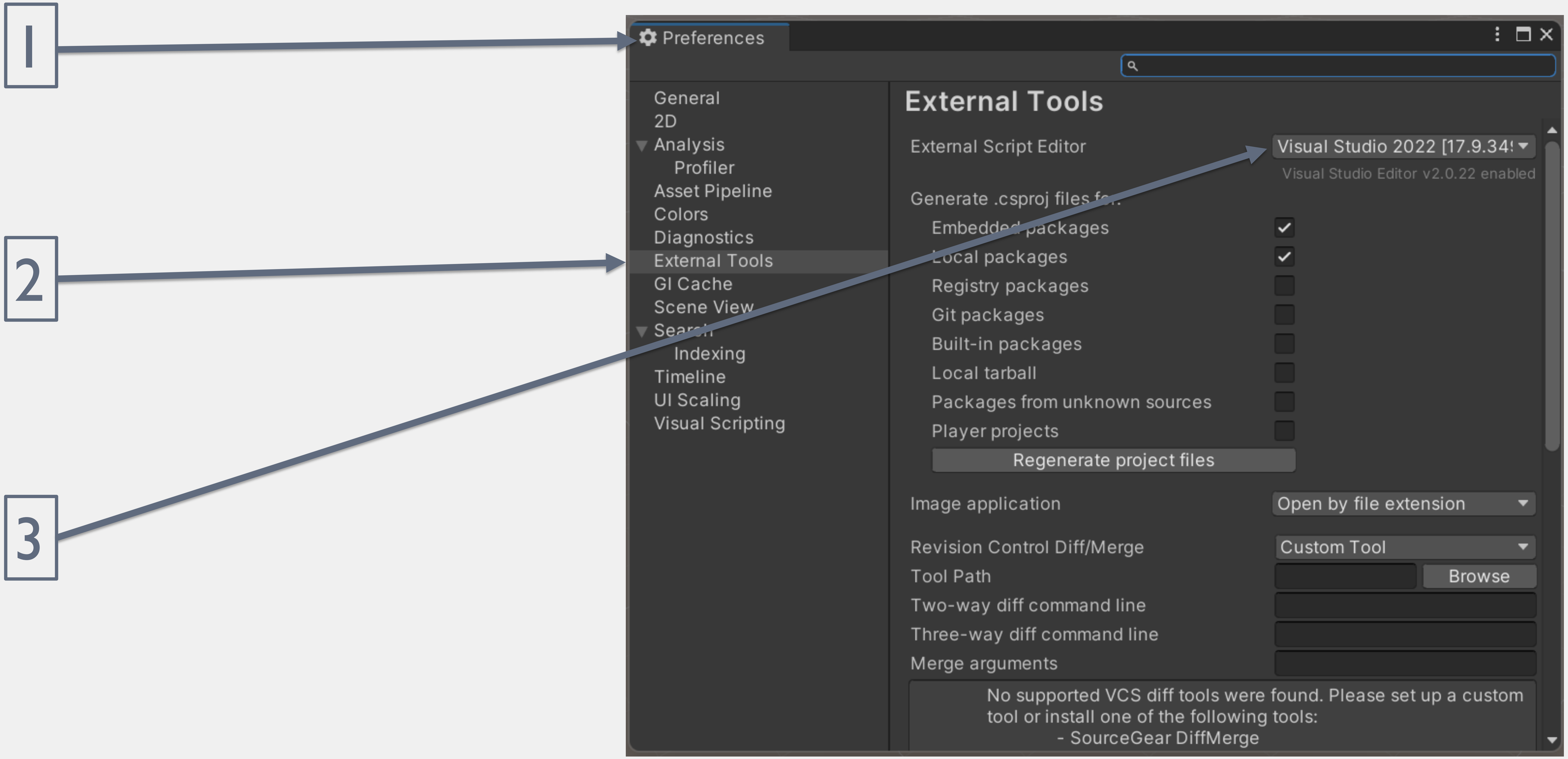
Scripting Basics

Kate Wang & Mieszko Muskala

Scripting in Unity

- Set Visual Studio as External Script Editor



Preferences

General
2D
Analysis
Profiler
Asset Pipeline
Colors
Diagnostics
External Tools
GI Cache
Scene View
Search
Indexing
Timeline
UI Scaling
Visual Scripting

External Tools

External Script Editor: Visual Studio 2022 [17.9.34!

Visual Studio Editor v2.0.22 enabled

Generate .csproj files for:

- ☑ Embedded packages
- ☑ Local packages
- □ Registry packages
- □ Git packages
- □ Built-in packages
- □ Local tarball
- □ Packages from unknown sources
- □ Player projects

Regenerate project files

Image application: Open by file extension

Revision Control Diff/Merge: Custom Tool

Tool Path: Browse

Two-way diff command line

Three-way diff command line

Merge arguments

No supported VCS diff tools were found. Please set up a custom tool or install one of the following tools:
- SourceGear DiffMerge

