Methods as Arguments

Before we get into the next shortcut, we need to understand how methods are passed to other methods as arguments. This is possible and sometimes necessary in C#.

For example, say we need to check if there are even values in an array (you don't need to know much about arrays here, except that they are lists of values).

First you need an array of values and the `IsEven()` method that returns `true` if its argument is even:

```
int[] numbers = {1, 3, 5, 6, 7, 8};

public static bool IsEven(int num)
{
  return num % 2 == 0;
}
```

Pass both of these as arguments to the method `Array.Exists()`, which returns a boolean value:

```
bool hasEvenNumber = Array.Exists(numbers, IsEven);
```

You can see that `IsEven`, a method, is passed as the second argument to `Array.Exists()`.

In the background, this is what `Array.Exists()` does:

> The `IsEven()` method is called with each value in the array. We can imagine each of these being called:
>
> ```
> IsEven(1);
> IsEven(3);
> IsEven(5);
> IsEven(6);
> ```
>
> If any of these return `true`, `Array.Exists()` returns `true`.

By the end, `Array.Exists()` returns `true` because `isEven(6)` returns `true`.

There are other methods that accept methods as arguments, which you will encounter later on. For now, you need to understand that we can use a method's name like a variable, e.g. `IsEven` is a variable representing the method `IsEven()`. We pass this variable to another method, like `Array.Exists()`, which will probably invoke that method-argument at least once within its own body.

☑Instructions

1.

`Array.Find()` is another method that takes an array and a method as arguments. `Array.Find()` calls the method on each element of the array and returns the first element for which the method returns `true`.

An array `adjectives` and method `IsLong()` are defined for you. Call `Array.Find()` with these two arguments to find the first element in `adjectives` that is "long".

Store the returned string in a variable named `firstLongAdjective`.

Hint

Here's how to use the method:

```
Array.Find(adjectives, IsLong);
```

With the above code, the `IsLong()` method will be called with each value in the array. We can imagine each of these being called:

```
IsLong("rocky");
IsLong("mountainous");
```

`mountainous` is returned because `IsLong("mountainous")` it is the first to return `true`.