Build Machine Learning Web Apps using Gradio

Ahmad Fathan Hidayatullah, S.T., M.Cs.

What is Gradio?

- Gradio is an open-source Python package to build web application for ML model, API, or any arbitrary Python function.
- Features:
  - Simple, quickly create web app, and easy to use.
  - No coding expertise required.
  - Easy sharing.

Installation

- Prerequisite: Gradio requires Python 3.8 or higher.
- Installing Gradio using pip, which is included by default in Python.
- Run this in your terminal or command prompt:

```
pip install gradio
```

- **Tips**:
  - It is best to install Gradio in a virtual environment.
  - Go through this link for more detailed installation instructions.

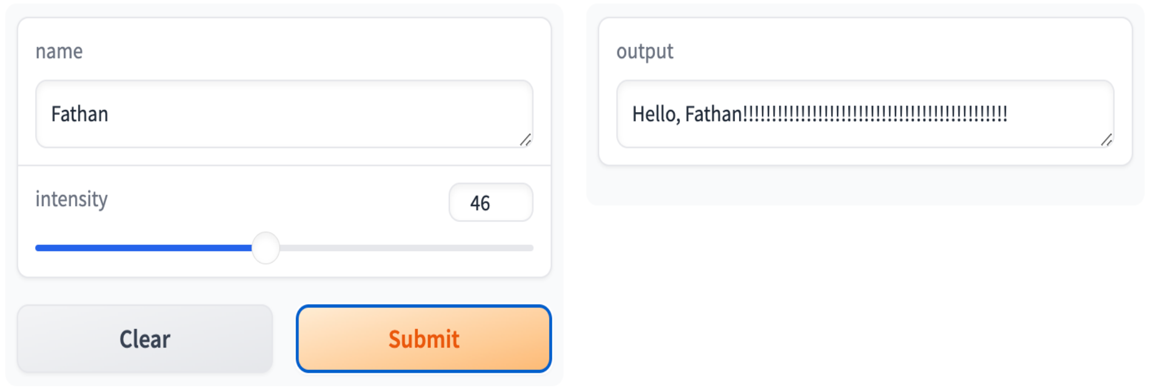
Building Your First Demo

- You can run Gradio in your favorite code editor, Jupyter notebook, Google Colab, or anywhere else you write Python.

```python
import gradio as gr


def greet(name, intensity):
    return "Hello, " + name + "!" * int(intensity)


demo = gr.Interface(
    fn=greet,
    inputs=["text", "slider"],
    outputs=["text"],
)


demo.launch()
```

Run Your Code

- If you've written the Python code in a file named, for example, **app.py**, then you would run python **app.py** from the terminal.
- The demo will open in a browser on **http://localhost:7860** if running from a file.
- If you are running within a notebook, the demo will appear embedded within the notebook.


name

Fathan

intensity 46

output

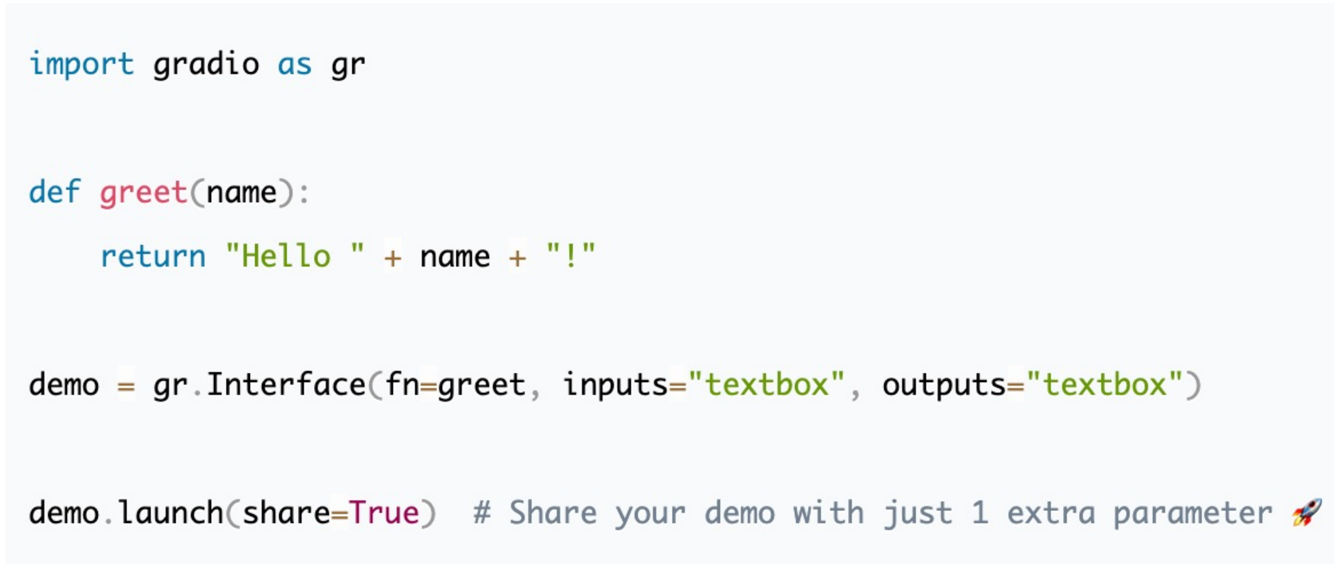
Hello, Fathan!!