Scripting in Unity

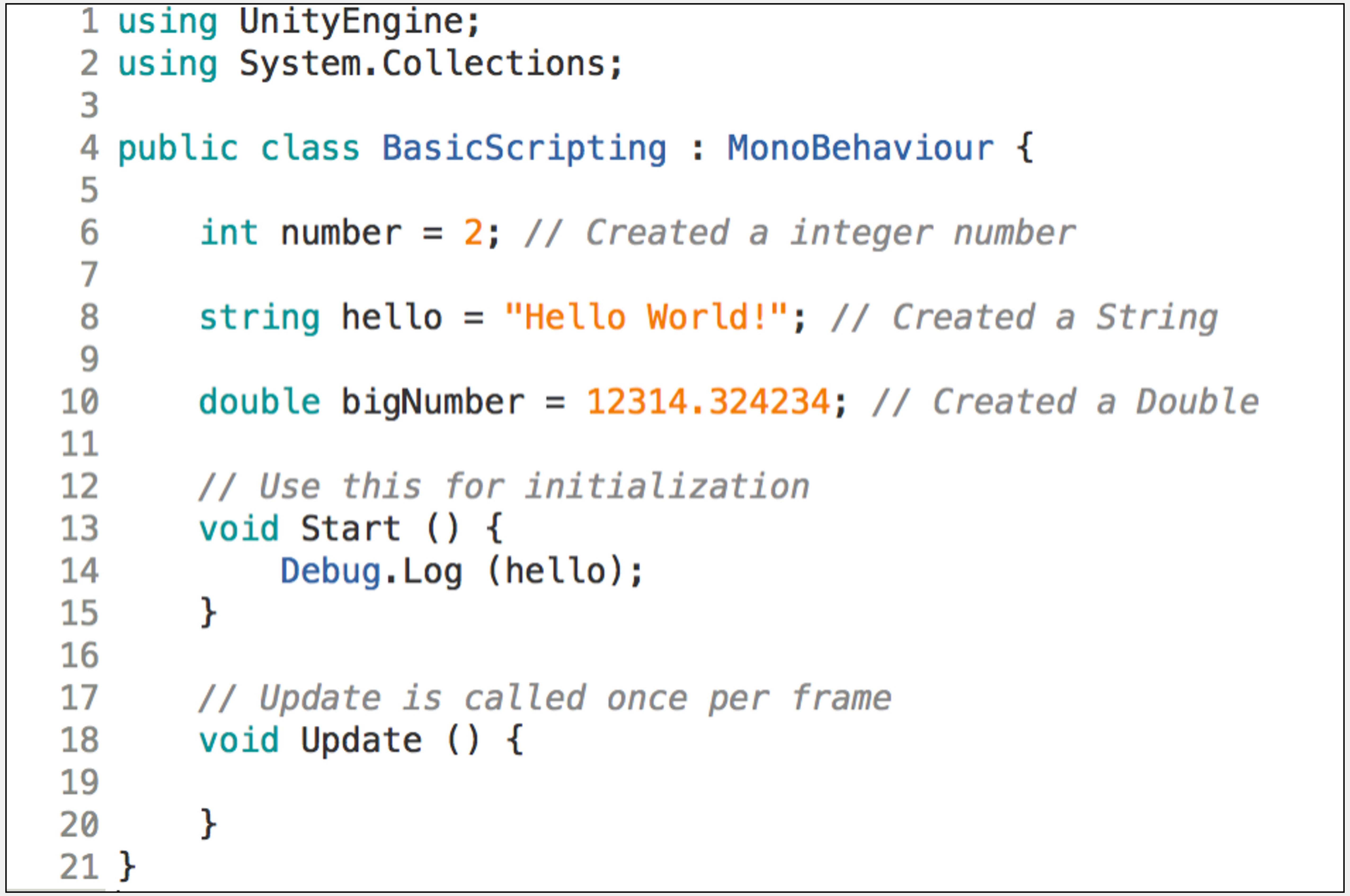
- Scripting allows developers to extend functionality
- Unity uses C#
- C# syntax is almost identical to what you learned in C++
- Typically scripts apply to a game object

Scripts as Behavior Components

- Scripts can be added to objects as components
- Scripts are used to create behavior
  - Change size or color
  - Apply intelligence to an object

