

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

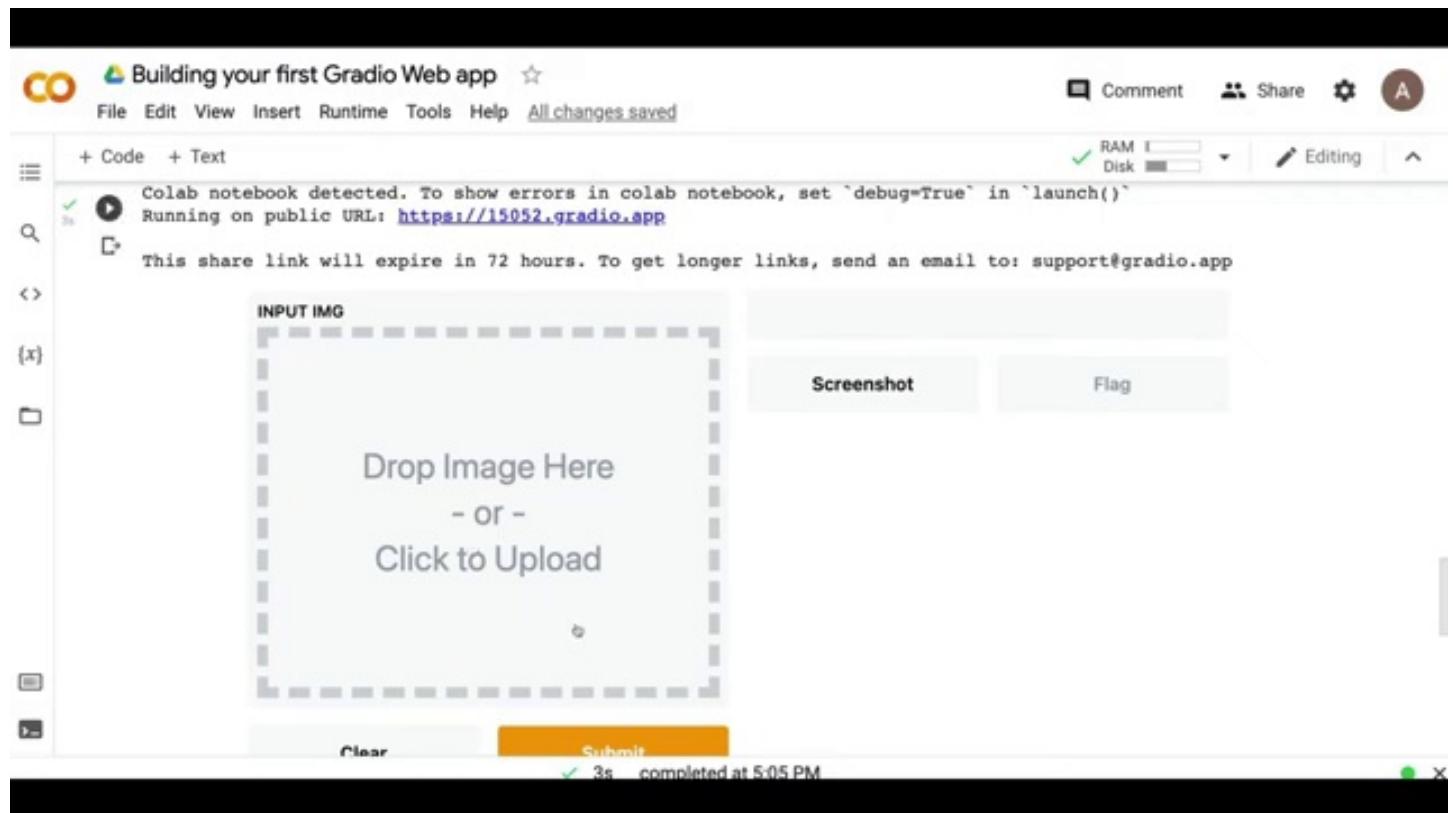
What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

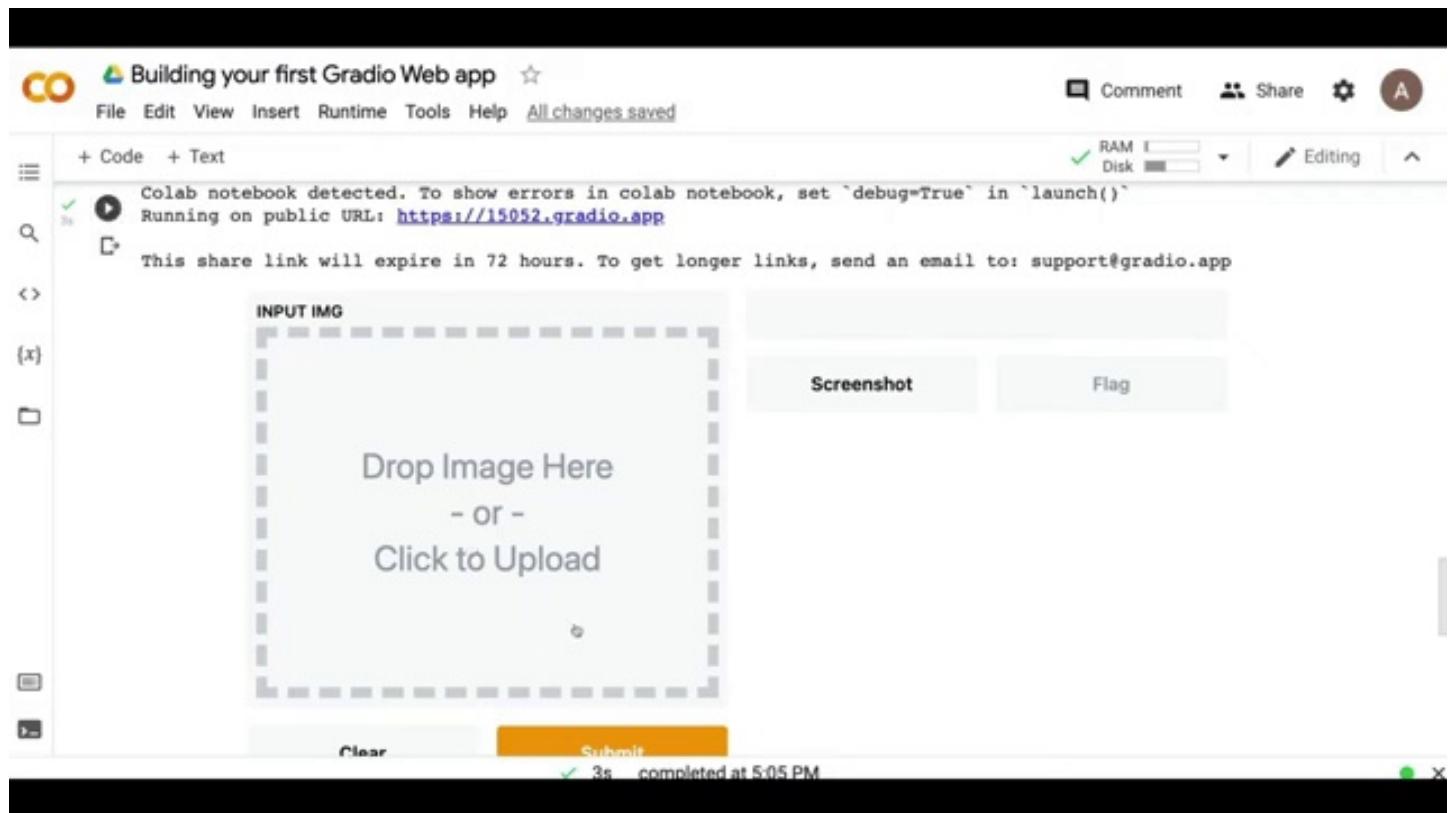
def greet(name):
    return "Hello " + name + "!"

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 0.00 seconds**



**Timestamp: 1.00 seconds**



**Timestamp: 2.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
[16] Running on public URL: <https://15052.gradio.app>

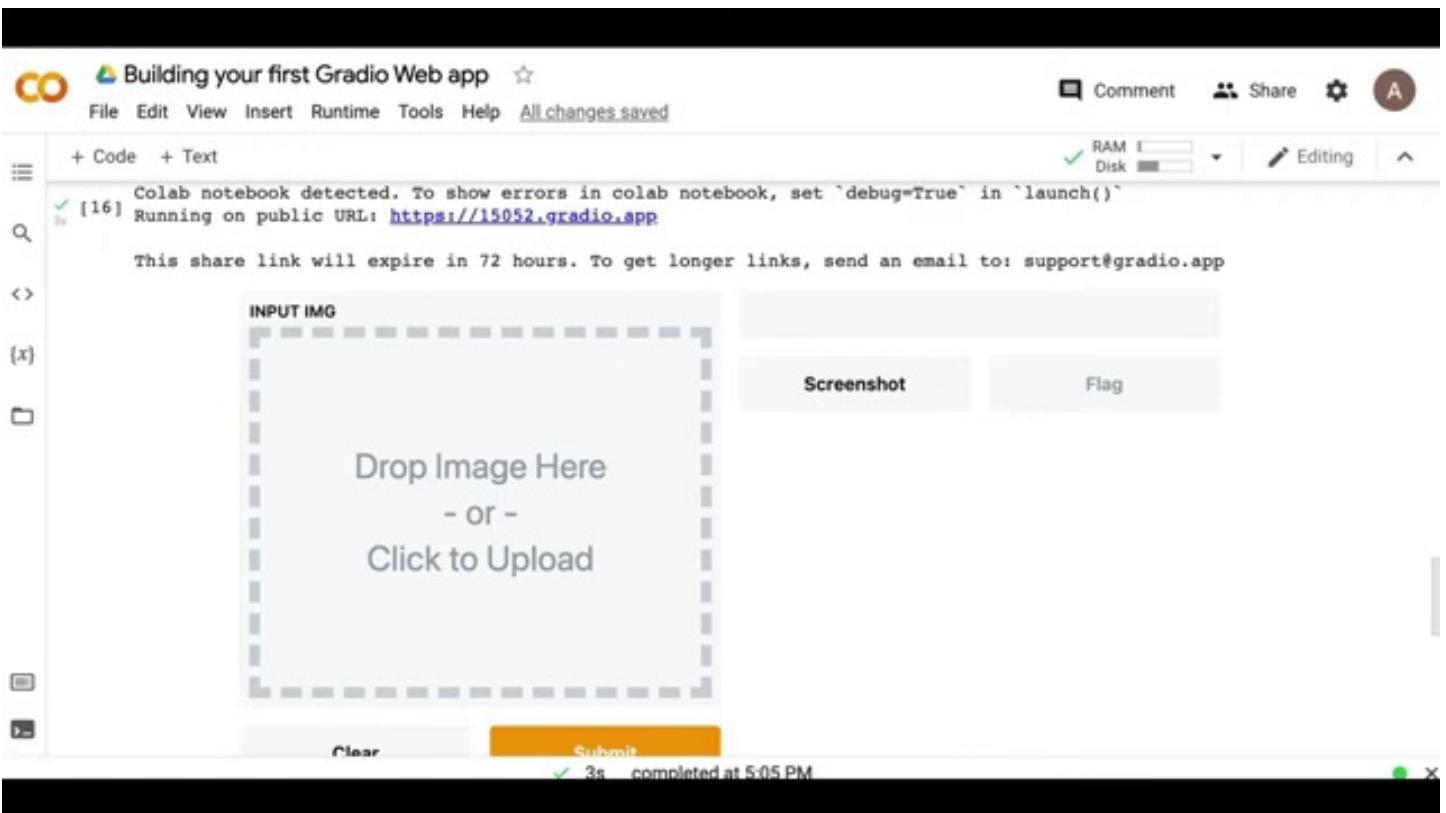
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG

Drop Image Here  
- OR -  
Click to Upload

Screenshot Flag

Clear Submit ✓ 3s completed at 5:05 PM



**Timestamp: 3.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
[16] Running on public URL: <https://15052.gradio.app>

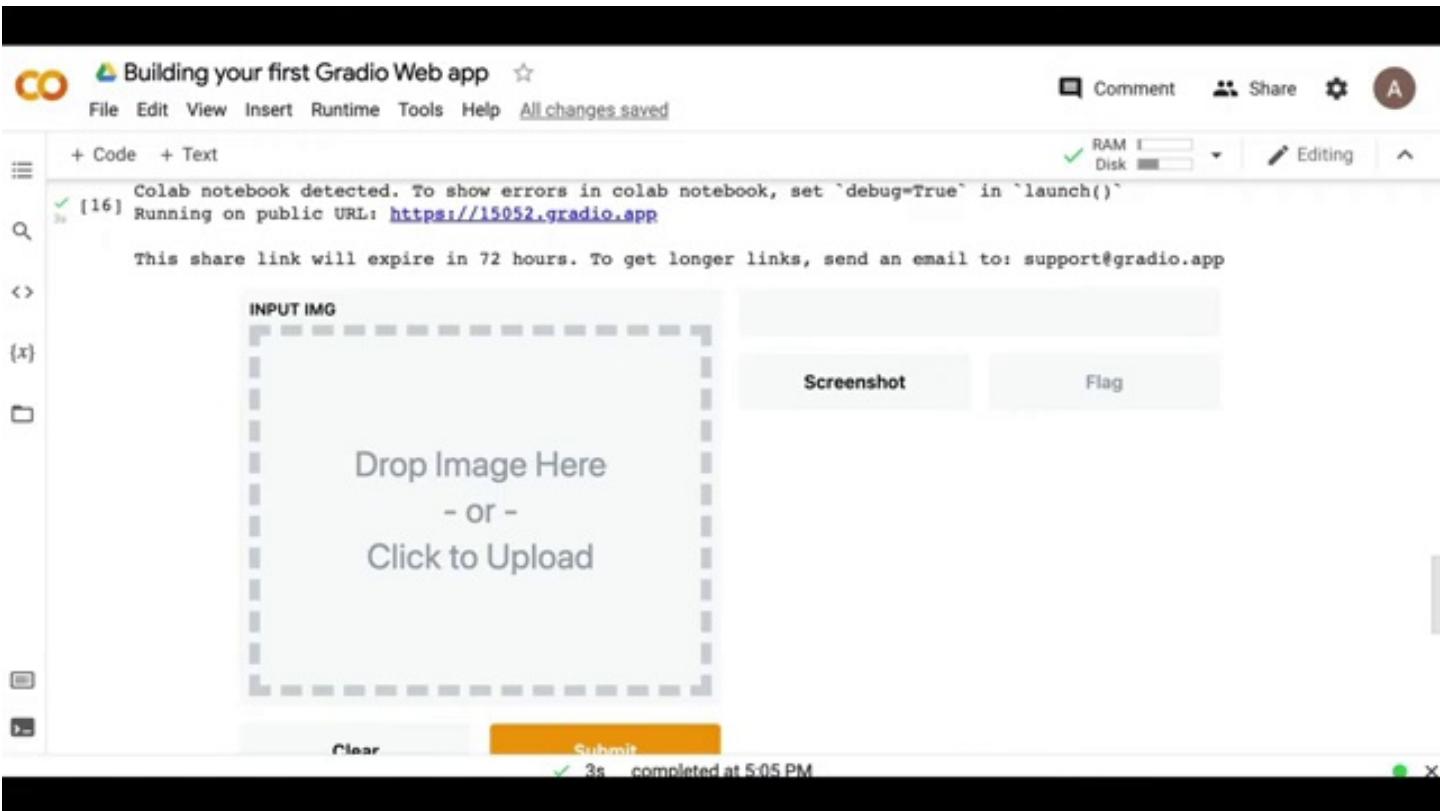
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG

Drop Image Here  
- OR -  
Click to Upload

Screenshot Flag

Clear Submit ✓ 3s completed at 5:05 PM



**Timestamp: 4.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
[16] Running on public URL: <https://15052.gradio.app>

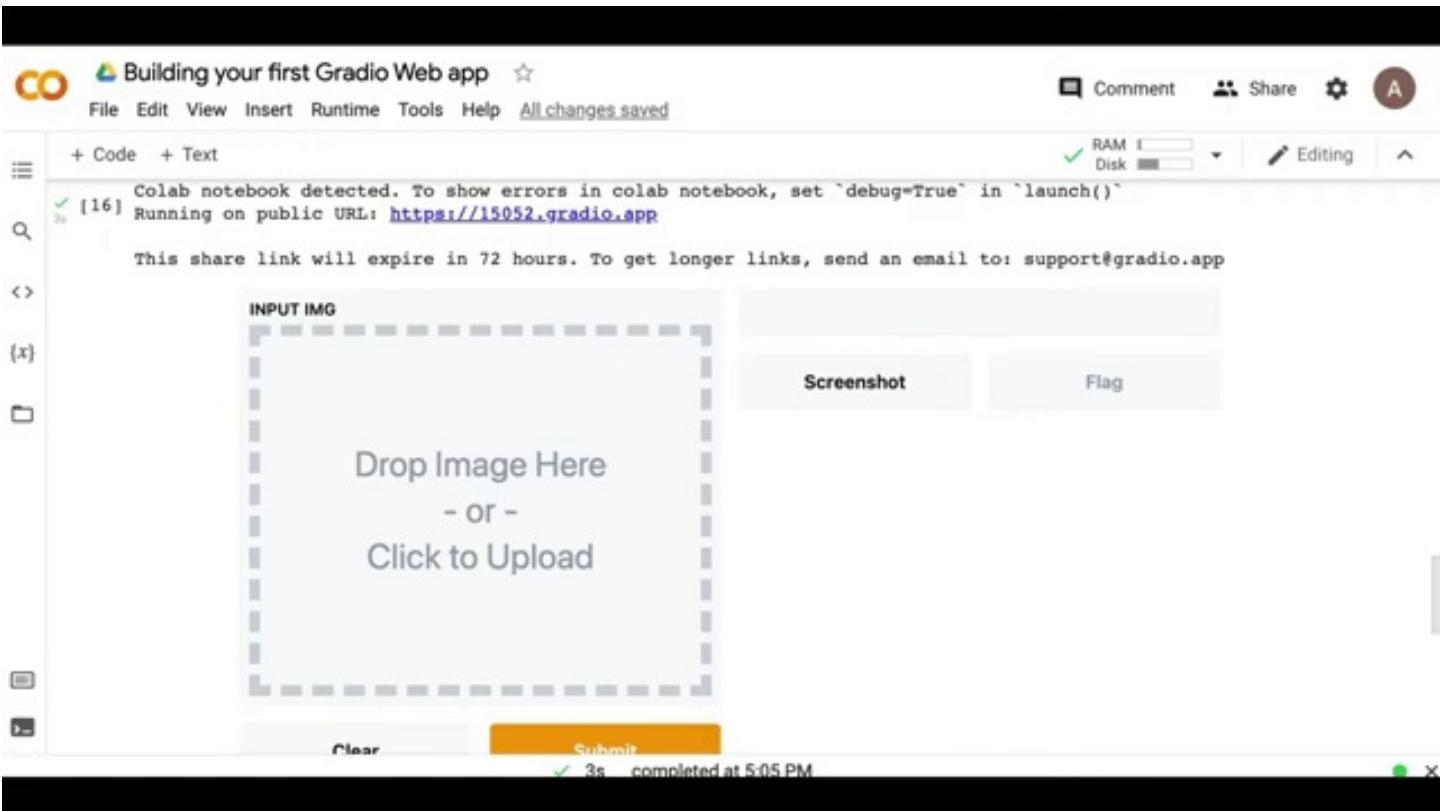
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG

Drop Image Here  
- OR -  
Click to Upload

Screenshot Flag

Clear Submit ✓ 3s completed at 5:05 PM



**Timestamp: 5.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
[16] Running on public URL: <https://15052.gradio.app>

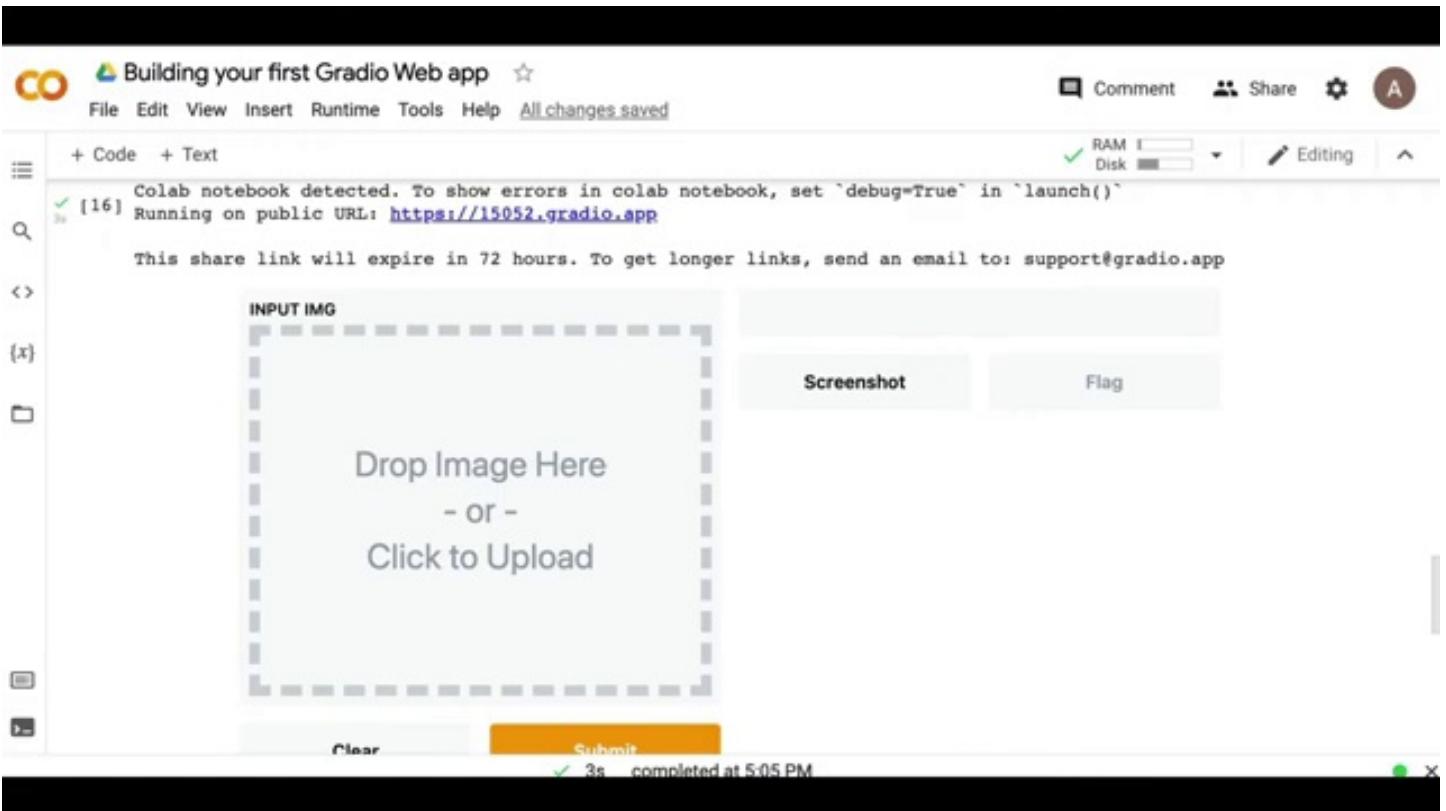
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG

Drop Image Here  
- OR -  
Click to Upload

Screenshot Flag

Clear Submit ✓ 3s completed at 5:05 PM



**Timestamp: 6.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 7.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
[16] Running on public URL: <https://15052.gradio.app>

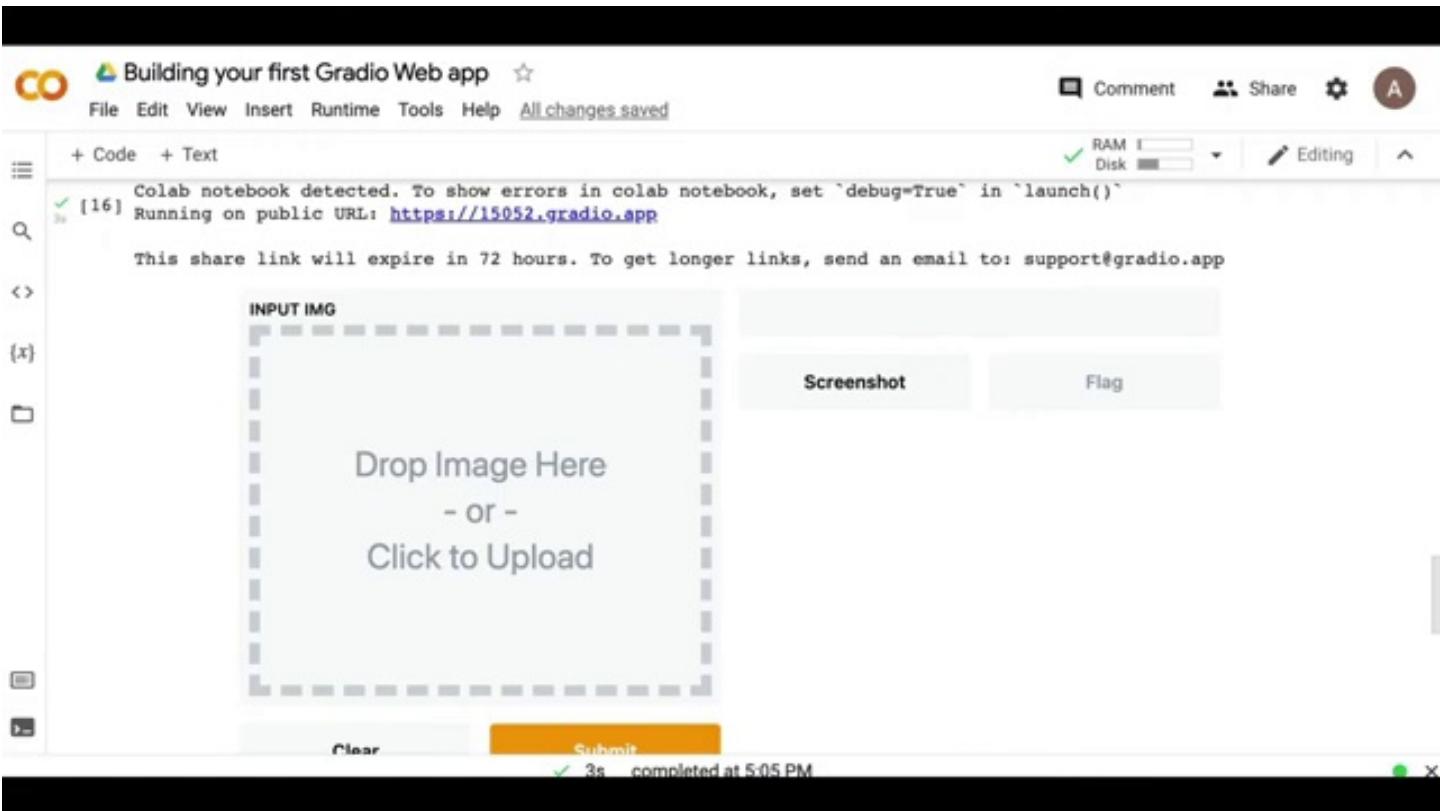
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG

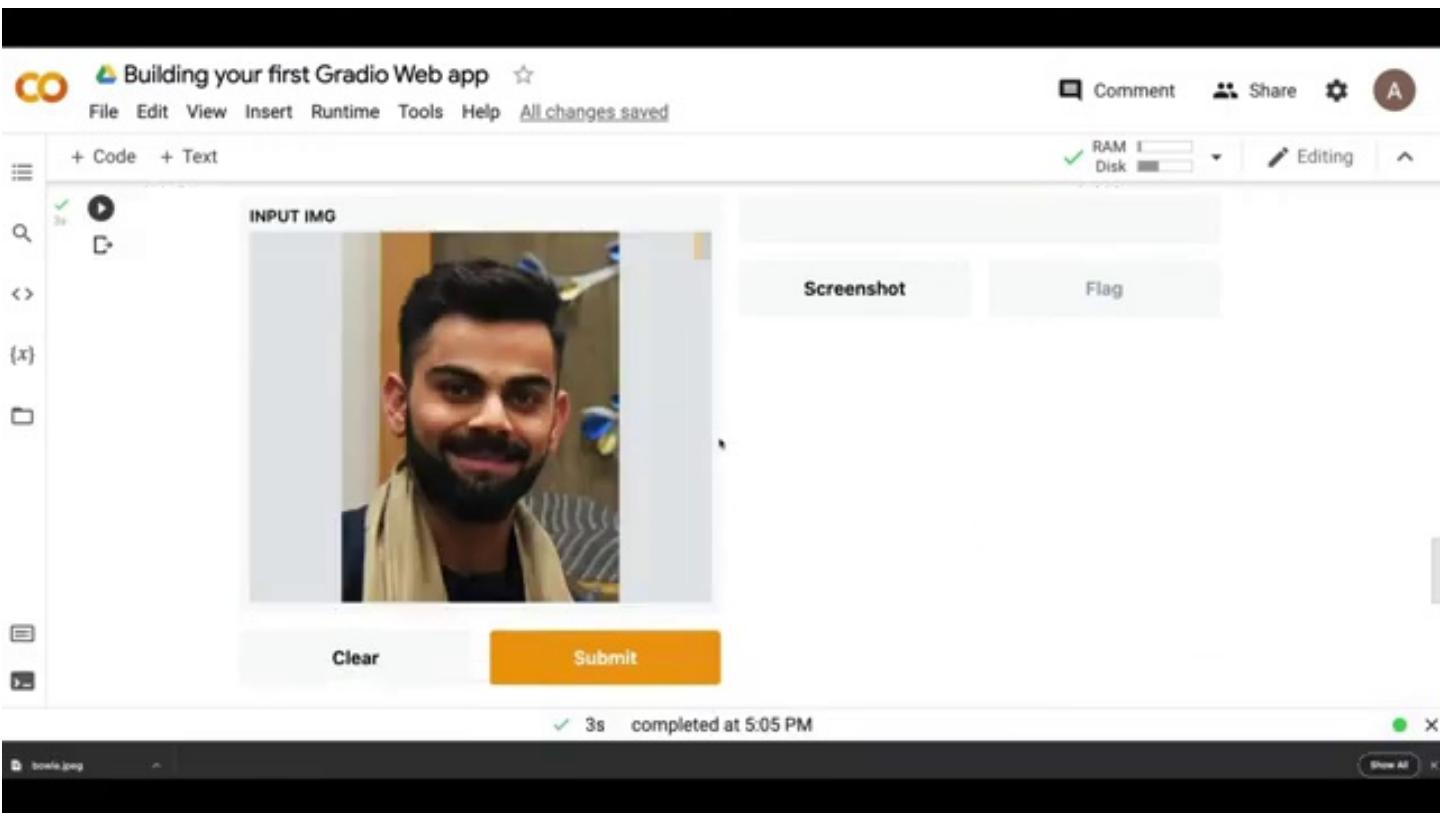
Drop Image Here  
- OR -  
Click to Upload

