1.1 Introduction

- A program is a set of instructions that a computer follows to perform a task
  - Programs are commonly referred to as software
  - Without software, computers cannot do anything
- Programmers, or software developers, create software
  - They are people with the training and skills necessary to design, create, and test programs
- This book introduces fundamental programming concepts using C#

AUBURN UNIVERSITY

1.2 Hardware and Software

- Hardware refers to all physical devices
  - A computer consists of many pieces of hardware that all work together
  - Each piece of hardware does its own work
- A typical computer system contains:
  - The CPU
  - Main memory
  - Secondary storage devices
  - Input devices
  - Output devices

AUBURN UNIVERSITY

1.2 The CPU

- The central processing unit is the part that actually runs programs
  - The most important part of a computer
  - Today CPUs are microprocessor
  - Commonly used CPU vendors are Intel and AMD

AUBURN UNIVERSITY

1.2 Main Memory

- The computer's work area
- Where the computer loads instructions of programs and data for processing
- Commonly known as RAM, random-access memory
  - Designed for CPUs to quickly access data stored at any random location in the RAM
- They are a volatile type of memory
  - When the computer is powered off, the contents in RAM are erased

1.2 Secondary Storage Devices

AUBURN UNIVERSITY

- Devices that can hold data for long periods of time, even when the power is off
  - Where important data and system files are stored
  - Most commonly used is the **disk drive** which stores data by magnetically encoding it onto a circular disk
  - **Solid state drives** have no moving parts, and operate faster than a traditional disk drive.
  - Optical devices such as **DVD-ROMs** and **CD-ROMs** are also popular
  - **USB drives** and **SD memory cards** are small devices that plug into

1.2 Input Devices

- Input is any data the computer collects from people and devices
- Devices that collect the data and send them to the computer are called input devices
- Commonly used input devices are touch screens, keyboards, mouses, scanners, microphones, and digital cameras

AUBURN UNIVERSITY

1.2 Output Devices

- Output is any data the computer produces for people or devices
- The device that generates output for a computer is called an output device
- Commonly used output devices are screens, speakers, and printers

AUBURN UNIVERSITY

1.2 Software

- Categorized mainly into system software and application software
  - System software are programs that control and manage the basic operations of a computer. Subcategories are:
    - Operating systems
    - Utility programs
    - Software development tools
  - Application software are programs that perform special tasks

AUBURN UNIVERSITY

1.3 How Computers Store Data

- All data stored in a computer is converted to sequence of 0s and 1s; each sequence is called a bit
- A computer's memory is divided into tiny storage locations called bytes
  - Eight bits make a byte
- Combinations of bits, 0s and 1s, are used to represent characters. For example,
  - The character 'A' is 65 in ASCII code, which is converted to binary format 1000001
  - When you press 'A', 1000001 will be stored in computer's memory

AUBURN UNIVERSITY

1.3 Digital and Digital Data

- "Digital" refers to anything that can only have two possible values
  - Digital data is the data that is stored in binary.
  - Digital devices are devices that work with binary data
  - Computers are digital devices

AUBURN UNIVERSITY

1.4 How a Program Works

- CPU reads instructions written in machine language called instruction set
- A program will be copied into memory for CPU to execute
- CPU uses the "fetch-decode-execute" cycle for processing
  - Fetch: Reads instructions from memory
  - Decode: Decodes the instructions that were just read to determine how to perform operations
  - Execute: Actually performs the operations

AUBURN UNIVERSITY

1.4 Programming Languages

AUBURN UNIVERSITY

- Machine languages: sequences of 0s and 1s
- Assembly languages: use short words known as "mnemonics" to write program
  - Must be translated by assembler
  - Still considered low-level languages
- High-level languages: more human readable languages that allow programmers to create programs without knowing how CPU works.
  - Modern languages are high-level

1.4 Keywords, Operators, and Syntax

AUBURN UNIVERSITY

- High level languages use keywords that have special meaning and cannot be used for any purpose other than to write programs
- Operators are keywords that represent special program functions such as addition, subtraction, and multiplication
- Syntax is a set of rules that must be strictly followed to write computer-understandable instructions
  - Syntax error is a mistake or violation of these rules
- Each instruction in a program is

1.4 Compilers and Interpreters

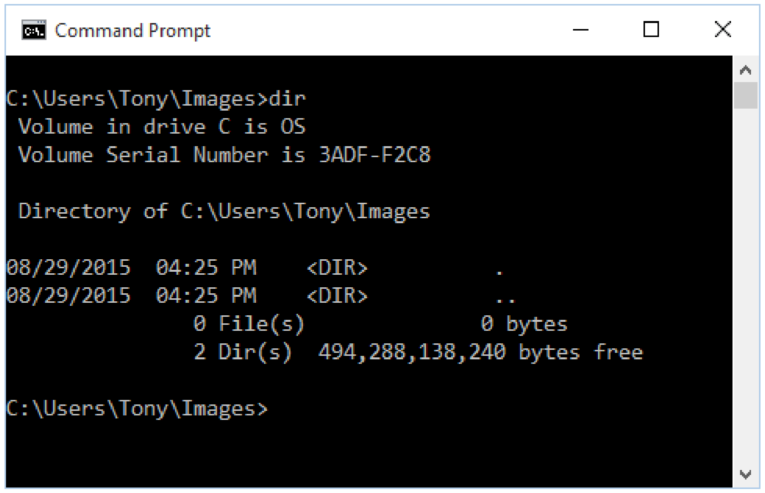
- A compiler translates a high-level language program to a separate machine program for CPU to read and execute
  - Source code: the statements a programmer writes in a high-level language
  - Compilation: translates a text-based source code to binary codes
- Interpreter reads, translates, and executes the instructions of a high-level language program
  - Examples are PHP, Perl, Python, and ASP.NET

AUBURN UNIVERSITY

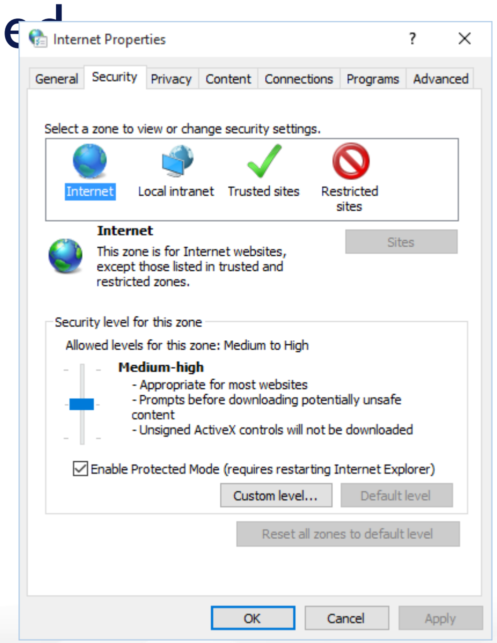
1.5 Graphical User Interface

AUBURN UNIVERSITY

- User interfaces allow users to interact with the computer. Categories are:
  - Command line interface (aka console interface)
  - Graphical user interface (GUI)- now the most commonly use



```
C:\Users\Tony\Images>dir
 Volume in drive C is OS
 Volume Serial Number is 3ADF-F2C8

 Directory of C:\Users\Tony\Images

08/29/2015  04:25 PM    <DIR>          .
08/29/2015  04:25 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  494,288,138,240 bytes free

C:\Users\Tony\Images>
```



Select a zone to view or change security settings.

Internet

This zone is for Internet websites, except those listed in trusted and restricted zones.

Security level for this zone

Allowed levels for this zone: Medium to High

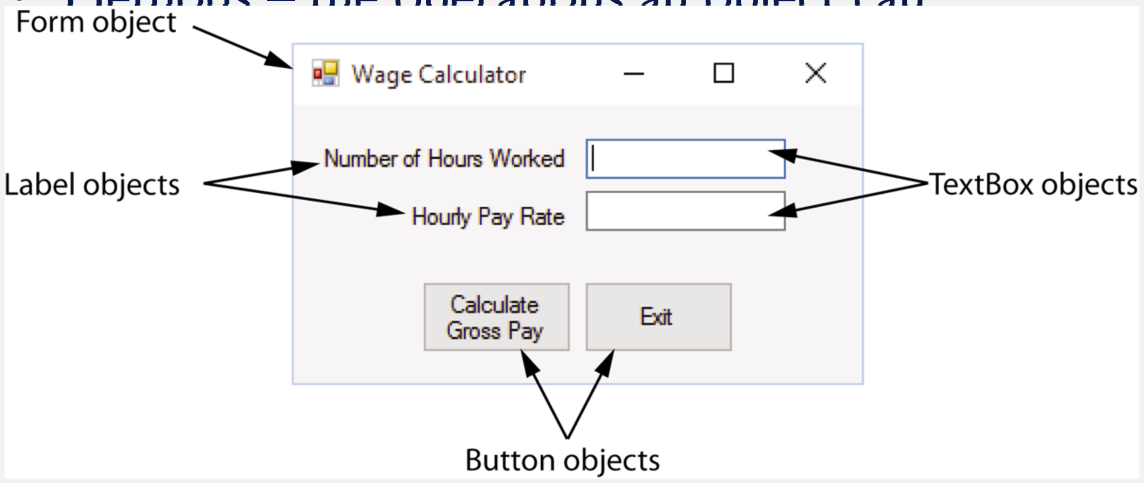
Medium-high

- Appropriate for most websites
- Prompts before downloading potentially unsafe content
- Unsigned ActiveX controls will not be downloaded

