

UNIT 2 – USING VB TO EXPAND OUR KNOWLEDGE OF PROGRAMMING

UNIT 2 – USING VB TO EXPAND OUR KNOWLEDGE OF PROGRAMMING	1
IMPORTANT PROGRAMMING TERMINOLOGY AND CONCEPTS	2
<i>Program</i>	2
<i>Programming Language</i>	2
<i>Code</i>	2
<i>Algorithm</i>	2
A COMPUTER AS A DATA PROCESSING MACHINE.....	2
A QUICK INTRODUCTION TO THE VISUAL BASIC DEVELOPMENT ENVIRONMENT.....	3
STARTING A NEW PROJECT AND MAKING SURE IT IS SAVED PROPERLY.....	3
THE MAIN FEATURES OF THE VISUAL STUDIO DEVELOPMENT ENVIRONMENT.....	4
INTRODUCTION TO VISUAL BASIC: ASTRONOMICAL CALCULATOR	5
QUESTIONS	5
PICTORIAL DESCRIPTION OF THE EXECUTION OF THE “ASTRONOMICAL CALCULATOR” PROGRAM.....	7
AN ACCELERATED INTRODUCTION TO VISUAL BASIC	8
VISUAL BASIC (VB FOR SHORT)	8
COMPARING THE MAIN STRUCTURES OF APP INVENTOR TO THE MAIN STRUCTURES OF VB	8
LOOPING STRUCTURES IN VB NOT AVAILABLE IN APP INVENTOR	11
UNDERSTANDING THE LEARNVB BY COMPARING TO APP INVENTOR PROGRAM.....	13
DATA TYPES	16
DATA (INFORMATION) – A PARTIAL LIST OF VB DATA TYPES	16
<i>Important Points about Data Types</i>	17
<i>Questions</i>	17

IMPORTANT PROGRAMMING TERMINOLOGY AND CONCEPTS

Program

- A **program** is a sequence of **instructions** that a computer can **interpret** and **execute**.

Programming Language

- A **programming language** is a very **precise** and **unambiguous** language that is designed to allow **instructions** to be given to a computer.

Code

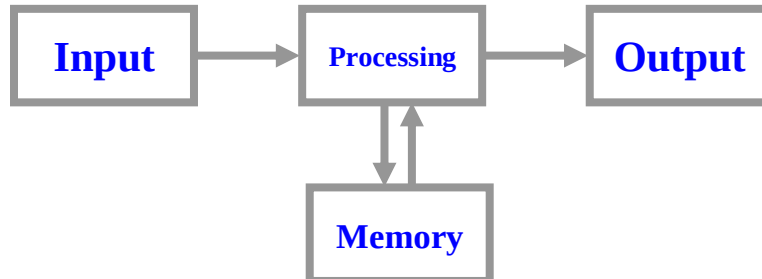
- Programming instructions are often called “code.” Programmers say that they are “writing code” when they write programs.

Algorithm

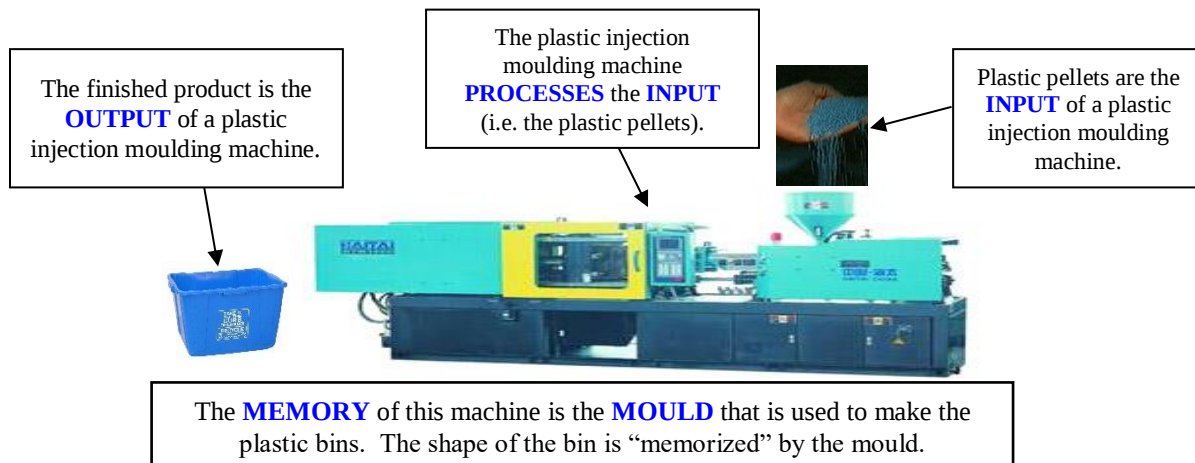
- An **algorithm** is a systematic procedure (finite series of steps) by which a problem is solved. Long division is an example of an algorithm that you learned in elementary school.
- In cooking, algorithms are called **recipes**.
- For many problems, there exist many different algorithms.
- For some problems, there are no known efficient algorithms (too slow and/or require too much memory).
e.g. What are the prime factors of a number?
- Some problems cannot be solved by a computer (**i.e.** no algorithm exists that can be implemented on a computer).

A Computer as a Data Processing Machine

A simple but very useful model of a computer is shown below. A computer can be viewed, at a very simple level, as a machine that **processes data** (information). As the diagram suggests, information is given to a computer, the information is then processed by the computer and finally, the results are displayed.



This process is similar to industrial processes such as **plastic injection moulding**. The diagram below shows the basic idea of how a plastic injection moulding machine produces its output.