Screenshot Flag

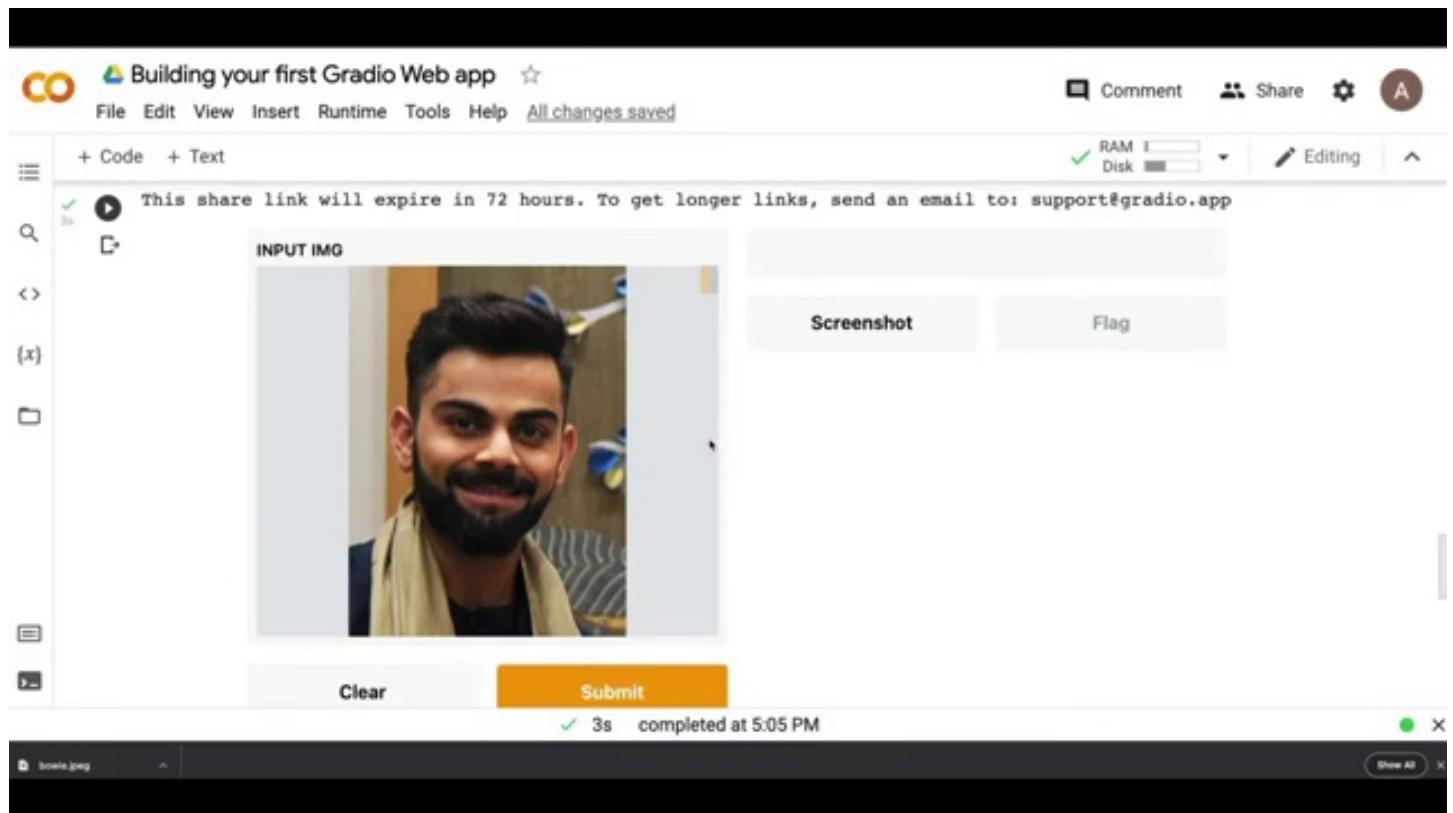
Clear Submit ✓ 3s completed at 5:05 PM



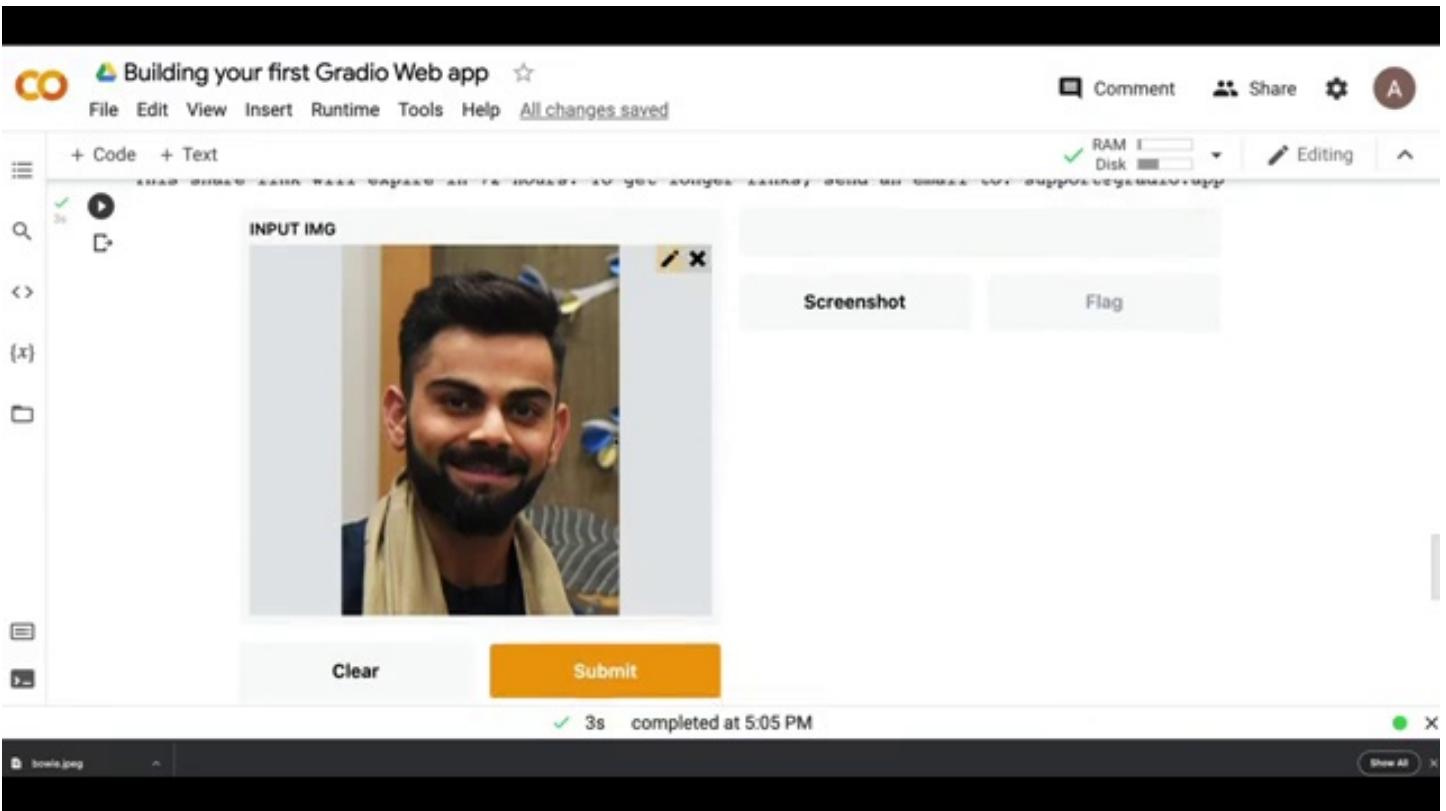
**Timestamp: 8.00 seconds**



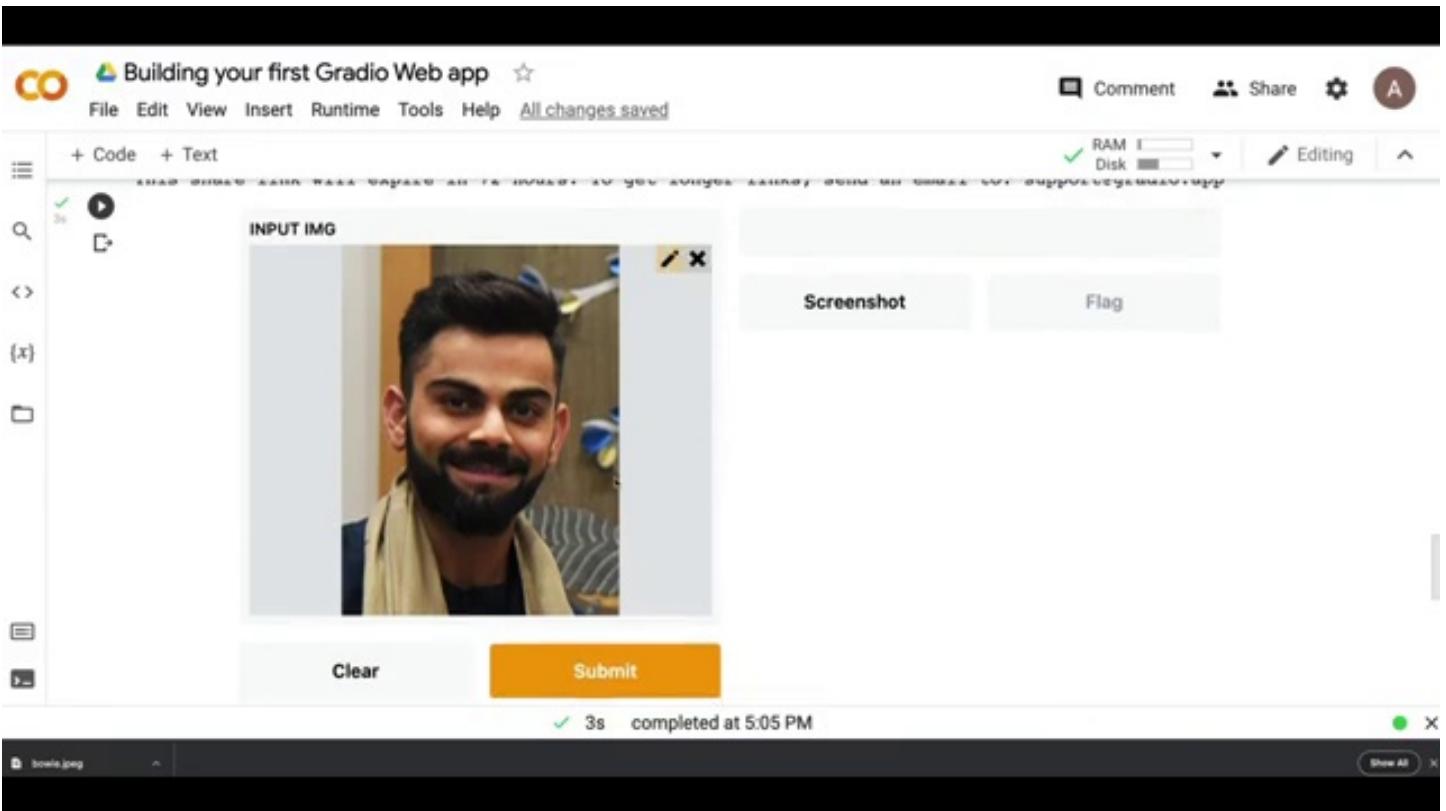
**Timestamp: 9.00 seconds**



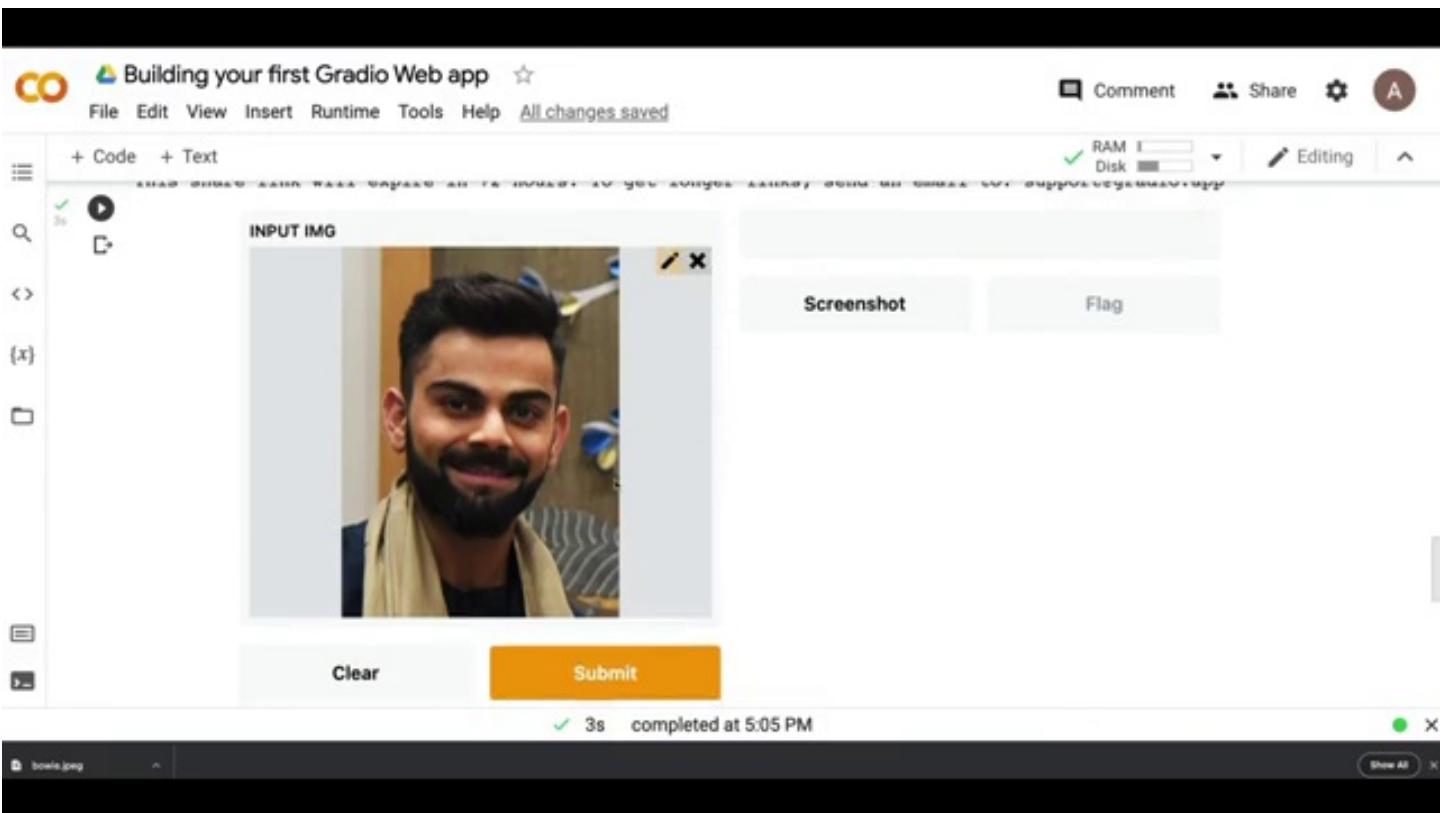
**Timestamp: 10.00 seconds**



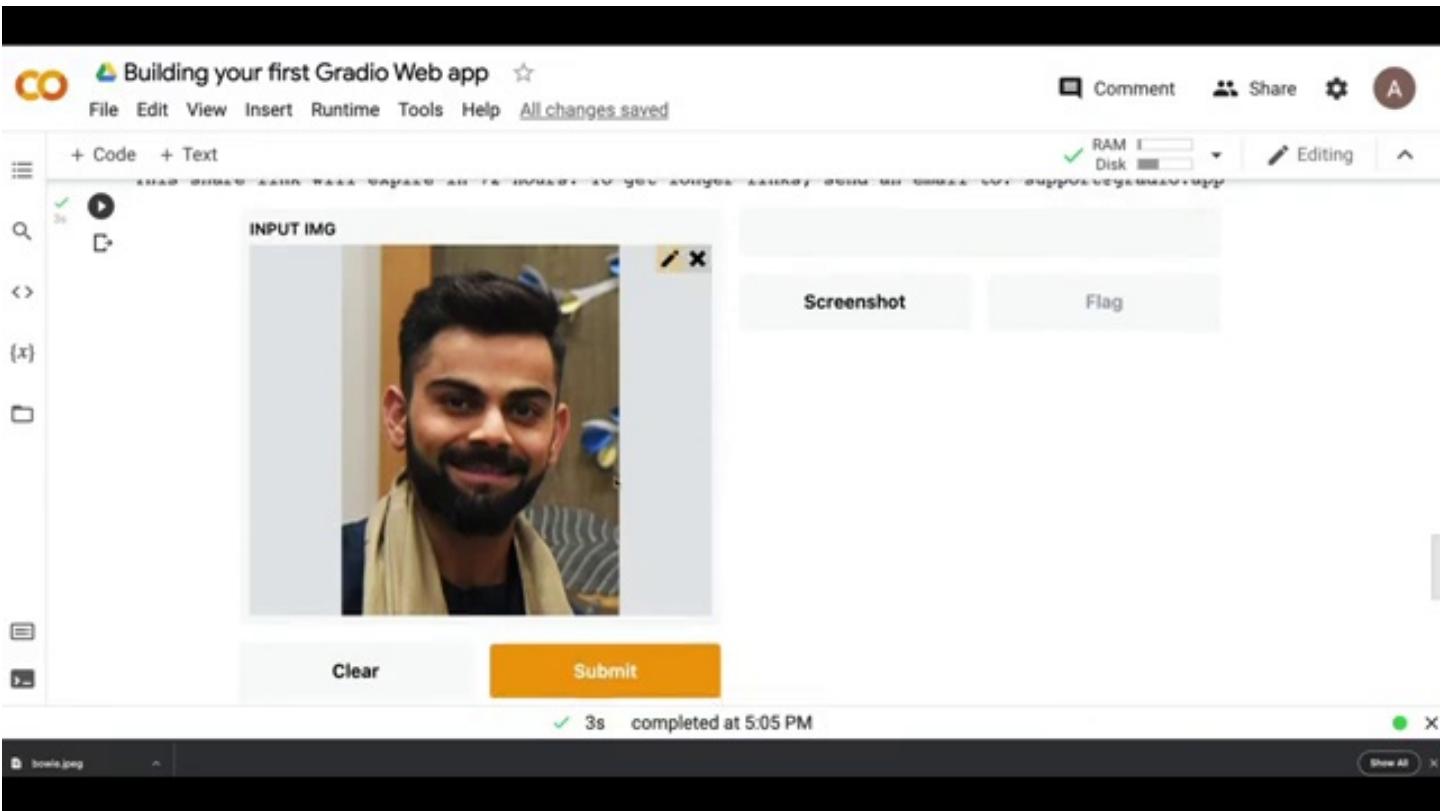
**Timestamp: 11.00 seconds**



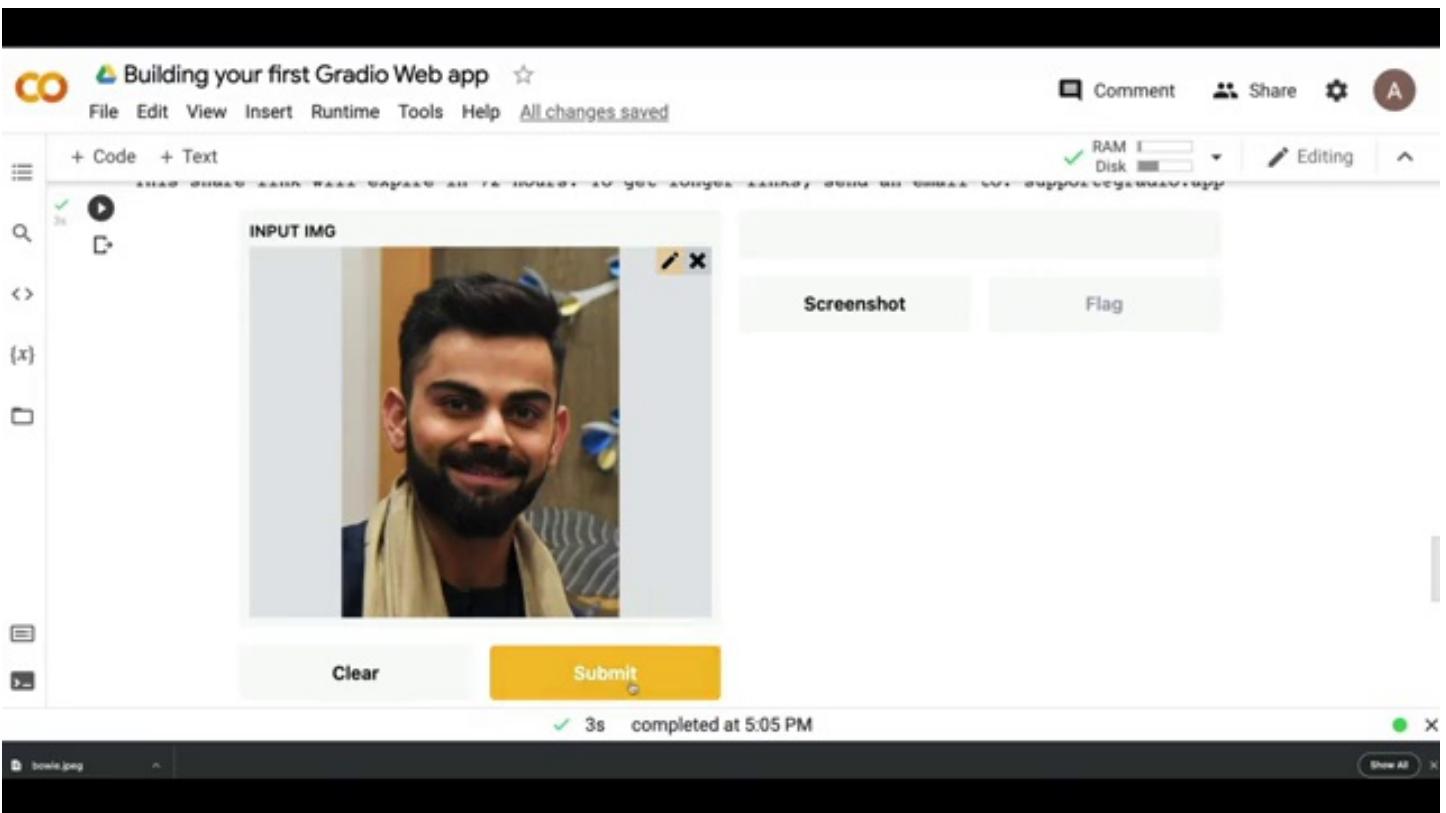
**Timestamp: 12.00 seconds**



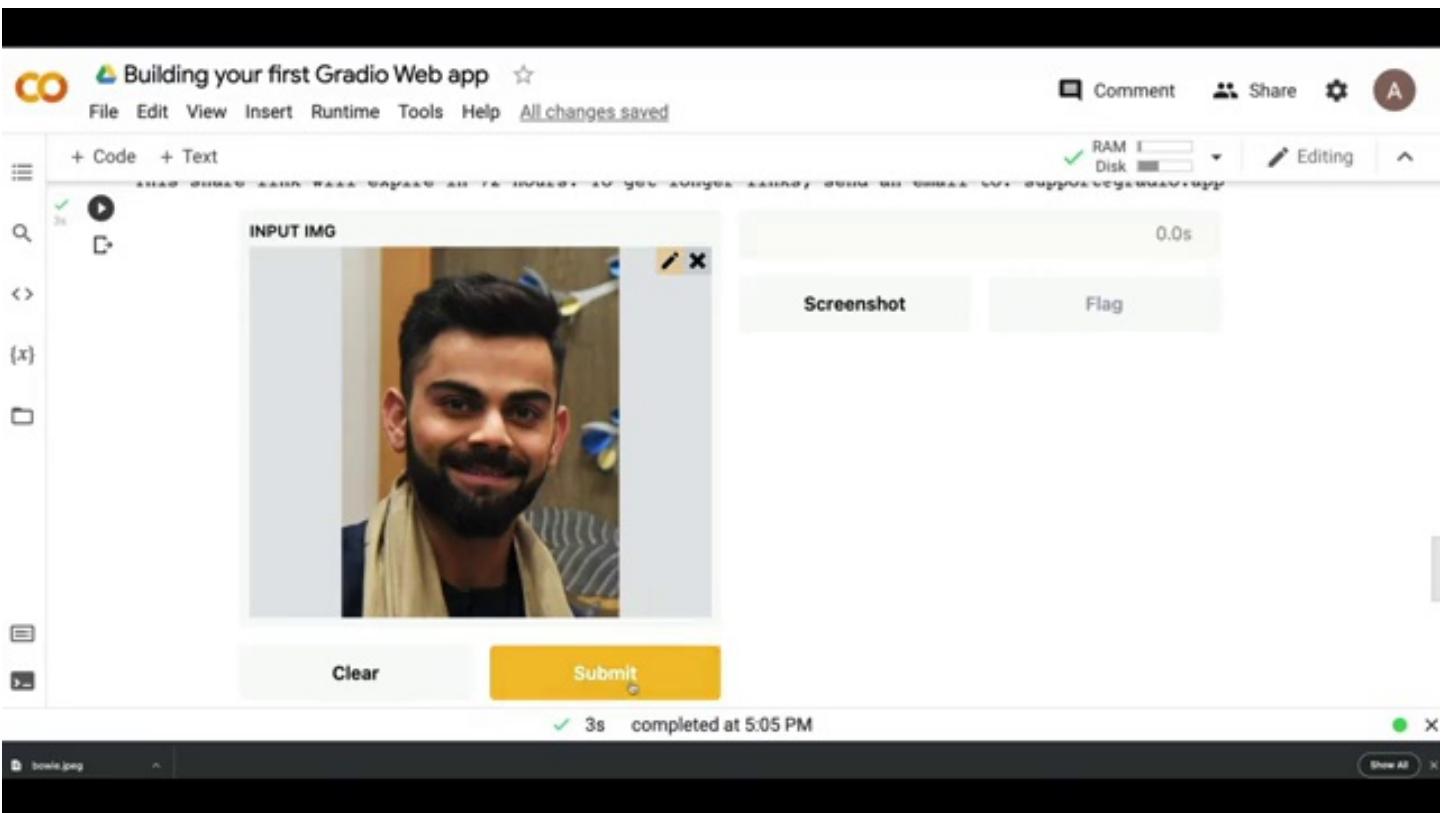
**Timestamp: 13.00 seconds**



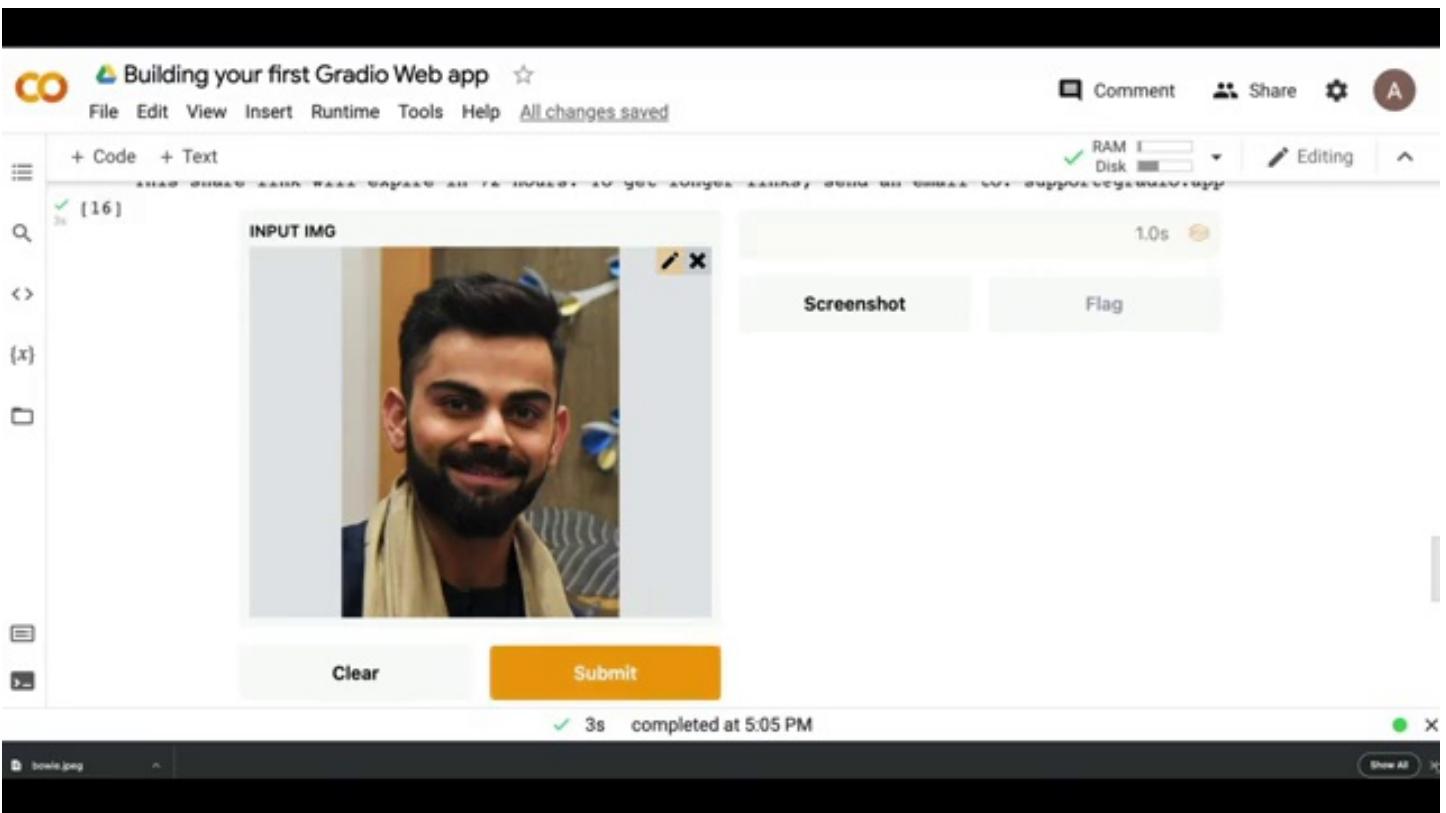
**Timestamp: 14.00 seconds**



**Timestamp: 15.00 seconds**



**Timestamp: 16.00 seconds**



**Timestamp: 17.00 seconds**

NAME

Clear

Submit

Screenshot

Flag

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

⋮

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `Inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
```

**Timestamp: 18.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG



OUTPUT

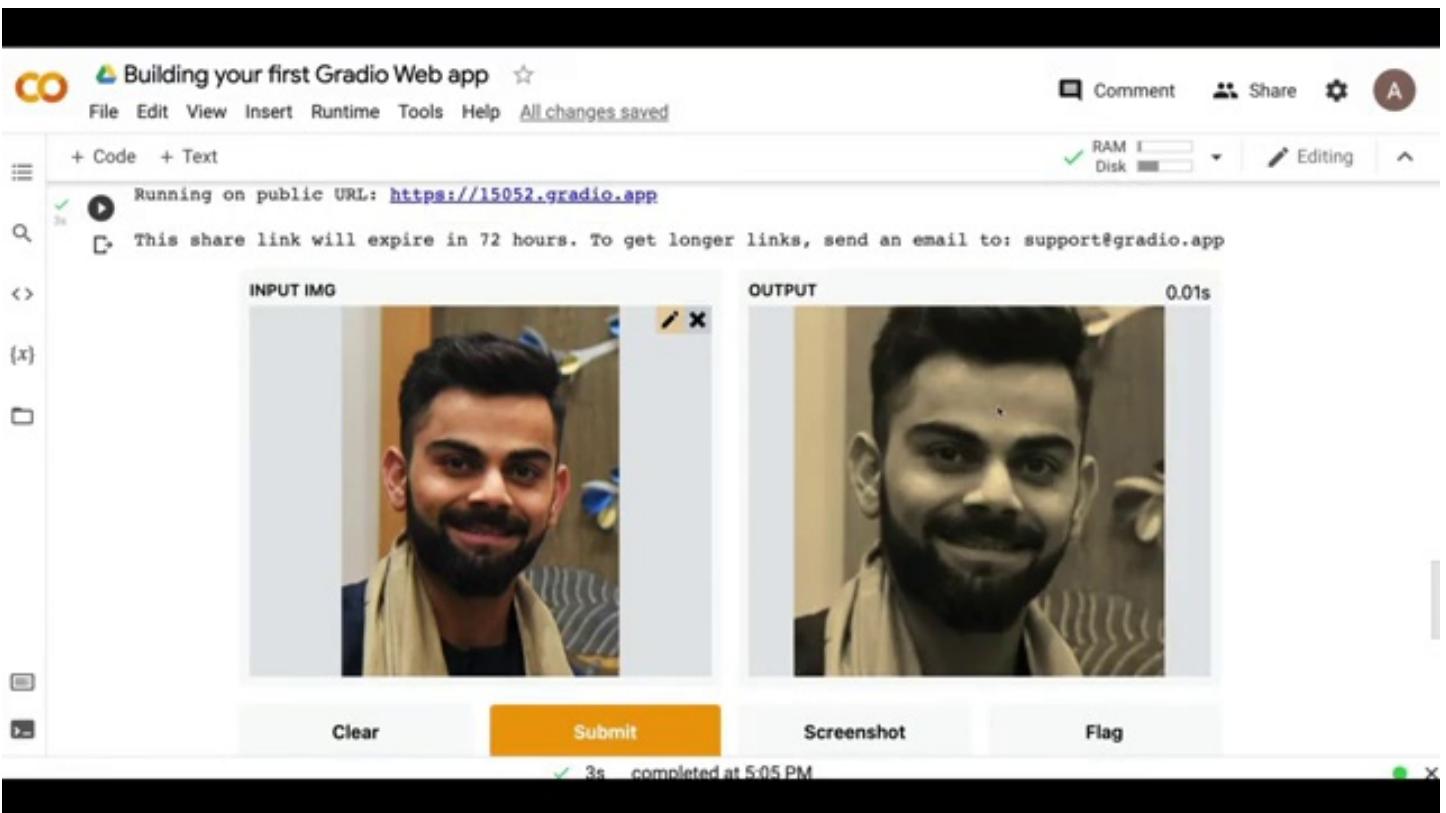


0.01s

Clear Submit Screenshot Flag

✓ 3s completed at 5:05 PM

**Timestamp: 19.00 seconds**



**Timestamp: 20.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG



OUTPUT



0.01s

Clear Submit Screenshot Flag

✓ 3s completed at 5:05 PM

**Timestamp: 21.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share  

+ Code + Text

RAM Disk  Editing 

3s  

Clear  Submit Screenshot Flag

view the api  | built with 

(<Flask 'gradio.networking'>,  
<http://127.0.0.1:7863/>,  
✓ 3s completed at 5:05 PM 

**Timestamp: 22.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 23.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

```
✓ import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

D Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 24.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 25.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

3s completed at 5:05 PM

**Timestamp: 26.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), ["image"])
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 27.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 28.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The image shows a simple web-based user interface. At the top, there is a text input field with the placeholder 'NAME' and the value 'I' entered. Below the input field are three buttons: 'Clear', 'Submit' (which is highlighted with a yellow background), and 'Screenshot'. To the right of the 'Submit' button are two small links: 'Screenshot' and 'Flag'.

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

**Timestamp: 29.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 30.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 31.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 32.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

3s completed at 5:05 PM

**Timestamp: 33.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(f=sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 34.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(f=sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 35.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 36.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 37.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
gr.Interface(fn, inputs=None, outputs=None,
verbose=False, examples=None, examples_per_page=10,
live=False, layout='unaligned', show_input=True,
show_output=True, capture_session=False,
interpretation=None, num_shap=2.0, theme=None,
repeat_outputs_per_model=True, title=None,
description=None, article=None, thumbnail=None,
css=None, server_port=None,
server_name='127.0.0.1', height=500, width=900,
allow_screenshot=True, allow_flagging=None,
flagging_options=None, encrypt=False,
show_tips=False, flagging_dir='flagged',
iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

RAM Disk Editing

**Timestamp: 38.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
gr.Interface(fn, inputs=None, outputs=None,
verbose=False, examples=None, examples_per_page=10,
live=False, layout='unaligned', show_input=True,
show_output=True, capture_session=False,
interpretation=None, num_shap=2.0, theme=None,
repeat_outputs_per_model=True, title=None,
description=None, article=None, thumbnail=None,
css=None, server_port=None,
server_name='127.0.0.1', height=500, width=900,
allow_screenshot=True, allow_flagging=None,
flagging_options=None, encrypt=False,
show_tips=False, flagging_dir='flagged',
iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

RAM Disk Editing

**Timestamp: 39.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The image shows a simple Gradio interface. At the top, there's a text input field with the placeholder 'NAME' and the value 'I' entered. Below the input field are three buttons: 'Clear', 'Submit' (which is highlighted with a yellow background), and 'Screenshot'. To the right of these buttons are two more buttons: 'Screenshot' and 'Flag'.

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

**Timestamp: 40.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
gr.Interface(fn, inputs=None, outputs=None,
verbose=False, examples=None, examples_per_page=10,
live=False, layout='unaligned', show_input=True,
show_output=True, capture_session=False,
interpretation=None, num_shap=2.0, theme=None,
repeat_outputs_per_model=True, title=None,
description=None, article=None, thumbnail=None,
css=None, server_port=None,
server_name='127.0.0.1', height=500, width=900,
allow_screenshot=True, allow_flagging=None,
flagging_options=None, encrypt=False,
show_tips=False, flagging_dir='flagged',
iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

RAM Disk Editing

**Timestamp: 41.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 42.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 43.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 44.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 45.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 46.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 47.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 48.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 49.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 50.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.



The image shows a screenshot of a Gradio interface. At the top, there is a text input field with the placeholder "NAME" and the letter "I" typed into it. Below the input field are two buttons: "Clear" on the left and a large orange "Submit" button on the right. To the right of the input field, there are two smaller buttons: "Screenshot" and "Flag".

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

**Timestamp: 51.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 52.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[def __init__(shape=None, image_mode='RGB', invert_colors=False,
           source='upload', tool='editor', type='numpy', label=None, optional=False)
        [
            Open in tab View source
        Component creates an image upload box with editing capabilities.
        Input type: Union[numpy.array, PIL.Image, file-object]
        Demos: image_classifier.py, image_mod.py, webcam.py, digit_classifier.py
    sephia_img = input_img.dot(sephia_filter)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG OUTPUT 0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 53.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 54.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 55.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 56.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 57.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG      OUTPUT      0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 58.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

https://44140.gradio.app')

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

**Timestamp: 59.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

INPUT IMG



X

OUTPUT



0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 60.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,  
'http://127.0.0.1:7862/',  
'https://44140.gradio.app'>
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sephia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sephia\_img = input\_img.dot(sephia\_filter.T)  
 sephia\_img /= sephia\_img.max()  
 return sephia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`

✓ 3s completed at 5:05 PM

**Timestamp: 61.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or open in a browser on <http://localhost:7860> if running from a script.

NAME  
Type your name here

Clear Submit Screenshot Flag Share

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

**Timestamp: 62.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,  
'http://127.0.0.1:7862/',  
'https://44140.gradio.app'>
```

import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`

3s completed at 5:05 PM

**Timestamp: 63.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

(x)

def sepia(input\_img):  
 sephia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sephia\_img = input\_img.dot(sephia\_filter.T)  
 sephia\_img /= sephia\_img.max()  
 return sephia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

**Timestamp: 64.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[15] (<Flask 'gradio.networking'>,
      'http://127.0.0.1:7862/',
      'https://44140.gradio.app')

import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
3s completed at 5:05 PM
```

**Timestamp: 65.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

(x)

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

**Timestamp: 66.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[15] (<Flask 'gradio.networking'>,
      'http://127.0.0.1:7862/',
      'https://44140.gradio.app')

import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
3s completed at 5:05 PM
```

**Timestamp: 67.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

(x)

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

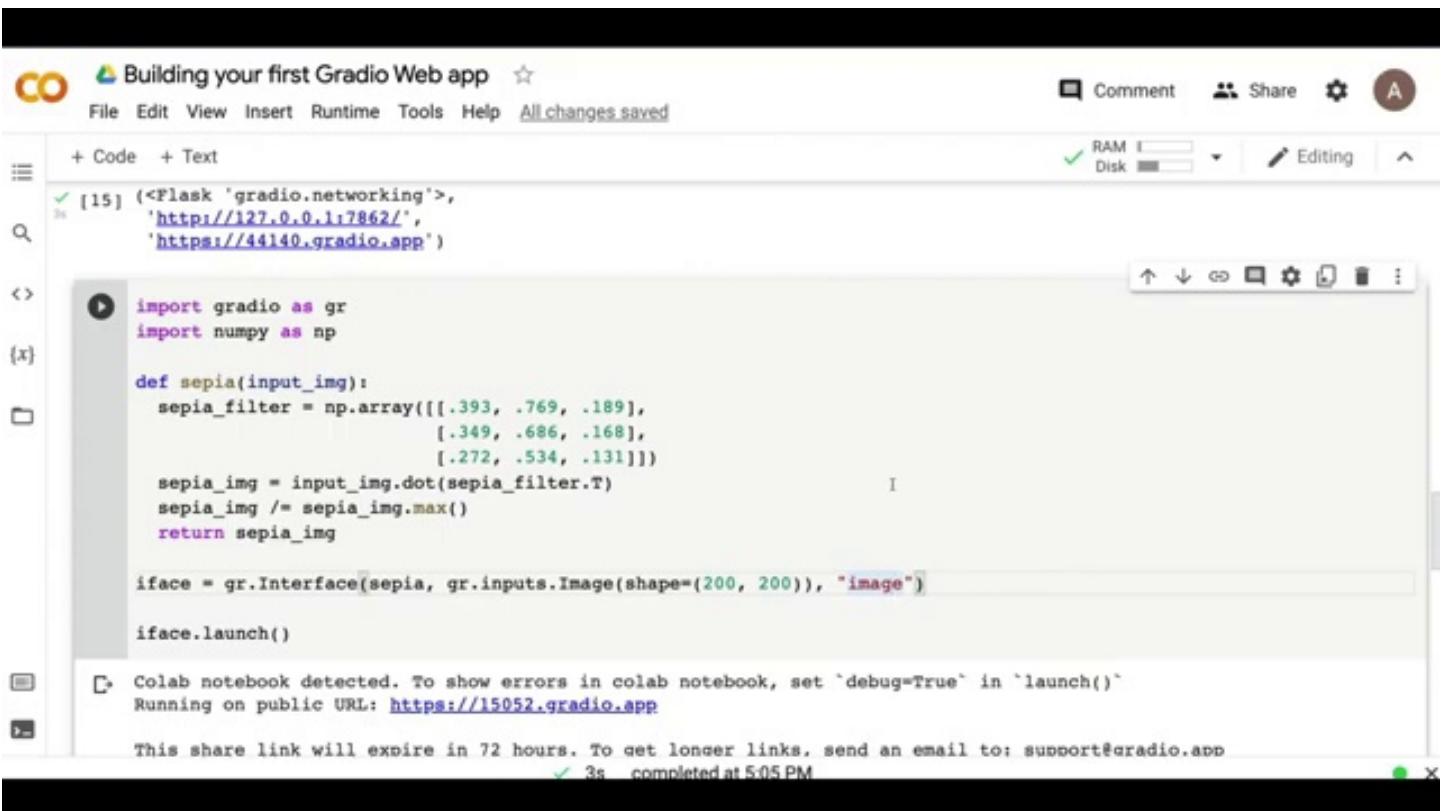
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

Comment Share A



**Timestamp: 68.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

{x}

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

Comment Share A



**Timestamp: 69.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

{x}

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

Comment Share A

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Code' and 'Text', 'Comment', 'Share', 'Settings', and a user profile icon. On the left, there's a sidebar with icons for search, code, and file operations. The main area contains Python code for creating a Gradio interface. The code imports 'gradio' and 'numpy', defines a 'sepia' function using a sepia filter matrix, creates a 'iface' object, and finally calls 'iface.launch()'. A note indicates that the Colab notebook detected, so errors can be shown in the Colab notebook if 'debug=True' is set in 'launch()'. It also provides a public URL: <https://15052.gradio.app>. A message says the share link will expire in 72 hours. At the bottom, it shows a completion message: '✓ 3s completed at 5:05 PM'.

**Timestamp: 70.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

{x}

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

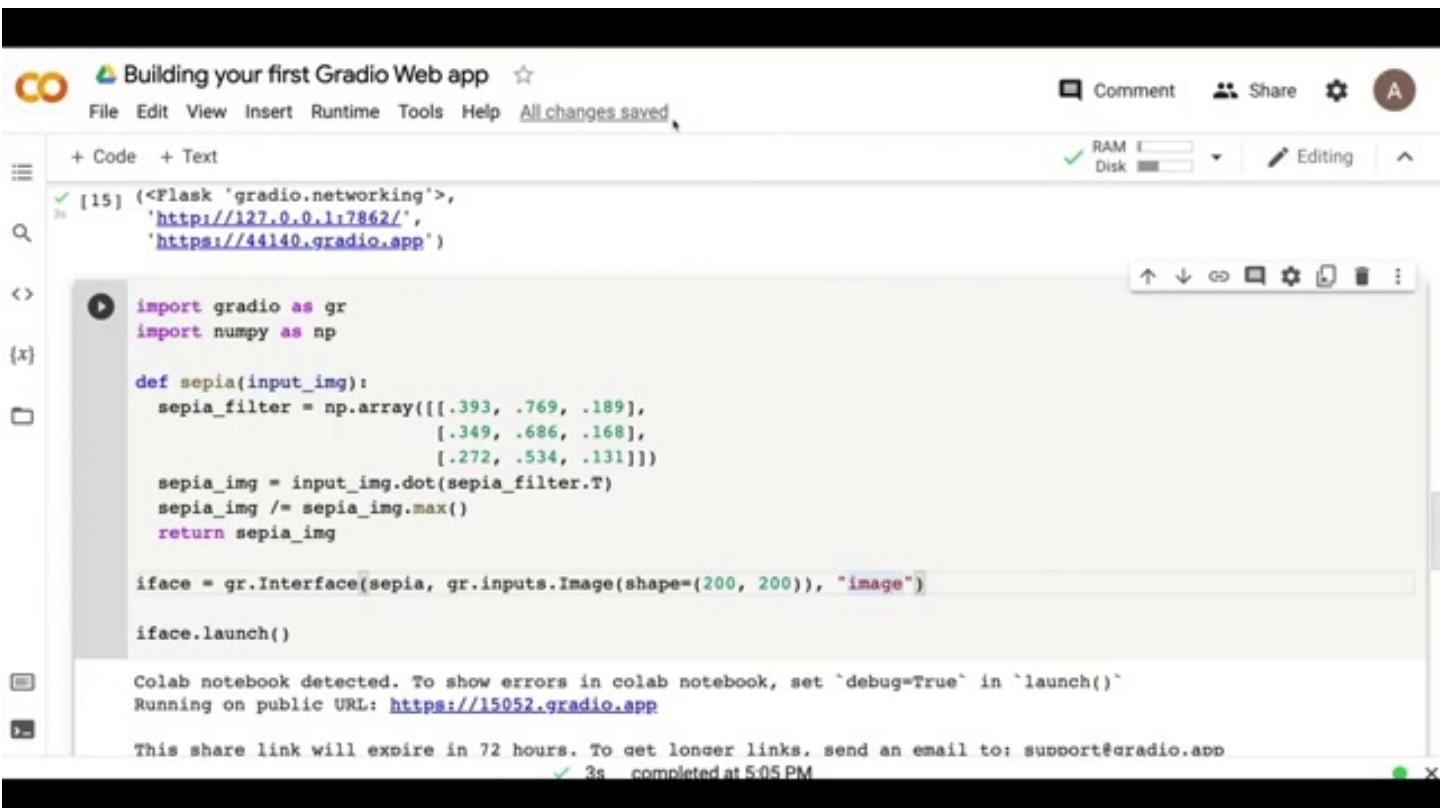
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

Comment Share A



**Timestamp: 71.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] (<Flask 'gradio.networking'>,  
+ http://127.0.0.1:7862/  
+ https://44140.gradio.app')

import gradio as gr  
import numpy as np

{x}

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:05 PM

RAM Disk Editing

**Timestamp: 72.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 73.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME  
Type your name here

Clear Submit Screenshot Flag Share

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core Interface class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

**Timestamp: 74.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG 

OUTPUT  0.01s

✓ 3s completed at 5:05 PM

**Timestamp: 75.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
iiface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iiface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

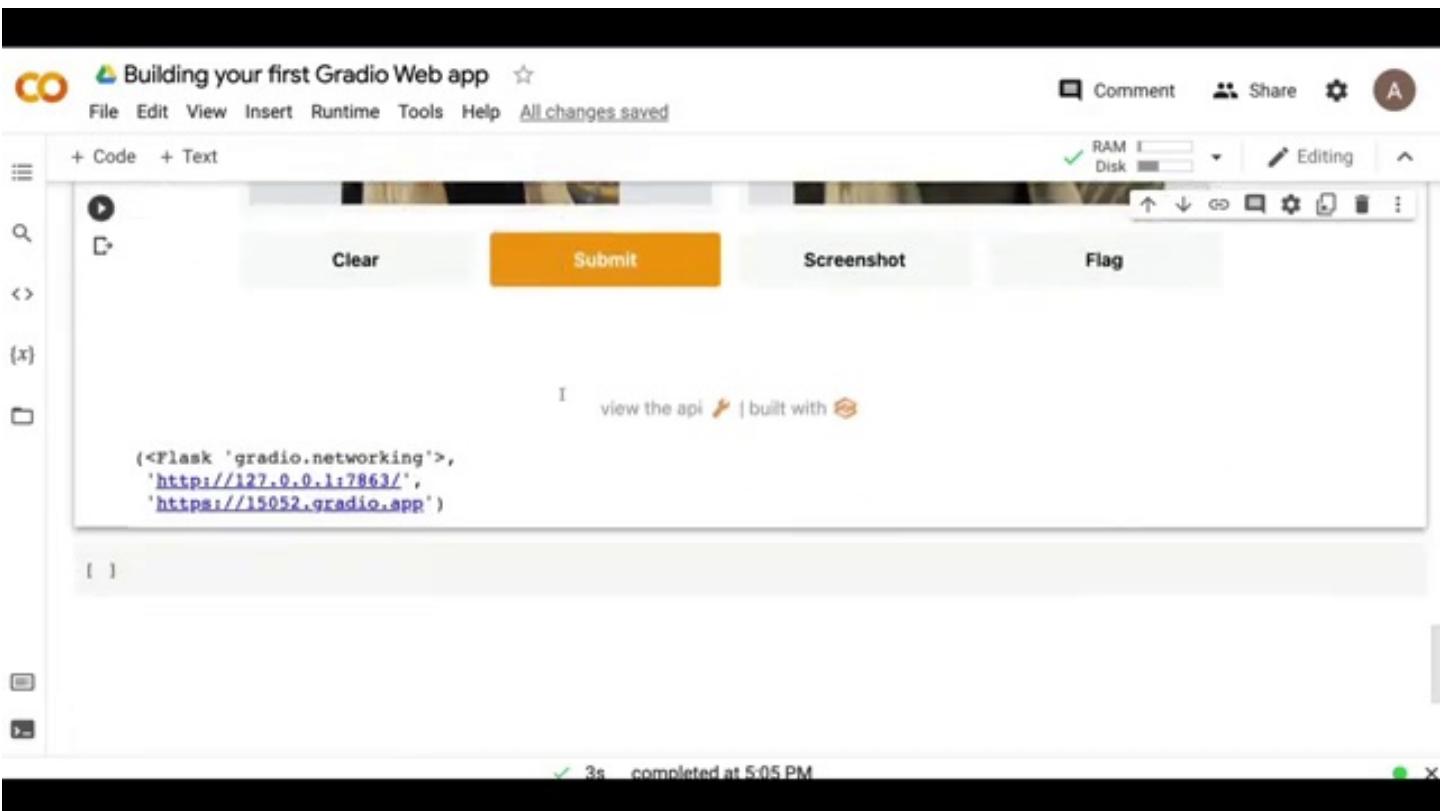
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

INPUT IMG OUTPUT 0.01s



✓ 3s completed at 5:05 PM

**Timestamp: 76.00 seconds**



**Timestamp: 77.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

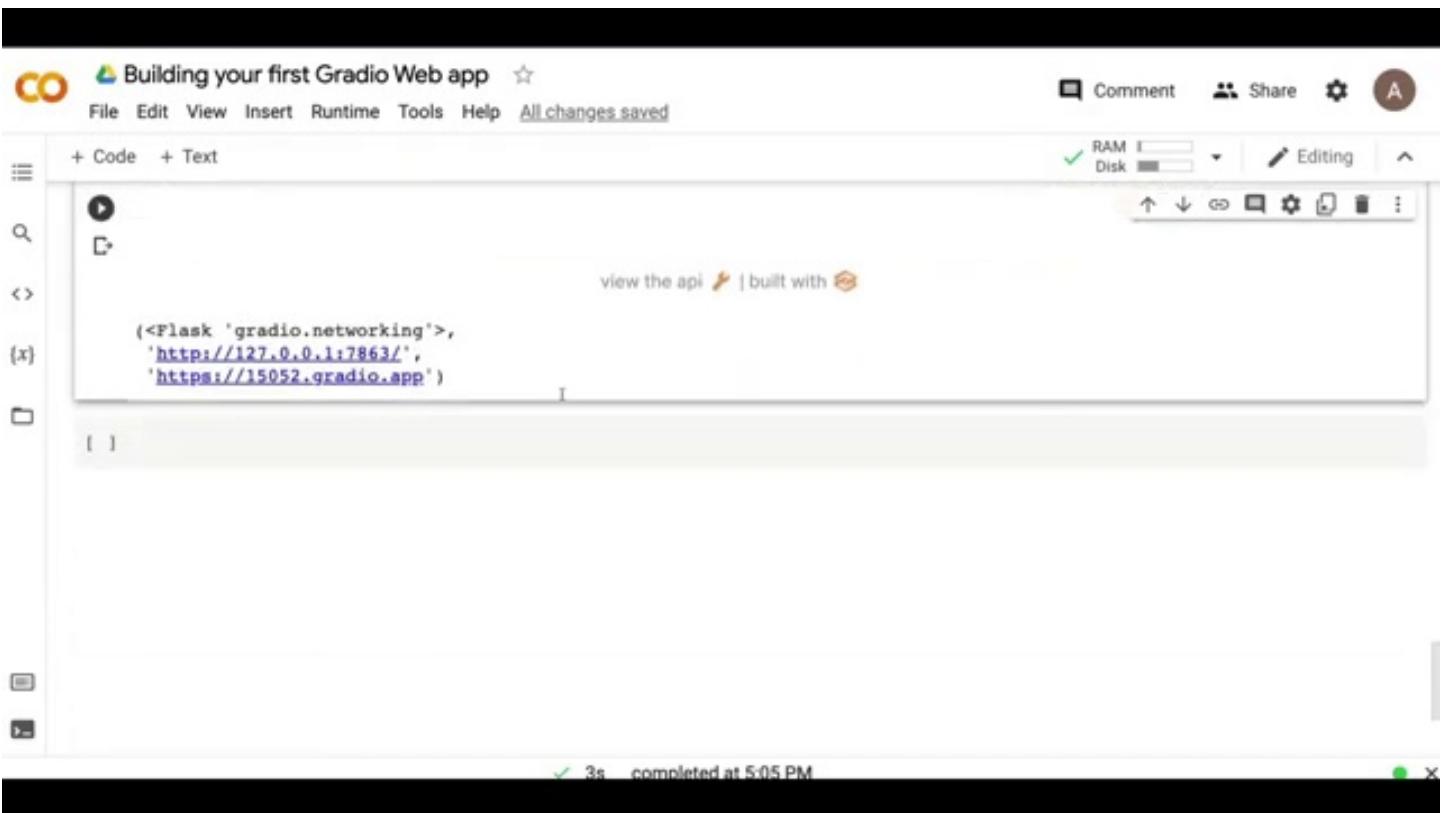
RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 78.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, disk, editing, and other tools. The main area contains a code editor with Python code related to Gradio networking, a preview section with a play button and a link to view the API, and a log section at the bottom indicating a successful completion at 5:05 PM.

**Timestamp: 79.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 80.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, Disk, and editing. The main area contains a code editor with Python code related to Gradio networking, a preview section with a play button, and a log section at the bottom indicating a successful completion of a task. On the left, there are various sidebar icons.

**Timestamp: 81.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, Disk, and editing. The main area contains a code editor with Python code related to Gradio networking, a preview section with a play button and a link to the API, and a log section at the bottom indicating a successful completion at 5:05 PM.

**Timestamp: 82.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7863/',  
     'https://15052.gradio.app')
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface, likely a Jupyter Notebook or similar, running on a Gradio application. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the right, there are 'Comment', 'Share', and a user profile icon. Below the menu is a toolbar with icons for code (+ Code), text (+ Text), RAM/Disk status, and editing tools. The main workspace displays a terminal-like interface with a command prompt. The output shows the creation of a Flask application named 'gradio.networking' with two endpoints: 'http://127.0.0.1:7863/' and 'https://15052.gradio.app'. Below the code, there's a text input field containing '[ ]'. At the bottom, a progress bar indicates a task completed in 3 seconds at 5:05 PM.

**Timestamp: 83.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

A screenshot of a web-based development interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user icon. Below the navigation is a toolbar with icons for code, text, RAM, Disk, and editing. The main area has tabs for 'Code' and 'Text'. In the 'Code' tab, there's a play button icon, a copy icon, and a link to 'view the api'. Below this is some Python code: 'from gradio import networking', 'app = networking.FlaskApp()', 'app.run()', and 'app.serve()'. There's also a link to 'built with'. The 'Text' tab is currently empty. On the left, there are various icons for file operations like search, refresh, and save. At the bottom, a log bar shows a green checkmark, '3s', and the completion time 'completed at 5:05 PM'.

**Timestamp: 84.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The screenshot shows a simple web interface created by Gradio. It features a single input field with the placeholder text 'NAME'. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller links: 'Screenshot' and 'Flag'. At the bottom left of the interface is a 'Clear' button.

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field – for example, we wanted it to be larger and have a text hint? If we use the `actual Input class` for `Textbox` instead of using the string:

**Timestamp: 85.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

A screenshot of a web-based development interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user icon. Below the navigation is a toolbar with icons for code, text, RAM, Disk, and editing. The main area has tabs for 'Code' and 'Text'. The 'Code' tab is active, showing a single line of Python code: 'from gradio import \*'. Below the code is a message: 'view the api | built with'. Underneath that is a block of code: '<Flask "gradio.networking">, http://127.0.0.1:7863/, https://15052.gradio.app''. On the left side, there are several small icons: a play button, a search icon, a file comparison icon, a folder icon, a refresh icon, and a help icon. At the bottom, there's a log entry: '✓ 3s completed at 5:05 PM'.

**Timestamp: 86.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, disk, editing, and other tools. The main area has tabs for 'Code' and 'Text'. The 'Code' tab is active, displaying Python code related to a Flask application for networking. The code includes URLs for both HTTP and HTTPS. Below the code editor is a terminal or logs area with a single line: '✓ 3s completed at 5:05 PM'. On the left side, there are several small icons representing different functions or panels.

**Timestamp: 87.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, disk, editing, and other tools. The main area has tabs for 'Code' and 'Text'. The 'Code' tab is active, displaying Python code for a Flask application. The code defines a single route with two URLs: 'http://127.0.0.1:7863/' and 'https://15052.gradio.app'. Below the code, there's a message 'view the api | built with'. The 'Text' tab is also visible. On the left, there are several small icons for file operations like copy, paste, and search. At the bottom, a progress bar indicates a task completed in 3 seconds at 5:05 PM.

**Timestamp: 88.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, disk, editing, and other tools. The main area has tabs for 'Code' and 'Text'. The 'Code' tab is active, displaying Python code related to a Flask application. The code includes imports and two URLs: 'http://127.0.0.1:7863/' and 'https://15052.gradio.app'. Below the code editor is a terminal-like area with a single line of text: '✓ 3s completed at 5:05 PM'. On the left side, there are several small, unlabeled icons.

**Timestamp: 89.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for code, text, RAM, disk, editing, and other tools. The main area contains a code editor with Python code related to Gradio networking, a preview section with a play button, and a status message indicating completion. On the left, there are various sidebar icons for file operations like search, copy/paste, and refresh.

**Timestamp: 90.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM

This screenshot shows a web-based development interface for a Gradio application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the far right of the top bar are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the top bar is a toolbar with icons for file operations like 'New', 'Open', 'Save', and 'Run'. To the left of the main workspace are several small icons: a play button, a magnifying glass for search, a double-headed arrow for comparison, a folder, and a refresh symbol. The main workspace contains a code editor with Python-like code related to Gradio networking, a preview area with a play button, and a status bar at the bottom indicating a successful build ('3s completed at 5:05 PM').

**Timestamp: 91.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

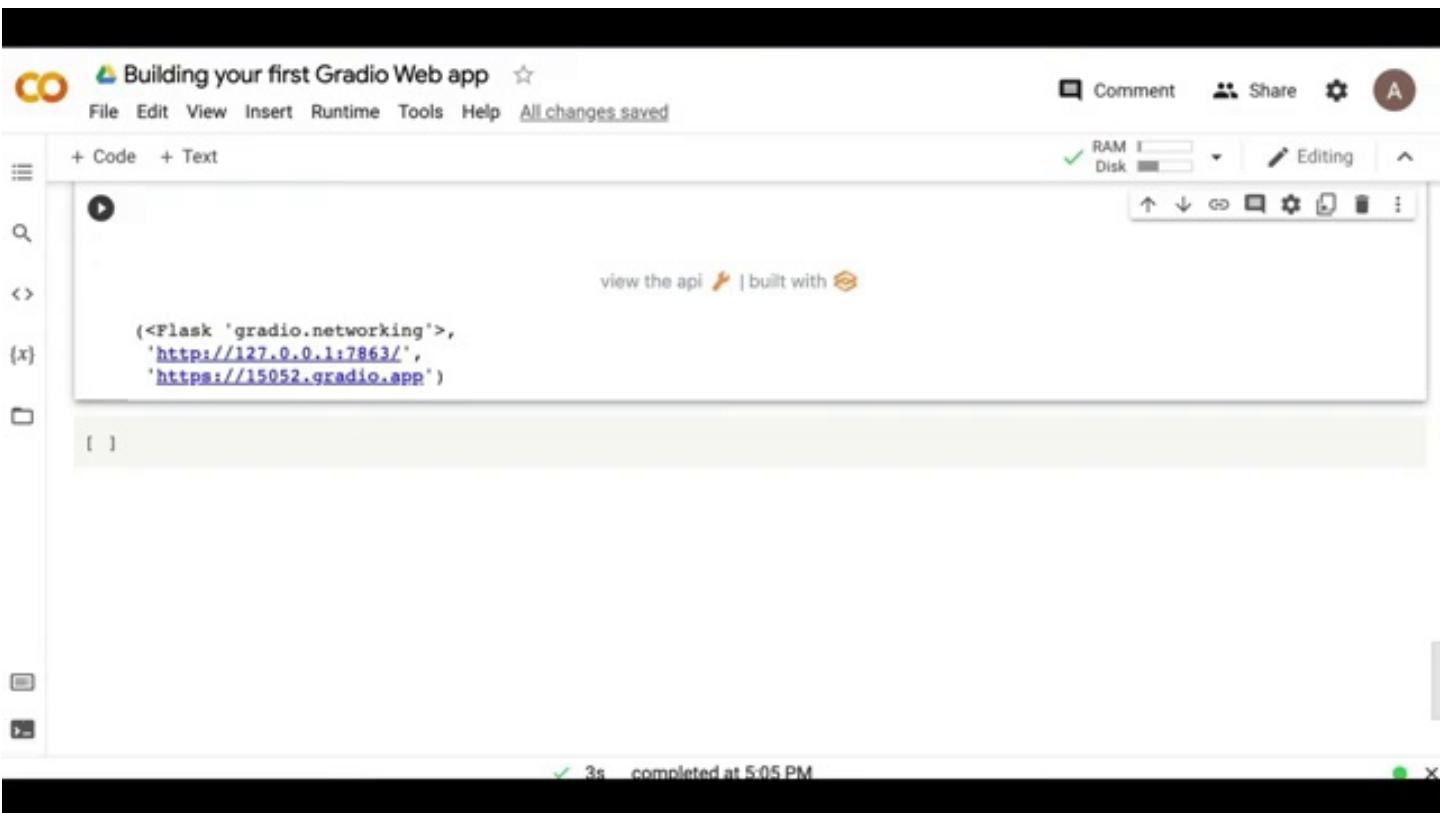
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 92.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

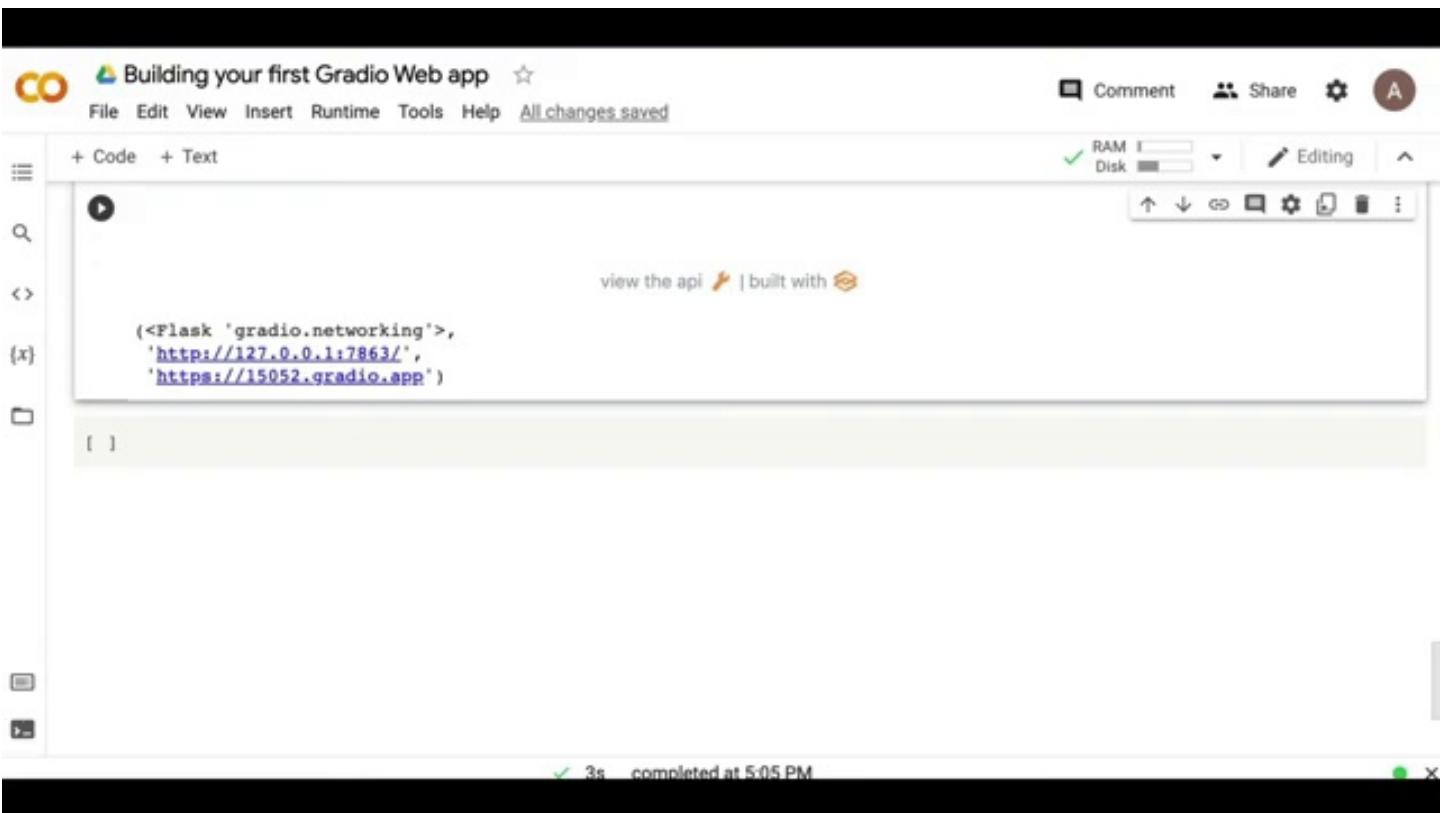
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 93.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

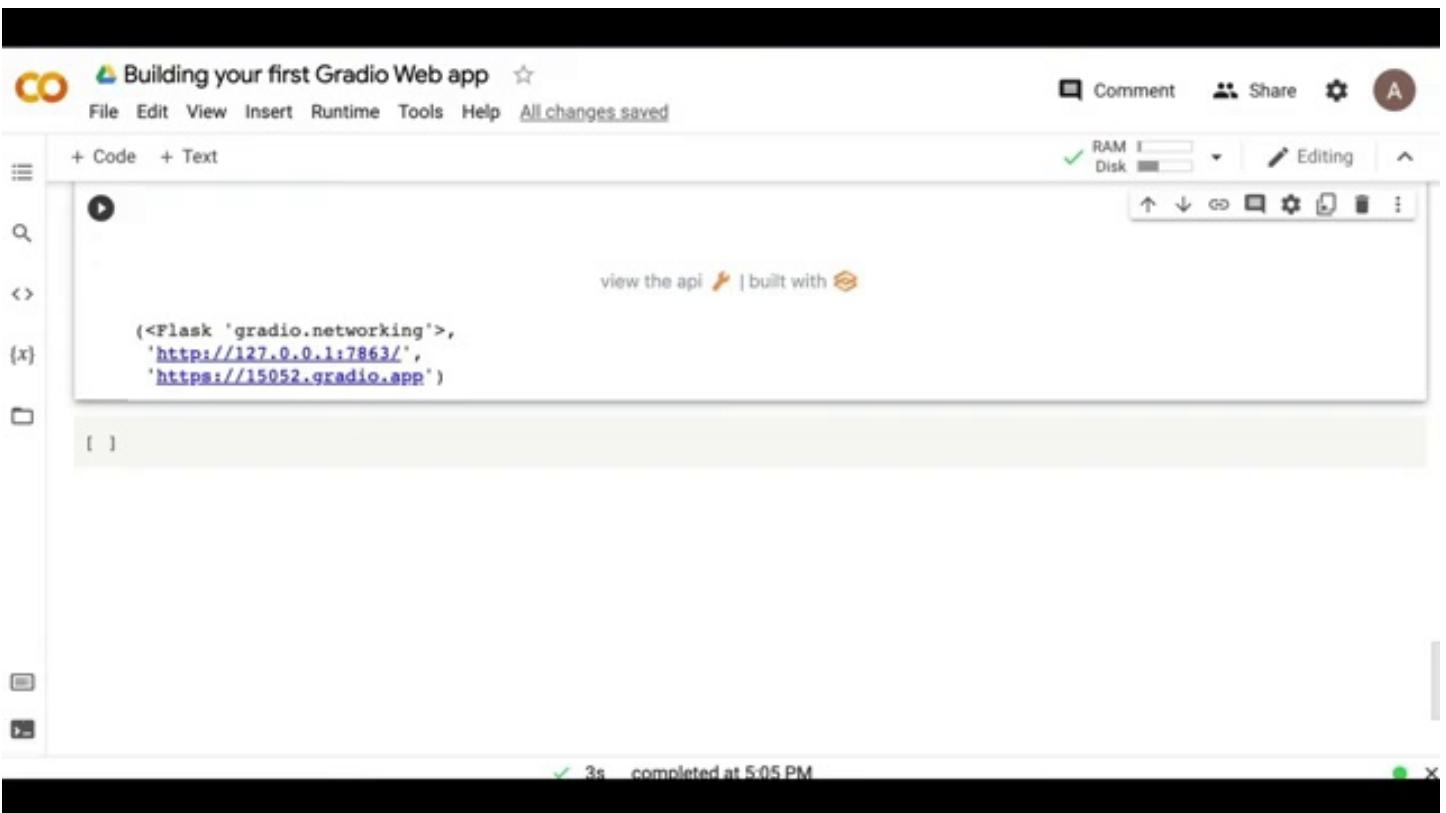
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 94.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

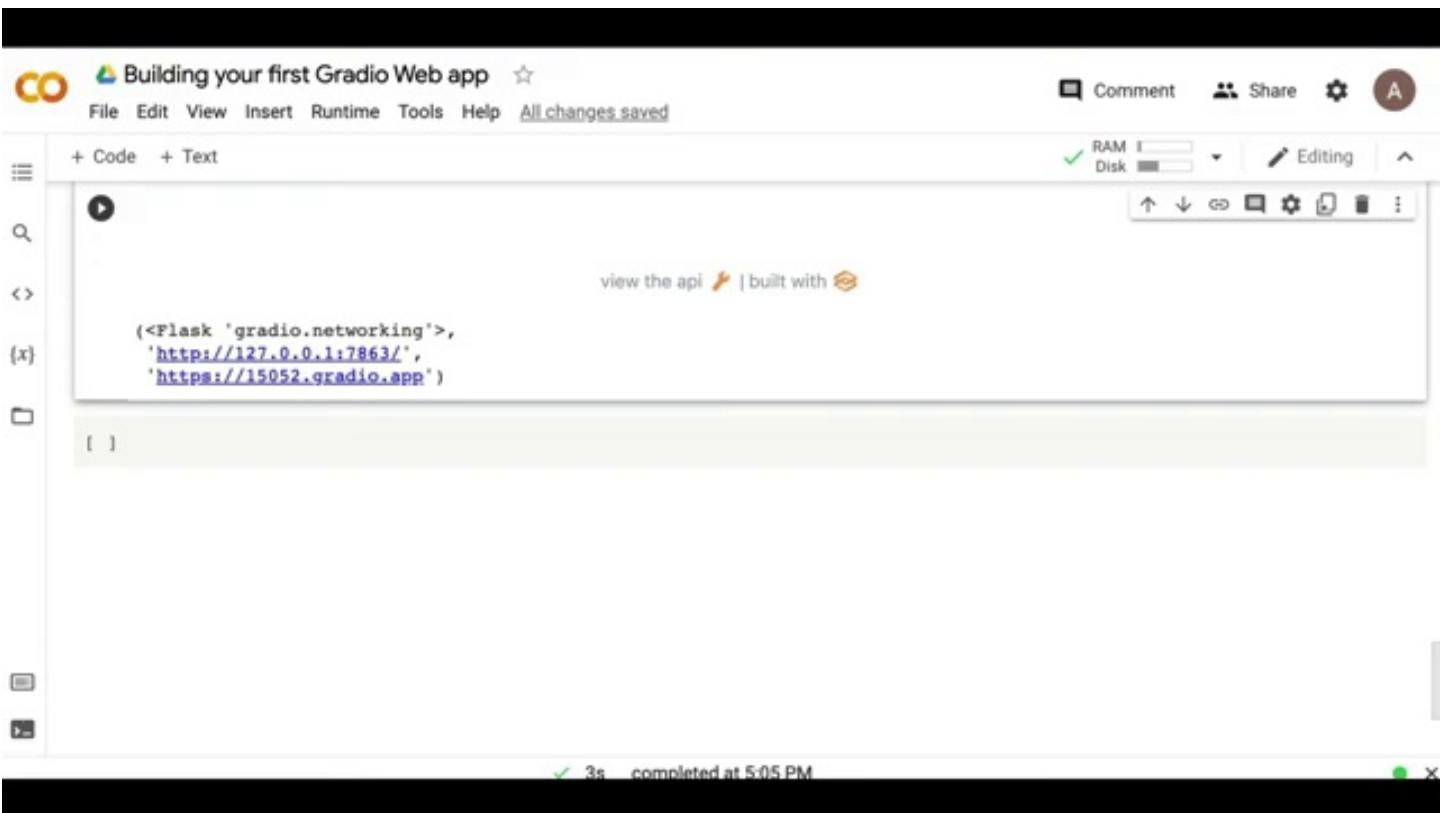
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 95.00 seconds**

[Screenshot](#)[Flag](#)[Clear](#)[Submit](#)

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 96.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

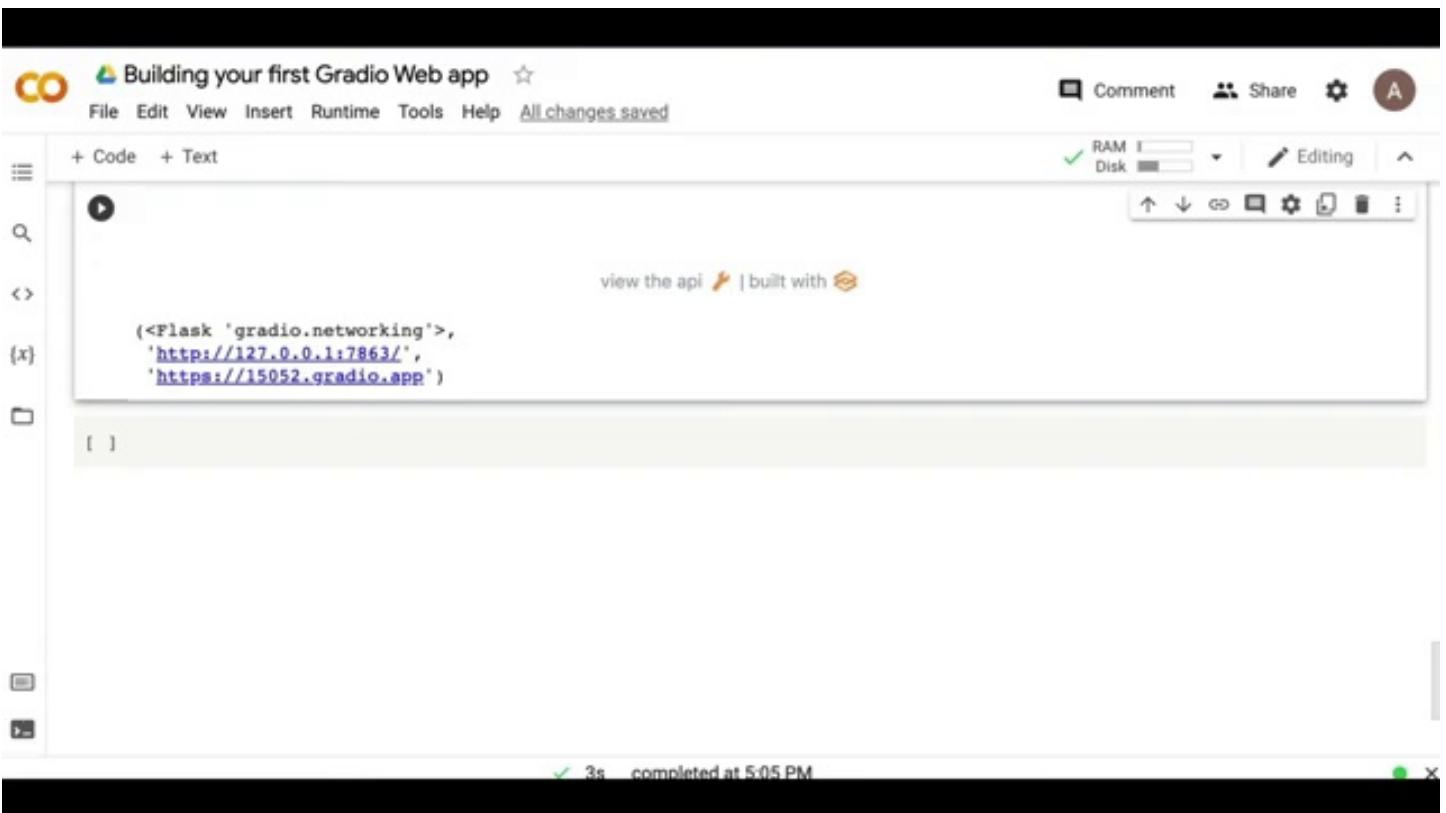
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 97.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

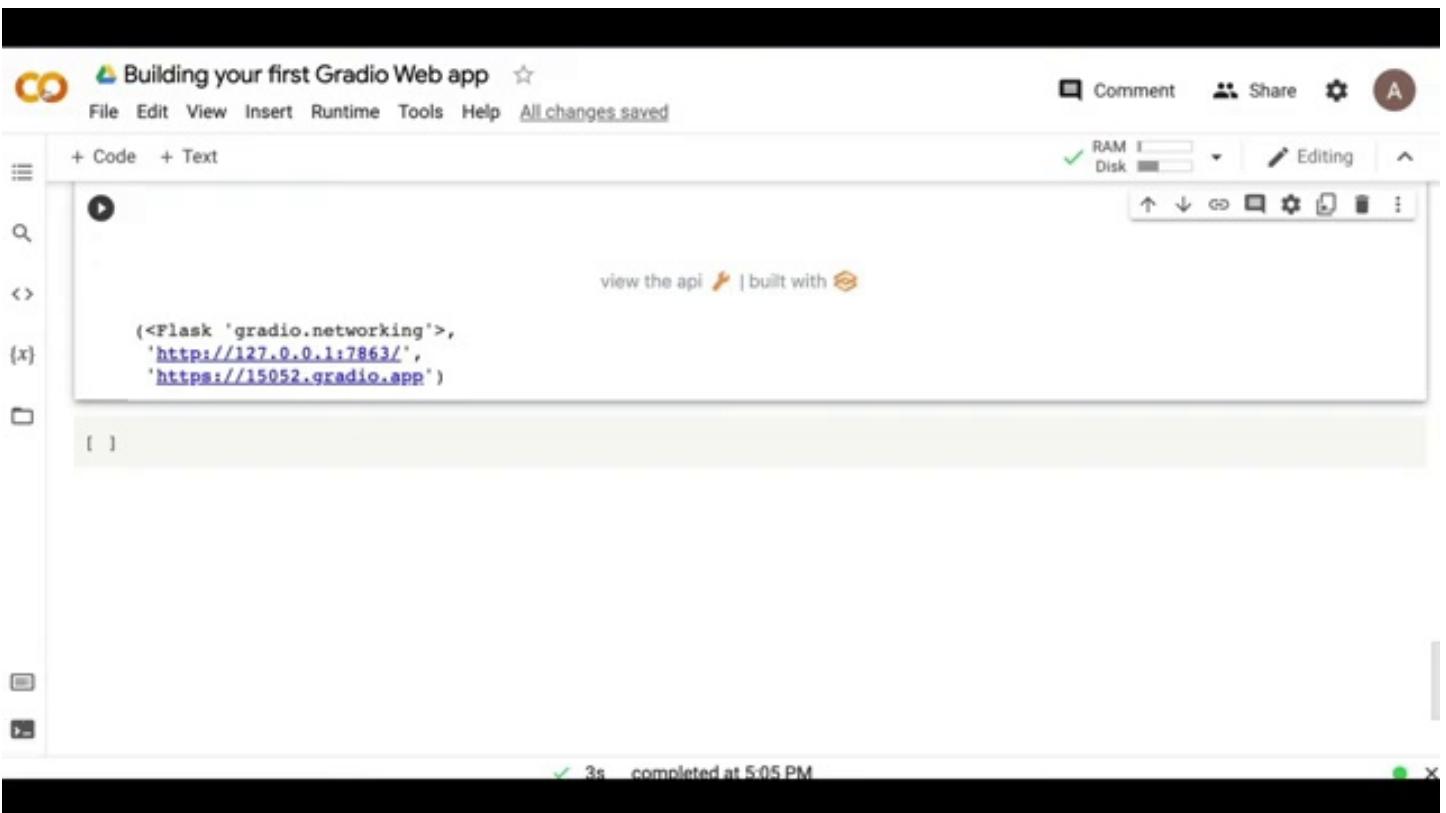
RAM Disk Editing

view the api | built with

```
{<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app'}
```

[ ]

✓ 3s completed at 5:05 PM



**Timestamp: 98.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

(x)

3s completed at 5:05 PM

The screenshot shows a web-based development environment. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Comment', 'Share', 'Settings', and a user profile. On the left, there are several small icons for file operations like search, copy, and paste. The main area contains a code editor with a single line of Python code:

```
<Flask 'gradio.networking',  
'http://127.0.0.1:7863/',  
'https://15052.gradio.app'>
```

Below the code editor is a toolbar with icons for file operations like upload, download, and settings. At the bottom of the screen, there's a progress bar indicating '3s completed at 5:05 PM'.

**Timestamp: 99.00 seconds**

The `numpy` wrap will components in a test. For example, if we wanted to compare multiple functions that have the same input and return types, we can when pass a list of functions for quick comparison.

## Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             (.272, .534, .131)])
    sephia_img = (input_img @ sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```



**Timestamp: 100.00 seconds**

Drop Image Here  
- Or -  
Click to Upload

Clear      Submit

Additionally, our Image input interface comes with an 'edit' button which opens tools for cropping, flipping, rotating, drawing over, and applying filters to images. We've found that manipulating images in this way will often reveal hidden flaws in a model.

In addition to images, Gradio supports other media input types, such as audio or video uploads. Read about these in the [Docs](#).

### Working with Data

You can use Gradio to support inputs and outputs from your typical data libraries, such as numpy arrays, pandas dataframes, and plotly graphs. Take a look at the demo below (ignore the complicated data manipulation in the function!)

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
```

**Timestamp: 101.00 seconds**

## Working with Data

You can use Gradio to support inputs and outputs from your typical data libraries, such as numpy arrays, pandas dataframes, and plotly graphs. Take a look at the demo below (ignore the complicated data manipulation in the function!)

