What is a Computer?

- A computer is a logical device for processing information.
  - Specifically, computers process data.
    - Data = structured information
  - Base: Silicon VLSI technology
    - VLSI = Very Large Scale Integrated circuits
- Computers are Powerful!
  - Can perform logical computations much faster than Humans.
  - Current speed (desktop!): 4 $x10^9$ basic operations/sec (GHz)
    - Each a simple logical operation (division, shift, write, etc)
- Computers are Limited…
  - Computation basically *sequential*…
    - One operation at a time.
    - In contrast, Humans use *parallel* processing (by neurons).
      - We are better at complex tasks (e.g., Vision, Pattern Recognition)
  - Computers not very 'adaptive'…
    - Standard computers mainly do what they are told.
  - Communication difficult (computers think logically):
    - Programming languages (and programmers) required!

Software vs. Hardware

- At the most basic level computers can be broken down into two components:
  - Hardware and Software
- Hardware = the physical components of the computer system.
  - Data Processing: The Central Processing Unit (CPU)
  - Data Storage: Memory storage devices:
    - RAM (primary), Hard drive (secondary), flash disks (peripheral), etc
  - Data Communication: Devices for Input/Output:
    - Input: Keyboard, mouse, etc
    - Output: Display, printer, speaker
- Software = the computer programs that run on a computer
  - These establish logical control over the hardware:
    - Manage the details of Data Processing, Storage, and Communication.
  - The Operating System (OS): primary system control
    - Windows, Ubuntu Linux, Mac OS X, Unix, etc
  - Application Software: MS-Word, PowerPoint, Excel, etc
  - User-built Applications: using a Programming Language

Computer Languages

- Computer languages can be classified into 3 types:
  - Machine Languages:
    - Languages that the Computer can directly understand…
      - Each operation a string of digits (1's and 0's)
    - Machine Dependent: only usable on one platform.
    - Difficult for humans to freely use.
  - Assembly Languages:
    - More 'English-like': Uses words from natural languages…
      - Each an abbreviation for a single machine language operation.
    - Translated to Machine Language by special programs:
      - Assemblers
    - Still not convenient for Humans.
  - High-Level Languages (HLLs):
    - So-called Programming Languages.
    - Single statements can accomplish bigger tasks:
      - Groups of a set of related basic operations.
    - Much more convenient for Humans.

Programming Languages

- Many Programming Languages have been developed.
  - Some well-known compiled High Level Languages include (older to newer):

High-Level Language	Primary Usage (General)
FORTRAN (FORmula TRANslation)	Scientific (matrix) Calculations
COBOL (Common Business Oriented Language)	Office Computing
C	Operating System Development
Ada	Embedded Systems, Industry
BASIC (Beginner's All-purpose Symbolic Instruction Code)	Education, Windows Applications
C++	Information Processing, Engineering and Scientific Applications
JAVA	Web-based Systems, etc

- Many others, including interpreted languages: Python, Perl, Ruby, etc
- Languages allow communication between humans and computers…
  - This involves converting abstract algorithms for solving problems into a form understandable by the computer.
    - An 'executable' (i.e., run-able) form.
  - Such a converted algorithm is called a program.
  - The people that do the conversion (at the high level) are us…the programmers.

Interpreted vs. Compiled Languages

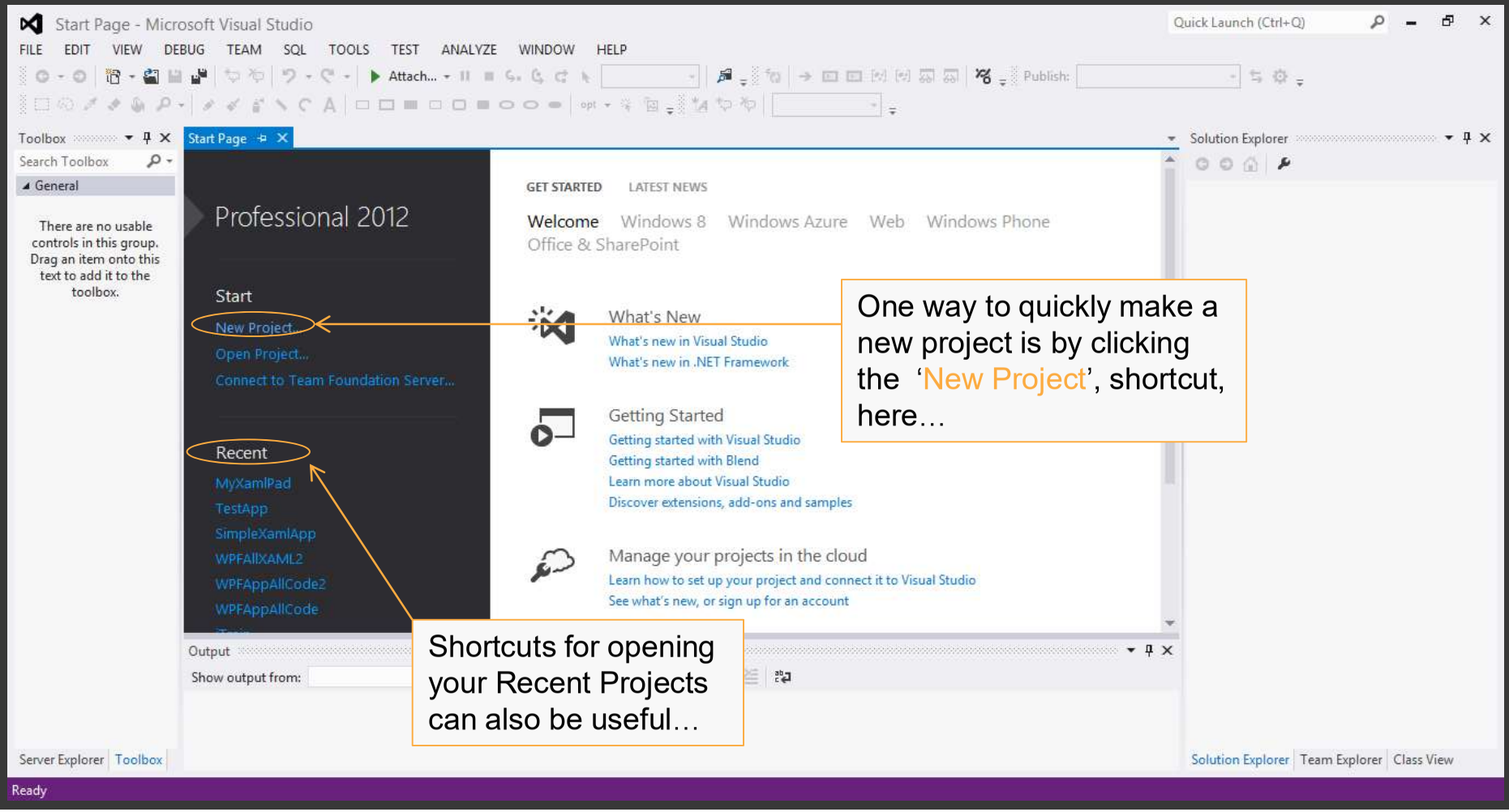
- Before execution, instructions in a program must also be converted:
  - from a text file (human-readable words in a HLL)…
    - …to an executable form (first to assembly, then to machine language)
- Two flavors exist for this conversion process:
  - In advance (compiled all at once):
    - Conversion by a program called a 'compiler'.
    - Faster, but less adaptable
      - …better for Engineering.
  - 'On the fly' (interpreted one instruction at a time):
    - Conversion by a program called an 'interpreter'.
    - Slower, but programs may be changed at run-time
      - …better for real-time Analysis and Management.
- Programming languages may be of either type...
  - Interpreted: Python, Perl, bash scripting (linux), javascript
  - Compiled: C, C++, C#, and VB .NET (visual basic)
  - Some (JAVA, VB .NET, C#) are essentially a combination of both:

Visual Basic vs. VB .NET

- BASIC
  - Beginner's All-purpose Symbolic Instruction Code
  - Developed as an extension of C, to be a general-purpose programming language.
- Visual BASIC (VB)
  - BASIC + a Graphical User Interface (GUI)
  - Greatly eases the creation of Windows applications
    - Especially, by facilitating the use of re-usable components
- Visual BASIC .NET
  - A programming language based on VB 6.0
  - Working on the .NET framework of the Microsoft Corporation
    - A Platform for cross-language development (C#, VB. NET, C++, F#)
    - Includes a large standard library called the BCL (Base Class Library)
- Visual Studio
  - Microsoft's Integrated Development Environment (IDE) for VB .NET.
  - Intended mainly for Windows Applications and Web Applications.
  - We will use Visual Studio 2012 to create all of our programs.

