Copyright and Usage

This document was created by faculty of the Mobile Application Development Program within the Information and Computing Studies department of the National Technical Institute for the Deaf (a college of the Rochester Institute of Technology) as part of a grant from the National Science Foundation (NSF).

The document is copyrighted. It is licensed for public use with some restrictions under a Creative Commons license.

Restrictions for the use of this document include:

- All references to this work must be fully attributed
- No part of this work can be used commercially
- No derivative variants of this work may be distributed

For more information, please click on the link below.

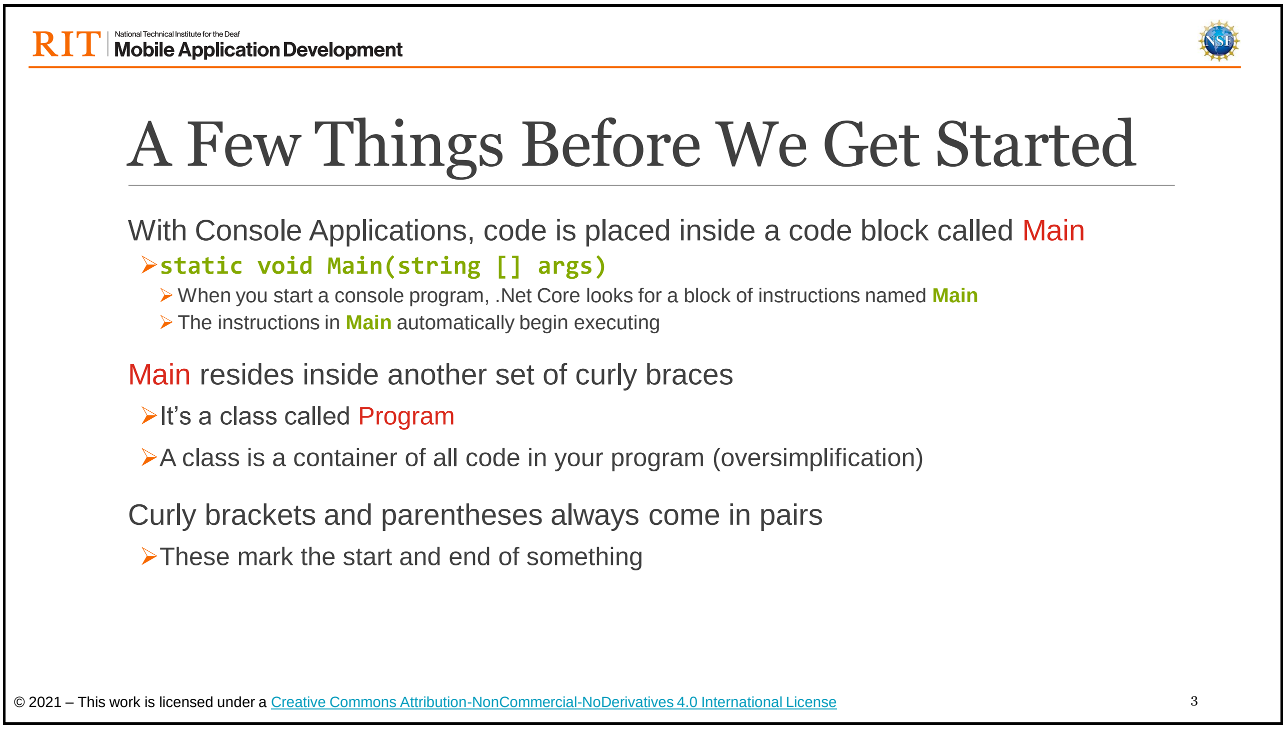
RIT
National Technical Institute for the Deaf
Mobile Application Development

1

Variables & Statements

A BRIEF INTRODUCTION TO VARIABLES, DATA TYPES, AND HOW C# USES THEM

2



RIT | National Technical Institute for the Deaf Mobile Application Development

A Few Things Before We Get Started

With Console Applications, code is placed inside a code block called Main

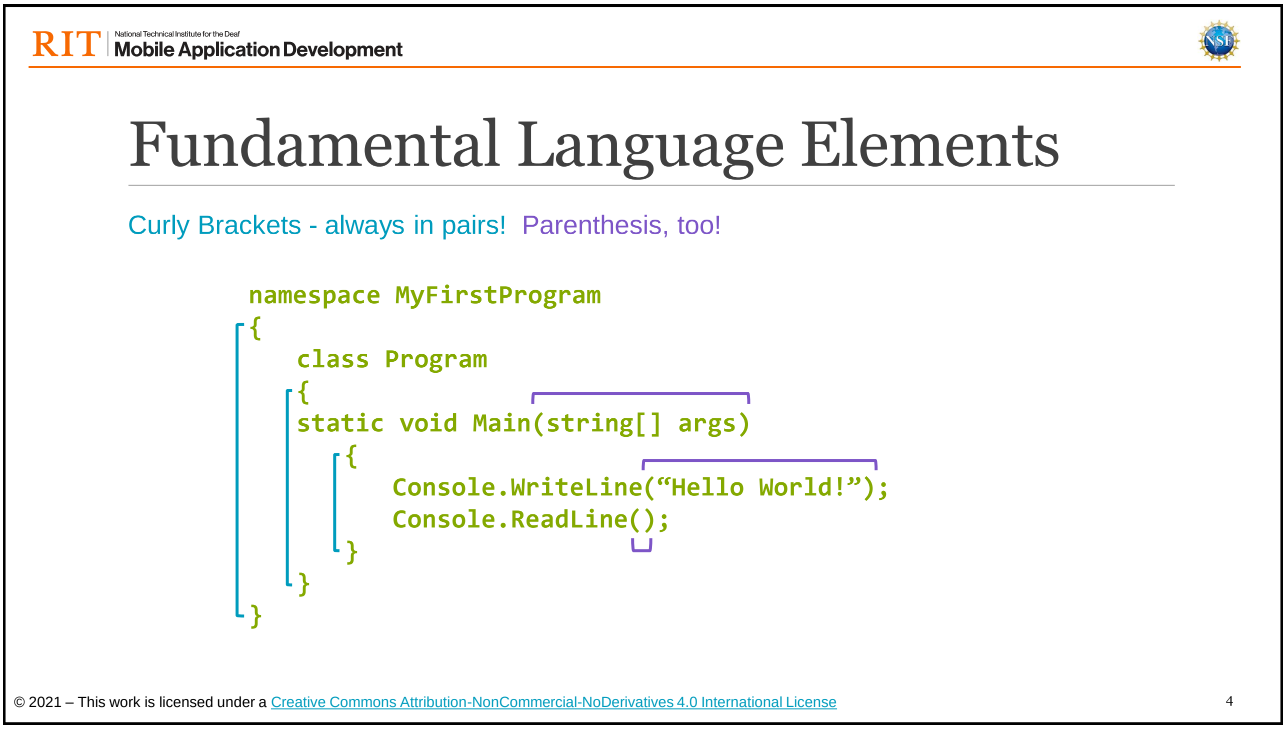
- `static void Main(string [] args)`
  - When you start a console program, .Net Core looks for a block of instructions named **Main**
  - The instructions in **Main** automatically begin executing

Main resides inside another set of curly braces

- It's a class called Program
- A class is a container of all code in your program (oversimplification)

Curly brackets and parentheses always come in pairs

- These mark the start and end of something



RIT | National Technical Institute for the Deaf Mobile Application Development

Fundamental Language Elements

Curly Brackets - always in pairs! Parenthesis, too!

```
namespace MyFirstProgram
{
    class Program
    {
    static void Main(string[] args)
        {
            Console.WriteLine(“Hello World!”);
            Console.ReadLine();
        }
    }
}
```

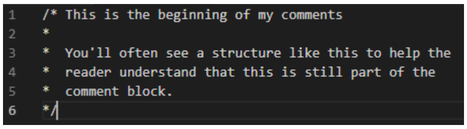
Comments

Entirely for human readers;
ignored by the compiler.

Two ways to comment in C#.