Variables

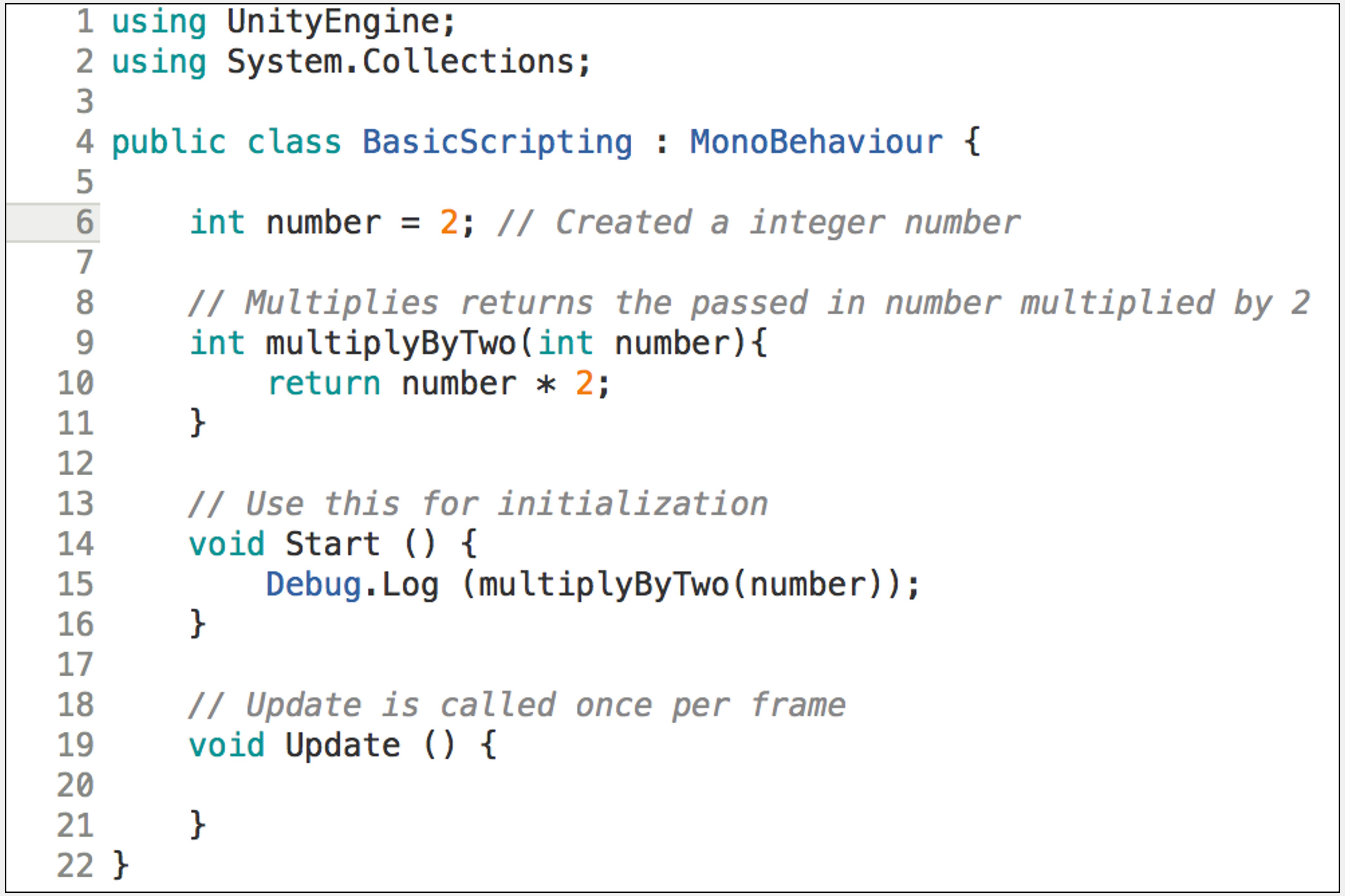
{type} variableName = {value};



```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    int number = 2; // Created a integer number

    string hello = "Hello World!"; // Created a String

    double bigNumber = 12314.324234; // Created a Double

    // Use this for initialization
    void Start () {
        Debug.Log (hello);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

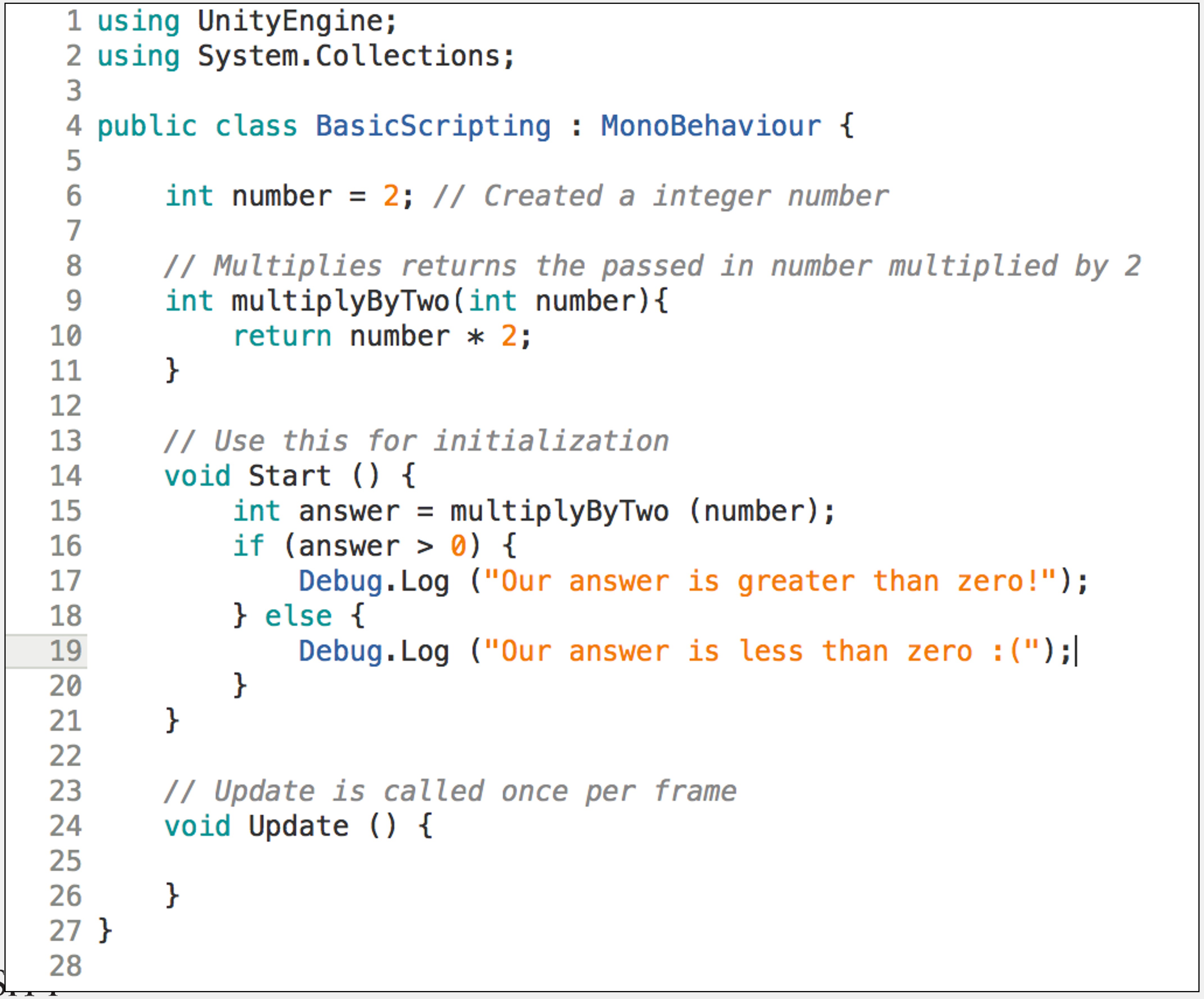
Functions

{type} functionName({type} inputVariable){ Do Something }



```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    int number = 2; // Created a integer number

    // Multiplies returns the passed in number multiplied by 2
    int multiplyByTwo(int number){
        return number * 2;
    }

    // Use this for initialization
    void Start () {
        Debug.Log (multiplyByTwo(number));
    }