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", 12, 14, 18}, {"Name": "Alice", 14, 17, 2}, {"Name": "Sara", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
```

**Timestamp: 102.00 seconds**

## Working with Data

You can use Gradio to support inputs and outputs from your typical data libraries, such as numpy arrays, pandas dataframes, and plotly graphs. Take a look at the demo below (ignore the complicated data manipulation in the function!)

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.loc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "12": 12, "14": 14, "18": 18}, {"Name": "Alice", "14": 14, "17": 17, "2": 2}, {"Name": "Sona", "8": 8, "9.5": 9.5, "12": 12}]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="This interface shows how to use Gradio to work with dataframes, plots, and numpy arrays.")
```

**Timestamp: 103.00 seconds**

## Working with Data

You can use Gradio to support inputs and outputs from your typical data libraries, such as numpy arrays, pandas dataframes, and plotly graphs. Take a look at the demo below (ignore the complicated data manipulation in the function!)

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.loc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", 12, 14, 18}, {"Name": "Alice", 14, 17, 2}, {"Name": "Sara", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="This interface demonstrates how to use Gradio to work with dataframes, plots, and numpy arrays.")
```

**Timestamp: 104.00 seconds**

```

regression_values = np.apply_along_axis(lambda row:
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] * regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "12": 12, "14": 14, "18": 18}, {"Name": "Alice", "14": 14, "17": 17, "2": 2}, {"Name": "Sona", "8": 8, "9.5": 9.5, "12": 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

```

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sales	Feb Sales	Mar Sales
1	Jon	12	14	18

Screenshot

Flag

**Timestamp: 105.00 seconds**

```

regression_values = np.apply_along_axis(lambda row:
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] * regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[["Jon", 12, 14, 18], ["Alice", 14, 17, 21], ["Sara", 8, 9.5, 12]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

```

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sales	Feb Sales	Mar Sales
1	Jon	12	14	18

Screenshot

Flag

**Timestamp: 106.00 seconds**

ScreenshotFlag

ClearSubmit

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

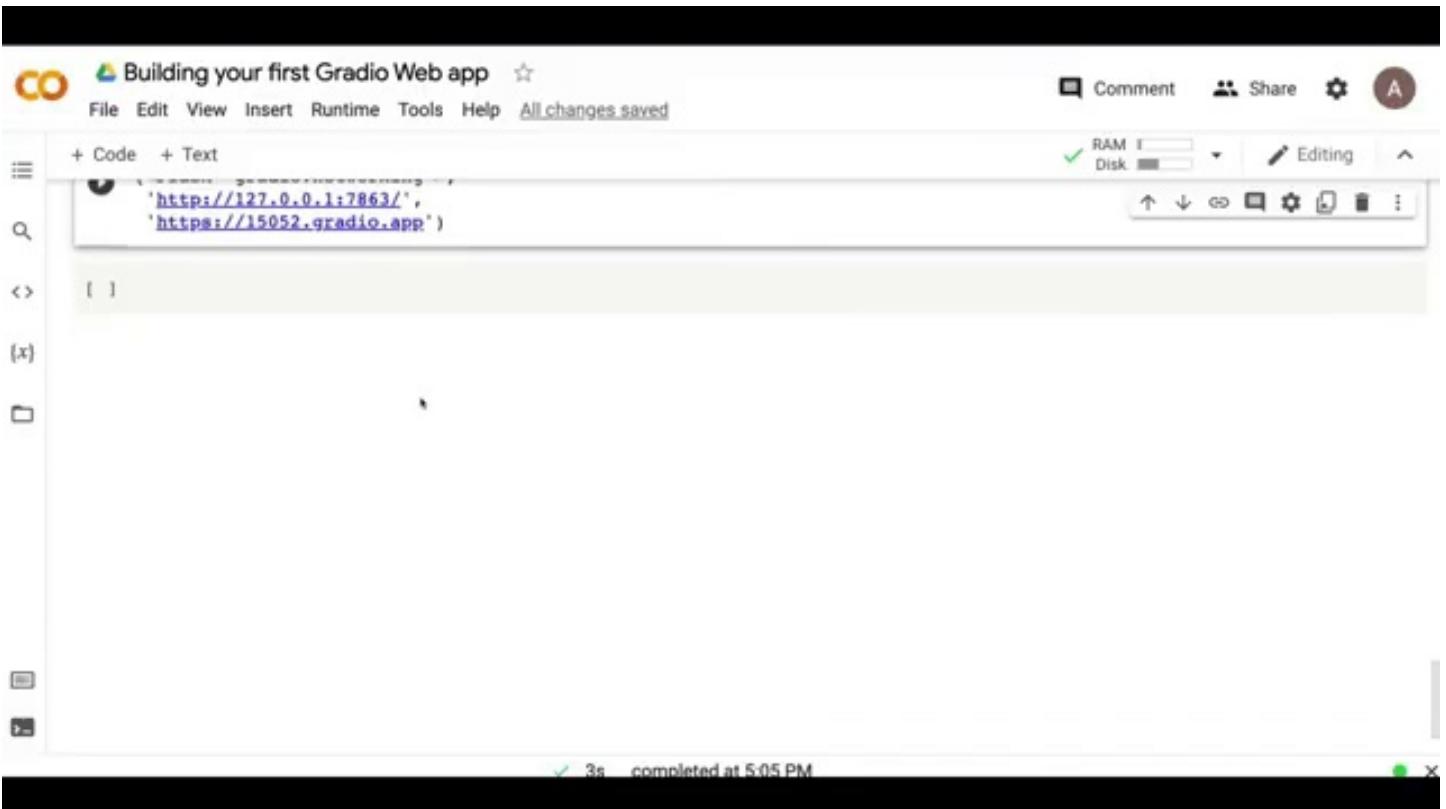
What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 107.00 seconds**



**Timestamp: 108.00 seconds**



+ Code + Text

RAM Disk Editing

```
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return employee_data, plt.gcf(), regression_values  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]  
    ),  
    [  
        "dataframe",  
        "plot",  
        "numpy"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 109.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[16] view the API built with

```
[16] (<Flask 'gradio.networking'>,
      'http://127.0.0.1:7863/',
      'https://15052.gradio.app')

(x) import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.qcf(), regression_values
```

RAM Disk Editing

3s completed at 5:05 PM

**Timestamp: 110.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[16] view the API | Built with

RAM Disk Editing

(x)

```
[16]
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7863/',
 'https://15052.gradio.app')

(x) import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
    ...

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.qcf(), regression_values
    ✓ 3s completed at 5:05 PM
```

**Timestamp: 111.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2)), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 112.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2)), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 113.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2)), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 114.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2)), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 115.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 116.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2)), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

✓ 3s completed at 5:05 PM

**Timestamp: 117.00 seconds**



## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 118.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 119.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]
```

**Timestamp: 120.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 121.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 122.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 123.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 124.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 125.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 126.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 127.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 128.00 seconds**

[Clear](#)[Submit](#)[Screenshot](#)[Flag](#)

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn: the function to wrap`
- `inputs: the input component type(s)`
- `outputs: the output component type(s)`

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 129.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 21}, {"Sana": 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 130.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "14", "18"}, {"Alice": 14, "17", "2}, {"Sana": 8, "9.5", "12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:05 PM

**Timestamp: 131.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:05 PM

**Timestamp: 132.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "14, 18}, [{"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    3s completed at 5:05 PM
```

**Timestamp: 133.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "Alice": 14, "Sana": 8}, {"Jon": 18, "Alice": 17, "Sana": 9.5}, {"Jon": 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    title="Sales Projections for Q1 2024"
)
```

✓ 3s completed at 5:05 PM

**Timestamp: 134.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "Alice": 14, "Sana": 9.5}, {"Jan": 18, "Feb": 17, "Mar": 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    title="Sales Projections for Employee Data"
)
```

✓ 3s completed at 5:05 PM

**Timestamp: 135.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:05 PM

**Timestamp: 136.00 seconds**



+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ]
```

✓ 3s completed at 5:05 PM

**Timestamp: 137.00 seconds**



+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ]
```

✓ 3s completed at 5:05 PM

**Timestamp: 138.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:05 PM

**Timestamp: 139.00 seconds**

Clear

Submit

Screenshot

Flag

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 140.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:05 PM

**Timestamp: 141.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:05 PM

**Timestamp: 142.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:05 PM

**Timestamp: 143.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        gr.inputs.DataFrame(headers=None, row_count=3, col_count=3,
            datatype='str', col_width=None, default=None, type='pandas', label=None)
    ],
    [
        "dataframe",
        "plot",
        "numpy"
    ]
)
```

Component accepts 2D input through a spreadsheet interface.  
Input type: Union[pandas.DataFrame, numpy.array, List[Union[str, float]], List[List[Union[str, float]]]]  
ifac Demos: filter\_records.py, matrix\_transpose.py, tax\_calculator.py

gr.inputs.DataFrame(  
 headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
 default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
,

[  
 "dataframe",  
 "plot",  
 "numpy"

✓ 3s completed at 5:05 PM

**Timestamp: 144.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 145.00 seconds**

Building your first Gradio Web app

All changes saved

+ Code + Text

```
np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

```

✓ 3s completed at 5:05 PM

**Timestamp: 146.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 147.00 seconds**



+ Code + Text

RAM Disk Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 148.00 seconds**



+ Code + Text

RAM

Disk

Editing

^

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 149.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 150.00 seconds**

[Clear](#)[Submit](#)[Screenshot](#)[Flag](#)

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
```

**Timestamp: 151.00 seconds**



+ Code + Text

RAM

Disk

Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
    ),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 152.00 seconds**



+ Code + Text

RAM

Disk

Editing

^

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
    ),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 153.00 seconds**



+ Code + Text

RAM

Disk

Editing

^

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 154.00 seconds**



+ Code + Text

RAM Disk Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 155.00 seconds**



+ Code + Text

RAM

Disk

Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
    ),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 156.00 seconds**



+ Code + Text

RAM

Disk

Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM



**Timestamp: 157.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
        np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 158.00 seconds**



+ Code + Text

RAM Disk Editing

```
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 159.00 seconds**



+ Code + Text

RAM Disk Editing

```
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 160.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
        np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 161.00 seconds**

2. Run the code below as a Python script or in a Python notebook (or in a colab notebook).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME  
Enter your name

Clear Submit Screenshot Flag

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 162.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 163.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 164.00 seconds**



+ Code + Text

RAM Disk Editing

```
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return employee_data, plt.gcf(), regression_values  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]  
    ),  
    [  
        "dataframe",  
        "plot",  
        "numpy"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:05 PM

**Timestamp: 165.00 seconds**



+ Code + Text

```
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return employee_data, plt.gcf(), regression_values  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon": 12, 14, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]  
    ),  
    [  
        "dataframe",  
        "plot",  
        "numpy"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

RAM

Disk

Editing

^

↑ ↓ ↗ ↘ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋

✓ 3s completed at 5:05 PM X

**Timestamp: 166.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

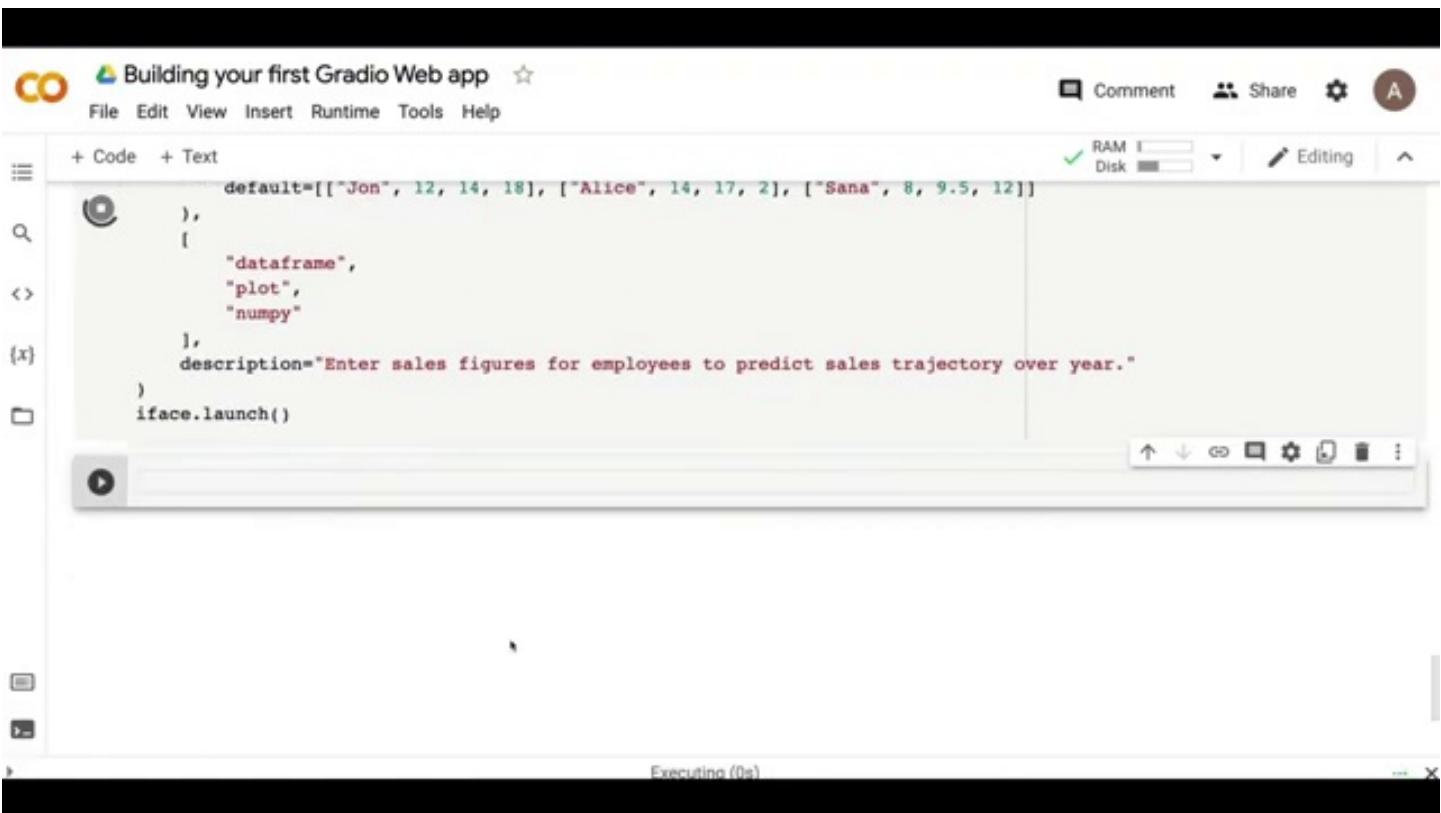
Comment Share A

+ Code + Text

```
    default=[["Jon", 12, 14, 18], ["Alice", 14, 17, 2], ["Sana", 8, 9.5, 12]]  
),  
{  
    "dataframe",  
    "plot",  
    "numpy"  
},  
description="Enter sales figures for employees to predict sales trajectory over year."  
iface.launch()  
  
▶
```

RAM Disk Editing

Executing (0s)



**Timestamp: 167.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
    "dataframe",
    "plot",
    "numpy"
),
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
```

↑ ↓ ⌂ ⚙ 🎨 🗃 🗁 🗁 🗁

Executing (1s) C > ls > setup > url r > url > o > ... > call > https > do > m > send > endh > send > s > co > wrap > cr > do hand ... X

**Timestamp: 168.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

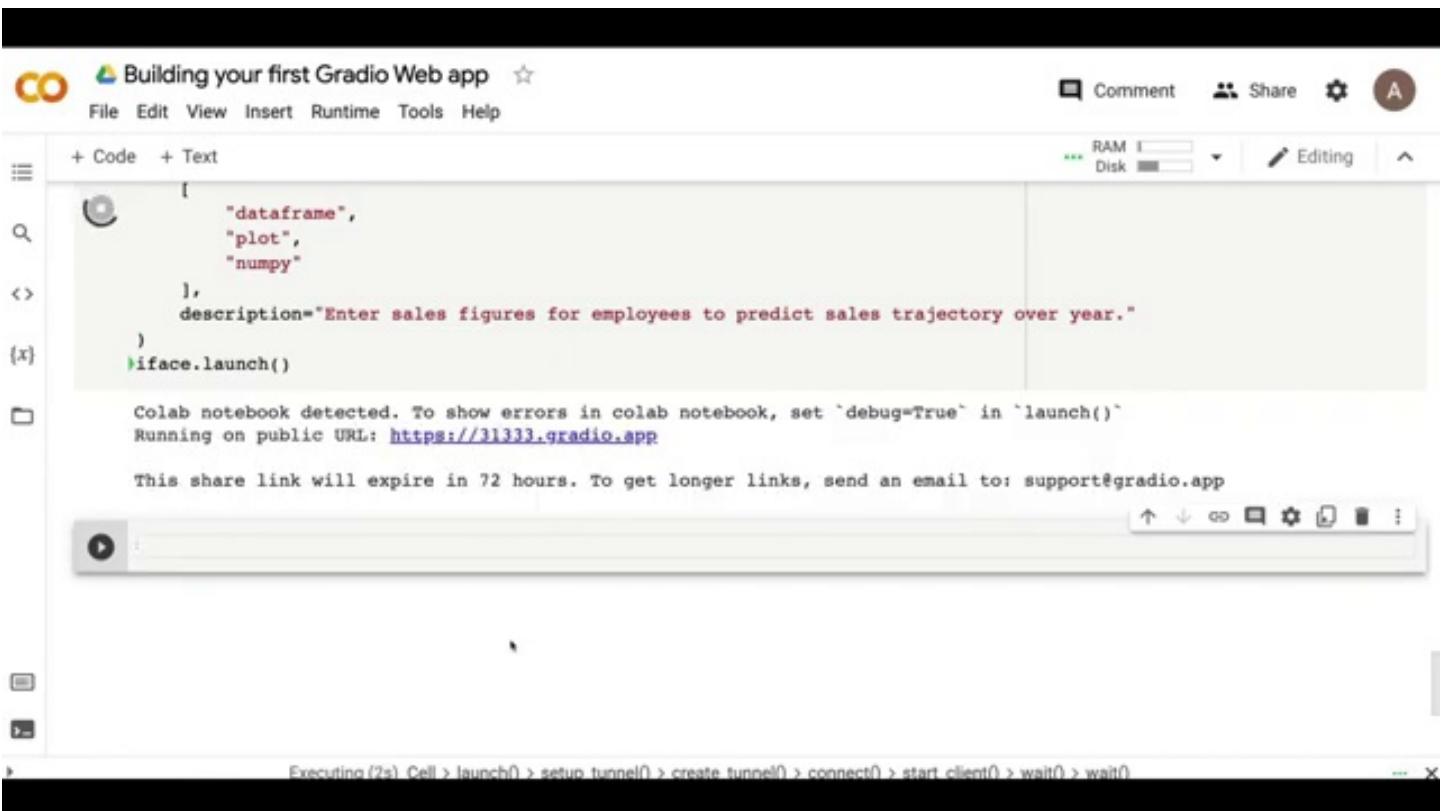
+ Code + Text

```
[{"dataframe": "plot", "numpy": "iface.launch()", "description": "Enter sales figures for employees to predict sales trajectory over year."}]
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Executing (2s): Cell > launch() > setup\_tunnel() > create\_tunnel() > connect() > start\_client() > wait() > wait()



**Timestamp: 169.00 seconds**



**Timestamp: 170.00 seconds**



**Timestamp: 171.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[  ] "dataframe",
  "plot",
  "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal	Apr Sal
1	Jon	12	14	18	
2	Alice	14	17	2	
3	Sana	8	9.5	12	

Screenshot Flag

✓ 3s completed at 5:09 PM

**Timestamp: 172.00 seconds**

To get started running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME	Screenshot	Flag
<input type="text"/>	<a href="#">Screenshot</a>	<a href="#">Flag</a>
<a href="#">Clear</a>	<a href="#">Submit</a>	

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 173.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

[Screenshot](#) [Flag](#)

Clear Submit

✓ 3s completed at 5:09 PM

**Timestamp: 174.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

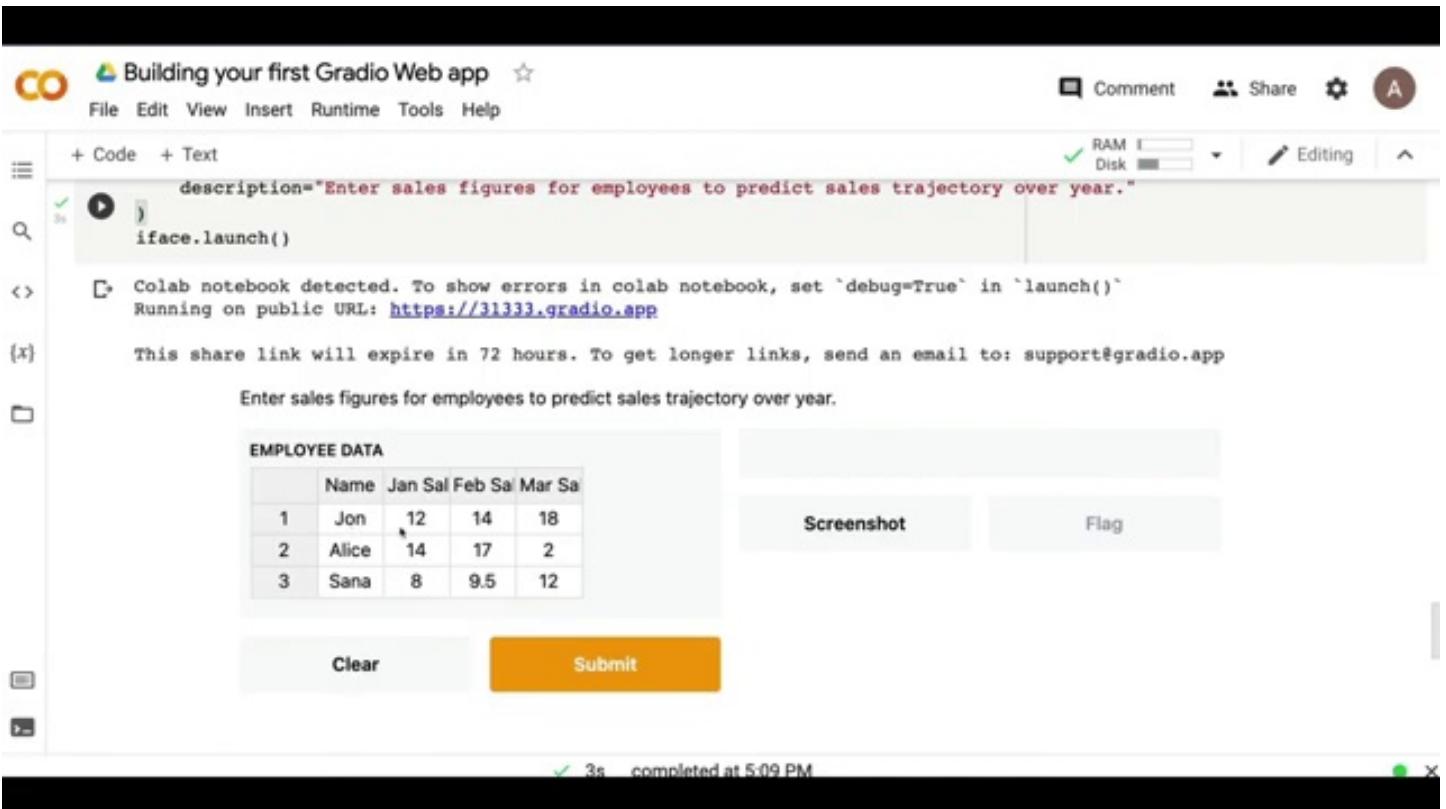
**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**Screenshot** **Flag**

Clear **Submit**

✓ 3s completed at 5:09 PM



**Timestamp: 175.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

[Screenshot](#) [Flag](#)

Clear [Submit](#)

✓ 3s completed at 5:09 PM

**Timestamp: 176.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://31333.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

Enter sales figures for employees to predict sales trajectory over year.

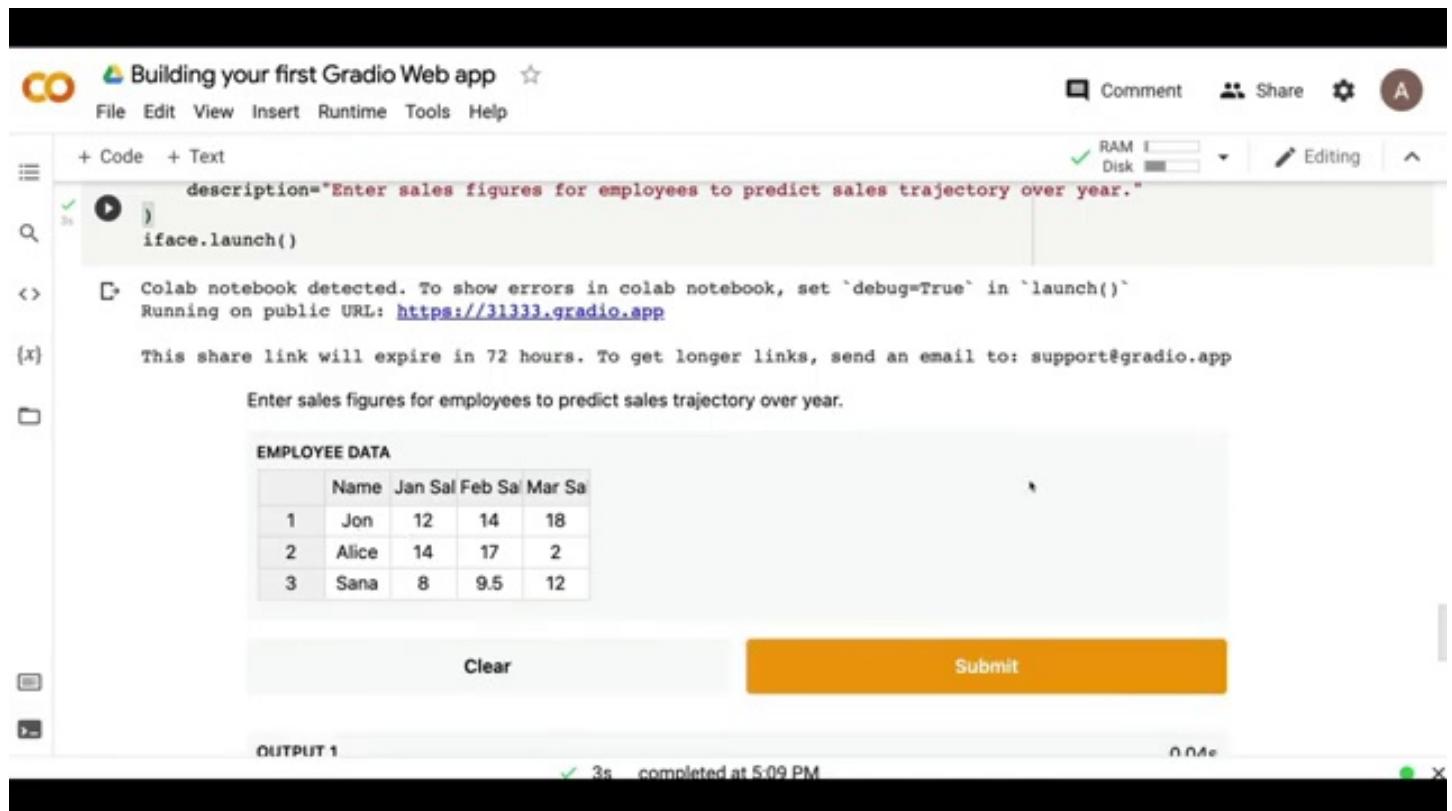
**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT 1**

✓ 3s completed at 5:09 PM

RAM Disk Editing A



**Timestamp: 177.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."  
    iface.launch()  
  
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31333.gradio.app
```

(x) This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

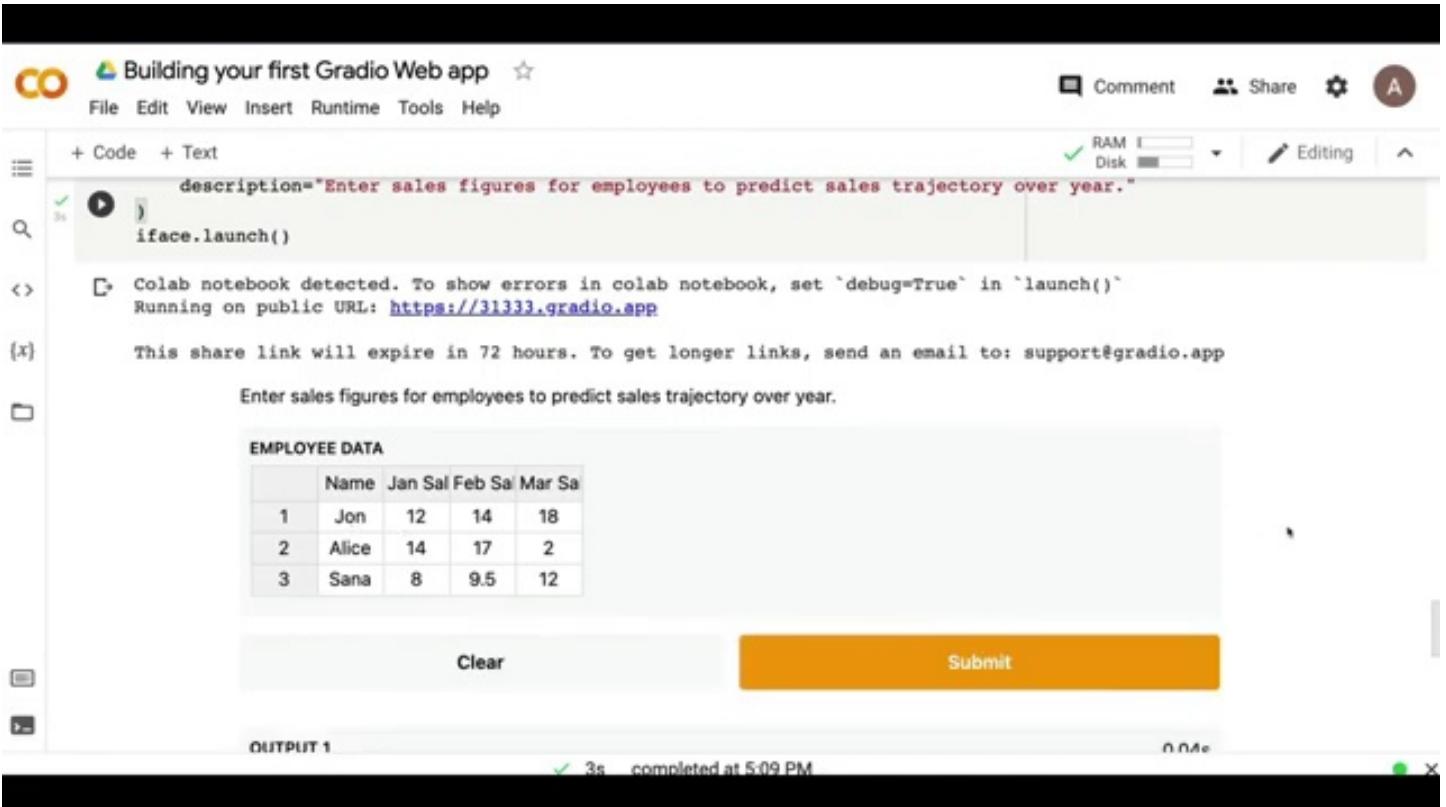
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Clear Submit

OUTPUT 1 ✓ 3s completed at 5:09 PM



**Timestamp: 178.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."  
    iface.launch()  
  
↳ Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31333.gradio.app  
(x) This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

RAM Disk Editing

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Clear Submit

OUTPUT 1

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

0.04s ✓ 3s completed at 5:09 PM

**Timestamp: 179.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Clear Submit

0.04s

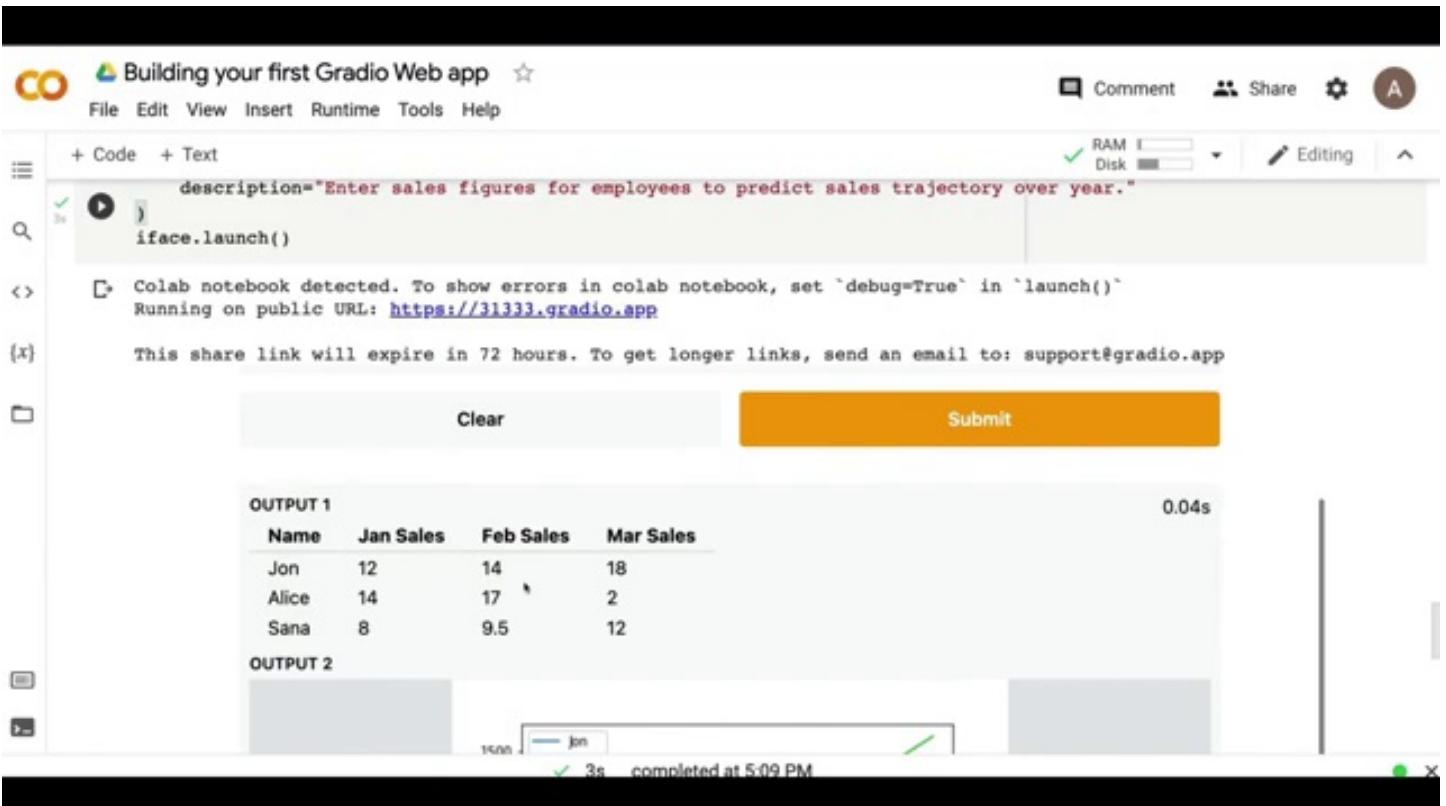
**OUTPUT 1**

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

**OUTPUT 2**

100%  ✓ 3s completed at 5:09 PM

X



**Timestamp: 180.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

**OUTPUT 1**

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

0.04s

**OUTPUT 2**



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents sales, ranging from 1000 to 1500. The X-axis represents time. All three employees show a positive trend, with Sana having the highest sales and Alice the lowest.

1500  
1000

Jon Alice Sana

✓ 3s completed at 5:09 PM

**Timestamp: 181.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
[{"dataframe": "plot", "numpy"}, ], description="Enter sales figures for employees to predict sales trajectory over year."iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan	Sai	Feb	Sai	Mar	Sa
1	Jon	12		14		18	
2	Alice	14		17		2	
3	Sana	8		9.5		12	

✓ 3s completed at 5:09 PM

**Timestamp: 182.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

RAM Disk Editing

```
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return employee_data, plt.gcf(), regression_values  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 14, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 21}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]  
),  
[  
    "dataframe",  
    "plot",  
    "numpy"  
],  
description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:09 PM

**Timestamp: 183.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 184.00 seconds**

To get started running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME

Clear      Submit      Screenshot      Flag

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 185.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

RAM Disk Editing

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values
```

iface = gr.Interface(sales\_projections,
 gr.inputs.DataFrame(
 headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
 default=[[{"Jon": 12, "14", "18"}, {"Alice": 14, "17, 2}, {"Sana": 8, "9.5, 12}]]),
 [
 "dataframe",
 "plot",
 "numpy"
 ],
 description="Enter sales figures for employees to predict sales trajectory over year."

✓ 3s completed at 5:09 PM

**Timestamp: 186.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "14", "18"}, {"Alice": 14, "17, 2}, {"Sana": 8, "9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 187.00 seconds**



+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 188.00 seconds**



+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 189.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 190.00 seconds**



+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 191.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 192.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

✓ RAM Disk Editing

Code editor area:

```
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set 'debug=True' in 'launch()'  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

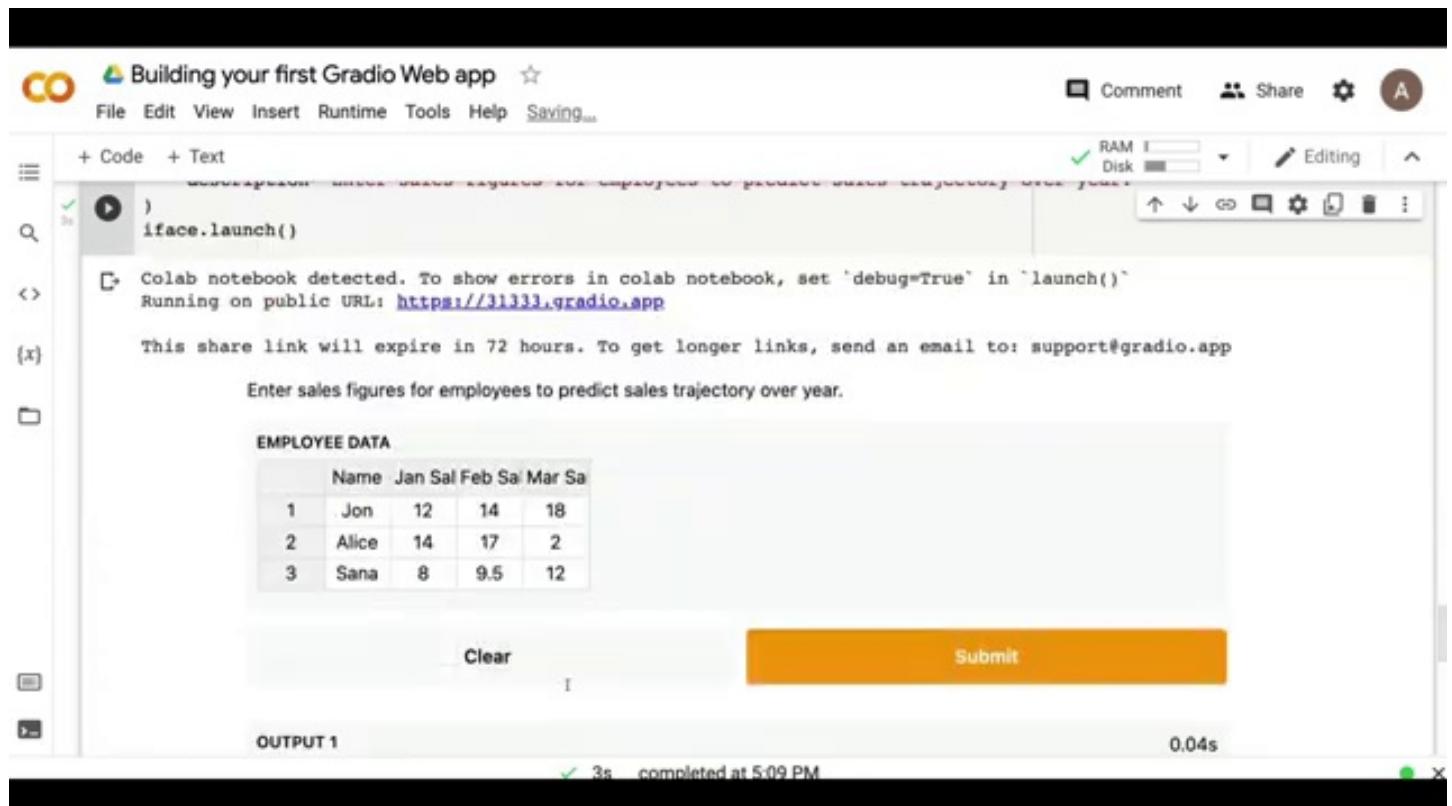
EMPLOYEE DATA

	Name	Jan	Feb	Mar	Sa
1	Jon	12	14	18	
2	Alice	14	17	2	
3	Sana	8	9.5	12	

Clear Submit

OUTPUT 1

✓ 3s completed at 5:09 PM 0.04s



**Timestamp: 193.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Clear Submit

OUTPUT 1

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

0.04s

OUTPUT 2

3s completed at 5:09 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Saving...' status indicator. On the far right are 'Comment', 'Share', 'Settings', and a user profile icon. Below the navigation is a toolbar with icons for file operations like 'New', 'Open', 'Save', etc., and a 'RAM' and 'Disk' monitoring section. The main area has tabs for 'Code' and 'Text'. The 'Code' tab contains a single line of Python code: 'iface.launch()'. A note indicates that a Colab notebook is detected and provides a public URL. A message about share link expiration is also present. Below the code is a 'Clear' button and a large orange 'Submit' button. Under the 'Submit' button is a 'OUTPUT 1' section containing a table with sales data for three individuals: Jon, Alice, and Sana. The table has columns for Name, Jan Sales, Feb Sales, and Mar Sales. The data is as follows:

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

To the right of the table is a timestamp '0.04s'. Below the table is another section labeled 'OUTPUT 2' with a progress bar indicating '3s completed at 5:09 PM'. The bottom right corner features a green circular progress indicator.

**Timestamp: 194.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Clear Submit

0.04s

**OUTPUT 1**

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

**OUTPUT 2**



The chart displays monthly sales for three individuals: Jon, Alice, and Sana. The Y-axis represents sales volume, ranging from 0 to 1500. The X-axis represents the months of January, February, and March. Jon's sales show a steady increase from 12 in Jan to 18 in Mar. Alice's sales are relatively stable, starting at 14 in Jan, peaking at 17 in Feb, and ending at 2 in Mar. Sana's sales show a significant increase from 8 in Jan to 12 in Mar.

1500

Jon  
Alice  
Sana

✓ 3s completed at 5:09 PM

**Timestamp: 195.00 seconds**

To get started running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME	Screenshot	Flag
<input type="text"/>	<a href="#">Screenshot</a>	<a href="#">Flag</a>
<a href="#">Clear</a>	<a href="#">Submit</a>	

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 196.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

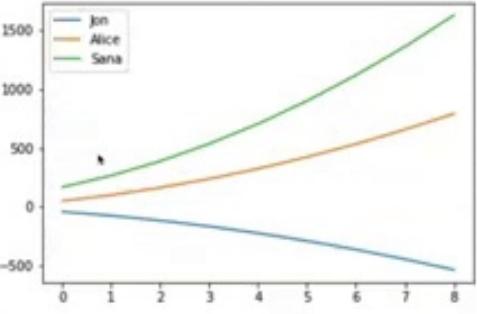
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31333.gradio.app>

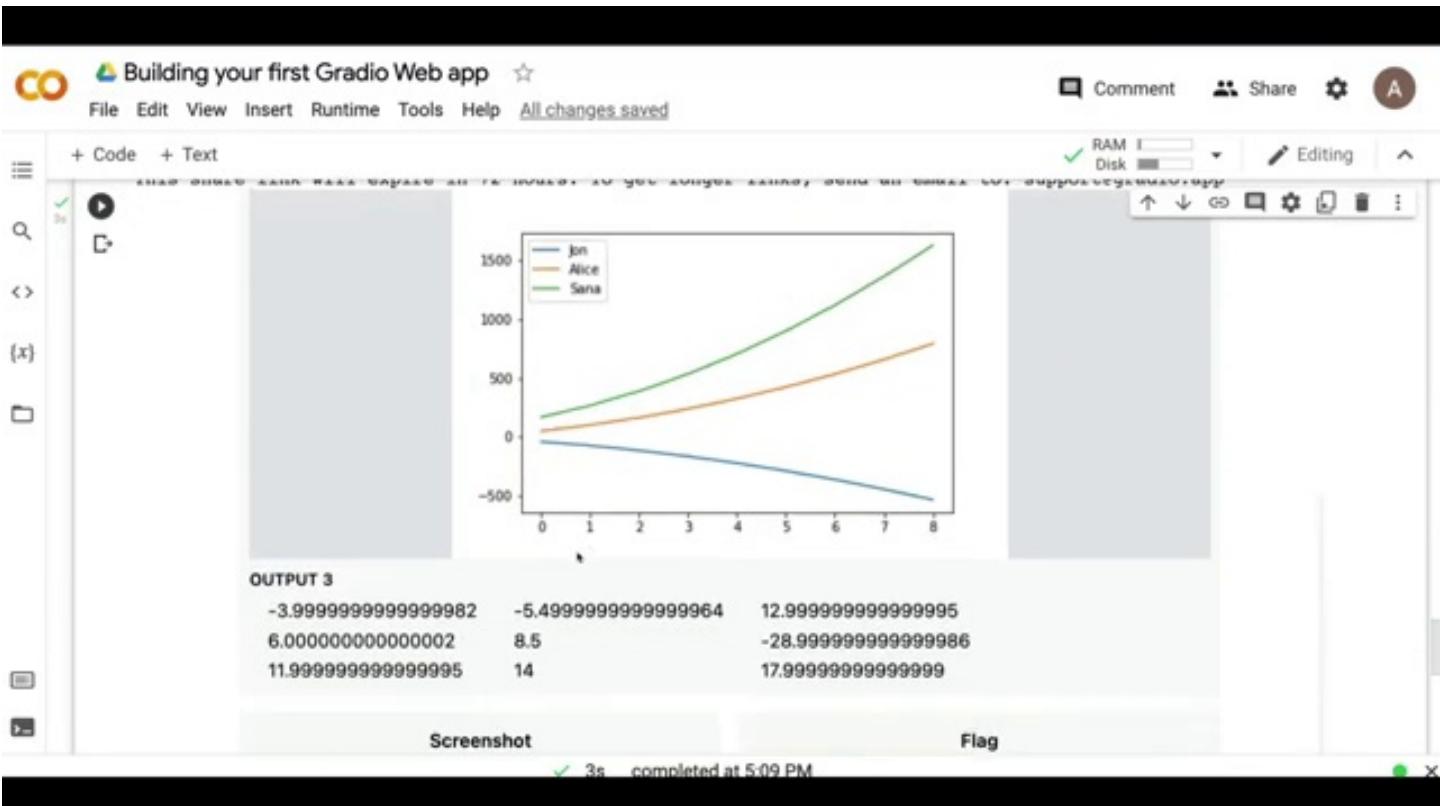
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

OUTPUT 3

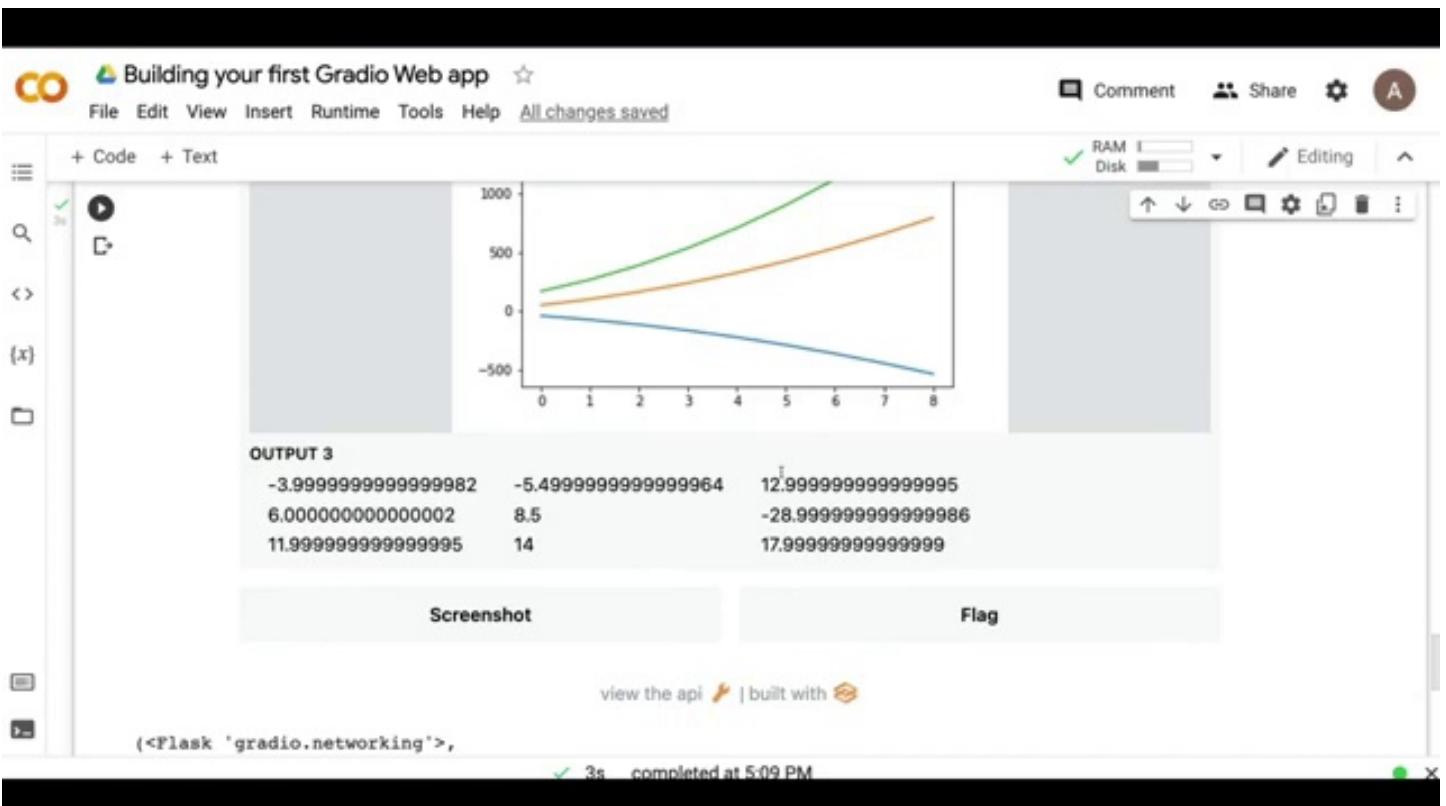
3s completed at 5:09 PM



**Timestamp: 197.00 seconds**



**Timestamp: 198.00 seconds**



**Timestamp: 199.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@g...

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Clear Submit

OUTPUT 1

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

0.04s

OUTPUT 2

✓ 3s completed at 5:09 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the far right of the top bar are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the top bar, there are two tabs: '+ Code' and '+ Text'. The '+ Text' tab is active, displaying a text area with a play button icon and the text: 'This share link will expire in 72 hours. To get longer links, send an email to: support@g...'. Below this is a note: 'Enter sales figures for employees to predict sales trajectory over year.' Underneath the note is a section titled 'EMPLOYEE DATA' containing a table with three rows of employee sales data. The table has columns for Name, Jan Sal, Feb Sal, and Mar Sal. The data is as follows:

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Below the table are two buttons: 'Clear' and 'Submit', with 'Submit' being orange. Under the 'EMPLOYEE DATA' section is a section titled 'OUTPUT 1' containing another table with the same data. This table has columns for Name, Jan Sales, Feb Sales, and Mar Sales. The data is identical to the input table. To the right of this table is a timestamp '0.04s'. At the bottom of the page is a footer bar with a green checkmark icon, the text '3s completed at 5:09 PM', and a close button (X). The left sidebar contains various icons for file operations like copy, paste, and search.

