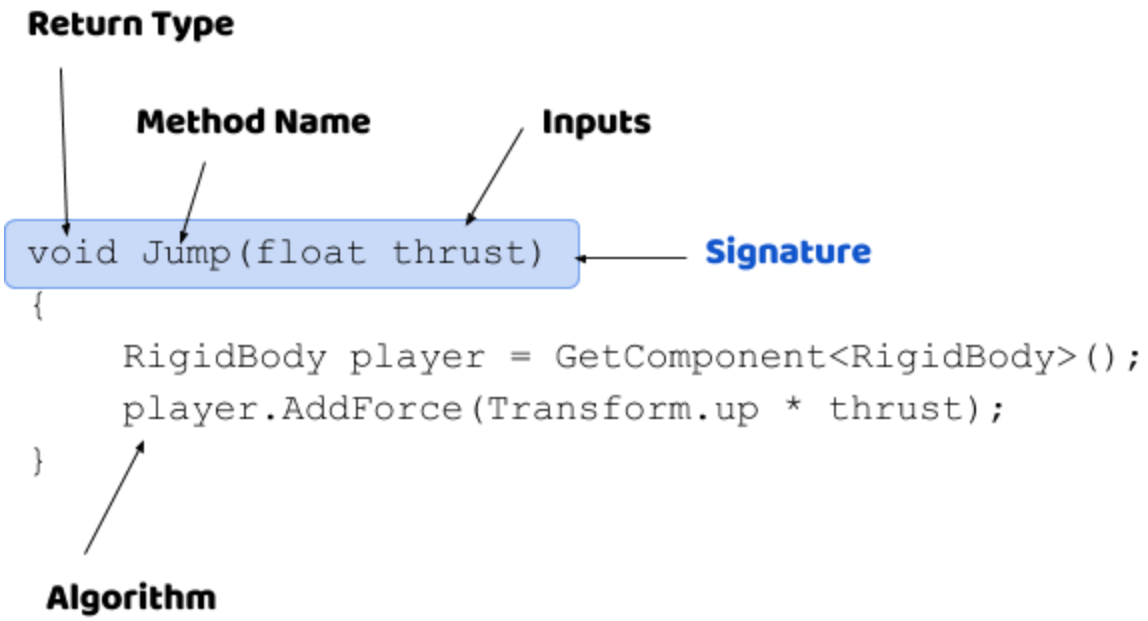
C# Basics: Methods

So you want to design some code that does something, more than likely you're designing a Method. Methods are the verbs of our programming language.



In the above example we have defined a method named Jump. What the code does is add an upward force of 'thrust' units on the player. Note thrust is the name of a variable, and whatever value we assign it is what will be used. So if thrust is set to 1000, the player is thrust upwards 1000 units.

You can think of a Method as a machine box, where you put something in (Input) and the machine does some work (Algorithm) and you get back something (output) And its up to you how you want to design your method.

Input

If you need Inputs you design it to have inputs. For example you may have a Method you are designing called DrawLine, and what it does is draws lines! But you may want to ask as Input what size line you want to draw, where you want the line to start and where it should end.

You could even get creative and ask what color you want it to be drawn. There are soooo many ways to design a method and its all up to your creativity!

Algorithm

This is the meat of a method, this is a step by step instruction telling the computer what to do. And you can be creative here too! There are many ways you can make a Method work, or another way of saying it there are many Algorithms that can make a method.

For example: If the method you want to design is DriveToSchool, you can choose to make an algorithm that goes to school in a straight line (going right through buildings) or an algorithm that follows the rules of the road!

Output

Methods are usually designed for two reasons,
1) To do something (Algorithm only)
2) To do some calculation and return the a variable with the value.

For example: We can design a method that calculates how much health we have called CheckHealth, and it returns a float with our health amount and takes no Inputs.

```
float CheckHealth(){...]
```

Syntax

With methods there is the syntax of defining/designing a method, and there is syntax of using it

Designing a Method

Return Type

this is a variable type that is typed right at the beginning of the design. If there is no return type we can write void. Examples of return type are int, float, string, void etc.

Method Name

We write this after our Return Type. This is the name of our method, because every method needs a name or we don't know what to call it. You can call your method whatever you like, but good names are those that tell us what the method does. After the Method Name you must put a pair of parentheses, and we put the inputs inside them.

Inputs

In the parentheses after Method Name you put your inputs needed, if any. If you have no inputs leave empty parentheses. Each input is a variable with a type and a name. We write each input needed and name it. We separate each Input with a comma

```
void DrawLine(int firstPoint, int secondPoint)
```

Signature

If we take the three parts of Return type, Method Name, and Inputs and put them on one line it's called a Method Signature. In order to use a Method later you need to know or share with someone its signature!

Body

After writing the signature of a function we put 2 curly braces after it. Any code inside those curly braces is called the Algorithm. All code for a method must be in between these curly braces.

Using a Method

To use a predefined Method you simply write the name of the method and put parentheses around it and put in any inputs it needs. After you have used the method be sure to put a semicolon. If you have a Method that returns a value you can assign it to a variable you have made by using the '='.

Signature

```
void Jump()
```

Usage

```
Jump();
```

Signature

```
void DrawLine(int firstPoint, int secondPoint)
```

Usage

```
DrawLine(29,33);
```

Signature

```
float CheckHealth()
```

Usage

```
float currentHealth = CheckHealth();
```