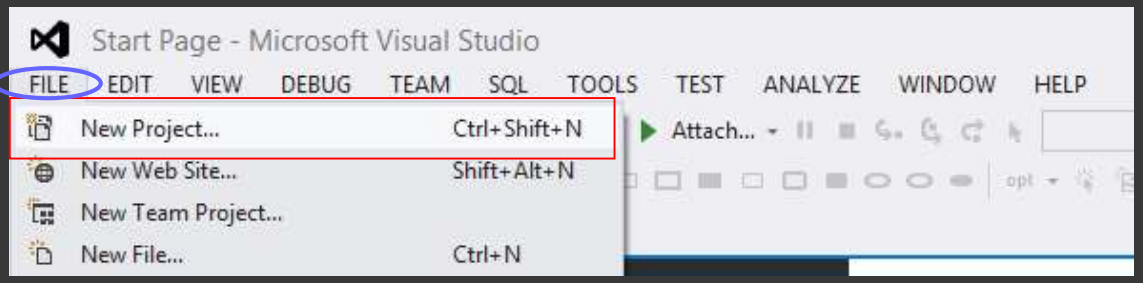
Starting Visual Studio 2012

- Go to Programs > All Programs > Microsoft Visual Studio 2012 (click)
  - After a few moments, Visual Studio 2012 (VS 2012) should open…
    - With the Start Page shown in the central window.
    - As shown, there are shortcuts for Project Creation and Project Opening, here…



Creating a New Project

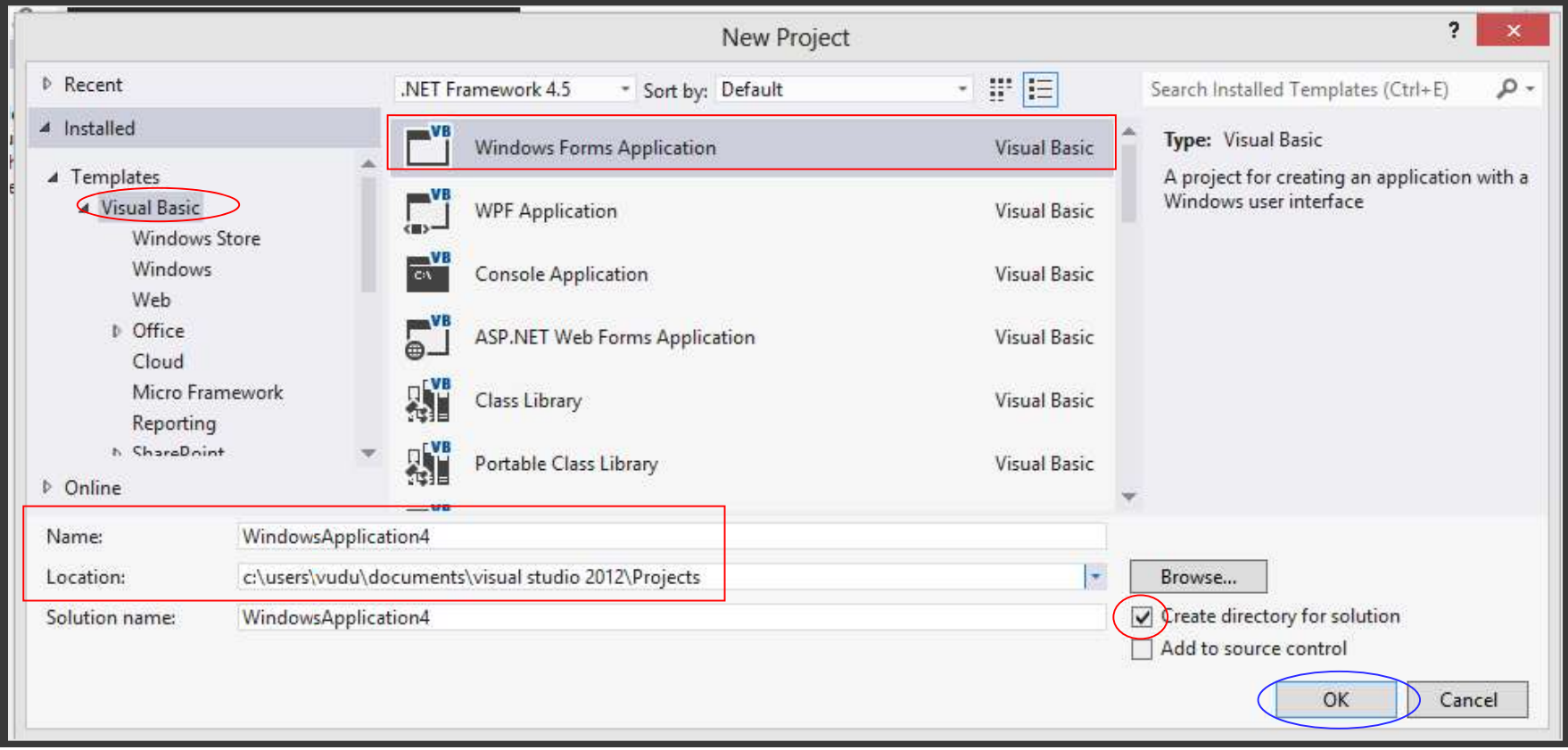
- Instead, let's create a new Visual Basic Project using the VS Menu…:
  - First, open the VS 2012 Menu > File Tab and click 'New Project'…



