Lesson Summary

Key Points

- **Variables** are pieces of reusable data.
- They are typically written in **camelCase**, which starts with a lower case letter, after which each new word is capitalised without a space.
- **Reference Types** point to data that exists elsewhere (i.e. multiple variables can point to one thing).
- **Common reference types** include game objects, components, classes, and assets.
- **Value Types** exist where they are declared (i.e. assigning one value to another variable of that type copies it, creating multiple instances).
- **Common value types** include numbers, text, booleans, vectors, and structs.
- Variables can be **public**, **private** or **serialised**. Most of the time, if you only want to view a variable in the Inspector, you can simply serialise it with the **[SerializeField]** attribute.
- **Collections** refer to multiple items of data in the same variable, and include **Arrays** (which are generally fixed) and **Lists** (which can be resized and changed).
- **Initialisation** refers to a variable's default value. Object types, including collections, are null by default, while values will be the default value for their type, such as false, or zero.
- Variables can be set or initialised using the **assignment operator =** which can be used to set one variable to the value of another or to a **literal** type, such as a number or string of text.

Basic Variables

```
1. using UnityEngine
2.
3. public class Player : MonoBehaviour
4. {
5.     [SerializeField] float playerHealth = 100;
```

```
6.      [SerializeField] bool isDead;
7.
8.      public void Damage(float dmg)
9.      {
10.         playerHealth -= dmg;
11.
12.         if(playerHealth < 0)
13.         {
14.             // Player Has Died
15.             isDead = true;
16.         }
17.     }
18.
19.     public void AddHealth(float health)
20.     {
21.         playerHealth += dmg;
22.
23.         if(playerHealth > 100)
24.         {
25.             // Player's health is full
26.             playerHealth = 100;
27.         }
28.     }
29. }
```