 | DOCUMENTATION

Version: 5.4 ([switch to 5.5b](#))

Scripting API

- UnityEngine
 - UnityEngine.Advertisements
 - UnityEngine.Analytics
 - UnityEngine.Apple
 - UnityEngine.Assertions
 - UnityEngine.Audio
 - UnityEngine.Diagnostics
 - UnityEngine.Events
 - UnityEngine.EventSystems
 - UnityEngine.Experimental
 - UnityEngine.iOS
 - UnityEngine.Networking
 - UnityEngine.Purchasing
 - UnityEngine.Rendering
 - UnityEngine.SceneManagement
 - UnityEngine.Scripting
 - UnityEngine.Serialization
 - UnityEngine.SocialPlatforms
 - UnityEngine.Sprites
 - UnityEngine.Tizen
 - UnityEngine.UI
 - UnityEngine.VR
 - UnityEngine.Windows
 - UnityEngine.WSA
- Classes
 - AccelerationEvent
 - AnchoredJoint2D
 - AndroidInput
 - AndroidJavaClass
 - AndroidJavaObject

Camera

class in UnityEngine / Inherits from: [Behaviour](#)

[SWITCH TO MANUAL](#)

Description

A Camera is a device through which the player view the scene.

A screen space point is defined in pixels. The bottom-left corner is (0,0) and the top-right corner is (1024, 1024).

A viewport space point is normalized and relative to the camera.

A world space point is defined in global coordinates.

See Also: [camera component](#).

Static Variables

allCameras	Returns the array of all cameras in the scene.
allCamerasCount	The number of cameras in the scene.
current	The camera that is currently selected in the Hierarchy.
main	The first camera in the scene that is not disabled.
onPostRender	Event that is triggered after the camera has finished rendering the scene.
onPreCull	Event that is triggered before the camera culls the scene.
onPreRender	Event that is triggered before the camera starts rendering the scene.

Variables

actualRenderingPath	The rendering path used by the camera. The default value is "Default".
-------------------------------------	--

- Camera
- Color
- Collider
- Debug
- GameObject
- Light
- Material
- Mathf
- MeshCollider
- MeshRenderer
- MonoBehaviour
- PhysicMaterial
- Random
- ...

C#
(C Sharp)

Loops!

For Loop

```
for (int i = 0; i < 10; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

INITIALIZATION

CONDITION

INCREMENT



```
for (int i = 0; i < 10; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

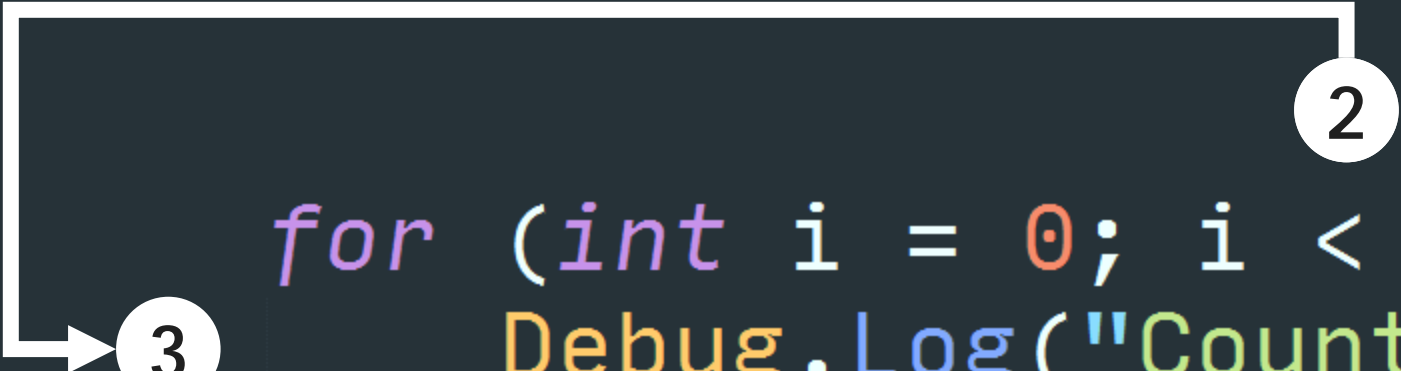
Loop Flow

1

```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```



```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

```
graph TD; 2((2)) -- loop --> 3((3)); 3 -- loop --> 2;
```