**Timestamp: 200.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@g...

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

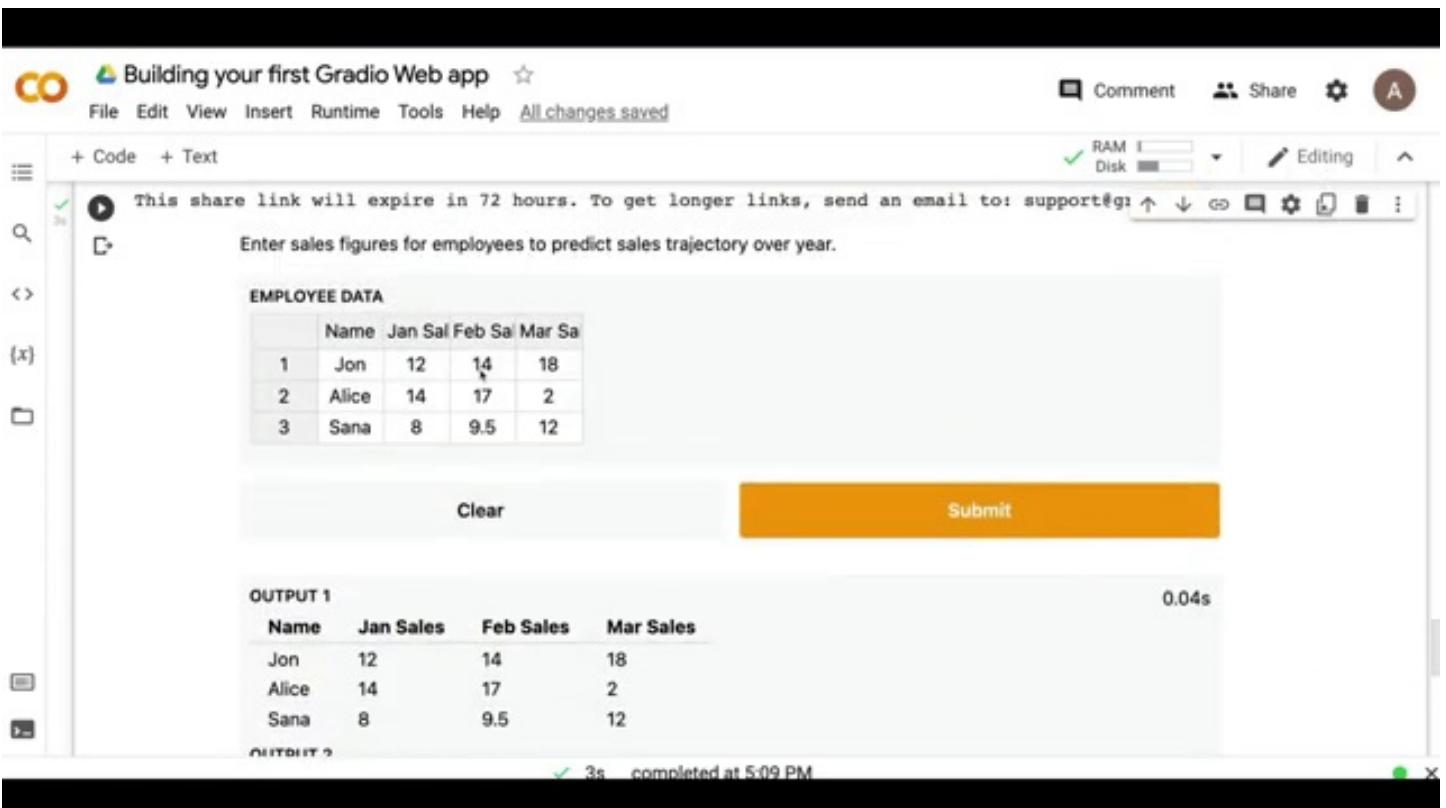
	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	14	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Clear Submit

**OUTPUT 1** 0.04s

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	14	18
Alice	14	17	2
Sana	8	9.5	12

**OUTPUT 2** ✓ 3s completed at 5:09 PM X



**Timestamp: 201.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 14, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ])
```

✓ 3s completed at 5:09 PM

**Timestamp: 202.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
]
```

✓ 3s completed at 5:09 PM

**Timestamp: 203.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
def sales_projections(employee_data):
    sales_data = employee_data.iloc[:, 1:4].astype("int").to_numpy()
    regression_values = np.apply_along_axis(lambda row:
        np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
    projected_months = np.repeat(np.expand_dims(
        np.arange(3,12), 0), len(sales_data), axis=0)
    projected_values = np.array([
        month * month * regression[0] + month * regression[1] + regression[2]
        for month, regression in zip(projected_months, regression_values)])
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
]
```

✓ 3s completed at 5:09 PM

**Timestamp: 204.00 seconds**



**Timestamp: 205.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

RAM Disk Editing

```
headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
,  
[  
    "dataframe",  
    "plot",  
    "numpy"  
,  
description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()  
  
... Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
```

Executing (1s) C → laun... → version ch... → cl... → requie... → requie... → se... → se... → urion... → make req... → netrespo... → beg... → read sta... → readin... → recy.i... → res... → X

**Timestamp: 206.00 seconds**

To get started running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME

Clear

Submit

Screenshot

Flag

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 207.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

... Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

Executing (2s) C > login > setup\_fn > url\_requ > urloco > op > op > call\_ch > https\_o > do\_o > getresno > he > read\_st > readl > recv\_i > rea ... X

**Timestamp: 208.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share ⚙ A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

... Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

Executing (2s). Cell > launch() > url\_ok()

**Timestamp: 209.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

3s completed at 5:09 PM

**Timestamp: 210.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

✓ 3s completed at 5:09 PM X

**Timestamp: 211.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

✓ 3s completed at 5:09 PM X

**Timestamp: 212.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

```
gr = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

✓ 3s completed at 5:09 PM X

**Timestamp: 213.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan	Sal	Feb	Sal	Mar	Sa
1	Jon	12	100	18			
2	Alice	14	17	2			
3	Sana	8	9.5	12			

Screenshot Flag

Clear Submit

✓ 3s completed at 5:09 PM

**Timestamp: 214.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan	Sal	Feb	Sal	Mar	Sa
1	Jon	12		100		18	
2	Alice	14		17		2	
3	Sana	8		9.5		12	

Screenshot Flag

Clear Submit

✓ 3s completed at 5:09 PM

**Timestamp: 215.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        },
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

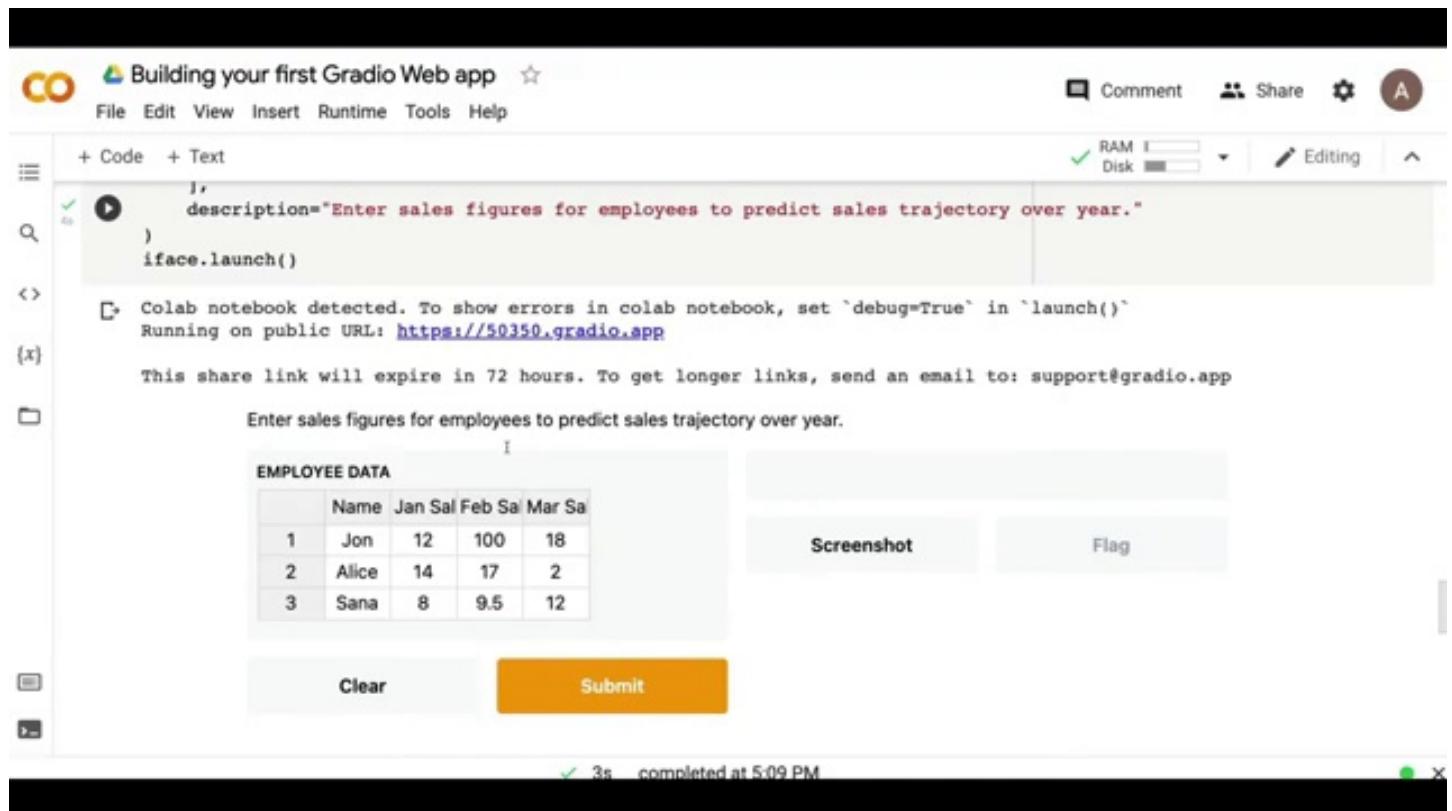
EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

[Screenshot](#) [Flag](#)

Clear [Submit](#)

✓ 3s completed at 5:09 PM



**Timestamp: 216.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

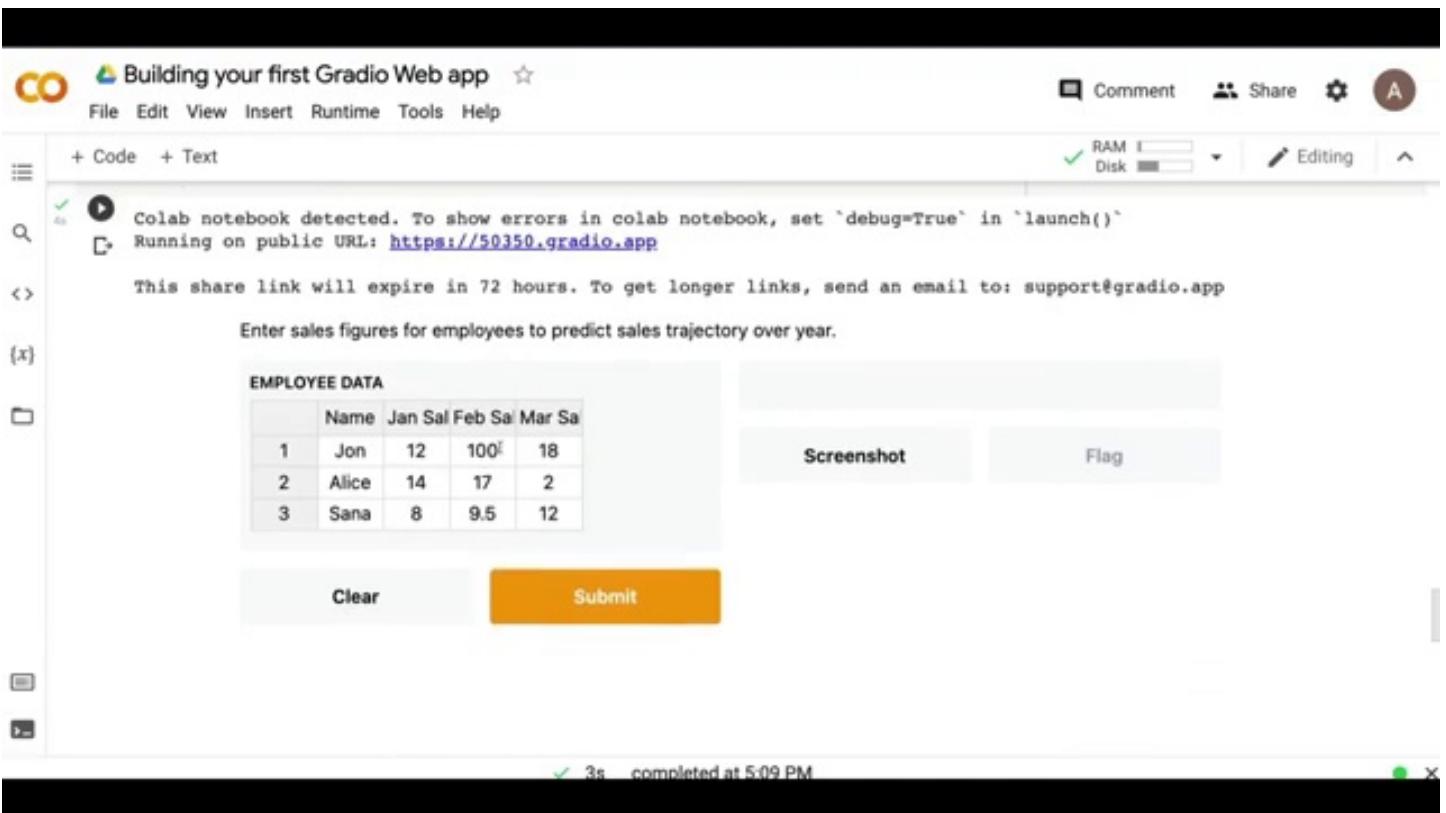
**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**Screenshot** **Flag**

Clear **Submit**

✓ 3s completed at 5:09 PM



**Timestamp: 217.00 seconds**

To get started running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME	Screenshot	Flag
<input type="text"/>	<a href="#">Screenshot</a>	<a href="#">Flag</a>
<a href="#">Clear</a>	<a href="#">Submit</a>	

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 218.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

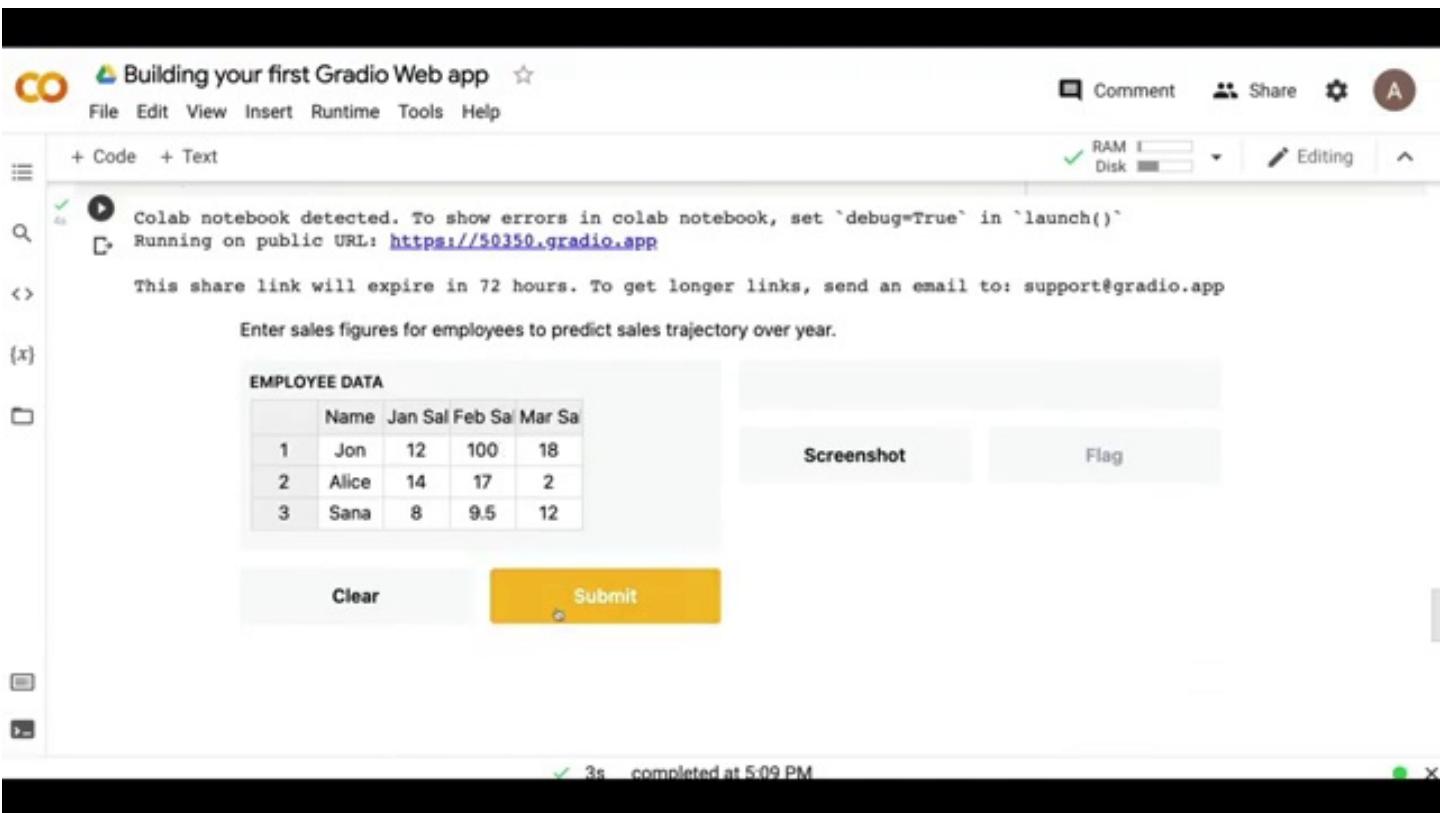
**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Screenshot Flag

Clear Submit

✓ 3s completed at 5:09 PM



**Timestamp: 219.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

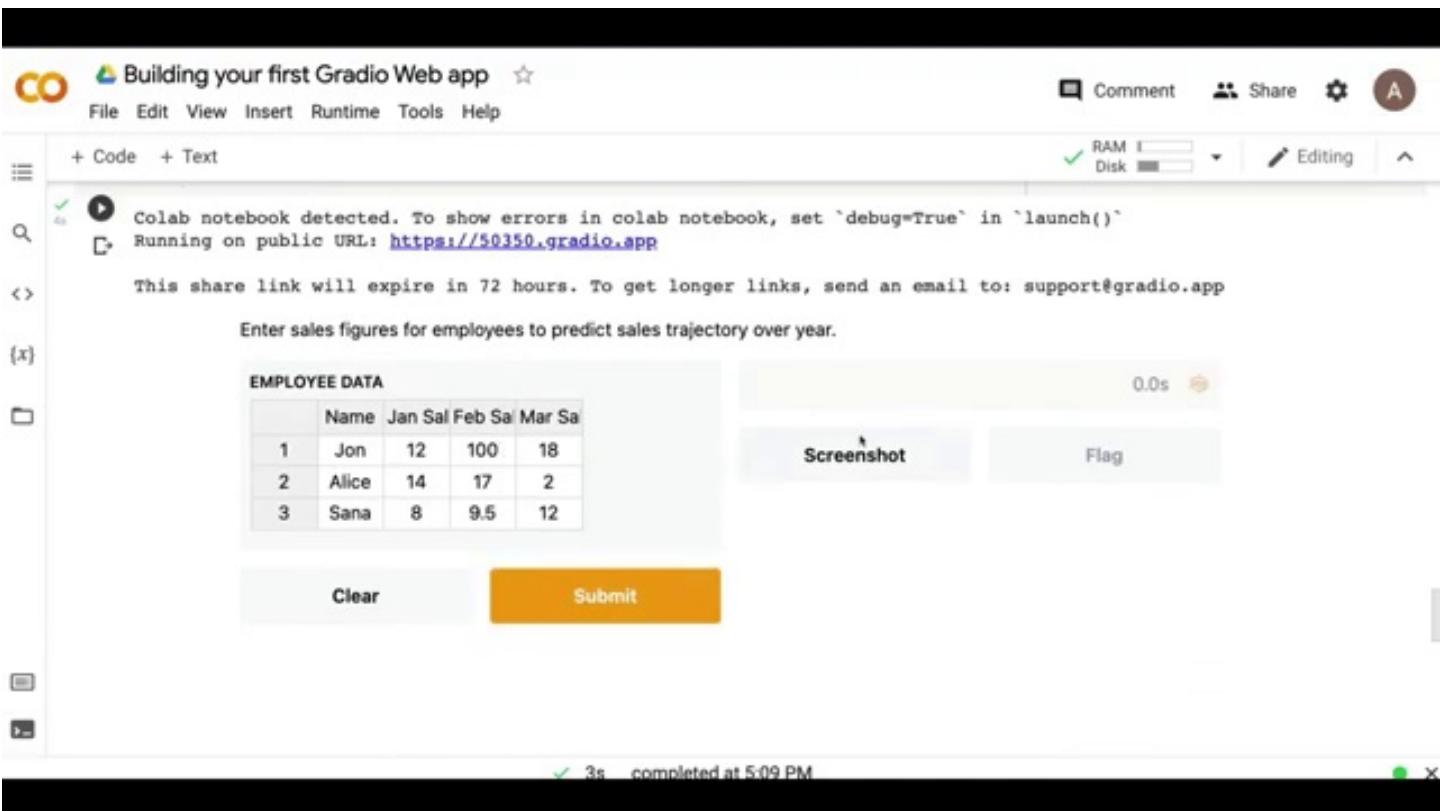
	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

0.0s

Screenshot Flag

Clear Submit

✓ 3s completed at 5:09 PM



**Timestamp: 220.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

(x) Clear Submit

0.02s

**OUTPUT 1**

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

**OUTPUT 2**



The chart displays sales data for three individuals: Jon, Alice, and Sana. The Y-axis represents sales values ranging from 1000 to 2500. The X-axis represents time points. The legend indicates: Jon (blue), Alice (orange), and Sana (green). The chart shows a general upward trend for all individuals over time.

3s completed at 5:09 PM

**Timestamp: 221.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing A

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Clear Submit

0.02s

OUTPUT 1

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

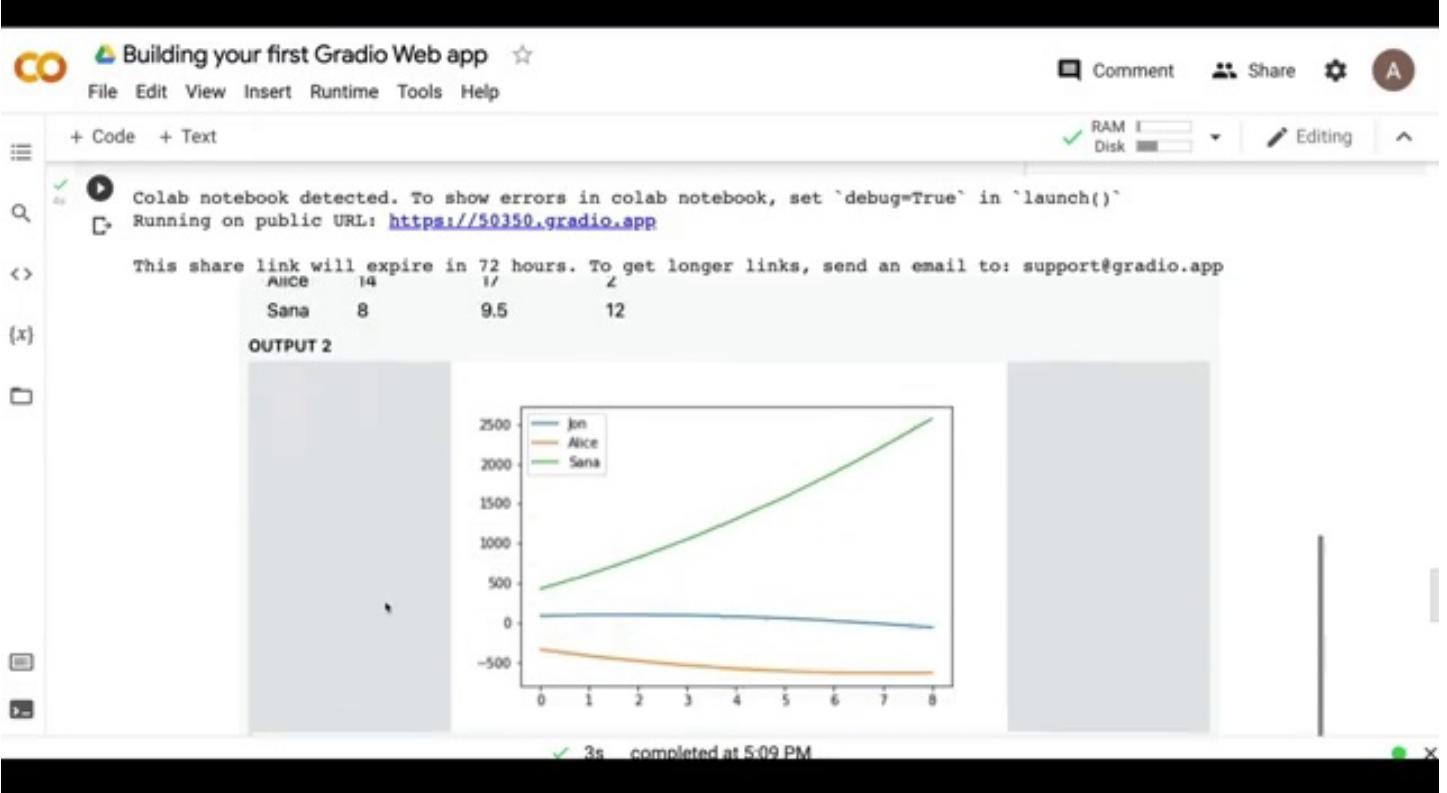
OUTPUT 2



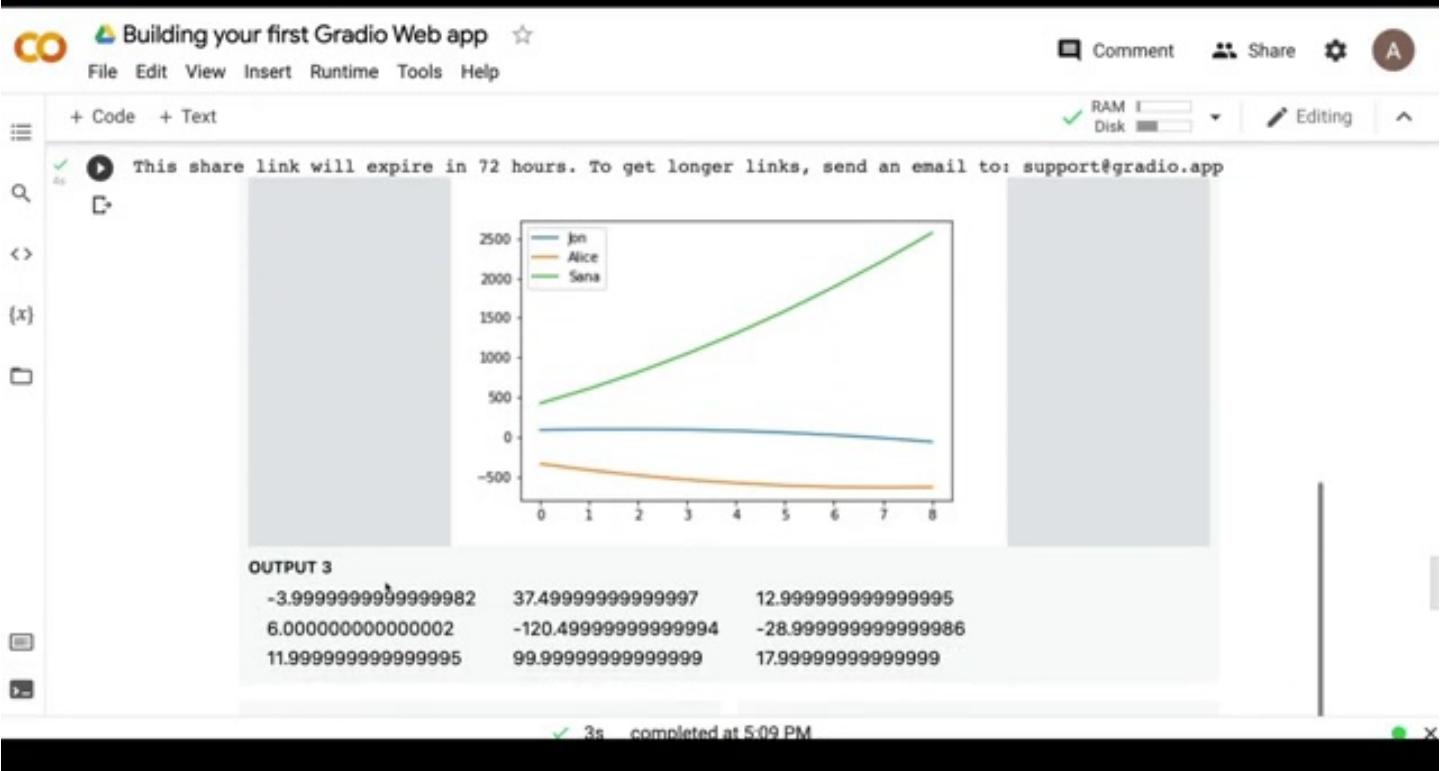
Jon  
Alice  
Sana

✓ 3s completed at 5:09 PM

**Timestamp: 222.00 seconds**



**Timestamp: 223.00 seconds**



**Timestamp: 224.00 seconds**



+ Code + Text

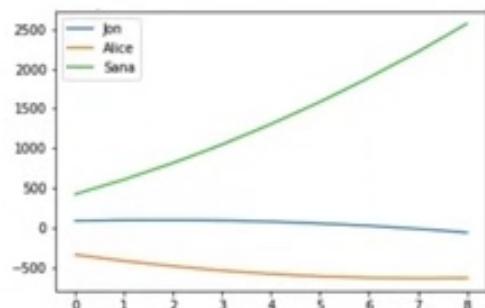
RAM

Disk

Editing



## OUTPUT 2



## OUTPUT 3

-3.999999999999982	37.49999999999997	12.99999999999995
6.000000000000002	-120.4999999999994	-28.99999999999986
11.99999999999995	99.9999999999999	17.99999999999999

✓ 3s completed at 5:09 PM



**Timestamp: 225.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

Comment Share ⚙ A

+ Code + Text RAM Disk Editing ^

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

Clear Submit

**OUTPUT 1** 0.02s

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18
Alice	14	17	2
Sana	8	9.5	12

✓ 3s completed at 5:09 PM

**Timestamp: 226.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing A

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 227.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 228.00 seconds**

To get Gradio running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME	Screenshot	Flag
<input type="text"/>		
<a href="#">Clear</a>	<a href="#">Submit</a>	

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 229.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 230.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return employee_data, plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 231.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return emplplt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 232.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 233.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf(), regression_values

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:09 PM

**Timestamp: 234.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
np.array(np.polyfit(np.polyfit([x,y], row, 2)), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 100, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:09 PM

**Timestamp: 235.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        for month, regression in zip(projected_months, regression_values)):
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
    ✓ 3s completed at 5:09 PM
```

**Timestamp: 236.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        for month, regression in zip(projected_months, regression_values)):
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
    ✓ 3s completed at 5:09 PM
```

**Timestamp: 237.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        for month, regression in zip(projected_months, regression_values)):
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "plot",
    "numpy"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
    ✓ 3s completed at 5:09 PM
```

**Timestamp: 238.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        for month, regression in zip(projected_months, regression_values)):
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]]),
    [
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://50350.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
    ✓ 3s completed at 5:09 PM
```

**Timestamp: 239.00 seconds**

To get Gradio running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

NAME  
John

Clear      Submit      Screenshot      Flag

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 240.00 seconds**



+ Code + Text

RAM Disk Editing

```
        for month, regression in zip(projected_months, regression_values)):
    plt.plot(projected_values.T)
    plt.legend(employee_data["Name"])
    return plt.gcf()

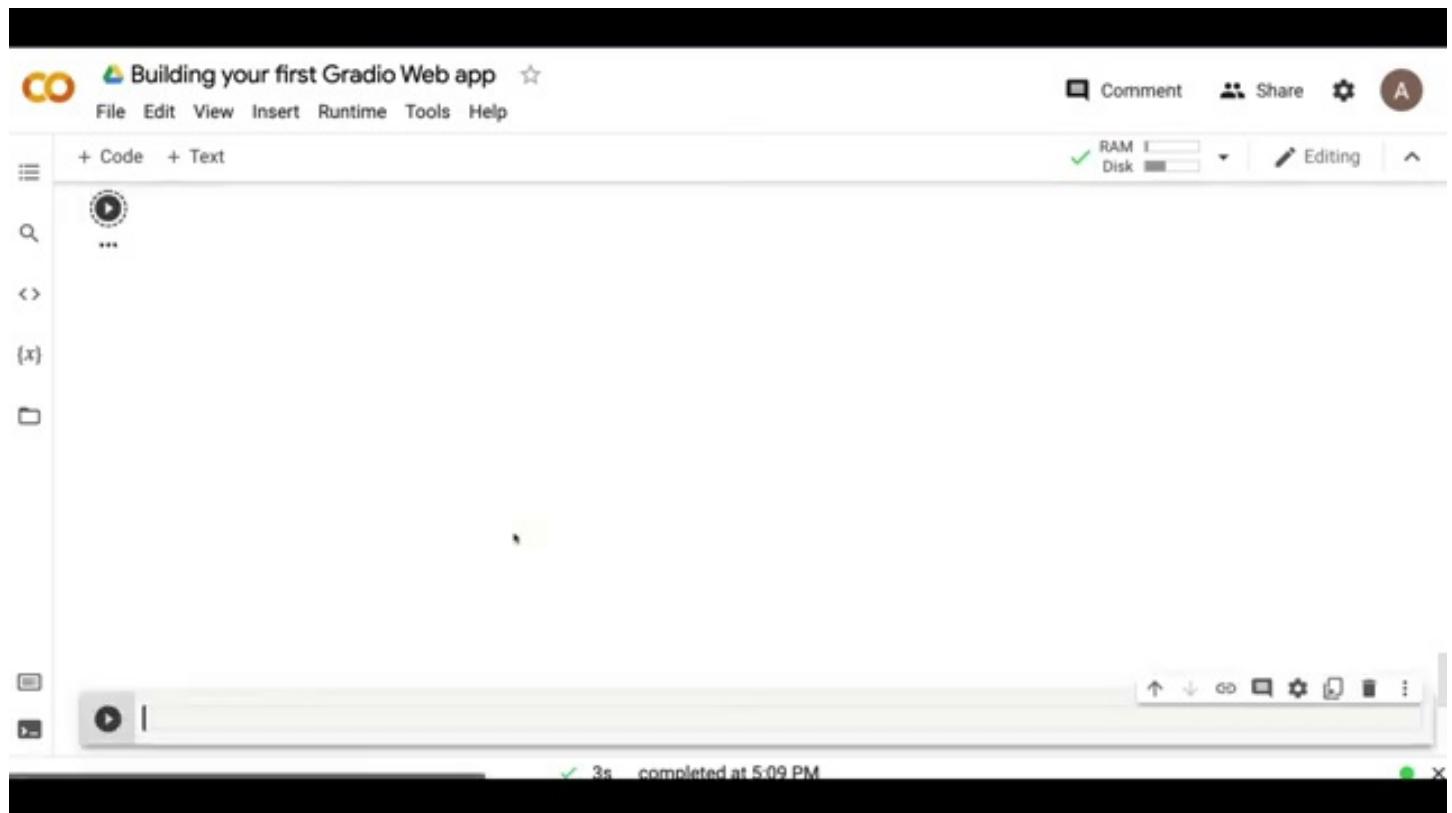
iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[{"plot": "plot"}],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://50350.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:09 PM

**Timestamp: 241.00 seconds**



**Timestamp: 242.00 seconds**



**Timestamp: 243.00 seconds**



**Timestamp: 244.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

... Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

(x) This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

RAM Disk Editing

Executing (2a): Cell > launch() > setup\_tunnel() > create\_tunnel() > connect() > auth() > auth\_pubkey() > wait\_for\_response() > wait() > wait()

**Timestamp: 245.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

RAM Disk Editing

(x)

3s completed at 5:10 PM



**Timestamp: 246.00 seconds**



**Timestamp: 247.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

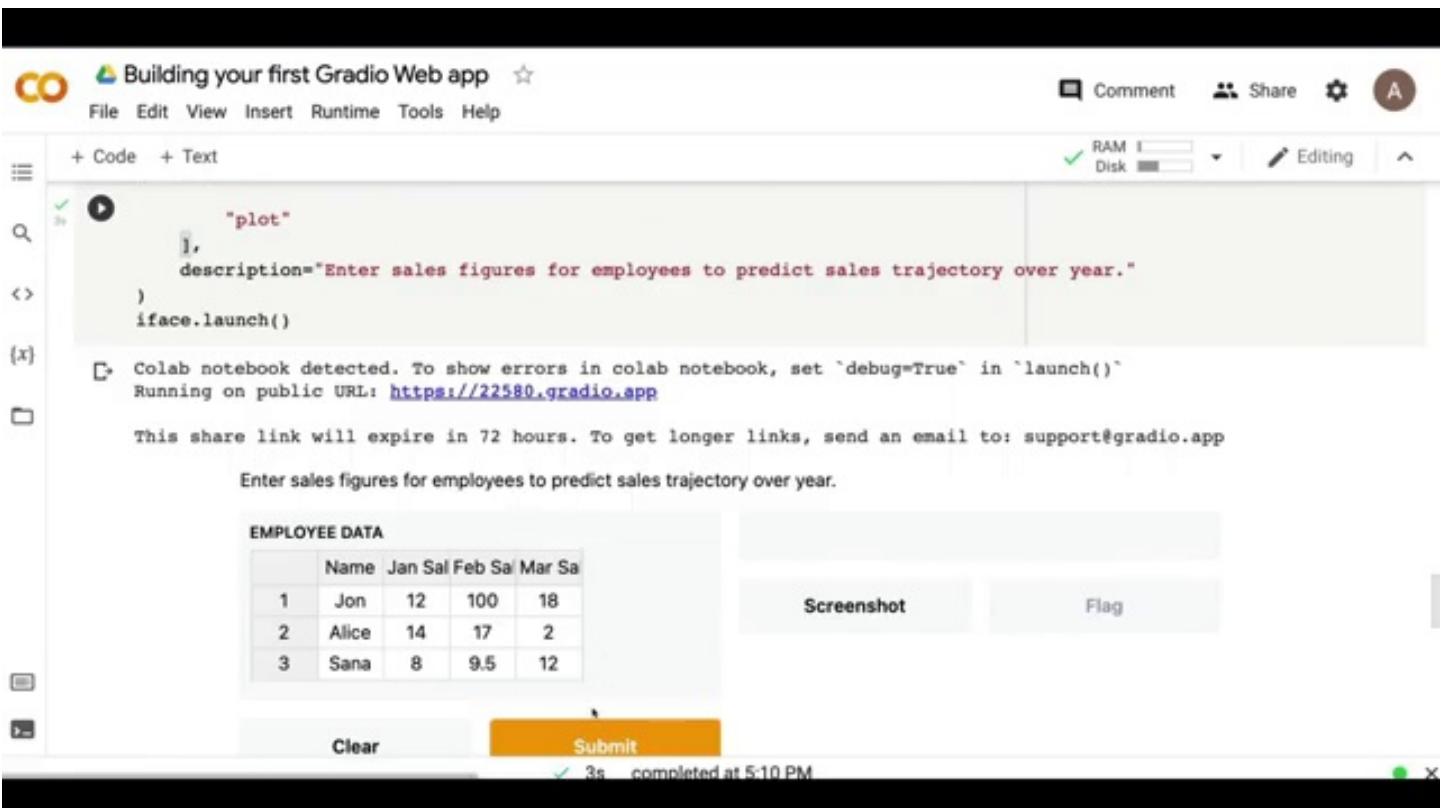
	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

[Screenshot](#) [Flag](#)

Clear Submit

✓ 3s completed at 5:10 PM

RAM Disk Editing A



**Timestamp: 248.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

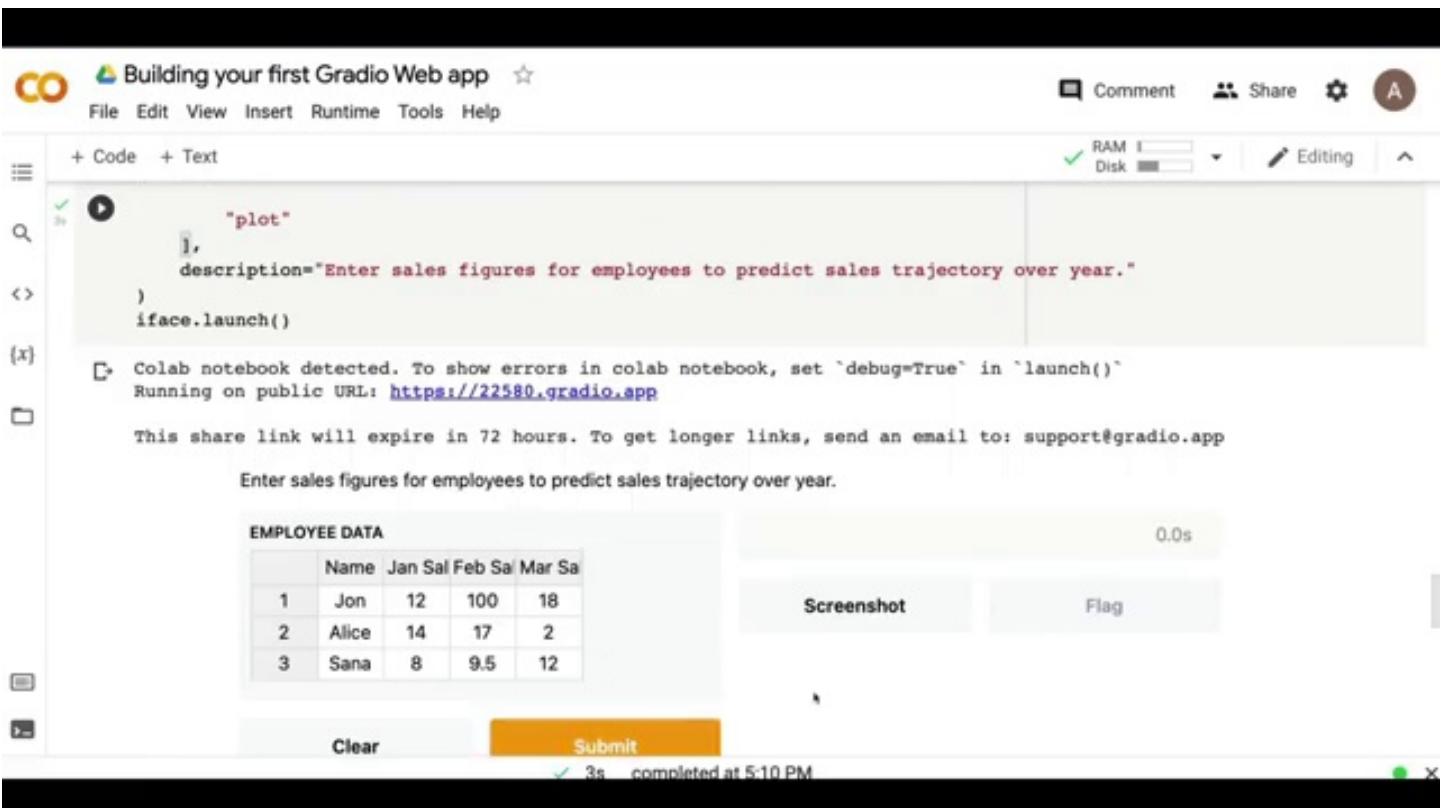
EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal	Apr Sal
1	Jon	12	100	18	
2	Alice	14	17	2	
3	Sana	8	9.5	12	

0.0s

Screenshot Flag

Clear Submit ✓ 3s completed at 5:10 PM



**Timestamp: 249.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

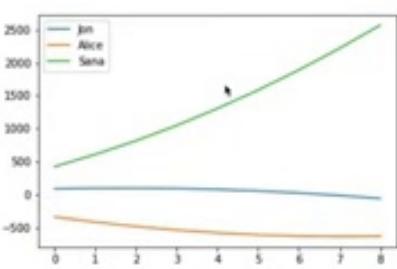
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

(x) Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease slightly to about 10 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease steadily to about -200 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase steadily to about 2200 by Month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 250.00 seconds**

To get Gradio running with a simple example, follow these three steps:

1. Install Gradio from pip.

```
pip install gradio
```

2. Run the code below as a Python script or in a Python notebook (or in a [colab notebook](#)).

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

**Timestamp: 251.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

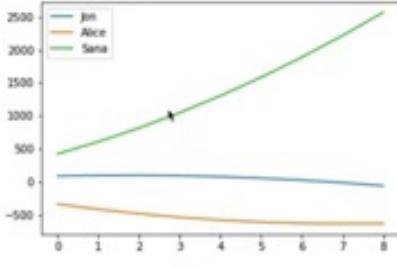
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales figures for three employees (Jon, Alice, and Sana) over an 8-month period. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase steadily to about 2500 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 252.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

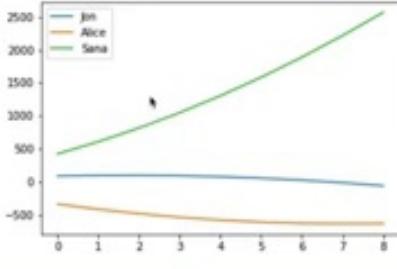
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 253.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

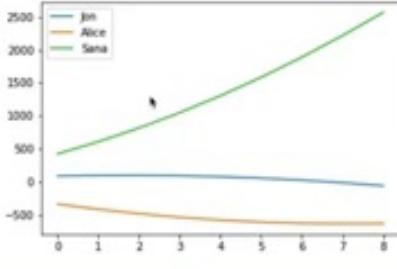
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 254.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

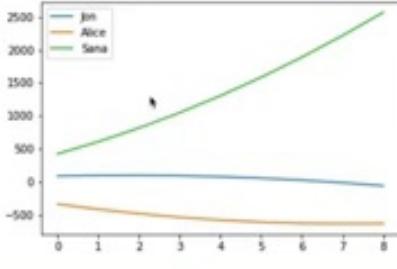
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 255.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

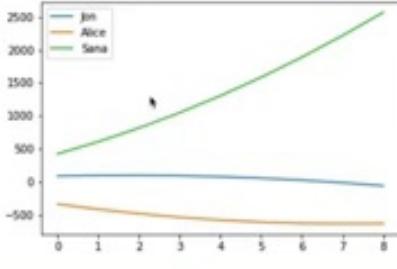
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 256.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

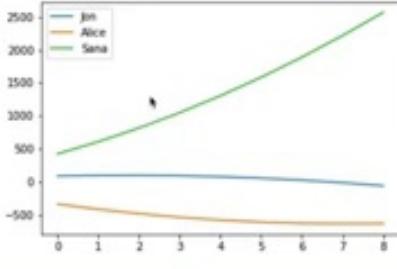
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 257.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

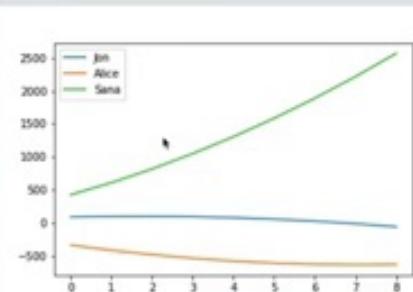
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, 100 in Feb, and 18 in Mar, showing a slight downward trend. Alice's sales (orange line) start at 14 in Jan, 17 in Feb, and 2 in Mar, showing a significant decline to approximately -400 by Mar. Sana's sales (green line) start at 8 in Jan, 9.5 in Feb, and 12 in Mar, showing a steady increase to about 2200 by Mar.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 258.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

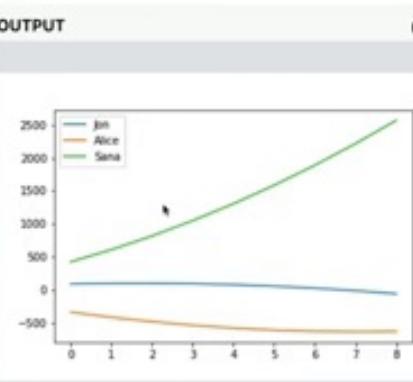
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the projected sales for three employees over an 8-month period. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. The legend identifies the employees: Jon (blue line), Alice (orange line), and Sana (green line). Sana's sales show a strong upward trend, starting at 500 and reaching approximately 2200 by month 8. Alice's sales remain relatively flat, starting near -200 and ending around -300. Jon's sales also show a slight upward trend, starting at 100 and ending at approximately 150.

Month	Jon	Alice	Sana
0	100	-200	500
2	120	-200	1000
8	150	-300	2200

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 259.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

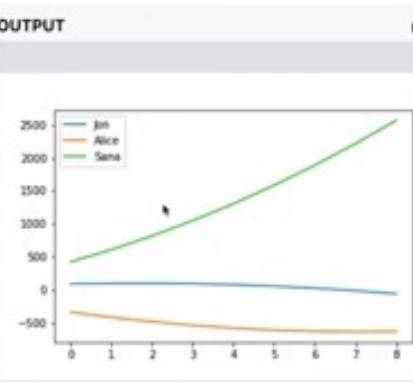
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the projected sales for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (0 to 2500) and the X-axis represents Months (0 to 8). Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -300 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 260.00 seconds**



+ Code + Text

RAM

Disk

Editing



This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app



Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

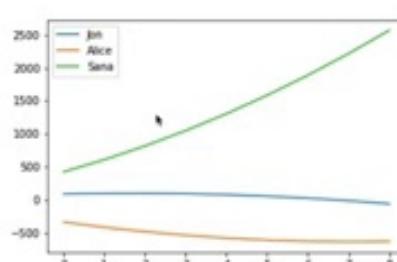


Clear

Submit

**OUTPUT**

0.02s



✓ 3s completed at 5:10 PM



**Timestamp: 261.00 seconds**

The screenshot shows a simple user interface. At the top, there is a header with the word "Gradio". Below it is a search bar with the placeholder "Search...". Underneath the search bar is a large input field with the label "NAME" and a yellow "Submit" button. To the left of the input field is a "Clear" button. At the bottom right of the page are three buttons: "Screenshot", "Flag", and "Docs".

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
```

**Timestamp: 262.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

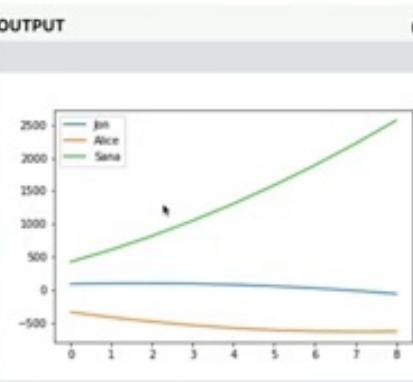
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the projected sales for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (0 to 2500) and the X-axis represents Months (0 to 8). Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -300 by month 8. Sana's sales start at 8 and increase to approximately 2200 by month 8.

