display a two-line message
Write the code in the box below

1.3 Type *TextWindow.* in the editor to start IntelliSense. Key up/d
Write down the help information that appears on the right of

WriteLine:

.....

.....

.....

.....

Data:

.....

.....

.....

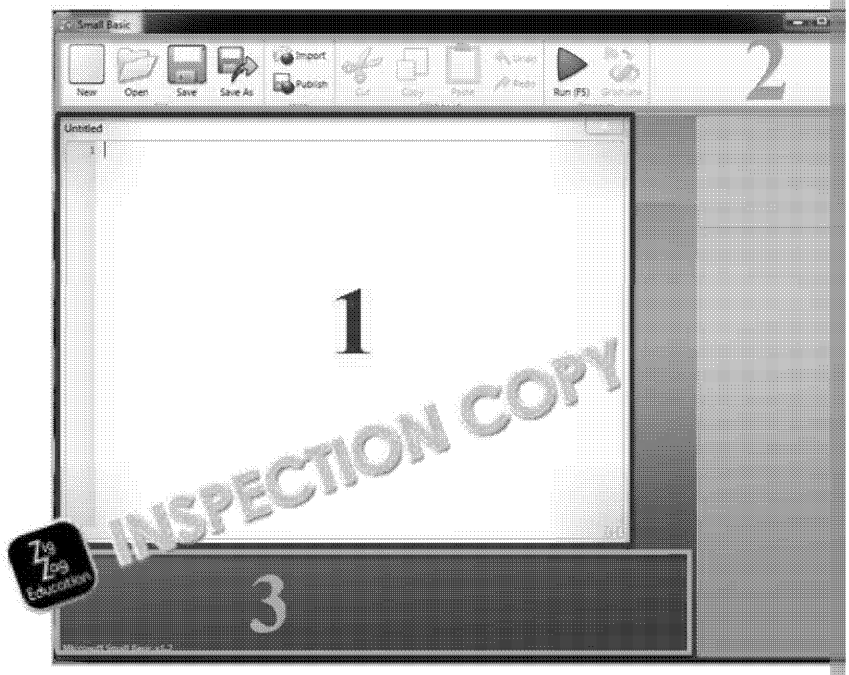
Returns:

.....

.....

.....

1.4 Label the different sections of the Small Basic environment.



1) .....

2) .....

3) .....

1.5 What does IDE stand for?

.....

1.6 Research and write down the names of **three** IDEs.

.....

.....

.....

TextWindow Properties and Operations

The first program consists of two statements. These are repeated be

```
TextWindow.WriteLine("Welcome to Small Basic")
TextWindow.WriteLine("This is my first Program")
```

The first statement writes the line 'Welcome to Small Basic' in the t
window is the black output window that is displayed when we run c

TextWindow is called an ***object***.

There are many objects in Small Basic. Each object has properties an

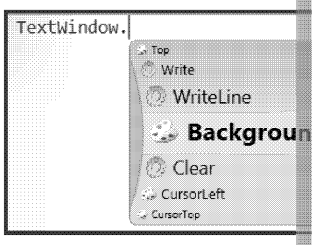
Properties: Data that the object knows about itself.

Operations: Actions that the object can perform.

Activity 2

2.1 Use IntelliSense to complete the table below.

Type *TextWindow.* in the editor.



Name	Property/Operation	
Top	Property	Gets or se th
Write		
WriteLine()		
BackgroundColor		
Clear()		
CursorLeft		
CursorTop		
ForegroundColor		
Hide()		
Left		
Pause()	Operation	Wait fo
PauseWithoutMessage()		
PauseIfVisible()		
Read()		
ReadNumber()		
Show()		
Title		

2.2 Type the following code.

CODE IT >>>

```
TextWindow.ForegroundColor = "Cyan"
TextWindow.WriteLine("Changing the text colou
```

Run the program and write your observation in the box below

Try r[illegible] 'Cyan' with the following colours:

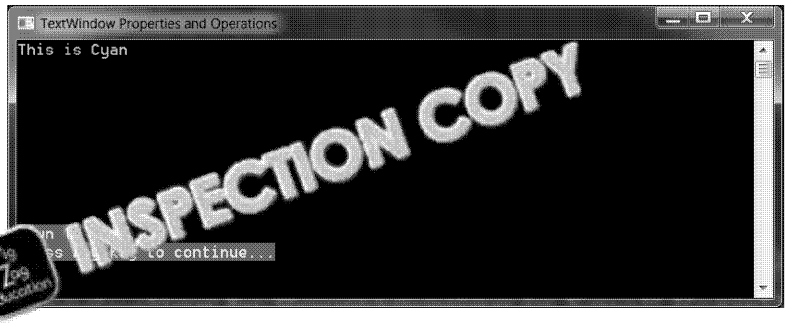
[illegible]ck	*Green*	*DarkBlue*
Blue	*Magenta*	*DarkCyan*
Yellow	*Red*	*DarkGray*
Gray	*White*	*DarkYellow*

3.1 Type the following code:

CODE IT >>>

```
'Exploring the properties/operation of TextWi
TextWindow.Title = "TextWindow Properties and
TextWindow.ForegroundColor = "Cyan"
TextWindow.WriteLine("This is Cyan")
TextWindow.ForegroundColor = "Yellow"
TextWindow.BackgroundColor = "Magenta"
TextWindow.CursorTop = 10
TextWindow.WriteLine("Down 10")
```

Output



Explain the output.

Write vs WriteLine

Write	Wri
Writes text or numbers to the text window. Unlike WriteLine, this will not append a new line character. This means that anything written to the text window after this call will be on the same line.	Writes text or numbe A new line character the output so that the is written to the text new line.

3.2 What is the difference in output between the following prog

Program 1

```
'Using the Write operation
TextWindow.Title = "The Write operation"
TextWindow.Write("Text Window has 7 propert
TextWindow.Write("and 10 operations")
```

Program 2

```
'Using the WriteLine operation
TextWindow.Title = "The WriteLine operation"
TextWindow.WriteLine("Text Window has 7 prope
TextWindow.WriteLine("and 10 operations")
```

Difference in output:

Reading Input from the U

The following pr your friend to enter their name, and the mess o end:

CODE IT >>>

```
' Reading input from the user
TextWindow.Title = "Reading input"
TextWindow.Write("What is your name --> ")
name = TextWindow.Read()
TextWindow.WriteLine("Hey " + name + " Small Basic
```

The two unfamiliar lines are:

```
name = TextWindow.Read()
TextWindow.WriteLine("Hey " + name + " Small Basic i
```

Small Basic will pause and wait until the user types some text and pr
the user types is then stored in the variable name.

A **variable** is a memory location that has a name. The value of this l
changed while the program is running.

Notice that in the line:

```
TextWindow.WriteLine("Hey " + name + " Small Basic i
```

We can print the value of a variable by including it in the WriteLine
the concatenation (+) to join the name with other text.

Rules for naming variables

1. A variable MUST begin with a letter of the alphabet, but canno
   keyword such as ***for***, ***while***, ***if***.
2. The name of the variable can contain letters, digits and unders

When choosing the name of the variable, it is worth using a name th
that will be stored; for example, school is a better variable name th
user's school.

Write your Question

Write a question for a program that your friend will write for you. Tr
TextWindow properties and operations discussed in this lesson.

To make your question more challenging, give your friend the output and ask them to write the code to

CODE IT >>> Example

- *Set the title of the text window to "TextWindow task"*
- *Set the top position of the text window to 100, and the left posi*
- *Set the top position of the cursor to 15, and the left position to*
- *Set the foreground colour of the text to "Yellow"*
- *Display the sentence "Life to its fullest"*

Comments

02 Reading input from the user.sb * - C:\Users\pdawkins\Desktop\Introduction to Progra

```
1 'Reading input from the user
2 TextWindow.Title = "Reading input"
3 TextWindow.Write("What is your name --> ")
4 name = TextWindow.Read()
5 TextWindow.WriteLine("Hey " + name + " Small Basic is g
6
```

Comments are statements that are written in the code but are not e
compiler. In Small Basic, comments must begin with an apostrophe (

The first line in the program code is a comment. Comments are use
purposes:

- give an explanation of what the programmer is thinking
- to explain a piece of code that is not very clear
- to provide additional information (contact information) to t

It is very good practice to include comments in your code. You must
between providing enough comment and over commenting your cod
is not good.

Rate your Understanding

Rate your understanding by placing a tick or a cross beside each outc

Outcomes
Write a Small Basic program with output.
Write a Small Basic program with input, output and variable.
Use IntelliSense to select the most appropriate property and operat to use with an object.

Prep

1) Download and install Small B[illegible] your computer at home. W you encounter duri[illegible]tion.

   To lov[illegible] Small Basic go to **http://smallbasic.com/**

   .....

   .....

   .....

   .....