- `// A single line (inline) comment.`
- `/* The text of a multi-line comment may be spread over several lines. */`

Read Chapter 4 in *The C# Player's Guide* for more information about comments



```
/* This is the beginning of my comments
 *
 *  You'll often see a structure like this to help the
 *  reader understand that this is still part of the
 *  comment block.
 */
```

You might see multi-line comments formatted like this

The World Of Coding

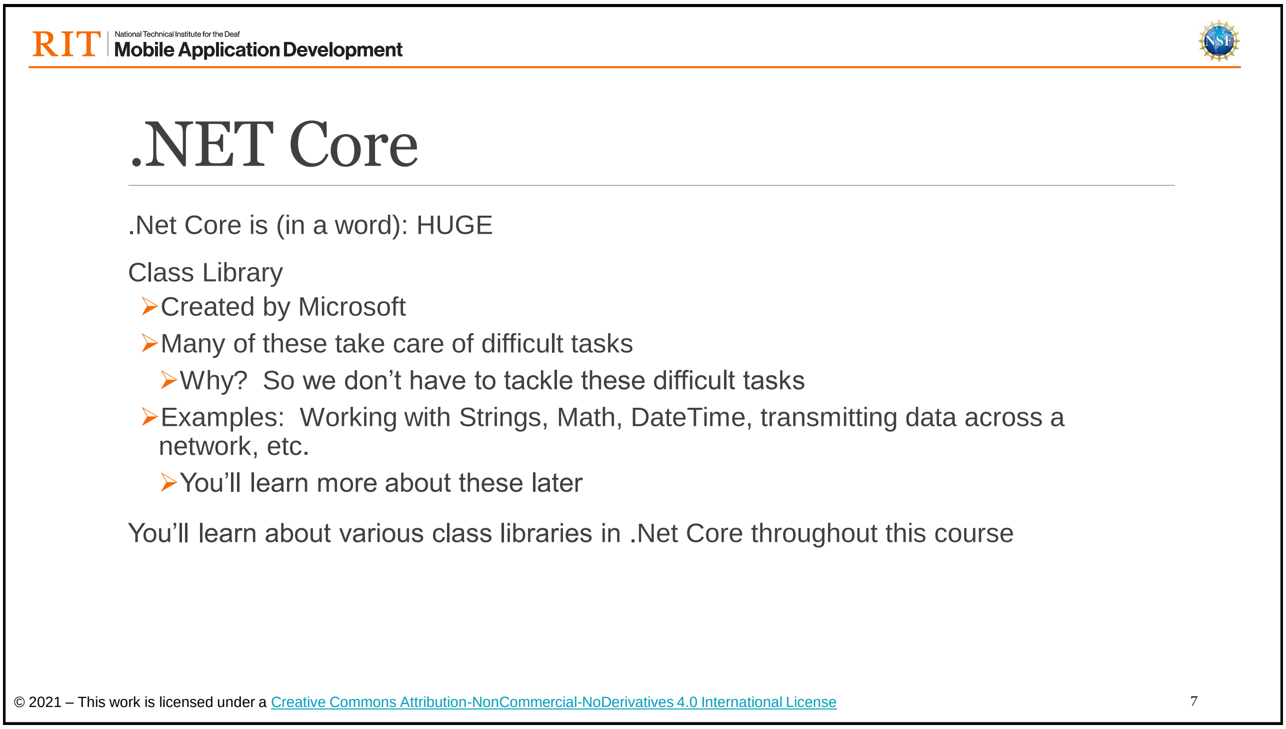
Half of learning code is the syntax of a programming language, in this case, C#

- Syntax involves nouns, verbs, punctuations, etc.

The other half is learning about the pre-built functionality of programming

- Such as Write() and WriteLine()

Pre-built functionality comes from the .NET Core



RIT | National Technical Institute for the Deaf **Mobile Application Development**

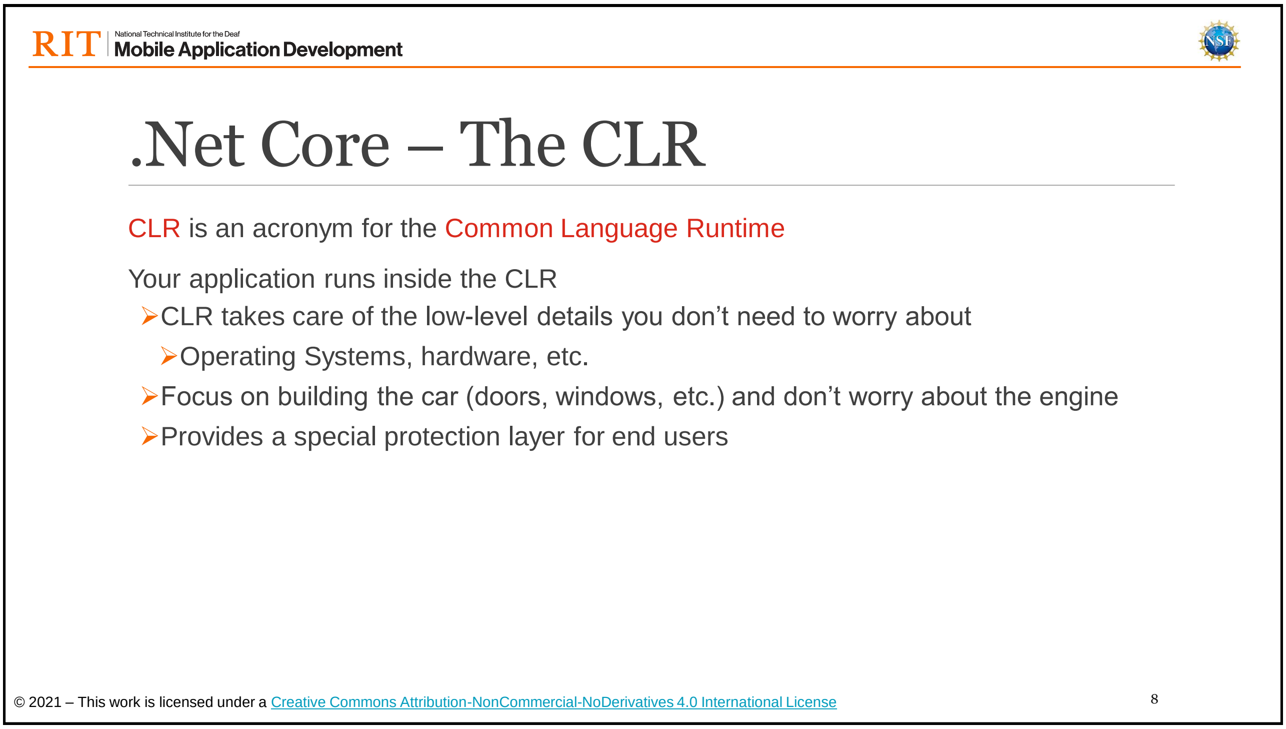
.NET Core

.Net Core is (in a word): HUGE

Class Library

- Created by Microsoft
- Many of these take care of difficult tasks
  - Why? So we don't have to tackle these difficult tasks
- Examples: Working with Strings, Math, DateTime, transmitting data across a network, etc.
  - You'll learn more about these later

You'll learn about various class libraries in .Net Core throughout this course



RIT | National Technical Institute for the Deaf **Mobile Application Development**

.Net Core – The CLR

CLR is an acronym for the Common Language Runtime

Your application runs inside the CLR

- CLR takes care of the low-level details you don't need to worry about
  - Operating Systems, hardware, etc.
- Focus on building the car (doors, windows, etc.) and don't worry about the engine
- Provides a special protection layer for end users

Definition Of “Variable”

Definition from Oxford Dictionaries:

Adjective

- ➢not consistent or having a fixed pattern; liable to change
- ➢able to be changed or adapted

Noun

- ➢an element, feature, or factor that is liable to vary or change

Note the word that appears in every definition: **change**

In programming, a variable is a place in memory to store information

- ➢Think of it as a bucket or box to put something in
- ➢When the program runs, what’s “in the bucket” may change or stay the same

This is really important!

On the previous slide, we said:

- ➢“In programming, a variable is a place in memory to store information” and to think of it like a bucket to put stuff in

Here’s the important part: ***The variable (bucket) can hold one (and only one) value!!!!***

- ➢For example, your age, your cell phone number, etc.
- ➢It cannot hold both your first name and age

RIT | National Technical Institute for the Deaf
Mobile Application Development

Variables: Names

We need to create a name for each piece of information (data) we want to work with

Each variable in C# has a unique name

The name of the variable allows you to access the specific data associated with that variable

- ➢You can have 1,000 variables and each would have its own unique name
- ➢Names should be easily identifiable; not arbitrary
- ➢Names should clearly indicate what data is being stored there
- ➢For example, studentTuitionBalance is more clear than owesTheCollege

Which bucket did I store the first name in?

