Define Parameters

Remember calling methods with arguments, like `Math.Min(3, 4)`? Methods that you define can use arguments as well, making them more versatile and useful.

While we are defining our method, we don't know the actual argument values that will be used when calling the method. But we do know the expected data type and how it will be used. We can use this information to define a *parameter*, which sort of works like a variable within a method. Imagine it as a placeholder for the actual argument value.

```
static void YourMethodName(string identity)
{
  Console.WriteLine(identity);
}
```

The `YourMethodName()` method now has one parameter named `identity` of type `string`.

Separate multiple parameters with commas:

```
static void YourMethodName(string identity, int age)
{
  Console.WriteLine($"{identity} is {age} years old.");
}
```

When you call your method, the values to be used for each parameter are called *arguments*. In this case `"Yoda"` and `900` are arguments for the `identity` and `age` parameters.

```
YourMethodName("Yoda", 900);
```

☑Instructions

1.
The `VisitPlanets()` method has been re-written for you here.

Add an `int` parameter named `numberOfPlanets` to the method.

Hint

The first line of the method will look like:

```
static void VisitPlanets(int numberOfPlanets)
```

2.

Change the method body so that it uses the parameter. If someone were to call your method, it should print how many planets they visited. For example, calling `VisitPlanets(3)` would cause this to be printed:

```
You visited 3 new planets...
```

Hint

You'll need to use string interpolation in the body of the method, which requires `$"{}"` syntax. In this example, `multiplier` is a variable:

```
$"He had a heart {multiplier} sizes too small"
```

3.

Call `VisitPlanets()` three times in `Main()` with different arguments.