2) Type and run the following program. Explain what the program

```
'Asking for information and printing a message
TextWindow.ForegroundColor = "Yellow"
TextWindow.Write("What is your name --> ")
name = TextWindow.Read()
TextWindow.Write(name + " how old are you --> ")
age = TextWindow.Read()
TextWindow.Write(name+ " what form are you in --> ")
form = TextWindow.Read()
TextWindow.WriteLine("")
TextWindow.WriteLine(name + " in form " +form+ " is
```

   .....

   .....

   .....

   .....

   .....

3) Circle the names of any variables used in the program.

4) Draw a square around any comment statements in the code ab

L2. Making Decisions

In this lesson you will understand:

- **how to write conditions**
- **how to use the IF statement**
- **how to use the IF-ELSE statement**

The IF Statement

The IF statement executes a statement if a condition is true.

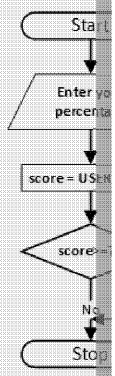
The syntax of the IF statement is:

IF** (condition) **Then
Statement
End

The program below asks the user to enter their end of Year 7 Quarte
message based on the result entered. If the user got 70% or more, t
'Well Done'.

CODE IT >>>

```
'Test result
TextWindow.Write("What percentage did you get in your test? ")
score = TextWindow.ReadNumber()

'If student scores 70% or more
If (score >= 70) Then
  TextWindow.WriteLine("Well Done")
EndIf
```



The condition in the IF statement is tested ***score >= 70*** and the m
condition is true, i.e. if score is either equal to or greater than 70.

The following program uses two IF statements to print a message ba
obtained in the Year 7 Quarter 4 assessment:

CODE IT >>>

```
'test result
TextWindow.Write("What percentage did you get in y
score = TextWindow.ReadNumber()

'If student score 70% or more
If (score >= 70) Then
  TextWindow.WriteLine("Well Done")
EndIf

'Now if the student score less than 70%
If (score < 70) Then
  TextWindow.WriteLine("Better luck next time")
EndIf
```

The IF-ELSE Statement

The IF-ELSE statement can be used to join the two IF statements on
If the condition is true, the first statement is executed, otherwise th
executed.

The syntax of the IF-ELSE statement is:

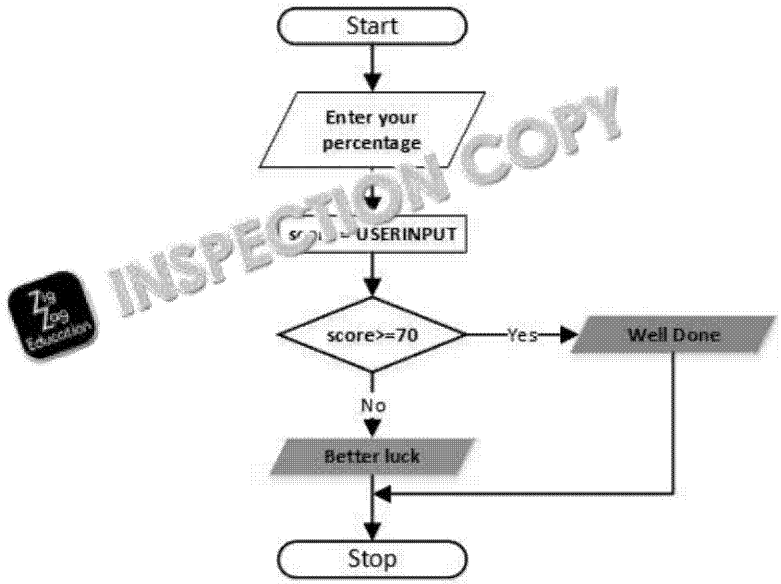
IF (*condition****) Then***
True Statement
Else
False Statement
EndIf

Note that after the IF-ELSE statement, one and only one (true or false
executed.

DE IT >>>

```
'test result
TextWindow.Write("What percentage did you get in y
score = TextWindow.ReadNumber()

'IF ELSE statement to print one statement or the o
If (score >= 70) Then
  TextWindow.WriteLine("Well Done")
Else
  TextWindow.WriteLine("Better luck next time")
EndIf
```

This following flow diagram shows the algorithm for the program abo
decision at least one of the output statements is executed (green or
This is different from the previous IF statement, where it is optional f
statements to be executed.



Worked Example

Check whether a number is odd or even.

An even number is a whole number that when divided by two gives a
odd number is a whole number that when divided by two gives a rem

For example, 6 is even because 6 ÷ 2 = 3 **remainder 0**, whereas 7 is o
7 ÷ 2 = 3 **remainder 1**.

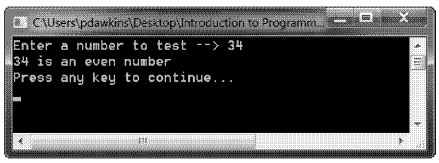
Write a program that asks the user for a number and tests whether th
even.

CODE IT >>>

```
'Test whether a number is odd or even
TextWindow.Write("Enter a number to test --> ")
number = TextWindow.ReadNumber()
remainder = Math.Remainder(number, 2)

'Now test whether the remainder is 0
If (remainder = 0) Then
  TextWindow.WriteLine(number + " is an even numbe
Else
  TextWindow.WriteLine(number + " is an odd number
EndIf
```

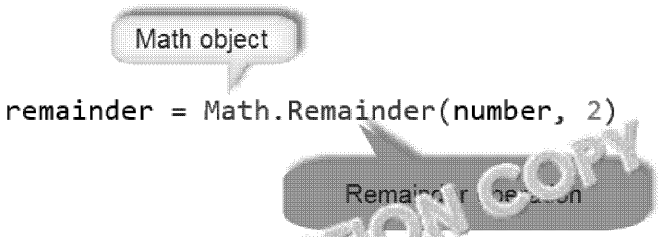
Two output

Test with 34 (even)



Test with

In the program above, you encounter a new object, namely the **Math**
the *Remainder(dividend, divisor)* operation.



The remain
will divid
by the div
the remain

The remain
stored in
called rem

Task IntelliSense to list the only property of the Math object an
and explain what the operations do.

Math property

.....

.....

.....

Math operation

1) .....

.....

.....

2) .....

.....

.....

Activity 4

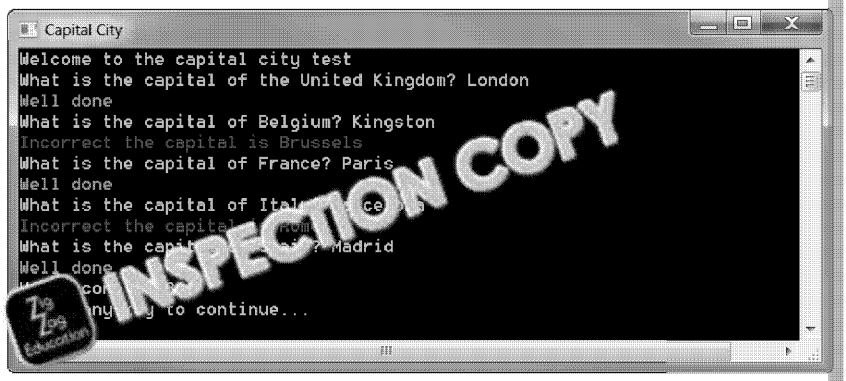
4.1 Write a program that tests students' knowledge of the following
The program should print a score out of five after the test.

Country	Capital City
the United Kingdom	London
Belgium	Brussels
France	Paris
Italy	Rome
Spain	Madrid

The program should ask ***What is the capital of <country's nam***

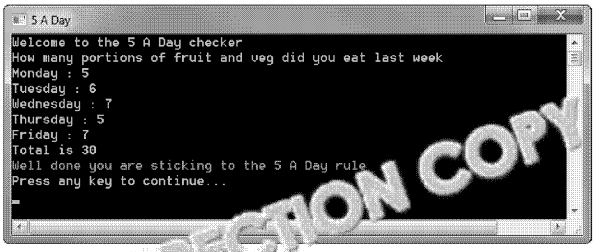
The score should start at zero; if the student's answer is correc
increase by one.

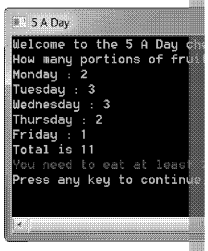
4.2 Modify the program above so that ***Incorrect*** is printed wheneve
entered. Use green to print the 'Well done' message and red t
message. An example is shown below.



4.3 As a part of its Live Well campaign, the NHS recommends that
fruit and vegetables per day. Write a program that asks the st
portions of fruit and vegetables they eat in a particular week, t
based on the average eaten for that week. If, on average, the
portions, the message should be ***Well done you are sticking to***

If the average is below 5, the program should calculate the sho
student ***You need to eat at least <calculated amount> more p***





Use the prep time to do research for your homework below.

Homework

Issue date: .....

Due date: .....

Homework – Small Basic

- This homework MUST be completed in the space provided belc
- You are expected to spend approximately 1 hour on this homev

1) Define the following programming terms:

Object: .....

.....

.....

.....

.....

.....