Month	Jon	Alice	Sana
0	12	14	8
1	12	13	8.5
2	12	12	9.5
3	12	11	10.5
4	12	10	11.5
5	12	9	12.5
6	12	8	13.5
7	12	7	14.5
8	12	-300	2200

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 263.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share ⚙ A

+ Code + Text

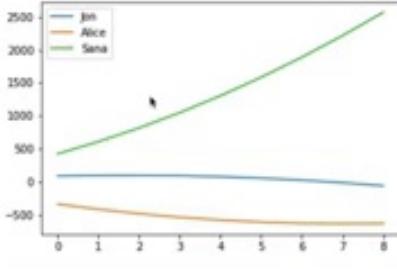
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The graph displays the projected sales for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (0 to 2500) and the X-axis represents Months (0 to 8). Jon's sales remain constant at 12. Alice's sales decrease from 14 to approximately -300. Sana's sales increase from 8 to approximately 2200.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 264.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share ⚙ A

+ Code + Text

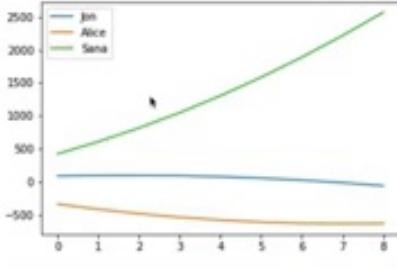
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The graph displays the projected sales for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (0 to 2500) and the X-axis represents Months (0 to 8). Jon's sales remain constant at 12. Alice's sales decrease from 14 to approximately -400. Sana's sales increase from 8 to approximately 2200.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 265.00 seconds**



+ Code + Text

RAM

Disk

Editing

^



This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app



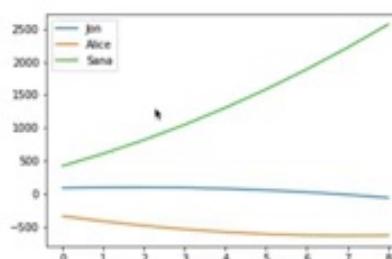
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 266.00 seconds**



+ Code + Text

RAM

Disk

Editing



This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app



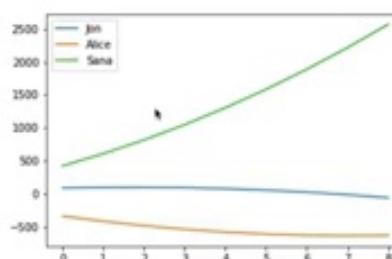
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 267.00 seconds**



+ Code + Text

RAM

Disk

Editing



This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app



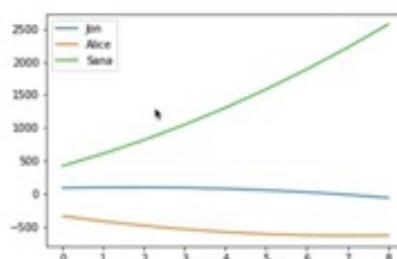
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 268.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

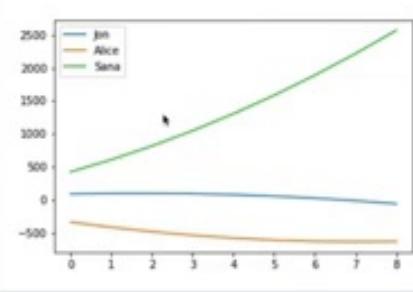
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the projected sales for three employees (Jon, Alice, and Sana) over an 8-month period. Jon's sales remain constant at 12. Alice's sales decrease from 14 to approximately -250. Sana's sales increase from 8 to approximately 2500.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 269.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

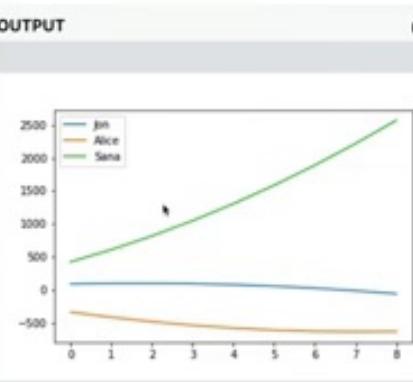
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the projected sales for three employees (Jon, Alice, and Sana) over an 8-month period. Jon's sales remain constant at 12. Alice's sales start at 14 and decrease to approximately -200 by month 8. Sana's sales start at 8 and increase to approximately 2500 by month 8.

Month	Jon	Alice	Sana
0	12	14	8
1	12	13	9.5
2	12	12	11
3	12	11	13.5
4	12	10	16
5	12	9	18.5
6	12	8	21
7	12	7	23.5
8	12	-200	2500

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 270.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

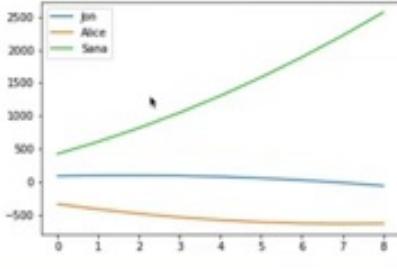
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The graph displays the sales figures for three employees (Jon, Alice, and Sana) over an 8-month period. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain relatively flat around 100. Alice's sales start at 14 and decrease steadily to approximately -300 by month 8. Sana's sales start at 8 and increase steadily to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 271.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

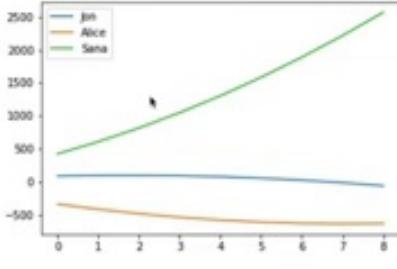
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The graph displays the sales figures for three employees (Jon, Alice, and Sana) over an 8-month period. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales remain relatively flat around 100. Alice's sales start at 14 and decrease steadily to approximately -300 by month 8. Sana's sales start at 8 and increase steadily to approximately 2200 by month 8.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 272.00 seconds**

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

A screenshot of a web-based user interface. At the top left, there is a text input field with the placeholder "Name Here...". Below the input field is a large, prominent orange button with the word "Submit" in white. To the right of the input field, there are two small buttons: "Screenshot" and "Flag". At the bottom left of the interface, there is a "Clear" button.

**Timestamp: 273.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

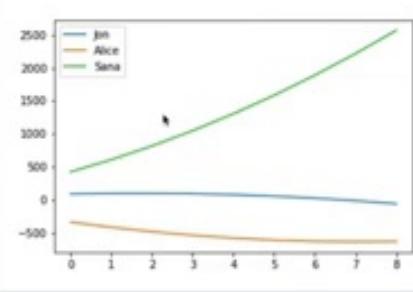
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, 100 in Feb, and 18 in Mar, showing a slight upward trend. Alice's sales (orange line) start at 14 in Jan, 17 in Feb, and 2 in Mar, showing a downward trend towards -500 by Mar. Sana's sales (green line) start at 8 in Jan, 9.5 in Feb, and 12 in Mar, showing a steady upward trend reaching approximately 2200 by Mar.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 274.00 seconds**



+ Code + Text

RAM

Disk

Editing

^

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**EMPLOYEE DATA**

**OUTPUT** 0.02s

The graph displays three lines representing the sales trajectories of three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, 100 in Feb, and 18 in Mar, showing a slight downward trend. Alice's sales (orange line) start at 14 in Jan, 17 in Feb, and 2 in Mar, showing a sharp decline to approximately -400 by Mar. Sana's sales (green line) start at 8 in Jan, 9.5 in Feb, and 12 in Mar, showing a steady increase to approximately 2200 by Mar.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 275.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

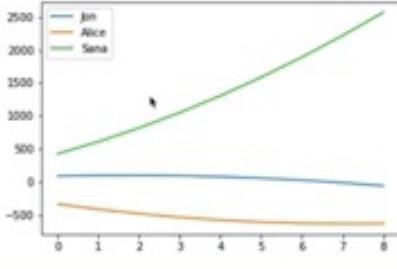
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, 100 in Feb, and 18 in Mar, showing a slight upward trend. Alice's sales (orange line) start at 14 in Jan, 17 in Feb, and 2 in Mar, showing a downward trend. Sana's sales (green line) start at 8 in Jan, 9.5 in Feb, and 12 in Mar, showing an upward trend.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 276.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

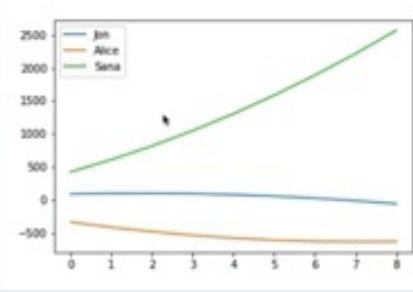
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, 100 in Feb, and 18 in Mar, showing a slight upward trend. Alice's sales (orange line) start at 14 in Jan, 17 in Feb, and 2 in Mar, showing a downward trend. Sana's sales (green line) start at 8 in Jan, 9.5 in Feb, and 12 in Mar, showing an upward trend.

Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 277.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

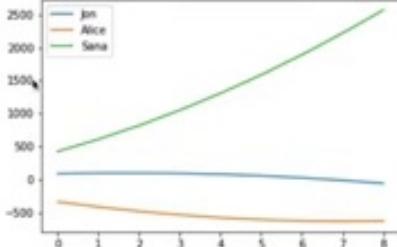
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -50 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2200 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 278.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

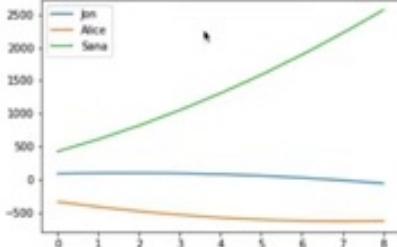
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, peak at 100 in Feb, and end at 18 in Mar. Alice's sales (orange line) start at 14 in Jan, peak at 17 in Feb, and end at 2 in Mar. Sana's sales (green line) start at 8 in Jan, peak at 9.5 in Feb, and end at 12 in Mar. All three lines show a slight downward trend from Mar to the end of the period.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 279.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

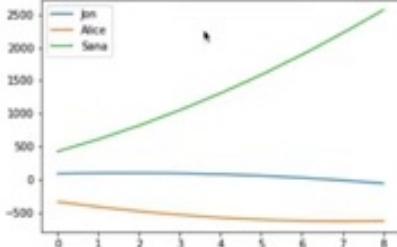
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The graph displays the sales trajectory for three employees: Jon, Alice, and Sana. The x-axis represents time in months from 0 to 8. The y-axis represents sales values ranging from -500 to 2500. Jon's sales (blue line) start at 12 in Jan, peak at 100 in Feb, and end at 18 in Mar. Alice's sales (orange line) start at 14 in Jan, peak at 17 in Feb, and end at 2 in Mar. Sana's sales (green line) start at 8 in Jan, peak at 9.5 in Feb, and end at 12 in Mar. All three lines show a slight downward trend from March to August.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 280.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

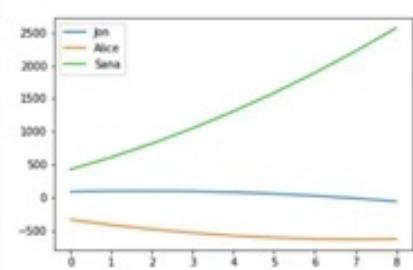
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -200 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2200 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 281.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

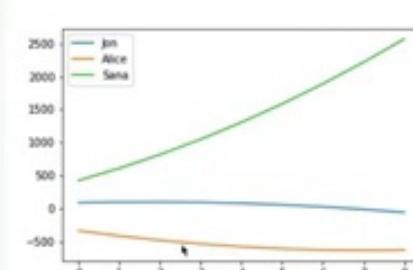
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months (M) ranging from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -50 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2200 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 282.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

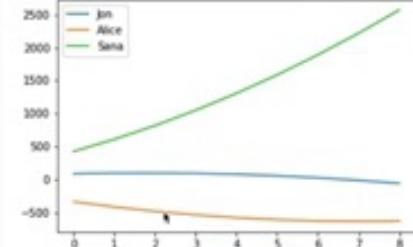
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -200 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2200 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 283.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 284.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

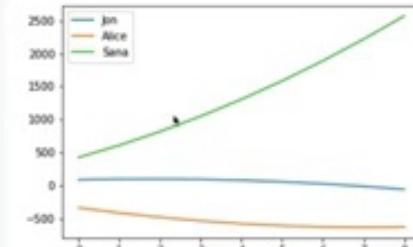
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -200 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2500 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 285.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

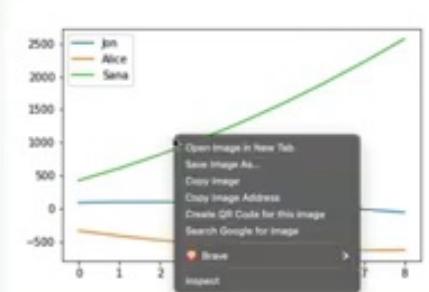
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 286.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

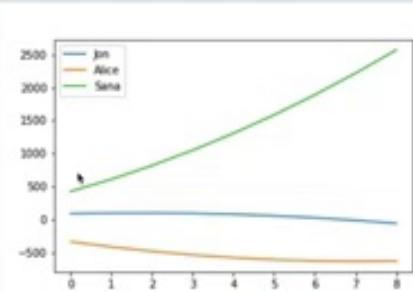
+ Code + Text

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales (blue line) start at 12 in Month 0 and decrease to approximately -100 by Month 8. Alice's sales (orange line) start at 14 in Month 0 and decrease to approximately -200 by Month 8. Sana's sales (green line) start at 8 in Month 0 and increase to approximately 2500 by Month 8.

Clear Submit

Screenshot Flag

✓ 3s completed at 5:10 PM

**Timestamp: 287.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

(x)

EMPLOYEE DATA

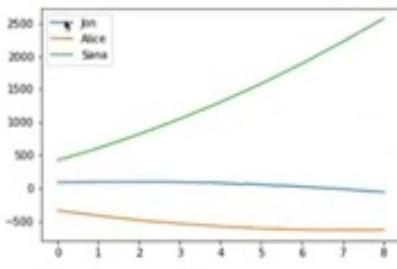
	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT

0.02s

Clear Submit

3s completed at 5:10 PM



The graph displays the projected sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (0 to 2500) and the X-axis represents Months (0 to 8). Jon's sales (blue line with square markers) start at 12 in Month 0 and rise steadily to approximately 18 in Month 8. Alice's sales (orange line with circle markers) start at 14 in Month 0 and decline slightly to about -500 in Month 8. Sana's sales (green line with triangle markers) start at 8 in Month 0 and rise sharply to approximately 2500 in Month 8.

**Timestamp: 288.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

(x) Enter sales figures for employees to predict sales trajectory over year.

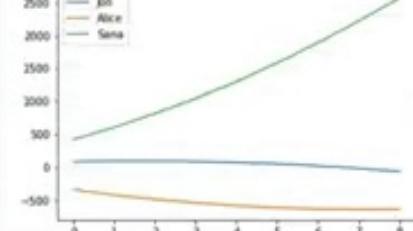
**EMPLOYEE DATA**

	Name	Jan	Sal	Feb	Sal	Mar	Sa
1	Jon	12	100	18			
2	Alice	14	17	2			
3	Sana	8	9.5	12			

**OUTPUT** 0.02s

Clear Submit

3s completed at 5:10 PM



The chart displays the sales trajectory for three employees: Jon, Alice, and Sana. The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Jon's sales increase linearly from approximately 500 in month 0 to about 2200 in month 8. Alice's sales decrease slightly from about 100 in month 0 to around -100 in month 8. Sana's sales remain relatively flat, starting at 100 and ending at 120.

Month	Jon	Alice	Sana
0	500	100	100
1	600	95	100
2	700	90	100
3	800	85	100
4	900	80	100
5	1000	75	100
6	1100	70	100
7	1200	65	100
8	1300	60	120

**Timestamp: 289.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    }
iface.launch()

Colab notebook detected. To show errors in colab notebook, set "debug=True" in `launch()`
Running on public URL: https://22580.gradio.app

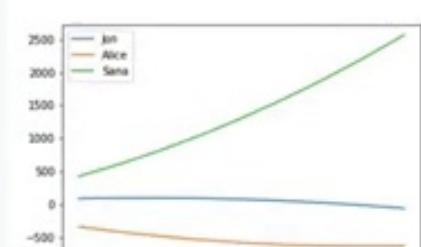
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

(x) Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit ✓ 3s completed at 5:10 PM

**Timestamp: 290.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

(x)
iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[["Jon", 12, 100, 18], ["Alice", 14, 17, 2], ["Sana", 8, 9.5, 12]]
    ),
    "plot",
),
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 291.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

```
np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 100, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

D Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 292.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

```
np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, 100, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

D Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 293.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing ^

```
month * month * regression[0] + month * regression[1] + regression[2]
for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
gr.inputs.DataFrame(
    headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
    default=[[{"Jon": 12, 100, 18}, {"Alice": 14, 17, 2}, {"Sana": 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 294.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 295.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 296.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
gr.inputs.DataFrame(
headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 297.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
gr.inputs.DataFrame(
    headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
    default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 298.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 299.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18}], [{"Alice": 14, "17, 2}, {"Sana": 8, "9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 300.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

(x)
iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

D Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 301.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

(x)
iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

RAM Disk Editing

**Timestamp: 302.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

(x)
iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

D Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 303.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 304.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 305.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 306.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 307.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18"}, {"Alice": 14, "17", "2}, {"Sana": 8, "9.5", "12}]]),
    [
        ...
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 308.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18"}, {"Alice": 14, "17", "2"}, {"Sana": 8, "9.5", "12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 309.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18"}, {"Alice": 14, "17", "2"}, {"Sana": 8, "9.5", "12}]]),
    [
        "plot",
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 310.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 311.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 312.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 313.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 314.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 315.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot",
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 316.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 317.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 318.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 319.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 320.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 321.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100": 18}, {"Alice": 14, "17": 2}, {"Sana": 8, "9.5": 12}])
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 322.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18"}, {"Alice": 14, "17", "2"}, {"Sana": 8, "9.5", "12}]]),
    [
        "plot",
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 323.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot",
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 324.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon": 12, "100", "18"}, {"Alice": 14, "17", "2"}, {"Sana": 8, "9.5", "12}]]),
    [
        "plot",
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 325.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values))
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot",
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

✓ 3s completed at 5:10 PM

**Timestamp: 326.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
return pit.gcr()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[

    "plot",
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA OUTPUT 0.02s

3s completed at 5:10 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Comment' button. Below the navigation is a toolbar with icons for 'RAM' (green checkmark), 'Disk' (grey), 'Editing' (pencil), and other system controls. The main area contains a code editor with Python code for creating a Gradio interface. A message box indicates a Colab notebook detection and provides a public URL. Below the code is a text input field with placeholder text. At the bottom, there's a progress bar showing '3s completed at 5:10 PM'.

**Timestamp: 327.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}])
),
[
    "plot",
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

Colab notebook detected. To show errors in colab notebook, set 'debug=True' in 'launch()'
Running on public URL: https://22580.gradio.app
```

✓ 3s completed at 5:10 PM

**Timestamp: 328.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 329.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
sales_data = employee_data.loc[:, "Jan":].asType("int", np, numpy())
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
```

✓ 3s completed at 5:10 PM

**Timestamp: 330.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 331.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [  
        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 332.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression = Path
    np.arr() pd
projected_ = PIL
    np.arr() processing_utils
projected_ = Radio
    month = Slider
        for mo () tempfile
    plt.plot(p() test_data
plt.legend(Textbox)
    def __init__(lines=1, placeholder=None)
return plt = Timeseries
    Video
iface = gr.Int()
gr.inputs([
    headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
    default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]],
),
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 333.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression = Audio
np.arr = Checkbox
projected_ = CheckboxGroup
np.ara = Component
projected_ = Dataframe
month = Dropdown
for mo = FFmpeg
plt.plot(p = File
plt.legend = get_input_instance
return plt = Image
= InputComponent
iface = gr.Int() json
gr.inputs. = [
    headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
    default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]],
],
[
    "plot"
],
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 334.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.Audio(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 335.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.Audio(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 336.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 337.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 338.00 seconds**



+ Code + Text

RAM Disk Editing

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 339.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 340.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 341.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 342.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs([
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 343.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs([  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ],  
    "plot",  
    ),  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 344.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs([  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ],  
    "plot",  
    ),  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 345.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()
```

```
iface = gr.Interface(sales_projections,  
    gr.inputs([  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ],  
    "plot",  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 346.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs([  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ],  
    "plot",  
    ),  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 347.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [  
        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 348.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=[ Dataframe gr.inputs.DataFrame(headers=None, row_=  
            default=[ () test_data , 8, 9.5, 12]]  
    ),  
    [  
        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 349.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Name": "Jon", "Jan Sales": 12, "Feb Sales": 100, "Mar Sales": 18}, {"Name": "Alice", "Jan Sales": 14, "Feb Sales": 17, "Mar Sales": 2}, {"Name": "Sana", "Jan Sales": 8, "Feb Sales": 9.5, "Mar Sales": 12}]),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 350.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME

Name Here...

Screenshot

Flag

**Timestamp: 351.00 seconds**



+ Code + Text

RAM Disk Editing

```
regression_values = np.apply_along_axis(lambda row:
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(3,12), 0), len(sales_data), axis=0)
projected_values = np.array([
    month * month * regression[0] + month * regression[1] + regression[2]
    for month, regression in zip(projected_months, regression_values)])
plt.plot(projected_values.T)
plt.legend(employee_data["Name"])
return plt.gcf()

iface = gr.Interface(sales_projections,
    gr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]),
    [
        "plot"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 352.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 353.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 354.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 355.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 356.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
regression_values = np.apply_along_axis(lambda row:  
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)  
projected_months = np.repeat(np.expand_dims(  
    np.arange(3,12), 0), len(sales_data), axis=0)  
projected_values = np.array([  
    month * month * regression[0] + month * regression[1] + regression[2]  
    for month, regression in zip(projected_months, regression_values)])  
plt.plot(projected_values.T)  
plt.legend(employee_data["Name"])  
return plt.gcf()  
  
iface = gr.Interface(sales_projections,  
    gr.inputs.DataFrame(  
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],  
        default=[[{"Jon", 12, 100, 18}, {"Alice", 14, 17, 2}, {"Sana", 8, 9.5, 12}]]  
    ),  
    [br/>        "plot"  
    ],  
    description="Enter sales figures for employees to predict sales trajectory over year."  
)  
iface.launch()
```

✓ 3s completed at 5:10 PM

**Timestamp: 357.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
iiface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

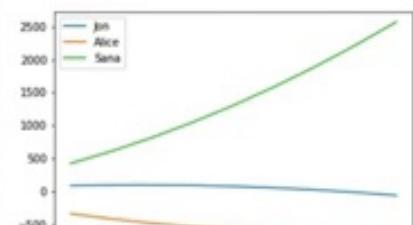
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 358.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 359.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 360.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

↳ Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://22580.gradio.app

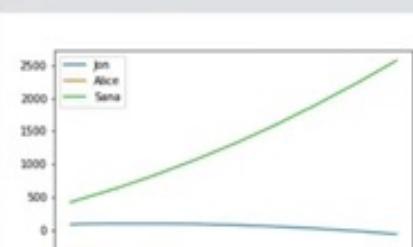
(x) This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.
```

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit ✓ 3s completed at 5:10 PM

**Timestamp: 361.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 362.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

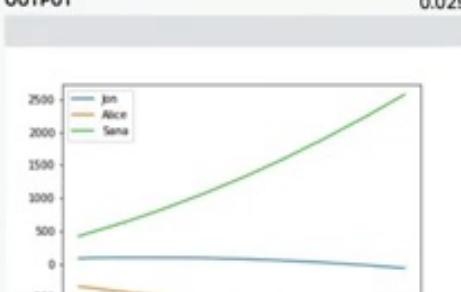
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 363.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 364.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

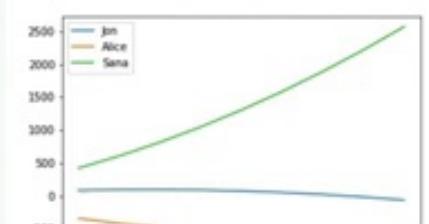
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 365.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

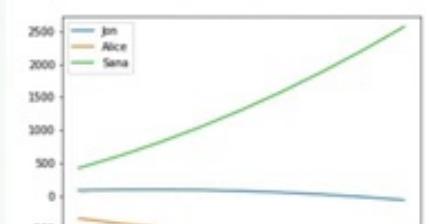
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 366.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

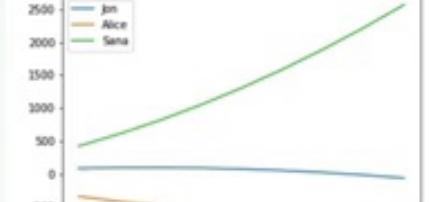
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 367.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

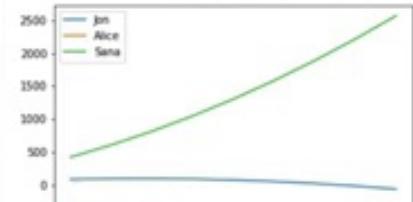
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 368.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

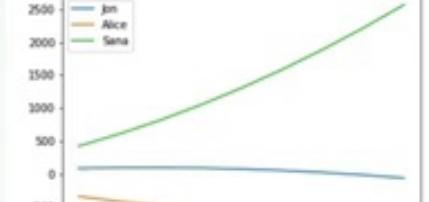
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 369.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

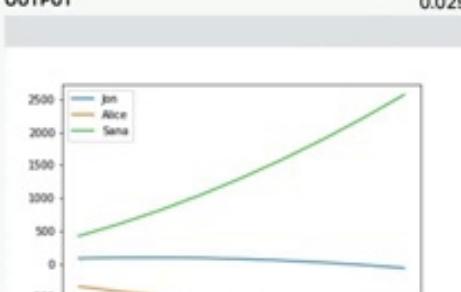
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 370.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

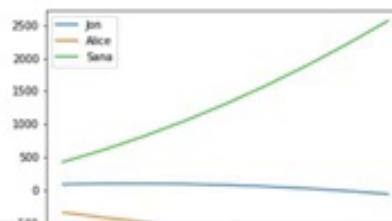
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 371.00 seconds**



+ Code + Text

RAM

Disk

Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

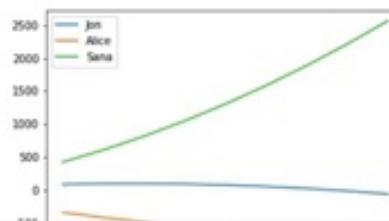
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 372.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 373.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

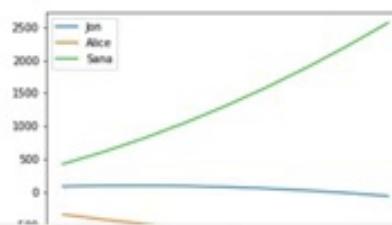
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 374.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

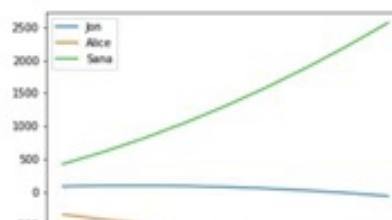
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA				
	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

## OUTPUT

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 375.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

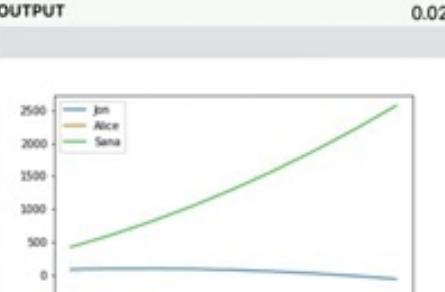
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 376.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

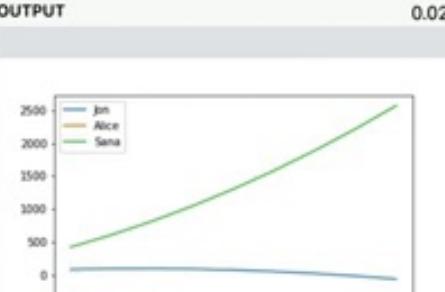
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 377.00 seconds**



+ Code + Text

RAM Disk Editing

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

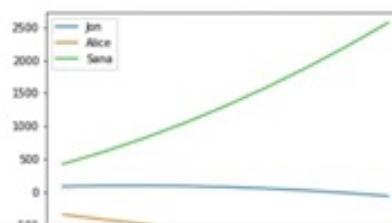
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 378.00 seconds**



+ Code + Text

RAM Disk Editing

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

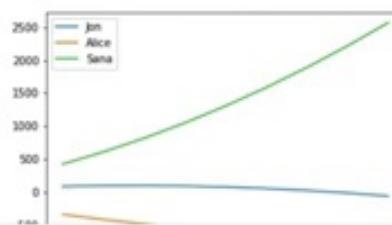
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 379.00 seconds**



+ Code + Text

RAM Disk Editing

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

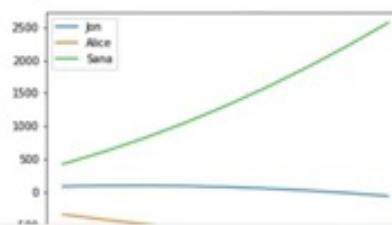
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM

**Timestamp: 380.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

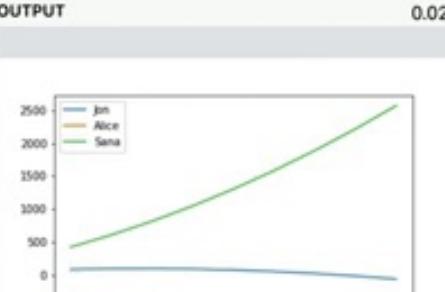
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 381.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

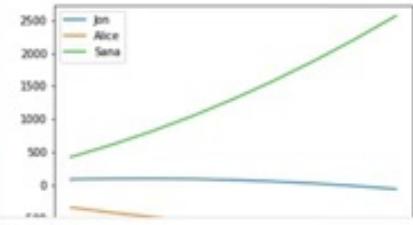
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 382.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

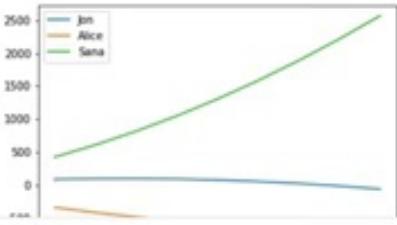
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 383.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME

Name Here...

Screenshot

Flag

**Timestamp: 384.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

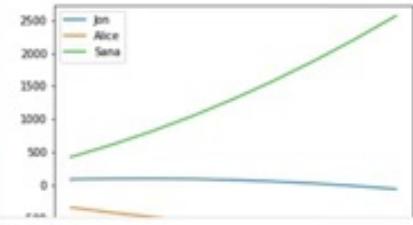
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 385.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

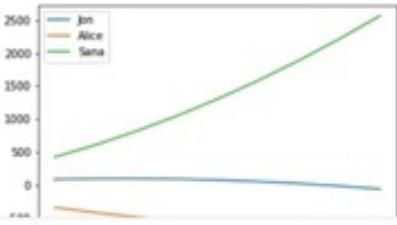
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 386.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

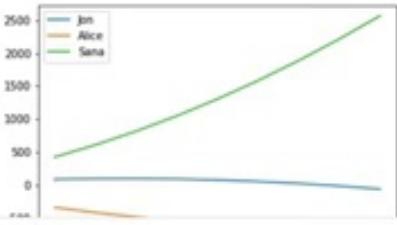
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

OUTPUT 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 387.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 388.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 389.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

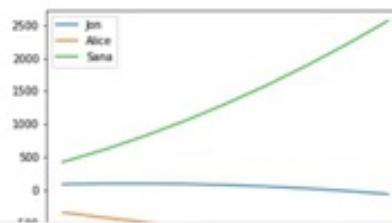
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 390.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

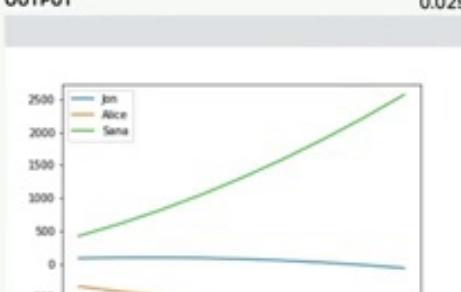
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 391.00 seconds**



+ Code + Text

RAM Disk Editing

```
        description="Enter sales figures for employees to predict sales trajectory over year."
    )
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

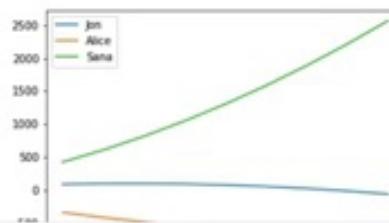
Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT**

0.02s



Clear

Submit

✓ 3s completed at 5:10 PM



**Timestamp: 392.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 393.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 394.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 395.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 396.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

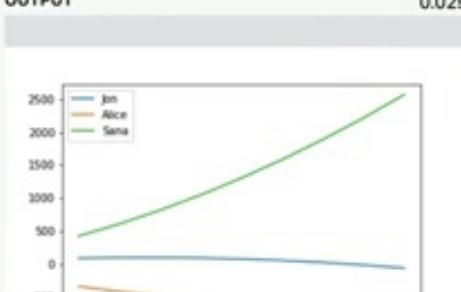
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 397.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()
```

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22580.gradio.app>

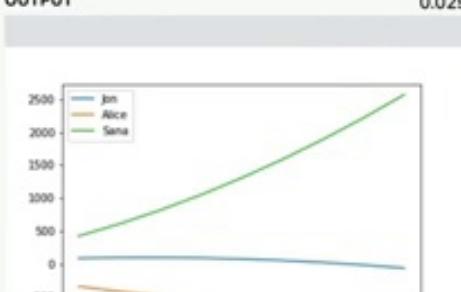
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Enter sales figures for employees to predict sales trajectory over year.

**EMPLOYEE DATA**

	Name	Jan Sal	Feb Sal	Mar Sal
1	Jon	12	100	18
2	Alice	14	17	2
3	Sana	8	9.5	12

**OUTPUT** 0.02s



Clear Submit

✓ 3s completed at 5:10 PM

**Timestamp: 398.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 399.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

**Timestamp: 400.00 seconds**

```
iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The screenshot shows a simple UI component. It has a text input field with the placeholder "Type your name here". Below the input is a yellow "Submit" button. To the right of the input field are two buttons: "Screenshot" and "Flag".

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr
```

**Timestamp: 401.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or open in a browser on <http://localhost:7860> if running from a script.

The image shows a screenshot of a Gradio interface. At the top, there is a text input field with the placeholder 'NAME' and the value 'I'. Below it are two buttons: 'Clear' and a yellow 'Submit' button. At the top right, there are three buttons: 'Screenshot', 'Flag', and 'NAME'. The 'Screenshot' button is grey, while the other two are white with black text.

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 402.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or open it in a browser on <http://localhost:7860> if running from a script.

The image shows a simple web-based user interface. At the top, there is a text input field with the placeholder 'NAME' and the value 'I' entered. Below the input field are two buttons: 'Clear' on the left and a large yellow 'Submit' button on the right. To the right of the 'Submit' button are two small links: 'Screenshot' and 'Flag'. The entire interface is contained within a light gray box.

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 403.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or open in a browser on <http://localhost:7860> if running from a script.

The interface shows a single input field labeled "NAME" with the value "I". Below the input field are three buttons: "Clear", "Submit" (highlighted in orange), and "Screenshot".

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 404.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The interface shows a single text input field with the placeholder 'NAME'. The input field contains the letter 'I'. Below the input field are two buttons: 'Clear' on the left and a yellow 'Submit' button on the right. At the top right of the interface, there are three additional buttons: 'Screenshot', 'Flag', and another button that appears to be partially cut off.

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 405.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The image shows a simple web-based form interface. At the top, it says "NAME". Below that is a text input field containing the letter "I". Underneath the input field are two buttons: "Clear" on the left and "Submit" on the right, which is highlighted with an orange background. To the right of the "Submit" button are two tabs: "Screenshot" and "Flag".

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 406.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The image shows a simple web-based user interface. At the top, there is a text input field with the placeholder 'NAME' and the value 'I'. Below the input field are two buttons: 'Clear' on the left and 'Submit' on the right, which is highlighted with an orange background. To the right of the input field are two tabs: 'Screenshot' and 'Flag'. The overall design is clean and minimalist.

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 407.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The image shows a simple web-based form. At the top, it says "NAME". Below that is a text input field containing the letter "I". Underneath the input field are two buttons: "Clear" on the left and "Submit" on the right, which is highlighted with an orange background. To the right of the "Submit" button are two tabs: "Screenshot" and "Flag".

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 408.00 seconds**

```
import gradio as gr

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

iface = gr.Interface(fn=greet, inputs="text", outputs="text")
iface.launch()
```

3. The interface below will appear automatically within the Python notebook, or pop in a browser on <http://localhost:7860> if running from a script.

The interface shows a single text input field with the placeholder 'NAME'. The input field contains the letter 'I'. Below the input field are two buttons: 'Clear' (gray) and 'Submit' (yellow). To the right of the input field are three tabs: 'Screenshot' (gray), 'Flag' (gray), and another 'Screenshot' tab (gray).

### The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

**Timestamp: 409.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 410.00 seconds**

The screenshot shows a simple web-based interface. At the top, there is a text input field labeled "NAME". Below it is a large orange "Submit" button. To the left of the submit button is a "Clear" button. To the right are two buttons: "Screenshot" and "Flag".

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
```

**Timestamp: 411.00 seconds**

- **inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface, there is a 'Clear' button.

### Multiple Inputs and Outputs

**Timestamp: 412.00 seconds**

- **inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text is 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two small rectangular buttons: one labeled 'Screenshot' and another labeled 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 413.00 seconds**

- **inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text is 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two small rectangular buttons: one labeled 'Screenshot' and another labeled 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 414.00 seconds**

- **inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text is 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two small rectangular buttons: one labeled 'Screenshot' and another labeled 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 415.00 seconds**

- **inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and launch() them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual Input class for TextBox instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Clear      Submit

Screenshot      Flag

### Multiple Inputs and Outputs

**Timestamp: 416.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME' containing the placeholder 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface is a 'Clear' button.

### Multiple Inputs and Outputs

**Timestamp: 417.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME' containing the placeholder 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface is a 'Clear' button.

### Multiple Inputs and Outputs

**Timestamp: 418.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME' containing the placeholder 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface is a 'Clear' button.

### Multiple Inputs and Outputs

**Timestamp: 419.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a yellow 'Submit' button. At the bottom left of the interface is a 'Clear' button. At the top right are two buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 420.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 421.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME' containing the placeholder 'Name Here...'. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface is a 'Clear' button.

### Multiple Inputs and Outputs

**Timestamp: 422.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 423.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 424.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 425.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 426.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 427.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large, prominent orange button labeled 'Submit'. To the right of the input field, there are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 428.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field. The input field is a `Textbox` component with the placeholder text "Name Here...". Below the input field is a large orange button labeled "Submit". There are also "Clear" and "Screenshot" buttons. To the right of the input field are "Screenshot" and "Flag" buttons.

### Multiple Inputs and Outputs

**Timestamp: 429.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 430.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. At the bottom of the interface, there are two buttons: 'Clear' on the left and 'Submit' on the right.

### Multiple Inputs and Outputs

**Timestamp: 431.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 432.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 433.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. At the bottom of the interface, there are two buttons: 'Clear' on the left and 'Submit' on the right.

### Multiple Inputs and Outputs

**Timestamp: 434.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. There are also 'Clear' and 'Screenshot' buttons.

### Multiple Inputs and Outputs

**Timestamp: 435.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 436.00 seconds**

- **Inputs:** the input component type(s)
- **outputs:** the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the input field. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two small buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

**Timestamp: 437.00 seconds**

- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual `Input` class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.Input.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text in the field is 'Name Here...'. Below the input field is a large orange 'Submit' button. To the left of the input field is a 'Clear' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'.

### Multiple Inputs and Outputs

Let's say we had a much more complex function, with multiple inputs and outputs. In the example below, we have a function that takes a string, boolean, and number, and returns a string

**Timestamp: 438.00 seconds**

- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and launch() them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for TextBox instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.Inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the field. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface, there is a 'Clear' button.

### Multiple Inputs and Outputs

Let's say we had a much more complex function, with multiple inputs and outputs. In the example below, we have a function that takes a string, boolean, and number, and returns a string

**Timestamp: 439.00 seconds**

- outputs: the output component type(s)

With these three arguments, we can quickly create interfaces and launch() them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for TextBox instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.Inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible inside the field. Below the input field is a large orange 'Submit' button. To the right of the input field are two smaller buttons: 'Screenshot' and 'Flag'. At the bottom left of the interface, there is a 'Clear' button.

### Multiple Inputs and Outputs

Let's say we had a much more complex function, with multiple inputs and outputs. In the example below, we have a function that takes a string, boolean, and number, and returns a string

**Timestamp: 440.00 seconds**

- outputs: the output component type(s)

With these three arguments, we can quickly create interfaces and launch() them. But what if you want to change how the UI components look or behave?

### Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for TextBox instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

The screenshot shows a Gradio interface with a single input field labeled 'NAME'. The placeholder text 'Name Here...' is visible. Below the input field are three buttons: 'Clear' (gray), 'Submit' (orange), and 'Screenshot' (gray). To the right of the 'Submit' button is a 'Flag' button.

### Multiple Inputs and Outputs

Let's say we had a much more complex function, with multiple inputs and outputs. In the example below, we have a function that takes a string, boolean, and number, and returns a string

**Timestamp: 441.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[1] ! pip install gradio
[2] import gradio as gr
[x]
[5] def hello_world(name):
    return "Hello.... 🎉 ..."+name + "!!!!"
[6] hello_world("llittlecoder")
'Hello.... 🎉 ...llittlecoder!!!!'
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
[10] interface.launch()
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://31668.gradio.app
```

✓ 2s completed at 4:47 PM

**Timestamp: 442.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

[?] interface = gradio.interface.launch() - hello\_world, inputs = [text], outputs = [text]

interface.launch()

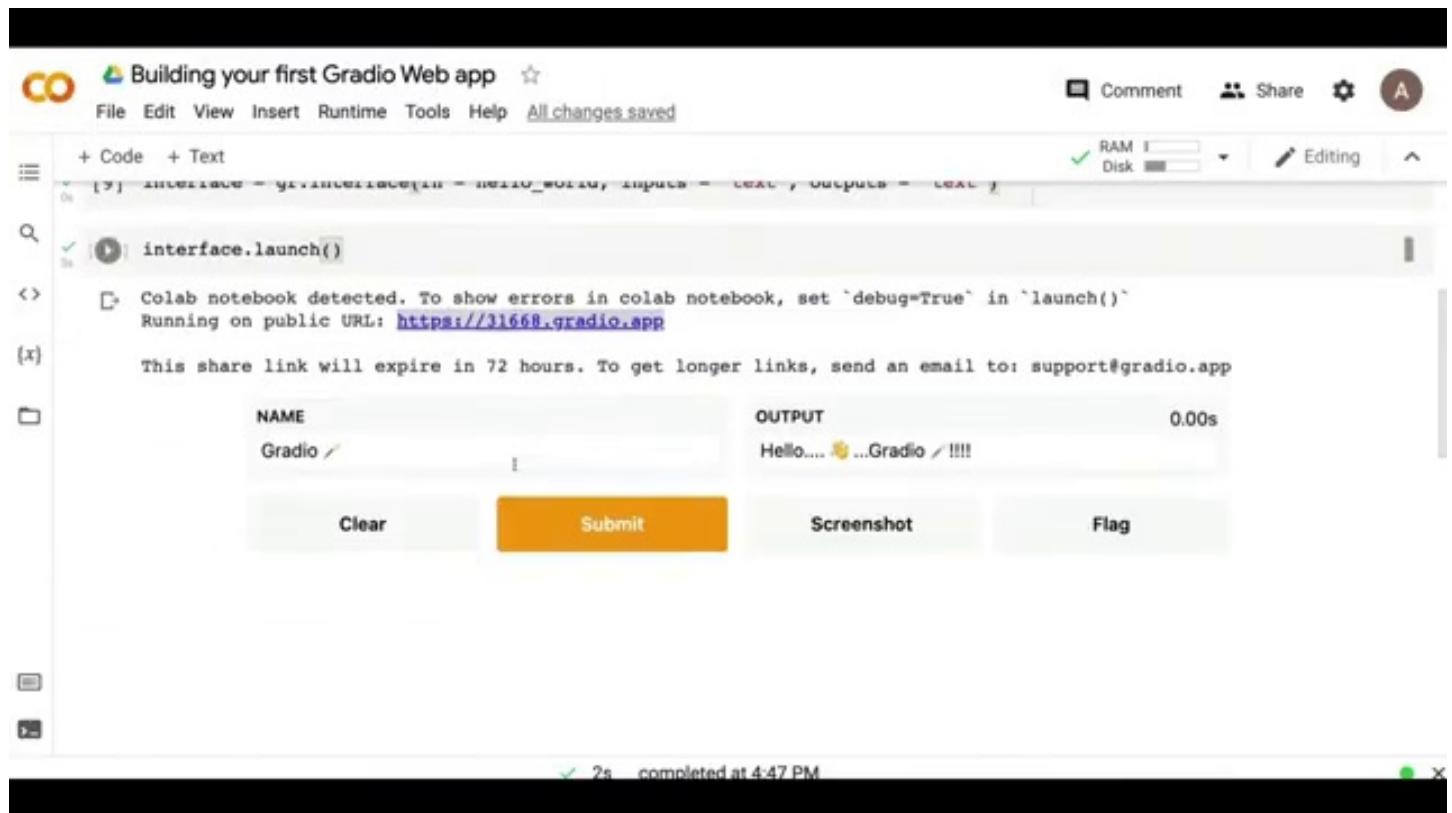
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio ✓ !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 443.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 444.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

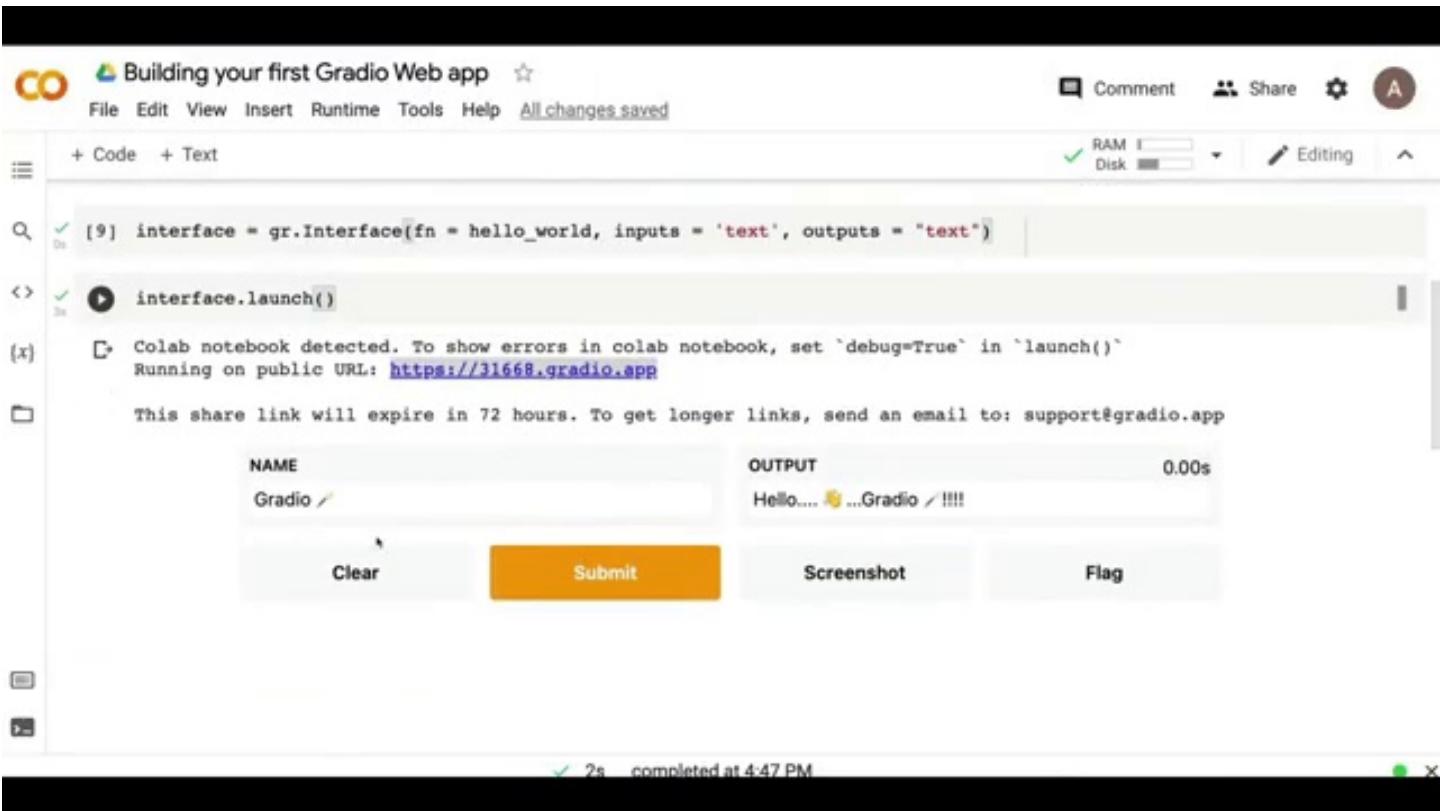
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio ✓	Hello.... 🎉 ...Gradio ✓ !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 445.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

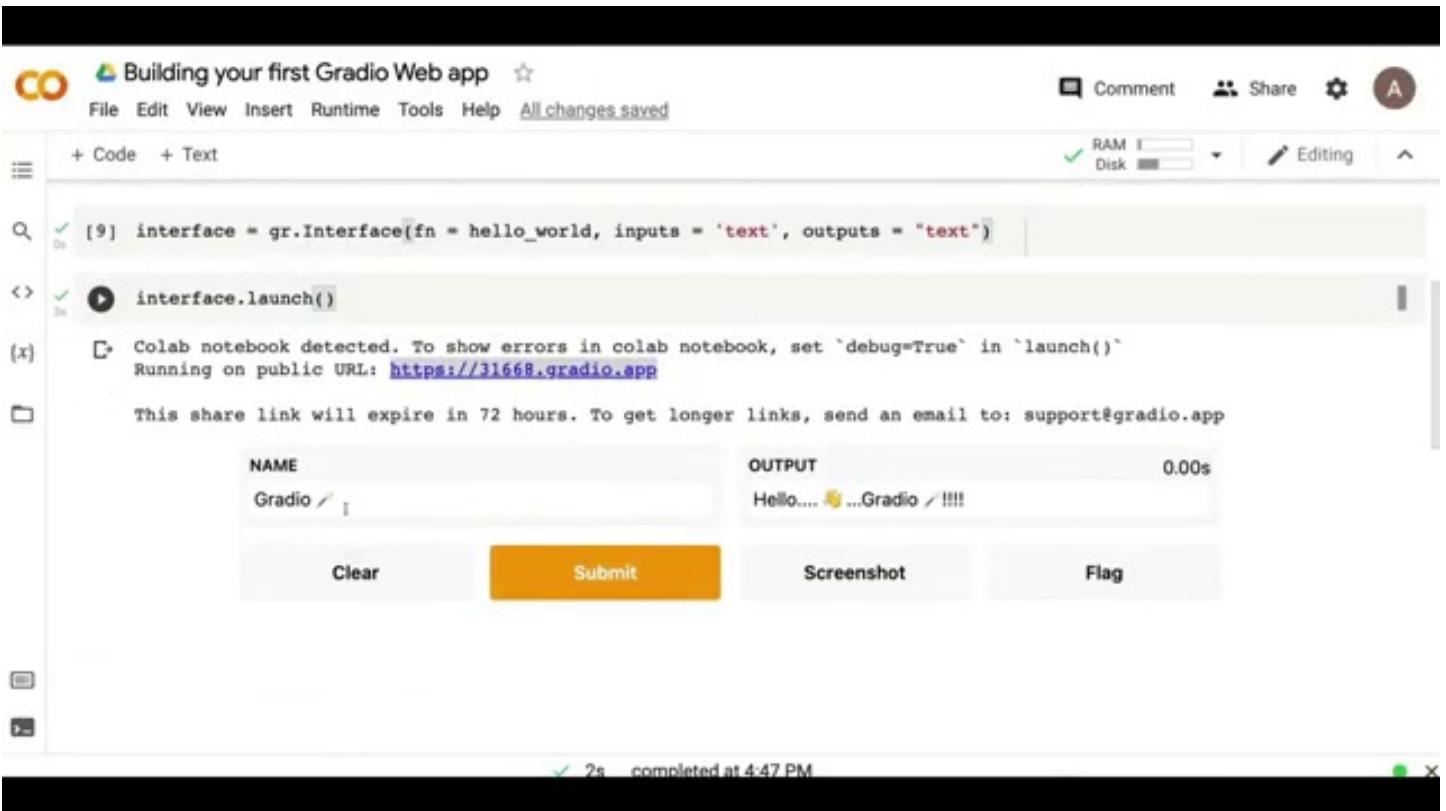
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 446.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