    // Update is called once per frame
    void Update () {

    }
}
```

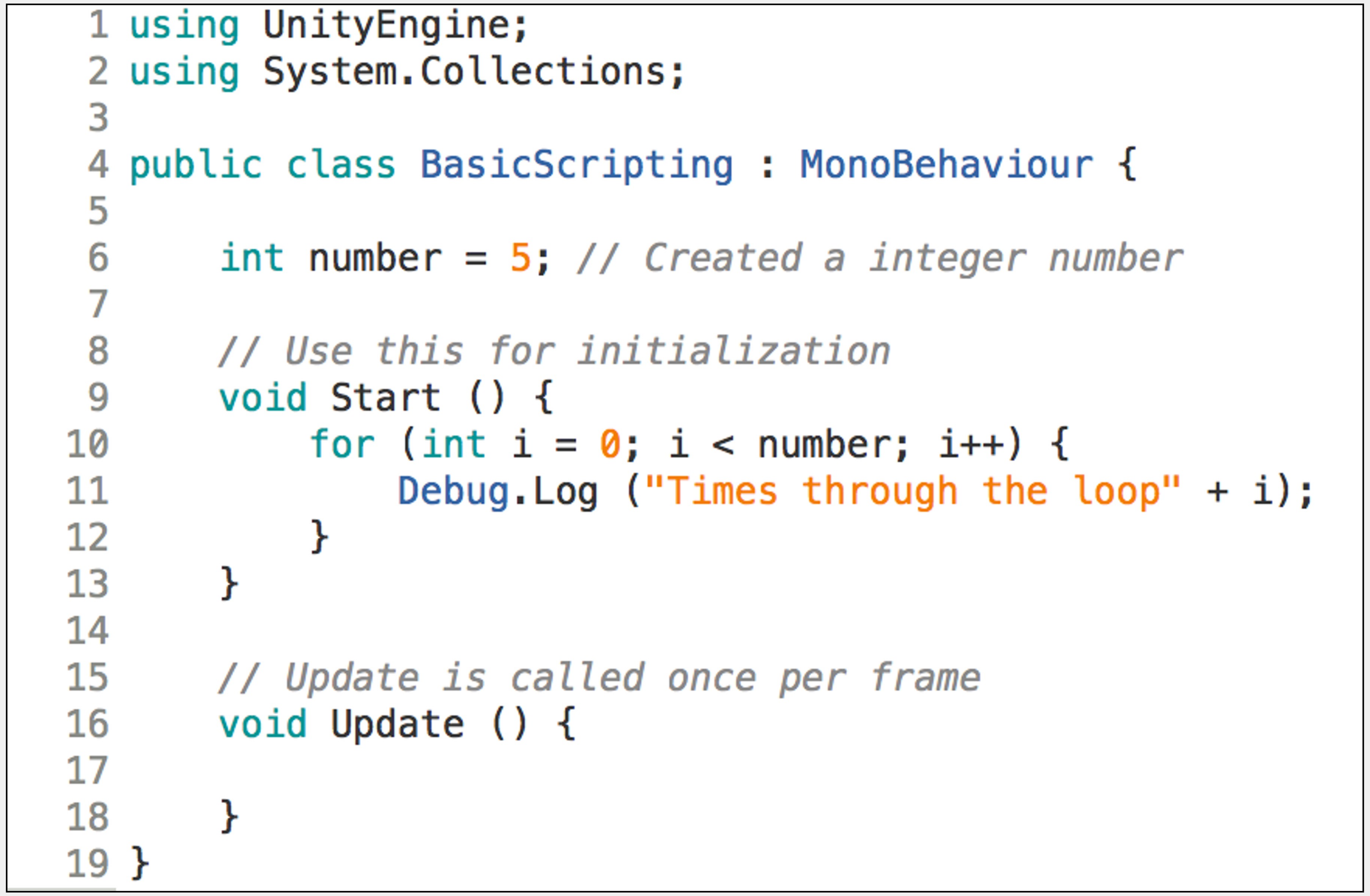
Conditionals



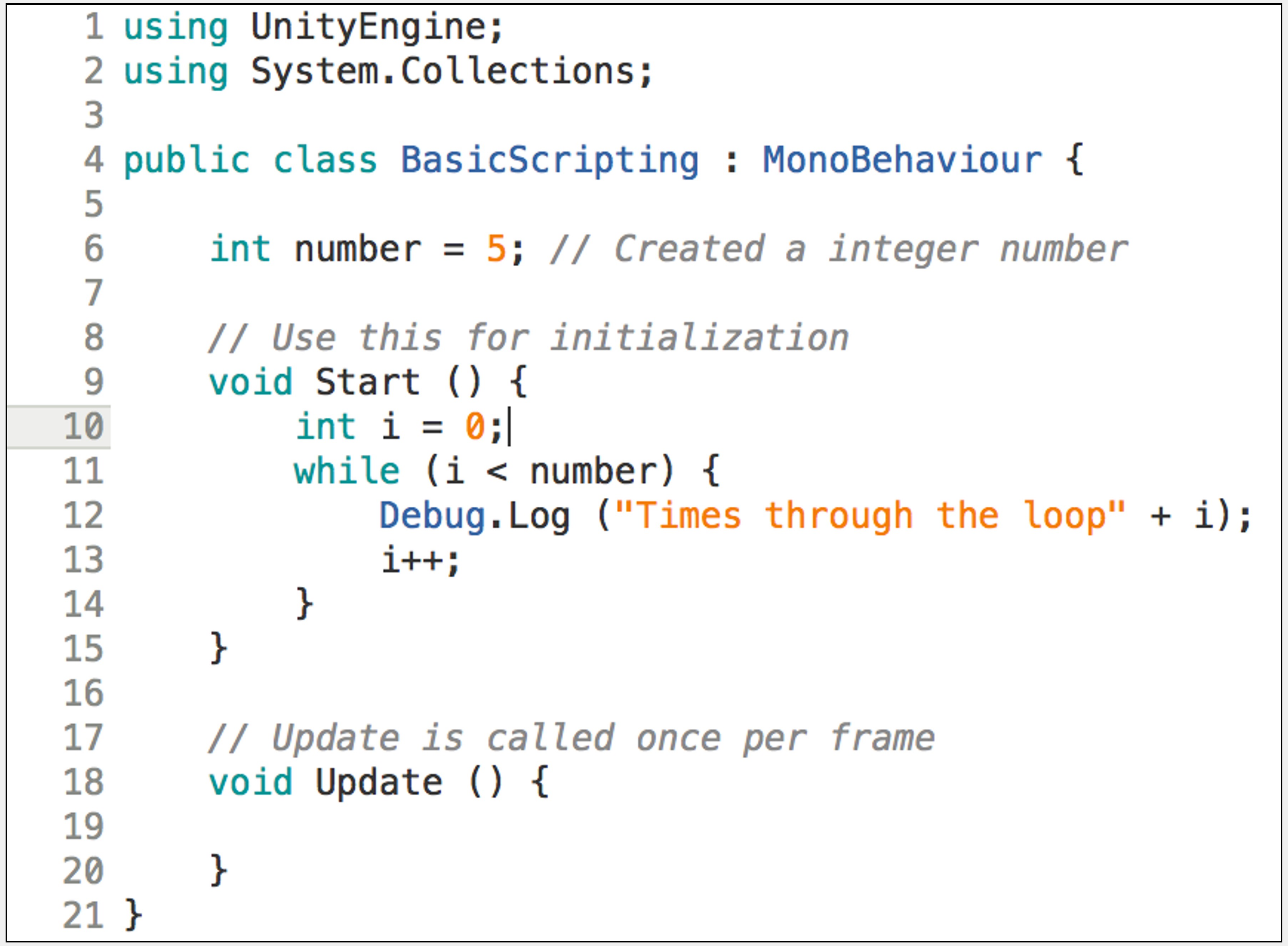
```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    int number = 2; // Created a integer number

    // Multiplies returns the passed in number multiplied by 2
    int multiplyByTwo(int number){
        return number * 2;
    }