RIT | National Technical Institute for the Deaf
Mobile Application Development

Variables: Names – II

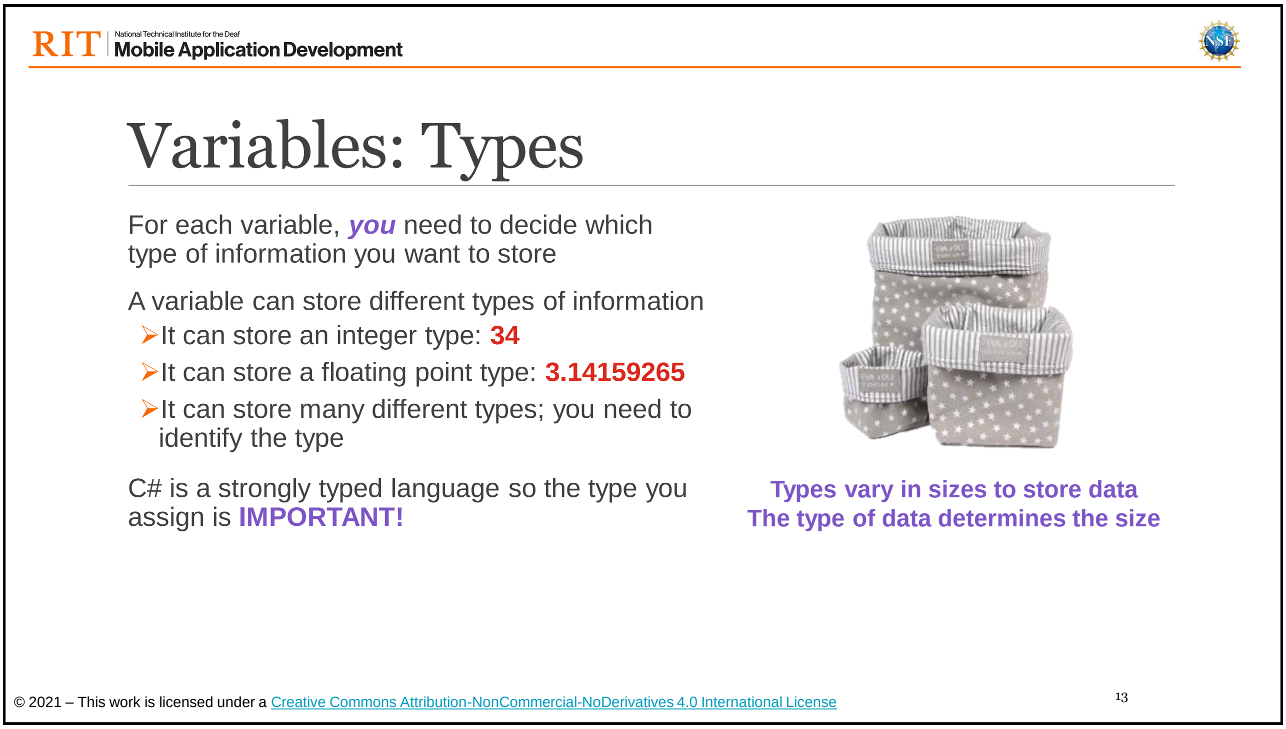
Use (lower) camel casing for names

- ➢One or more words with no space
- ➢First word is in lowercase format
- ➢Remaining word element begins with uppercase letter

Examples:

- ➢**school**
- ➢**myFirstCar**
- ➢**thisIsAReallyLongVariableName**

Which bucket did I store the first name in?



RIT | National Technical Institute for the Deaf Mobile Application Development

Variables: Types

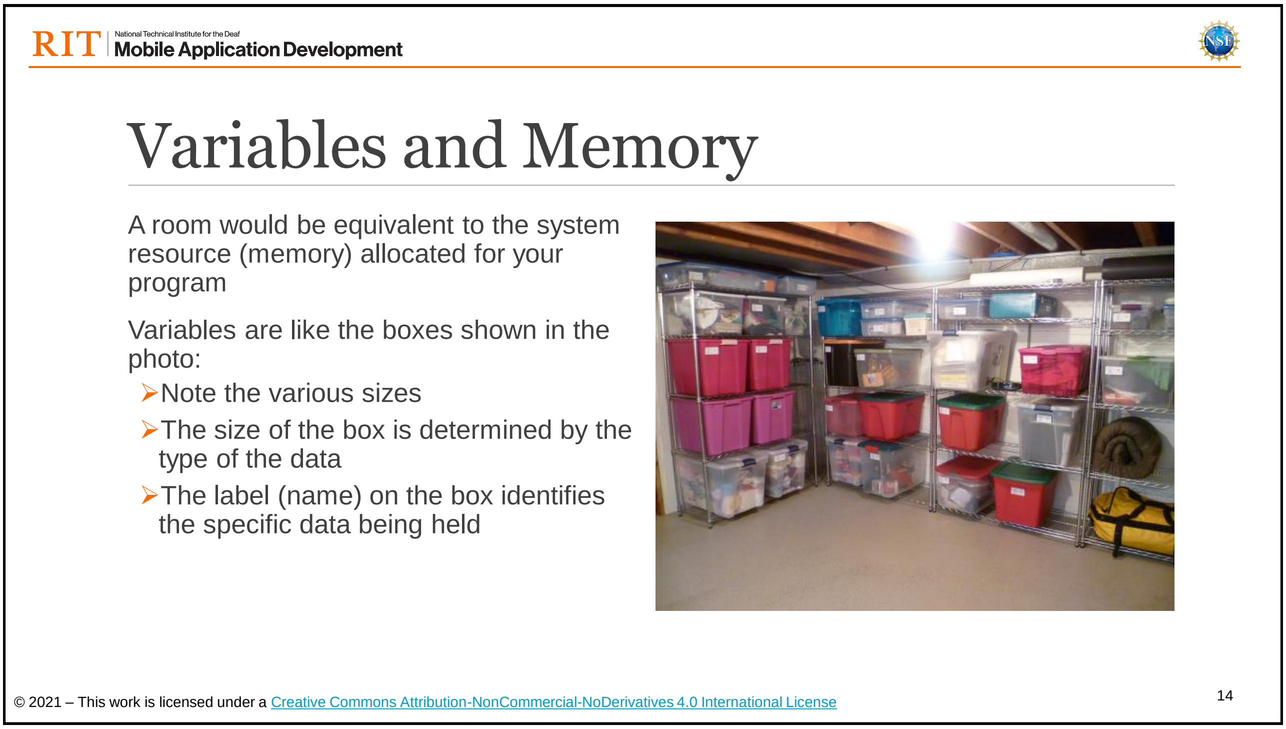
For each variable, ***you*** need to decide which type of information you want to store

A variable can store different types of information

- It can store an integer type: **34**
- It can store a floating point type: **3.14159265**
- It can store many different types; you need to identify the type

C# is a strongly typed language so the type you assign is **IMPORTANT!**

Types vary in sizes to store data
The type of data determines the size



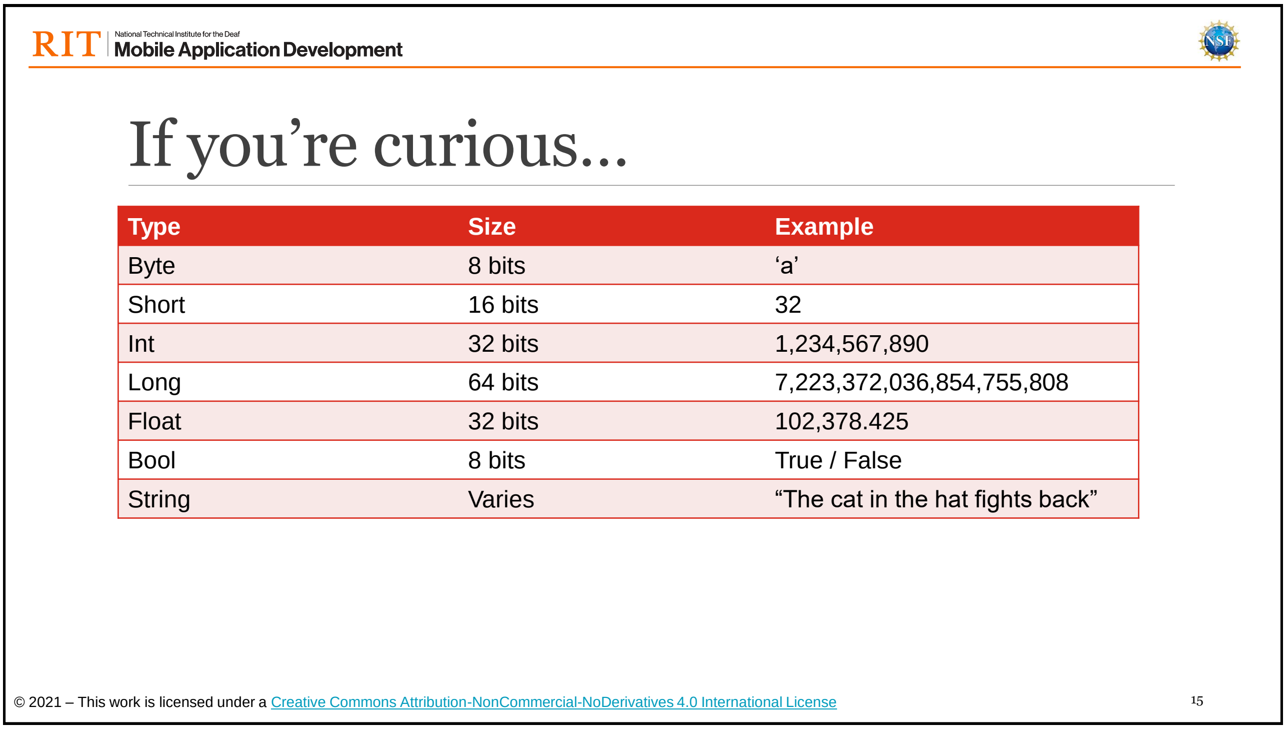
RIT | National Technical Institute for the Deaf Mobile Application Development

Variables and Memory

A room would be equivalent to the system resource (memory) allocated for your program