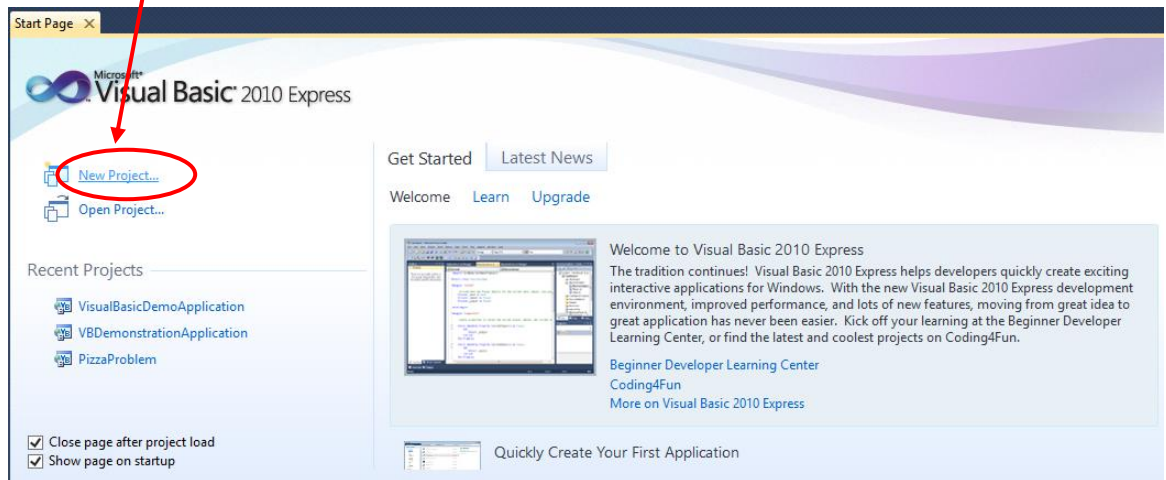
A QUICK INTRODUCTION TO THE VISUAL BASIC DEVELOPMENT ENVIRONMENT

Starting a New Project and Making Sure it is Saved Properly

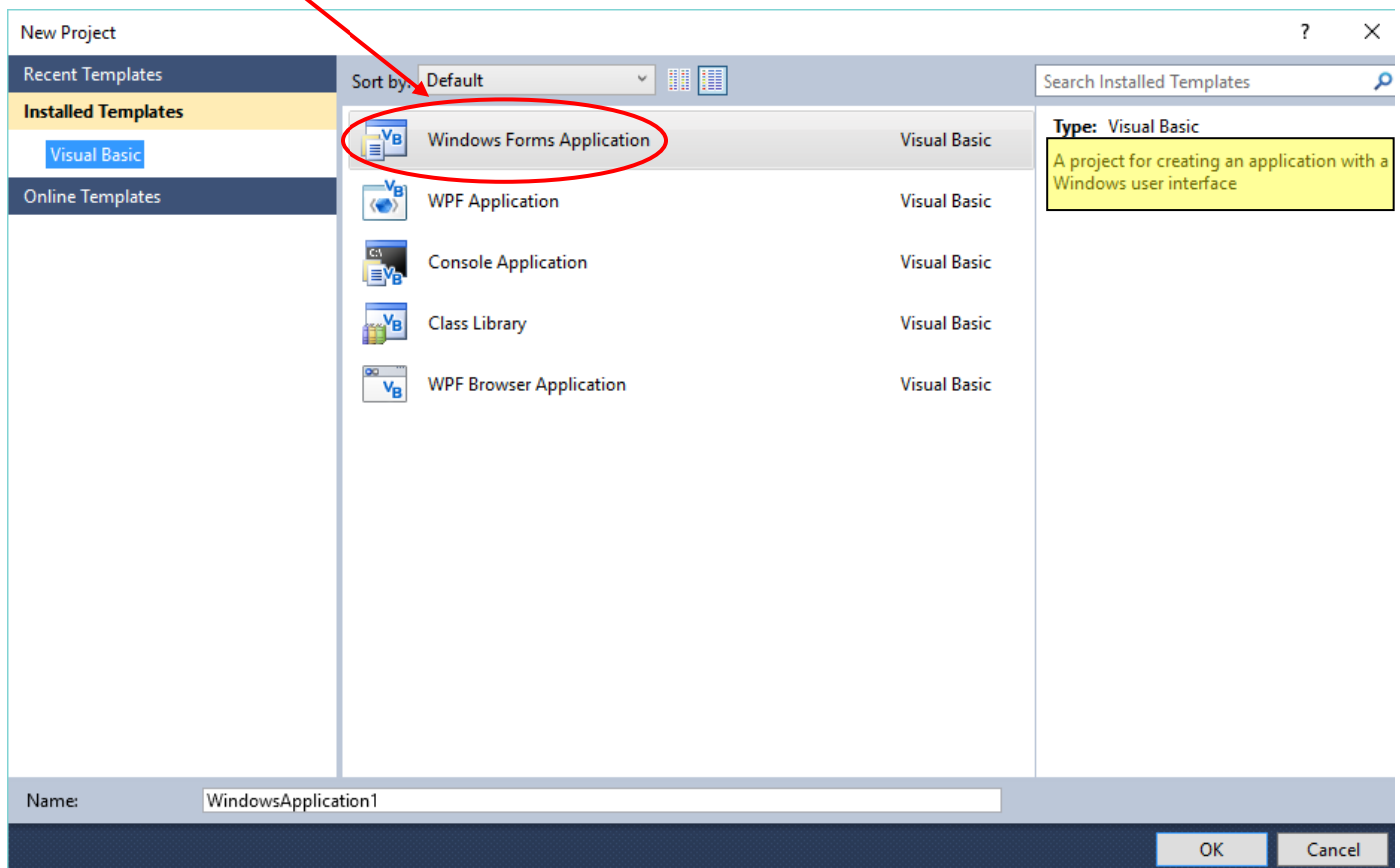
1. Double-click the “VB 2010 Express” icon.
2. After the Visual Basic 2010 Express is loaded into memory, the “Start Page” is displayed. Click on “New Project” to begin a new project.



