**C# Syntax**

**Data Types**

1. **String- example: ( “Jeremiah”)**
2. **Char- example: ( ‘J’)** character or letter just one letter
3. **Int- example: ( 2023)** any number
4. **Decimal- example: ( “+ 1.85m)** the m is for the representation of the decimal A floating point number, is a positive or negative whole number with a decimal point
5. **Bool- example: ( + true** for a true/false value
6. **Date time example: DateTime.UtcNow** represents date and time needs to be initialized way I did it here was the UtcNow method

**Variables: Pieces of data that may change during their life time**

Stores values in a temporarily memory address that you can use or change to use the value use the variable name

Variable names should use **camel case**, which is a style of writing that uses a lower-case letter at the beginning of the first word and an upper-case letter at the beginning of each subsequent word

To create a new variable, you must first declare the data type of the variable, then give it a name.

**String jeremiahMoore**

In this case, we're creating a new variable of type string called jeremiahMoore. From now on, this variable can only hold string values.

**String Interpolation provides a more readable, convenient syntax to format strings.** It's easier to read than string composite formatting.

string name = "Jeremiah";

var date = DateTime.Now;

// Composite formatting:

Console.WriteLine("Hello, {0}! Today is {1}, it's {2:HH:mm} now.", name, date.DayOfWeek, date);

// String interpolation:

Console.WriteLine($"Hello, {name}! Today is {date.DayOfWeek}, it's {date:HH:mm} now.");

// Both calls produce the same output that is similar to:

// Hello, Jeremiah! Today is Wednesday, it's 19:40 now.

**Branching**

branch your code's execution path by evaluating Boolean expressions.

Write code that evaluates conditions using if, else, and else if statements

Lets program make decisions

# **Comparison operators**

The < (less than), > (greater than), <= (less than or equal), and >= (greater than or equal) comparison, also known as relational, operators compare their operands. Those operators are supported by all integral and floating-point numeric types.

## Less than operator <

The < operator returns true if its left-hand operand is less than its right-hand operand, false otherwise

Console.WriteLine(7.0 < 5.1); // output: False

Console.WriteLine(5.1 < 5.1); // output: False

**Console.WriteLine(0.0 < 5.1); // output: True**

## Greater than operator >

The > operator returns true if its left-hand operand is greater than its right-hand operand, false otherwise:

**Console.WriteLine(7.0 > 5.1); // output: True**

Console.WriteLine(5.1 > 5.1); // output: False

Console.WriteLine(0.0 > 5.1); // output: False

## Less than or equal operator <=

The <= operator returns true if its left-hand operand is less than or equal to its right-hand operand, false otherwise

Console.WriteLine(7.0 <= 5.1); // output: False

**Console.WriteLine(5.1 <= 5.1); // output: True**

**Console.WriteLine(0.0 <= 5.1); // output: True**

## Greater than or equal operator >=

## The >= operator returns true if its left-hand operand is greater than or equal to its right-hand operand, false otherwise:

**Console.WriteLine(7.0 >= 5.1); // output: True**

**Console.WriteLine(5.1 >= 5.1); // output: True**

Console.WriteLine(0.0 >= 5.1); // output: False

# **Methods**

A method is a code block that contains a series of statements. A program causes the statements to be executed by calling the method and specifying any required method arguments. In C#, every executed instruction is performed in the context of a method. Can apply multiple methods to the same variable

Methods are declared in a [class](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/class), [struct](https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/struct), or [interface](https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/interfaces) by specifying the access level such as public or private, optional modifiers such as abstract or sealed, the return value, the name of the method, and any method parameters. These parts together are the signature of the method.

abstract class Motorcycle

{

// Anyone can call this.

public void StartEngine() {/\* Method statements here \*/ }

// Only derived classes can call this.

protected void AddGas(int gallons) { /\* Method statements here \*/ }

// Derived classes can override the base class implementation.

public virtual int Drive(int miles, int speed) { /\* Method statements here \*/ return 1; }

// Derived classes must implement this.

public abstract double GetTopSpeed();

}

Calling a method on an object is like accessing a field. After the object name, add a period, the name of the method, and parentheses. Arguments are listed within the parentheses, and are separated by commas. The methods of the Motorcycle class can therefore be called as in the following example:

class TestMotorcycle : Motorcycle

{

public override double GetTopSpeed()

{

return 108.4;

}

static void Main()

{

TestMotorcycle moto = new TestMotorcycle();

moto.StartEngine();

moto.AddGas(15);

moto.Drive(5, 20);

double speed = moto.GetTopSpeed();

Console.WriteLine("My top speed is {0}", speed);

}

}

**Switch statements**

Learn how to add branching logic that matches one variable or expression against many possible values.

* Use the switch-case construct to match a variable or expression against several possible outcomes.
* Convert code that uses an if-elseif-else construct into a switch-case construct.

**For Loop**

Use the for iteration statement to loop a pre-set number of times and control the iteration process.

In this module, you will:

* Use the for statement to loop through a block of code.
* Modify how the .NET Runtime executes the looping logic, changing the value of the iterator, the condition and the pattern.

**Arrays**

Collections of data a sequence of individual data elements that can be accessed through a single variable name: elements of data need to be the same type

Data type followed by[] means it an array