  - The New Project Dialog will appear (see next slide)…

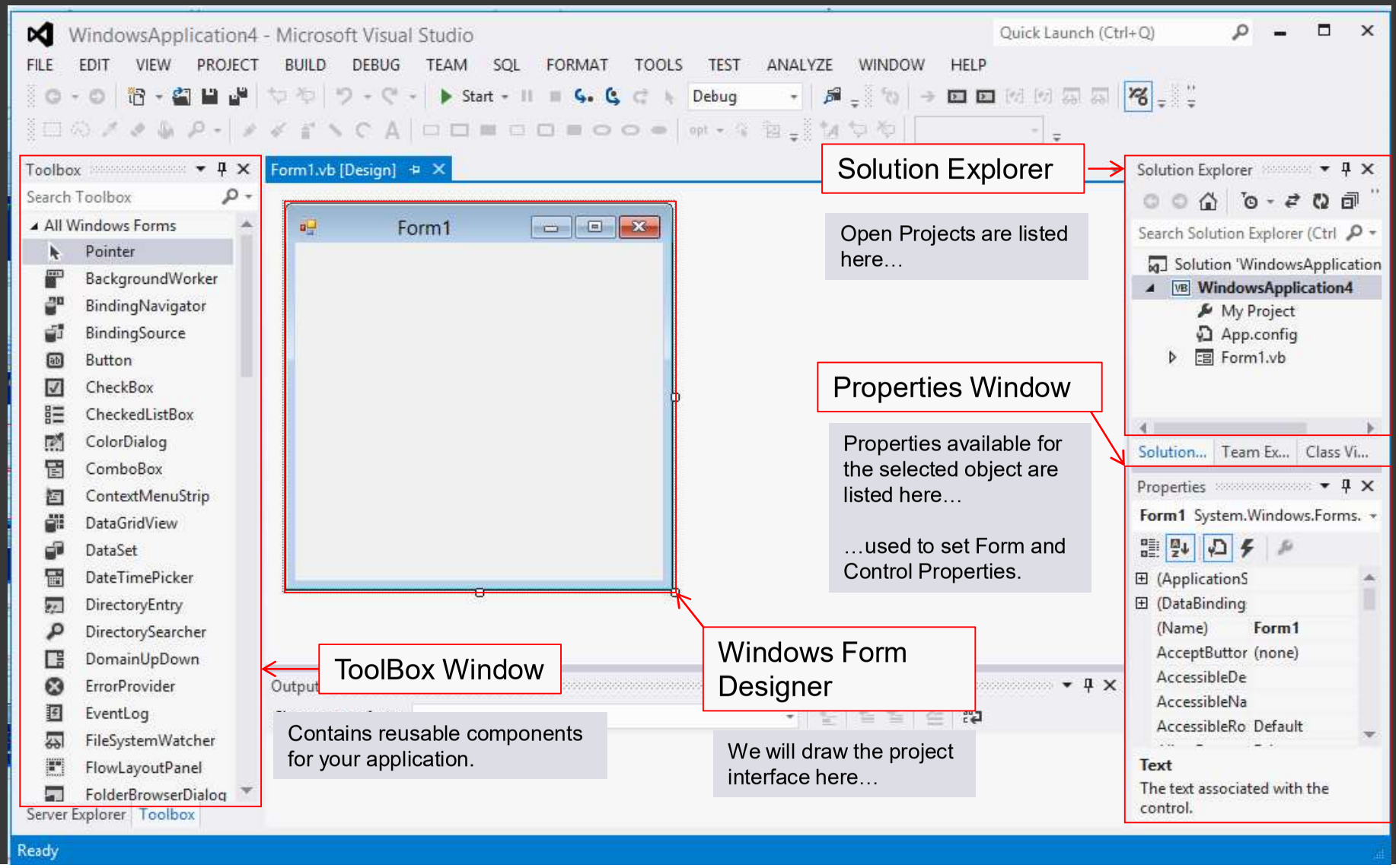
Creating a New Project (cont.)

- Use the 'New Project' Dialog to set the new project's type, name, etc:
  1. Select the Visual Basic Templates from the left-hand window…
     - Then, select 'Windows Forms Application' as our project type.
  2. Choose a Name and Location to store your Project; for now…
     - Keep the default Project Name (WindowsApplication1) and Location (later, copy to your USB)
  3. Finally, make sure 'Create directory for Solution' is checked…
     - And press OK …



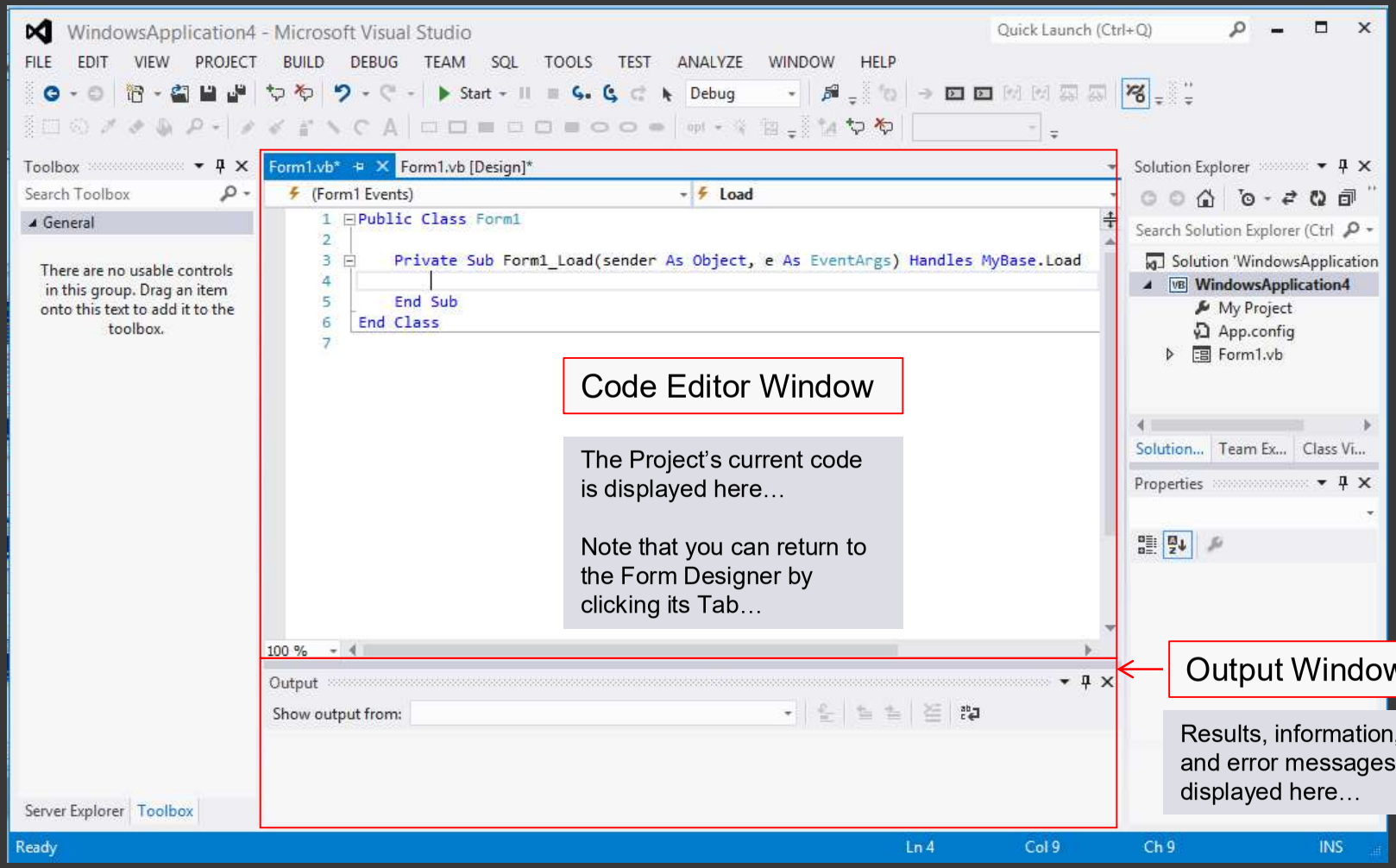
Visual Studio 2012 Main Screen

- The main screen will appear:



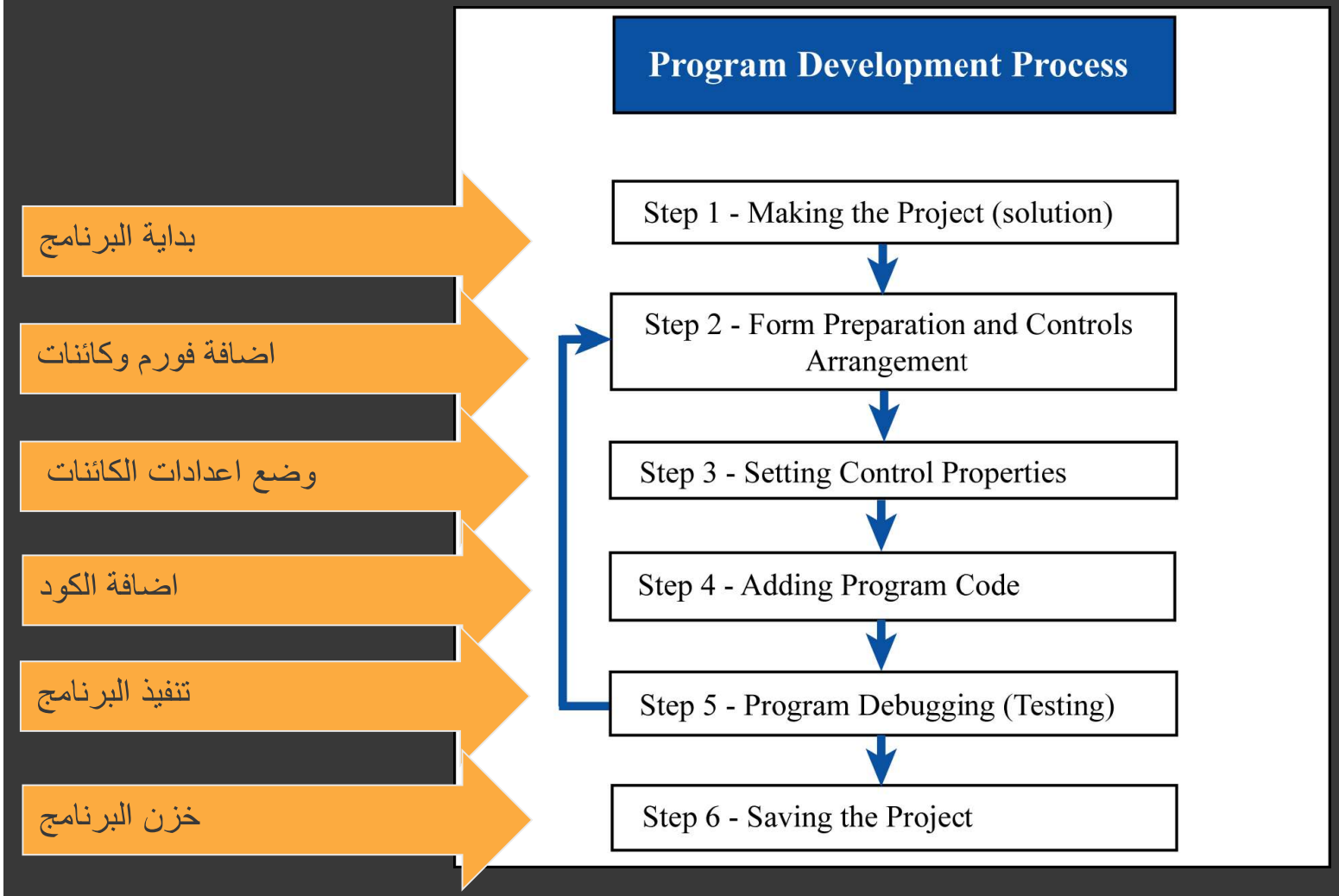
Visual Studio Main Screen (cont)

- Double-clicking the Design Window brings up the Code Editor.
  - This shows your project's current VB code.



Flow Chart for Program Preparation
خطوات تنفيذ البرنامج

- In this course, we will build VB projects by Incremental Development Process



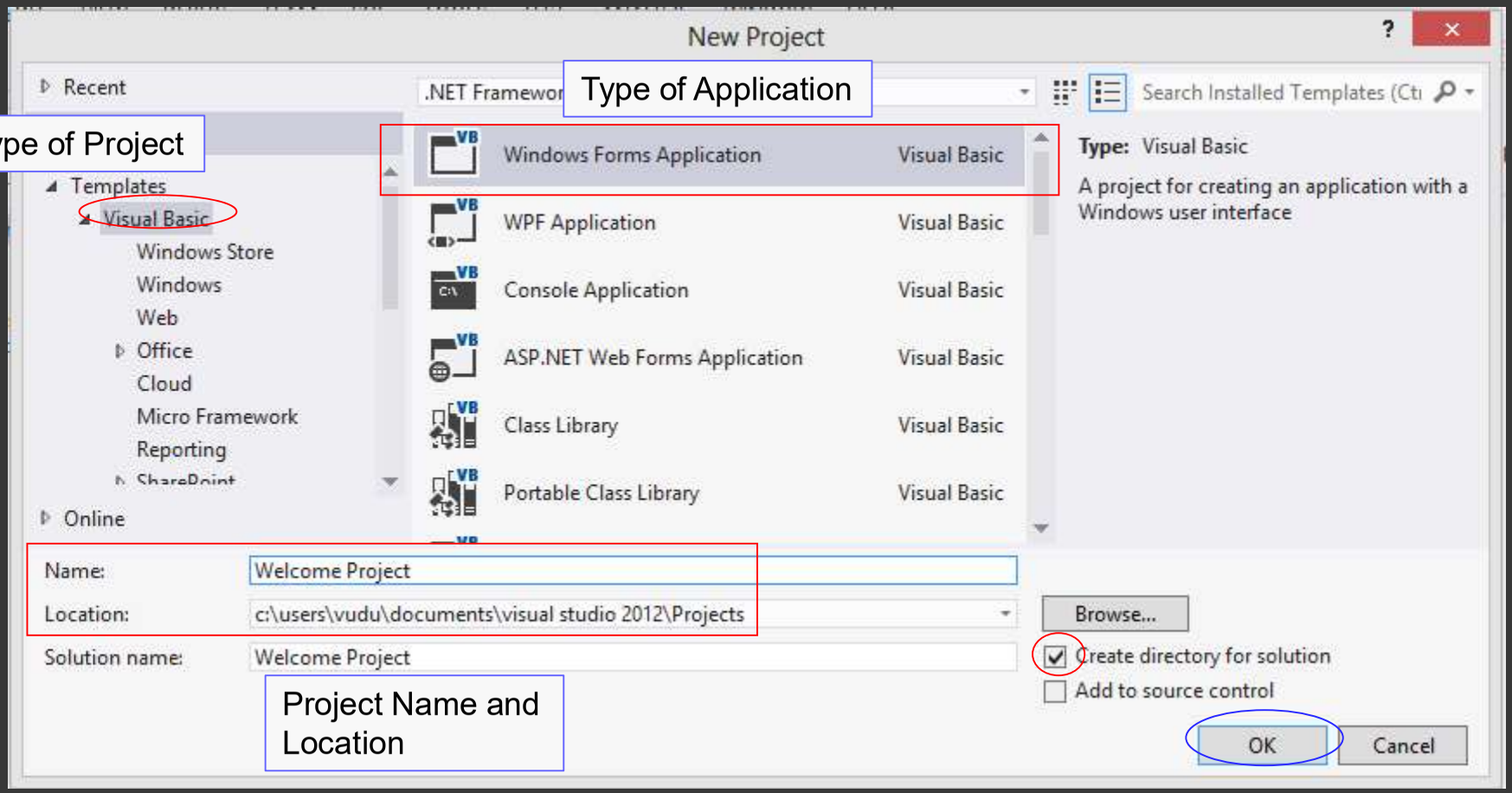
Let's Make a Simple Program

- We start by making a Program Plan:
  - A simple description of the desired characteristics and functionality.
    - Often includes an efficient method of solution (algorithm)
      - Example: a plan for adding two decimal numbers.