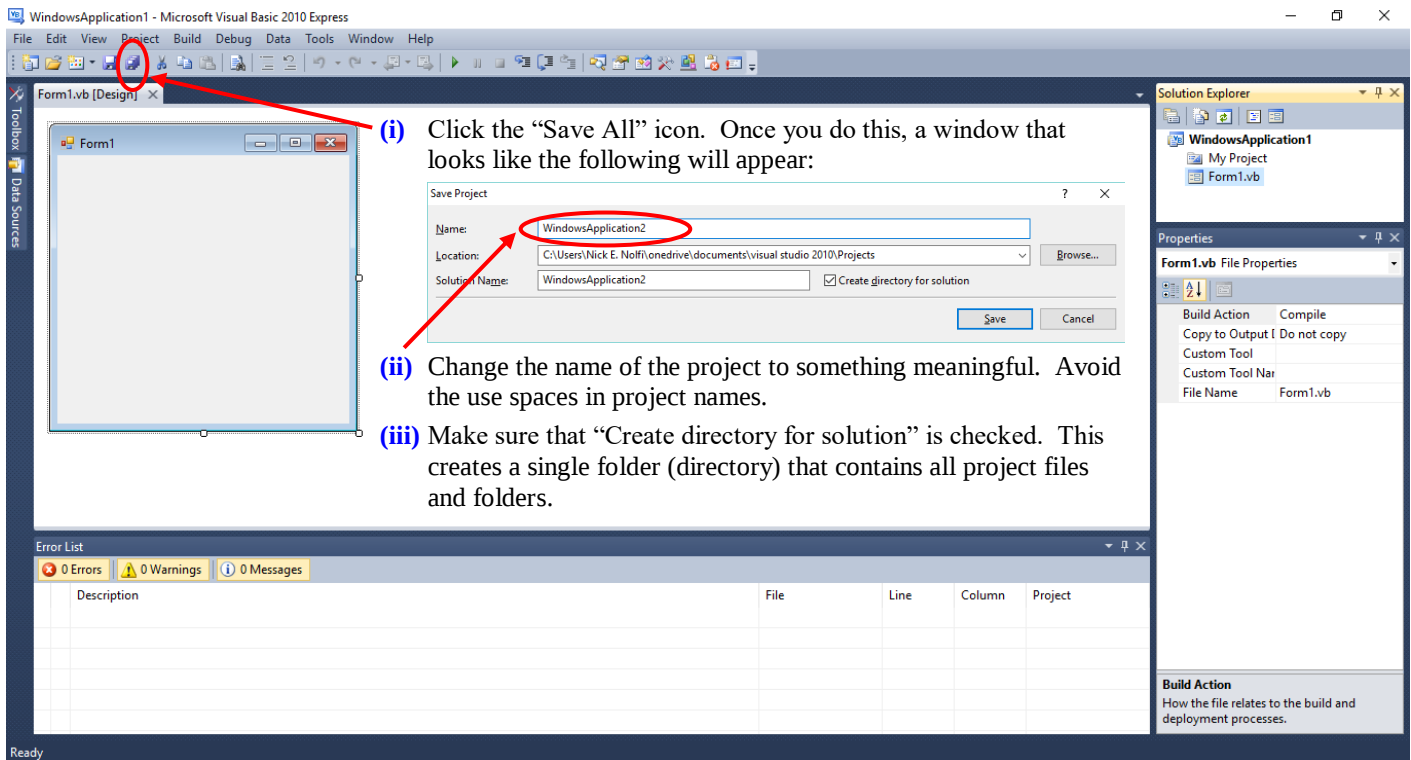
Microsoft
Visual Basic
2010 Express



3. Choose “Windows Forms Application” and click “OK.” A form is simply a standard Windows window.

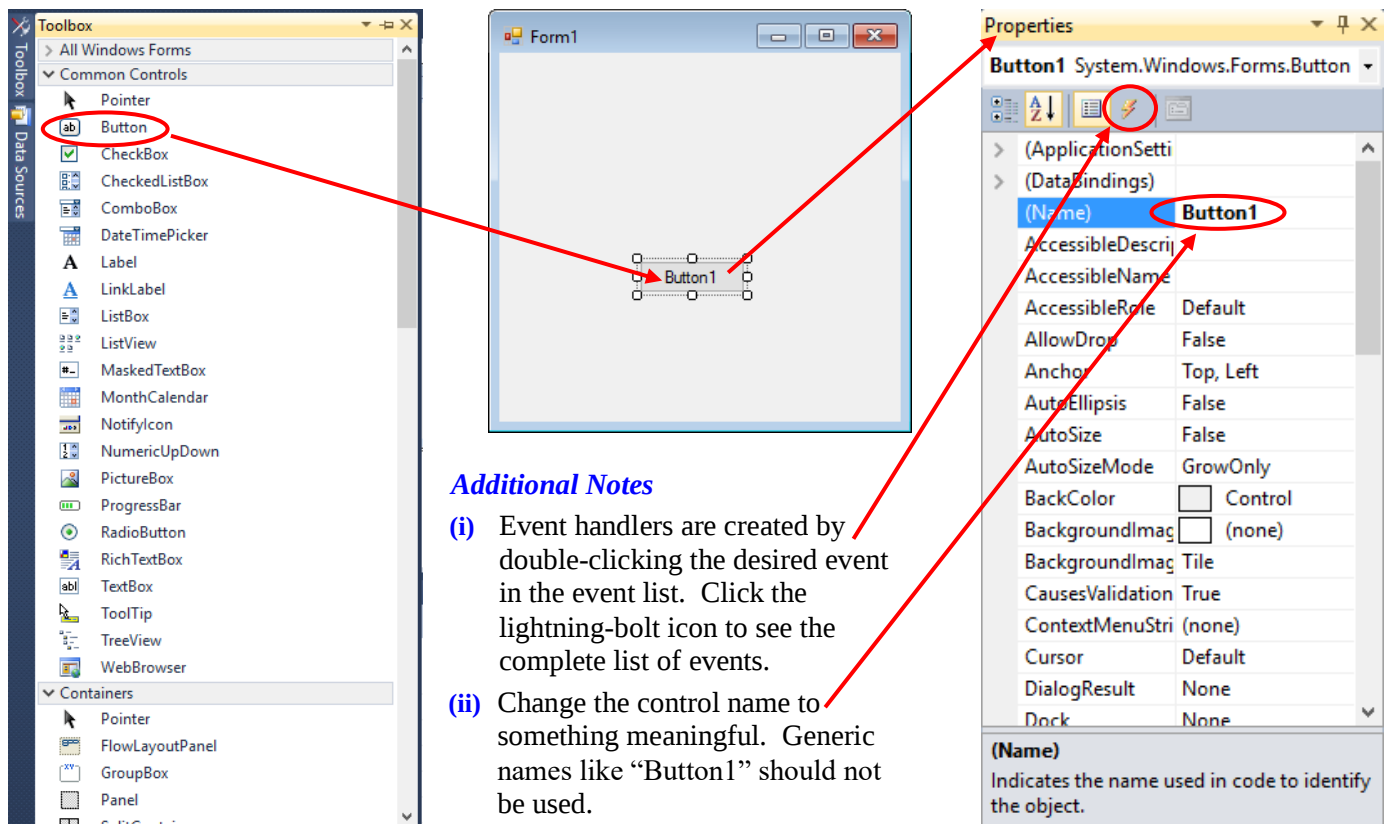


- A blank project with one form (window) is created after “OK” is clicked in the previous step. The project should be **saved properly** right away to prevent the possibility of data loss. To do this, follow the steps given below.

