Functions with Return

- 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.
- Two more 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.
- 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.
- 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.

- Functions that are not void functions must return a data type. They do work, then return a data type.
- The type of a method must return data of the same type. If there is an int function, the code must return an int. If it is a string, it must return a string.

-
  ```
  void Start () {
          myName = returnMeme (jumpStreet);
  }
  ```

```
public string jeff = “my name is jeff”;
string returnMeme (string _meme) {
        if(_meme == “jeff”) {
                return “mynamejeff”;
        } else {
                return “get meme’d”
        }
}
```

-
  ```
  void Start () {
          itUnlucky = ReturnNum (superstition);
  }
  ```

```
public int superstition = 13;
int ReturnNum (int _num) {
        if(_num == 13) {
                return _num;
        } else {
```

```
                break;
        }
}
```