- Simple 'Welcome' program (plan):
  - **Program purpose**: Display a simple message; exit.
  - We will use 2 Buttons (each called a 'Control')
    - We will use Visual Studio's Design Window to create these.
  - **Desired functionality** (program behavior):
    - TextBox:
      - Allows our user to input his/her name
    - Clicking Button 1 ('"Welcome" Button):
      - Display 'Hello <User Name>! Welcome to Visual Basic.'
    - Clicking Button 2 ('Exit' Button):
      - Exit (close the program)
  - We will add each Control to our Form using the Design Window…
    - …and then add some simple VB .NET code.

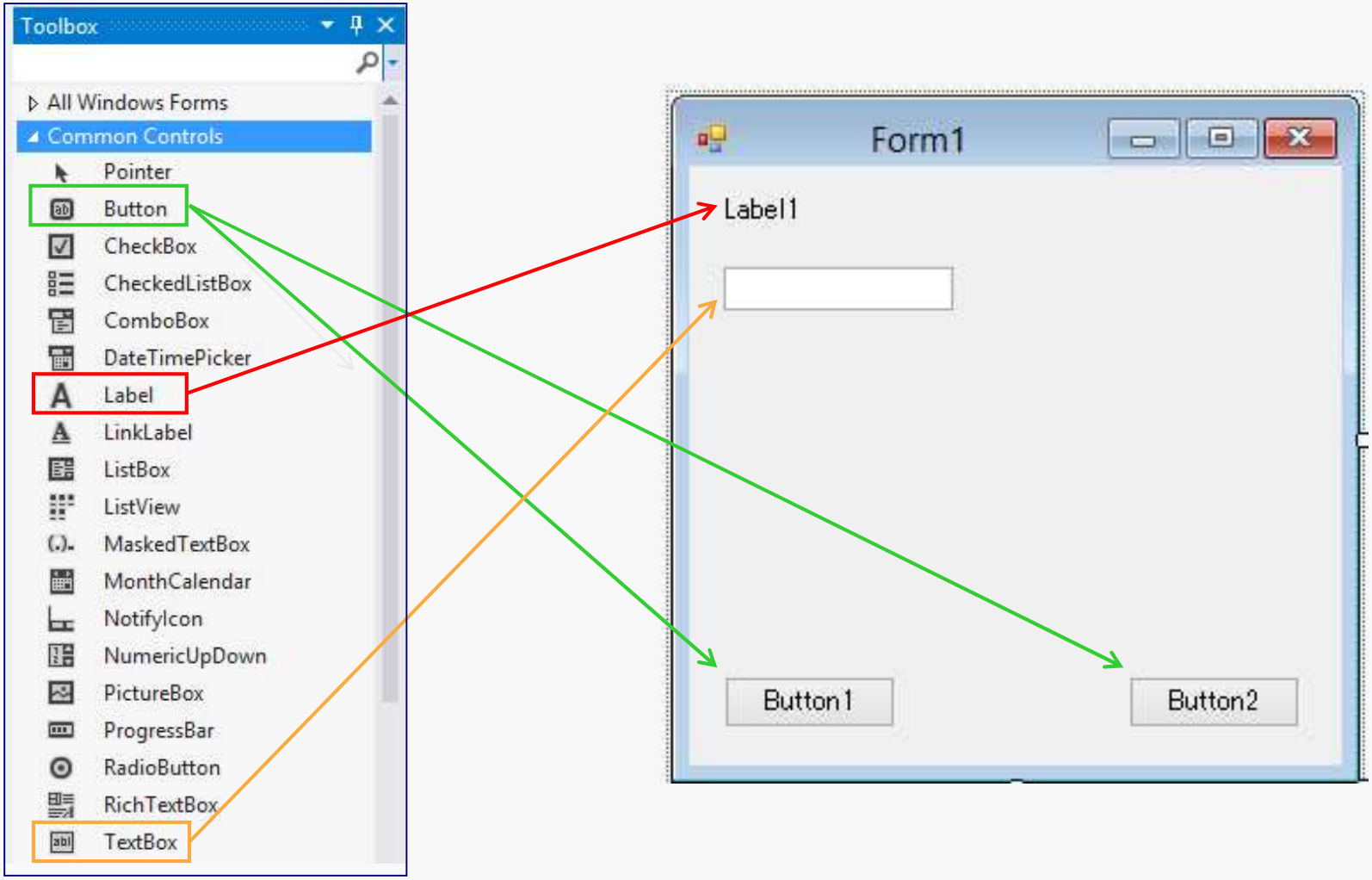
Step 1: Making the Project

- Click 'New Project' in the start screen to display the New Project Dialog:
  - Choose settings to make a VB Windows Form (WinForm) Project, as below:
  - Name your project 'Welcome Project'; also, keep your default location.