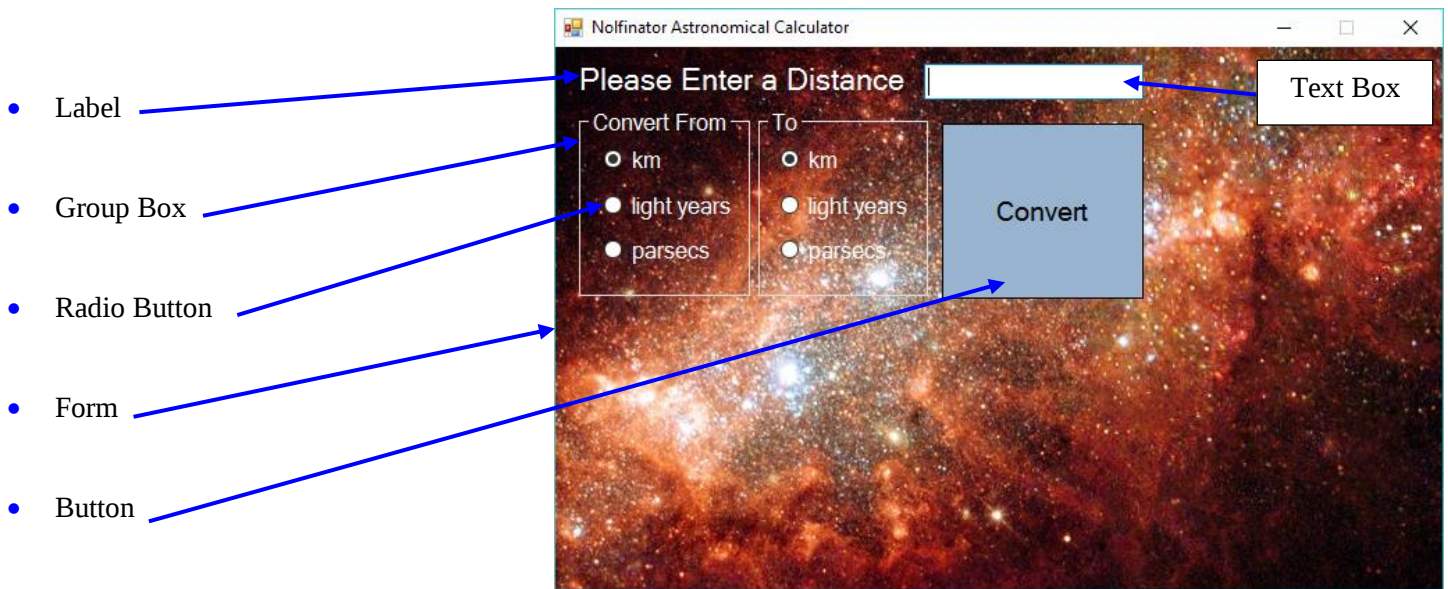


The Main Features of the Visual Studio Development Environment

- Select a control (component) from the toolbox.
- Place the selected control on the form.
- Use the “Properties” window to modify properties of the control.



INTRODUCTION TO VISUAL BASIC: ASTRONOMICAL CALCULATOR



Questions

Copy **S:\OUT\Nolfinator\ICS3U0\01-Visual Basic 2010\Astronomical Calculator** to your “G” drive. Open the Visual Basic solution as demonstrated in class. Study the code as well as the form designer. Then answer the following questions:

1. Unlike App Inventor, Visual Basic requires that variables be **declared** to have a specific **data type**. Explain the meaning of the data types found in the “Astronomical Calculator” application. (i.e. **Long**, **Double** and **String**)
2. How can you tell the difference between **local variables** and **global variables** in Visual Basic?
3. What is a **Sub** in Visual Basic?
4. What is the purpose of the “if” statement at the end of the event handler called “ConvertButton_Click?”
5. What is the purpose of the “Val” function? What can go wrong if it’s not used?
6. What is a **named constant**? What are the advantages of using named constants?
7. What is the purpose of comments in computer programs? How are comments created in Visual Basic?
8. What are strings? Why are quotation marks used to enclose the values of strings?
9. Radio buttons have a property called “Checked.” What is the purpose of this property?
10. The “&” operator is called the string concatenation operator. What is its purpose? Which block in App Inventor serves the same purpose?
11. The string "" is called the **null string** or **empty string**. What does it mean?
12. What is the purpose of the “CStr” function?
13. What is the reason that certain lines in Visual Basic code end with a space followed by an underscore?
14. What is a “class?”

```
Public Class AstronomicalCalculatorForm
```

```
'GLOBAL Named Constants. These make programs both easier to read AND easier to modify.
```

```
Const KM_IN_ONE_LIGHT_YEAR As Long = 9460528400000
```

```
Const KM_IN_ONE_PARSEC As Long = 30856775800000
```

```
'If any GLOBAL variables are required, they should be declared here, that is, outside of the procedures.
```

```
'Event handler: The procedure "ConvertButton_Click" handles the "Click" Event on "ConvertButton"
```

```
Private Sub ConvertButton_Click(sender As System.Object, e As System.EventArgs) Handles ConvertButton.Click
```

```
    'MEMORY: Declare LOCAL Variables
```

```
    Dim distance As Double, convertedDistance As Double, timeToTraverseDistance As Double
```

```
    Dim fromUnits As String, toUnits As String
```

```
    'INPUT: Copy value in text box to the variable "distance."
```

```
    'The function "Val" converts from string (text) form to numeric form.
```

```
    distance = Val(distanceTextBox.Text)
```

```
    'PROCESSING: Determine which radio buttons are selected and perform appropriate calculations.
```

```
    If fromKmRadioButton.Checked And toLightYearsRadioButton.Checked Then
```

```
        convertedDistance = distance / KM_IN_ONE_LIGHT_YEAR
```

```
        timeToTraverseDistance = distance / KM_IN_ONE_LIGHT_YEAR
```

```
        fromUnits = "km"
```

```
        toUnits = "light year(s)"
```

```
    ElseIf fromKmRadioButton.Checked And toParsecsRadioButton.Checked Then
```

```
        convertedDistance = distance / KM_IN_ONE_PARSEC
```

```
        timeToTraverseDistance = distance / KM_IN_ONE_LIGHT_YEAR
```

```
        fromUnits = "km"
```

```
        toUnits = "parsec(s)"
```

```
    ElseIf fromLightYearsRadioButton.Checked And toKmRadioButton.Checked Then
```

```
        convertedDistance = distance * KM_IN_ONE_LIGHT_YEAR
```

```
        timeToTraverseDistance = distance
```

```
        fromUnits = "light year(s)"
```

```
        toUnits = "km"
```

```
    ElseIf fromLightYearsRadioButton.Checked And toParsecsRadioButton.Checked Then
```

```
        convertedDistance = distance * KM_IN_ONE_LIGHT_YEAR / KM_IN_ONE_PARSEC
```

```
        timeToTraverseDistance = distance
```

```
        fromUnits = "light year(s)"
```

```
        toUnits = "parsec(s)"
```

```
    ElseIf fromParsecsRadioButton.Checked And toKmRadioButton.Checked Then
```

```
        convertedDistance = distance * KM_IN_ONE_PARSEC
```

```
        timeToTraverseDistance = convertedDistance / KM_IN_ONE_LIGHT_YEAR
```

```
        fromUnits = "parsec(s)"
```

```
        toUnits = "km"
```

```
    ElseIf fromParsecsRadioButton.Checked And toLightYearsRadioButton.Checked Then
```

```
        convertedDistance = distance * KM_IN_ONE_PARSEC / KM_IN_ONE_LIGHT_YEAR
```

```
        timeToTraverseDistance = convertedDistance / KM_IN_ONE_LIGHT_YEAR
```

```
        fromUnits = "parsec(s)"
```

```
        toUnits = "light year(s)"
```

```
    Else
```

```
        convertedDistance = distance
```

```
        fromUnits = ""
```

```
        toUnits = ""
```

```
    End If
```

```
    'OUTPUT: Display the results (Note that "<>" means "not equal to")
```

```
    If fromUnits <> toUnits Then
```

```
        outputLabel.Text = CStr(distance) & " " & fromUnits & " = " & CStr(convertedDistance) & " " & toUnits
```

```
        outputLabel2.Text = "Moving at the speed of light, it would take " & vbCrLf & _  
            CStr(timeToTraverseDistance) & " year(s) to travel " & distance & " " & fromUnits & "."
```

```
    Else
```

```
        outputLabel.Text = "WTF!!!! Why would you convert to the SAME unit?"
```