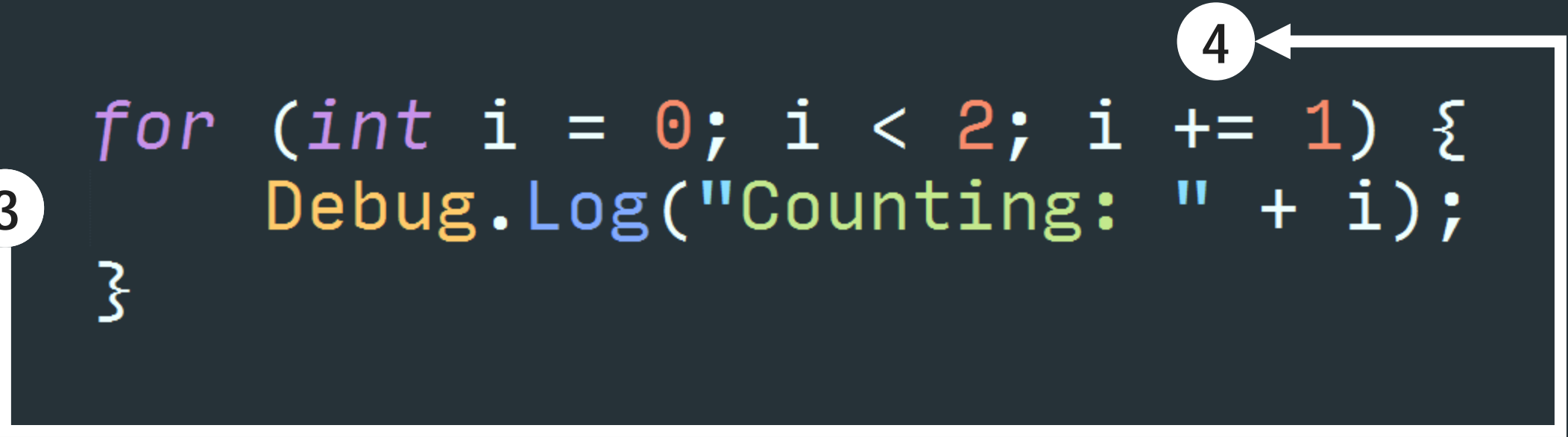
```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

The image shows a code snippet with a flowchart illustrating a loop iteration. The code is a C#-style for loop that iterates from i=0 to i=1. The flowchart consists of two circular nodes, 2 and 3, connected by a horizontal line. Node 2 is positioned above the closing brace of the loop, and node 3 is positioned to the left of the opening brace. A white arrow points from node 2 to node 3, and another white arrow points from node 3 back to node 2, forming a cycle that represents the loop's execution flow.

```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

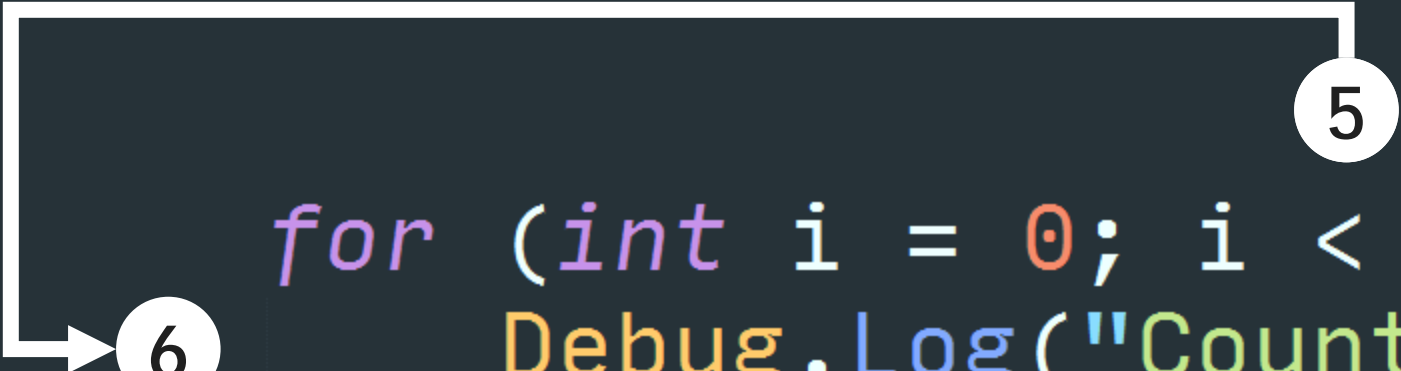
4

3





```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```



```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

6

7



```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

8

```
for (int i = 0; i < 2; i += 1) {  
    Debug.Log("Counting: " + i);  
}
```

→ LOOP OVER

More Loops

- Unity [tutorial](#)

Instantiate??



Object.Instantiate

```
public static Object Instantiate(Object original);  
public static Object Instantiate(Object original, Transform parent);  
public static Object Instantiate(Object original, Transform parent, bool worldPositionStays);  
public static Object Instantiate(Object original, Vector3 position, Quaternion rotation);  
public static Object Instantiate(Object original, Vector3 position, Quaternion rotation, Transform parent);
```

Parameters

original	An existing object that you want to make a copy of.
position	Position for the new object (default Vector3.zero).
rotation	Orientation of the new object (default Quaternion.identity).
parent	The transform the object will be parented to.
worldPositionStays	If when assigning the parent the original world position should be maintained.

Returns

Object A clone of the original object.

Casting & Manipulating

```
// Spawning and casting
Vector3 spawnPoint = new Vector3(1f, 0f, 0f);
Quaternion spawnRotation = Quaternion.identity;
GameObject clone = (GameObject) Instantiate(Prefab, spawnPoint, spawnRotation, transform);

// Now we have a GameObject, rather than an Object. We can use any of the methods
// available on a GameObject:

// Apply a random scale
Vector3 randomScale = new Vector3(1f, Random.Range(1f, 3f), 1f);
clone.transform.localScale = randomScale;
```

Arrays

```
int[] HighScores;
```



ARRAY TYPE



Ways to Create an Array

```
// Empty integer array  
int[] HighScores;
```

```
// Empty integer array with four element  
int[] HighScores = new int[4];
```

```
// Integer array with specific values  
int[] HighScores = { 10, 12, 15, 20 };
```

Resources

- Ray Wenderlich – [Video](#) on arrays
- Unity [tutorial](#) on arrays
- Blog [post](#): data structures in Unity and when to use them
- Unity [tutorial](#) on Lists and Dictionaries