Text Field delegate methods explained:

```
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    searchTextField.endEditing(true)
    print(searchTextField.text!)
    return true
```

textFieldShouldReturn is a method that returns a boolean, this function is called by the text field and is triggered when the user taps the return key or anything similar on the keyboard. Meaning the code inside the function is executed when the return key is pressed.

```
func textFieldDidEndEditing(_ textField: UITextField) {
    searchTextField.text = ""
}
```

textFieldDidEndEditing is a method that is executed after the user is done editing the text inside the text field this function can be used to change the placeholder of the text field or change components of the view after the user is done editing the text field. In this case, this function will be called in two places, one when the user taps the return key on the keyboard and when the user taps the search button on the screen resulting in clearance of the text field.

```
func textFieldShouldEndEditing(_ textField: UITextField) -> Bool {
    if textField.text != ""{
        return true
    }else {
        textField.placeholder = "Enter a City"
        return false
    }
}
```

textFieldShouldEndEditing is a method that allows the developer to basically trap the user in editing mode, this is useful when our app is dependent on the information from the user. In this case, we want the user to enter a city name so we can fetch the data to display it on the screen until the user hasn't entered a text we will not let them exit editing mode and show a message in the place holder. The reason in this method we haven't specified a particular text field is that we want this method to apply to all the text fields on the page.

Protocols:

Swift protocols define blueprints of methods and properties, but not an actual implementation, these blueprints can be adopted by classes, structs, and enums, in some cases, they can also act as a data type in functions. Protocols solve the problem of unnecessary inheritance from class, sometimes not all the functionality is needed from a class so implementing protocols makes code simpler and easier to maintain, in addition, structs can conform to protocols making them more functional since structs do not have inheritance. More than one protocol can conform to one object and they must be separated by commas. The term “Conform” is just a fancy way of saying, a struct or a class promises to have all the methods or properties declared in the protocol. Sometimes we will see protocols named as SomethinSomethingDelegate or SomethingSomehitngDataSource, this is a naming convention introduced by apple, these delegates and data sources are basically protocols, but protocols that have the delegate suffix generally have functions that deal with user interactions and data source protocols deal with data.