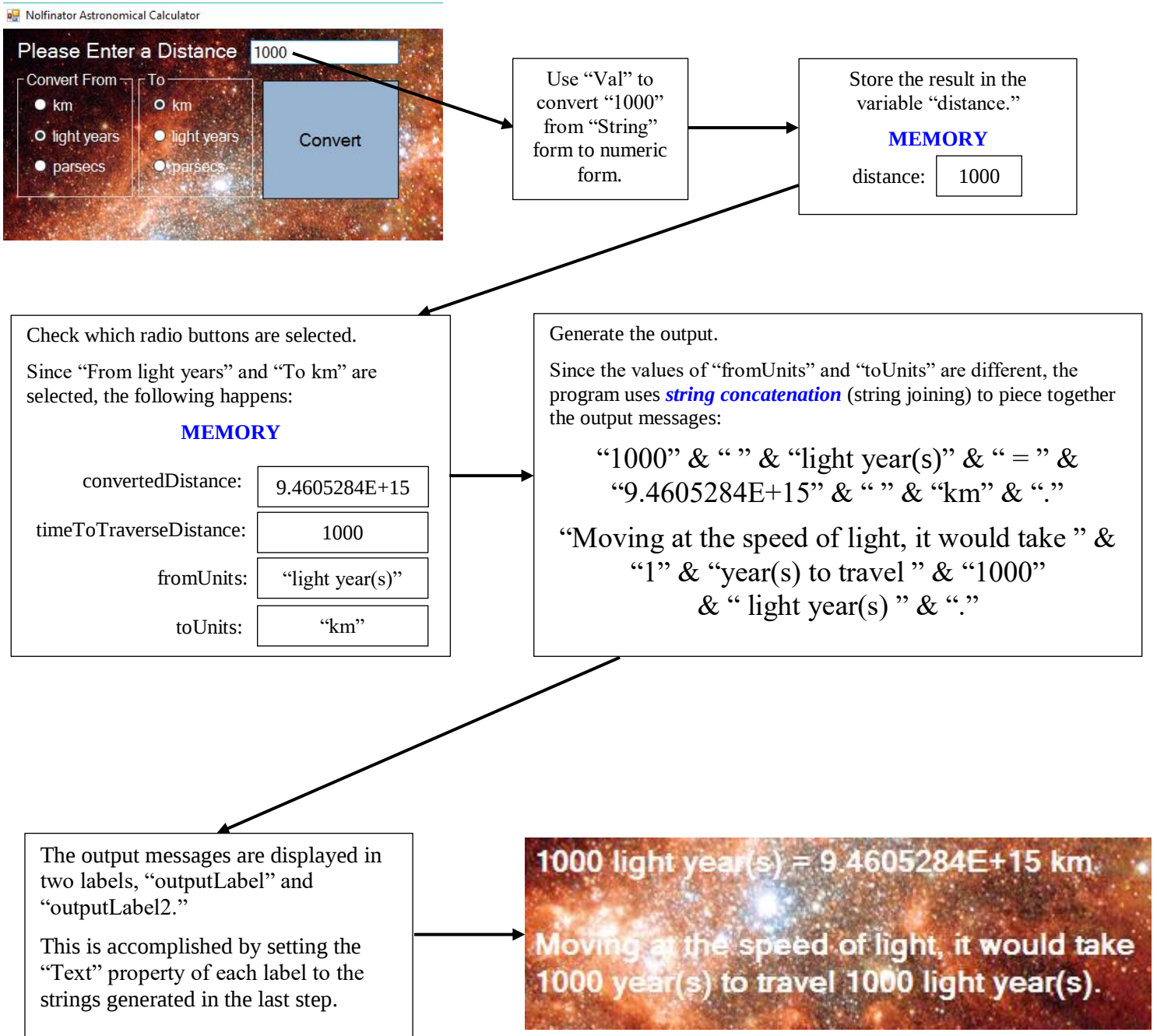
```
        outputLabel2.Text = ""
```

```
    End If
```

```
End Sub
```

```
End Class
```

Pictorial Description of the Execution of the “Astronomical Calculator” Program



AN ACCELERATED INTRODUCTION TO VISUAL BASIC

Visual Basic (VB for Short)

- **Why Visual?**

As with App Inventor, the user interface can be created by pointing, dragging and dropping.
The user interface can be created *without* writing a single line of code, that is, entirely visually!

- **Why BASIC?**

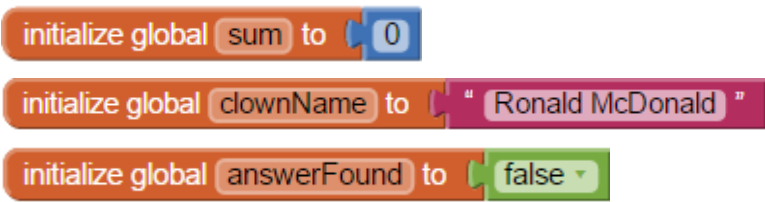
BASIC is an acronym for “**B**eginner’s **A**ll-purpose **S**ymbolic **I**nstruction **C**ode”

BASIC is a *family* of general-purpose, high-level programming languages whose design philosophy is *ease of use*.
The first version dates back to 1964! (That’s old! Mr. Nolfi was born in 1963!)

For more background information, see <http://en.wikipedia.org/wiki/BASIC>.

Comparing the Main Structures of App Inventor to the Main Structures of VB

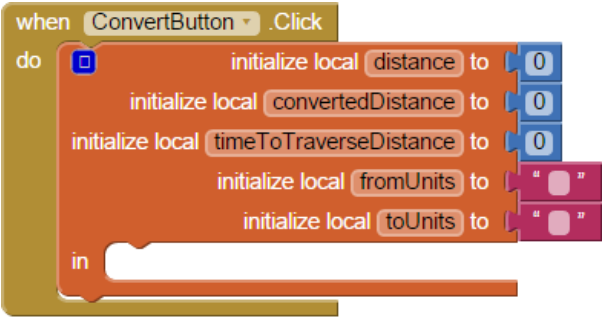
GLOBAL VARIABLES

App Inventor	Visual Basic
In App Inventor, global variables must be declared using “initialize global” blocks. Such blocks require that both the name and the initial value of the variable be specified. However, App Inventor does not have a mechanism for <i>explicitly</i> defining the type of data that a variable will store. This is determined <i>implicitly</i> once a value is assigned to the variable. 	<pre>Public Class Form1 'Global variables are DECLARED at the class 'level. Unlike App Inventor, the data type of 'variables must be explicitly specified in 'declaration statements. Initial values of 'variables can also be assigned in 'declarations. The command "Dim" is used to 'begin a variable declaration statement. Dim sum As Integer = 0 Dim clownName As String = "Ronald McDonald" Dim answerFound As Boolean = False . . . End Class</pre>

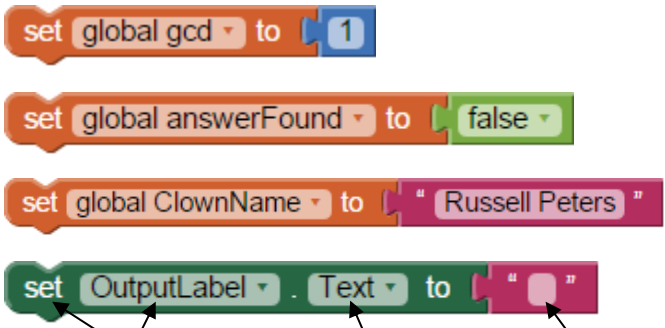
Note: Due to lack of space, some of the VB statements shown below are spread over two or more physical lines. The character combination “_” (a space followed by an underscore) is used in VB to indicate that a statement continues on the next line.

