Void Functions

- Accessibility relates to the visibility of variables and fields within their script, as well as their visibility to other scripts.
- Access modifiers are pieces of code that modify how accessible a block of code and its variables are to its script and other scripts.
- Two common modifiers are public and protected. Public types can be seen by other editors and scripts. They are completely available and accessible by everything. Protected is slightly different. Protected code can be seen by its type or any nested code blocks, but not ones outside its type. It is private otherwise.
- The most common modifier is private. Private means the code can only be accessed within its type. It prevents other scripts from accessing, reading, or executing it. It also prevents any code outside of the block from reading it. While one can explicitly declare code private, it isn't necessary. That is because it is the default setting for code in C#. If you don't specifically put a modifier in front of a declaration, it will be assigned a private status.
- The last two modifiers are internal and protected internal. Internal is only available to itself and associated assemblies. Non-nested code blocks default to this setting. Protected internal is essentially a combination of the protected and internal types. It is slightly more accessible than the internal or protected types by themselves.
- Iteration code blocks must always return something, unlike void functions. If you don't return something, you will err out.
- Void functions are a type of function. Void functions do not return anything. They just execute the code within its block and then terminate.

- public class voidExample : Monobehaviour {

    string voidFunc = “no return”;

    public void nest () {

        string voidFunc = “void functions do not return anything”

        print(voidFunc)

        }

}

- public class noReturn : Monobehaviour {

    public int voidThing = 5;

    public string truth;

        public void ohMan () {

            int no = 0;

            protected int return = 0;

            int answer = (no + return);

            print(answer);

        }

        public void learning () {

            truth = “void functions only execute the code and terminate. They

            do not return anything”

            print(truth);

        }

    }

- public class aVoid : Monobehaviour {

```
public int voidThing = 0;
        public void ohMan () {
                protected int voidReturn = 0;
                int answer = (voidThing + voidReturn);
                print(answer + “this is how many things void functions return cool
huh);
        }
}
```