Variable: .....

.....

.....

.....

.....

.....

2) Question 2 relates to the program below.

```
'Addition test
TextWindow.Title = "Addition Test"
TextWindow.WriteLine("*** Welcome to your Addition quiz ***")
TextWindow.WriteLine("")
score = 0

For i=1 To 10
  'first get two random numbers
  num1 = Math.GetRandomNumber(10)
  num2 = Math.GetRandomNumber(20)

  'Print the question to the screen
  TextWindow.Write("What is " + num1 + " + " + num2 + " --> ")
  answer = TextWindow.ReadNumber()

  If answer = (num1 + num2) Then
    TextWindow.ForegroundColor = "Green"
    TextWindow.WriteLine("Well done")
    score = score + 1
    TextWindow.WriteLine("")
  Else
    correct = num1 + num2
    TextWindow.ForegroundColor = "Red"
    TextWindow.WriteLine("Wrong ... the Correct answer is " +
    TextWindow.WriteLine("")
  EndIf
  TextWindow.ForegroundColor = "Gray"
EndFor

TextWindow.WriteLine("Your total score is --> " + score)
```

Circle and annotate an example of each of the following prog

- comment
- operation
- variable
- property
- object

What will the program do when you run it? *You can type and test t*

.....

.....

.....

.....

.....

.....

.....

.....

.....

3) What is the difference between the following statements?

```
TextWindow.Write("This is a Write statement")
TextWindow.WriteLine("This is a WriteLine stat
```

.....

.....

.....

.....

.....

.....

.....

Assessment

Grade	Meaning	Teache
1	Outstanding	
2	Good	
3	Requires improvement	

Student comments to feedback:

L3. Repetition

In this lesson you will understand:

- **how to repeat using the FOR loop**
- **how to repeat using the WHILE loop**

The FOR Loop

The FOR loop is used to repeat a set of instructions a given number
for the FOR loop is:

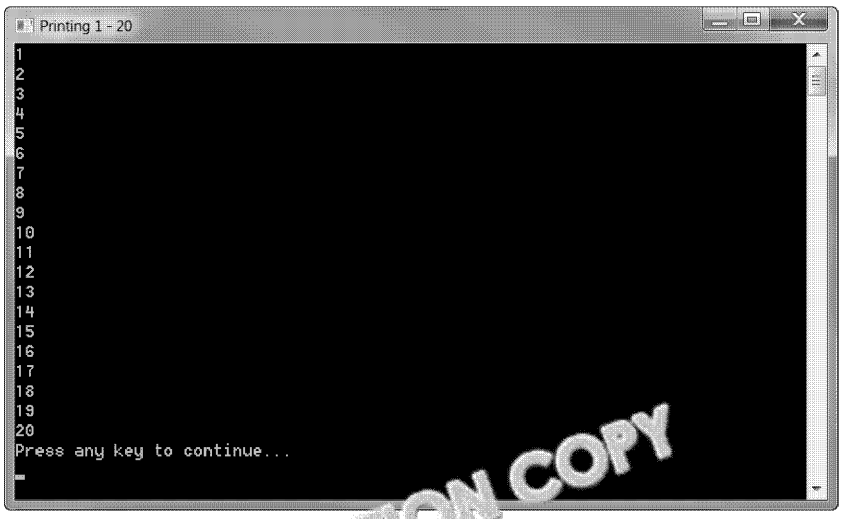
For *variable_name =* ***val*** *To* ***val2***
Statement to a
EndF

Suppose we want to print the numbers from 1 to 20, we could do thi

CODE IT >>>

```
'Printing 1 to 20
For i=1 To 20
  TextWindow.WriteLine(i)
EndFor
```

Output



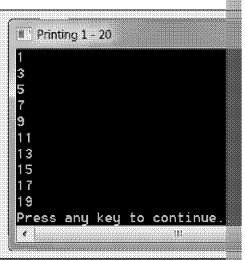
When the FOR loop is written, the initial and the end values are state
actually initialised at the starting value and incremented at the end o
that variable can be used in the loop; in the example above, the v
the body of the FOR loop.

If we wanted to go up by two (to print all the odd numbers from 1 to
the **step** modifier. The computer will increment the value of the vari
we iterate through the statement.

Suppose we want to print the numbers from 1 to 20, we could do thi

CODE IT >>>

```
'Printing 1 to 20
For i=1 To 20 Step 2
  TextWindow.WriteLine(i)
EndFor
```



The optional step keyword is used to specify the increment value tha the value by. If the step is missing, then a default value of 1 is used.

A negative value can be used in the step clause, allowing us to be abl variable to a value that is higher than the end value and count down example, the following program will count from 10 down to 1:

CODE IT >>>

```
'Printing 10 to 1
For i=10 To 1 Step -1
  TextWindow.WriteLine(i)
EndFor
```

Notice that without the step clause there would be no output given t larger than the end value.

Activity 5

5.1 Write a program that uses the FOR loop to print all of the even 20.

Example of the output

5.2 Write a similar program to Activity 5.1 above, this time countin

Example of the output

The WHILE Loop

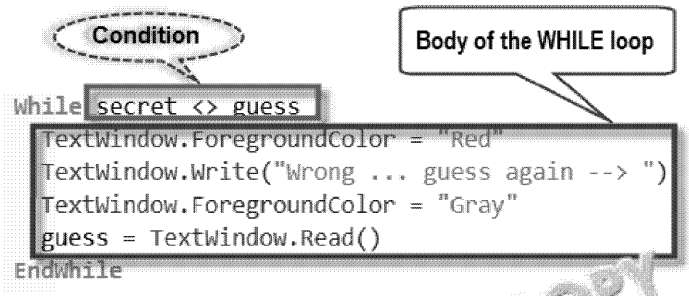
The WHILE loop is another way of repeating a set of instructions. T
repeated as long as a given condition is true. The WHILE loop is cal
loop because it is used whenever the loop count is not known befor
the WHILE loop is:

While *(condition)*
    Statement to execute
EndWhile

In the example below, the user is required to guess the password tha
The program will ask the user for the password until the correct pass

```
TextWindow.Title = "Password"
secret = "twyford"
TextWindow.Write("Guess the password --> ")
guess = TextWindow.Read()
While secret <> guess
  TextWindow.ForegroundColor = "Red"
  TextWindow.Write("Wrong ... guess again --> ")
  TextWindow.ForegroundColor = "Gray"
  guess = TextWindow.Read()
EndWhile
 TextWindow.WriteLine("Welcome ...")
```

The sections of the WHILE loop are labelled below.



As long as t
(secret <> g
of the WHIL
repeated.

It is importa
least one st
loop that ha
the conditio
example it i

Activity 6

6.1 How High / How Low

In the following program, the computer generates a random n
100. The user is required to guess this number. If the guess i
number, then the program prints ***too high***. The program will p
guess is lower than the number.

You are required to fill in the missing section of the code. A p
program is shown below.

Code to complete

```
1  'Number guessing game
2  TextWindow.Title = "Guessing game"
3  number = Math.GetRandomNumber(100)
4  TextWindow.Write("Guess a number between 1 and
5  guess = TextWindow.ReadNumber()
6
7  'Now test the number
8  While (number <> guess)
9    If number > guess Then
10
11
12
13   Else
14     TextWindow.ForegroundColor = "Red"
15     TextWindow.WriteLine("too high")
16     TextWindow.ForegroundColor = "Gray"
17   EndIf
18
19
20 EndWhile
21
22 TextWindow.ForegroundColor = "Gray"
23 TextWindow.WriteLine("Well done")
```

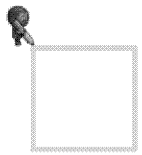
Print too low

take another guess

Example of output

```
Guessing game
Guess a number between 1 and 100 --> 5
too low
take another guess --> 87
too low
take another guess --> 81
too high
take another guess --> 78
too high
take another guess --> 76
too low
take another guess --> 77
Well done
Press any key to continue...
```

Improve your solutions by making them more efficient by u
below.

Outline	Instruction
	*Repeat 4 times do* *move forward by 100 pixels* *turn right by 90 degrees*
 Sides 150 pixels Angles – 60 and 120 degrees	
 Sides 100 pixels 60-degree turn	

Try some more on ***https://code.org/learn*** *Select* ***Artist****!*

L4. Graphics

In this lesson you will understand:

- **how to use GraphicsWindow properties and operations**

GraphicsWindow

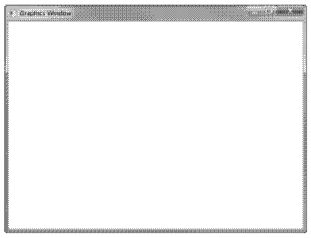
Small Basic provides an object called ***GraphicsWindow*** that has a se
operations defined over it to work with graphics.

TextWindow has properties and operations that use text and numb
has properties and operations that use graphics.