☑ Enable Protected Mode (requires restarting Internet Explorer)

1.6 Objects

- Most programming languages use object-oriented programming in which a program component is called an "object"
- Program objects have properties (or fields) and methods
  - Properties – data stored in an object
  - Methods – the operations an object can



1.6 Controls

- Objects that are visible in a program GUI are known as controls
  - Commonly used controls are Labels, Buttons, and TextBoxes
  - They enhance the functionality of your programs
- There are invisible objects in a GUI such as Timers, and OpenFileDialog

AUBURN UNIVERSITY

1.6 The .NET Framework

- The .NET Framework is a collection of classes and other codes that can be used to create programs for Windows operating system
- C# is a language supported by the .NET Framework
- Controls are defined by specialized classes provided by the .NET Framework
- You can also write your own class to perform a special task

1.7 The Program Development Process

- The process of creating a program is known as the programming development cycle
- It has six phases:
  - Understand the program's purpose
  - Design the GUI
  - Design the program's logic
  - Write the code
  - Correct syntax errors
  - Test the program and correct logic errors

1.7 Algorithm, Pseudocode, Flowchart

- An algorithm is a set of well-defined, logical steps that must be taken to perform a task
- An algorithm that is written out in plain English is called pseudocode
- A flowchart is a diagram that graphically depicts the steps of an algorithm

AUBURN UNIVERSITY

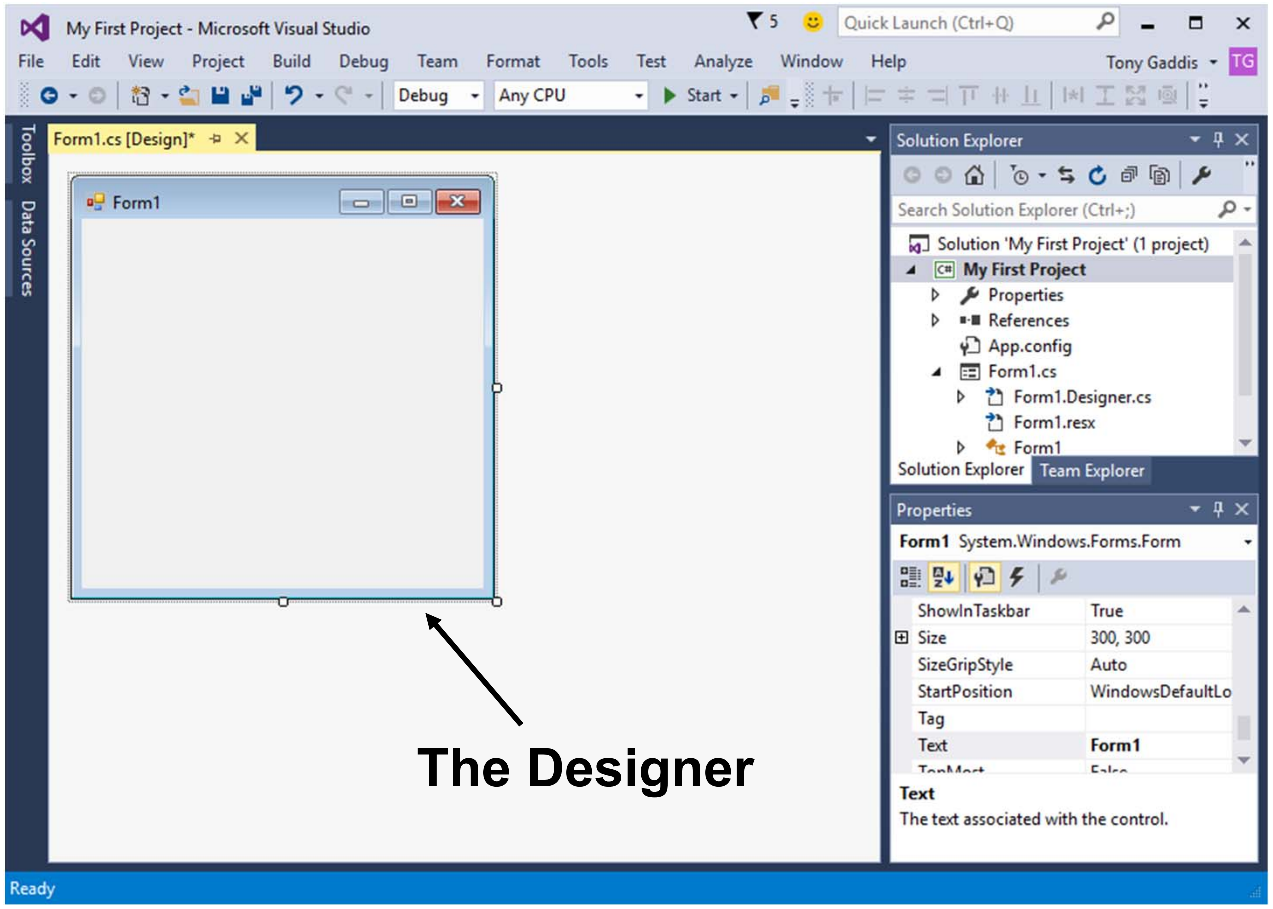
1.8 Getting Started with the Visual Studio Environment

- Visual Studio 2012 is a professional integrated development environment (IDE)
- The Visual Studio Environment includes:
  - Designer Window
  - Solution Explorer Window
  - Properties Window

AUBURN UNIVERSITY

1.8 Getting Started with the Visual Studio Environment

AUBURN UNIVERSITY


My First Project - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Test Analyze Window Help
Debug
Any CPU
Start
Form1.cs [Design]*
Form1
Solution Explorer
Solution 'My First Project' (1 project)
My First Project
Properties
References
App.config
Form1.cs
Form1.Designer.cs
Form1.resx
Form1
Properties
Ready

Form1 System.Windows.Forms.Form

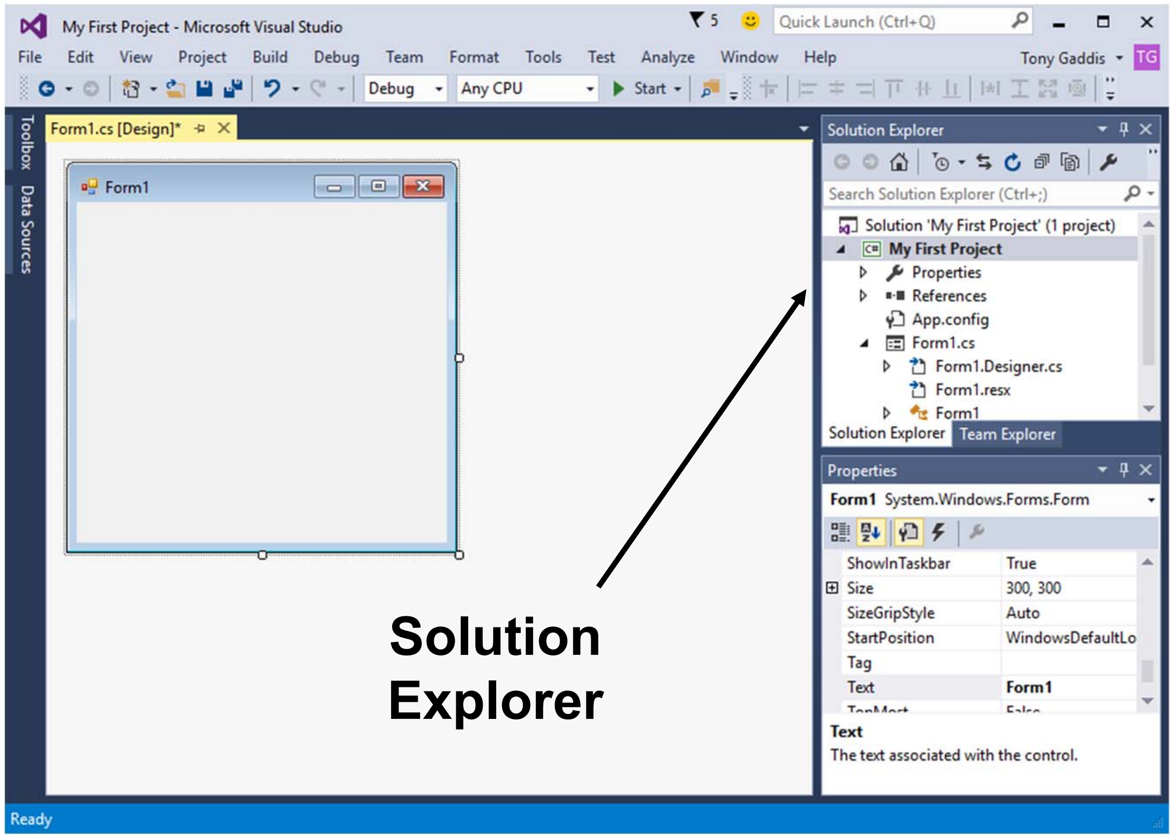
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLo
Tag	
Text	Form1

Text
The text associated with the control.