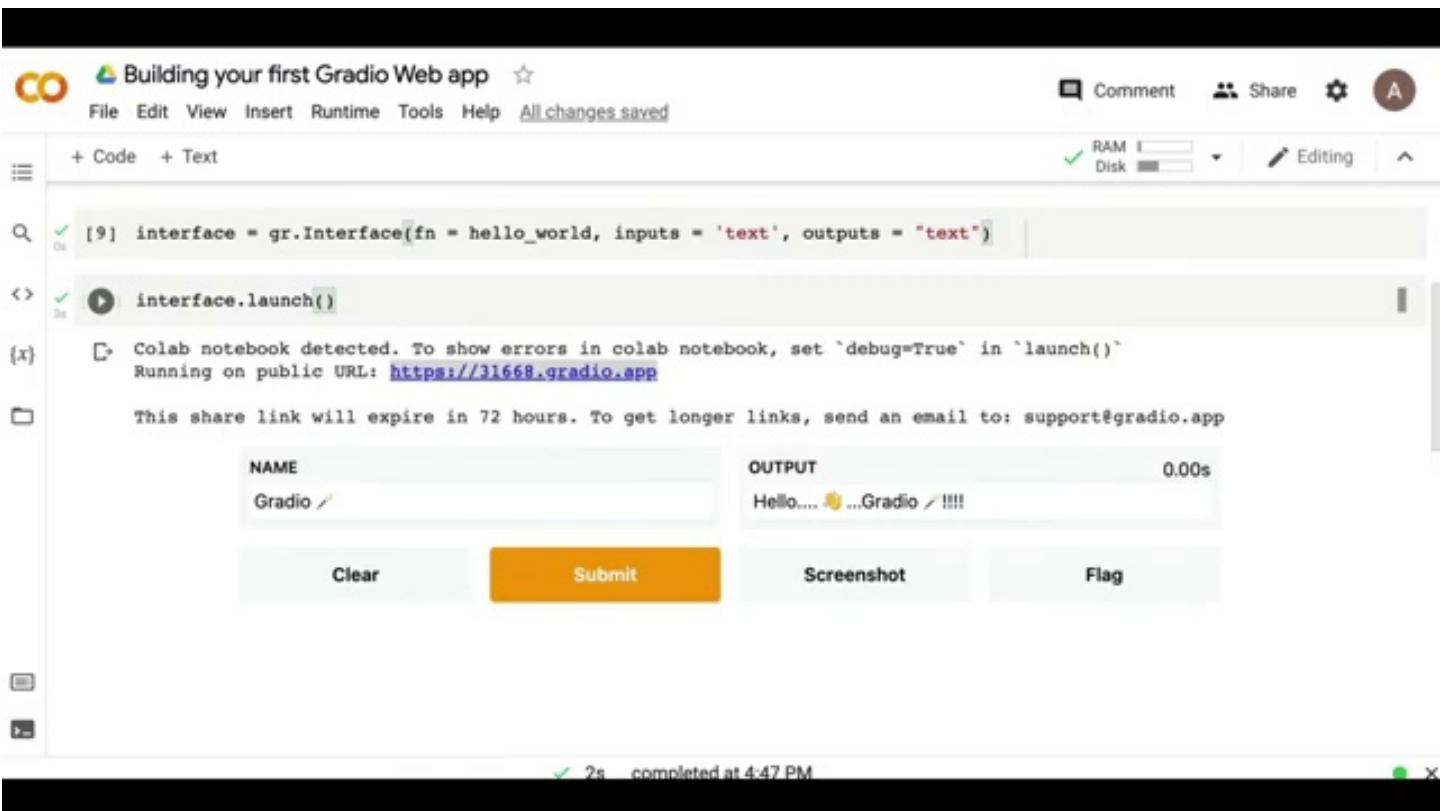
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 447.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
[10] interface.launch()
```

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

**Timestamp: 448.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
[10] interface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://31668.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

**Timestamp: 449.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

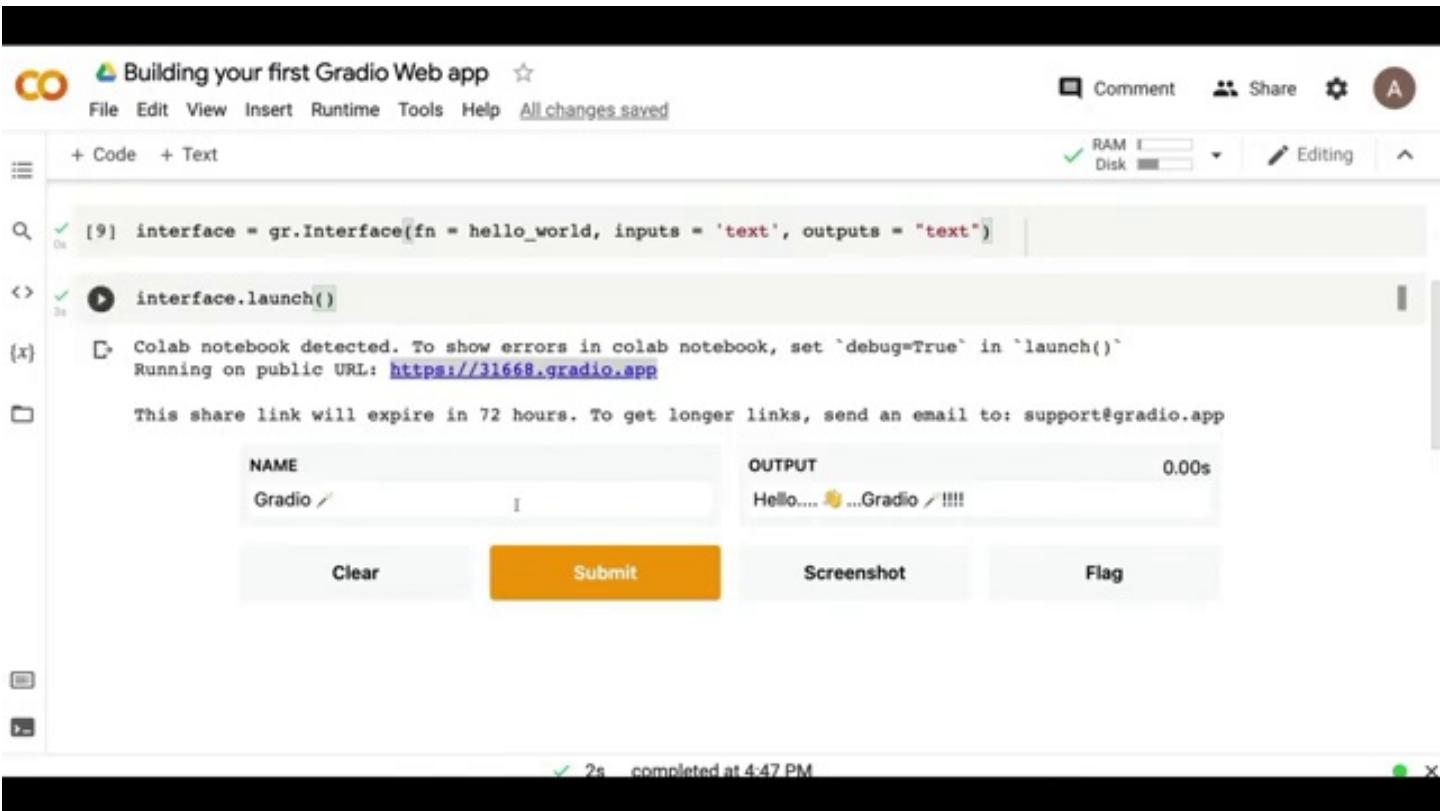
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio	!!!!

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 450.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

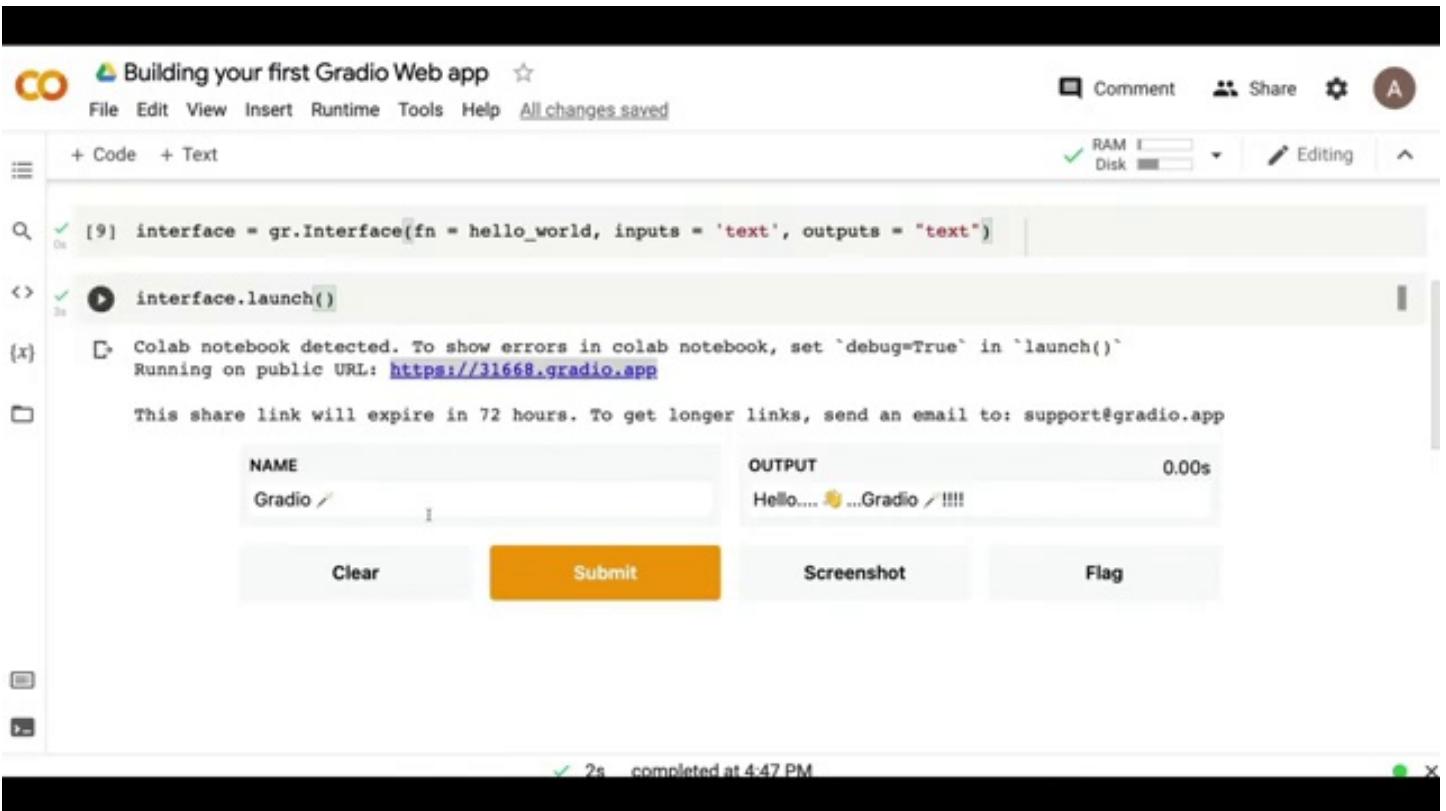
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 451.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

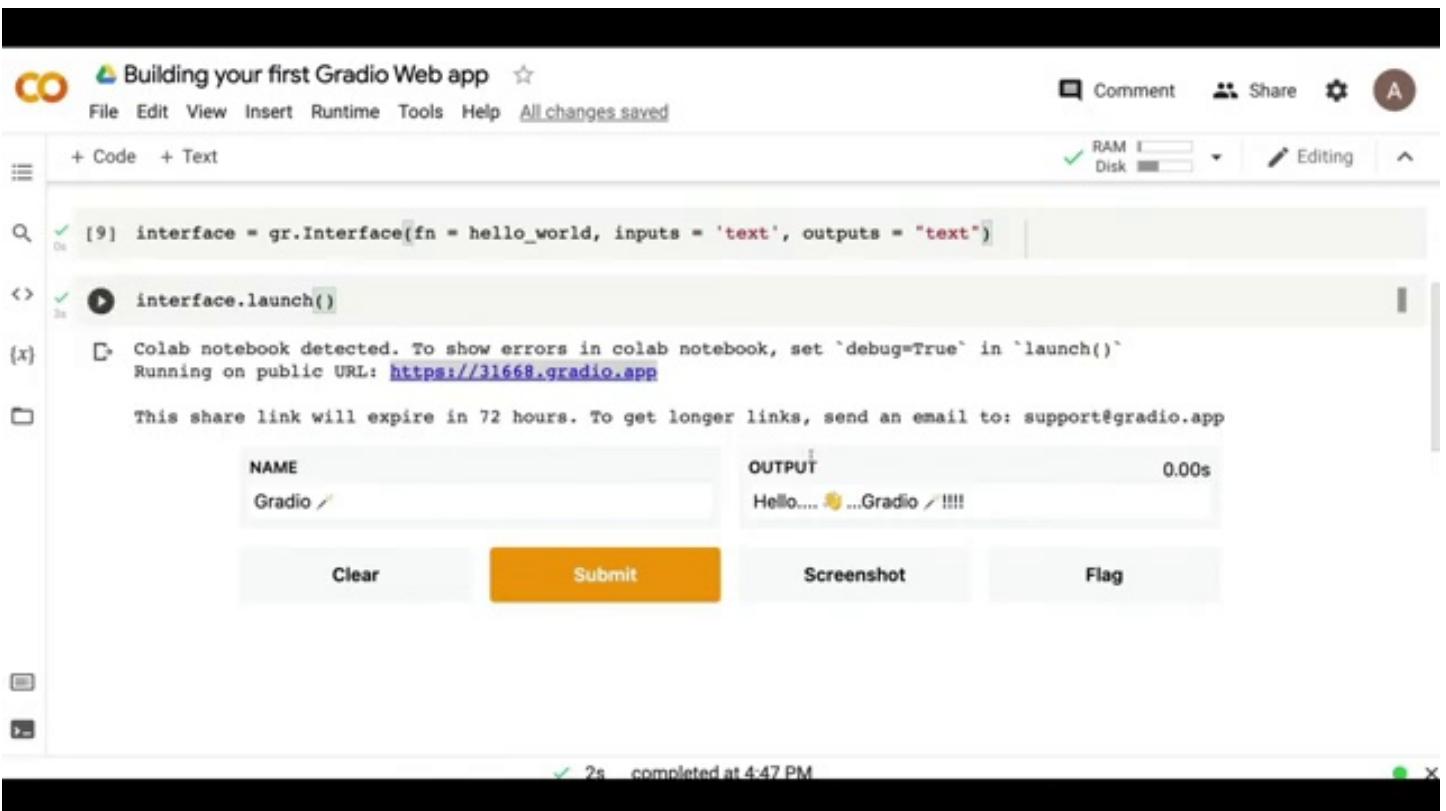
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 452.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

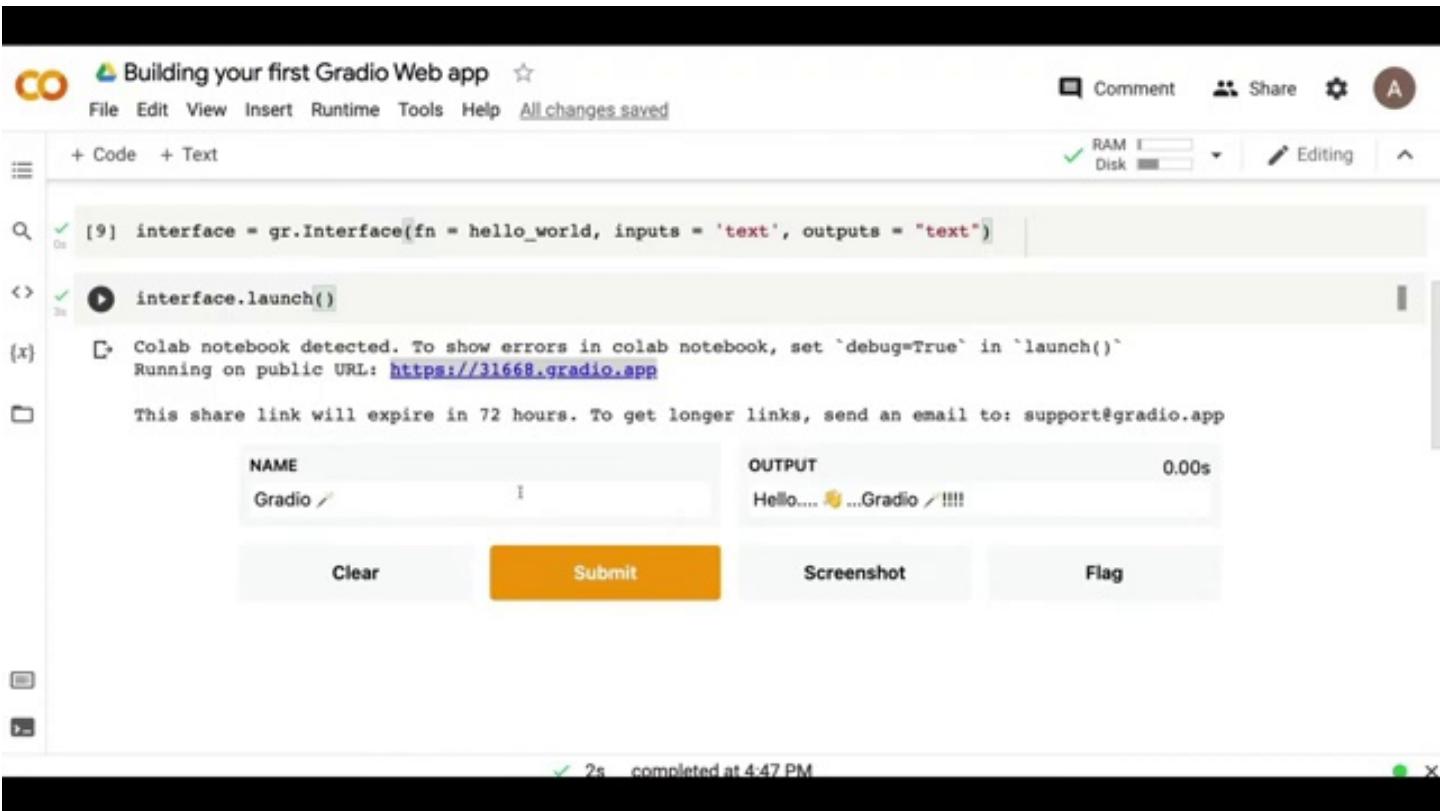
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	TIME
Gradio	Hello.... 🎉 ...Gradio !!!!!	0.00s

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 453.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

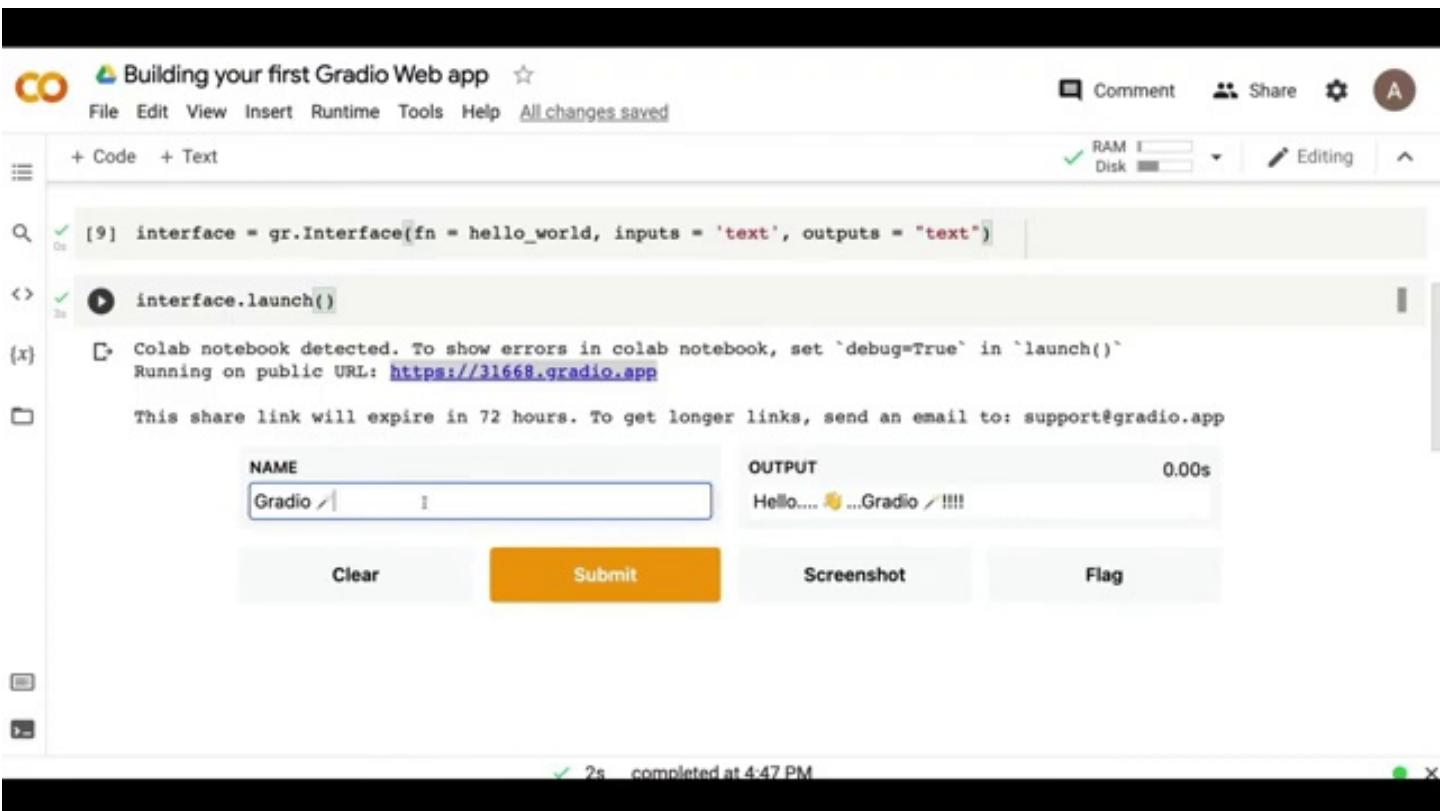
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 454.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 455.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

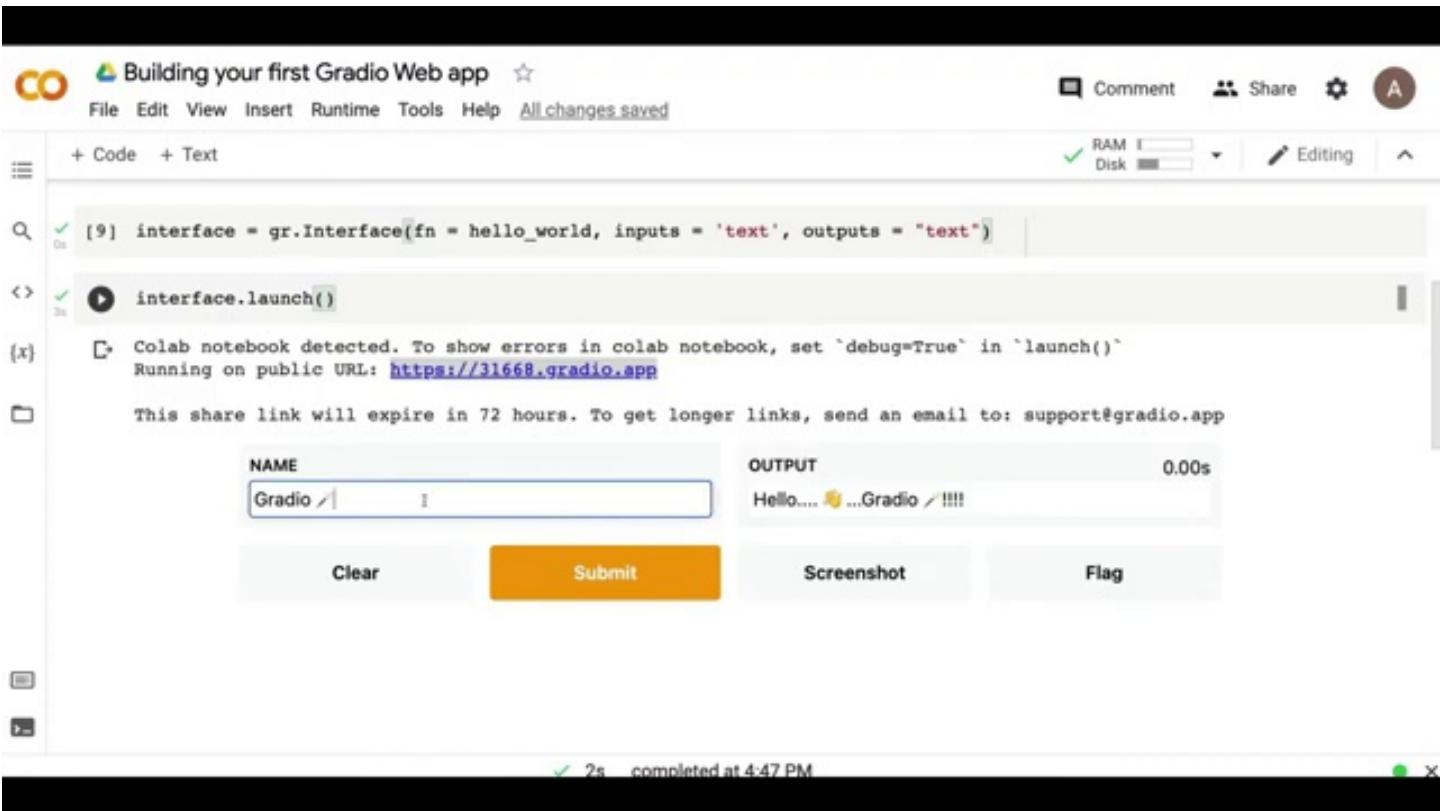
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 456.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

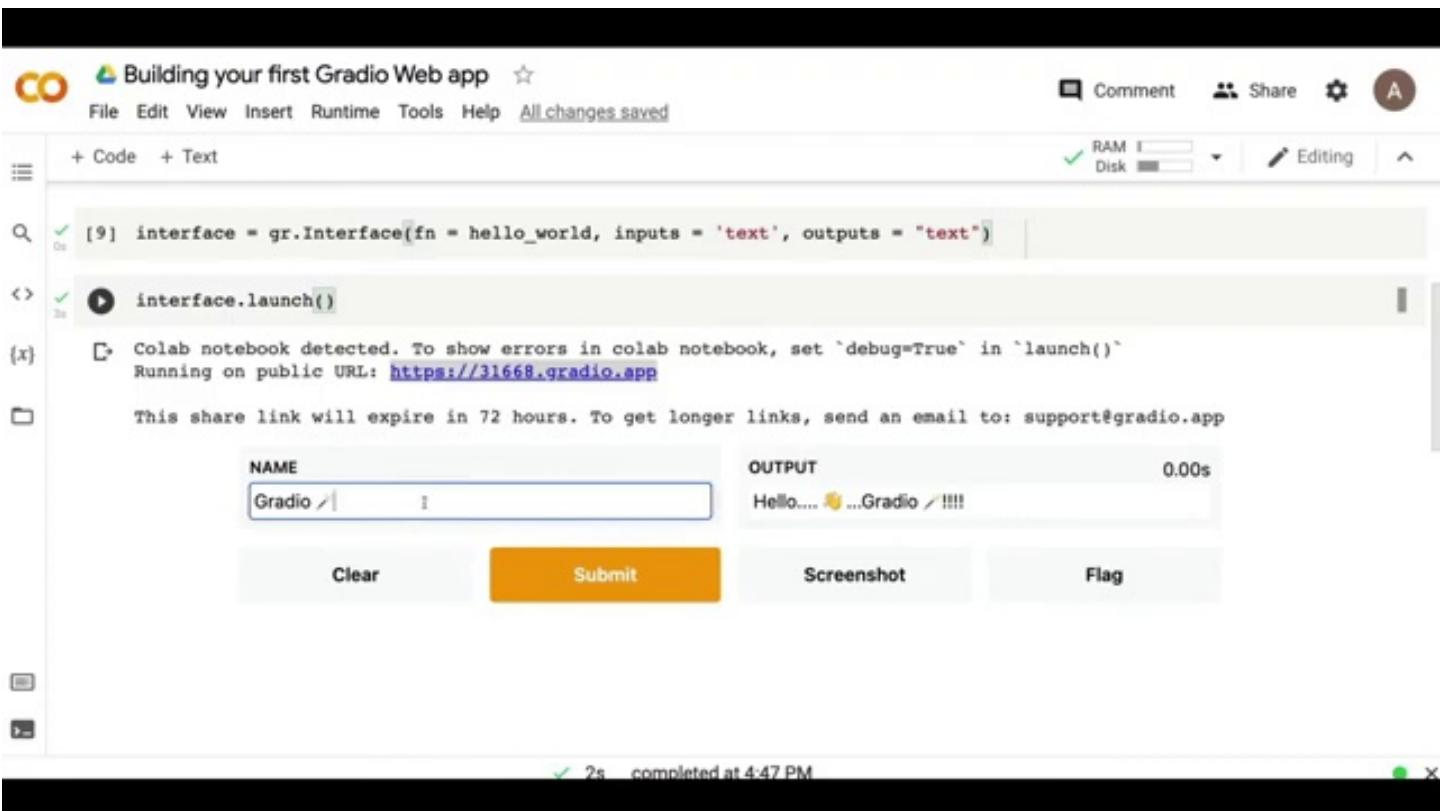
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 457.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

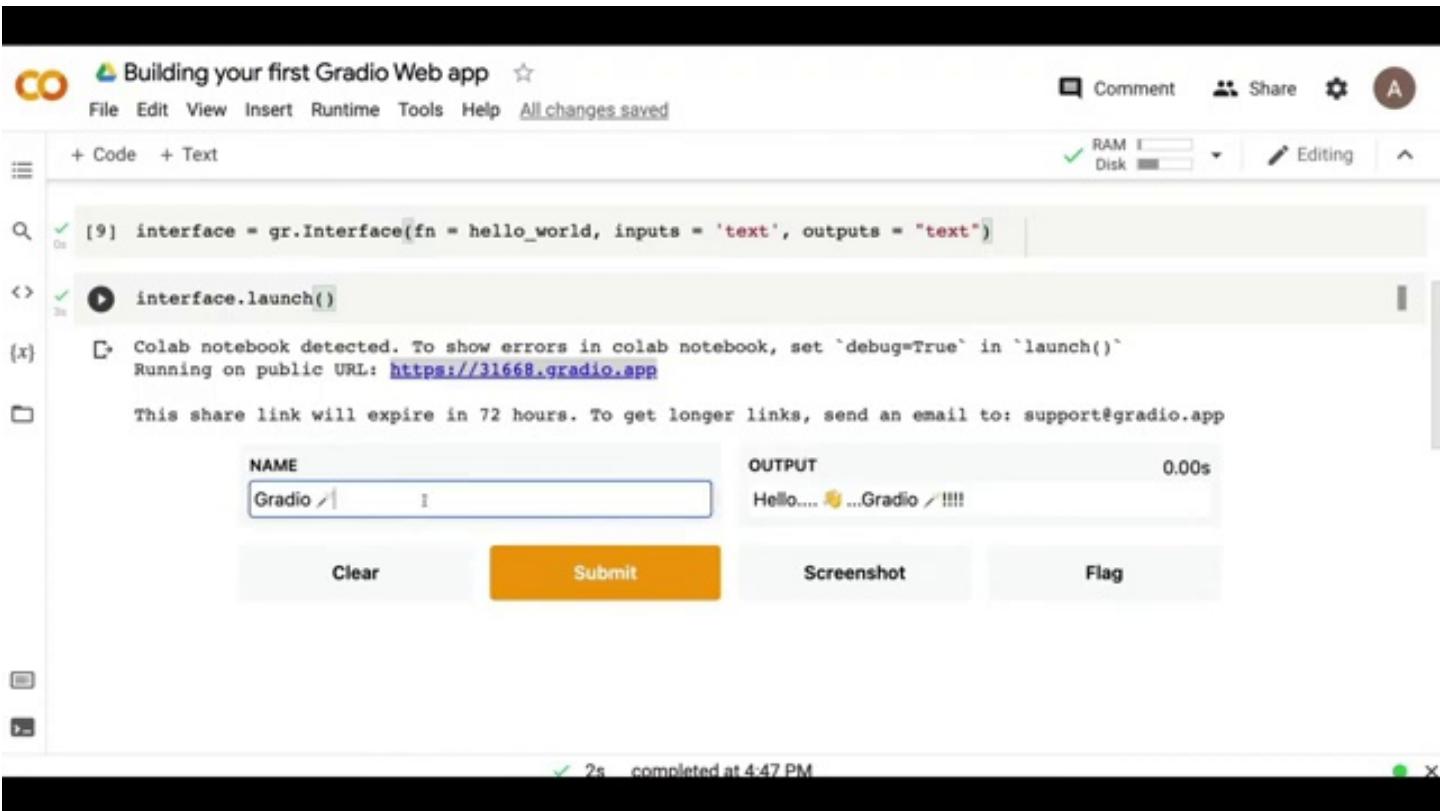
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 458.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

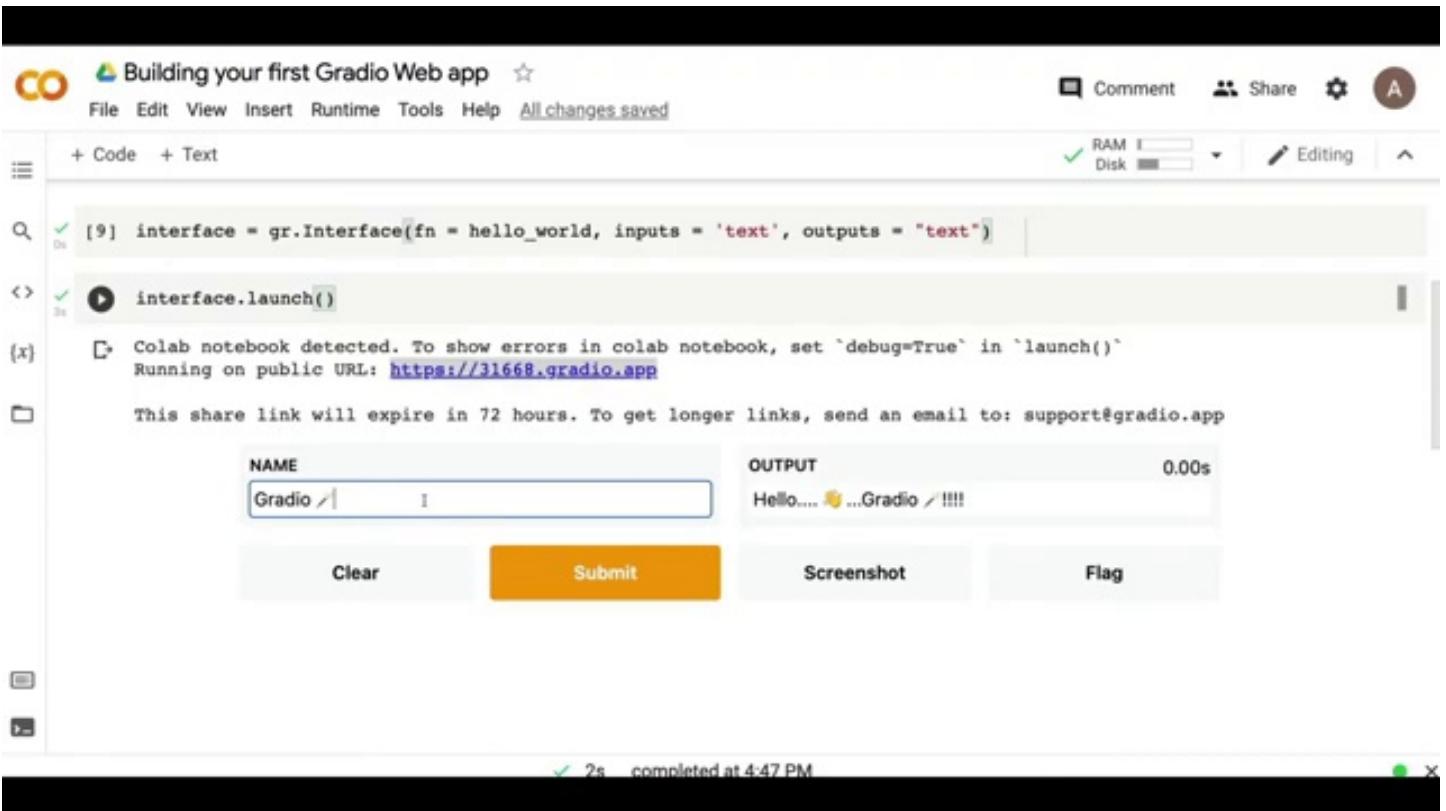
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 459.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

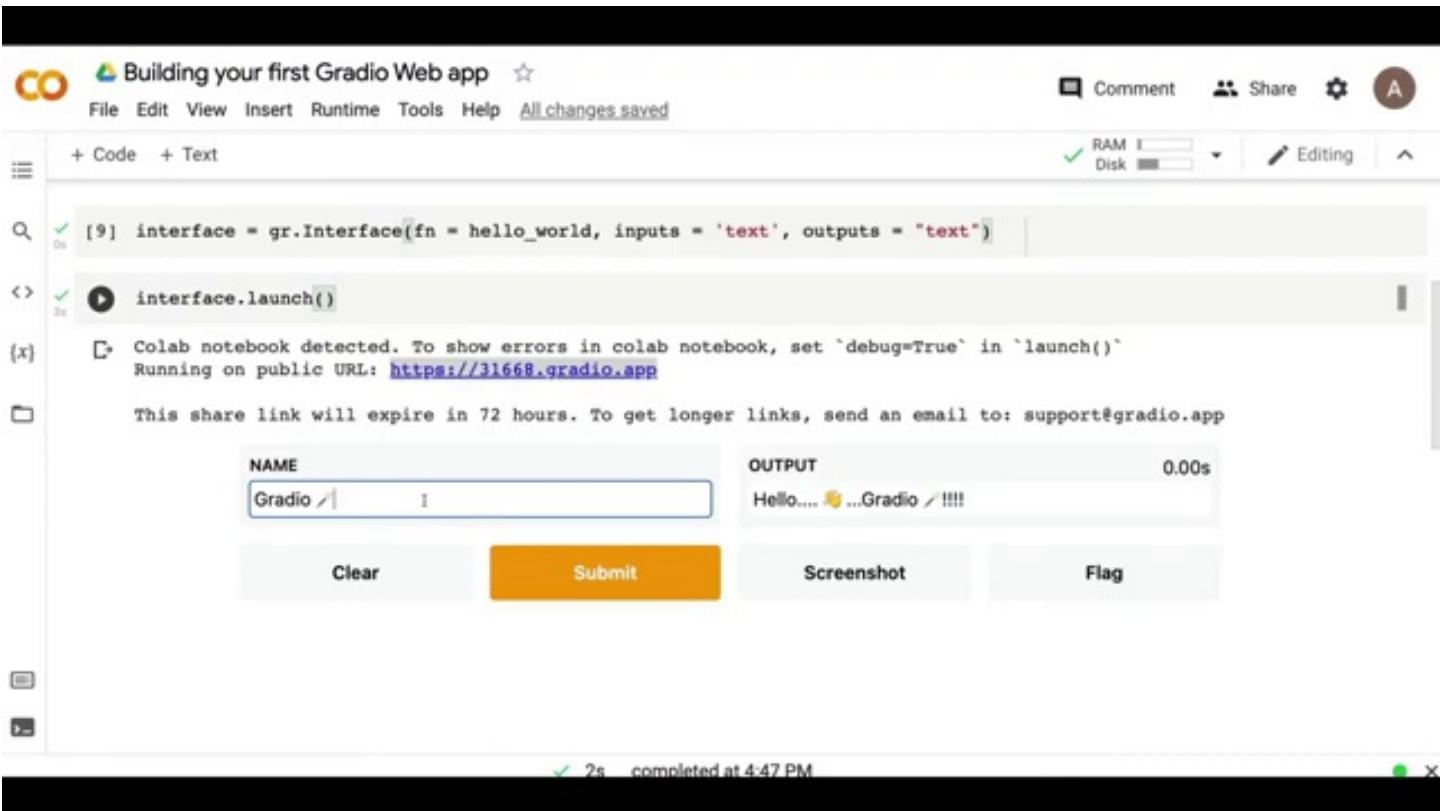
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 460.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

```
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

interface.launch()

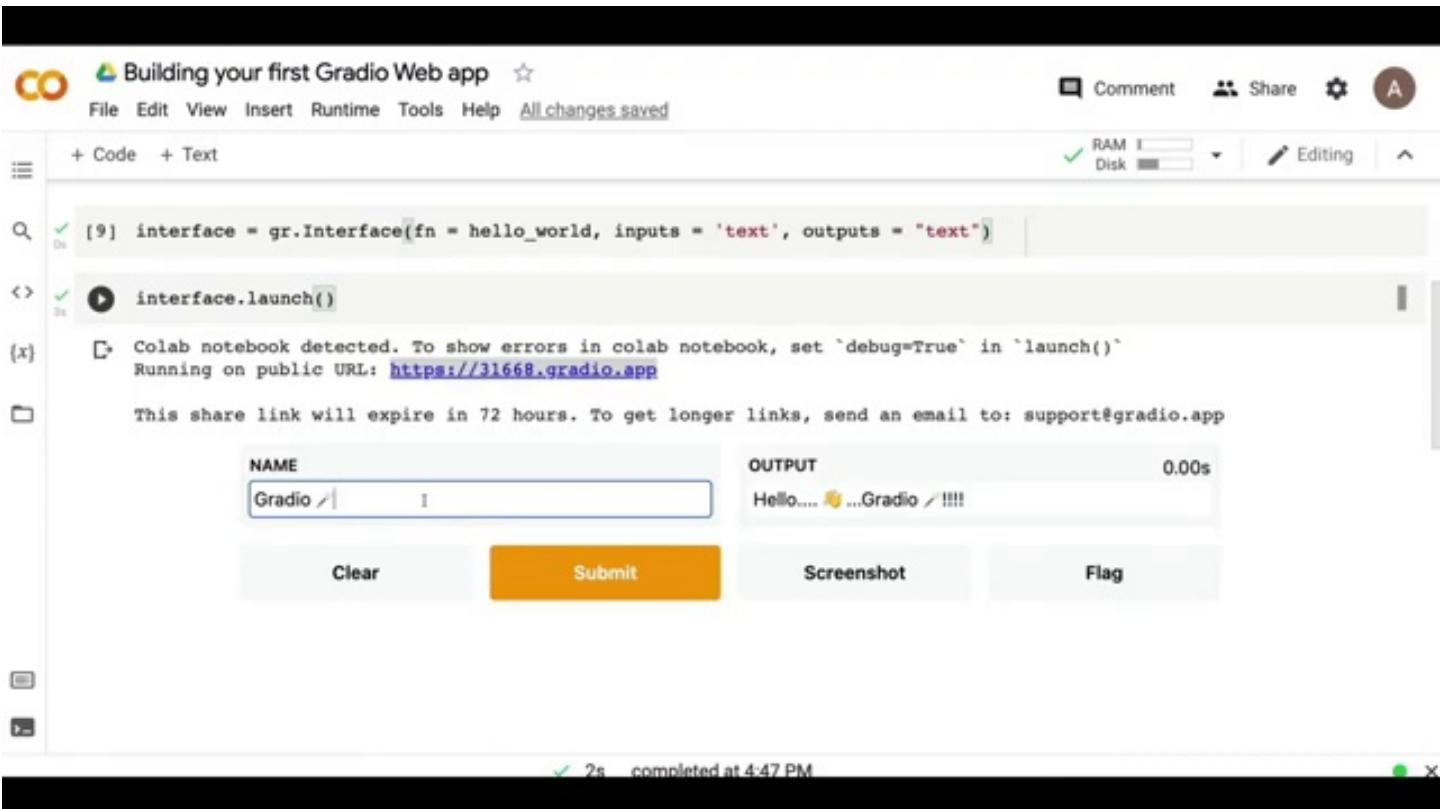
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 461.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Jupyter Notebook interface with a single code cell containing the following Python code:

```
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")  
interface.launch()  
  
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31668.gradio.app
```

Below the code cell, a message indicates that the share link will expire in 72 hours and provides an email address for longer links. A table displays the interface configuration:

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

At the bottom of the cell, there are buttons for Clear, Submit, Screenshot, and Flag. A status bar at the bottom of the notebook window shows a green checkmark, the time '2s', and the completion message 'completed at 4:47 PM'.

**Timestamp: 462.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Jupyter Notebook interface with a single code cell containing the following Python code:

```
[9] interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")  
interface.launch()  
  
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31668.gradio.app
```

Below the code cell, a message indicates that the share link will expire in 72 hours and provides an email address for longer links. A table displays the interface configuration:

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio / !!!!	

At the bottom of the cell, there are buttons for Clear, Submit, Screenshot, and Flag. A status bar at the bottom of the notebook window shows a green checkmark, the time '2s', and the completion message 'completed at 4:47 PM'.

**Timestamp: 463.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")  
interface.launch()  
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31668.gradio.app  
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME	OUTPUT	0.00s
Gradio /	Hello.... 🎉 ...Gradio ✓ !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Code' and 'Text', and status indicators for 'RAM' and 'Disk'. The main area contains a code editor with Python code: 'interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")' and 'interface.launch()'. A note indicates that a Colab notebook is detected, and it provides a public URL: 'https://31668.gradio.app'. A message states that the share link will expire in 72 hours. Below the code editor is a table with columns 'NAME', 'OUTPUT', and '0.00s'. The 'NAME' row contains 'Gradio /'. The 'OUTPUT' row shows the result of the 'hello\_world' function: 'Hello.... 🎉 ...Gradio ✓ !!!!'. At the bottom of the interface are buttons for 'Clear', 'Submit', 'Screenshot', and 'Flag'. A progress bar at the bottom indicates a completion time of '2s' at '4:47 PM'.

**Timestamp: 464.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

[10] interface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Colab notebook interface with a Gradio web application. The code in the cell is:

```
interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

The cell output shows the result of running `interface.launch()`:

```
[10] interface.launch()
```

With the message:

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

The share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

A table displays the output:

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Buttons for Clear, Submit, Screenshot, and Flag are present. A status bar at the bottom indicates the task completed in 2s at 4:47 PM.

**Timestamp: 465.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 466.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
interface = gr.interface(in = hello_world, inputs = text , outputs = text )
```

RAM Disk Editing

interface.launch()

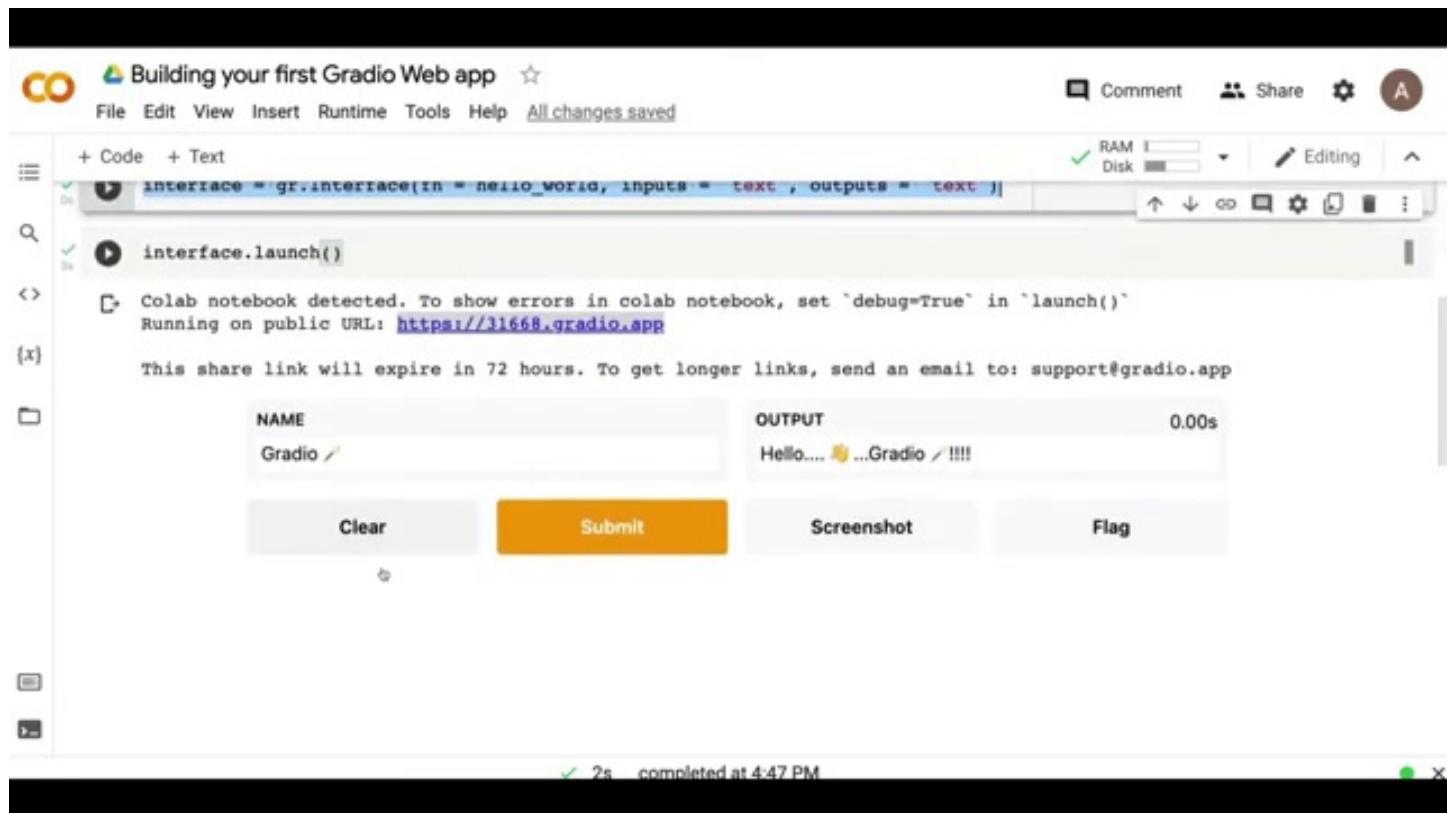
Colab notebook detected. To show errors in colab notebook, set "debug=True" in "launch()"  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 467.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")  
interface.launch()  
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://31668.gradio.app  
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME	OUTPUT	TIME
Gradio ✓	Hello.... 🎉 ...Gradio ✓ !!!!	0.00s

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Comment', 'Share', 'Settings', and a user profile. A sidebar on the left contains icons for file operations like 'New', 'Open', 'Save', and 'Share'. The main area has tabs for '+ Code' and '+ Text'. The code tab is active, displaying the following Python code:interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")  
interface.launch()  
(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>  
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.appA message box indicates that a Colab notebook was detected, suggesting errors can be shown in the Colab notebook if `debug=True` is set in the `launch()` call. It also provides a public URL: <https://31668.gradio.app>. A note states that the share link will expire in 72 hours and offers to send a longer link via email. Below the code, there's a table showing the execution results:| NAME | OUTPUT | TIME |
| --- | --- | --- |
| Gradio ✓ | Hello.... 🎉 ...Gradio ✓ !!!! | 0.00s |

At the bottom of the interface are buttons for 'Clear', 'Submit', 'Screenshot', and 'Flag'. A progress bar at the bottom right shows a green checkmark and the text '2s completed at 4:47 PM'.

**Timestamp: 468.00 seconds**



**Timestamp: 469.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

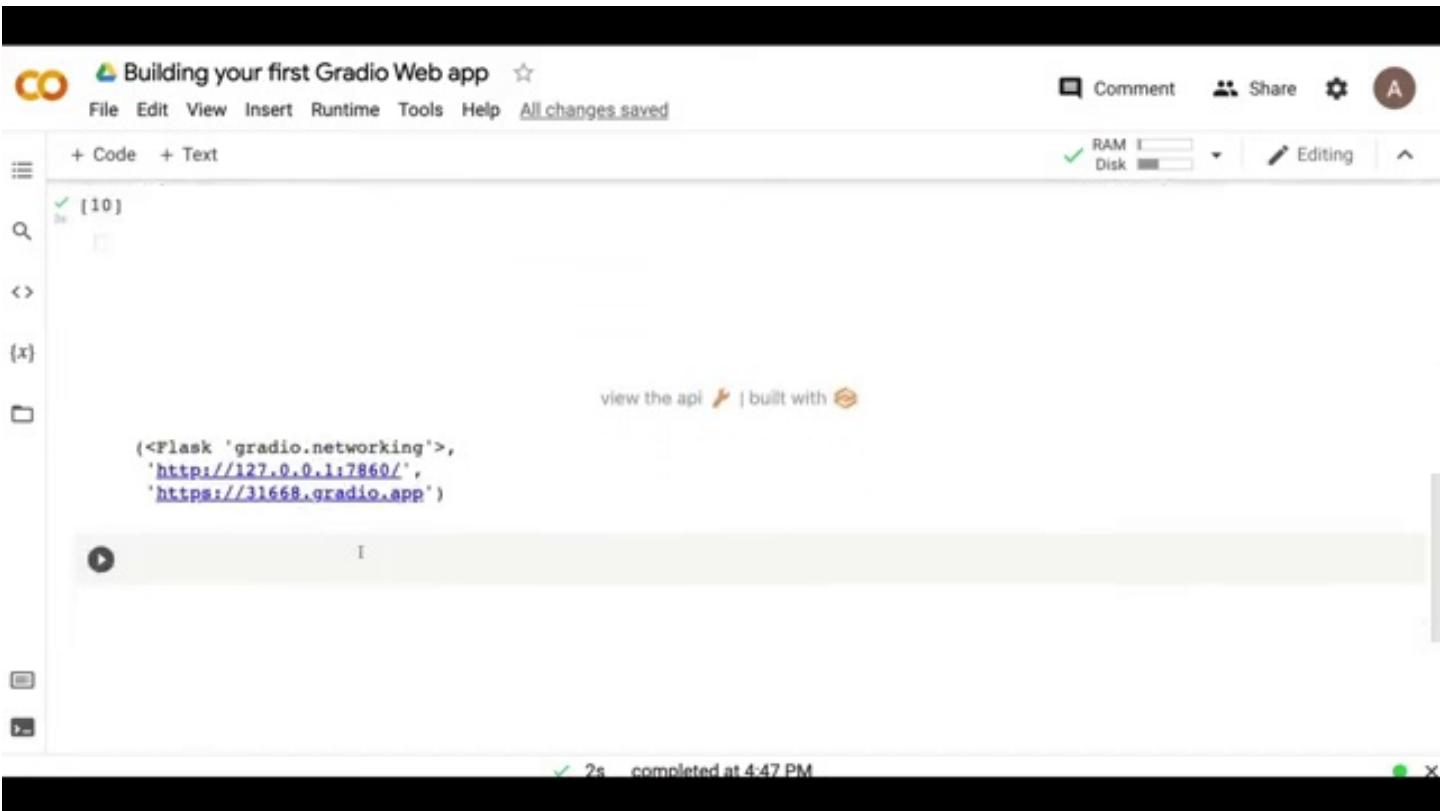
+ Code + Text

[10]

view the api  | built with 

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')
```

2s completed at 4:47 PM



**Timestamp: 470.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

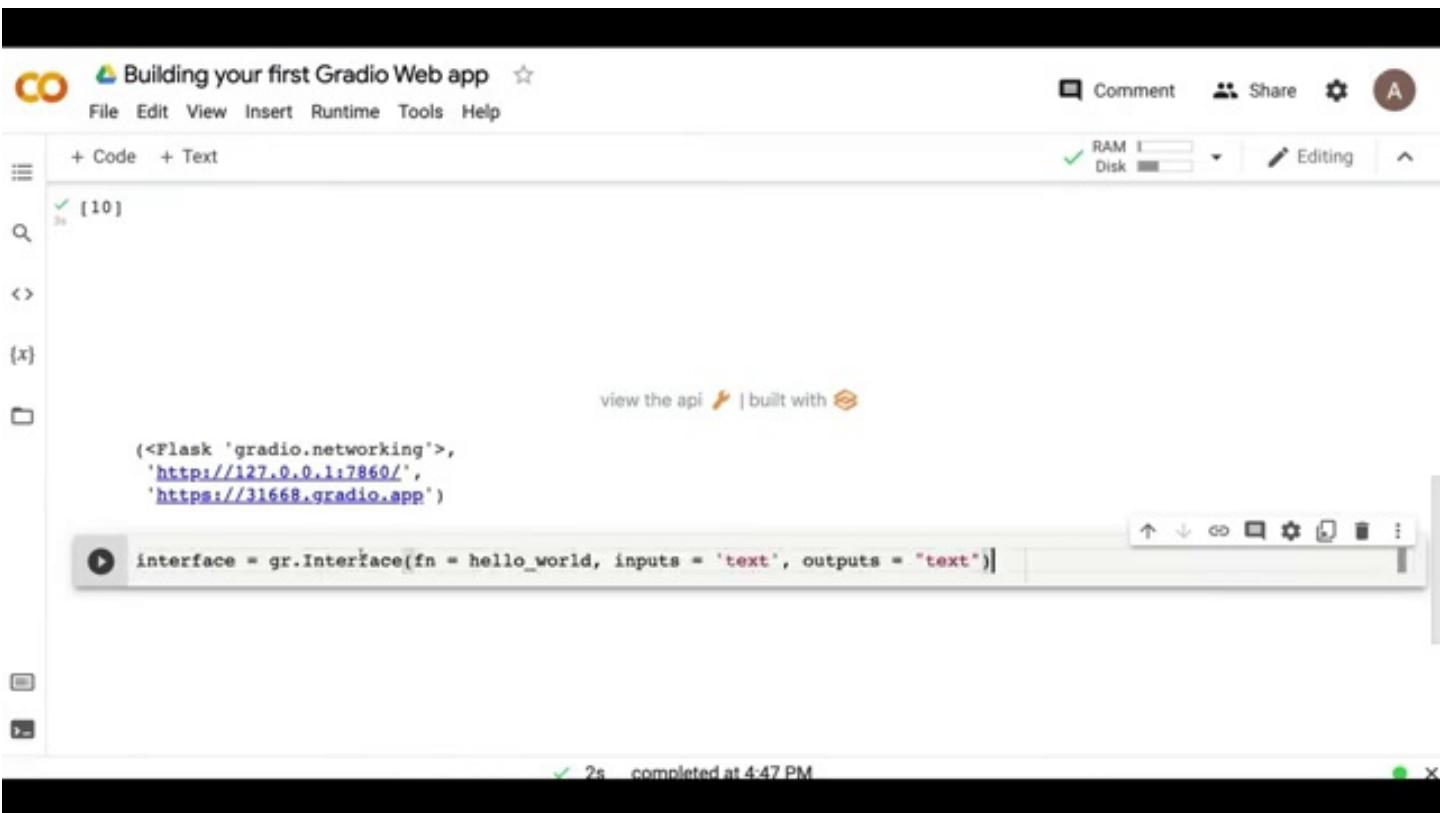
RAM Disk Editing

view the api 🚀 | built with 🎨

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')
```

interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

2s completed at 4:47 PM



**Timestamp: 471.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

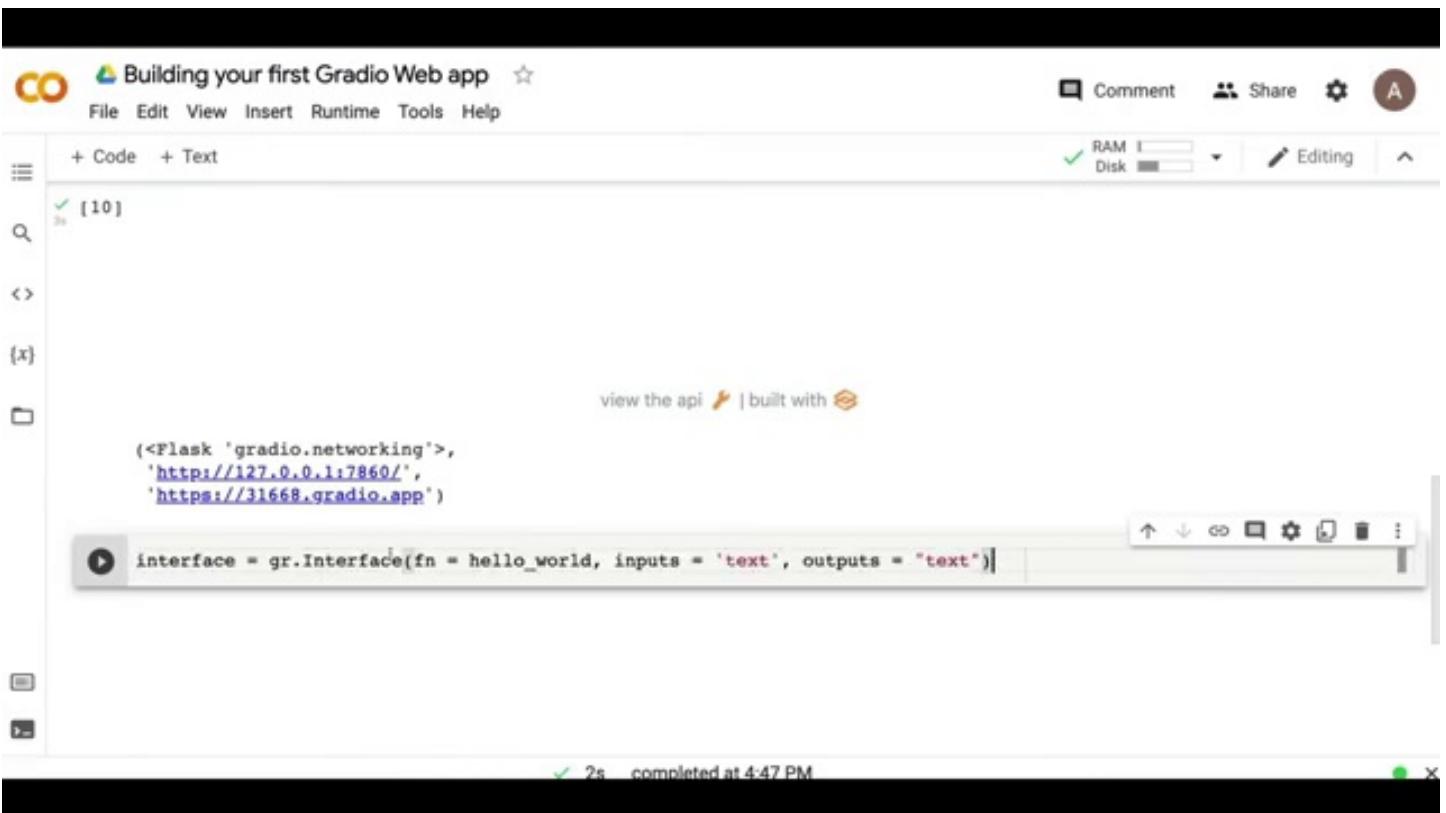
RAM Disk Editing

view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')
```

interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

2s completed at 4:47 PM



**Timestamp: 472.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

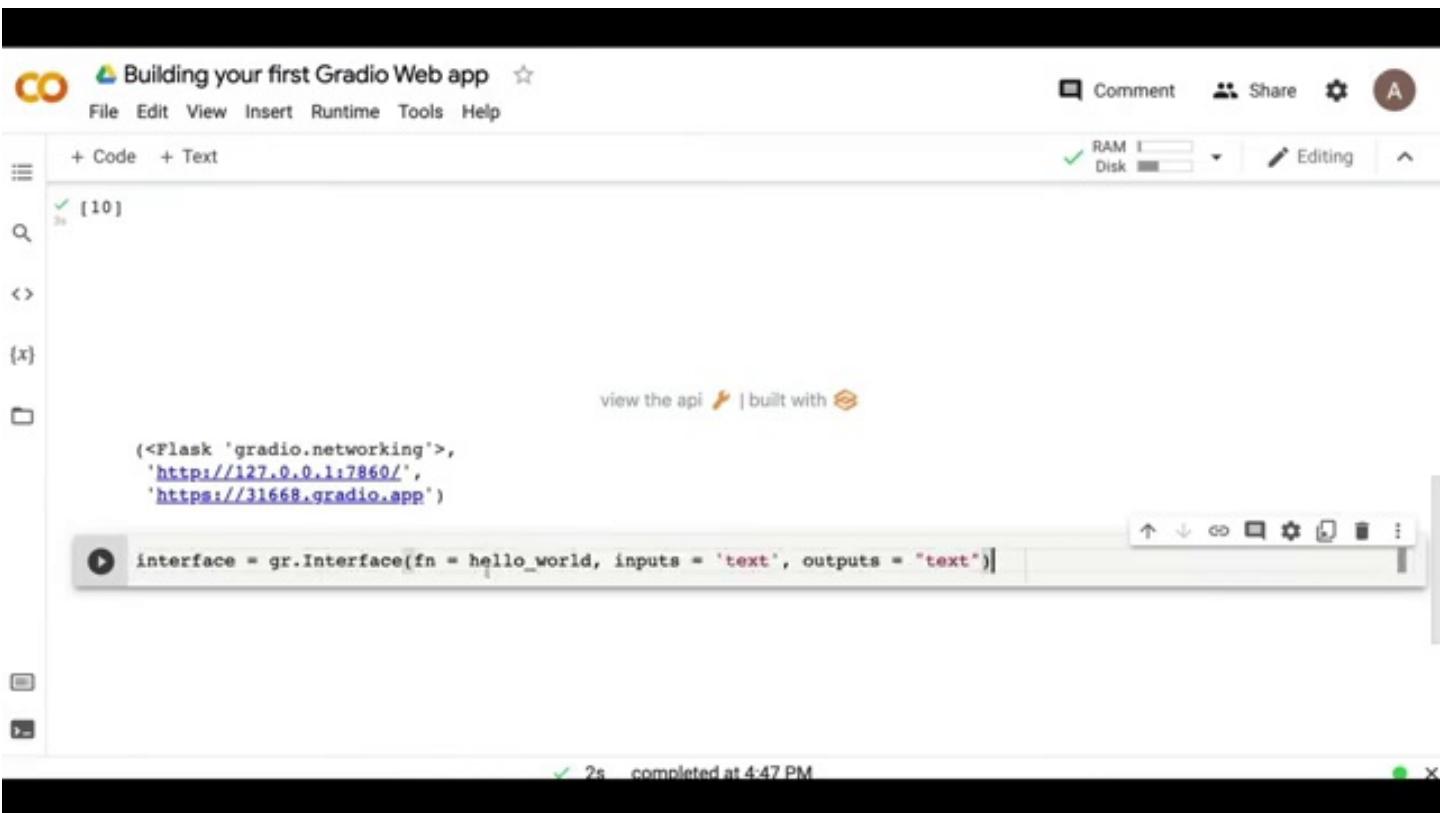
[10]

RAM Disk Editing

view the api | built with