LOCAL VARIABLES

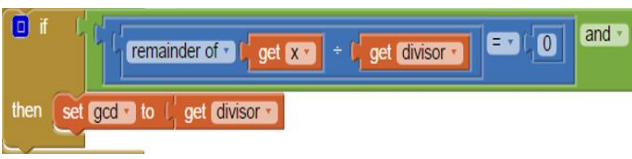
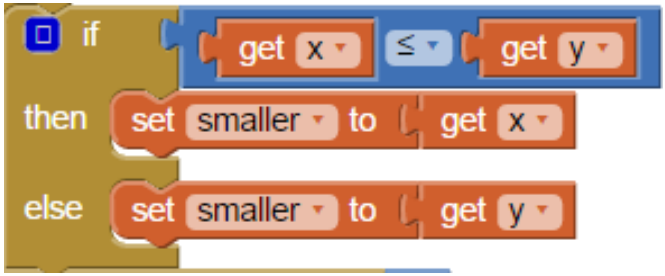
Local variables make debugging easier and allow memory to be used more efficiently. Use them whenever possible!

App Inventor	Visual Basic
Local variables are declared within procedures and subsections of procedures using “initialize local” blocks. Recall that such variables only “exist” on a temporary basis and in a restricted section of code. 	<pre>'Local variables are DECLARED within procedures and 'subsections of procedures (e.g. within "If" statements 'and loops). The following shows variables declared and 'initialized within an event-handling "Sub" procedure. Private Sub ConvertButton_Click(sender As _ System.Object, e As System.EventArgs) _ Handles ConvertButton.Click 'MEMORY: Declare Variables Dim distance As Double = 0 Dim convertedDistance As Double = 0 Dim timeToTraverseDistance As Double = 0 Dim fromUnits As String = "" Dim toUnits As String = "" . . End Sub</pre> “Sub” stands for <i>subroutine</i> or <i>subprogram</i>.

ASSIGNMENT STATEMENTS

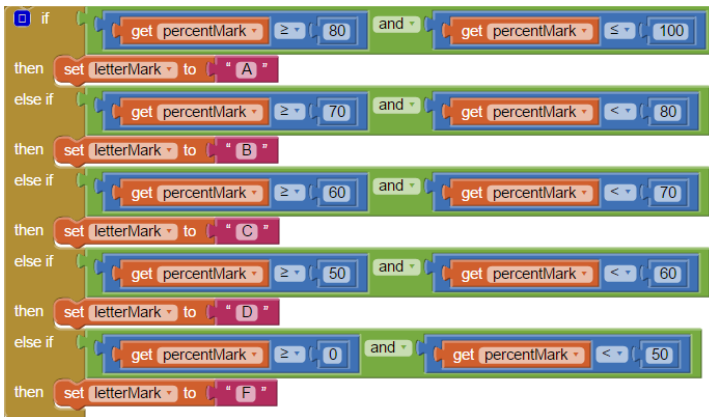
App Inventor	Visual Basic
Assignment statements assign (give) values to variables.  Name of Component Is Given the Value Name of Property The “Null” or “Empty” String	'The following assumes that the variables 'have already been declared. <pre>gcd = 1 clownName = "Russell Peters" answerFound = False</pre> 'Controls that are created using the Forms designer 'are declared automatically but the declarations are 'normally hidden from the applications programmer. 'Occasionally, it is necessary to access this '"hidden" code. This can be done through the '"Solution Explorer" window. <pre>OutputLabel.Text = ""</pre> Name of Object Name of Property Is Given the Value The “Null” or “Empty” String

“IF” STATEMENTS

App Inventor	Visual Basic
 	<pre>If x Mod divisor = 0 And _ y Mod divisor = 0 Then gcd = divisor End If</pre> <pre>If x <= y Then smaller = x Else smaller = y End If</pre>

“IF” STATEMENTS (continued)

App Inventor



Visual Basic

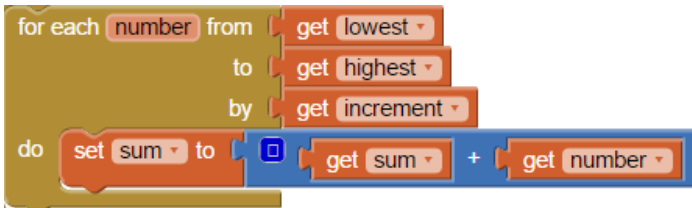
```

If percentMark >= 80 And percentMark <= 100 Then
    letterMark = "A"
ElseIf percentMark >= 70 And percentMark < 80 Then
    letterMark = "B"
ElseIf percentMark >= 60 And percentMark < 70 Then
    letterMark = "C"
ElseIf percentMark >= 50 And percentMark < 60 Then
    letterMark = "D"
ElseIf percentMark >= 0 And percentMark < 50 Then
    letterMark = "F"
End If

```

COUNTED LOOPS

App Inventor



Visual Basic

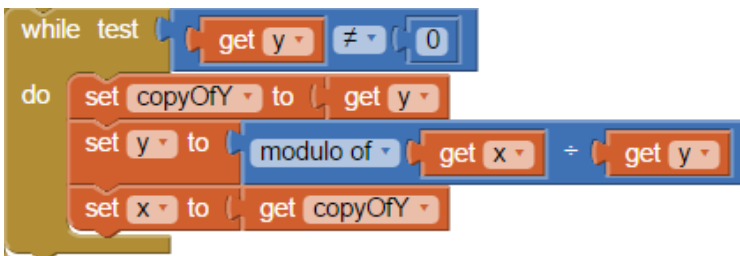
```

For number As Integer = lowest To highest _
    Step Increment
    sum = sum + number
Next

```

CONDITIONAL LOOPS

App Inventor



Visual Basic

'Perform the Euclidean algorithm to find
'the greatest common divisor of 'x' and 'y'

```

Do While y <> 0 ' "<>" means "not equal to"

    copyOfY = y
    y = x Mod y
    x = copyOfY

Loop

```

Looping Structures in VB not Available in App Inventor

VB has a far richer variety of features than App Inventor does. Listed below are some VB looping structures for which there are no analogues in App Inventor.