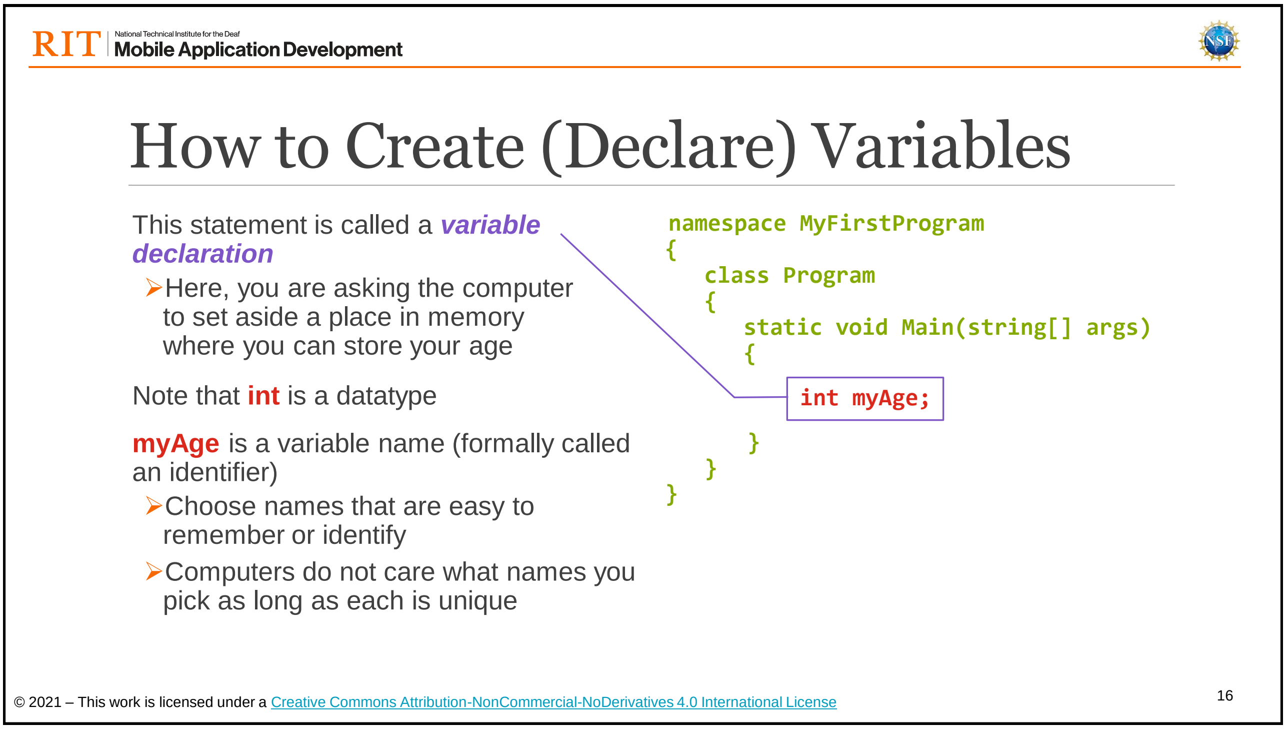
Variables are like the boxes shown in the photo:

- Note the various sizes
- The size of the box is determined by the type of the data
- The label (name) on the box identifies the specific data being held



If you're curious…

Type	Size	Example
Byte	8 bits	'a'
Short	16 bits	32
Int	32 bits	1,234,567,890
Long	64 bits	7,223,372,036,854,755,808
Float	32 bits	102,378.425
Bool	8 bits	True / False
String	Varies	"The cat in the hat fights back"



How to Create (Declare) Variables

This statement is called a ***variable declaration***

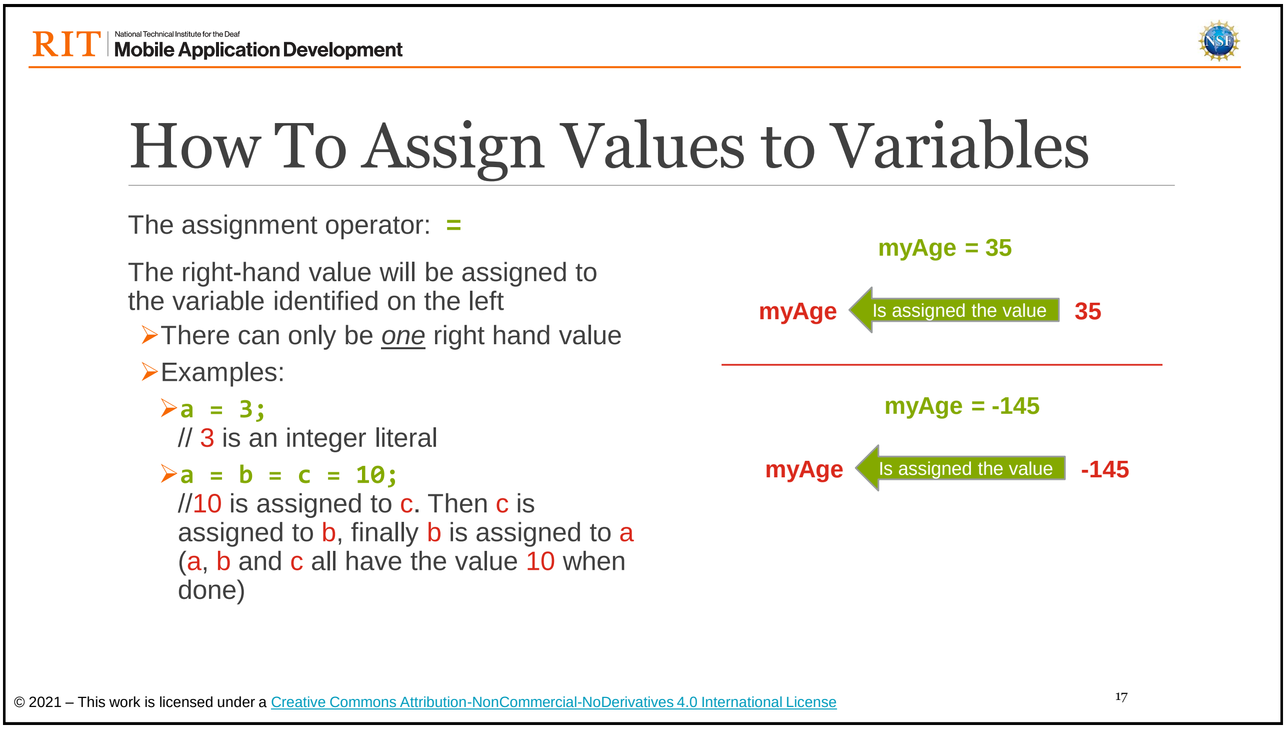
- Here, you are asking the computer to set aside a place in memory where you can store your age

Note that **int** is a datatype

myAge is a variable name (formally called an identifier)

- Choose names that are easy to remember or identify
- Computers do not care what names you pick as long as each is unique

```
namespace MyFirstProgram
{
   class Program
   {
      static void Main(string[] args)
      {
            int myAge;
      }
   }
}
```



RIT | National Technical Institute for the Deaf **Mobile Application Development**

How To Assign Values to Variables

The assignment operator: `=`

The right-hand value will be assigned to the variable identified on the left

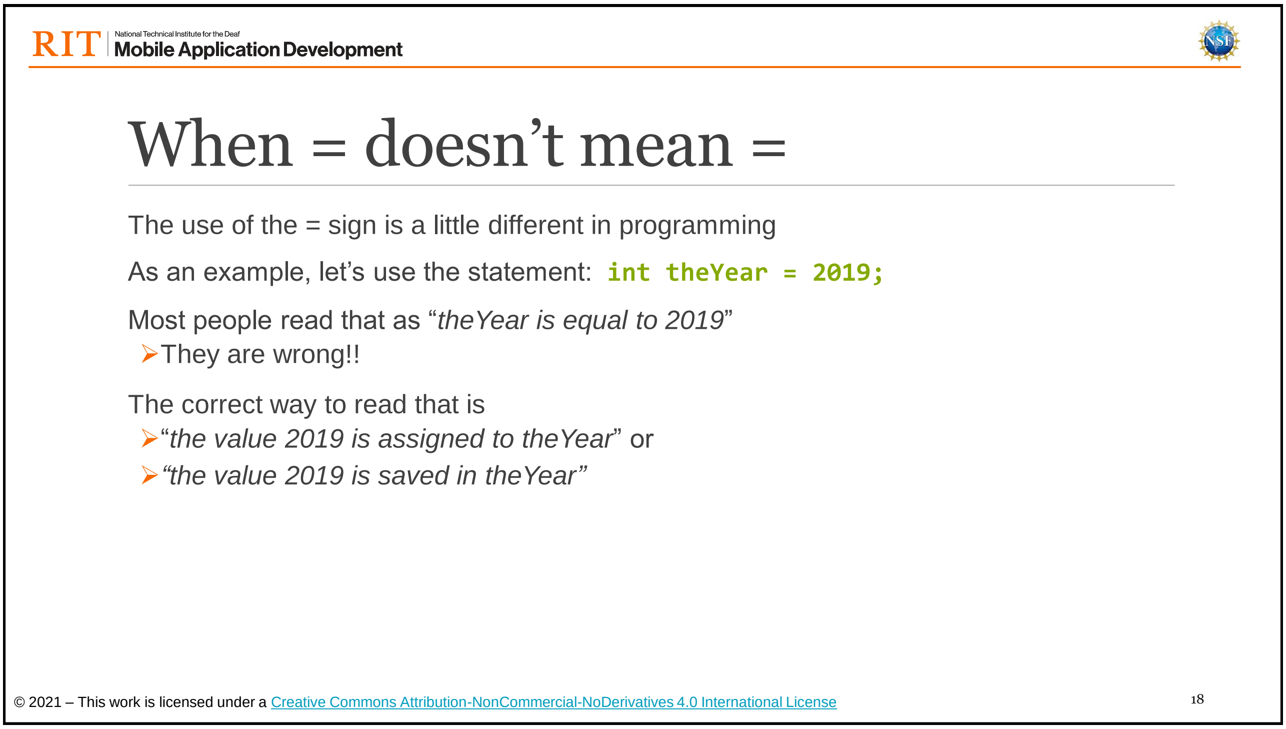
- There can only be *one* right hand value
- Examples:
  - `a = 3;`
    // 3 is an integer literal
  - `a = b = c = 10;`
    //10 is assigned to c. Then c is assigned to b, finally b is assigned to a (a, b and c all have the value 10 when done)

myAge = 35

myAge ← Is assigned the value **35**

myAge = -145

myAge ← Is assigned the value **-145**



RIT | National Technical Institute for the Deaf **Mobile Application Development**