```
<Flask 'gradio.networking',  
'http://127.0.0.1:7860/',  
'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 473.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
<Flask 'gradio.networking'>,  
  'http://127.0.0.1:7860/',  
  'https://31668.gradio.app' )
```

interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

2s completed at 4:47 PM

This screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, there are several icons for file operations like creating new files, opening existing ones, and saving. The main area has tabs for 'Code' and 'Text', with 'Code' currently selected. A code editor window displays Python code for setting up a Gradio interface. Below the code editor is a preview window showing a simple 'Hello World' application. The status bar at the bottom indicates a completion time of '2s' and a date/time of 'completed at 4:47 PM'.

**Timestamp: 474.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(<Flask 'gradio.networking'>,  
 'http://127.0.0.1:7860/',  
 'https://31668.gradio.app')
```

interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

2s completed at 4:47 PM



**Timestamp: 475.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

A screenshot of a web-based development environment for Gradio. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Saving...' status indicator. Below the bar are two tabs: '+ Code' and '+ Text'. A sidebar on the left contains icons for search, refresh, and file operations. The main area shows a code editor with Python code for creating a Gradio interface. The code defines a Flask application with two endpoints: one for http://127.0.0.1:7860/ and another for https://31668.gradio.app/. It then creates an 'interface' object using the 'gr.Interface' function with 'hello\_world' as the function, 'text' as the input type, and 'text' as the output type. To the right of the code editor is a toolbar with various icons. At the bottom, a progress bar indicates the execution took 2 seconds and was completed at 4:47 PM.

**Timestamp: 476.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 477.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

A screenshot of a web-based development environment for Gradio. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Saving...' status indicator. Below the navigation is a toolbar with icons for code (+ Code), text (+ Text), search (Q), compare (<>), and file operations (x). The main area contains a code editor with Python code for creating a Gradio interface. The code defines a Flask application with two endpoints: one for http://127.0.0.1:7860/ and another for https://31668.gradio.app/. It then creates an 'interface' object using gr.Interface with 'hello\_world' as the function, 'text' as the input type, and 'text' as the output type. To the right of the code editor is a status bar showing 'RAM' and 'Disk' usage, both with green checkmarks. Below the code editor is a message 'view the api | built with'. At the bottom, a progress bar shows '2s completed at 4:47 PM'.

**Timestamp: 478.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM



**Timestamp: 479.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app'}
```

interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

2s completed at 4:47 PM



**Timestamp: 480.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world, inputs = 'text', outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

A screenshot of a web-based development environment for Gradio. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Code' and 'Text'. A sidebar on the left contains icons for search, copy/paste, and file operations. The main area is a code editor with a syntax-highlighted Python script. The script defines a Flask application and a Gradio interface named 'hello\_world' with text inputs and outputs. Below the code editor is a status bar showing '2s completed at 4:47 PM'. The bottom right corner has a small terminal window icon.

**Timestamp: 481.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = 'text', outputs = "text")
```

2s completed at 4:47 PM

This screenshot shows the Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, there are several icons for file operations like creating new files, opening, saving, and deleting. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. Below the tabs is a search bar and a diff viewer icon. A sidebar on the left contains icons for a file, a folder, and a terminal. The central workspace displays Python code for setting up a Gradio interface. At the bottom, a progress bar indicates the execution took 2 seconds and completed at 4:47 PM.

**Timestamp: 482.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app'

interface = gr.Interface(fn = hello_world,
    inputs = 'text', outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 483.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

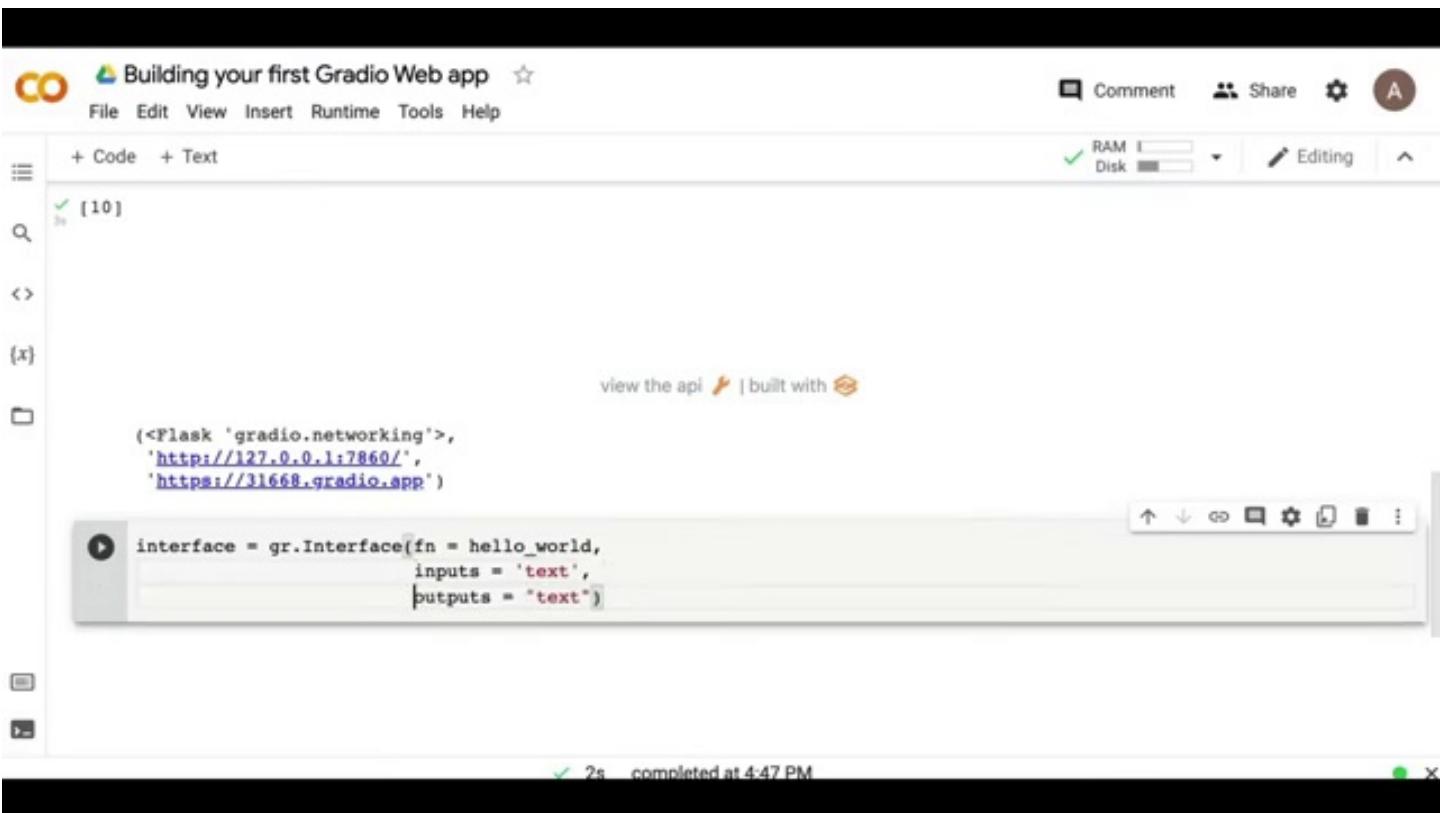
[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = 'text',
                         outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 484.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	TIME
Gradio ✓	Hello.... 🎉 ...Gradio ✓ !!!!	0.00s

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

**Timestamp: 485.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

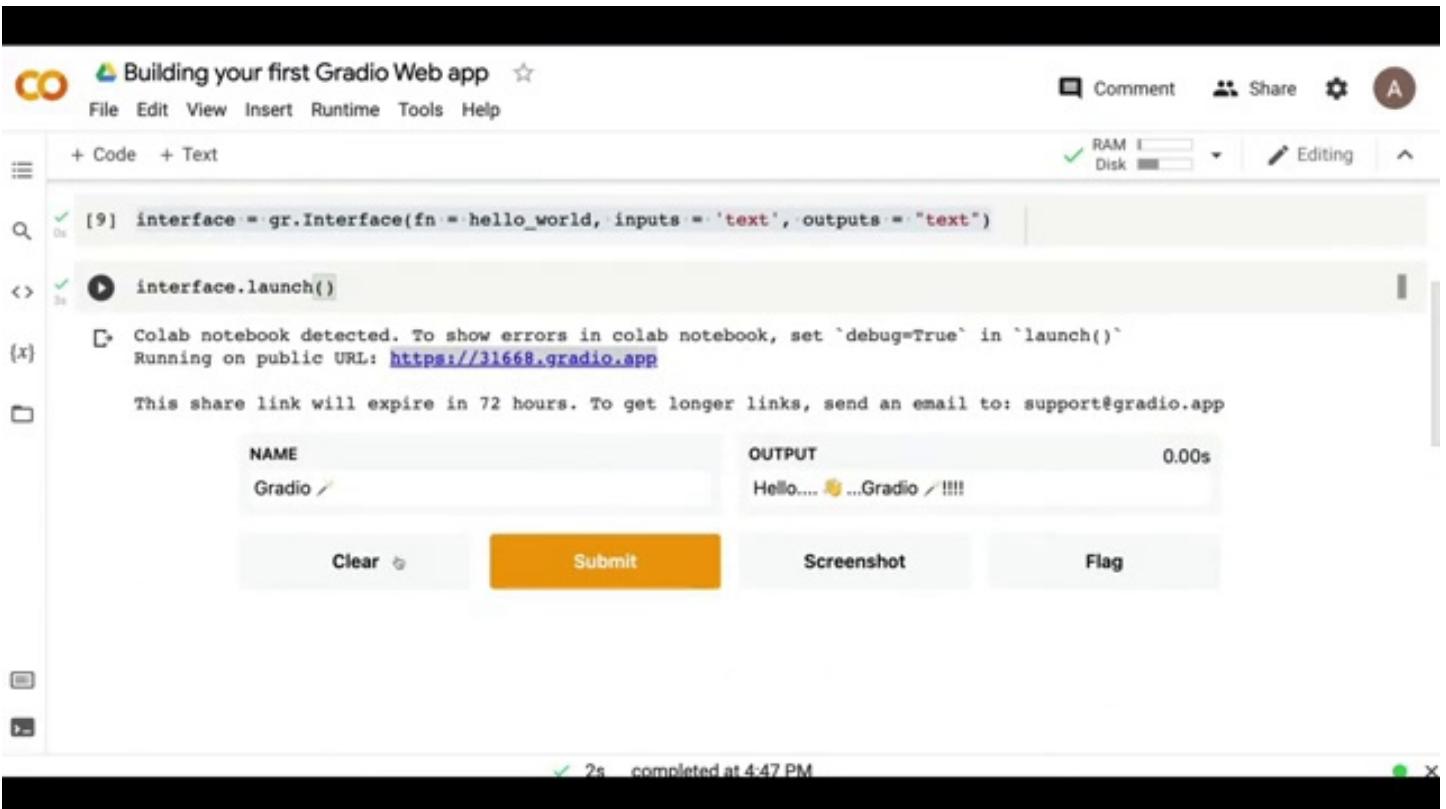
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 486.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

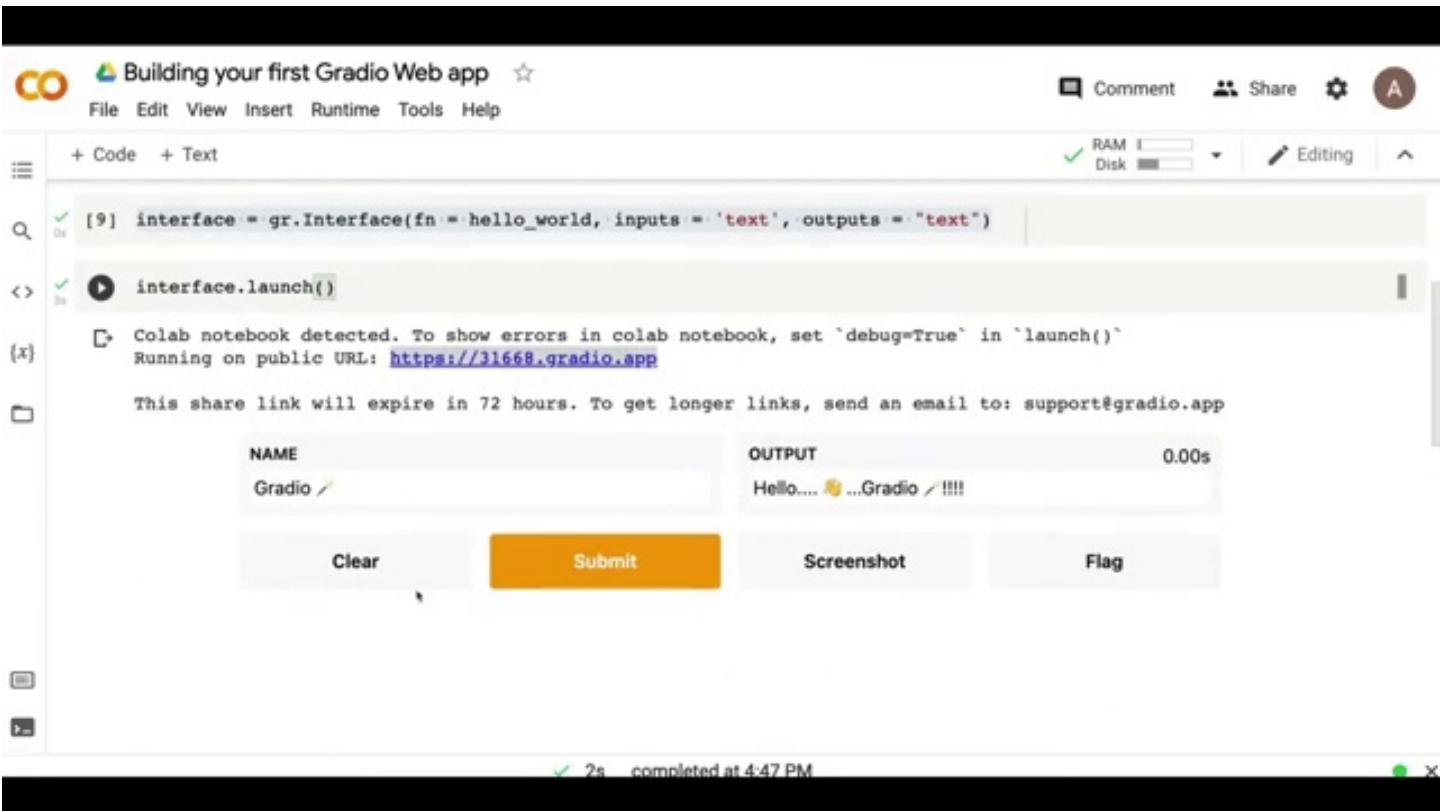
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	0.00s
Gradio	Hello.... 🎉 ...Gradio !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM



**Timestamp: 487.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 488.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	TIME
Gradio	Hello.... 🎉 ...Gradio !!!!	0.00s

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' options. On the far right are 'Comment', 'Share', 'Settings', and a user profile icon. Below the navigation is a toolbar with icons for 'Code' and 'Text'. A status bar at the top right shows 'RAM' and 'Disk' usage. The main area contains a code editor with a single line of Python code: 'interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")'. Below the code is a button labeled 'interface.launch()'. A note indicates that a Colab notebook is detected, and it provides a public URL: 'https://31668.gradio.app'. A message states that the share link will expire in 72 hours. Below this is a table showing the output of the application. The table has three columns: 'NAME', 'OUTPUT', and 'TIME'. There is one row with the name 'Gradio' and the output 'Hello.... 🎉 ...Gradio !!!!'. The time taken is '0.00s'. At the bottom of the table are four buttons: 'Clear', 'Submit' (which is highlighted in orange), 'Screenshot', and 'Flag'. A progress bar at the bottom indicates the task was completed in 2 seconds at 4:47 PM.

**Timestamp: 489.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[9] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	OUTPUT	TIME
Gradio	Hello.... 🎉 ...Gradio !!!!	0.00s

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

The screenshot shows a Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' options. Below the navigation is a toolbar with icons for 'Code' and 'Text'. On the left, there are several small icons: a list, a search bar, a file folder, and a refresh symbol. The main area contains a code editor with Python code, a terminal-like output window, and a results table. The code defines a Gradio interface and launches it. The output window shows a warning about Colab detection and provides a public URL. The results table shows one entry with the name 'Gradio' and the output 'Hello.... 🎉 ...Gradio !!!!'. At the bottom, there are buttons for 'Clear', 'Submit', 'Screenshot', and 'Flag', and a status message indicating the task completed in 2 seconds at 4:47 PM.

**Timestamp: 490.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

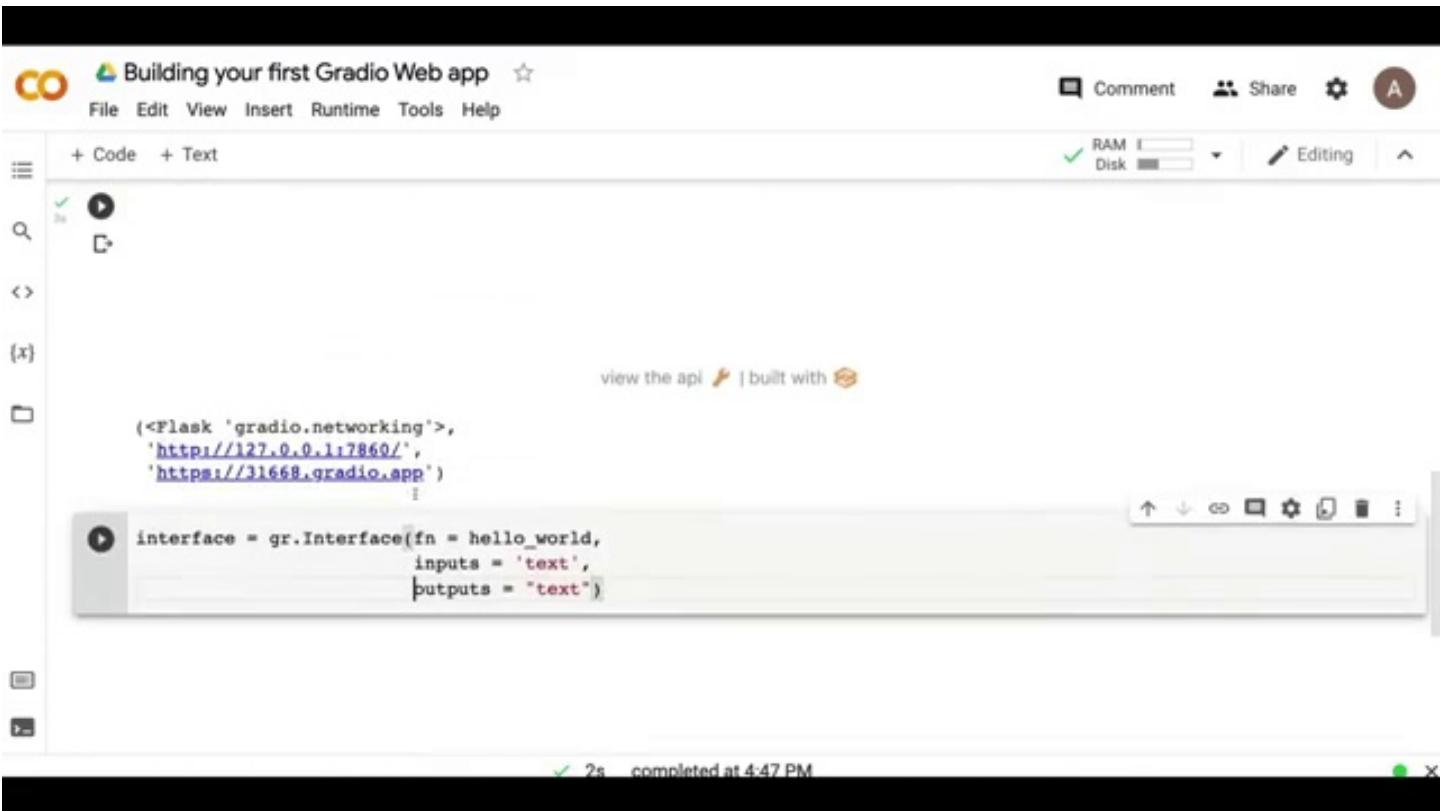
+ Code + Text

RAM Disk Editing

view the api | built with

```
<Flask 'gradio.networking',  
'http://127.0.0.1:7860/',  
'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world,  
    inputs = 'text',  
    outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 491.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = 'text',
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

The screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, there are several icons for file operations like creating new files, opening existing ones, and saving. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. The code editor contains Python code for a Flask application and a Gradio interface. Below the code editor is a preview window showing a simple UI with a text input field and a text output field. A status bar at the bottom indicates a completion time of '2s' and a date/time of 'completed at 4:47 PM'.

**Timestamp: 492.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

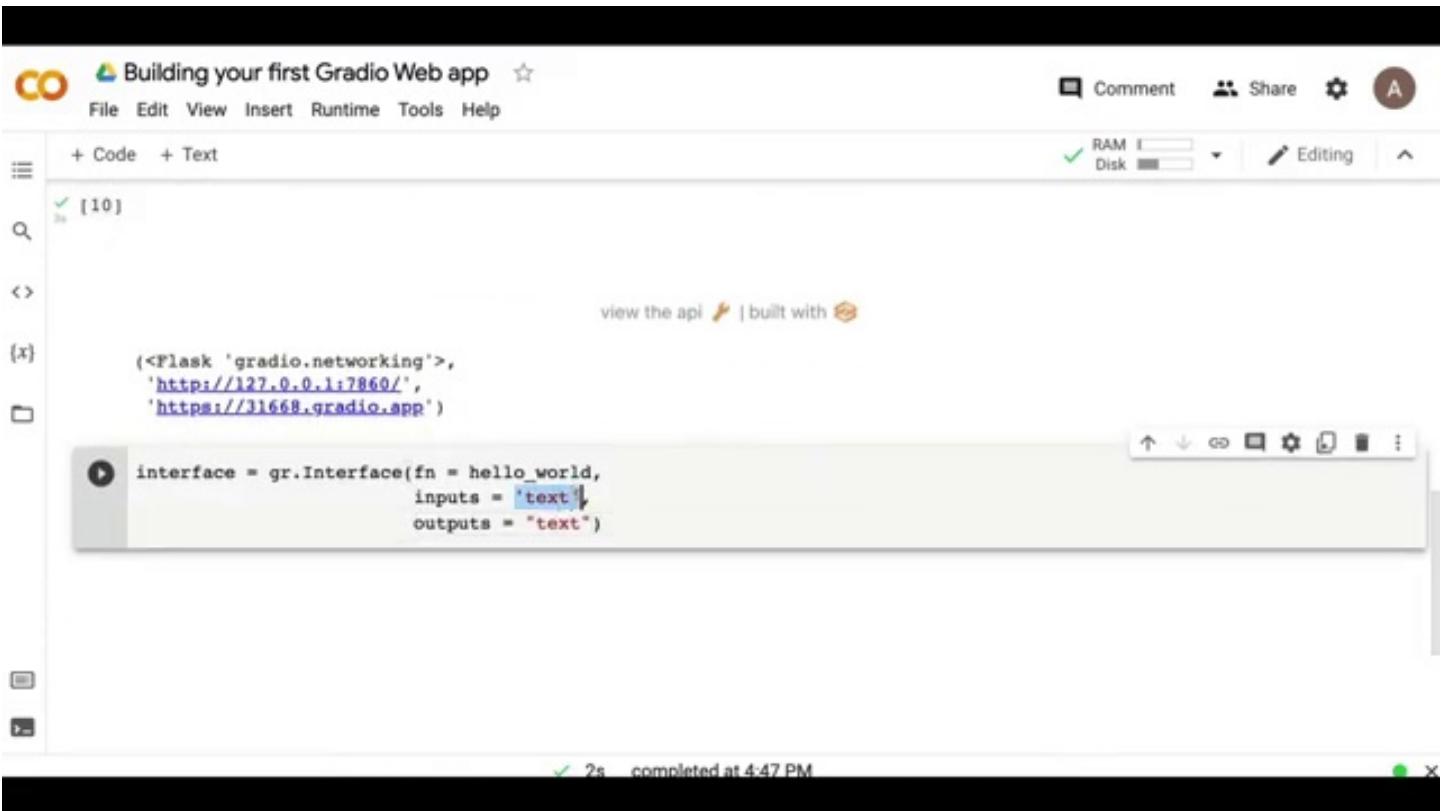
view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                        inputs = 'text',
                        outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM



**Timestamp: 493.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = df,
                         outputs = "text")
```

2s completed at 4:47 PM

A screenshot of a web-based code editor titled "Building your first Gradio Web app". The interface includes a top navigation bar with File, Edit, View, Insert, Runtime, Tools, and Help options. On the right side of the header are Comment, Share, and settings icons. Below the header is a toolbar with buttons for RAM and Disk usage, Editing mode, and other controls. The main workspace shows a code editor with a syntax-highlighted Python script. The script defines a Flask application and a Gradio interface named "hello\_world" that takes a DataFrame input and produces text output. A status bar at the bottom indicates the code was completed in 2 seconds at 4:47 PM.

**Timestamp: 494.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

This screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the navigation bar is a toolbar with icons for 'Code' and 'Text'. A sidebar on the left contains icons for search, refresh, and file operations. The main area has a code editor window titled '[10]' containing Python code for creating a Gradio interface. To the right of the code editor is a preview window showing a simple 'Hello World' application. Below the preview is a status bar indicating the execution time was 2 seconds and it completed at 4:47 PM. The bottom right corner features a small circular profile icon.

**Timestamp: 495.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

This screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, there are several icons for file operations like creating new files, opening existing ones, and saving. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. The code editor contains Python code for creating a Gradio interface. Below the code is a preview window showing a simple 'Hello World' application. A status bar at the bottom indicates the build took 2 seconds and completed at 4:47 PM.

**Timestamp: 496.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

This screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the navigation bar is a toolbar with icons for 'Code' and 'Text'. A sidebar on the left contains icons for search, refresh, and file operations. The main area has a code editor window titled '[10]' containing Python code for creating a Gradio interface. To the right of the code editor is a preview window showing a simple 'Hello World' application. Below the preview is a status bar indicating the build took 2 seconds and was completed at 4:47 PM. The bottom right corner features a small circular profile icon.

**Timestamp: 497.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

This screenshot shows the Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the navigation bar is a toolbar with icons for 'Comment', 'Share', 'Settings', and a user profile. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. A code editor window displays Python code for creating a Gradio interface. The code defines a Flask application with two routes: one for HTTP and one for HTTPS. It then creates an 'Interface' object with 'hello\_world' as the function, 'gr.' as inputs, and 'text' as outputs. To the right of the code editor is a status bar showing 'RAM' and 'Disk' usage, both with green checkmarks. Below the status bar is a message indicating the deployment was completed at 4:47 PM. On the far left, there are several small, unlabeled icons.

**Timestamp: 498.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 499.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs,
                         outputs = "text")() inputs
@get_input_instance
```

RAM Disk Editing

2s completed at 4:47 PM

The screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the left, there are icons for search, refresh, and file operations. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. A code editor window displays Python code for a Flask application. The code defines a function `hello\_world` and creates a `gr.Interface` object named `interface`. The interface has `inputs` and `outputs` defined. Below the code, there's a status bar showing '2s completed at 4:47 PM'. At the bottom right, there are icons for saving, running, and stopping the application.

**Timestamp: 500.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                        inputs = gr.input,
                        outputs = "text")() inputs
                        ⚡ get_input_instance
```

RAM Disk Editing

2s completed at 4:47 PM

A screenshot of a web-based code editor titled "Building your first Gradio Web app". The interface includes a top navigation bar with File, Edit, View, Insert, Runtime, Tools, and Help menus. On the left, there are icons for search, refresh, and file operations. The main area shows a code editor with a syntax-highlighted Python-like script. A tooltip "module" is visible over the "inputs" line. Below the code, a status bar indicates "2s completed at 4:47 PM". The bottom right corner features a small profile icon.

**Timestamp: 501.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM

**Timestamp: 502.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

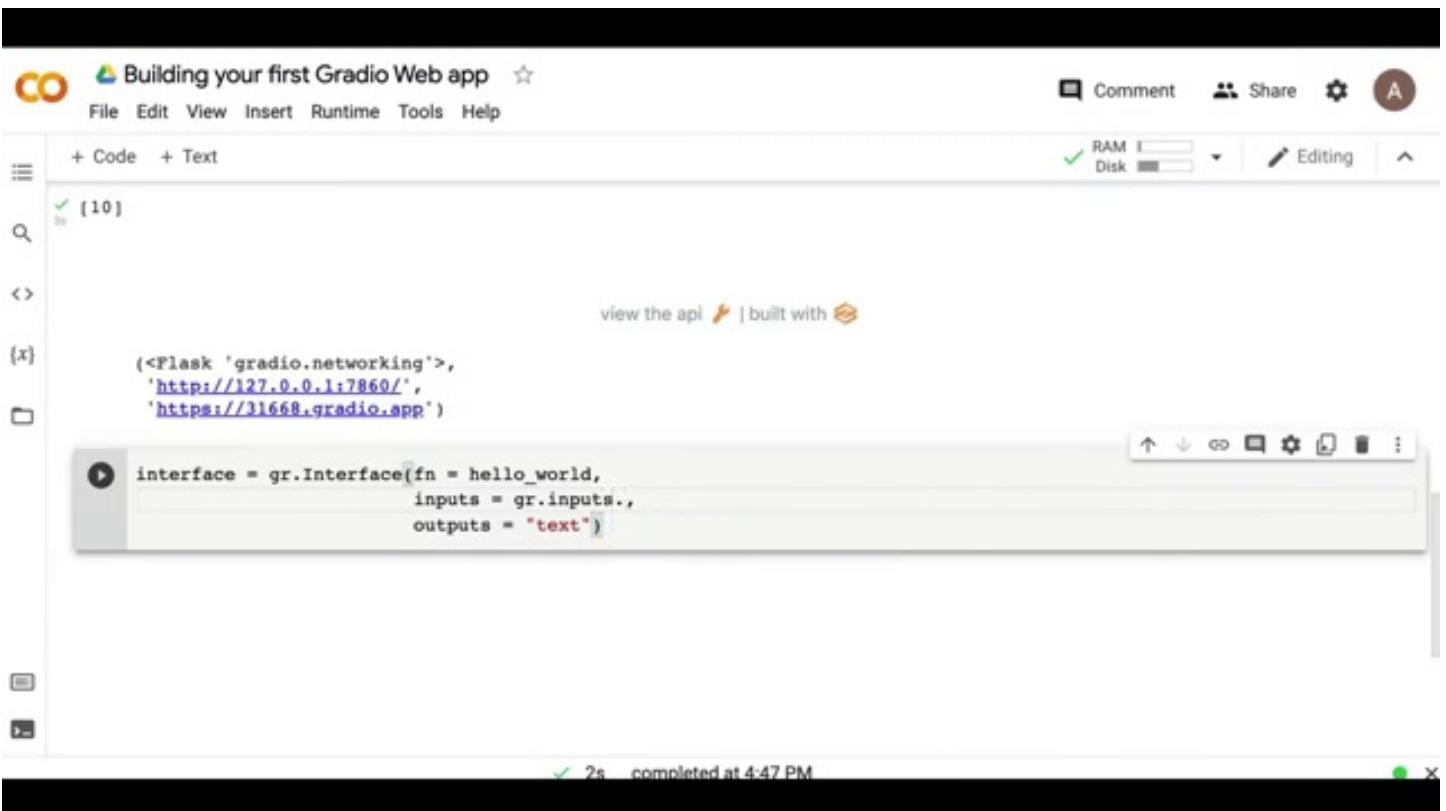
view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM



**Timestamp: 503.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

✓ 2s completed at 4:47 PM

The screenshot shows a web-based development environment for creating Gradio web applications. The main area contains Python code for defining a Gradio interface. A tooltip over the code highlights various Gradio component classes. The status bar at the bottom indicates the execution time and completion time.

**Timestamp: 504.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

✓ 2s completed at 4:47 PM

This screenshot shows a web-based development environment for creating Gradio web applications. The main area displays a Python script defining a Gradio interface. A tooltip provides a detailed list of available Gradio components. The status bar at the bottom indicates the execution time and completion time.

**Timestamp: 505.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

✓ 2s completed at 4:47 PM

**Timestamp: 506.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

✓ 2s completed at 4:47 PM

The screenshot shows a web-based development environment for Gradio. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A floating code completion dropdown is open over the code editor, listing various Gradio components such as 'Audio', 'Checkbox', 'CheckboxGroup', 'Component', 'Dataframe', 'Dropdown', 'FFmpeg', 'File', 'get\_input\_instance', 'Image', 'InputComponent', and 'json'. The main code editor contains Python code for defining a Gradio interface:

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

The status bar at the bottom right shows a green checkmark icon, the text '2s completed at 4:47 PM', and a small terminal icon.

**Timestamp: 507.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[10]

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.,
                         outputs = "text")
```

RAM Disk Editing

✓ 2s completed at 4:47 PM

The screenshot shows a web-based development environment for creating Gradio web applications. The main area displays Python code for defining a Gradio interface. A dropdown menu is open, listing various Gradio components and utilities. The status bar at the bottom indicates the execution time and completion time.

**Timestamp: 508.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

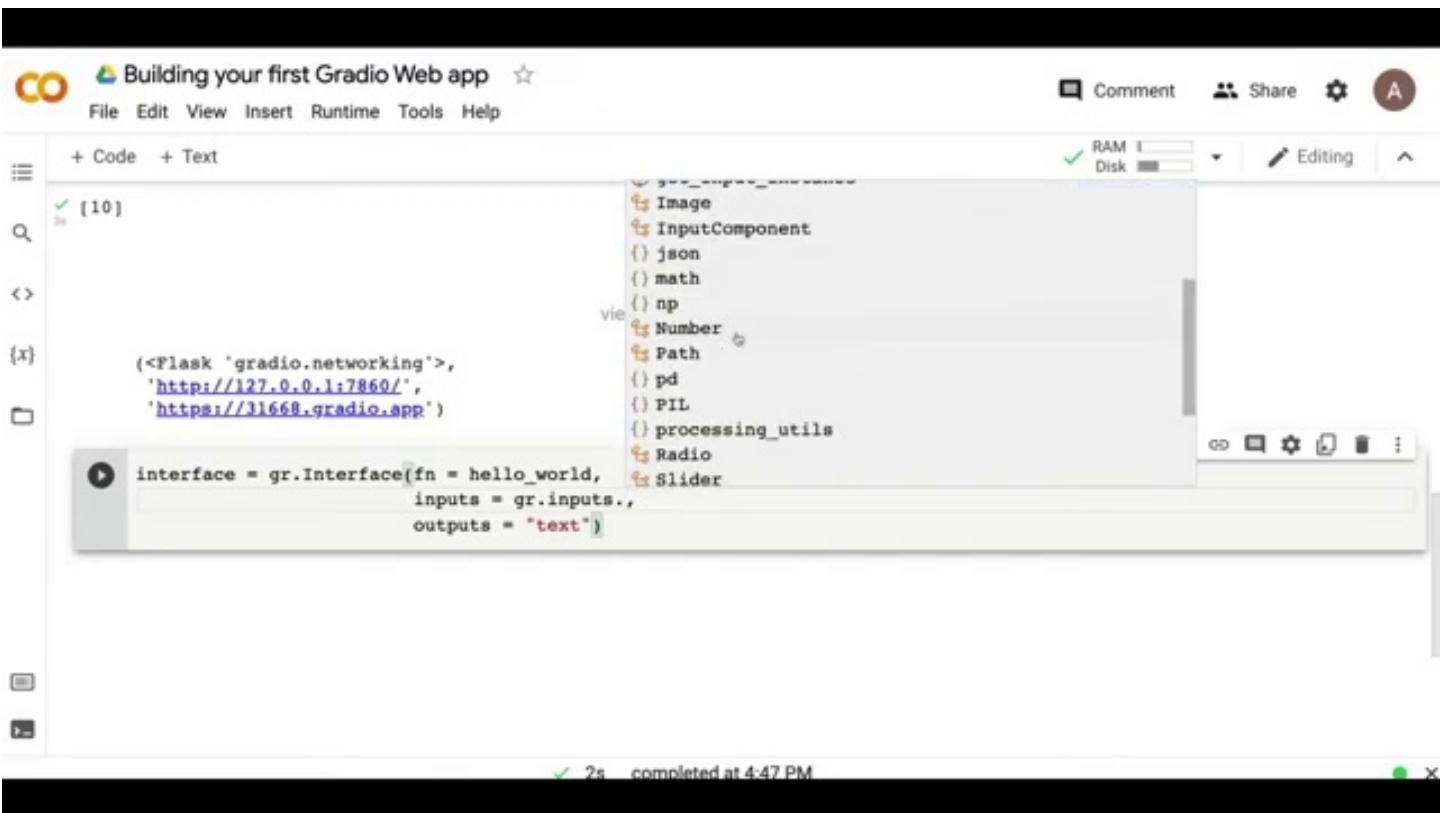
+ Code + Text

[10]

```
<Flask 'gradio.networking'>,  
 'http://127.0.0.1:7860/',  
 'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world,  
                         inputs = gr.inputs.,  
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM



**Timestamp: 509.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 510.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

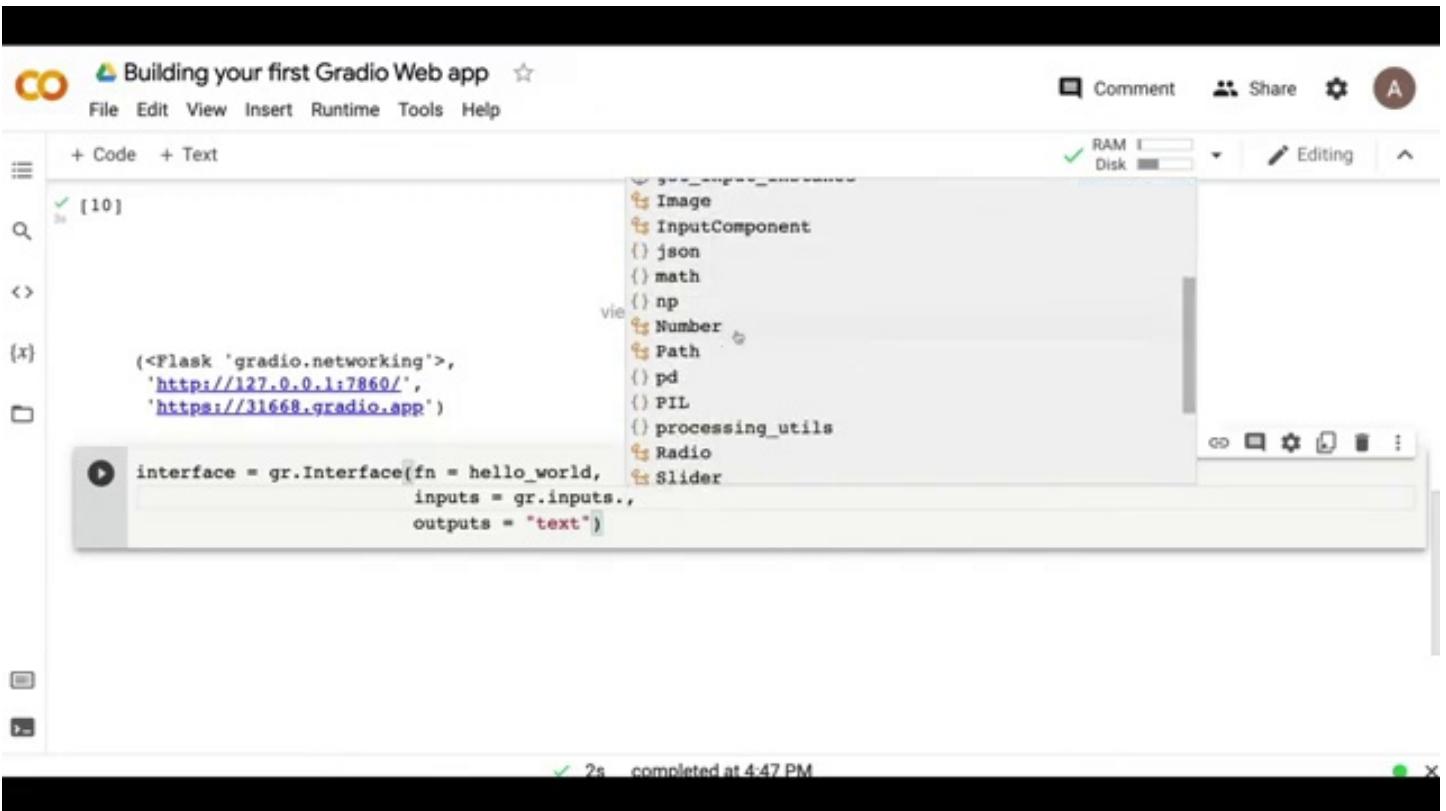
+ Code + Text

[10]

```
<Flask 'gradio.networking'>,  
 'http://127.0.0.1:7860/',  
 'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world,  
                         inputs = gr.inputs.,  
                         outputs = "text")
```

RAM Disk Editing

2s completed at 4:47 PM



**Timestamp: 511.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox,
                         outputs = "text")
```

Textbox def \_\_init\_\_(lines=1, placeholder=None, ...)

tempfile test\_data Timeseries

2s completed at 4:47 PM

The screenshot shows the Gradio web application builder interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'Saving...'. On the right of the nav bar are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the nav bar is a toolbar with icons for file operations like saving, opening, and deleting, along with buttons for 'RAM' and 'Disk' usage monitoring, and an 'Editing' mode switch.

The main workspace has tabs for '+ Code' and '+ Text'. The '+ Code' tab is active, displaying Python code for a Gradio interface:

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox,
                         outputs = "text")
```

A tooltip or dropdown menu is open over the 'Textbox' class name, showing its definition:

```
Textbox def __init__(lines=1, placeholder=None, ...)
```

Below the code, there's a message indicating the task completed successfully: '2s completed at 4:47 PM'.

**Timestamp: 512.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox,
                         outputs = "text")
```

Textbox def \_\_init\_\_(lines=1, placeholder=None...)

2s completed at 4:47 PM

This screenshot shows the Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a 'Saving...' status indicator. On the right of the top bar are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the top bar is a toolbar with icons for file operations like saving, opening, and deleting, along with a search icon. The main area has tabs for '+ Code' and '+ Text'. The code tab is active, displaying Python code for creating a Gradio interface. The text tab shows a preview of the interface with a 'Textbox' input and a 'Text' output. Below the preview is a status bar indicating '2s completed at 4:47 PM'.

**Timestamp: 513.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

✓ [10]

view the api 🔥 | built with 🎨

{x} (`<Flask 'gradio.networking'>,  
 'http://127.0.0.1:7860/'  
 'https://31668.gradio.app'`)

interface = gr.Interface(fn = hello\_world,  
 inputs = gr.inputs.Textbox,  
 outputs = "text")

2s completed at 4:47 PM

The screenshot shows a web-based development environment. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the navigation is a toolbar with icons for file operations like 'New', 'Open', 'Save', and a search icon. The main workspace has tabs for 'Code' and 'Text', with 'Code' currently selected. The code area contains a snippet of Python using the 'gradio' library to create a simple web application. It includes a URL for the local host and a secure URL for the deployed app. Below the code is a button with a play icon and the text 'interface = gr.Interface(fn = hello\_world, inputs = gr.inputs.Textbox, outputs = "text")'. At the bottom of the editor window, there's a progress bar indicating '2s completed at 4:47 PM'.

**Timestamp: 514.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

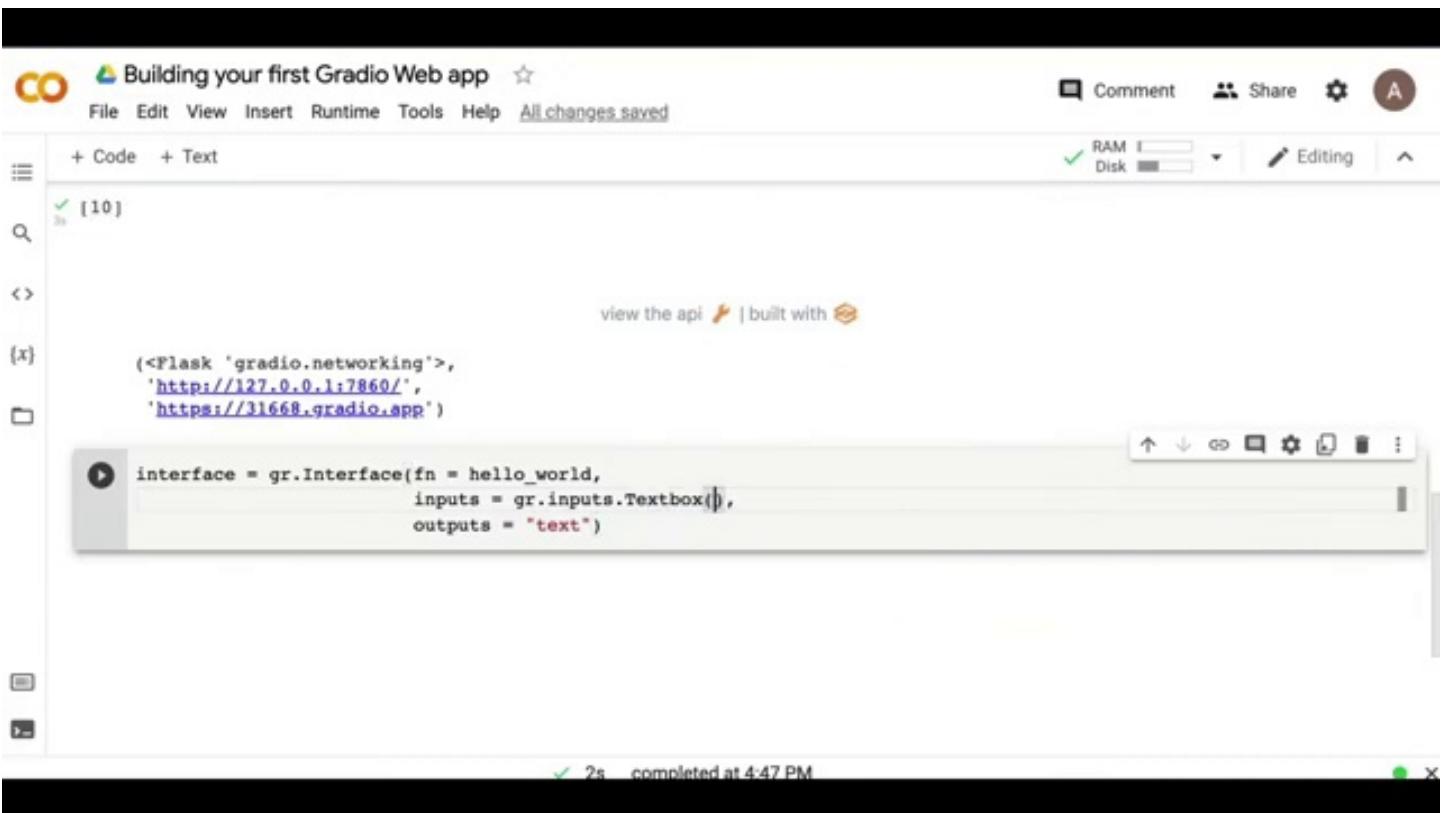
[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(),
                         outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 515.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[10]

view the api

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(),
                         outputs = "text")
```

Component creates a textbox for user to enter input. Provides a string.

