```
import tkinter as tk

def on_submit():
    result_label.config(text="Hello, " + entry.get())

window = tk.Tk()
window.title("Entry Widget Example")

entry = tk.Entry(window, width=30)
entry.pack(pady=10)

submit_button = tk.Button(window, text="Submit", command=on_submit)
submit_button.pack()

result_label = tk.Label(window, text="")
result_label.pack(pady=10)

window.mainloop()
```

This code creates a simple Tkinter GUI application that consists of an Entry widget, a Submit button, and a Label widget. Let's break down the code step by step:

1. Import Tkinter module:

```
import tkinter as tk
```

This line imports the Tkinter module, allowing us to create GUI elements.

2. Define the `on_submit` function:

```
def on_submit():
    result_label.config(text="Hello, " + entry.get())
```

This function is called when the Submit button is clicked. It retrieves the text entered into the Entry widget using the `get()` method and sets the text of the result_label accordingly.

3. Create the main window:

```
window = tk.Tk()
window.title("Entry Widget Example")
```

This code creates the main window of the application using `tk.Tk()`. The title of the window is set to "Entry Widget Example".

4. Create an Entry widget:

```
entry = tk.Entry(window, width=30)
entry.pack(pady=10)
```

This code creates an Entry widget, allowing users to input text. The `width` parameter specifies the width of the Entry widget in characters. The `pack()` method is used to add the widget to the main window and apply some padding (`pady=10`) to create space around it.

5. Create a Submit button:

```
submit_button = tk.Button(window, text="Submit", command=on_submit)
submit_button.pack()
```

This code creates a Button widget labeled "Submit". The `command` parameter is set to `on_submit`, meaning that when the button is clicked, the `on_submit` function will be called. The button is then added to the main window using the `pack()` method.

6. Create a Label widget for displaying the result:

```
result_label = tk.Label(window, text="")
result_label.pack(pady=10)
```

This code creates a Label widget initially with empty text. This label will be used to display the result of the submission. Like the other widgets, it's added to the main window using the `pack()` method with some vertical padding.

7. Start the Tkinter event loop:

```
window.mainloop()
```

This line starts the Tkinter event loop, which listens for user interactions such as button clicks and updates the GUI accordingly. The program will continue running until the user closes the main window.

When you run this script, it will create a window with an input field, a submit button, and a label. Users can enter text into the input field, click the submit button, and the label will display a greeting message based on the input.