The Designer

1.8 Getting Started with the Visual Studio Environment

AUBURN UNIVERSITY

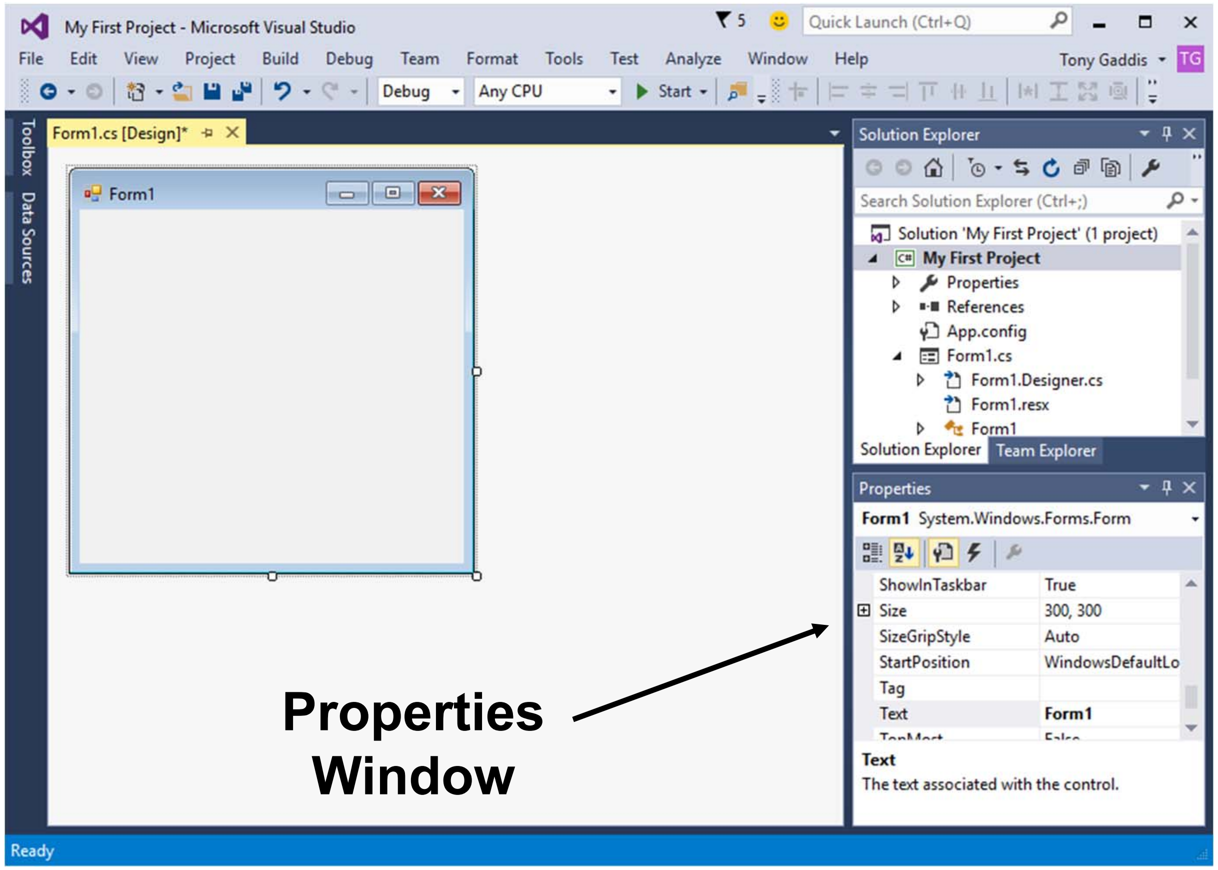

My First Project - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Test Analyze Window Help
Debug
Any CPU
Start
Quick Launch (Ctrl+Q)
Tony Gaddis
Toolbox
Data Sources
Form1.cs [Design]*
Form1
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'My First Project' (1 project)
My First Project
Properties
References
App.config
Form1.cs
Form1.Designer.cs
Form1.resx
Form1
Team Explorer
Properties
Form1 System.Windows.Forms.Form
Ready

ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLo
Tag	
Text	Form1

Text
The text associated with the control.

Solution Explorer

1.8 Getting Started with the Visual Studio Environment


My First Project - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Test Analyze Window Help
Debug
Any CPU
Start
Form1.cs [Design]*
Form1
Solution Explorer
Solution 'My First Project' (1 project)
My First Project
Properties
References
App.config
Form1.cs
Form1.Designer.cs
Form1.resx
Form1
Team Explorer
Properties
Form1 System.Windows.Forms.Form
Ready

ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLo
Tag	
Text	Form1

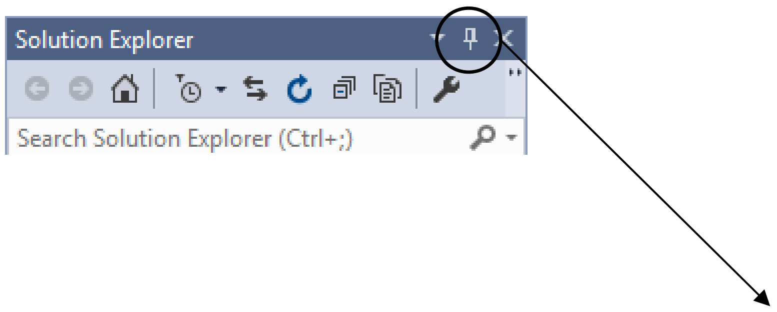
Text
The text associated with the control.

Properties Window

1.8 Getting Started with the Visual Studio Environment

AUBURN UNIVERSITY

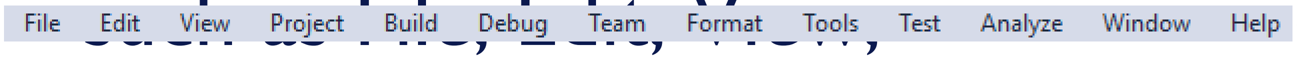
- Auto Hide allows a window to display only as a tab of the edges

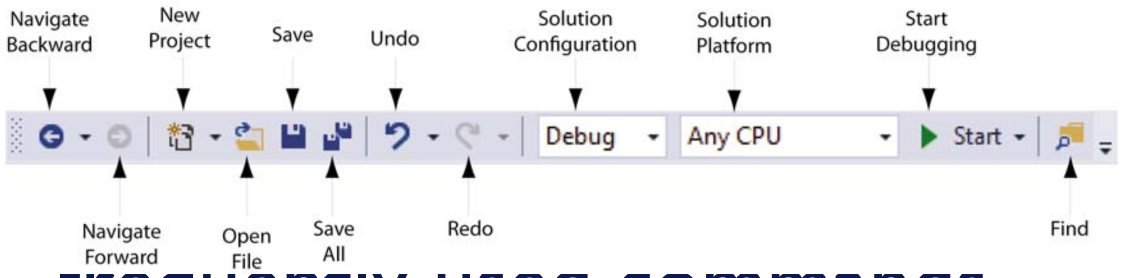


Pushpin icon means it supports Auto Hide

1.8 Menu Bar and Standard Toolbar

- Menu bar provides menus such as File, Edit, View, Project, etc.





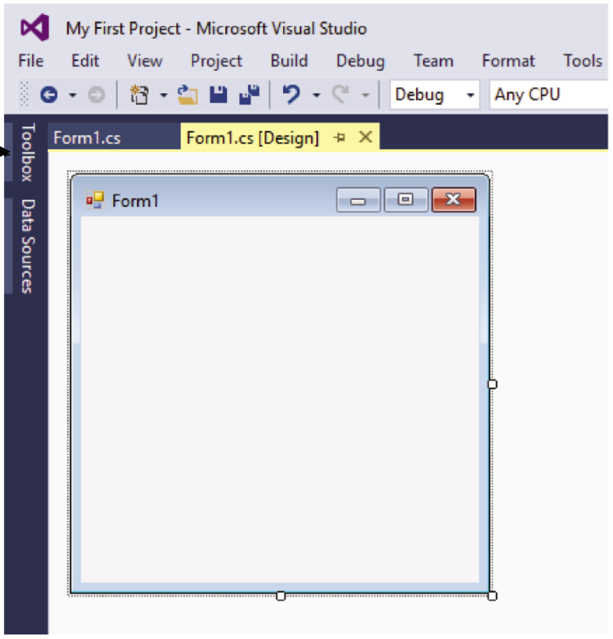
frequently used commands

AUBURN UNIVERSITY

1.8 The Toolbox

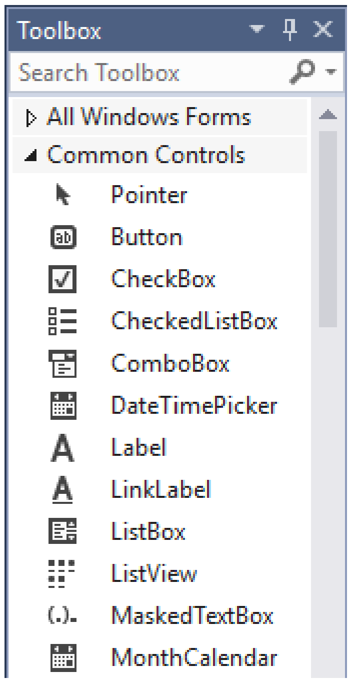
- Toolbox is a window for selecting controls to use in an application
  - Typically appears on the left side of Visual Studio environment
  - Usually in Auto Hide mode