When = doesn’t mean =

The use of the = sign is a little different in programming

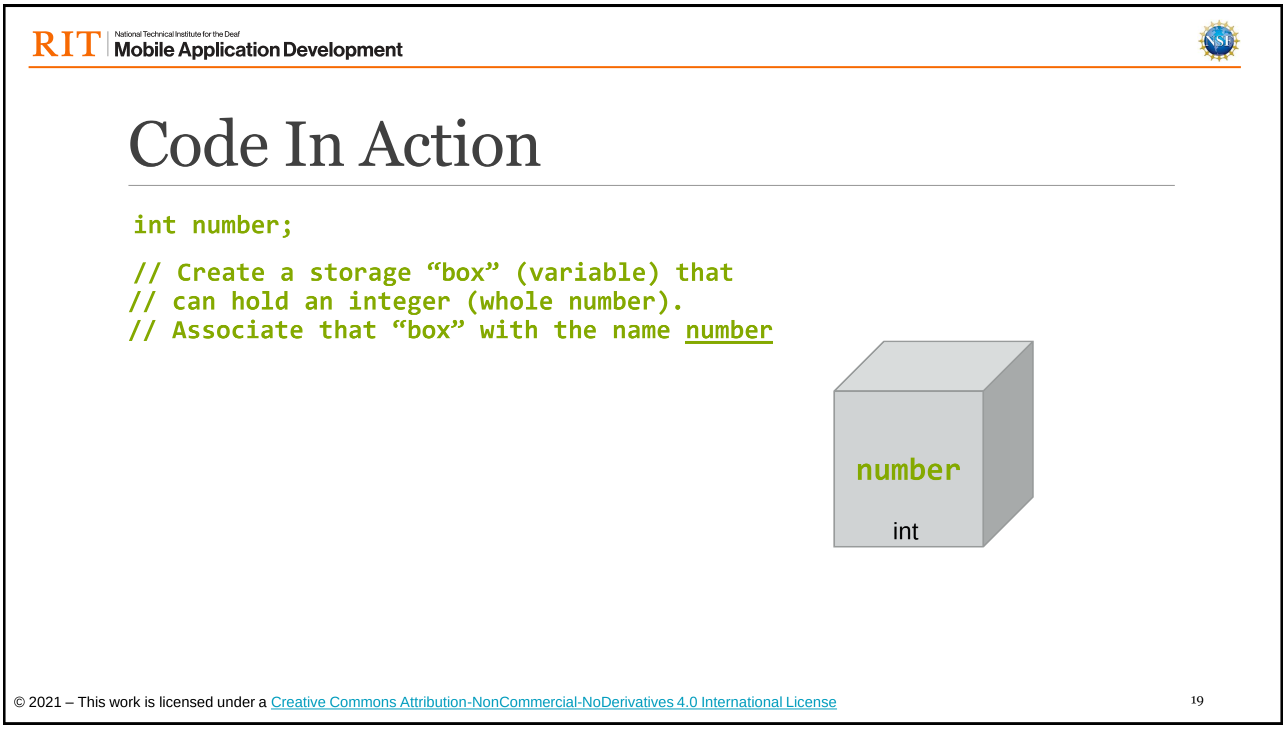
As an example, let’s use the statement: `int theYear = 2019;`

Most people read that as “*theYear is equal to 2019*”

- They are wrong!!

The correct way to read that is

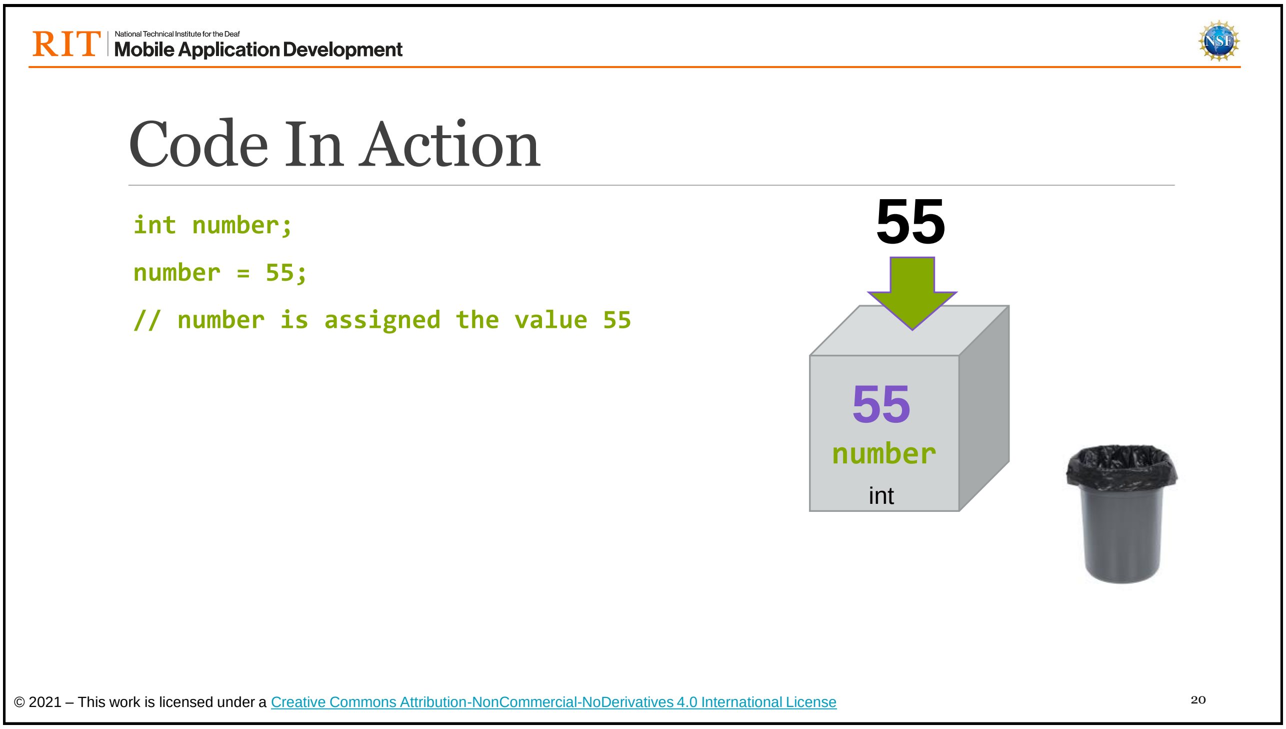
- “*the value 2019 is assigned to theYear*” or
- *“the value 2019 is saved in theYear”*



RIT | National Technical Institute for the Deaf
Mobile Application Development

Code In Action

```
int number;

// Create a storage "box" (variable) that
// can hold an integer (whole number).
// Associate that "box" with the name number
```

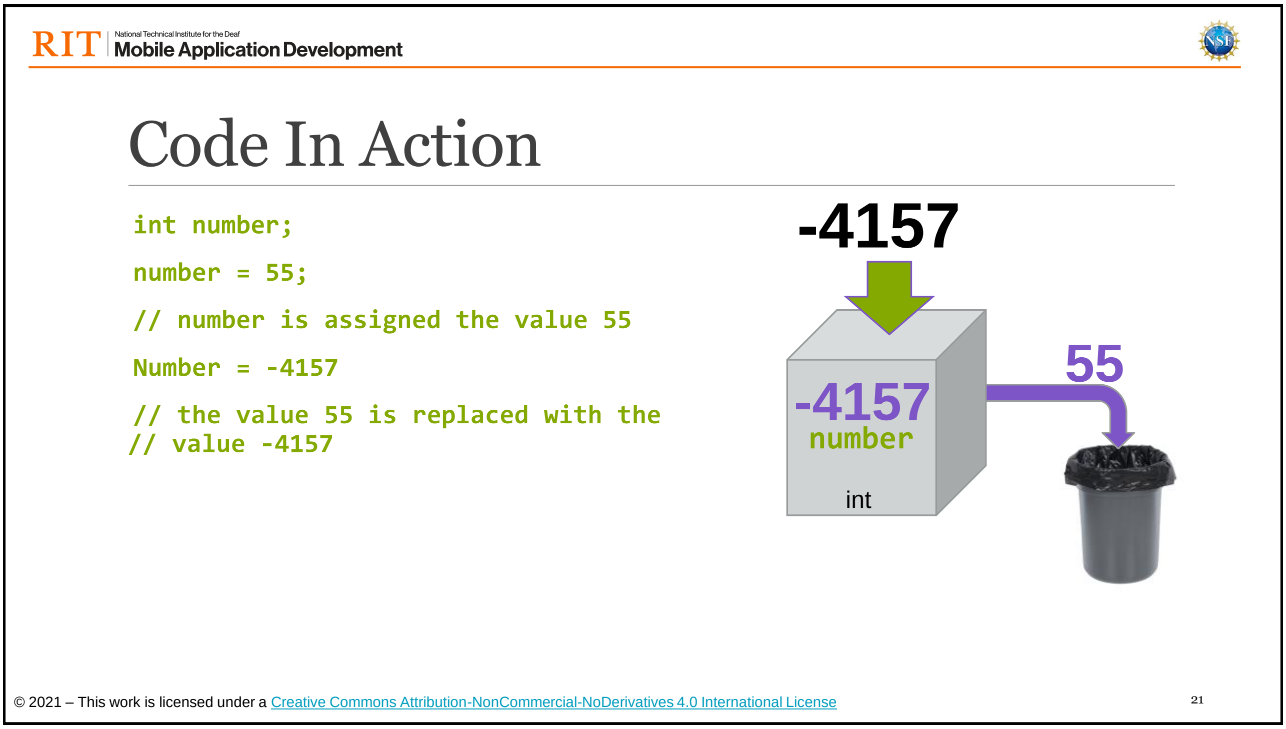


RIT | National Technical Institute for the Deaf
Mobile Application Development

Code In Action

```
int number;

number = 55;

// number is assigned the value 55
```