    // Use this for initialization
    void Start () {
        int answer = multiplyByTwo (number);
        if (answer > 0) {
            Debug.Log ("Our answer is greater than zero!");
        } else {
            Debug.Log ("Our answer is less than zero :(");
        }
    }

    // Update is called once per frame
    void Update () {

    }
}
```

Loops

- Supports for, while, and do-while loops



```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    int number = 5; // Created a integer number

    // Use this for initialization
    void Start () {
        for (int i = 0; i < number; i++) {
            Debug.Log ("Times through the loop" + i);
        }
    }

    // Update is called once per frame
    void Update () {

    }
}
```



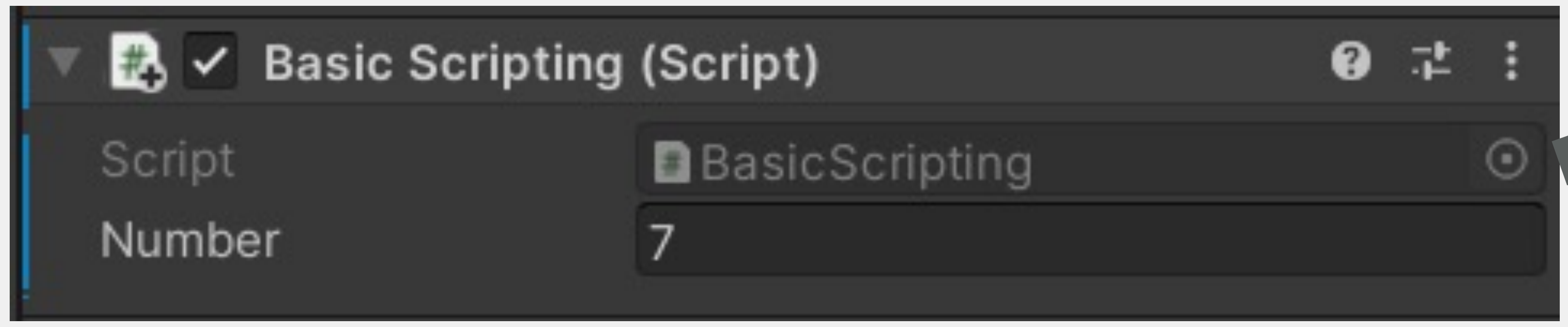
```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    int number = 5; // Created a integer number

    // Use this for initialization
    void Start () {
        int i = 0;
        while (i < number) {
            Debug.Log ("Times through the loop" + i);
            i++;
        }
    }