Toolbox tab



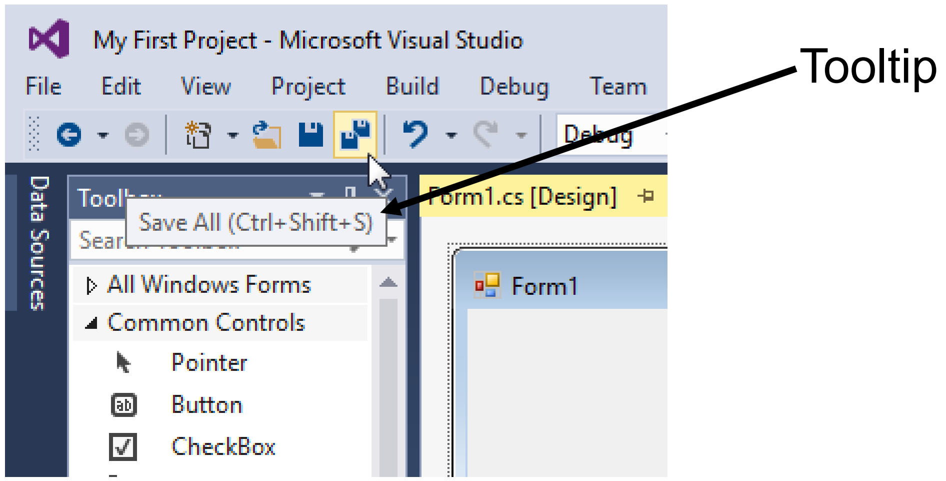
1.8 The Toolbox

- Toolbox is a window for selecting controls to use in an application
  - Divided into sections such as "All Windows Forms" and "Common Controls"



1.8 Tooltips

- A Tooltip is a small box that pops up when you hover the mouse pointer over an item on the toolbar or toolbox.



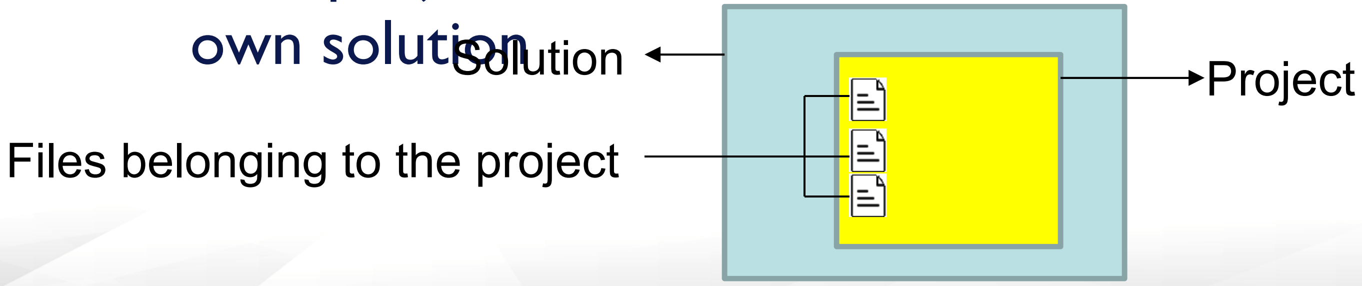
1.8 Docked and Floating Windows

- When a window such as Solution Explorer is docked, it is attached to one of the edges of the Visual Studio environment
- When a window is floating, you can click and drag it around the screen
  - A window cannot float if it is in Auto Hide mode
- Right click a window's title bar and select Float or Dock to change between them

1.8 Projects and Solutions

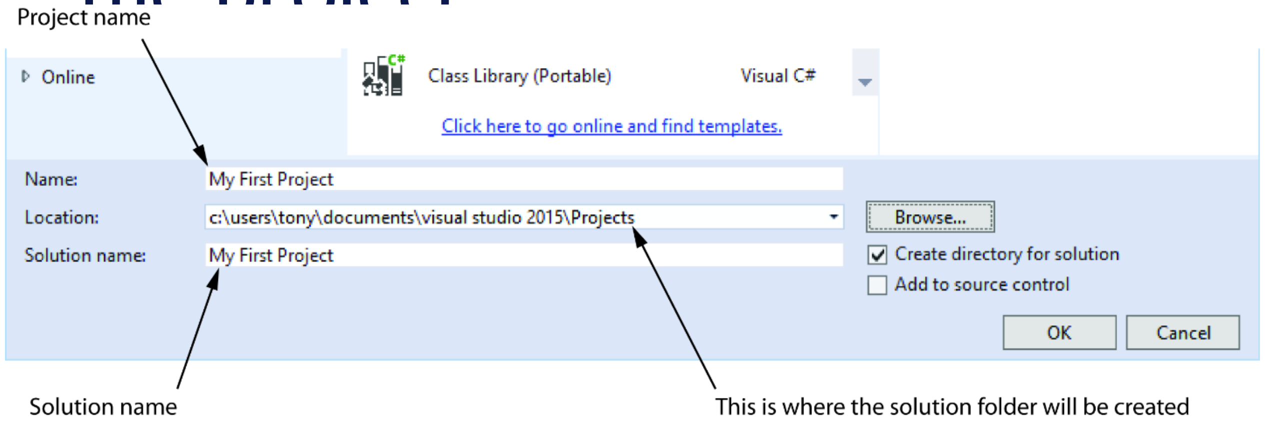
AUBURN UNIVERSITY

- Each Visual Studio application (including Visual C#) you will create is a project
  - A project contains several files.
  - Typically they are Form1.cs, Program.cs, etc.
- A solution is a container that can hold one or more Visual Studio (including Visual C#) projects
  - Each project, however, is saved in its own solution



1.8 Specifying the Project Name

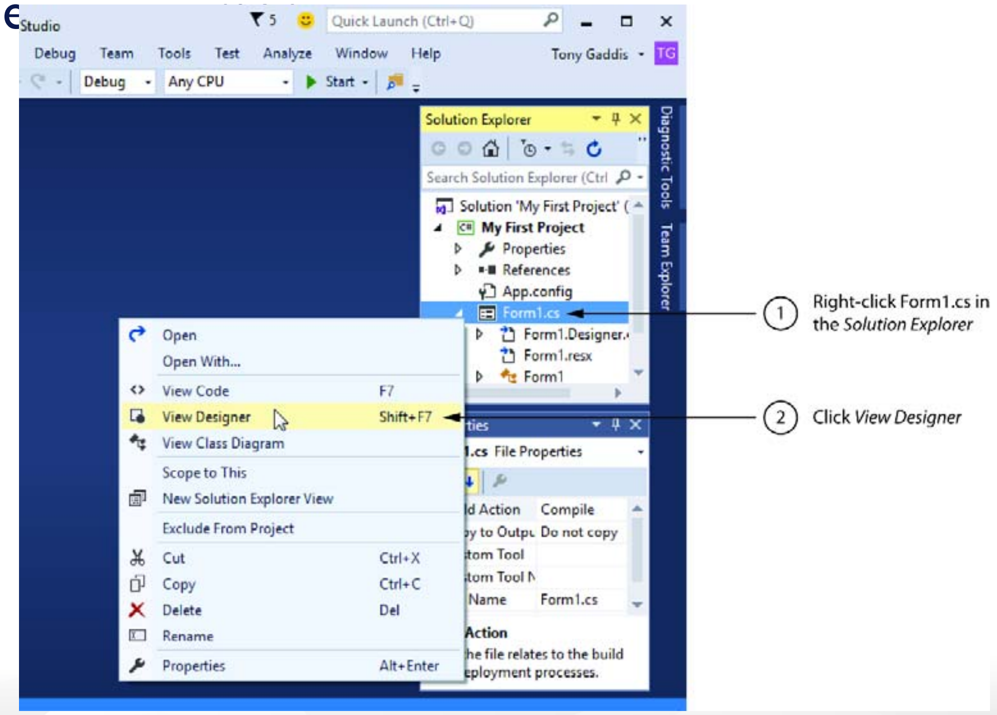
- You can specify the project name the first time you save the project


Project name
Online
Class Library (Portable)
Visual C#
Click here to go online and find templates.
Browse...
OK
Cancel
Solution name
This is where the solution folder will be created

Name: My First Project
Location: c:\users\tony\documents\visual studio 2015\Projects
Solution name: My First Project
[✓] Create directory for solution
[ ] Add to source control