Step 2: Form and Controls Arrangement

- We now add 1 Label , 1 TextBox, and 2 Buttons to our form…

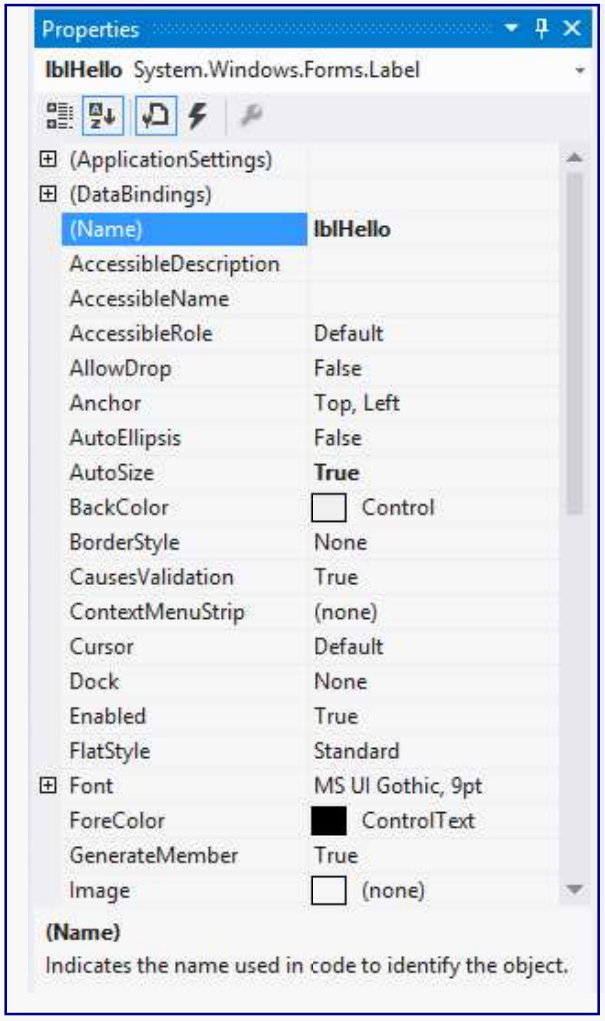


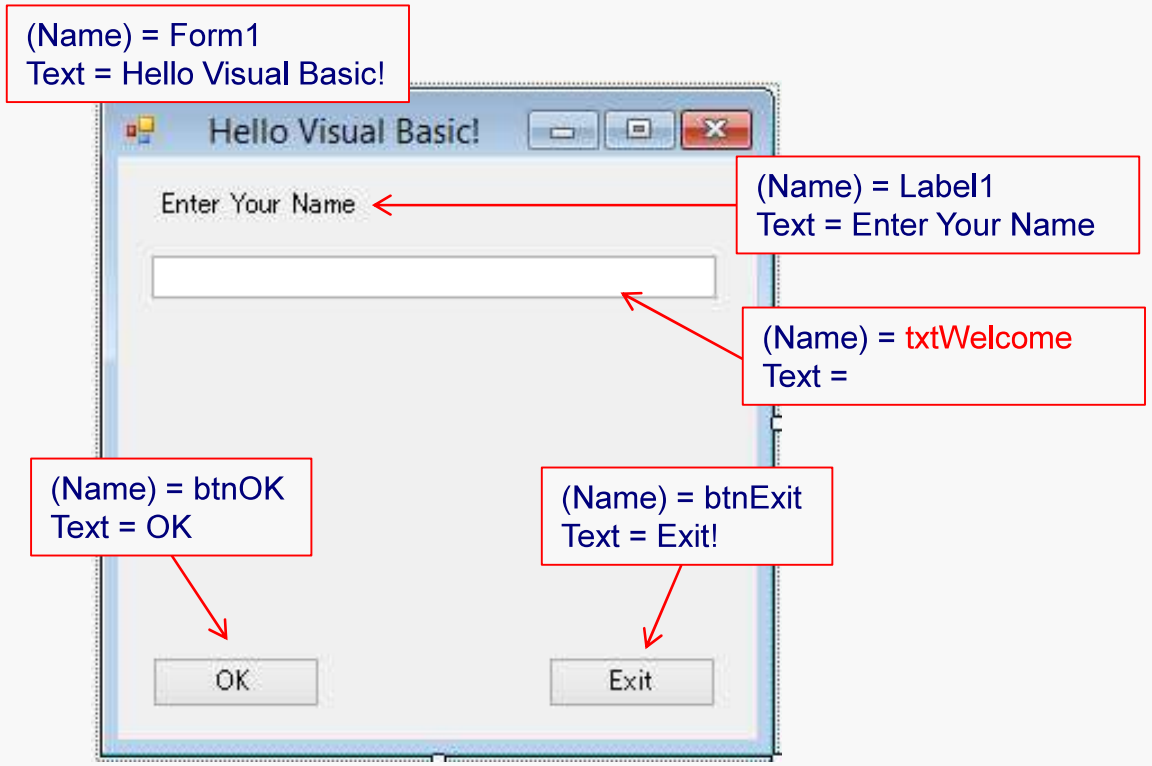
Step 3: Setting Control Properties

Adjust the Properties of each Control…

1. Select the desired control (by single-clicking)
   → Its Properties will be shown in the Properties Window

2. Click each desired Property, one by one.





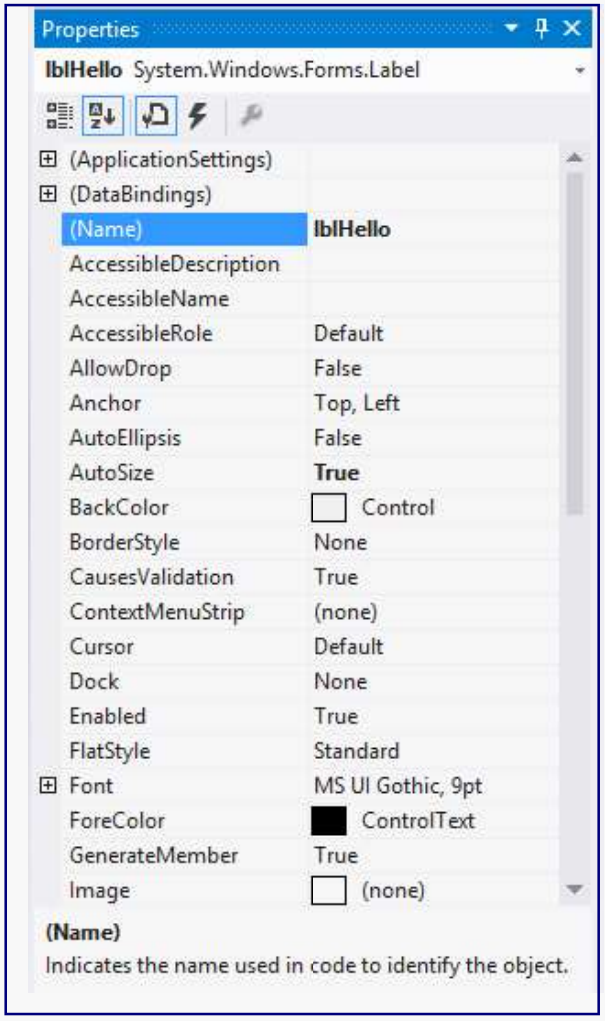
Notice the distinctive way we name our Controls…