    // Update is called once per frame
    void Update () {

    }
}
```

Passing in Values Through Inspector

- Set a variable to public

Or

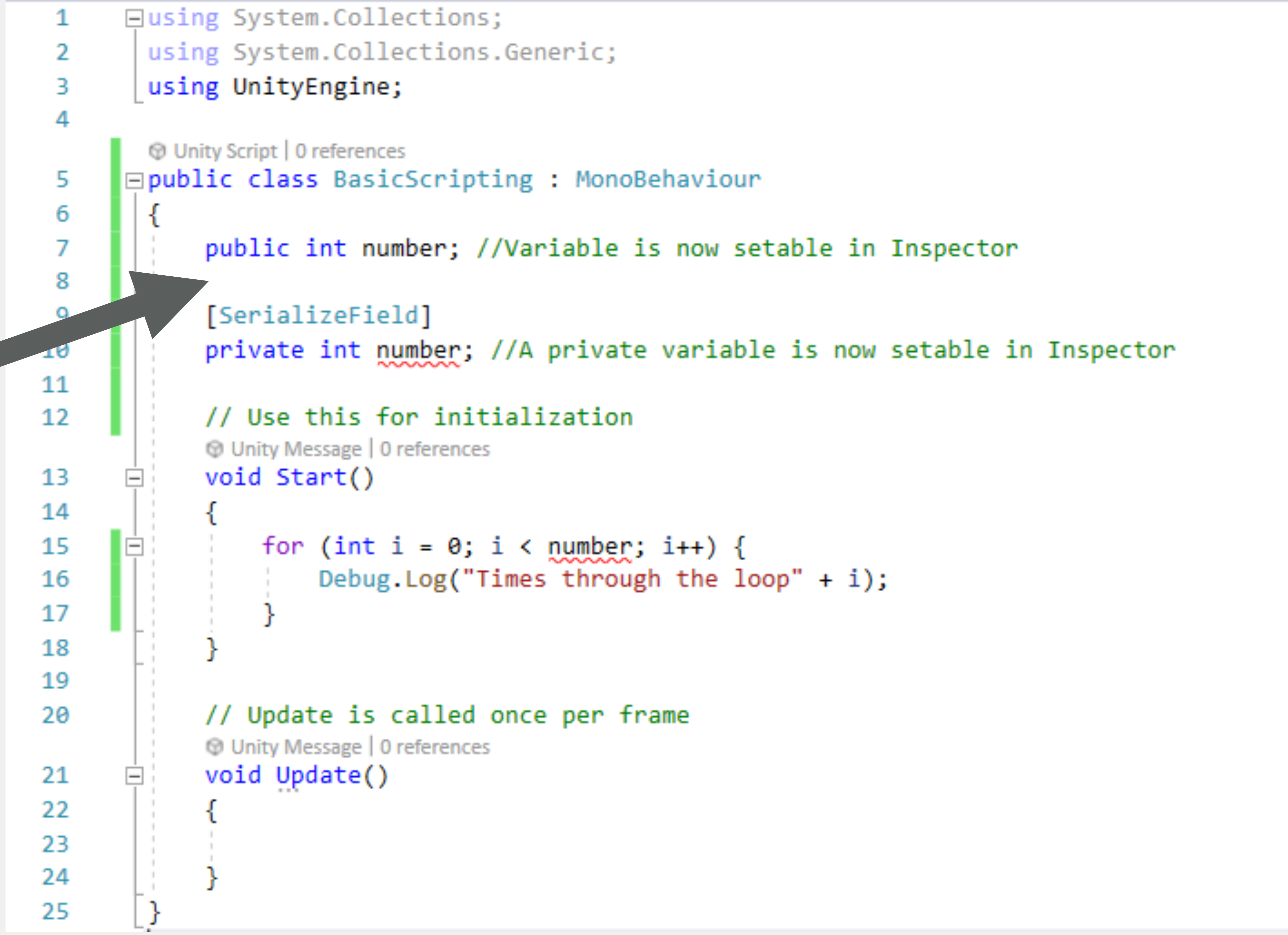
- Set a variable as SerializeField



Basic Scripting (Script)

Script: BasicScripting

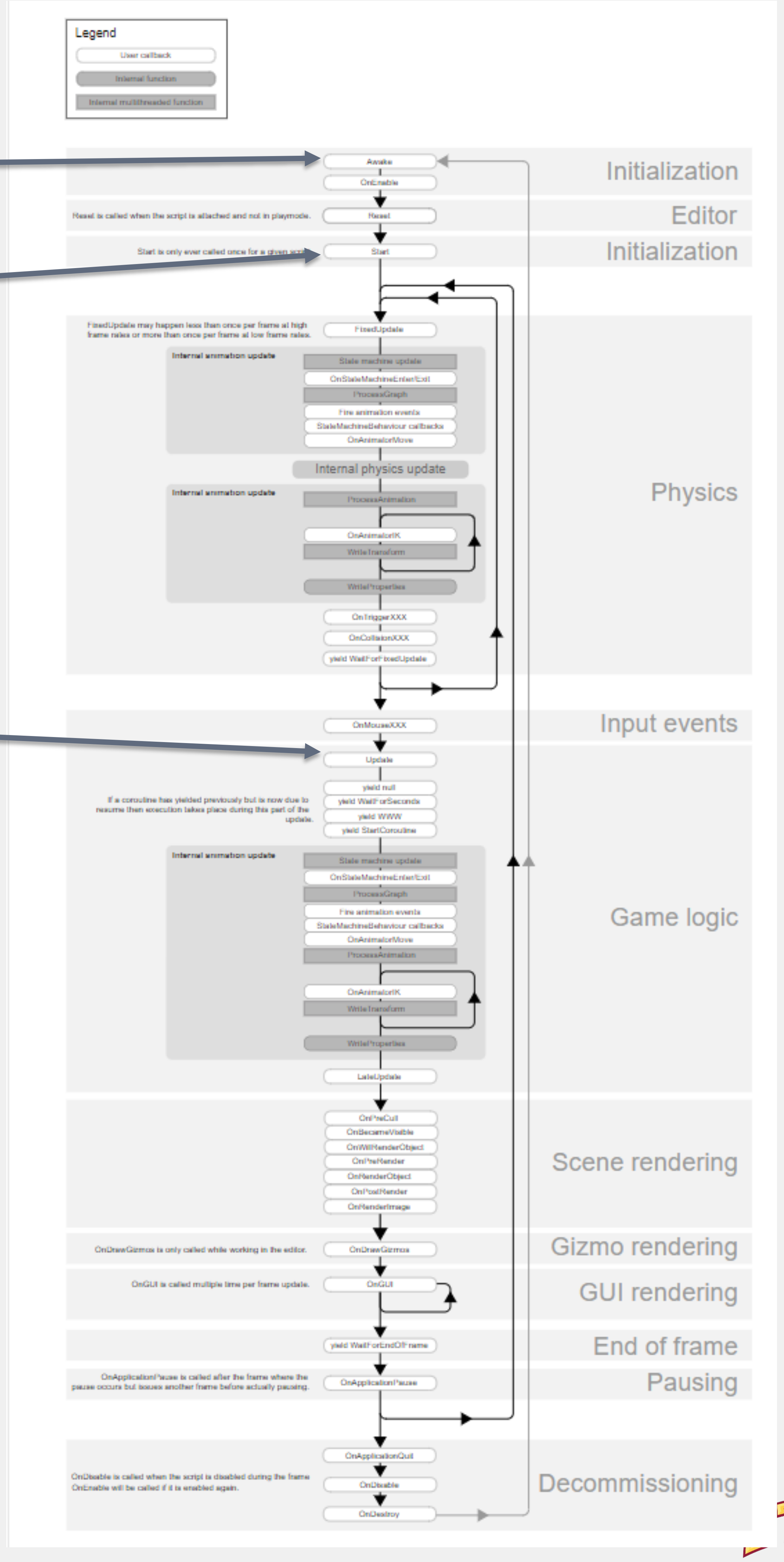
Number: 7



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class BasicScripting : MonoBehaviour
{
    public int number; //Variable is now setable in Inspector

    [SerializeField]
    private int number; //A private variable is now setable in Inspector

    // Use this for initialization
    Unity Message | 0 references
    void Start()
    {
        for (int i = 0; i < number; i++) {
            Debug.Log("Times through the loop" + i);
        }
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {

    }
}
```

Built in Unity Functions

- Awake()
- Start()
- Update()
- Other functions (https://docs.unity3d.com/Manual/ExecutionOrder.html)


Legend
User callback
Internal function
Internal multithreaded function
Awake
OnEnable
Initialization
Reset is called when the script is attached and not in playmode.
Reset
Editor
Start is only ever called once for a given script.
Start
Initialization
FixedUpdate may happen less than once per frame at high frame rates or more than once per frame at low frame rates.
FixedUpdate
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
Internal physics update
Internal animation update
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
OnTriggerXXX
OnCollisionXXX
yield WaitForFixedUpdate
Physics
OnMouseXXX
Input events
Update
If a coroutine has yielded previously but is now due to resume then execution takes place during this part of the update.
yield null
yield WaitForSeconds
yield WWW
yield StartCoroutine
Internal animation update
State machine update
OnStateMachineEnter/Exit
ProcessGraph
Fire animation events
StateMachineBehaviour callbacks
OnAnimatorMove
ProcessAnimation
OnAnimatorIK
WriteTransform
WriteProperties
LateUpdate
Game logic
OnPreCull
OnBecameVisible
OnWillRenderObject
OnPreRender
OnRenderObject
OnPostRender
OnRenderImage
Scene rendering
OnDrawGizmos is only called while working in the editor.
OnDrawGizmos
Gizmo rendering
OnGUI is called multiple time per frame update.
OnGUI
GUI rendering
yield WaitForEndOfFrame
End of frame
OnApplicationPause is called after the frame where the pause occurs but issues another frame before actually pausing.
OnApplicationPause
Pausing
OnApplicationQuit
OnDisable is called when the script is disabled during the frame OnEnable will be called if it is enabled again.
OnDisable
OnDestroy
Decommissioning