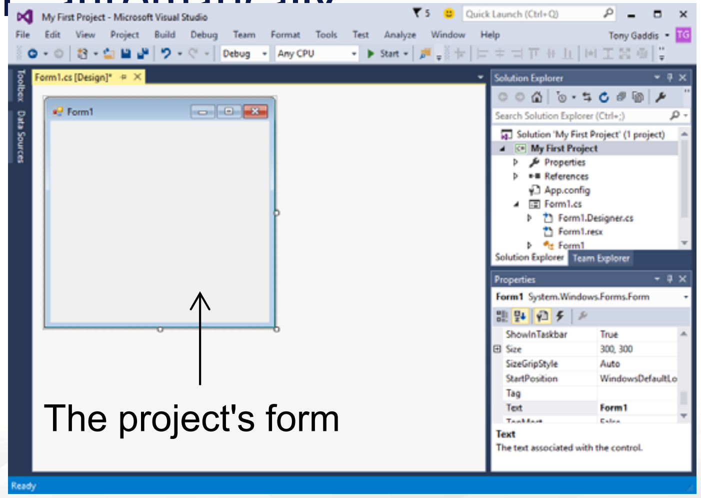
1.8 Displaying the Designer

- Sometimes when you open an existing project, the project's form will not be automatically displayed in the Designer
- You should:
  - Right click Form1.cs in the Solution Explorer
  - Click View Designer in the



2.1 Getting Started with Forms and Controls

- A Visual C# application project starts with creating its GUI with
  - Designer
  - Toolbox
  - Property window
- In the Designer, an empty form is automatically created
  - An application's GUI is made of forms and controls
  - Each form and control in the application's GUI must have a name as ID. The default blank form is named "Form1" automatically



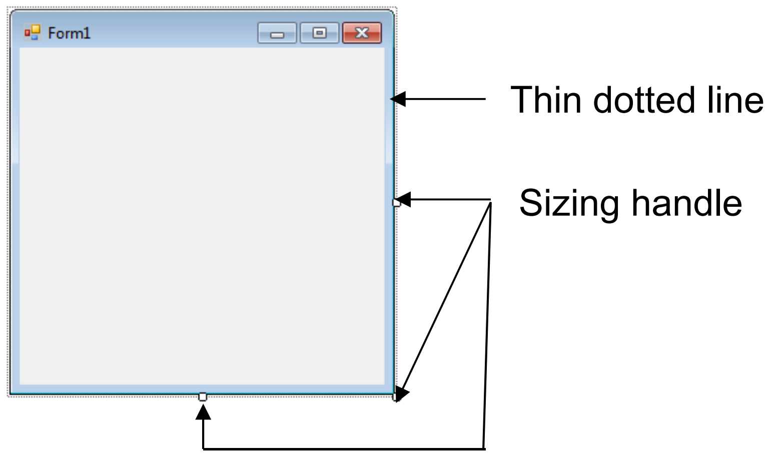
Form1 System.Windows.Forms.Form

ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLo
Tag	
Text	Form1

Text
The text associated with the control.

2.1 Form's Bounding Box and Sizing Handles

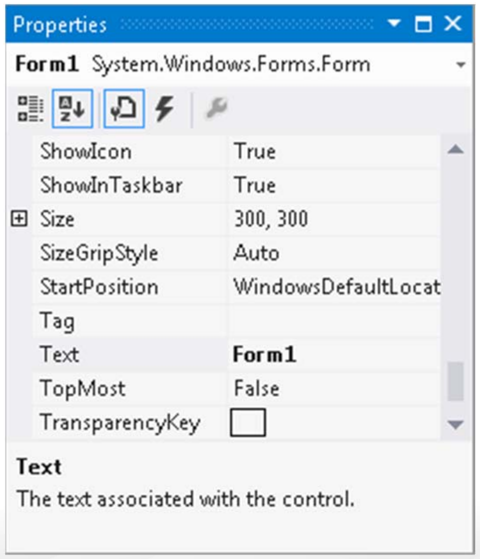
- The default empty form has a dimension (size) of 300 pixels wide by 300 pixels high
- A form in Designer is enclosed with thin dotted lines called the bounding box
- The bounding box has small sizing handles; you can use them to resize the form



2.1 The Property Window

AUBURN UNIVERSITY

- The appearance and other characteristics of a GUI object are determined by the object's properties
- The Properties window lists all properties
  - When selecting an object, its properties are displayed in Properties windows
  - Each property has 2 columns:
    - Left: property's name
    - Right: property's value



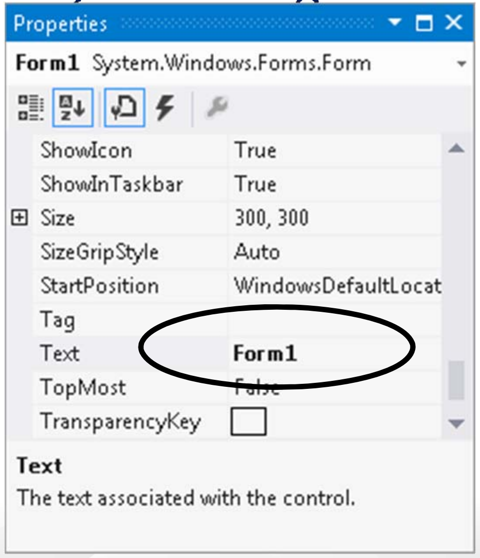
Form1 System.Windows.Forms.Form

ShowIcon	True
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocat
Tag	
Text	**Form1**
TopMost	False
TransparencyKey	☐

Text
The text associated with the control.

2.1 Changing a Property's Value

- Select an object, such as the Form, by clicking it once
- Click View and select Properties if the Properties window is not available
- Find the property's name in the list and change its value
  - The Text property determines the text to be displayed in the form's title bar
  - Example: Change the value from "Form1" to "My First Program"

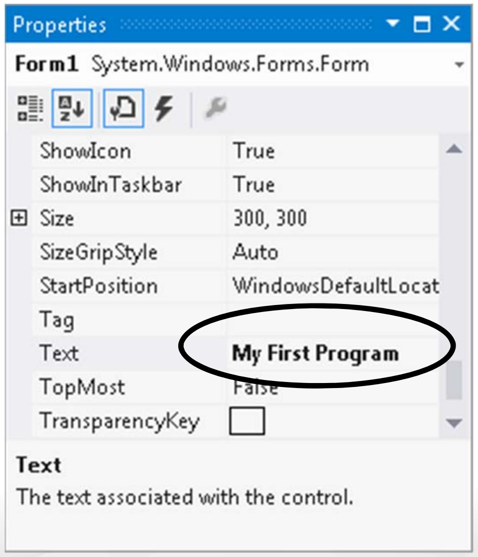


Properties

Form1 System.Windows.Forms.Form

ShowIcon	True
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocat
Tag	
Text	Form1
TopMost	False
TransparencyKey	

Text
The text associated with the control.



Properties

Form1 System.Windows.Forms.Form

ShowIcon	True
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocat
Tag	
Text	My First Program
TopMost	False
TransparencyKey	

Text
The text associated with the control.

2.1 Adding Controls to a Form

- In the Toolbox, select the Control (e.g. a Button), then you can either:
  - double click the Button control
  - click and drag the Button control to the form
- On the form, you can
  - resize the control using its bounding box and sizing handles
  - move the control's position by dragging it
  - change its properties in the Properties window

2.1 Rules for Naming Controls

- Controls' name are identifiers of the controls
- The naming conventions are:
  - The first character must be a letter (lower or uppercase does not matter) or an underscore (_)
  - All other characters can be alphanumerical characters or underscores
  - The name cannot contain spaces
- Examples of good names are:

```
showDayButton
DisplayTotal
_ScoreLabel
```

2.2 Creating the GUI for Your First Visual C# Application

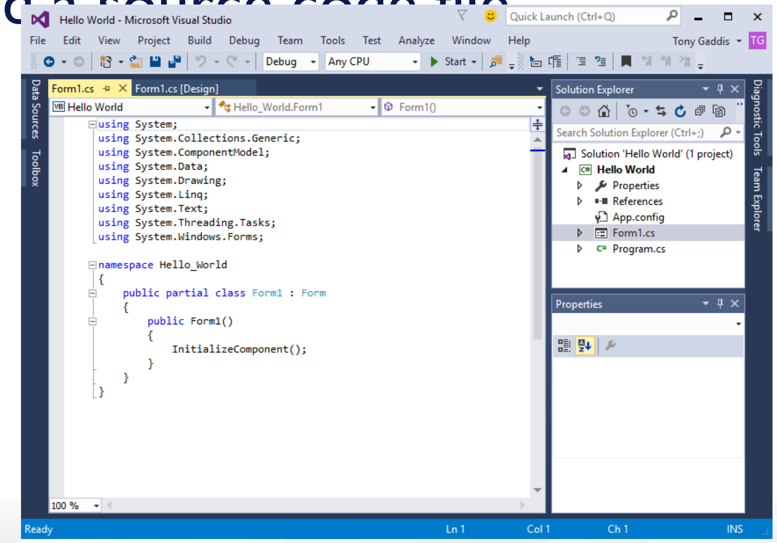
- Components: a Form and a Button control
- Purpose:
  - Create the application's GUI
  - Write the code that causes "Hello World" to appear when the user clicks the button (details are available in section 2.3)