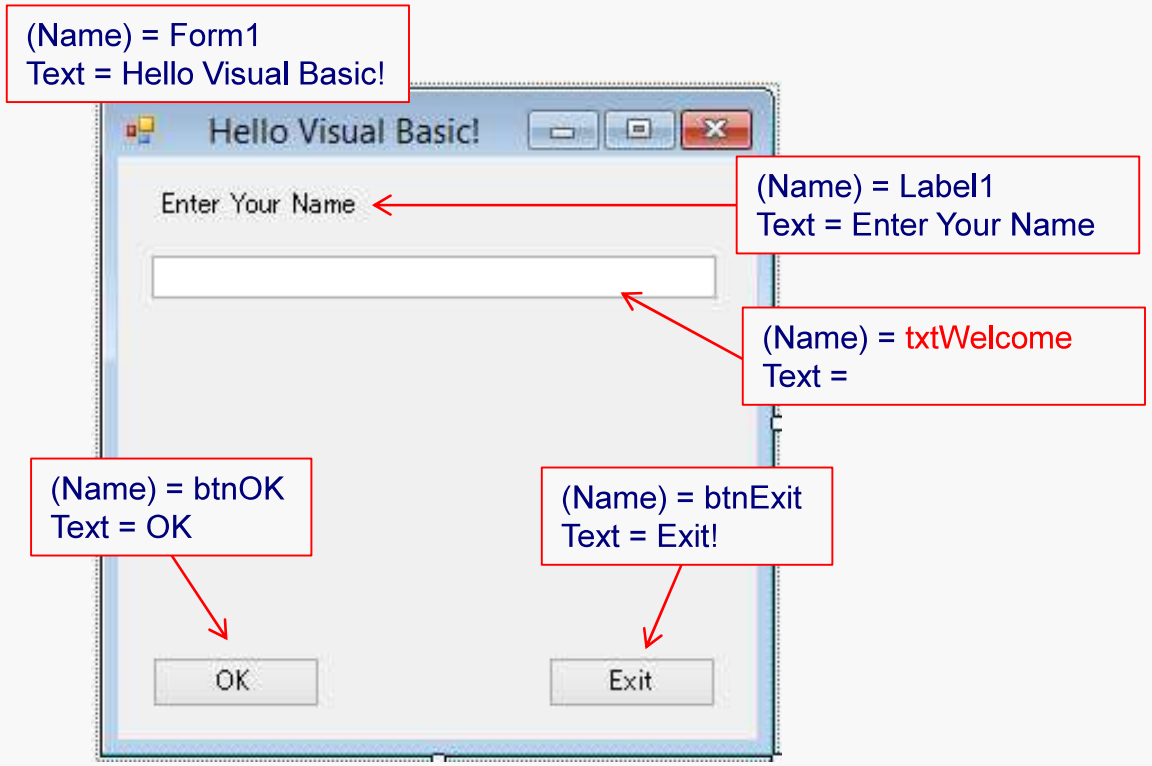
Step 3: Setting Control Properties

Adjust the Properties of each Control…

1. Select the desired control (by single-clicking)
   → Its Properties will be shown in the Properties Window

2. Click each desired Property, one by one.

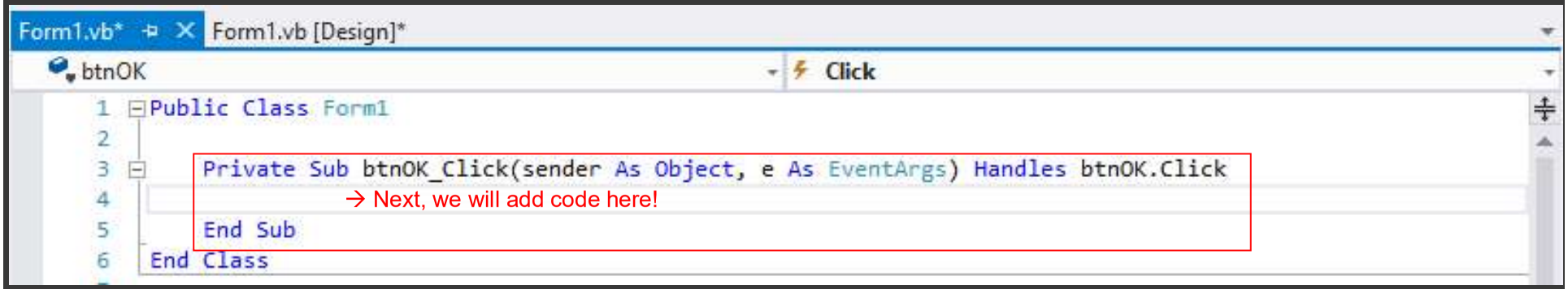




Notice the distinctive way we name our Controls…

Step 4: Adding Program Code

- Now, let's add the VB code to make the program work…
  - We do this by coding a Subroutine (mini-program) for each active Control.
    - By "active" we mean a Control that will be coded by us to respond to user input.
  - This type of Subroutine is called an Event Handler.
- Let's make our Program respond when a user clicks btnOK
  - To get started, just double-click the Control, btnOK in the Design Window...
    - This takes us to Code View and gives us an empty Event Handler (red box)



- Next, we will add VB code to Handle the Click Event of btnOK…
  - Events are identified as ControlName + . + EventName → btnOK.Click
  - Event Handers are named similarly, but using an underbar → btnOK_Click

Step 4 (cont.): Adding Program Code

- Now, add the VB code below to btnOK_Click:
  - Note: Any code we put it in btnOK_Click will execute whenever btnOK is pressed, one time from top to bottom.

```
Private Sub btnOK_Click(sender As Object, e As EventArgs) Handles btnOK.Click
    'Display a MessageBox greeting to our user
    MessageBox.Show("Hello " & txtWelcome.Text & ". Welcome to Visual Basic!", _
                    "Hello User Message")
End Sub
```

- Here, we will display a small MessageBox to welcome our user.
  - The 1st line (green text) is a comment statement, which does nothing.
  - The 2nd line, MessageBox.Show() accepts 2 arguments separated by a comma and a statement extender (_):

    1. Here, the 1st argument makes a String to hold our message, in 4 steps:
       a. First, we make a short String (“Hello “)
       b. The user’s name is then fetched from the Text Property of txtWelcome
       c. We then make a third String: ” Welcome to Visual Basic!”
       d. Our message is then made from these 3 Strings using the & operator.
    2. The 2nd argument specifies the text for the Title Bar of the MessageBox

Step 4 (cont.): Adding Program Code

- Next, let's add the VB code for btnExit_Click:
  - First, return to the Design Window (left tab), and double-click btnExit.
  - Next, add the code shown below to your new, empty Event Handler:

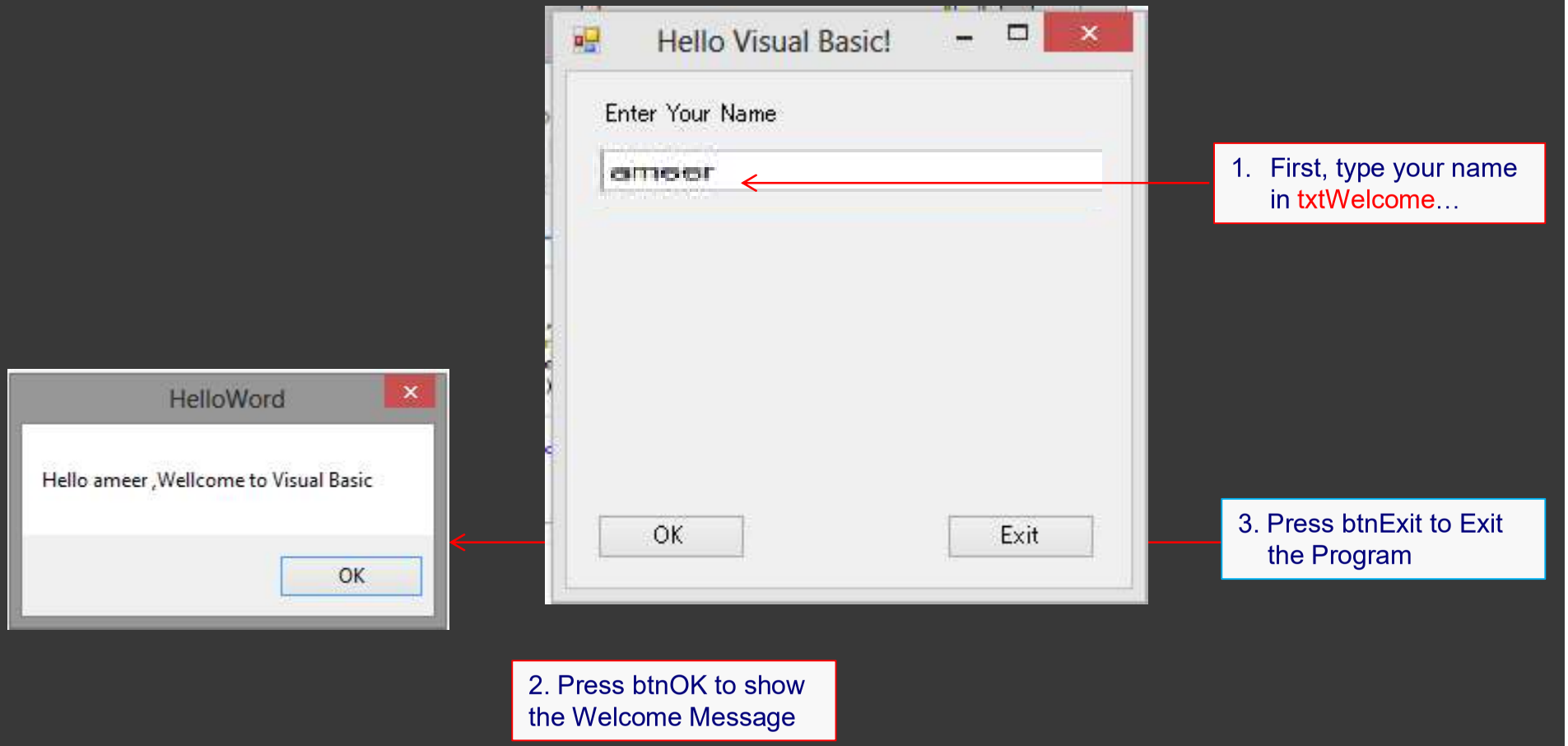
```
Private Sub btnExit_Click(sender As Object, e As EventArgs) Handles btnExit.Click
    End
End Sub
```

  - Here, we are coding to let the user exit our program by pressing btnExit.
    - Using a single VB Keyword → End

- This style of programming is known as "Event Driven Programming"
  - In this style, our program behaves like a simple automaton (robot)…
    - It waits for one of our defined user Actions to happen…
      1. User Clicks the OK button (btnOK)
      2. User Clicks the Exit button (btnExit)
    - Then, it responds to each action by executing the corresponding Handler.