Type and run the following code to display the graphics window:

CODE IT >>>

```
'Showing the GraphicsWindow
GraphicsWindow.Title = "Graphics Window"
GraphicsWindow.Show()
```



When you run the program, a graphics win

The title is ***Graphics Window***, as set by the
operation just displays the graphics window

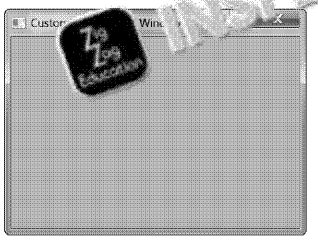
Click the red to close the window.

Customising the graphics window

Type and run the following code to customise the graphics window:

CODE IT >>>

```
'Customising the GraphicsWindow
GraphicsWindow.Title = "Customised Graphics Window
GraphicsWindow.BackgroundColor = "Plum"
GraphicsWindow.Width = 300
GraphicsWindow.Height = 200
GraphicsWindow.Show()
```



The graphics window is now customised by
height and background colour.

Drawing Lines

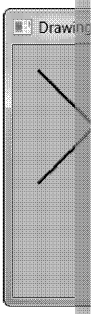
Now that we know how to customise and show the graphics window drawing some lines. Type and run the following code:

CODE IT >>>

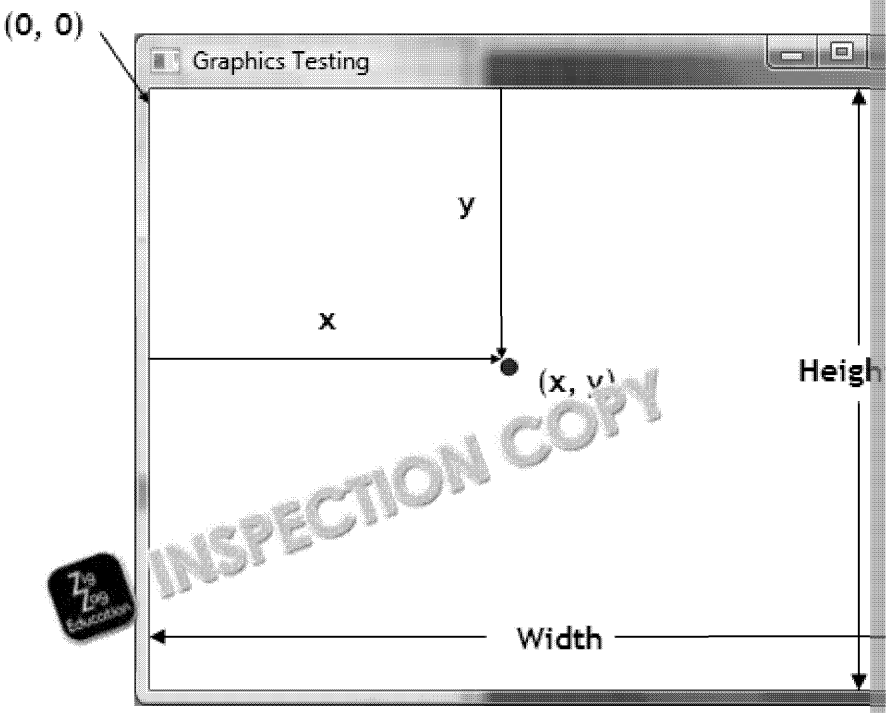
```
'Drawing lines on the GraphicsWindow
GraphicsWindow.Title = "Drawing lines - Graphics W
GraphicsWindow.BackgroundColor = "Plum"
GraphicsWindow.Width = 300
GraphicsWindow.Height = 200
GraphicsWindow.DrawLine(20, 20, 110, 110)
GraphicsWindow.DrawLine(20, 110, 110, 20)
GraphicsWindow.Show()
```

DrawLine(x1, y1, x2, y2) – will draw a straight line on the graphics window from point (x1, y1) to point (x2, y2).

In Small Basic, coordinate (0,0) starts at the top left corner of the window.

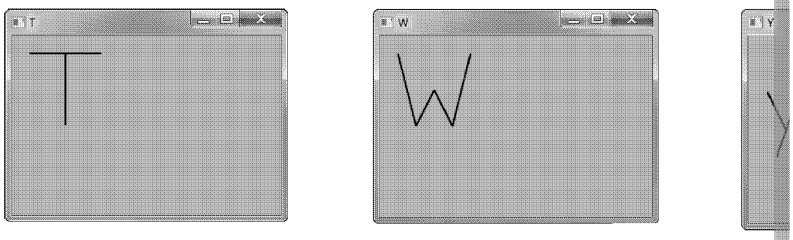


The following picture gives an outline of the coordinate system:

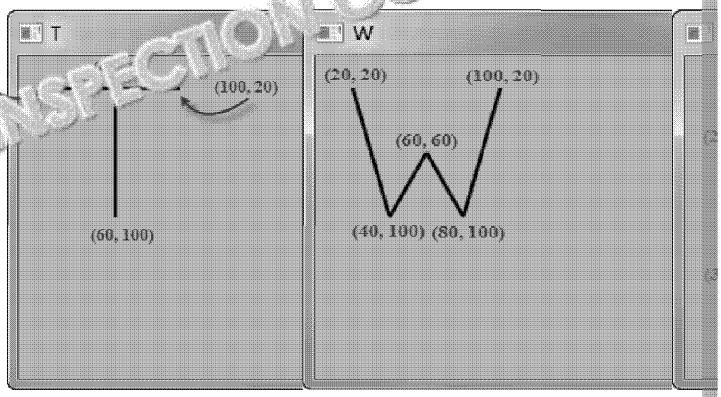


Activity 7

7.1 Write Small Basic code to produce the following output:



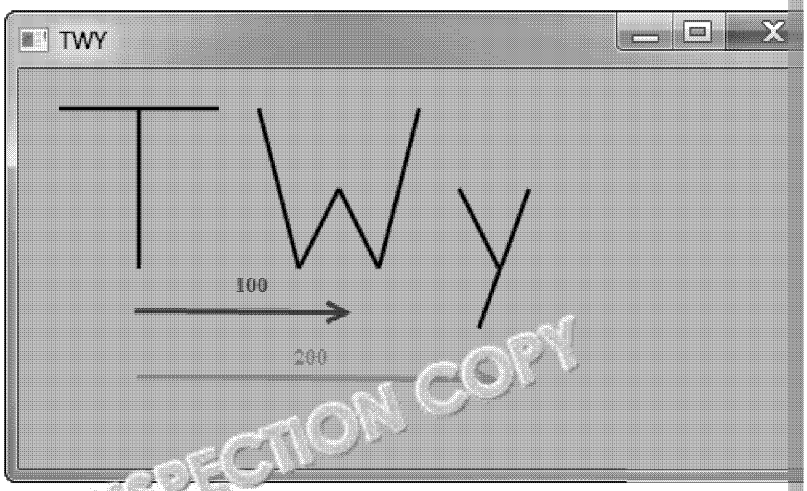
Use the following coordinates to [illegible]:



7.2 Combine **TWY** on the same graphics window.

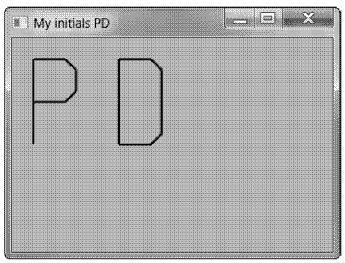
HINT: Think about each letter being translated 100 pixels from the previou[illegible] the x-values of W, and 200 to the x-values of Y.

Your output should look like the one below (without the arro[illegible]



7.2 [illegible]te your initials using lines.

HINT: Use a short, slanting line for curves. Here is an example:



Operations and Properties of GraphicsWindow

GraphicsWindow has a lot of operations and properties defined on it important definitions from Lesson 1 – Introduction to Small Basic.

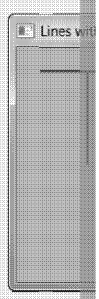
Object:	Individual thing that has properties and operations. W far – *TextWindow* and *GraphicsWindow*.
Properties:	Data that the object knows about itself.
Operations:	Actions that the object can perform.

Type `GraphicsWindow.` and use IntelliSense to find out what infor properties store and what actions the operations perform:

Name	Property/ Operation	Descriptio
BackgroundColor	Property	Gets or sets the background colour
PenColor		
PenWidth		
DrawRectangle()	Operation	Draws a rectangle on the screen
GetRandomColor()		
DrawRectangle()		
FillRectangle()		
DrawEllipse()		

CODE IT >>>

```
'Using PenColor
GraphicsWindow.Title = "Lines with colour"
GraphicsWindow.BackgroundColor = "Plum"
GraphicsWindow.Width = 400
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawLine(20, 20, 100, 20)
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(60, 20, 60, 100)
GraphicsWindow.Show()
```



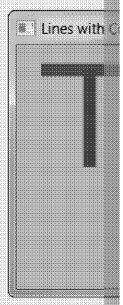
What is the name of the new property that is used in the code above

_____

Now modify the code to use the ***PenWidth*** property.