AUBURN UNIVERSITY

2.3 Introduction to C# Code

AUBURN UNIVERSITY

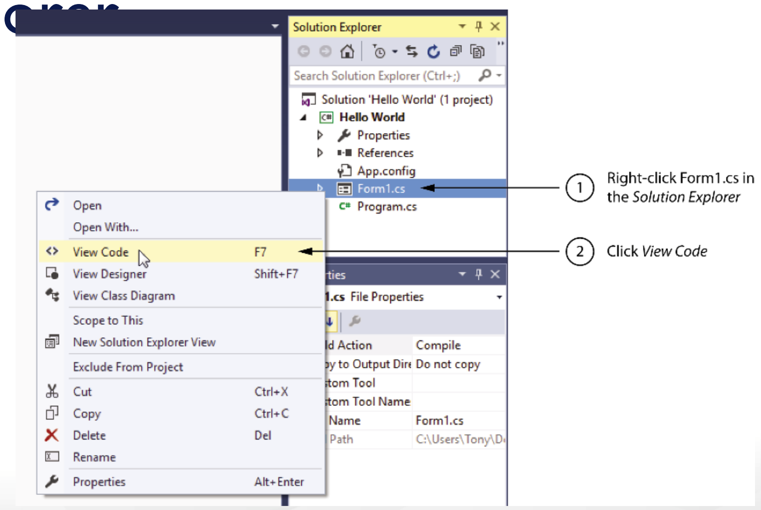
- C# code is primarily organized in three ways:
  - Namespace: a container that holds classes
  - Class: a container that holds methods
  - Method: a group of one or more programming statements that perform some operations
- A file that contains program code is called a source code file



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Hello_World
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

2.3 Source Codes in the Solution Explorer

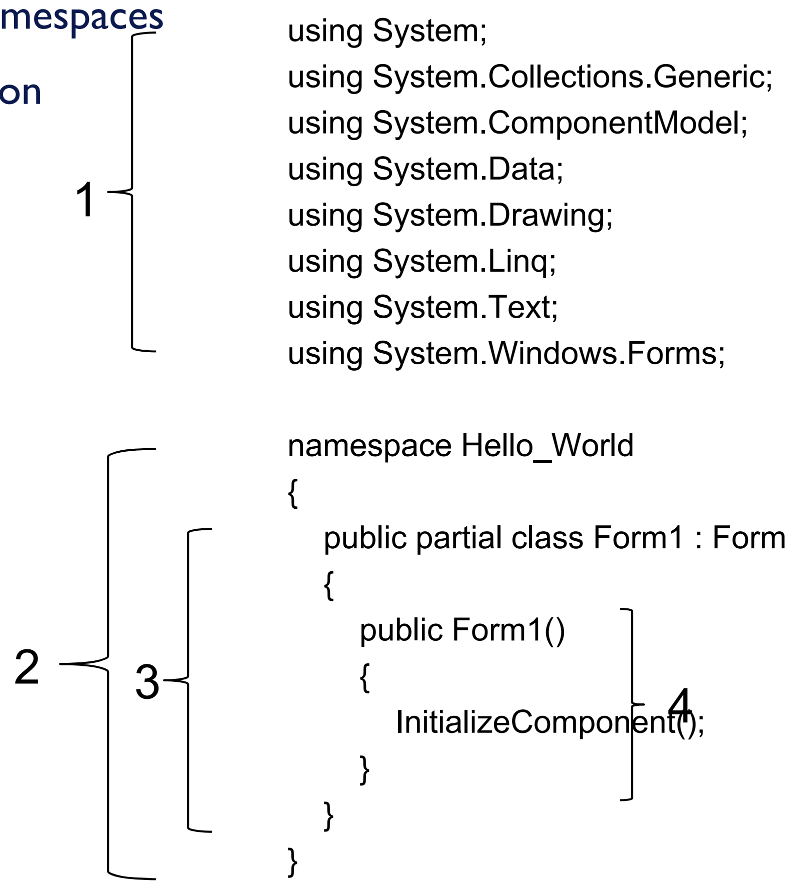
- Each time a new project is created the following two source code files are automatically created:
  - Program.cs file: contains the application's start-up code to be executed when the application runs
  - Form1.cs contains code that is associated with the Form1 form
- You can open them through the Solution Explorer



2.3 Organization of the Form1.cs

- A sample of Form1.cs:
  1. The using directives indicate which namespaces of .NET Framework this program will use.
  2. The user-defined namespace of the project not .NET Framework namespaces
  3. Class declaration
  4. A method

- C# code is organized as methods, which are contained inside classes, which are contained inside namespaces



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Hello_World
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

2.3 Adding Your Code

AUBURN UNIVERSITY

- GUI applications are event-driven which means they interact with users
- An event is a user's action such as mouse clicking, key pressing, etc.
- Double clicking a control, such as Button, will link the control to a default Event Handler
  - An event handler is a method that executes when a specific event takes place
  - A code segment similar to the following will be created automatically:

```
private void myButton_Click(object
  sender, EventArgs e)
{


}
```

2.3 Message Boxes

AUBURN UNIVERSITY

- A message box (aka dialog box) displays a message
- The .NET Framework provides a method named MessageBox.Show
  - C# can use it to pop up a window and display a message. A sample code is (bold line):

```
private void myButton_Click(object sender,
  EventArgs e)
{
   MessageBox.Show("Thanks for clicking the
  button!");
}
```

  - Placing it in the myButton_Click event handler can display the string in the message box when the button is clicked

2.4 Writing Code for the Hello World Application

AUBURN UNIVERSITY

- The completed source code of Form1.cs is:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Hello_World
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void myButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Thanks for clicking the button!");
        }
    }
```

2.5 Label Controls

- A Label control displays text on a form and can be used to display unchanging text or program output
- Commonly used properties are:
  - Text: gets or sets the text associated with Label control
  - Name: gets or sets the name of Label control
  - Font: allows you to set the font, font style, and font size
  - BorderStyle: allows you to display a border around the control's text
  - AutoSize: controls the way they can be resized
  - TextAlign: set the text alignments

AUBURN UNIVERSITY

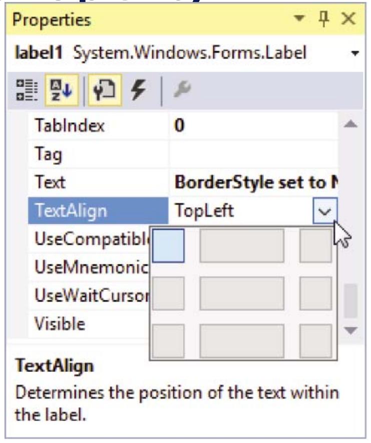
2.5 Handling Text Alignments

AUBURN UNIVERSITY

- The TextAlign property supports the following values:

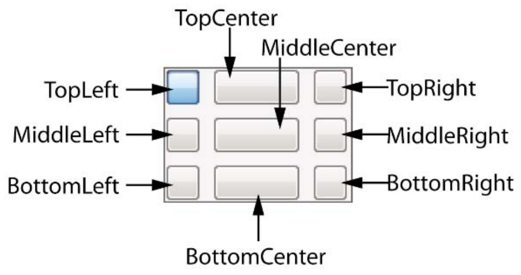
TopLeft	TopCenter	TopRight
MiddleLeft	MiddleCenter	MiddleRight
BottomLeft	BottomCenter	BottmRight

- You can select them by clicking the down-arrow button of the TextAlign property


Properties
label1 System.Windows.Forms.Label

TabIndex	0
Tag	
Text	BorderStyle set to N
TextAlign	TopLeft
UseCompatibl	
UseMnemonic	
UseWaitCursor	
Visible	

TextAlign
Determines the position of the text within the label.



2.5 Using Code to Display Output in a Label Control

AUBURN UNIVERSITY

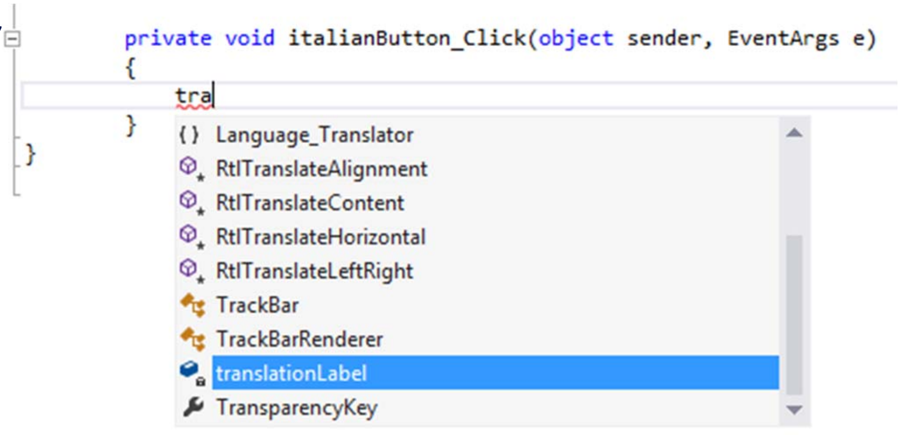
- By adding the following bold line to a Button's event handler, a Label control can display output of the application.