55
55
number
int

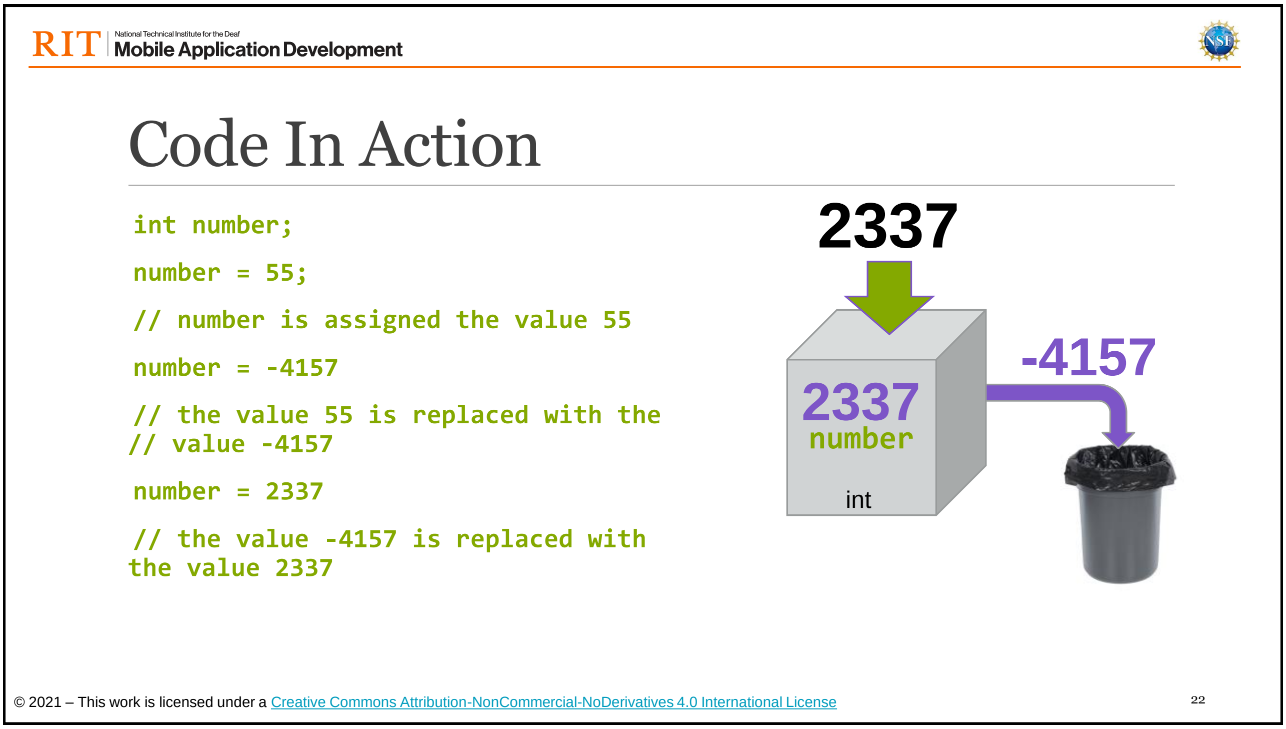


RIT | National Technical Institute for the Deaf
Mobile Application Development

Code In Action

```
int number;

number = 55;

// number is assigned the value 55

Number = -4157

// the value 55 is replaced with the
// value -4157
```

-4157
-4157
number
int
55

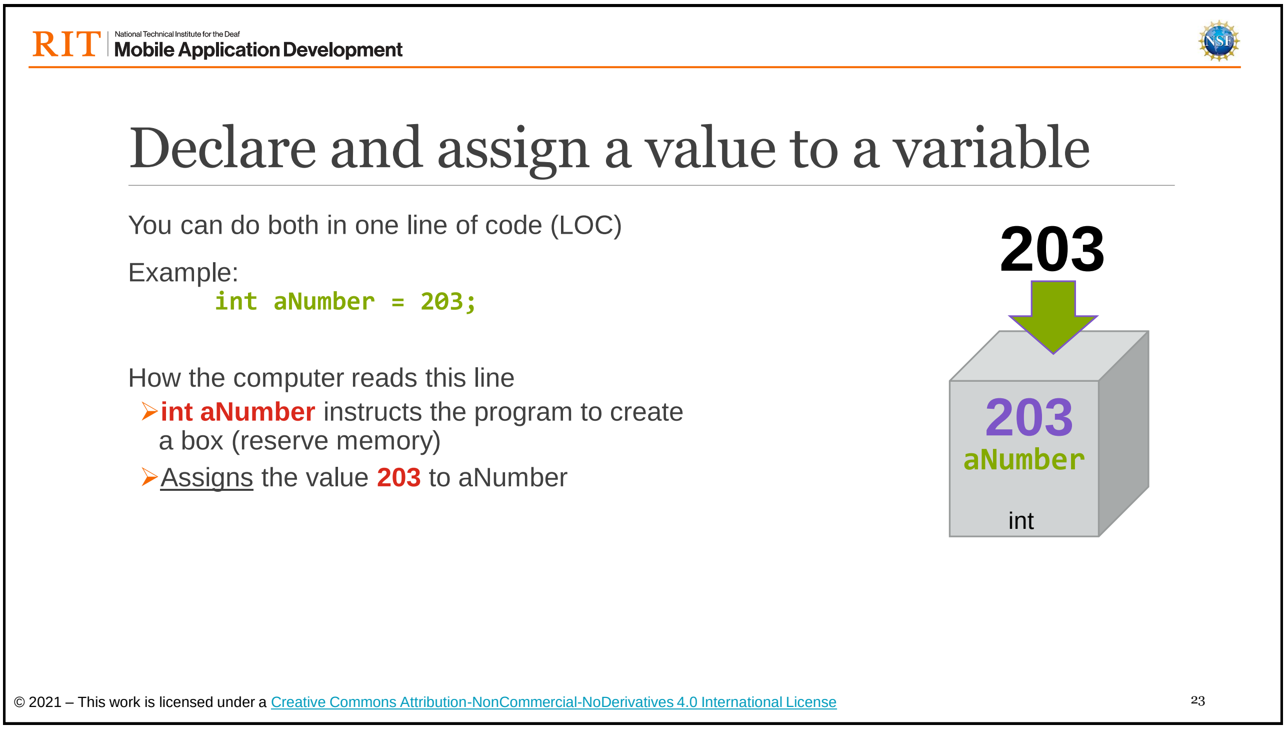


RIT | National Technical Institute for the Deaf
Mobile Application Development

Code In Action

```
int number;

number = 55;

// number is assigned the value 55

number = -4157

// the value 55 is replaced with the
// value -4157

number = 2337

// the value -4157 is replaced with
the value 2337
```

2337
2337
number
int
-4157



Declare and assign a value to a variable

You can do both in one line of code (LOC)