CODE IT >>>

```
'Using PenColor
GraphicsWindow.Title = "Lines with Colour"
GraphicsWindow.BackgroundColor = "Plum"
GraphicsWindow.Width = 300
GraphicsWindow.Height = 200

GraphicsWindow.PenWidth = 10
GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawLine(20, 0, 100, 20)
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(60, 20, 60, 100)
GraphicsWindow.Show()
```

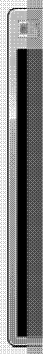


Using Loops

We can use loops to create some interesting patterns. Type the follo
the pattern below:

CODE IT >>>

```
'Using loops to create pattern
GraphicsWindow.Title = "Loops to create pattern"
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

'Set the pen colour
GraphicsWindow.PenColor = "Gold"

' Now Loop and change the pen width
For i=1 To 12
  GraphicsWindow.PenWidth = i
  GraphicsWindow.DrawLine(20, i*15, 180, i*15)
EndFor
```



Activity 8

8.1 Write Small Basic code to produce the following output:

Possible Approach

1) Set up the width, height and background colour
2) Loop with 12 iterations (FOR Loop 1-12)
3) Use the following to change the pen colour:

```
GraphicsWindow.PenColor =
GraphicsWindow.GetRandomColor()
```

8.2 Write small basic code to produce the following output:

Possible Approach

1) Set up the width, height and background colour
2) Loop with 10 iterations (FOR Loop 1-10), to draw the first half

```
GraphicsWindow.DrawLine(i*20, 20, 20, i*20)
```

3) Loop with 10 iterations (FOR Loop 10-1, step -1), to draw the second half

```
GraphicsWindow.DrawLine(i*20, 220, 220, i*20)
```

8.3 Write small basic code to produce the following output:

Possible Approach

1) Set up the width, height and background colour
2) Draw the blue lines from 8.2 above
3) Loop with 10 iterations (FOR Loop 1-10), to draw the first half

```
GraphicsWindow.DrawLine(20, 220-(i*20),
20 + (i*20), 220)
```

4) Loop with 10 iterations (FOR Loop 10-1, step -1), to draw the second half

```
GraphicsWindow.DrawLine(220-(i*20), 20, 220, 20 + (i
```

8.4 Use loops and DrawLine() to create a new pattern. Write in a

Possible Approach

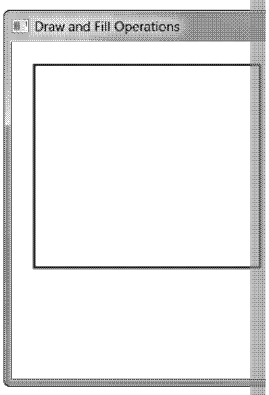
Shapes in GraphicsWindow

GraphicsWindow has several operations defined to draw common s
drawn by specifying the Draw method, in which case the outline is c
the Fill method.

Type and run the following code to draw two rectangles:

CODE IT >>>

```
Draw rectangles using Draw and Fill operations
GraphicsWindow.Title = "Draw and Fill Operations"
GraphicsWindow.Width = 600
GraphicsWindow.Height =

'Draw a rectangle using DrawRectangle
GraphicsWindow.PenColor = "Blue"
GraphicsWindow.DrawRectangle(20, 20, 200, 180)

'Draw a rectangle using the FillRectangle
GraphicsWindow.BrushColor = "Red"
GraphicsWindow.FillRectangle(250, 20, 100, 120)
```



The following code will draw two ellipses – one normal ellipse and th
that a circle is just an ellipse with the same height as width.

CODE IT >>>

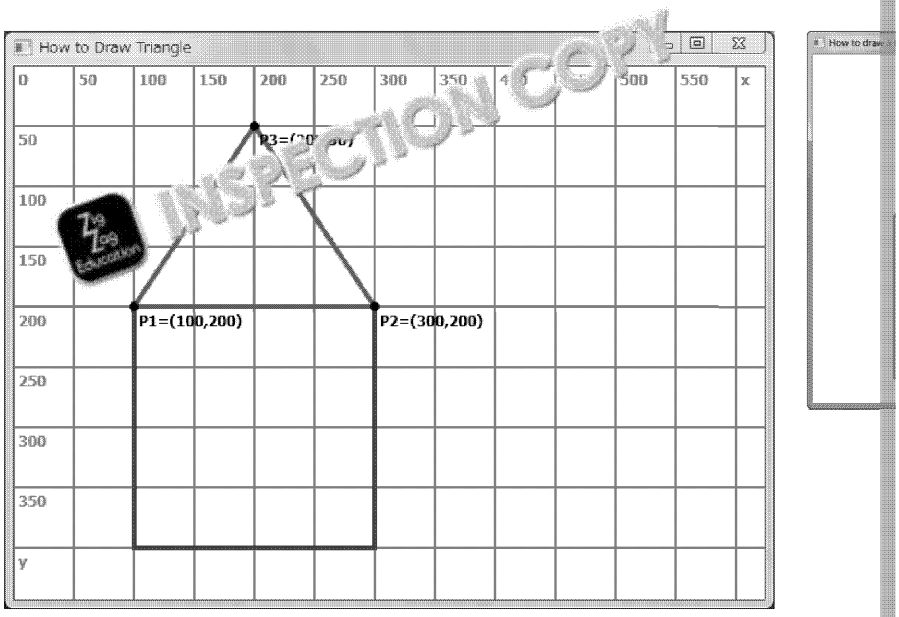
```
'Draw circles
GraphicsWindow.Title = "Draw and Fill Operations"
GraphicsWindow.Width = 600
GraphicsWindow.Height = 400

'Draw a normal ellipse
GraphicsWindow.DrawEllipse(10, 20, 400,100)

GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.FillEllipse(20, 140, 200, 200)
```

Activity 9

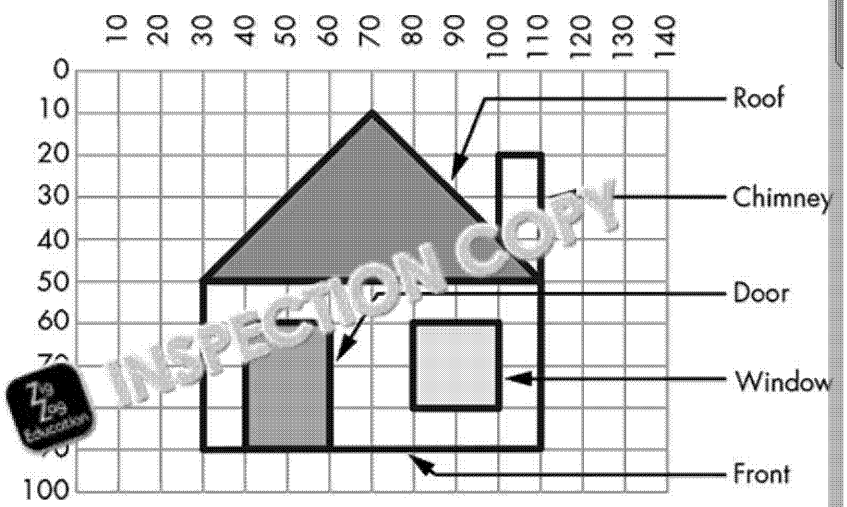
The following code uses the DrawTriangle() and DrawRectangle() ope
outline of the house below.

CODE IT >>>

```
' Draw the outline of a house
GraphicsWindow.Title = "How to draw a house"
GraphicsWindow.DrawTriangle(200, 50, 300, 200, 100
GraphicsWindow.DrawRectangle(100, 200, 200, 200)
```



9.1 Write Small Basic code to produce the following output:



HINT: DrawTriangle() will draw the outline, but FillTriangle() will fill
triangle. Use both DrawTriangle() and FillTriangle to achieve both t

Squared Pattern

The code below will create 20 squares of increasing size which form a

CODE IT >>>

```
'Squared Pattern with a Loop
GraphicsWindow.Title = "Squares"
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "Plum"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200

For i=1 To 100 Step 5
  GraphicsWindow.DrawRectangle(100-i, 100-i, i*2,
EndFor
```

Type the code and annotate it. In particular, annotate the *DrawRect* and ensure that you understand the purpose of each output to the o variable *i* is used to control the size of the square.

Activity 10

10.1 Modify the code above and use the DrawEllipse() operation to pattern. Write the code on the left-hand side of the box below

CODE IT >>>

Random Ellipse

Randomness is used in many programs to allow patterns and colours to in nature. The following program creates a pattern that uses *GetRand* random colour and *GetRandomNumber()* to select a random numbe

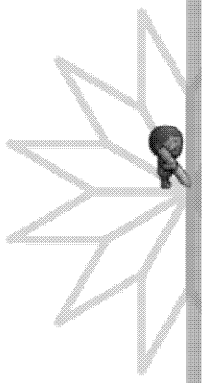
CODE IT >>>

```
' Randomness
GraphicsWindow.BackgroundColor = "Black"
For i=1 To 1000
  GraphicsWindow.BrushColor = GraphicsWindow.GetRa
  x = Math.GetRandomNumber(GraphicsWindow.Width)
  y = Math.GetRandomNumber(GraphicsWindow.Height)
  GraphicsWindow.FillEllipse(x, y, 10, 10)
EndFor
```

Note that *GetRandomNumber()* is defined in the *Math* object.