```
private void showAnswerButton_Click(object
sender, EventArgs e)
{
   answerLabel.Text = "Theodore Roosevelt";
}
```

- Notice that
  - •the equal sign (=) is known as assignment operator
  - •the item receiving value must be on the left of = operator
  - •the Text property accepts string only
  - •if you need to clear the text of a Label, simply assign an empty string ("") to clear the Text property

2.6 Making Sense of IntelliSense

- IntelliSense provides automatic code completion as you write programming statements
- It provides an array of options that make language references easily accessible
- With it, you can find the information you need, and insert language elements directly



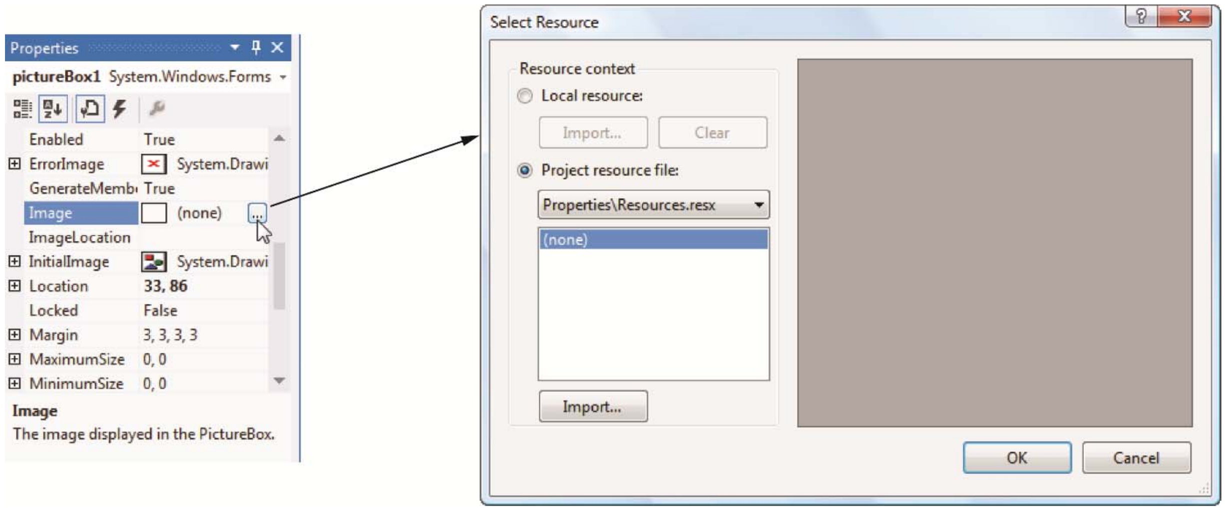
```
private void italianButton_Click(object sender, EventArgs e)
{
    tra
}
```

2.7 PictureBox Controls

- A PictureBox control displays a graphic image on a form
- Commonly used properties are:
  - Image: specifies the image that it will display
  - SizeMode: specifies how the control's image is to be displayed
  - Visible: determines whether the control is visible on the form at run time

2.7 The Image Property's Select Resource Window

- The Image Property has a Select Resource window. To use it:
  - click the ellipses button to open it
  - click the Import button and locate the image file to display



2.7 Creating Clickable Images

AUBURN UNIVERSITY

- You can double click the PictureBox control in the Designer to create a Click event handler and then add your codes to it. For example,

```
private void catPictureBox_Click(object
sender, EventArgs e)
{
   MessageBox.Show("Meow");
}
```

And

```
private void spiderPictureBox_Click(object
sender, EventArgs e)
{
   spiderPictureBox.Visible = false
}
```

2.7 Sequential Execution of Statements

- Programmers need to carefully arrange the sequence of statements in order to generate the correct results
- In the following example, the statements in the method execute in the order that they appear:

```
Private void showBackButton_Click(object
sender, EventArgs e)
{
   cardBackPictureBox.visible = true;
   cardFacePictureBox.visible = true;
}
```

- Incorrect arrangement of sequence can cause logic errors

2.8 Comments, Blank Links, and Indentation

AUBURN UNIVERSITY

- Comments are brief notes that are placed in a program's source code to explain how parts of the program work
- A line comment appears on one line in a program

```
// Make the image of the back visible
cardBackPictureBox.Visible = true;
```

- A block comment can occupy multiple consecutive lines in a program

```
/*
Line one
Line two
*/
```

2.8 Using Blank Lines and Indentation

- Programmers frequently use blank lines and indentation in their codes to make the code more human-readable
- Compare the following two identical codes:

```
namespace Wage_Calculator
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void exitButton_Click(object sender, EventArgs e)
      {
         // Close the form.
         this.Close();
      }
   }
}
```

```
namespace Wage_Calculator
{
         public partial class Form1 : Form
         {
                  public Form1()
                  {
                  InitializeComponent();
                  }
                  private void exitButton_Click(object sender, EventArgs e)
                  {
                  // Close the form.
                  this.Close();
                  }
         }
}
```

AUBURN UNIVERSITY

2.9 Writing the Code to Close an Application's Form

AUBURN UNIVERSITY

- To close an application's form in code, use the following statement:

```
this.Close();
```

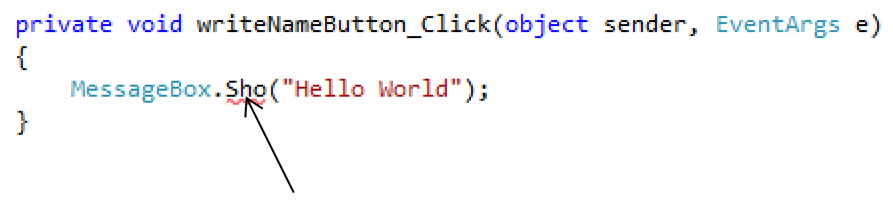
- A commonly used practice is to create an Exit button and manually add the code to it:

```
private void exitButton_Click(object sender,
EventArgs e)
{
  // Close the form.
  this.Close();
}
```

2.10 Dealing with Syntax Errors

AUBURN UNIVERSITY

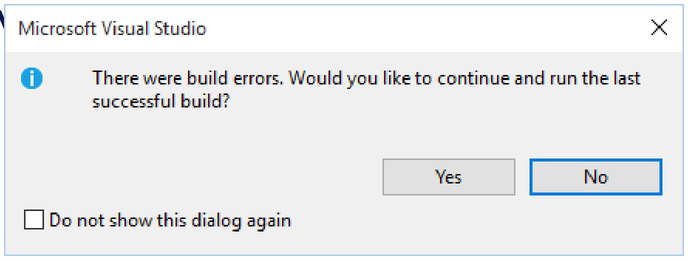
- Visual Studio code editor examines each statement as you type it and reports any syntax errors that are found
- If a syntax error is found, it is underlined with a jagged line



```
private void writeNameButton_Click(object sender, EventArgs e)
{
    MessageBox.Sho("Hello World");
}
```

This jagged line indicates an error

- If a syntax error exists and you attempt to compile and execute, you will see the following window



Microsoft Visual Studio

There were build errors. Would you like to continue and run the last successful build?

3.12 Using the Debugger to Locate Logic Errors

- A **logic error** is a mistake that does not prevent an application from running, but causes the application to produce incorrect results.
  - Mathematical errors
  - Assigning a value to the wrong variable
  - Assigning the wrong value to a variable
  - etc.
- Finding and fixing a logic error usually requires a bit of detective work.
- Visual Studio provides debugging tools that make locating logic errors easier.

AUBURN UNIVERSITY

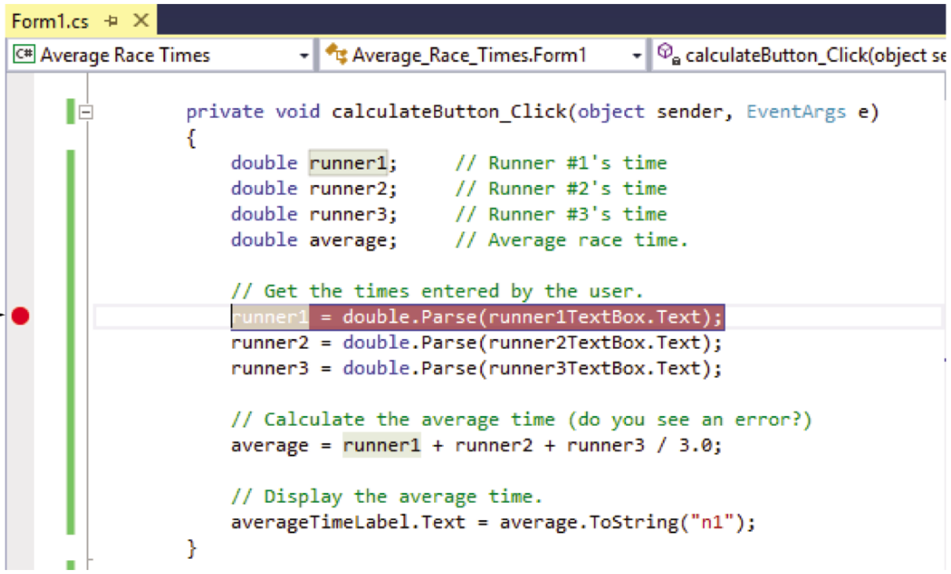
3.12 Breakpoints

- A **breakpoint** is a line you select in your source code.
- When the application is running and it reaches a breakpoint, the application pauses and enters *break mode*.
- While the application is paused, you may examine variable contents and the values stored in certain control properties.

AUBURN UNIVERSITY

3.12 Breakpoints

Click the mouse pointer here.

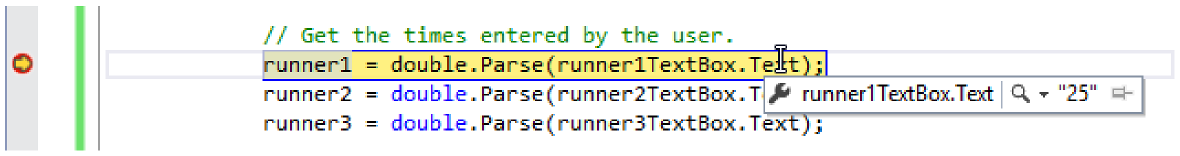


```
private void calculateButton_Click(object sender, EventArgs e)
{
    double runner1;     // Runner #1's time
    double runner2;     // Runner #2's time
    double runner3;     // Runner #3's time
    double average;     // Average race time.