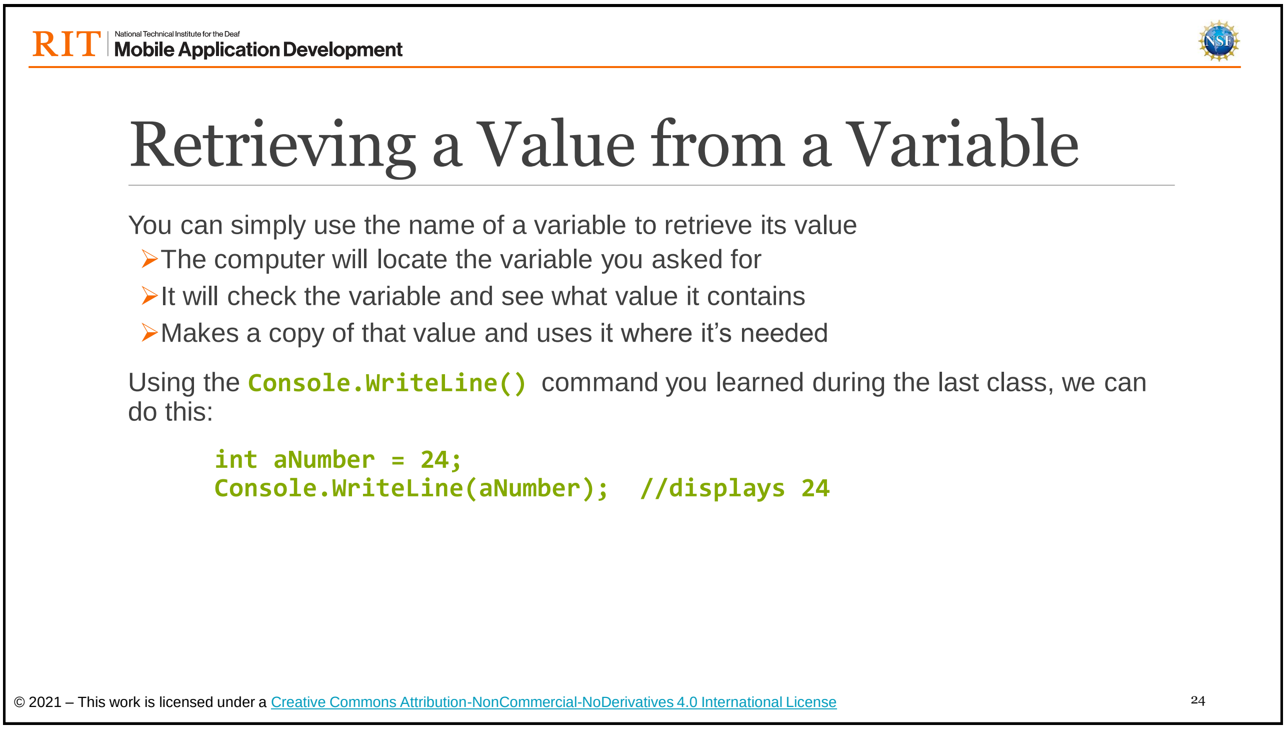
Example:

```
int aNumber = 203;
```

How the computer reads this line

- ➢**int aNumber** instructs the program to create a box (reserve memory)
- ➢Assigns the value **203** to aNumber

203
203
aNumber
int



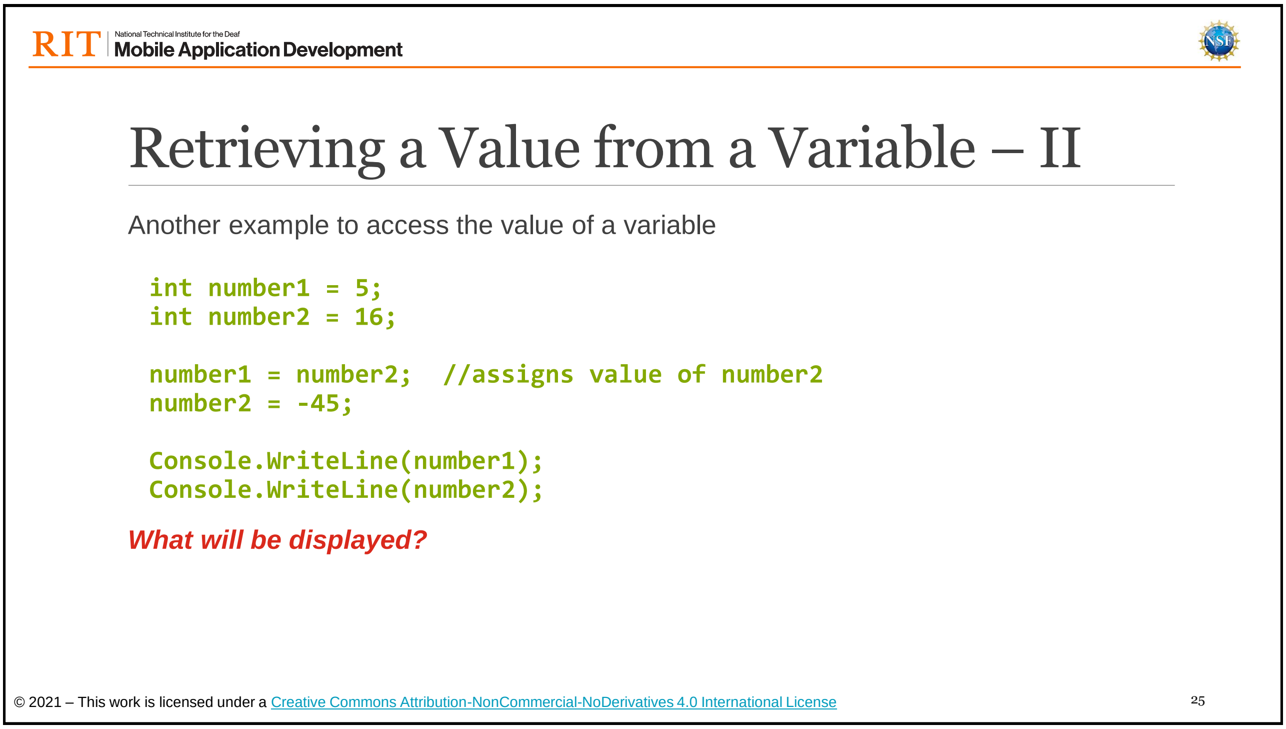
Retrieving a Value from a Variable

You can simply use the name of a variable to retrieve its value

- ➢The computer will locate the variable you asked for
- ➢It will check the variable and see what value it contains
- ➢Makes a copy of that value and uses it where it’s needed

Using the `Console.WriteLine()` command you learned during the last class, we can do this:

```
int aNumber = 24;
Console.WriteLine(aNumber);  //displays 24
```



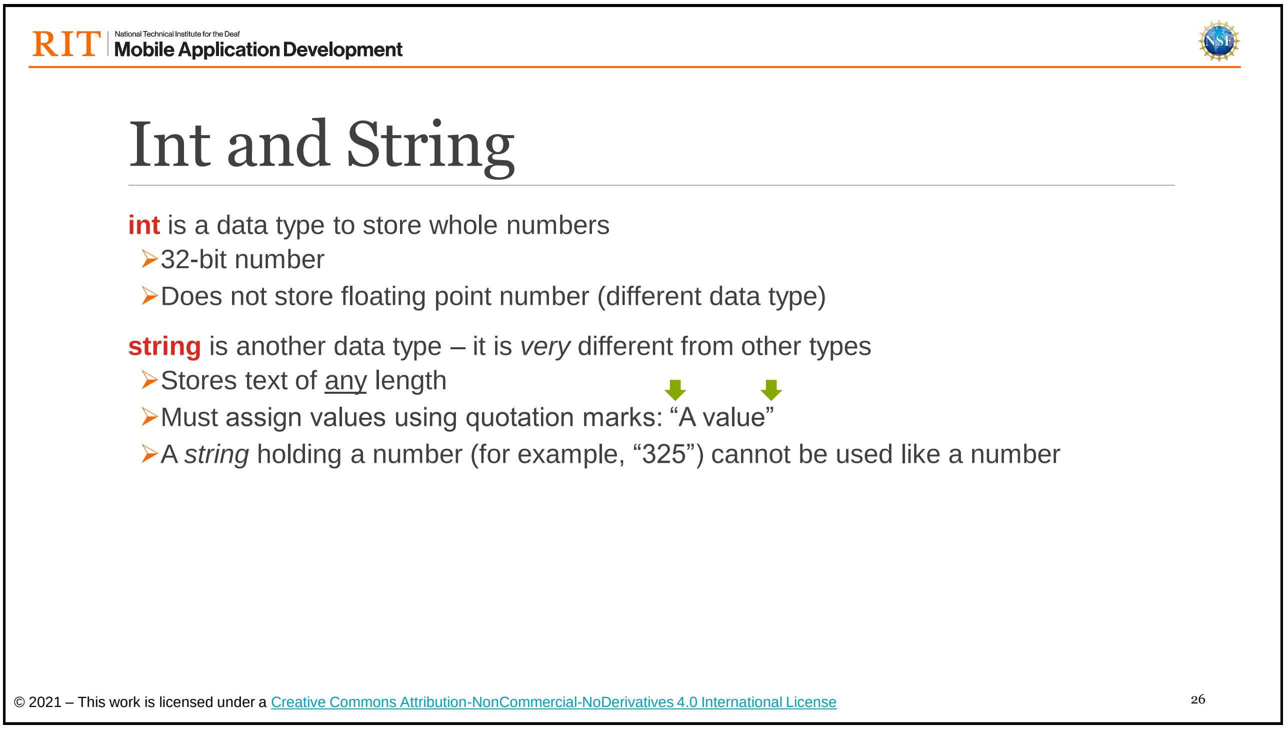
Retrieving a Value from a Variable – II

Another example to access the value of a variable

```
int number1 = 5;
int number2 = 16;

number1 = number2;  //assigns value of number2
number2 = -45;

Console.WriteLine(number1);
Console.WriteLine(number2);
```

What will be displayed?