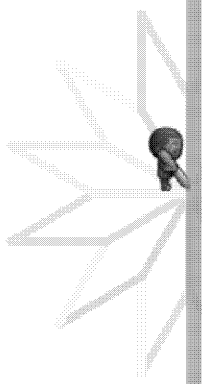
Prep

The following code creates the flower.

Code	Flow
when run repeat 12 times do move forward by 60 pixels turn right by 30 degrees move forward by 60 pixels turn right by 150 degrees move forward by 60 pixels turn right by 30 degrees move forward by 60 pixels	

Rev[illegible]he code to create a multicoloured flower. Use the block b[illegible]

set color | random color

Updated code	Multicolou[illegible]
	

Design a new pattern below and use the instructions to write code t[illegible]

Pattern	Code

L5. Turtle Graphics

In this lesson you will understand:

- **how to use the Turtle object**

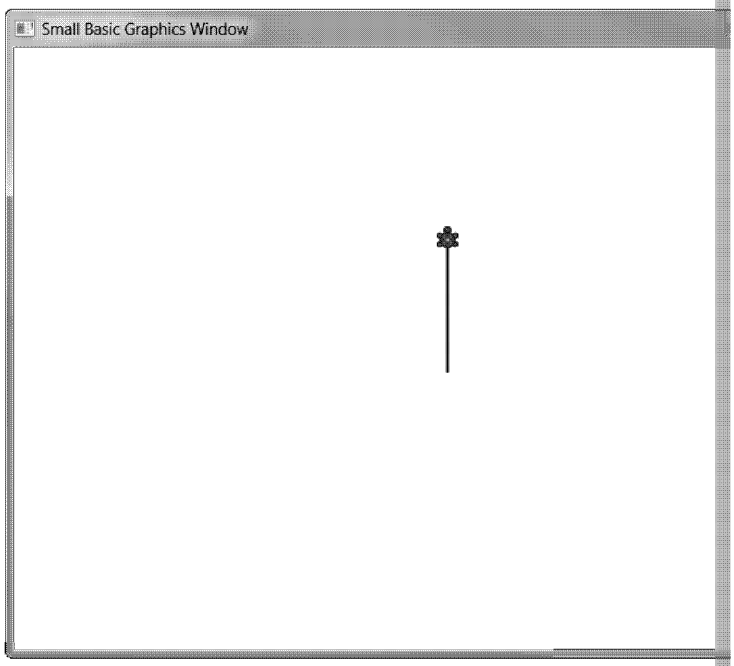
The Turtle

The third object that we will discuss is the Turtle object. The Turtle
properties and operations defined over it. The Turtle object is ofte

The first program simply starts a windo with the Turtle and moves

CODE IT >>

```
' Show and move the turtle
Turtle.Show()
Turtle.Move(100)
```



The `Show()` operation allows the Turtle to be visible. The `Move()` op
whole number as an input – 100 in the example above. This allows t
forwards 100 pixels.

Activity 11

Use IntelliSense to find the description for the following properties a
already done for you.

Turle Object

Turtle

The Turtle provides Logo-like functionality to draw shapes by manipulating the properties of a pen and drawing primitives.

Angle
Speed
X
Y
Hide
Move
Show
Turn
TurnLeft
TurnRight

Gets or sets the current angle of the Tu
will turn the Turtle instantly to the n

Drawing a Square

The following code will draw a square:

CODE IT >>> Ou

```
' Using Turtle to draw a square
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
```

From the code above it is clear that in drawing the square the Move
operations are called four times, which makes this code suitable as t
with four iterations. The following code will produce the same outpu

CODE IT >>>

```
'Draw square using FOR Loop
For  i=1 To 4
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```

We can use GraphicsWindow.PenColor to change the colour of the li
draws. The following programs draw the same square but using colo

CODE IT >>>

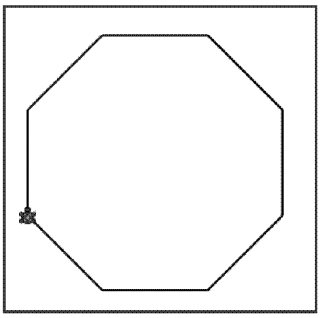
```
'Draw square using FOR Loop
For  i=1 To 4
  GraphicsWindow.PenColor = GraphicsWindow.GetRand
  Turtle.Move(100)
  Turtle.TurnRight()
EndFor
```

Drawing Other Shapes

We can draw an octagon (an eight sided polygon) by using the follow

CODE IT >>>

```
'drawing an octagon
For  i=1 To 8
  Turtle.Move(100)
  Turtle.Turn(45)
EndFor
```



Notice that there are eight iterations. We turn at an angle of 45 degr
between the sides is 45 degrees. With this information we can write
draw any polygon by changing a variable.

The program below will draw any polygon (including a square) by cha
sides to a whole number.

CODE IT >>>

```
'Drawing any polygon
sides = 12

length = 100 / sides
angle = 360 / sides

For i=1 To sides
  Turtle.Move(length)
  Turtle.Turn(angle)
EndFor
```

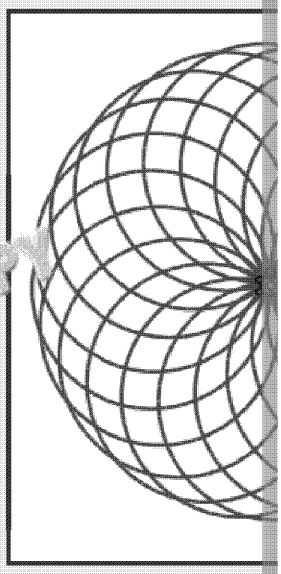
Note that when the value of sides is large enough (about 50 and abo
distinguish between the polygon and a circle because the pixels are li

Making Patterns

We can insert a loop into the previous example to create some intere
The program below draws a circle then turns 18 degrees and draws a

CODE IT >>>

```
'Drawing pattern
GraphicsWindow.Title = "Circle
pattern"
GraphicsWindow.PenColor = "Red"
sides = 50

length = 400 / sides
angle = 360 / sides

Turtle.Speed = 9
For i=1 To 20

  For i=1 To sides
    Turtle.Move(length)
    Turtle.Turn(angle)
  EndFor

  Turtle.Turn(18)
EndFor
```

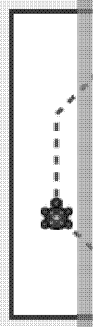


PenUp/PenDown

The PenUp() operation is used to allow the Turtle to move around wi
The PenDown() operation is used whenever we want the Turtle to co
The following program draws an octagon that uses the PenUp() and

CODE IT >>>

```
'PenUp and PenDown operations

GraphicsWindow.Title = "Penup and down"
GraphicsWindow.PenColor = "Red"