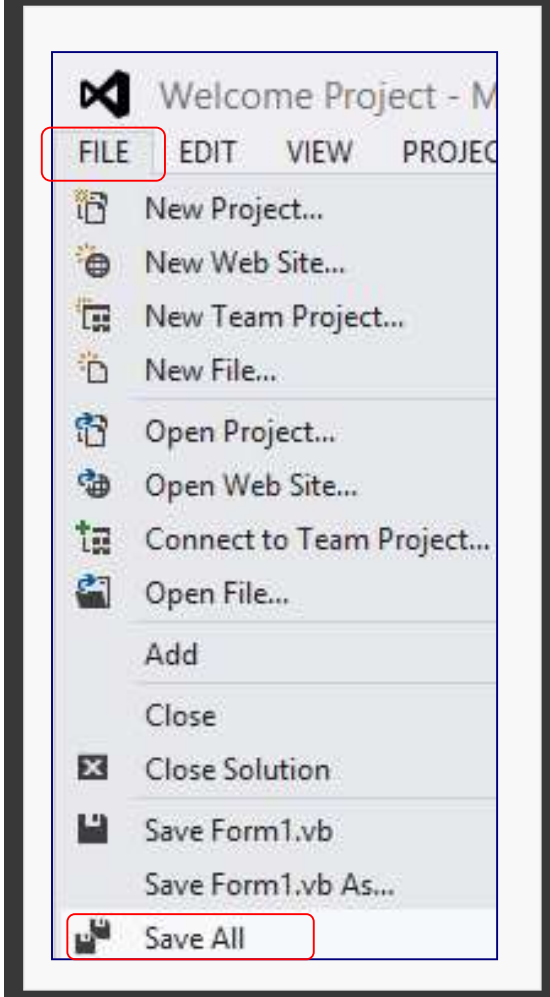
Step 5: Program Testing

- Click the green triangle (Start) to Compile and Run your Program:
  - Here, Compiling means taking your VB source code and converting it into a machine-usable form.
  - Then, test your program (as the User):

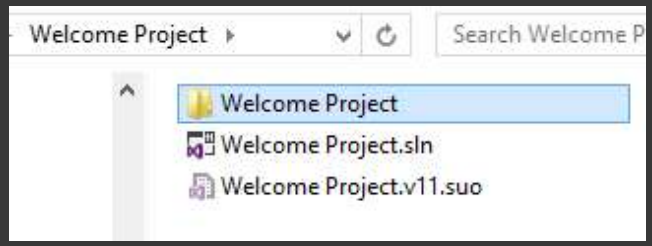


Step 6a: Saving the Program

- To save your Program (Visual Studio Solution):
  1. Select 'File' from the Visual Studio 2102 Main Menu…
  2. Select 'Save All' to save all files (note: there is no general-use 'Save As').



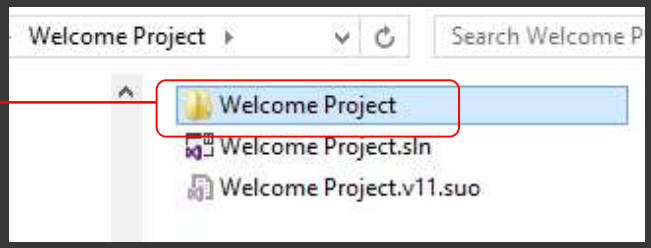
- To confirm, first check your Visual Studio Projects Folder:
  - MyDocuments > Visual Studio 2012 > MyProjects > Welcome Project

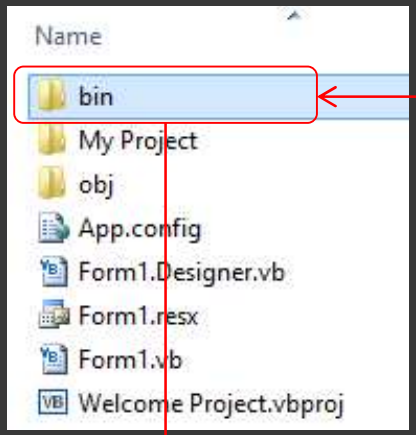


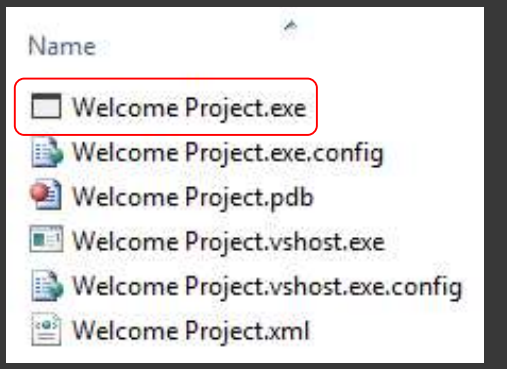
  - Here, you are in your 'Welcome Project' Solution Folder, and you will see :
    - The 'Welcome Project' folder is your Project Folder
      - Note: You have only 1 Project in this Solution.
    - 'Welcome Project.sln' is your Solution File
      - (This is the icon you click to open your solution in VS 2012)

Step 6b: Confirming the Save

- Next, let's find your executable file …
  - (= Welcome Project.exe)
  - First, enter your Project Folder…



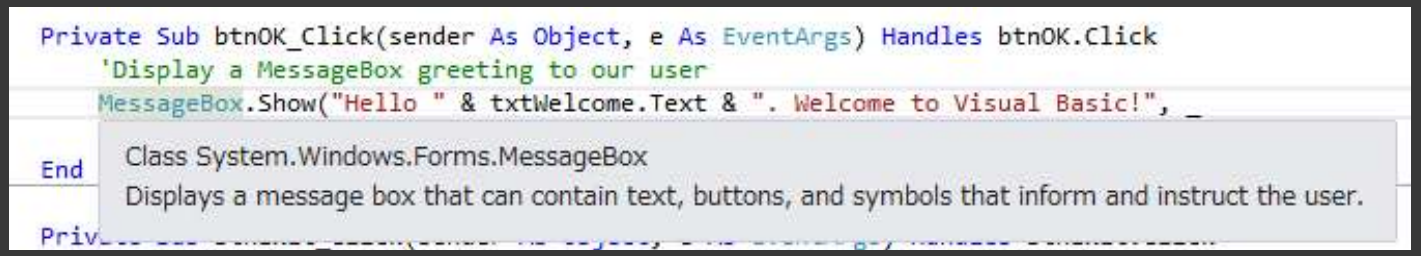




  - Then, enter your Project's bin folder to view your exe file.
    - You may run your program directly by clicking this icon…
      - Note: your Project will NOT open.
    - Or, more conveniently, from within Visual Studio (as usual).

Using Visual Studio Help

- Visual Studio 2012 features an extensive set of Help Tools, including:
  - A Window-based Help System allowing you to view documentation;
  - An intelligent, programmable tool-tip based system called Intellisense



- You will become familiar with Intellisense as you gain experience; however, be aware that you may access the VS Help Window in several ways:
  1. Through the Visual Studio Main Menu (> Help > View Help):