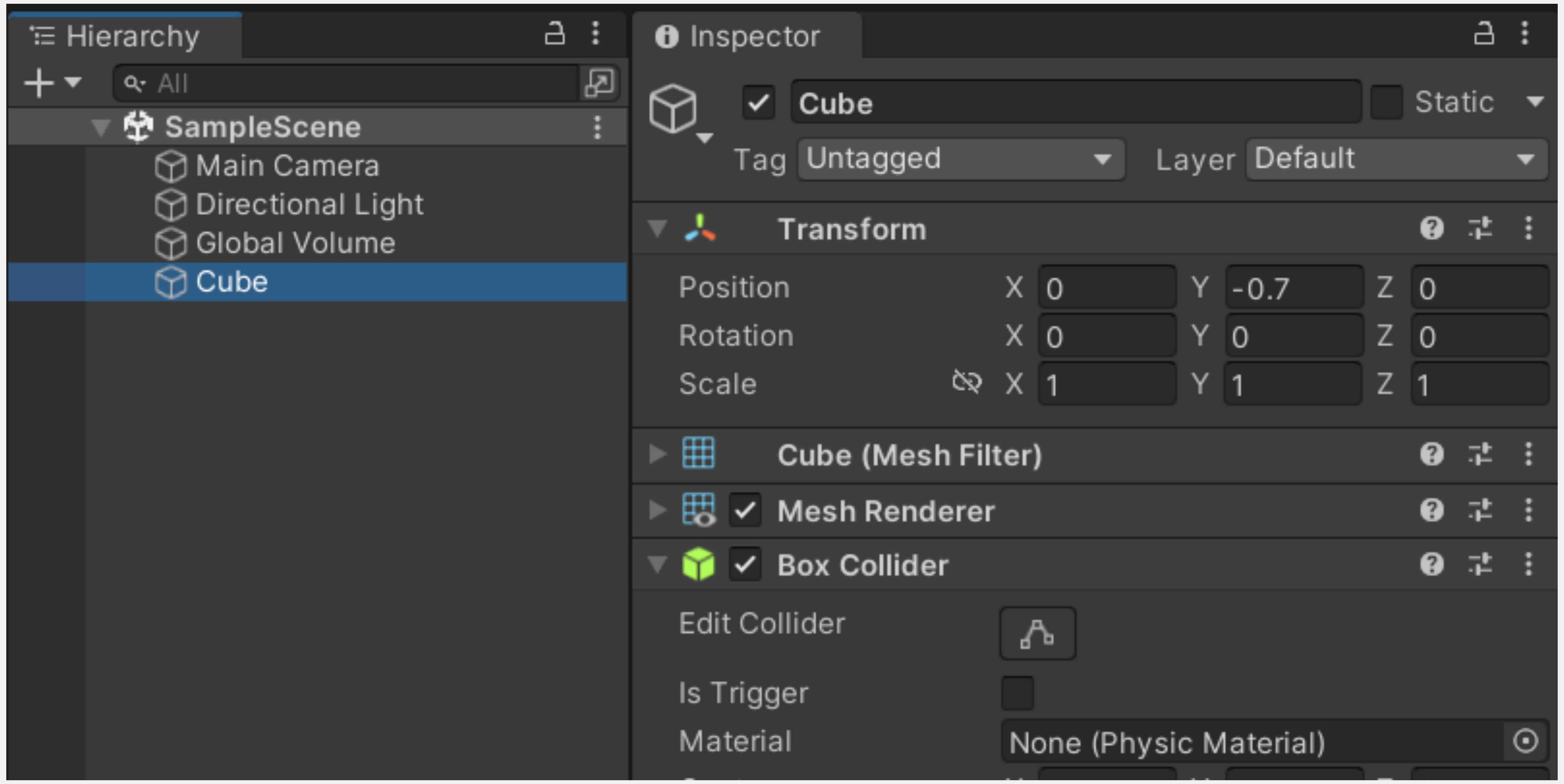
Awake Function

- A default function in Unity objects
- Should be treated like a class constructor
- Called once, only once, upon initialization

Start Function

- Called after Awake() upon the first frame if and only if the object is enabled
- Called once and only once
- Called before Update()

Getting a Component



Cube ☑ □ Static

Tag Untagged Layer Default

Transform

Position X 0 Y -0.7 Z 0

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Cube (Mesh Filter)