sides = 8
length = 400 / sides
angle = 360 / sides

For i=1 To sides
  For j=1 To 6
    Turtle.Move(length / 12)
    Turtle.PenUp()
    Turtle.Move(length / 12)
    Turtle.PenDown()
  EndFor
  Turtle.Turn(angle)
EndFor
```



There are two loops in this program. The inner loop with counter j d
and down six times and the outer loop draws the shape (an octagon

Activity 10

10.1 Write Small Basic code to produce the following output:

Possible Approach

1) A FOR loop to draw a line (PenUp and PenDown)
2) Loop with six iterations (FOR Loop 1-6)

10.2 Write Small Basic code to produce the following output:

Possible Approach

1) Draw the hexagon as above
2) Loop for 24 times and turn 15 degrees after each loop (24 × 15 = 360)

Possible Approach

1) Draw the hexagon as above (solid lines)
2) Loop for 24 times and turn 15 degrees after each loop (24 × 15 = 360)

Possible Approach

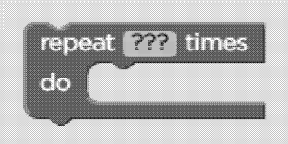
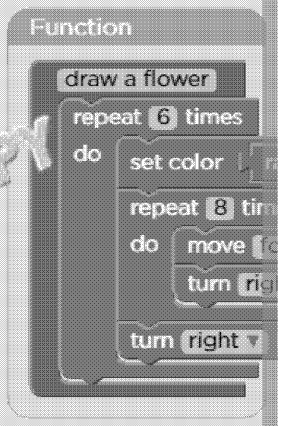
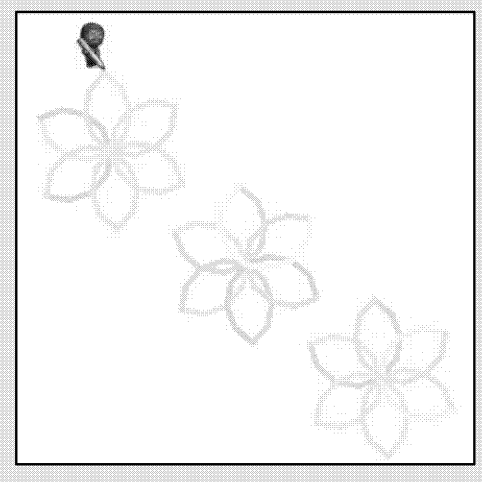
1) Draw the hexagon as above (solid lines)
2) Change the pen colour to a random colour
3) Loop for 24 times and turn 15 degrees after each loop (24 × 15 = 360)

Possible Approach

1) Draw the hexagon
2) Change the pen colour to a random colour
3) Loop for 24 times and turn 15 degrees after each loop (24 × 15 = 360)

Prep

A function is a programming tool to help you avoid repeating yours
below draws a flower, so you can use it any time you want to draw
a new jump block to draw these flowers. The flowers are 150 pixels

Instruction	**Function**
  	
Pattern	**Code to create the pa**
	

Explain an advantage of using functions in your code.

.....

.....

.....

.....

.....

L6. Subroutines

In this lesson you will understand:

- **how to use subroutines in a program**

Subroutines

A subroutine is a named section of a program that does a specific ta
many times.

We use subroutines for tasks that we will be performing frequently
example, if you are creating a geometry program and want to draw
frequently, you can use a subroutine to do this.

Ther wo aspects to subroutines: defining the subroutine and ca

Defining the subroutine

When defining a subroutine, the following syntax is used:

*Sub **SubroutineName***
Body of the subroutine
EndSub

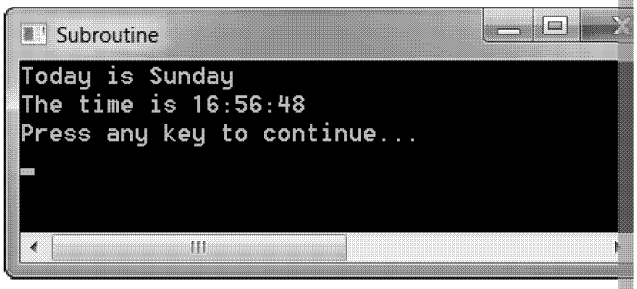
Calling the subroutine

When calling the subroutine, use the name of the subroutine followe

The following program defines a subroutine called displayDateInfo a
subroutine:

CODE IT >>>

```
'Defining the subroutine
Sub displayDateInfo
  TextWindow.Write("Today is ")
  TextWindow.WriteLine(Clock.WeekDay)
  TextWindow.Write("The time is ")
  TextWindow.WriteLine(Clock.Time)
EndSub

'******* The main program ***********
TextWindow.Title = "Subroutine"

'Calling the subroutine
displayDateInfo()
```



The image above shows an example of the output.

```
1  'defining the subroutine
2  Sub displayDateInfo
3    TextWindow.Write("Today is ")
4    TextWindow.WriteLine(Clock.WeekDay)
5    TextWindow.Write("The time is ")
6    TextWindow.WriteLine(Clock.Time)
7  EndSub
8
9  '******* The main program ***********
10 TextWindow.Title = "Subroutine"
11
12 'Calling the subroutine
13 displayDateInfo()
14
```

Lines 2–7 define the subroutine; the subroutine is called on line 13.

Advantages of Using Subroutines

Subroutines have the following advantages:

1. **Subroutines make your code shorter** as there is no need to repe subroutine if you want to repeat the task. In the example above date information in our program again, we simply call the subrou
2. **Subroutines can be used to break down large programs into sm manageable pieces**, and then put the smaller pieces together to problem. For example, if you have a large registration system to subroutines to collect students' information, add a student, colle add a course, or print information, and then put these subroutin complete registration system.
3. **Subroutines make your programs more readable**. When you cr names that state what the subroutine does. This is important fo when you revisit your code at a later date.

Subroutines and Variables

A subroutine can access variables that are used outside its body. The
defines a subroutine called largerNumber() that compares num1 and
larger value to a variable called 'larger'.

In the main section of the program, the user is prompted for two nu
The subroutine largerNumber() is then called and the larger value pr

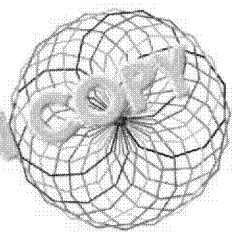
CODE IT >>>

```
'****** Subroutine **********
Sub largerNumber
  If num1 > num2 Then
    larger = num1
  Else
    larger = num2
  EndIf
EndSub


'****** Main Program ********
TextWindow.Title = "larger"
TextWindow.Write("Enter a number ")
num1 = TextWindow.ReadNumber()
TextWindow.Write("Enter a second number ")
num2 = TextWindow.ReadNumber()

'Call the subroutine largerNumber
largerNumber()
TextWindow.Write("The larger number is ")
TextWindow.WriteLine(larger)
```

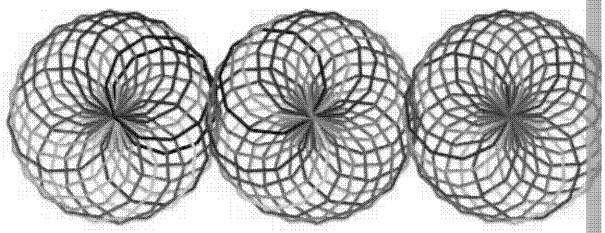
Notice that num1, num2 and larger are used inside and outside the s

Subroutines and GraphicsWindow

In the previous lesson you created a program to draw the picture bel



We c e subroutines to extend our program to draw the followin



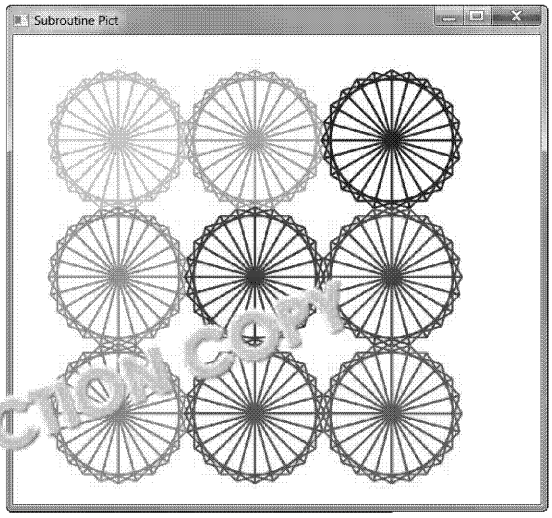
The following program uses a subroutine to create three graphics. N
subroutine called DrawPicture(). A further DrawPicture() is in a loop.

CODE IT >>>

```
'***** Subroutines ******
Sub DrawPicture
  Turtle.Speed = 10
  For j=1 To 24
    For i=1 To sides
      Turtle.Move(length)
      Turtle.Turn(angle)
    EndFor
    Turtle.Turn(15)
   GraphicsWi  PenColor = GraphicsWindow.GetRan
  EndFor
En

'***** Main Program  ******
TextWindow.Write("How many sides --> ")
sides = TextWindow.ReadNumber()
TextWindow.Hide()
GraphicsWindow.Title = "Subroutine Pict"
angle = 360/sides
length = 200 / sides
Turtle.X = 100
Turtle.Y = 100
For k=1 To 3
  DrawPicture()
  Turtle.PenUp()
  Turtle.X = Turtle.X + 120
  Turtle.PenDown()
EndFor
```

Activity 11

11.1 Write Small Basic code to produce the following output:



Possible Approach

1) Modify the code above by putting the FOR loop that is in the M
   FOR loop with three iterations.
2) Increase the y value by 130 pixels after each iteration:
   `Turtle.Y = Turtle.Y + 130`
3) Select a random colour after each DrawPicture.

Prep

Arrays are used in many programming languages. Research and fin

1) With the aid of a diagram, explain what an array is.

   .....

   .....

   .....

   .....

2) Give an example of a situation in which it is better to create an
   set of variables.

   .....

   .....

   .....

   .....

L7. Arrays

In this lesson you will understand:

- **how to use arrays**

What is an Array?

An array is a special kind of variable that can hold more than one va
piece of data stored in the array is called an element, and each elem
by using an index.

For example, if we want to st[illegible] five animals in a zoo, instead o
variables called
animal1, *animal2*, *animal3*, *animal4* and *animal5*,
we [illegible] a single variable called ***animals[]***, and then access the
inde[illegible]us ***animals[3]***.

The following program asks the user to enter five animals then use a
animals to the screen.

CODE IT >>>

```
'Arrays
TextWindow.Title = "Animal Array"
For i=1 To 5
  TextWindow.Write("Enter the name of animal " + i
  animals[i] = TextWindow.Read()
EndFor

TextWindow.WriteLine("")
'Now print the names back to the screen
For k=1 To 5
  TextWindow.WriteLine("Animal " + k + ": "+ anima
EndFor
```

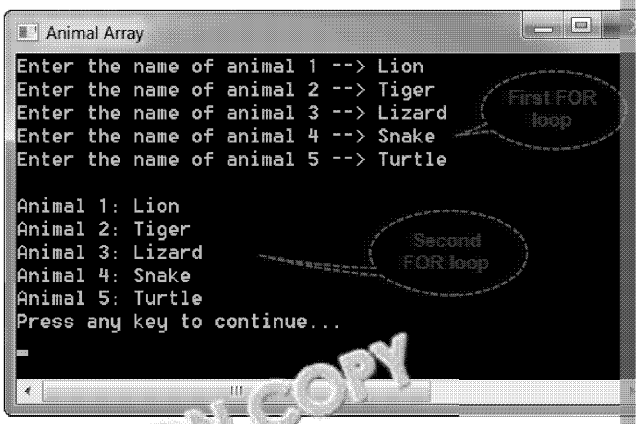
Notice that the variable i in the FOR loop is used to index the array.