    // Get the times entered by the user.
    runner1 = double.Parse(runner1TextBox.Text);
    runner2 = double.Parse(runner2TextBox.Text);
    runner3 = double.Parse(runner3TextBox.Text);

    // Calculate the average time (do you see an error?)
    average = runner1 + runner2 + runner3 / 3.0;

    // Display the average time.
    averageTimeLabel.Text = average.ToString("n1");
}
```

AUBURN UNIVERSITY

3.12 Break Mode

- In Break mode, to examine the contents of a variable or control property, hover the cursor over the variable or the property's name in the Code editor.



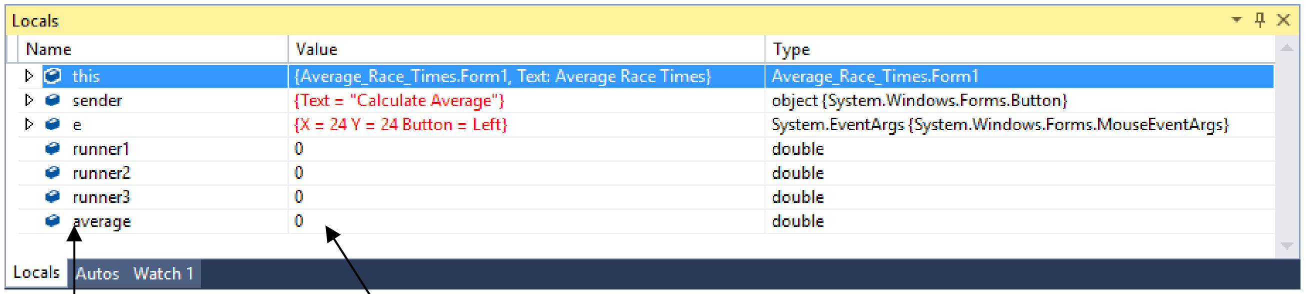
```
// Get the times entered by the user.
runner1 = double.Parse(runner1TextBox.Text);
runner2 = double.Parse(runner2TextBox.T
runner3 = double.Parse(runner3TextBox.Text);
```

3.12 Autos, Locals, and Watch Windows

AUBURN UNIVERSITY

- The ***Autos* window** displays a list of the variables appearing in the current statement, the three statements before, and the three statements after the current statement. The current value and the data type of each variable are also displayed.
- The ***Locals* window** displays a list of all the variables in the current procedure. The current value and the data type of each variable are also displayed.
- The ***Watch* window** allows you to add the names of variables you want to watch. This window displays only the variables you have added. Visual Studio lets you open multiple *Watch* windows.
- You can open any of these windows by clicking *Debug* on the menu bar, then selecting *Windows*, and then selecting the window that you want to

3.12 The Locals Window



Locals

Name	Value	Type
this	{Average_Race_Times.Form1, Text: Average Race Times}	Average_Race_Times.Form1
sender	{Text = "Calculate Average"}	object {System.Windows.Forms.Button}
e	{X = 24 Y = 24 Button = Left}	System.EventArgs {System.Windows.Forms.MouseEventArgs}
runner1	0	double
runner2	0	double
runner3	0	double
average	0	double

Current values

Local variables

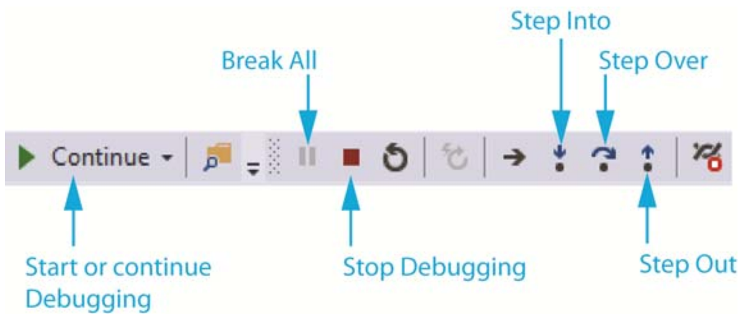
3.12 Single-Stepping

- Visual Studio allows you to **single-step** through an application's code once its execution has been paused by a breakpoint.
- This means that the application's statements execute one at a time, under your control.
- After each statement executes, you can examine variable and property values.
- This process allows you to identify the line or lines of code causing the error.

AUBURN UNIVERSITY

3.12 Single-Stepping

- To single-step, do any of the following:
  - Press F11 on the keyboard, or
  - Click the Step Into command on the toolbar, or
  - Click *Debug* on the menu bar, and then select *Step Into* from the *Debug* menu



What is Version Control?

AUBURN UNIVERSITY

- **Version control** systems are software that help you track changes you make in your code over time.
- As you edit to your code, you tell the version control system to take a snapshot of your files. The version control system saves that snapshot permanently so you can recall it later if you need it.

- Without version control, you're tempted to keep multiple copies of code on your computer. This is dangerous-it's easy to change or delete a file in the wrong copy of code, potentially losing work. Version control systems solve this problem by managing all versions of your code but presenting you with a single version at a time.

https://docs.microsoft.com/en-us/azure/devops/git/what-is-version-control

What is Git?

- **Git** is the most commonly used version control system today and is quickly becoming the standard for version control.
- Git is a distributed version control system, meaning your local copy of code is a complete version control repository. These fully-functional local repositories make it is easy to work offline or remotely. You commit your work locally, and then sync your copy of the repository with the copy on the server. This paradigm differs from centralized version control where clients must synchronize code with a server before creating new versions of code.
- Git's flexibility and popularity make it a great choice for any team. Many developers and college graduates already know how to use Git. Git's user community has created many resources to train developers and Git's popularity make it easy to get help when you need it. Nearly every development environment has Git support and Git command line tools run on every major operating system.

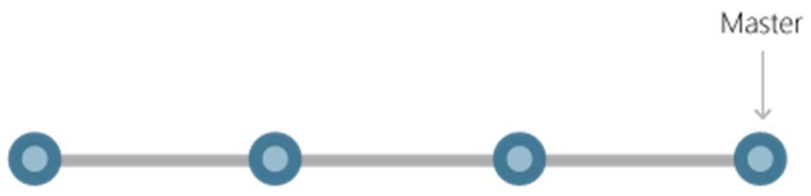
https://docs.microsoft.com/en-us/azure/devops/git/what-is-git

Git Basics

- **Commits:**

Every time you save your work, Git creates a commit.

A commit is a snapshot of all your files at a point in time. If a file has not changed from one commit to the next, Git uses the previously stored file. This design differs from other systems which store an initial version of a file and keep a record of deltas over time.



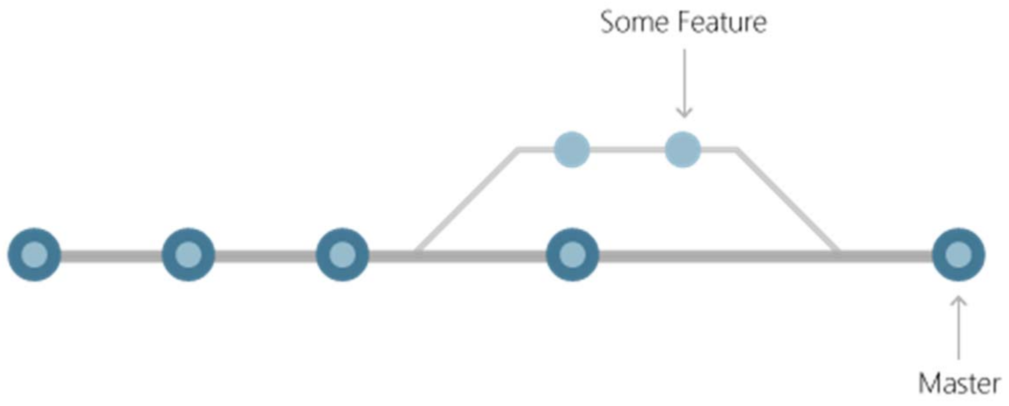
- Commits create links to other commits, forming a graph of your development history . You can revert your code to a previous commit, inspect how files changed from one commit to the next, and review information such as where and when changes were made. Commits are identified in Git by a unique cryptographic hash of the contents of the commit. Because everything is hashed, it is impossible to make changes, lose information, or corrupt files without Git detecting it.

https://docs.microsoft.com/en-us/azure/devops/git/what-is-git

Git Basics

- **Branches:**

Each developer saves changes their own local code repository. As a result, you can have many different changes based off the same commit. Git provides tools for isolating changes and later merging them back together. Branches, which are lightweight pointers to work in progress, manage this separation. Once your work created in a branch is finished, merge it back into your team's main (or master) branch.



https://docs.microsoft.com/en-us/azure/devops/git/what-is-git