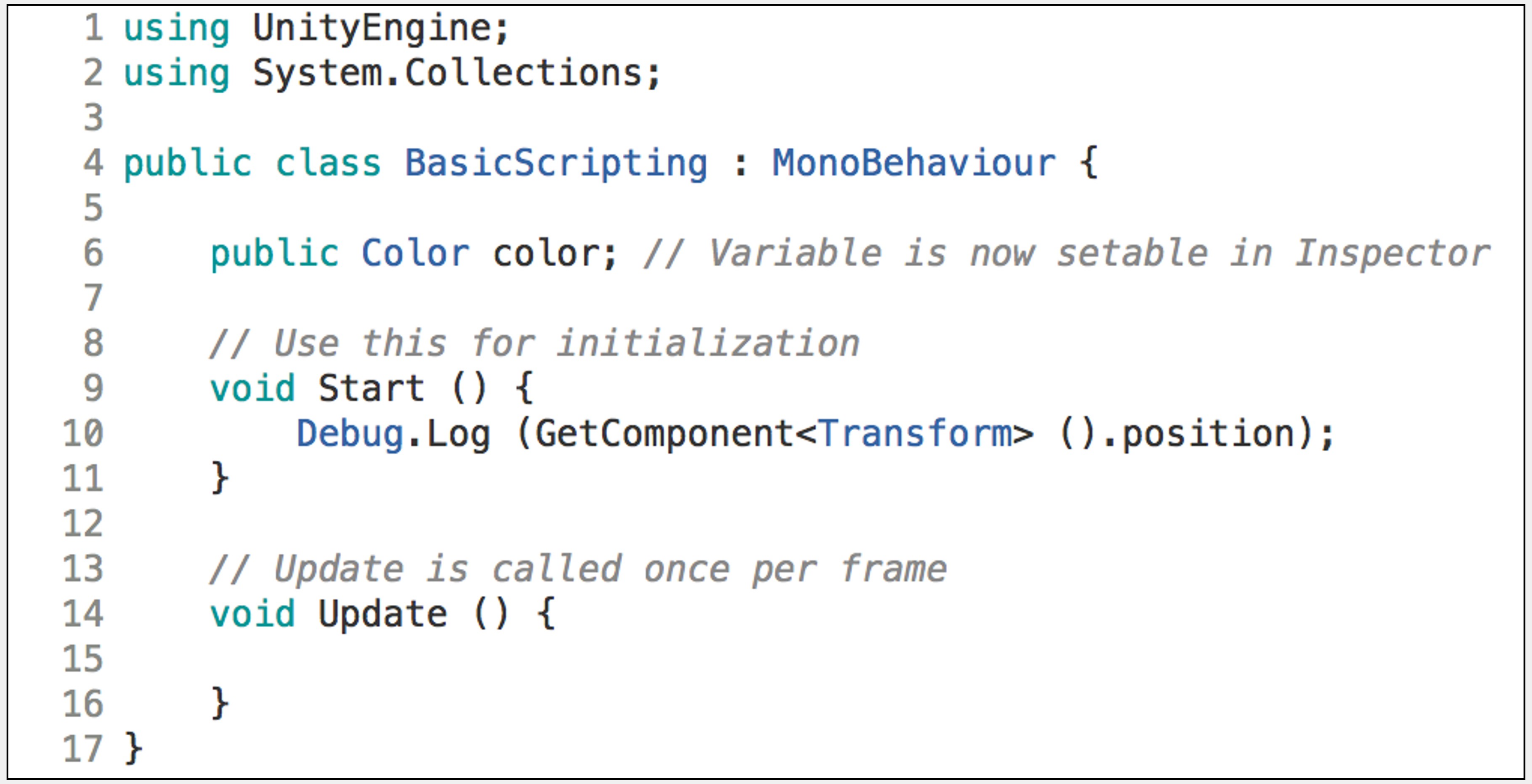
☑ Mesh Renderer

☑ Box Collider

Edit Collider

Is Trigger □

Material None (Physic Material)



```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    public Color color; // Variable is now setable in Inspector

    // Use this for initialization
    void Start () {
        Debug.Log (GetComponent<Transform> ().position);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

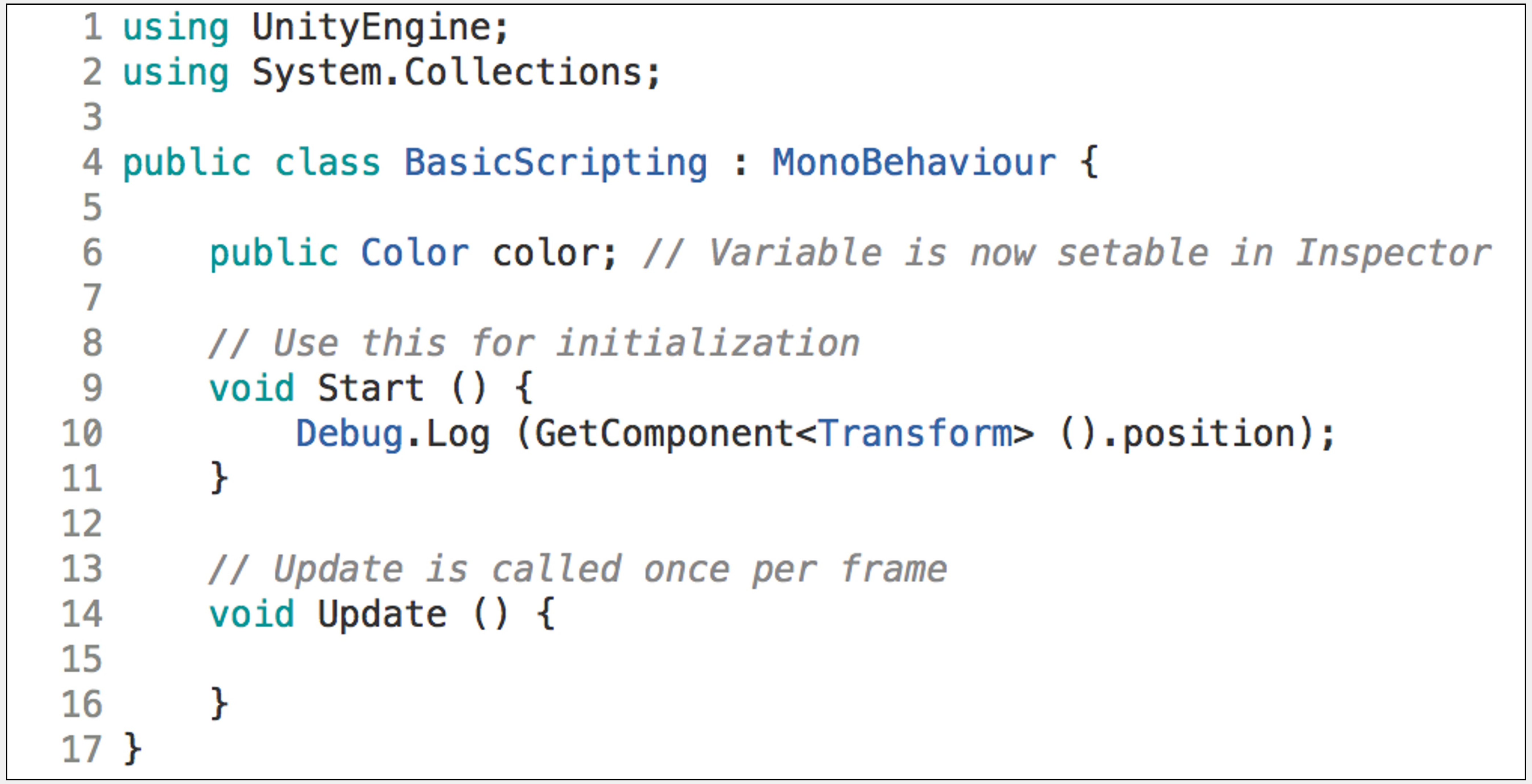
Update Function

- Update() is called every frame when the object is enabled
- This is the most used function in Unity
- Time.deltaTime gives you the amount of time since Update() was called last. Use this for animating!

Translate and Rotate Objects

- The Transform Component of an object holds its Position, Rotation, and Scale

- Use GetComponent<>() to change these values



```
using UnityEngine;
using System.Collections;

public class BasicScripting : MonoBehaviour {

    public Color color; // Variable is now setable in Inspector

    // Use this for initialization
    void Start () {
        Debug.Log (GetComponent<Transform> ().position);
    }

    // Update is called once per frame
    void Update () {

    }
}
```

In Class Activity

- Create a first-person view “Player” for your farm
- Implement a script so that WASD keys control your player to move forward & backward and rotate left & right
- Bonus: Enable mouse control for smooth player rotation
- Be creative!