```
animals[i] = TextWindow.Re[illegible]
```

has the effect of storing the na[illegible] the animals in positions

animals[1]
[illegible]
[illegible]mals[3]
animals[4]
animals[5]

The second FOR loop uses the variable k to print the values back to t



Activity 12

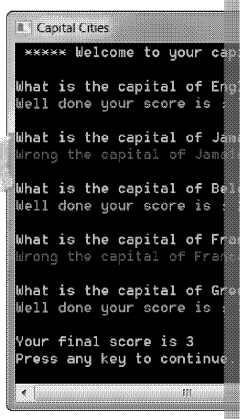
12.1 Write a program that asks for the username of five users and a password for each. The program should store these in two arrays called ***usernames*** and ***passwords***.

The program should then print each username and password next to each other on the screen. Print the username in red and the password in yellow.

12.2 Copy the code below to start your program. Extend the progra asks what the capital is of each country in the countries array t the user got the answer correct or not.

The program should print a final score at the end. The followin output from the program:

```
'Capital city
TextWindow.Title = "Capital Cities"
countries[1] = "England"
countries[2] = "Jamaica"
countries[3] = "Belgium"
countries[4] = "France"
countries[5] = "Gre

capit  = London"
c  [2] = "Kingston"
capitals[3] = "Brussels"
capitals[4] = "Paris"
capitals[5] = "Athens"

score = 0
TextWindow.WriteLine(" ***** Welcome to your capital
```



12.3 The table below shows a list of cities and the year that they hos

Year	Host City
2016	Rio
2012	London
2008	Beijing
2004	Athens
2000	Sydney

Write a program that creates two arrays called ***year*** and ***host***
should then create a quiz that asks what city hosted the Olym
year. The program should print a final score out of 5 at the en

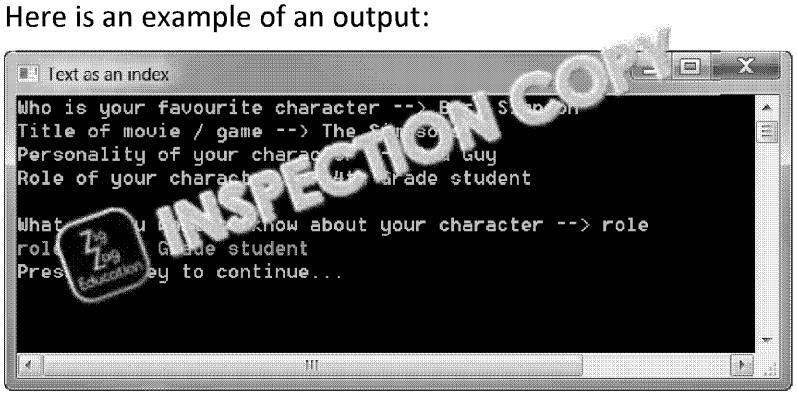
Using Text as an Index

In Small Basic, text can be used to index arrays. The following program
the user's favourite character in an array, and then prints back a reques

CODE IT >>>

```
'Text as index in an array
TextWindow.Title = "Text as an index"
TextWindow.Write("Who is your favourite character --
character["name"] = TextWindow.Read()
TextWindow.Write("Title of movie / game --> ")
character["title"] = TextWindow.Read()
TextWindow.Write("Personality of your character -->
character["personality"] = TextWindow.Read()
TextWindow.Write("Role of your character --> ")
character["role"] = TextWindow.Read()

'Now print some information to the screen
TextWindow.WriteLine("")
TextWindow.Write("What do you want to know about your
answer = TextWindow.Read()
TextWindow.ForegroundColor = "Green"
TextWindow.WriteLine(answer + " is " + character[ans
TextWindow.ForegroundColor = "Gray"
```

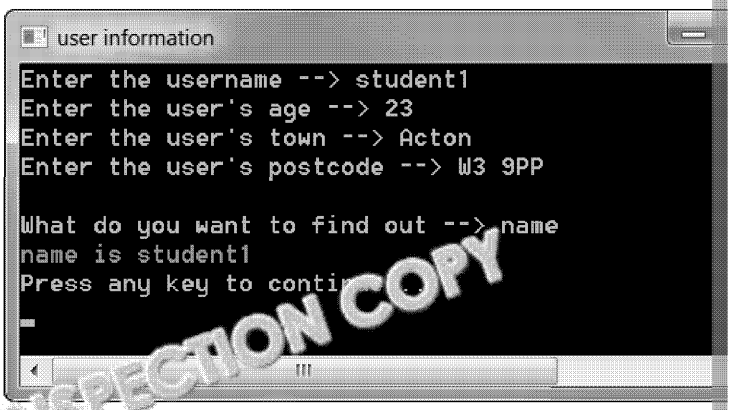
Here is an example of an output:



Notice that text rather than a number is used to index the array. Wh
index it is not case sensitive – array indices don't have to match exac
therefore, *"role"* and *"Role"* are both read as the same variable.

Activity 13

13.1 Write a program that uses an array called ***user*** that stores the
postcode of a user. The program should then ask what you wa
user and print this information to the screen. Here is an exam



Mu ne sional Arrays

Array have more than one index – these arrays are called multi
example, if you want to store information about all of your friends i
the specific information that you require, as you would do in the co
phone. It turns out that the first index could be your friend's nickna
could be the bit of information that you want to store. The following
this:

CODE IT >>>

```
'Friend's infomation
TextWindow.Title = "Friends information"
friends["Ben"]["name"] = "Benjamin"
friends["Ben"]["phone"] = "07830393438"

friends["Bob"]["name"] = "Robert"
friends["Bob"]["phone"] = "07783484206"

friends["T"]["name"] = "Tia"
friends["T"]["phone"] = "07663930383"

TextWindow.Write("Who do you want to know about -
nickname = TextWindow.Read()

'Now print the information to t  c  een
TextWindow.ForegroundColor = "G  en"
TextWindow.WriteLi  (c  ame + " name is " +
friends[nickn  "  ame"])
Text  ndo  W  teLine(nickname + " phone # is " +
fr  [nickname]["phone"])
Tex  ndow.ForegroundColor = "Gray"
```

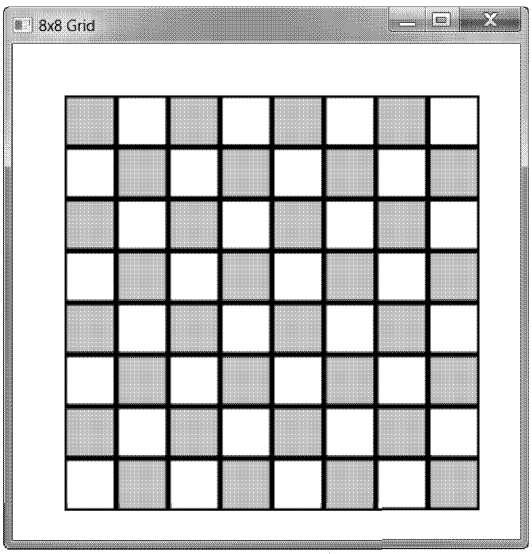
Here is an example of the output from the program:

The following program creates an 8 × 8 chess board.

CODE IT >>>

```
'Chess board
rows = 8
columns = 8
size = 40

GraphicsWindow.Title = "8x8 Grid"
For r=1 To rows
  For c=1 To columns
    If Math.Remainder(c,2) = Math.Remainder(r,2) Th
      GraphicsWindow.BrushColor = "LightGreen"
    Else
      GraphicsWindow.BrushColor = "White"
    EndIf
    boxes[r][c] = Shapes.AddRectangle(size, size)
    Shapes.Move(boxes[r][c], c * size, r * size)
  EndFor
EndFor
```

Output



Notice that we are using a multidimensional array called ***boxes[][]***. V
object and the AddRectangle() operation to add a square to the grid.

Useful Resources

1. Small Basic API reference – **http://smallbasic.com/doc/?id=8&languag**
2. Small Basic blog – **https://blogs.msdn.microsoft.com/smallbasic/**
3. The Hour of Code website – **https://code.org/**

My Glossary

Create a glossary of terms as you learn them.

Notes