Various forms of Conditional Loops: Four Different Ways of Implementing the Euclidean Algorithm

'In the first two examples, the looping condition appears at the beginning of the loop.
'This means that it is possible for the loop to be skipped entirely.

```
Do While y <> 0

    copyOfX = x
    x = y
    y = copyOfX Mod y
```

Loop

```
Do Until y = 0

    copyOfX = x
    x = y
    y = copyOfX Mod y
```

Loop

'In the next two examples, the looping condition is placed at the end of the loop. This means that at least one repetition must be performed.

```
Do

    copyOfX = x
    x = y
    y = copyOfX Mod b
```

Loop While y <> 0

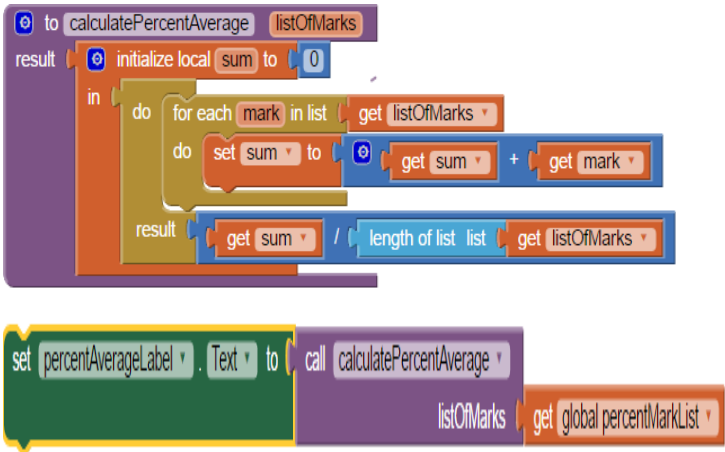
```
Do

    copyOfX = x
    x = y
    y = copyOfX Mod y
```

Loop Until y = 0

PROCEDURES

App Inventor



Visual Basic

```
'A procedure WITH a result is called a
'"Function" in Visual Basic

Private Function calculatePercentAverage _
    (listOfMarks As List(Of Single)) As Single

    Dim sum As Single

    For Each mark As Single In listOfMarks
        sum = sum + mark
    Next

    Return sum / listOfMarks.Count

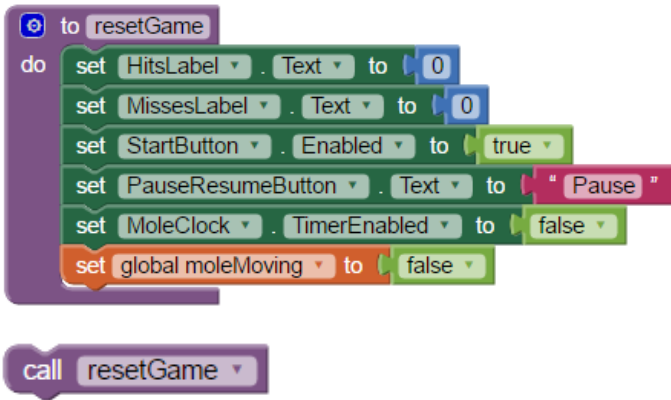
End Function

'Example call
percentAverageLabel.Text = _
    calculatePercentAverage(percentMarkList)
```

PROCEDURES (continued)

App Inventor

Visual Basic



'A procedure WITHOUT a result is called a
'"Sub" in Visual Basic

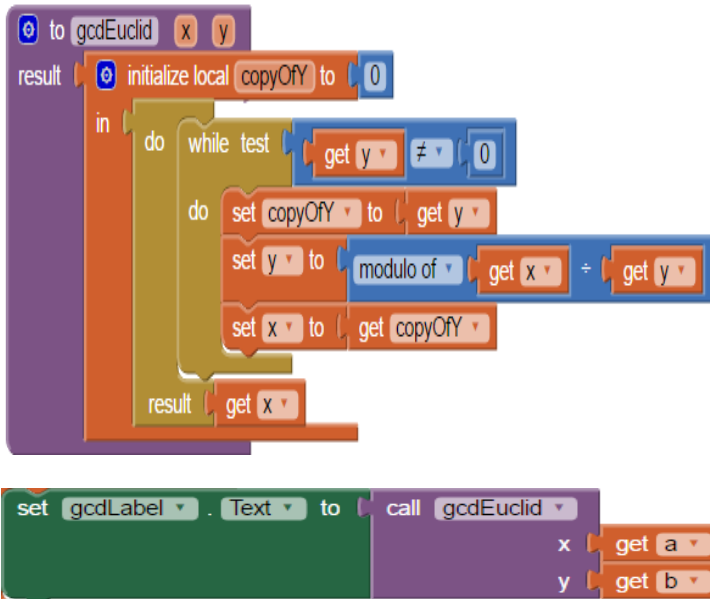
Private Sub resetGame()

```

hitsLabel.Text = 0
missesLabel.Text = 0
startButton.Enabled = True
pauseResumeButton.Text = "Pause"
moleClock.Enabled = False
moleMoving = False
  
```

End Sub

'Example call
resetGame()



Private Function gcdEuclid (ByVal x As _
Integer, ByVal y As Integer) As Integer

Dim copyOfY As Integer

```

While y <> 0
  copyOfY = y
  y = x Mod y
Loop
  
```

Return x

End Function

'Example call: a and b are Integer variables
percentGcd.Text = gcdEuclid(a, b)

UNDERSTANDING THE LEARNVBByCOMPARINGToAPPINVENTOR PROGRAM

1. List all the global variables in the “LearnVbByComparingToAppInventor” program.

<i>Variable Name</i>	<i>Data Type (Type of Data Stored)</i>	<i>Purpose of Variable</i>

2. Explain the difference between global and local variables. Why should you exercise caution when using global variables?

3. What is an array? Explain how the array called ‘prime’ is used in the Eratosthenes’ Sieve program. Is ‘prime’ a global or local array?

4. List all the global objects in the “LearnVbByComparingToAppInventor” program. (Don’t bother to list the objects created visually by using the form designer. Only list the objects that are defined through code.)

<i>Object Name</i>	<i>Type of Object</i>	<i>Purpose of Object</i>

5. List all the event handler procedures in the “LearnVbByComparingToAppInventor” program.

<i>Event Handler Name</i>	<i>Event Handled</i>	<i>Purpose of Procedure</i>

(Continued from previous page)

<i>Event Handler Name</i>	<i>Event Handled</i>	<i>Purpose of Procedure</i>

6. List all the general procedures in the “LearnVbByComparingToAppInventor” program.

<i>Procedure Name</i>	<i>Parameters (if any)</i>	<i>Purpose of Procedure</i>

7. Explain the difference between an event handler procedure and a general procedure.

8. List all the *methods* in the “LearnVbByComparingToAppInventor” program.

<i>Method Name</i>	<i>Class that Method belongs to</i>	<i>Purpose of Method</i>

9. **Variable declaration statements** are programming statements that are used to state the **name** and **type** of a variable.

Examples

```
Dim funkyPicStep As Integer = 10      'Initial value of 'funkyPicStep' is 10
Dim erasePicture As Boolean = False    'Initial value of 'erasePicture' is 'False'
```

List examples of variable declaration statements in the “LearnVbByComparingToAppInventor” program.

10. **Assignment statements** are programming statements that are used to give values to variables.

