Dictionaries

Tracking related data

So far, we've worked with single variables of various types and have discovered arrays and lists. There is another kind of variable that is very useful named the `Dictionary`.

Lookups and Values

A `Dictionary` in `C#` acts much like a physical dictionary, something that has information to lookup (perhaps a word) and an associated kind of information (perhaps a definition).

When creating a dictionary, we need to tell `C#` the type of our lookup key and the type of the associated information.

Example

Let's say we are trying to store the score associated with each player on our team. In this case, the lookup type will be a `string` (for the name), and the associated type will be an `int` (for the score).

Defining a Dictionary

That definition looks like:

```
var playerScores = new Dictionary<string, int>();
```

Adding to a Dictionary

And we can add information like this:

```
playerScores.Add("Robbie Lakeman", 1_247_700);
```

Finding information in a Dictionary

We can look up information with the same [] bracket syntax.

```
var score = playerScores["Robbie Lakeman"];
```

Missing Keys

However, if we look up a key that doesn't exist, we receive an exception, and our program stops. Later on, we'll see how to avoid, as well as handle, these kinds of errors.

Missing Keys

We can ask the `Dictionary` if it has a key.

```
var hasKey = playerScores.ContainsKey("Billy Mitchell"); // false
```

Changing information in a dictionary

Let's say that Robbie's score has increased by `100` to `1_247_800`, and we want to update this information.

We can use the `[]` lookup syntax on the **left hand side** of an assignment to *set* a value in a dictionary.

```
playerScores["Robbie Lakeman"] = 1_247_800;
```

Keys are case sensitive (value-sensitive)

Since `strings` are case-sensitive, we need to be careful when using them as keys.

The value at `playerScores["Robbie Lakeman"]` is different from the value at `playerScores["robbie lakeman"]`.

Looping through a dictionary

Like a `List`, a `Dictionary` can be looped through.

```
foreach (var playerScore in playerScores)
{
    Console.WriteLine($"{playerScore.Key} has a score of {playerScore.Value}")
}
```

- Each element in the dictionary has a key part and a value part
- The variable `playerScore` will have a type known as a `KeyValuePair`
- The `KeyValuePair` has two properties, the `Key` and the