25



Int and String

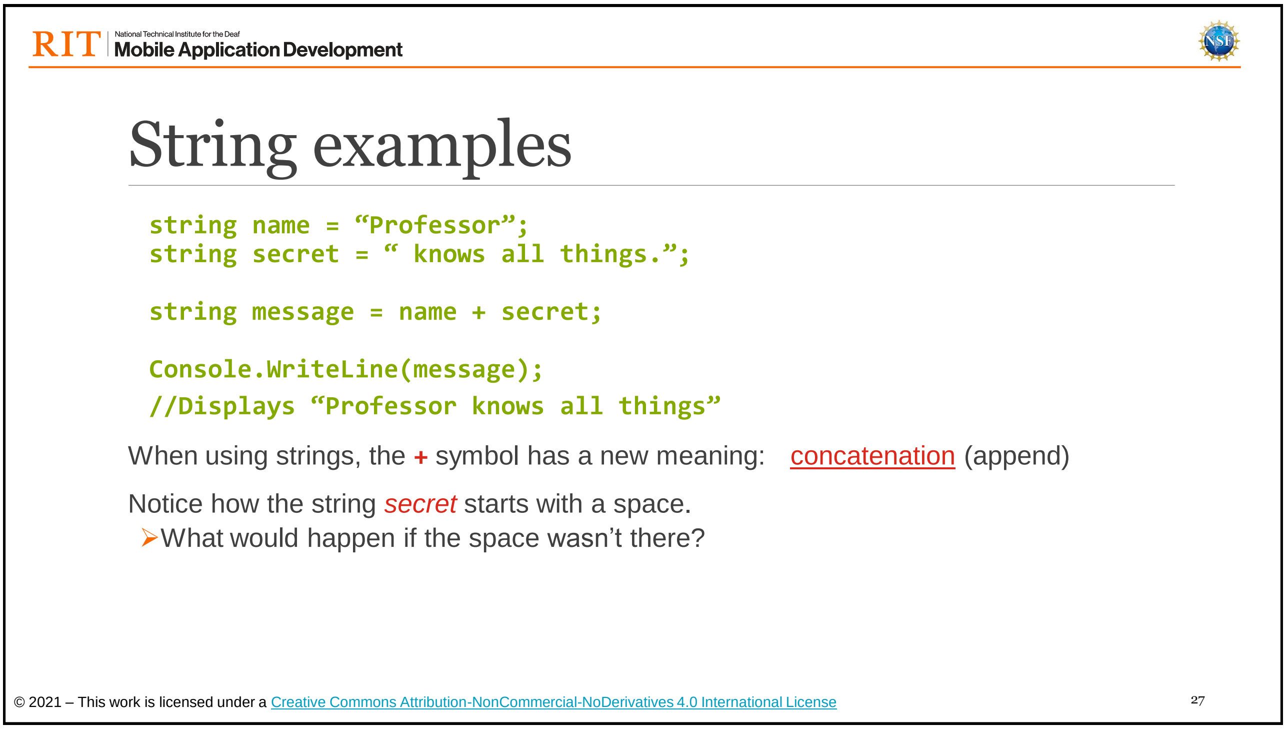
int is a data type to store whole numbers

- ➢32-bit number
- ➢Does not store floating point number (different data type)

string is another data type – it is *very* different from other types

- ➢Stores text of any length
- ➢Must assign values using quotation marks: “A value”
- ➢A *string* holding a number (for example, “325”) cannot be used like a number

26



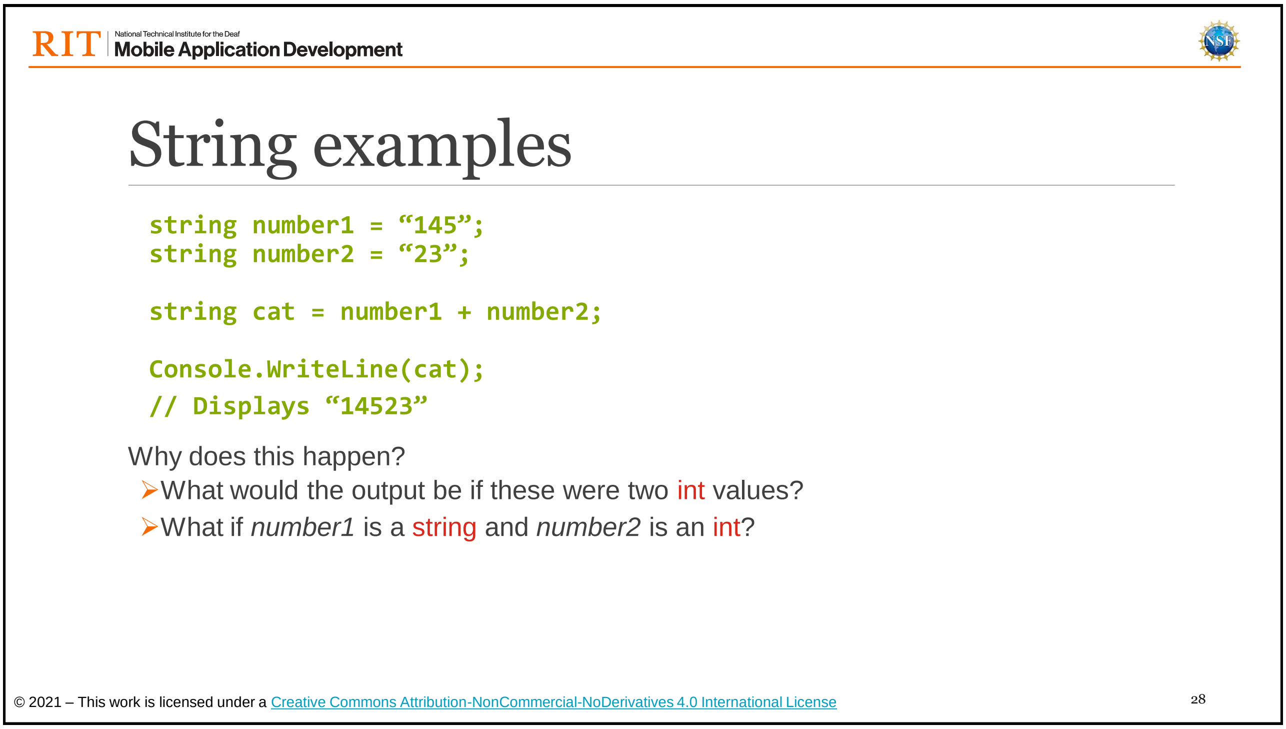
String examples

```
string name = “Professor”;
string secret = “ knows all things.”;

string message = name + secret;

Console.WriteLine(message);
//Displays “Professor knows all things”
```

When using strings, the **+** symbol has a new meaning: concatenation (append)

Notice how the string *secret* starts with a space.

- What would happen if the space wasn’t there?

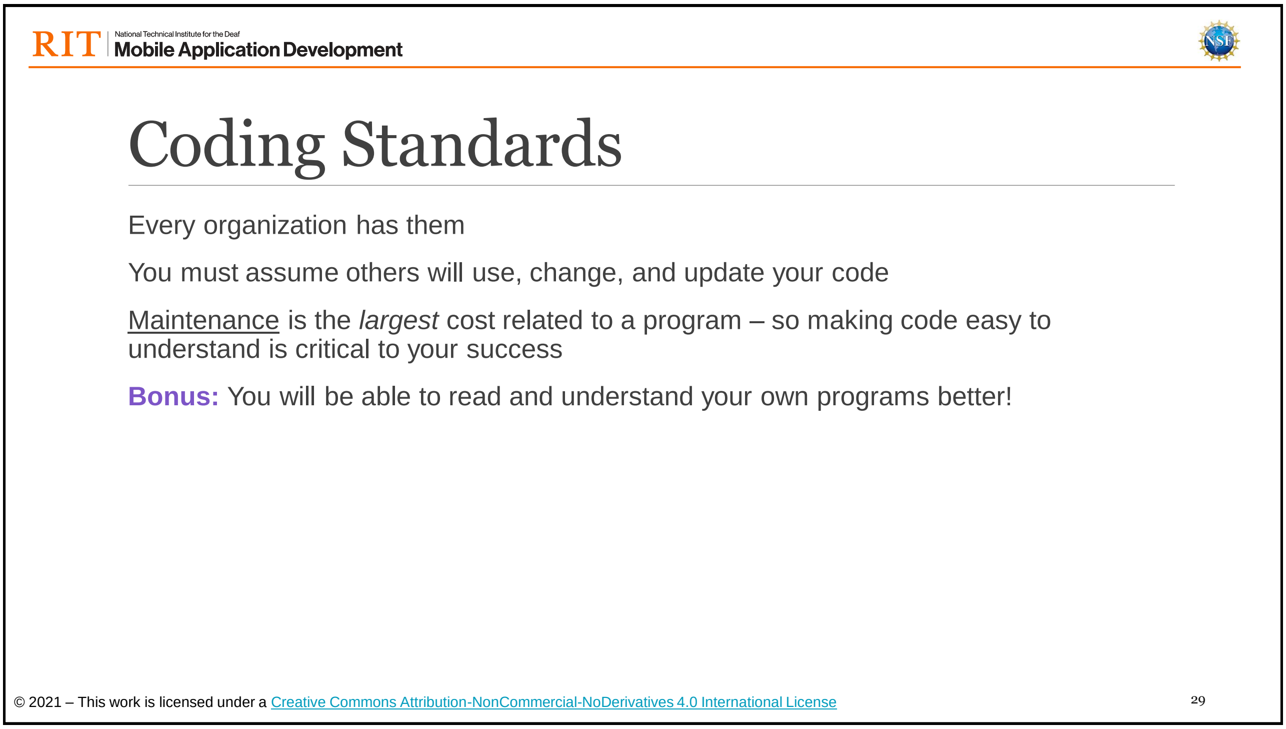


String examples

```
string number1 = “145”;
string number2 = “23”;

string cat = number1 + number2;

Console.WriteLine(cat);
// Displays “14523”
```

Why does this happen?

- What would the output be if these were two int values?
- What if *number1* is a string and *number2* is an int?



Coding Standards

Every organization has them

You must assume others will use, change, and update your code

Maintenance is the *largest* cost related to a program – so making code easy to understand is critical to your success

Bonus: You will be able to read and understand your own programs better!