Understanding the Interface Class

- The Interface class is designed to create demos for machine learning models which accept one or more inputs and return one or more outputs.
- The Interface class has three core arguments:
  - **fn**: the function to wrap a user interface (UI) around
  - **inputs**: the Gradio component(s) to use for the input. The number of components should match the number of arguments in your function.
  - **outputs**: the Gradio component(s) to use for the output. The number of components should match the number of return values from your function.

Sharing Your Demo

- Gradio lets you easily share a machine learning demo without having to worry about the hassle of hosting on a web server.
- Simply set **share=True** in launch(), and a publicly accessible URL will be created for your demo.

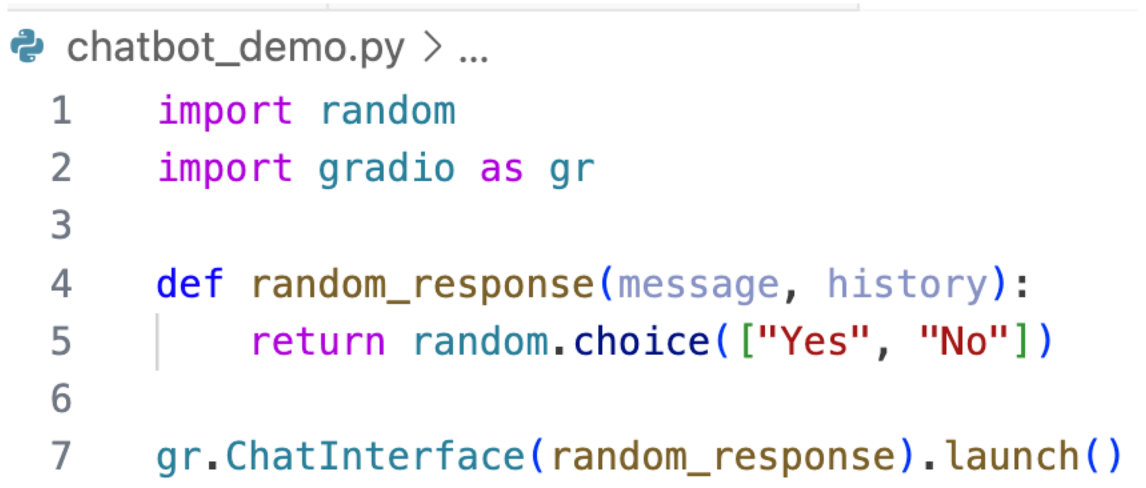


```python
import gradio as gr


def greet(name):
    return "Hello " + name + "!"


demo = gr.Interface(fn=greet, inputs="textbox", outputs="textbox")


demo.launch(share=True)  # Share your demo with just 1 extra parameter 🚀
```

What else does Gradio do?

Chatbots with **gr.ChatInterface**

- When working with gr.ChatInterface(), the first thing you should do is define your chat function.
- Your chat function should take two arguments: message and then history (the arguments can be named anything but must be in this order).
  - **message**: a str representing the user's input.
  - **history**: a list of list representing the conversations up until that point. Each inner list consists of two str representing a pair: [user input, bot response].



chatbot_demo.py > ...

```python
import random
import gradio as gr

def random_response(message, history):
    return random.choice(["Yes", "No"])

gr.ChatInterface(random_response).launch()
```

Custom Demos with **gr.Blocks**

- Gradio also offers a low-level approach for designing web apps with more flexible layouts and data flows with the **gr.Blocks** class.

```python
import gradio as gr


def greet(name):
    return "Hello " + name + "!"


with gr.Blocks() as demo:
    name = gr.Textbox(label="Name")
    output = gr.Textbox(label="Output Box")
    greet_btn = gr.Button("Greet")
    greet_btn.click(fn=greet, inputs=name, outputs=output, api_name="greet")

demo.launch()
```

Let's Practice