Examples

```
xShift = 2 'Assigns a value of '2' to the variable 'xShift'
'Assign a string value to the 'Text' property of the object 'EratosthenesLabel'
EratosthenesLabel.Text = "Removing Proper Multiples of " & CStr(number) & "..."
```

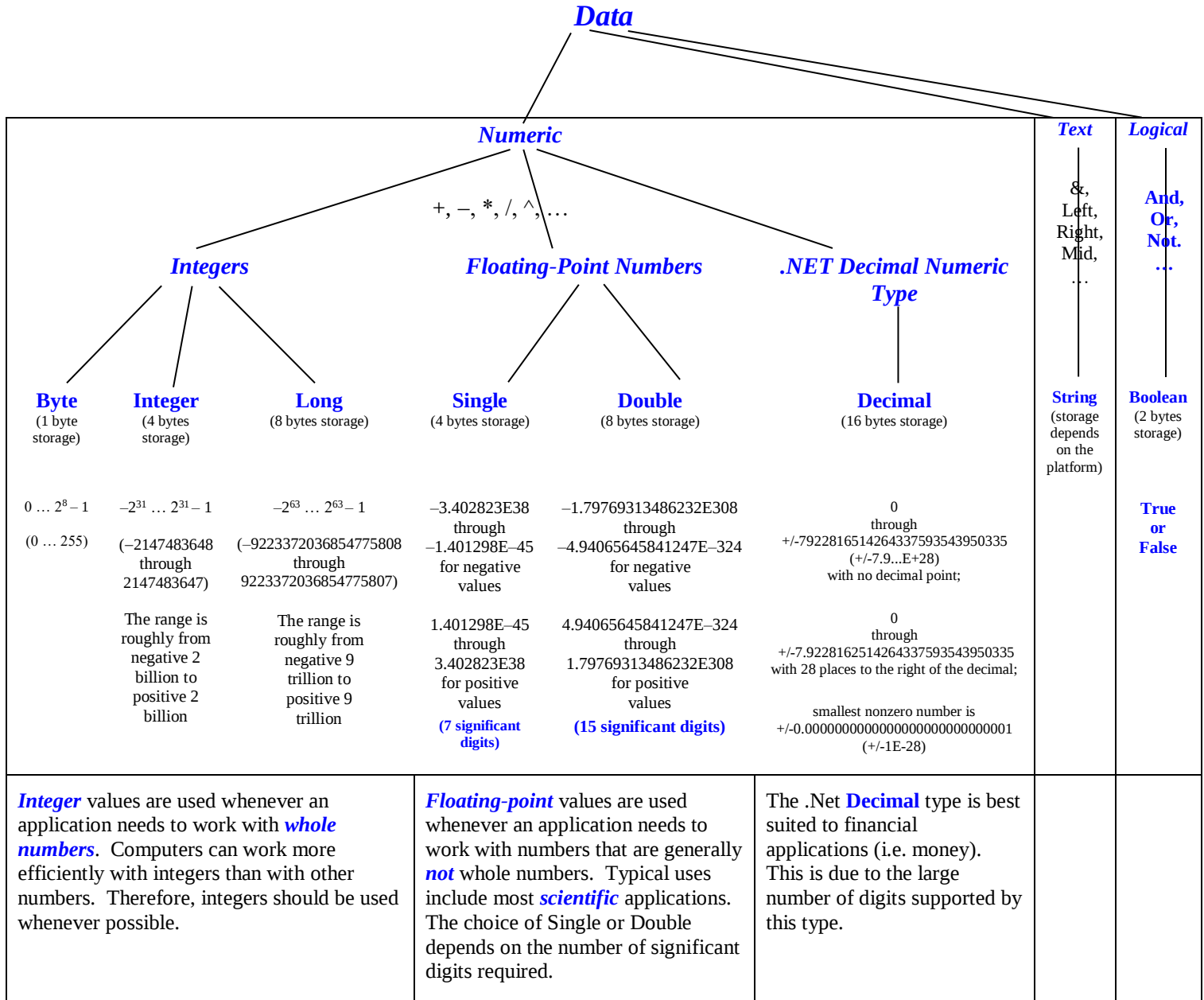
List examples of assignment statements in the “LearnVbByComparingToAppInventor” program.

11. Describe the structure of counted and conditional loops in Visual Basic 2010. Use examples from the “LearnVbByComparingToAppInventor” program to illustrate your answer.

DATA TYPES

Data (Information) – A Partial List of VB Data Types

As mentioned earlier in this unit, a computer can be viewed as a **data processing machine**. Since data can be categorized into various forms that require **differing amounts of memory** and **different types of operations**, programming languages offer diverse **data types**. A summary of the **most commonly used types of data** studied in this course is given in the following diagram.



For a complete list of VB.Net data types see <http://msdn.microsoft.com/en-us/library/47zceaw7%28v=vs.100%29.aspx> .

Important Points about Data Types

- Although computer circuits can process only the binary values 0 and 1, programs need to process a wide variety of types of data including **numbers**, **text** and **logical values** (i.e. values that are either true or false).
- Encoding schemes** are used to give a **meaning** to raw binary data. That is, encoding schemes use binary numbers to represent information. See the table below for a few common examples of encoding schemes.
- Variables need to be declared so that both of the following are known:

Amount of Memory Required

Encoding Scheme that should be used to interpret the Raw Binary Data

Bits and Bytes

1 bit = 1 **binary digit**
1 Byte = 8 bits (1 B = 8 b)

The following table gives several examples of commonly used encoding schemes.

Type of Data	Name of Encoding Scheme	Memory Required	Examples	
			Raw Binary Data Stored in RAM	What the Raw Binary Data Represent
Integer (Short in VB)	16-bit Twos Complement	2 bytes	0111111111111111	32767
String (Text)	Unicode	2 bytes	0111111111111111	喜翔
Integer (Integer in VB)	32-bit Twos Complement	4 bytes	11000011100110001101000000000000	-1013395456
Floating Point (Single in VB)	32-bit IEEE754	4 bytes	11000011100110001101000000000000	-305.625

Questions

- Why do programming languages offer so many different data types?
- Visit www.unicode.org and find the Unicode hexadecimal (base 16) code for each of the following characters. Then use a Web-based converter or the Windows calculator to convert to binary. (Windows calculator must be in “Scientific” view.)
 - ㇏ (Hiragana, Japanese) Hex code: Binary code:
 - ૨ (Gujarati, Indic) Hex code: Binary code:
- Now interpret the codes that you found in question 2 as 16-bit integers. Convert each code from binary form to decimal form. Again, you may use a Web-based converter or the Windows calculator.
- Without an encoding scheme, does raw binary data have any meaning?
- Complete the following table:

Standard Form	Scientific Notation	Scientific Notation (Programming Format)
23400000	2.34×10^7	2.34E7
	$9.10938188 \times 10^{-31}$ kg (mass of an electron)	
	1.99×10^{30} kg (mass of sun)	
		1.79769313486232E308 (largest Double value in VB)
0.000000475 m (wavelength of blue light)		
0.000000014 m (distance between conductors in a CPU, known as the fabrication process size)		

