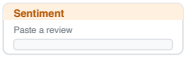
# Gradio

Gradio at a glance, wrapping any Python function in a shareable web UI from simple inputs to full chat apps.



## gr.Interface

Wrap one function in a UI with three arguments: fn, inputs, outputs.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| The whole demo in one line. | `gr.Interface(fn=greet, inputs="text", outputs="text")` |   |
| Multiple typed inputs. | `gr.Interface(fn, inputs=["text", gr.Slider(0, 3)], outputs="text")` |   |
| Multiple outputs. | `gr.Interface(fn, inputs="image", outputs=["label","text"])` |   |
| Title and description. | `gr.Interface(fn, "text", "text", title="Sentiment", description="Paste a review")` |   |
| Update live as you type. | `gr.Interface(fn, "text", "text", live=True)` |   |
| Launch it. | `gr.Interface(fn, "text", "text").launch()` |   |

gr.Interface(fn, inputs, outputs) renders a widget per input, calls fn in order, renders the returns; .launch() starts the server

## Input Components

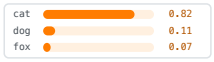
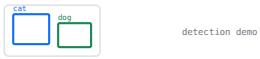
Typed widgets that become your function's arguments.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Free text input. | `gr.Textbox(label="Prompt", lines=3, ...)` |   |
| Numeric range. | `gr.Slider(minimum=0, maximum=1, value=0.7)` |   |
| Pick from a list. | `gr.Dropdown(choices=[...], multiselect=True)` |   |
| Boolean and choice. | `gr.Checkbox(...), gr.Radio(...)` |   |
| Upload an image. | `gr.Image(type="pil", sources=[...])` |   |
| Upload a file or audio. | `gr.File(...), gr.Audio(...)` |   |

each input component declares a type and becomes one positional argument of your fn, in the order you list them in inputs=[...]

## Output Components

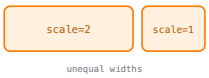
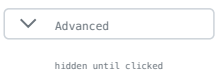
Widgets that render whatever your function returns.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Plain or rich text out. | `gr.Textbox(), Markdown(), HTML()` |   |
| Classification labels. | `gr.Label(num_top_classes=3)` |   |
| Structured data. | `gr.JSON(), gr.Dataframe()` |   |
| Media out. | `gr.Image(), Audio(), Video(), Gallery()` |   |
| A matplotlib or plotly figure. | `gr.Plot()` |   |
| Annotated image regions. | `gr.AnnotatedImage()` |   |

list outputs to match a tuple return; each component renders the matching return value, mirroring the inputs side

## gr.Blocks

Drop the Interface scaffold for full layout control.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Open a custom app. | `with gr.Blocks() as demo:` |   |
| Side-by-side columns. | `with gr.Row():`<br>`a = gr.Textbox(); b = gr.Textbox()` |   |
| Stacked columns. | `with gr.Column(scale=2): ...` |   |
| Tabbed sections. | `with gr.Tab("Settings"): ...` |   |
| Collapsible panel. | `with gr.Accordion("Advanced", open=False):` |   |
| Place a button to wire later. | `btn = gr.Button("Run", variant="primary")` |   |

with gr.Blocks() as demo: place components yourself, arrange with Row / Column / Tab / Accordion, then wire them with event listeners

## Event Listeners

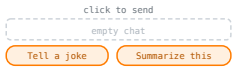
Bind fn -> inputs -> outputs to a trigger like .click or .change.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Run on button click. | `btn.click(fn=process, inputs=[a, b], outputs=out)` |   |
| React to a value change. | `slider.change(fn=update, inputs=slider, outputs=plot)` |   |
| Run when text is submitted. | `textbox.submit(fn=search, inputs=textbox, outputs=results)` |   |
| Run once on page load. | `demo.load(fn=load_data, outputs=table)` |   |
| Chain events with .then. | `btn.click(step1, ...).then(step2, ...)` |   |
| Update a component dynamically. | `return gr.update(visible=True, value="done")` |   |

listeners are methods on components: .click / .change / .submit / demo.load fire fn(inputs) -> outputs; chain with .then, mutate with gr.update

## gr.ChatInterface

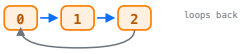
A full chat UI from one fn(message, history) (messages format).

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Chat demo in one call. | `gr.ChatInterface(fn=respond)` |   |
| The handler signature. | `def respond(message, history): return reply` |   |
| History is the messages format. | `history = [{"role": "user", "content": "hi"}, {"role": "assistant", "content": "hey"}]` |   |
| Stream tokens as they arrive. | `def respond(m, h): for tok in stream(): yield partial` |   |
| Seed example prompts. | `gr.ChatInterface(fn=respond, examples=["Tell a joke", "Summarize this"])` |   |
| Customize the transcript. | `gr.ChatInterface(fn=respond, chatbot=gr.Chatbot(height=500))` |   |

gr.ChatInterface(fn=respond) wraps respond(message, history) in a chat UI; history is messages-format dicts, yield to stream

## State, Examples & Flagging

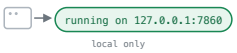
Remember per-session data, seed inputs, and capture feedback.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Per-session memory. | `count = gr.State(0)` |   |
| Read and write state. | `def inc(c): return c + 1`<br>`inputs=count, outputs=count` |   |
| Preset example inputs. | `gr.Interface(fn, "text", "label",`<br>`examples=[["good film"], ["awful"]])` |   |
| Cache example outputs. | `gr.Interface(fn, ..., examples=ex,`<br>`cache_examples=True)` |   |
| Turn on manual flagging. | `gr.Interface(fn, ..., flagging_mode="manual")` |   |
| Label the flag buttons. | `gr.Interface(fn, ...,`<br>`flagging_options=["wrong", "offensive"])` |   |

gr.State(initial) is per-session memory; examples=[...] seed clickable inputs; flagging_mode="manual" captures feedback to .gradio/flagged

## Launch & Share

Serve locally, get a public link, or mount on a server.

| PURPOSE | COMMAND | RESULT |
|---|---|---|
| Serve on localhost. | `demo.launch()` |   |
| Public share link (72h). | `demo.launch(share=True)` |   |
| Pick host and port. | `demo.launch(server_name="0.0.0.0",`<br>`server_port=8080)` |   |
| Password-protect the app. | `demo.launch(auth=("user", "pass"))` |   |
| Reload on file save (dev). | `gradio app.py` |   |
| Mount inside FastAPI. | `gr.mount_gradio_app(app, demo, path="/gradio")` |   |

launch() serves on 127.0.0.1:7860; share=True for a public *.gradio.live link, server_name/port and auth to control access, gradio app.py to auto-reload