Input type: str

Demos: hello\_world.py, diff\_texts.py

2s completed at 4:47 PM

**Timestamp: 516.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api ↗ Built with 🎨

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines=1),
                         outputs = "text")
```

Component creates a textbox for user to enter input. Provides a string.  
Input type: str  
Demos: hello\_world.py, diff\_texts.py

RAM Disk Editing

2s completed at 4:47 PM

**Timestamp: 517.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[10]

view the api 🔥 | built with 🎨

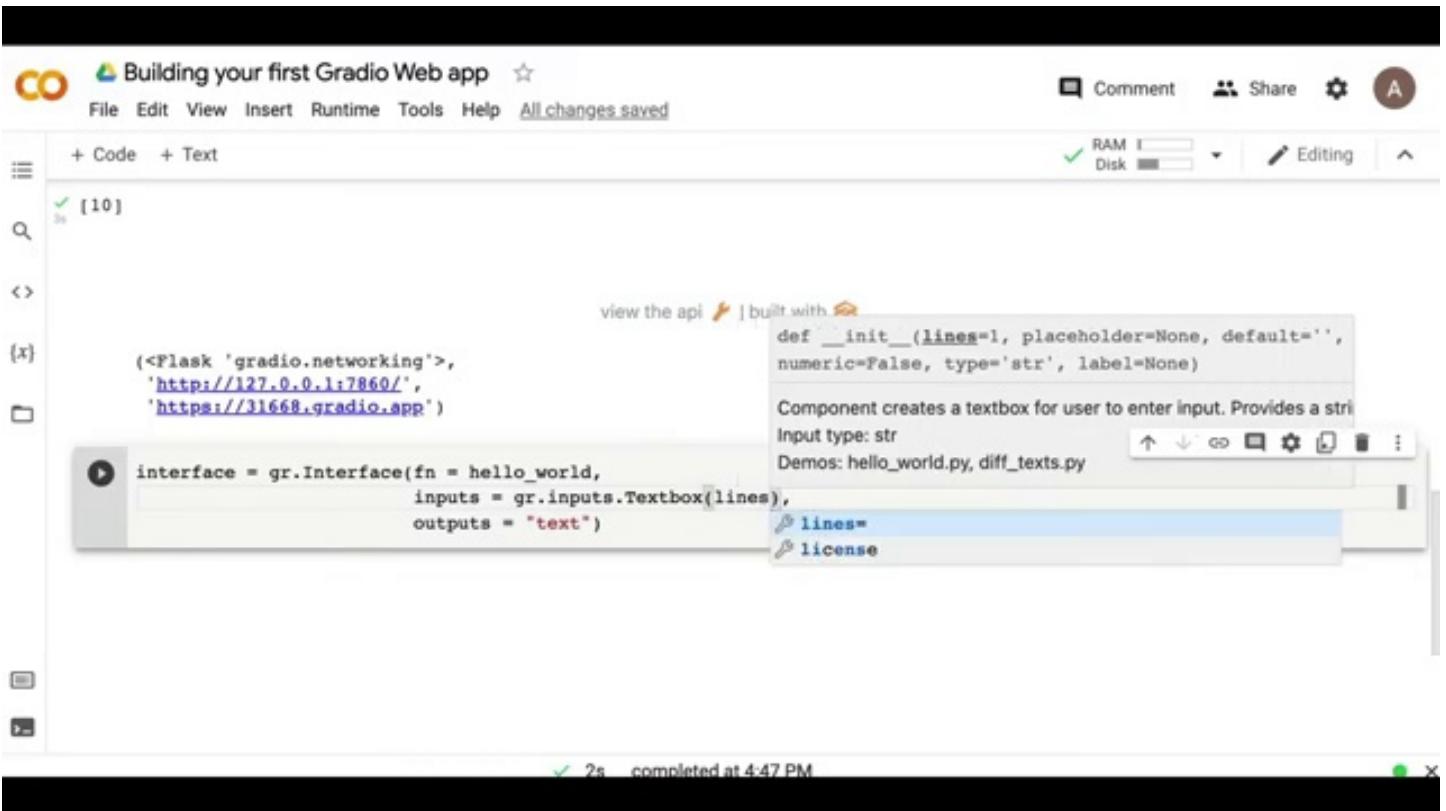
```
(x) <Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world,  
                         inputs = gr.inputs.Textbox(lines),  
                         outputs = "text")
```

RAM Disk Editing

Component creates a textbox for user to enter input. Provides a stri  
Input type: str  
Demos: hello\_world.py, diff\_texts.py

lines= license

2s completed at 4:47 PM



**Timestamp: 518.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api 🔥 | built with 🎨

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 1),
                         outputs = "text")
```

def \_\_init\_\_(lines=1, placeholder=None, default='', numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a string.

Input type: str

Demos: hello\_world.py, diff\_texts.py

2s completed at 4:47 PM

**Timestamp: 519.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api 🔥 | built with 🚀

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5),
                         outputs = "text")
```

def \_\_init\_\_(lines=1, placeholder=None, default='',
 numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a string.

Input type: str

Demos: hello\_world.py, diff\_texts.py

2s completed at 4:47 PM

**Timestamp: 520.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 521.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api 🔥 | built with 🚀

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, ),
                         outputs = "text")
```

def \_\_init\_\_(lines=1, placeholder=None, default='', numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a string.

Input type: str

Demos: hello\_world.py, diff\_texts.py

✓ 2s completed at 4:47 PM

**Timestamp: 522.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api 🔥 | built with 🎨

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder=""),
                         outputs = "text")
```

def \_\_init\_\_(lines=1, placeholder=None, default='', numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a str

Input type: str

Demos: hello\_world.py, diff\_texts.py

✓ 2s completed at 4:47 PM

**Timestamp: 523.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                        inputs = gr.inputs.Textbox(lines = 5, placeholder),
                        outputs = "text")
```

RAM Disk Editing

def \_\_init\_\_(lines=1, placeholder=None, default='', numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a str

Input type: str

Demos: hello\_world.py, diff\_texts.py

placeholder=

✓ 2s completed at 4:47 PM

**Timestamp: 524.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder=""),
                         outputs = "text")
```

def \_\_init\_\_(lines=1, placeholder=None, default='', numeric=False, type='str', label=None)

Component creates a textbox for user to enter input. Provides a str

Input type: str

Demos: hello\_world.py, diff\_texts.py

✓ 2s completed at 4:47 PM

**Timestamp: 525.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter"),
                         outputs = "text")
```

2s completed at 4:47 PM

The screenshot shows a web-based development environment for creating Gradio web applications. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the right of the nav bar are 'Comment', 'Share', a gear icon for settings, and a user profile icon with the letter 'A'. Below the nav bar is a toolbar with icons for 'RAM' (green checkmark), 'Disk' (grey), 'Editing' (pencil), and other system controls. The main workspace has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. It displays a code editor with Python code for a Gradio interface. The code defines a Flask application with two routes: one for HTTP and one for HTTPS. It uses the 'Textbox' input component from Gradio with 5 lines and a placeholder 'Enter', and the 'text' output component. Below the code editor is a status bar showing '2s completed at 4:47 PM'. To the left of the editor, there are several small icons for file operations like save, open, and delete.

**Timestamp: 526.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

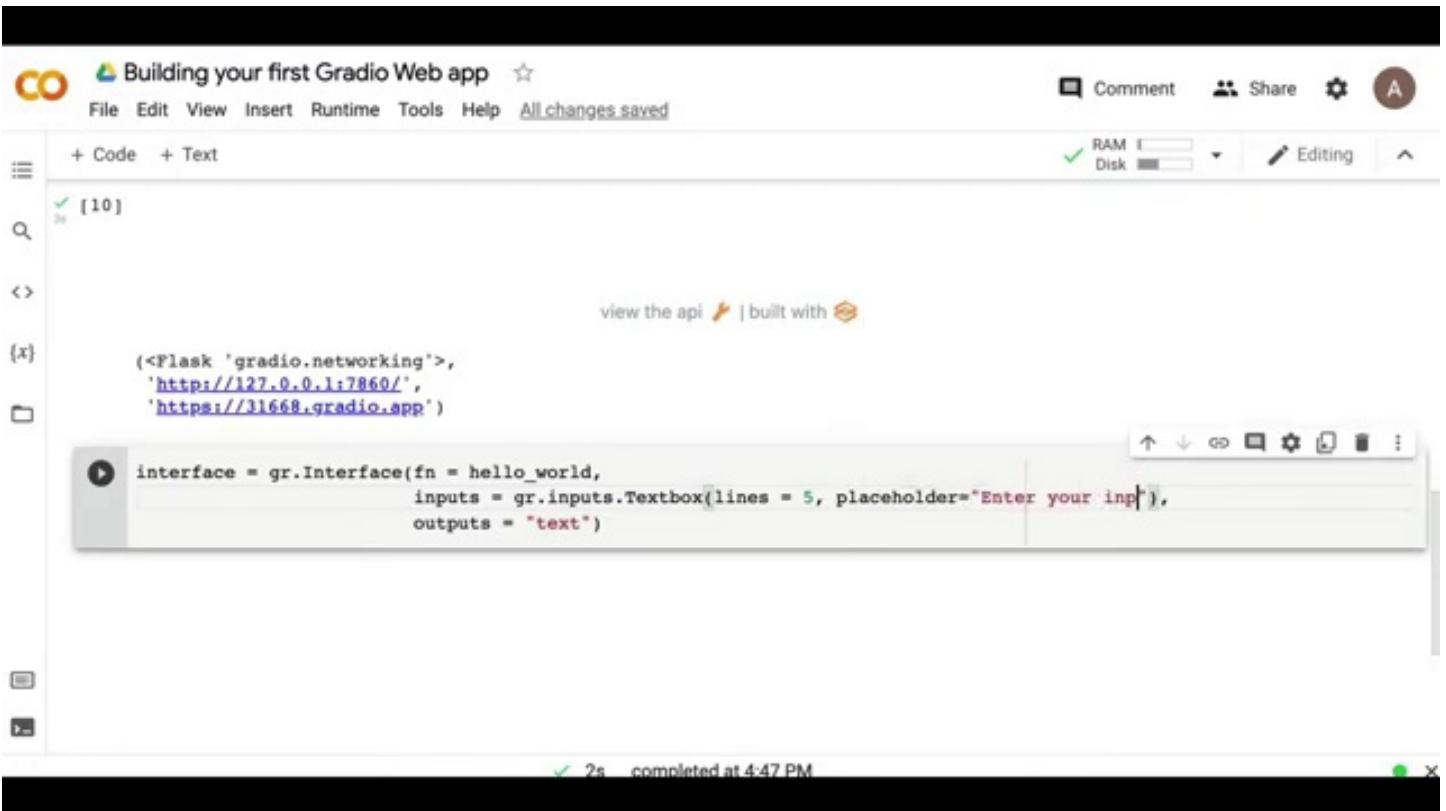
[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input"),
                         outputs = "text")
```

2s completed at 4:47 PM



**Timestamp: 527.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here.."),
                         outputs = "text")
```

2s completed at 4:47 PM

The screenshot shows a web-based development environment. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status message are icons for 'Comment', 'Share', 'Settings', and a user profile. Below the navigation bar is a toolbar with icons for 'Code' and 'Text', a search bar, and a refresh button. The main workspace contains a code editor with Python code for creating a Gradio interface. The code defines a Flask application with two routes: one for HTTP and one for HTTPS. It uses the 'gradio' library to create a simple text input field and a text output field. A status bar at the bottom indicates a completion time of '2s' and a timestamp of 'completed at 4:47 PM'.

**Timestamp: 528.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                         outputs = "text")
```

2s completed at 4:47 PM

The screenshot shows a web-based development environment for creating Gradio web applications. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the right side of the header are 'Comment', 'Share', 'Settings', and a user profile icon. Below the header, there are tabs for '+ Code' and '+ Text', with '+ Code' currently selected. The main workspace displays a Python script for defining a Gradio interface. The code uses the `gradio` library to create a simple application with a text input field and a text output field. A status bar at the bottom indicates a completion time of '2s' and a timestamp of 'completed at 4:47 PM'.

**Timestamp: 529.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                         outputs = "text")
```

2s completed at 4:47 PM

The screenshot shows a web-based development environment for creating Gradio web applications. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. On the right of the nav bar are 'Comment', 'Share', 'Settings', and a user profile icon. Below the nav bar is a toolbar with icons for 'RAM' (green checkmark), 'Disk' (grey), 'Editing' (pencil), and other system controls. The main workspace has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. It displays a Python script using the Gradio library to define a simple interface. The interface has a placeholder text input field and returns text output. Below the code is a preview area showing a play button and the generated Gradio interface. At the bottom, a progress bar indicates the task was completed in 2 seconds at 4:47 PM.

**Timestamp: 530.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

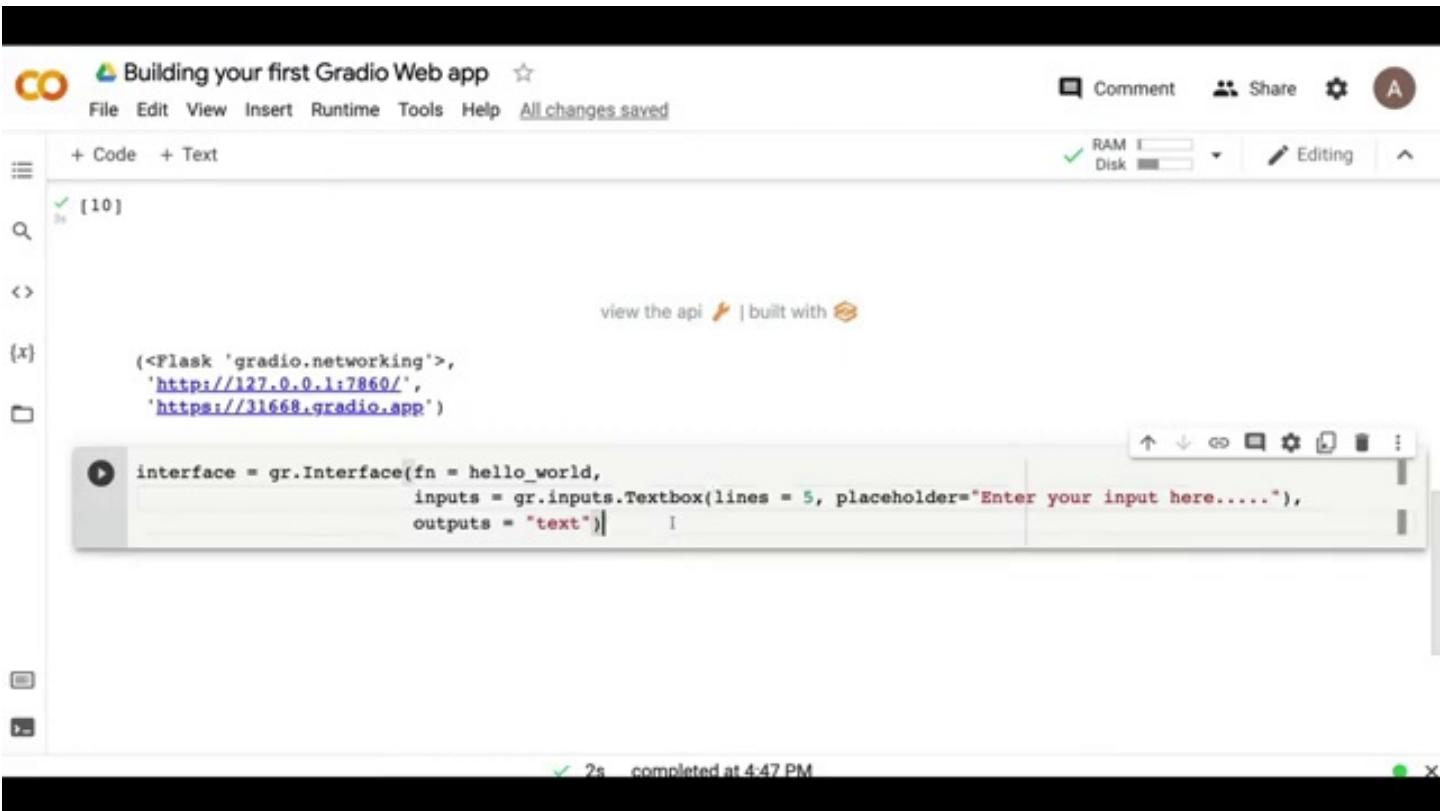
[10]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7860/',
 'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
    inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
    outputs = "text")|
```

2s completed at 4:47 PM



**Timestamp: 531.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 532.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[ 10 ]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')  
  
interface = gr.Interface(fn = hello_world,  
                        inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),  
                        outputs = "text")
```

+ Code + Text

Executing (0s)

**Timestamp: 533.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

[11] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                             outputs = "text")
```

0s completed at 5:03 PM

**Timestamp: 534.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

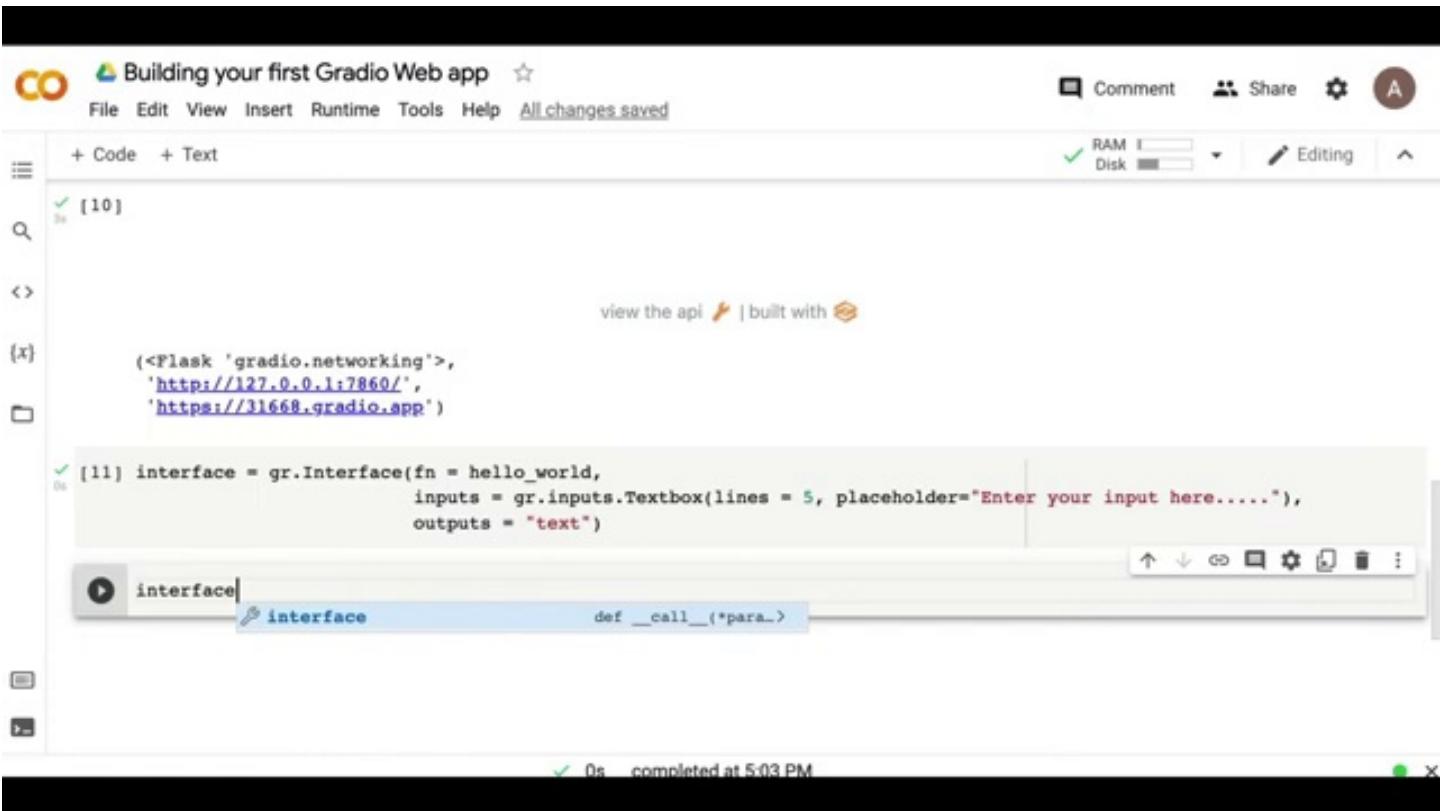
RAM Disk Editing

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')  
  
[11] interface = gr.Interface(fn = hello_world,  
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),  
                             outputs = "text")  
  
interface
```

0s completed at 5:03 PM



**Timestamp: 535.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

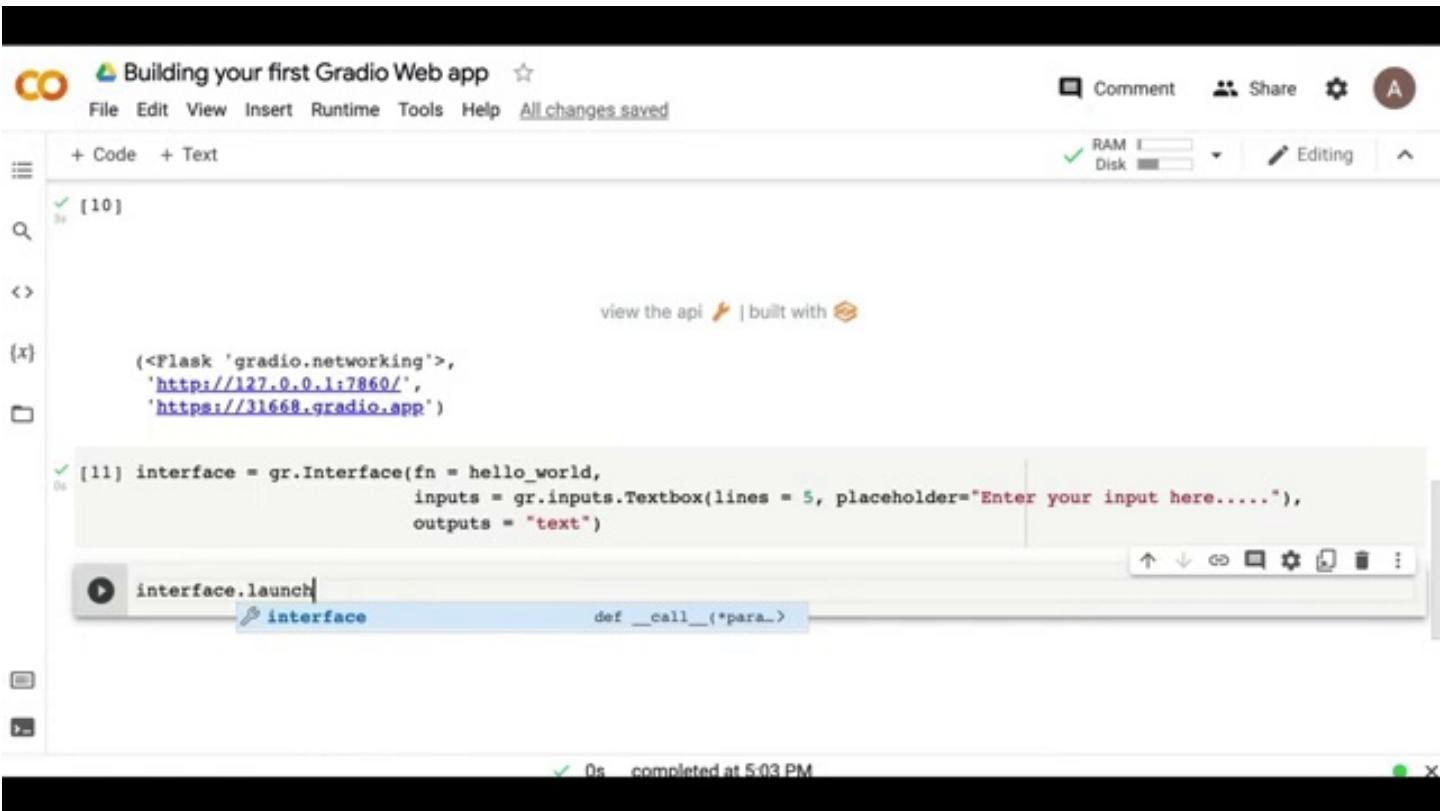
RAM Disk Editing

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')  
  
[11] interface = gr.Interface(fn = hello_world,  
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),  
                             outputs = "text")  
  
interface.launch()  
def __call__(*para...)
```

0s completed at 5:03 PM



**Timestamp: 536.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

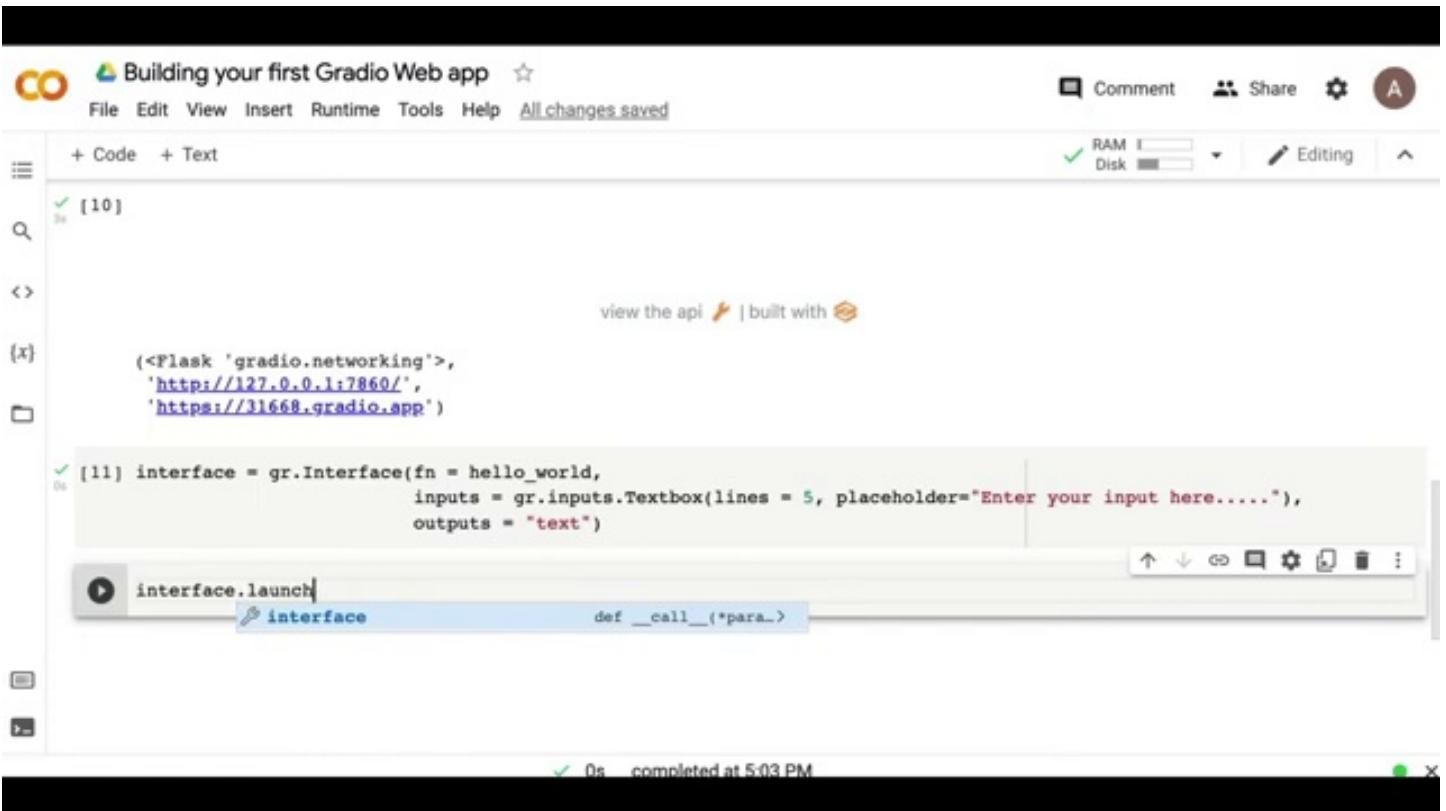
RAM Disk Editing

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')  
  
[11] interface = gr.Interface(fn = hello_world,  
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),  
                             outputs = "text")  
  
interface.launch()  
def __call__(*para...)
```

0s completed at 5:03 PM



**Timestamp: 537.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[10]

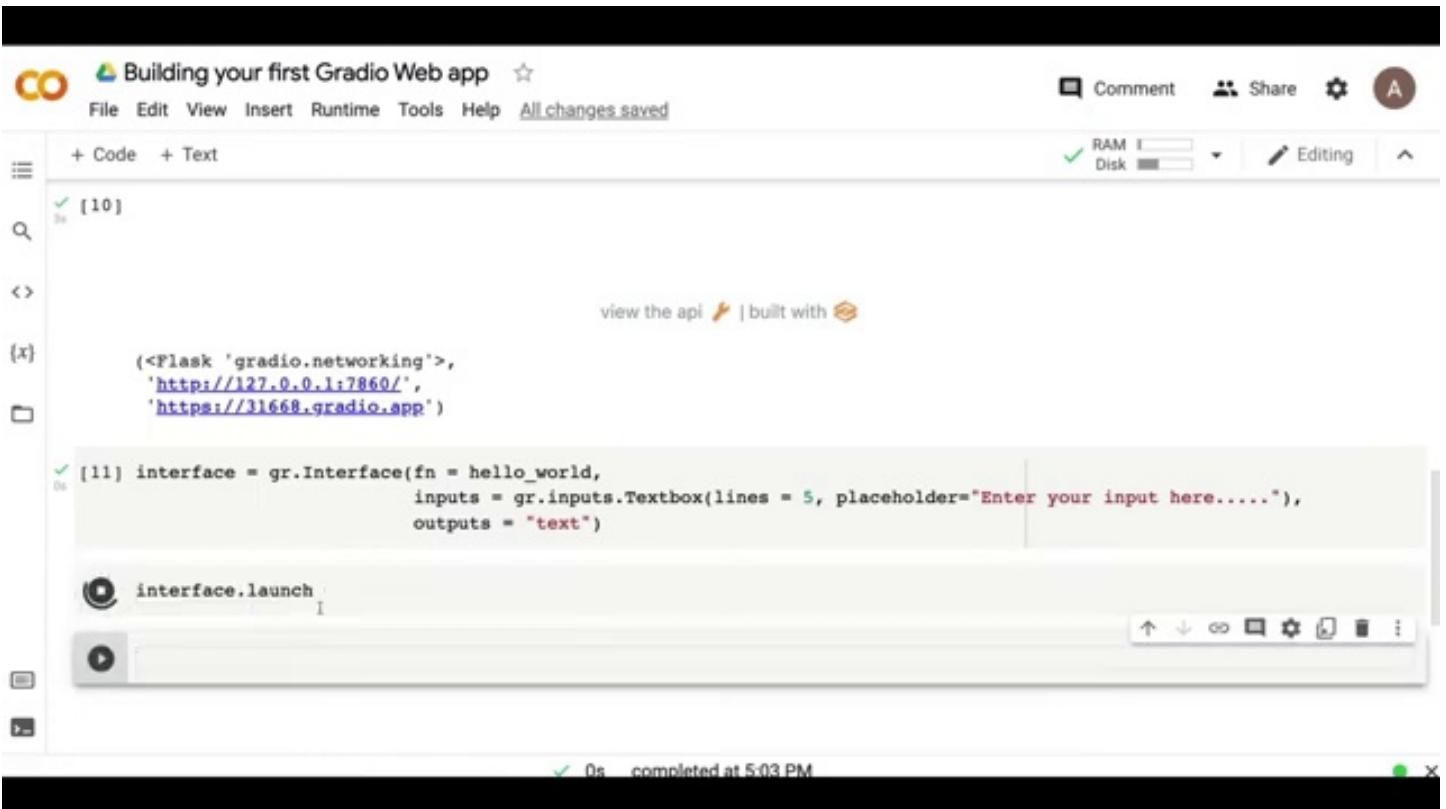
view the api 🔥 | built with 🎨

```
(x) (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

[11] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch

0s completed at 5:03 PM



**Timestamp: 538.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

[10]

view the api | built with

```
(x) (<Flask 'gradio.networking'>,  
     'http://127.0.0.1:7860/',  
     'https://31668.gradio.app')
```

[11] interface = gr.Interface(fn = hello\_world,  
 inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),  
 outputs = "text")

interface.launch

<bound method Interface.launch of Gradio Interface for: hello\_world  
-----  
inputs:  
|-Textbox(label="None")  
outputs:  
|-Text(label="None")

0s completed at 5:03 PM

**Timestamp: 539.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

[10] view the api 🚀 | built with 🎉

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7860/',
'https://31668.gradio.app'}
```

[x] [11] interface = gr.Interface(fn = hello\_world,
inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
outputs = "text")

interface.launch()

<bound method Interface.launch of Gradio Interface for: hello\_world  
-----  
inputs:  
|-Textbox(label="None")  
outputs:  
|-Textbox(label="None")>

0s completed at 5:03 PM

**Timestamp: 540.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[1] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch()

...

0s completed at 5:03 PM

This screenshot shows a web-based development environment for creating Gradio web applications. The main area displays Python code for defining a Gradio interface with a text input and output. Below the code, there's a button labeled 'interface.launch()' which has been executed. The status bar at the bottom right shows a green checkmark and the text '0s completed at 5:03 PM'. The top navigation bar includes options like File, Edit, View, Insert, Runtime, Tools, Help, and a 'Comment' section.

**Timestamp: 541.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[11] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                             outputs = "text")
```

(x) interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`

▶ Executing (0s) C → laun → setup\_tr → url requ → urlo → op → op → call ch → https o → do o → getrespo → he → read st → readl → recv i → rea → X

**Timestamp: 542.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 543.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

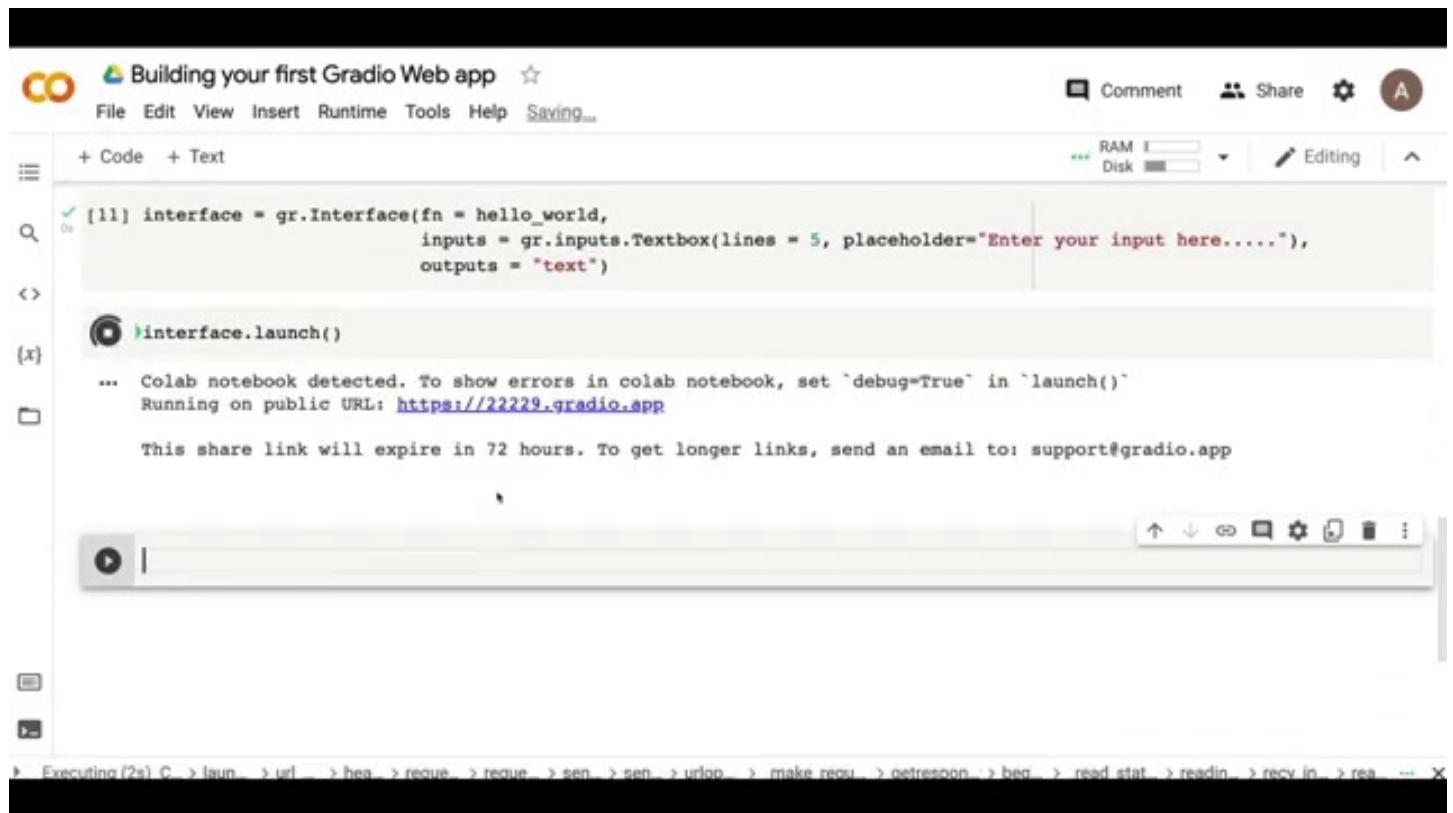
def greet(name):
    return "Hello " + name + "!"

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 544.00 seconds**



**Timestamp: 545.00 seconds**



**Timestamp: 546.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share A

+ Code + Text

RAM Disk Editing

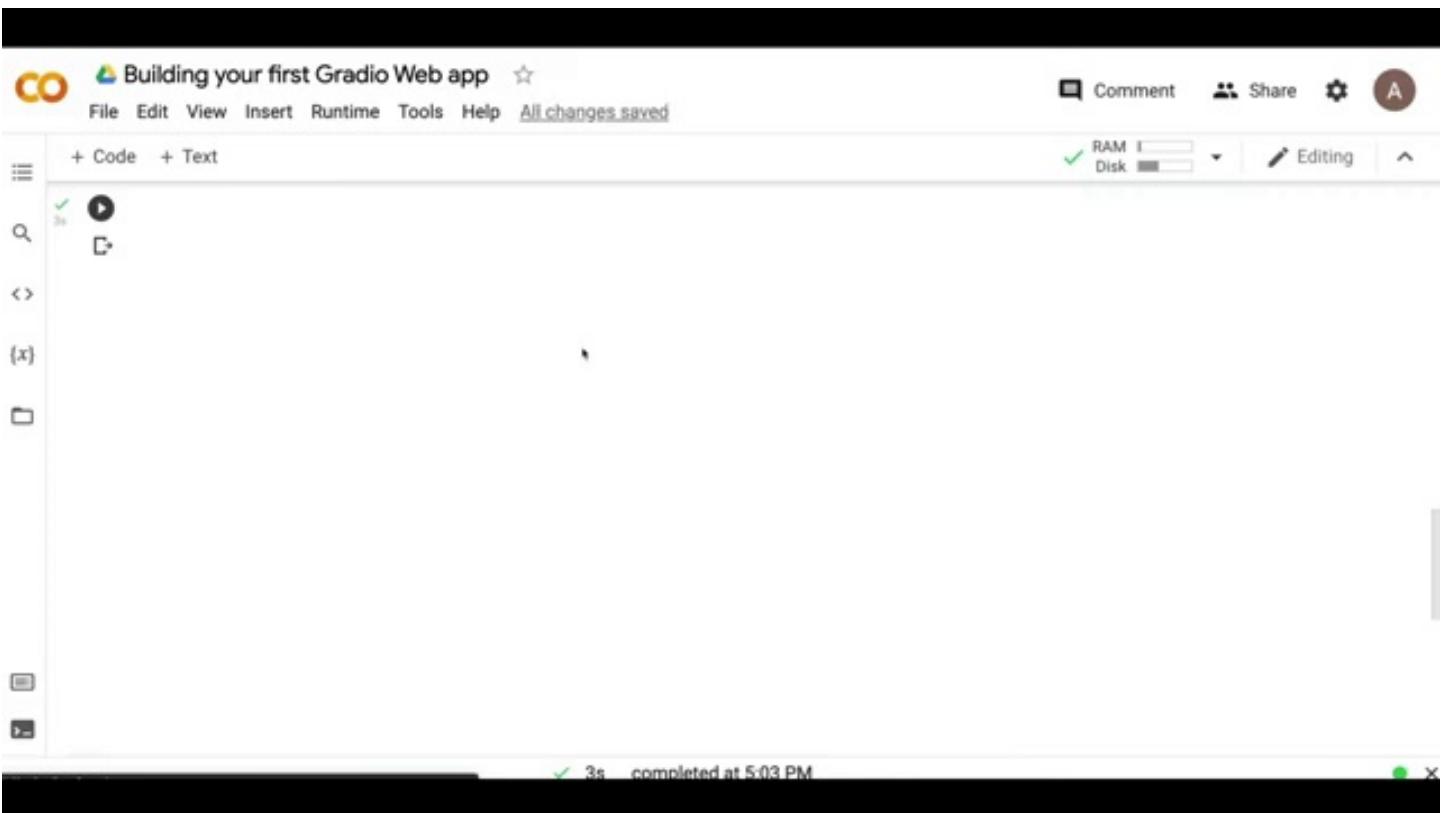
Code Editor Area:

```
(<Flask 'gradio.networking">,
'http://127.0.0.1:7861/',
'https://22229.gradio.app')
```

Run Button |

3s completed at 5:03 PM

**Timestamp: 547.00 seconds**



**Timestamp: 548.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

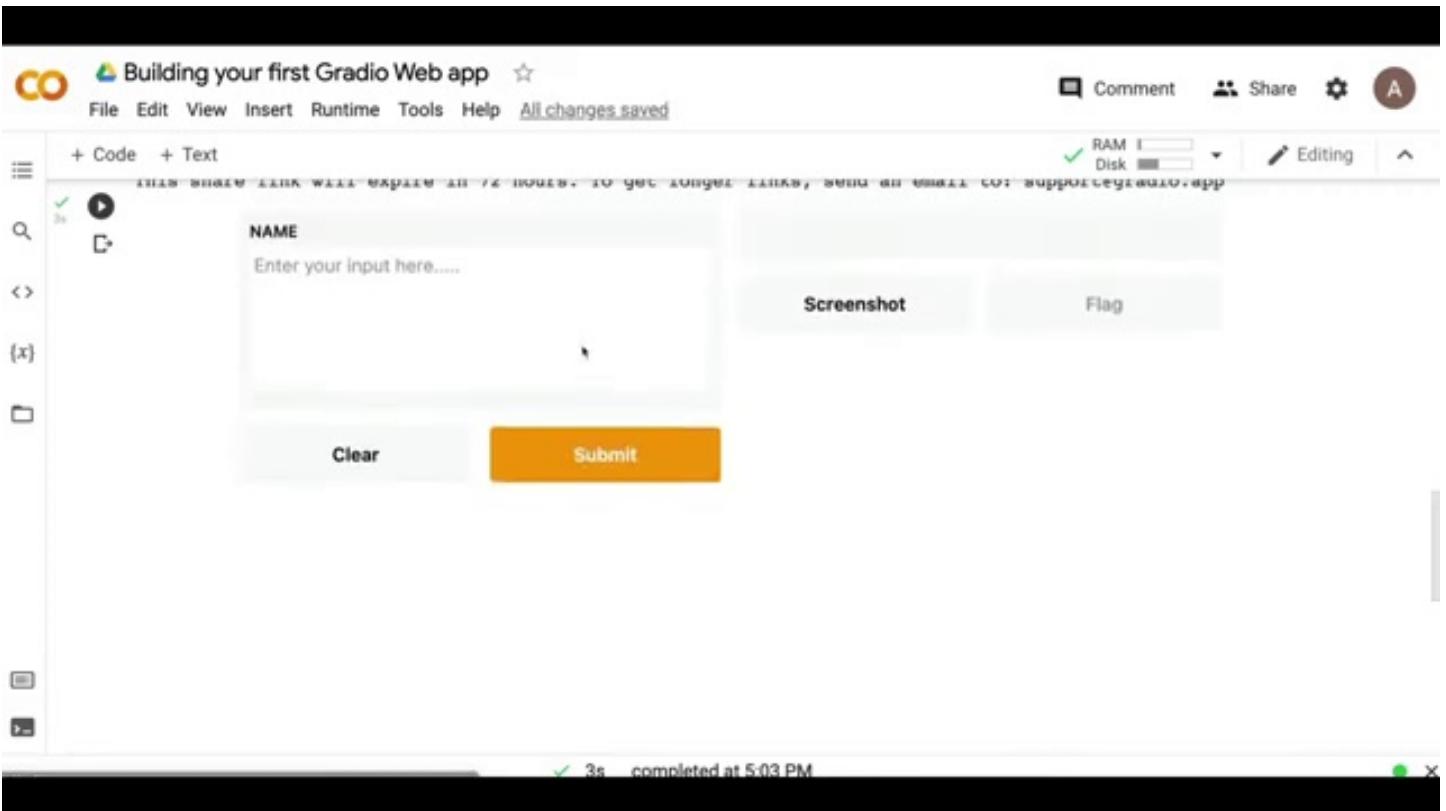
RAM Disk Editing

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 549.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[11] inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here...."), outputs = "text")
```

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

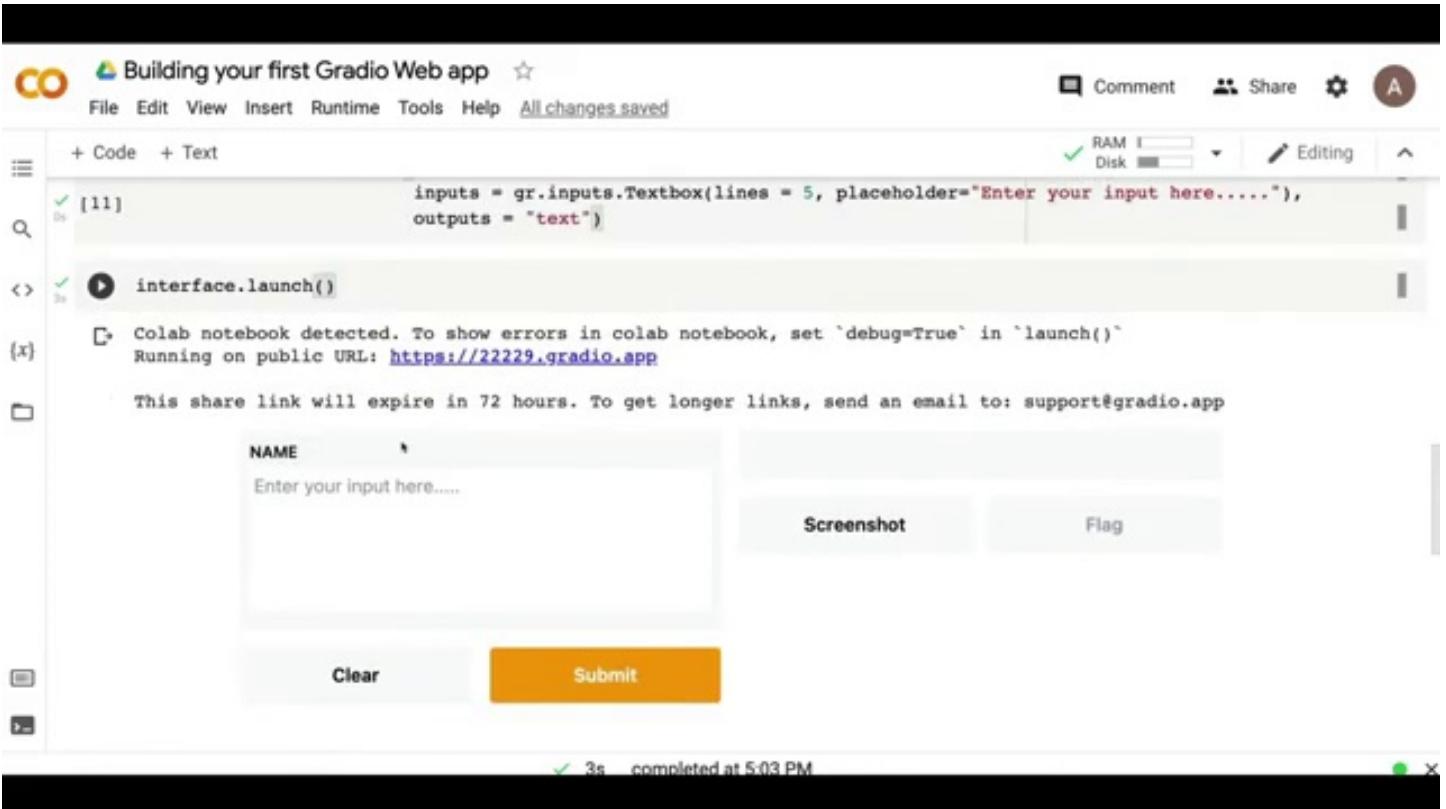
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 550.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[11] inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here...."), outputs = "text")
```

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

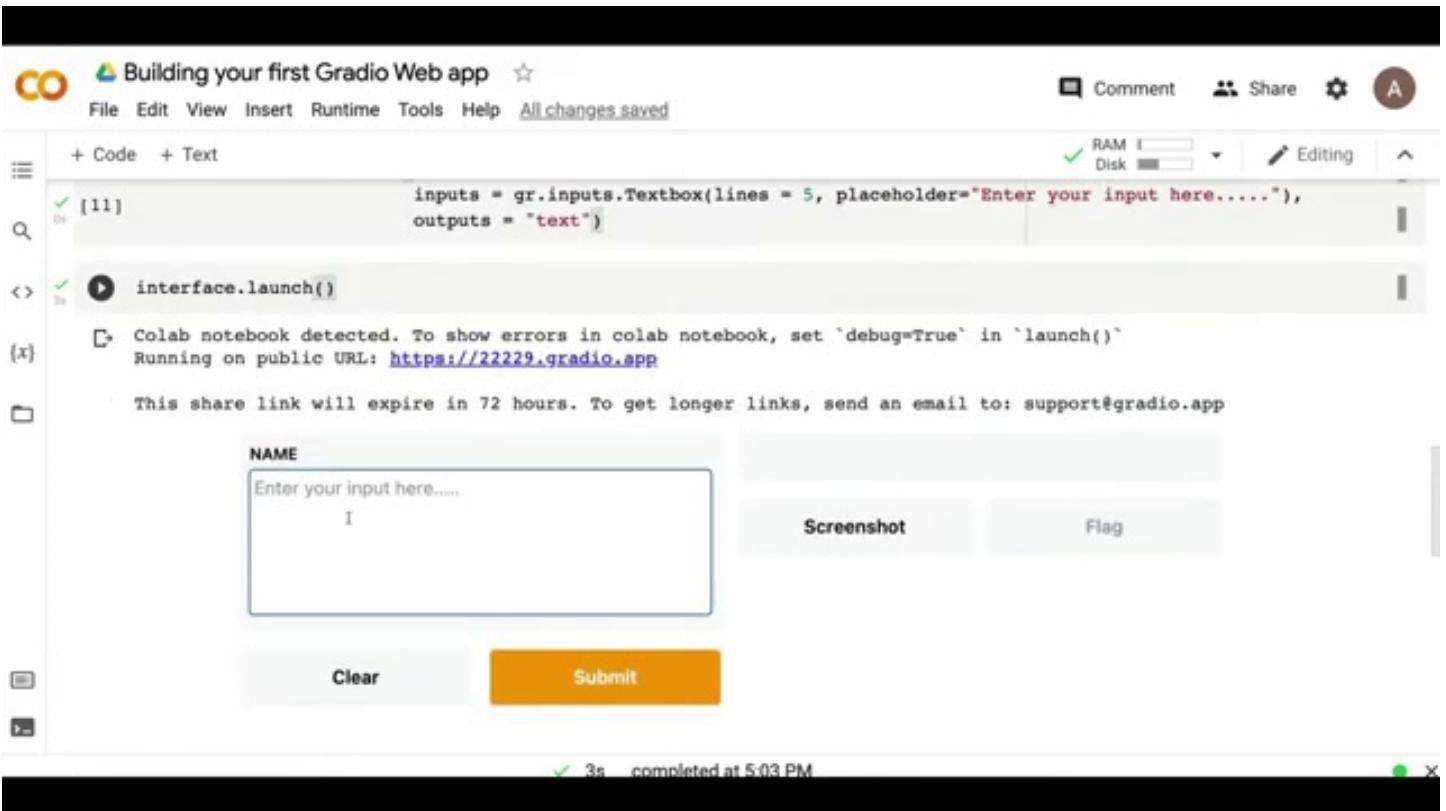
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 551.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[11] inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here...."), outputs = "text")
```

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

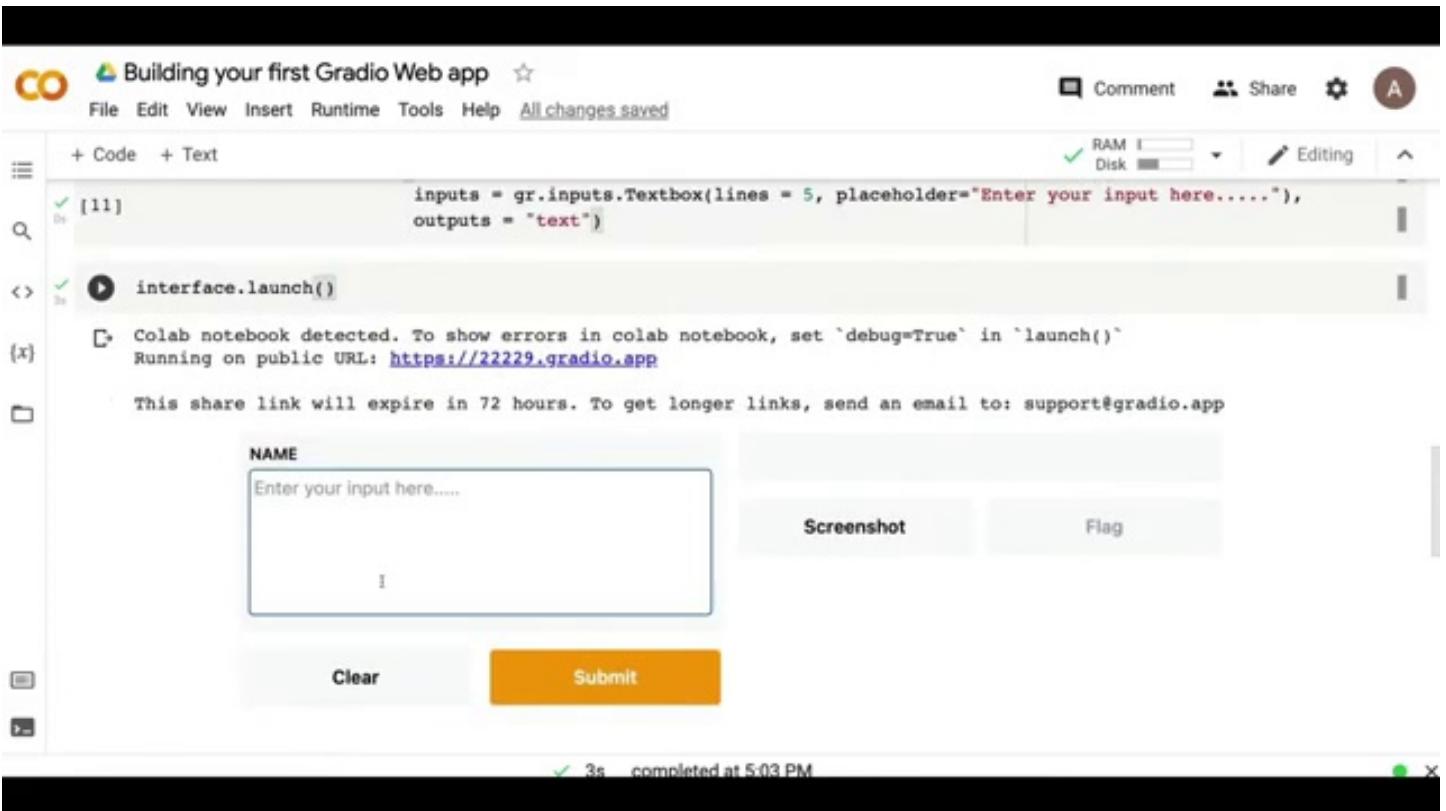
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 552.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[11] inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here...."), outputs = "text"
```

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

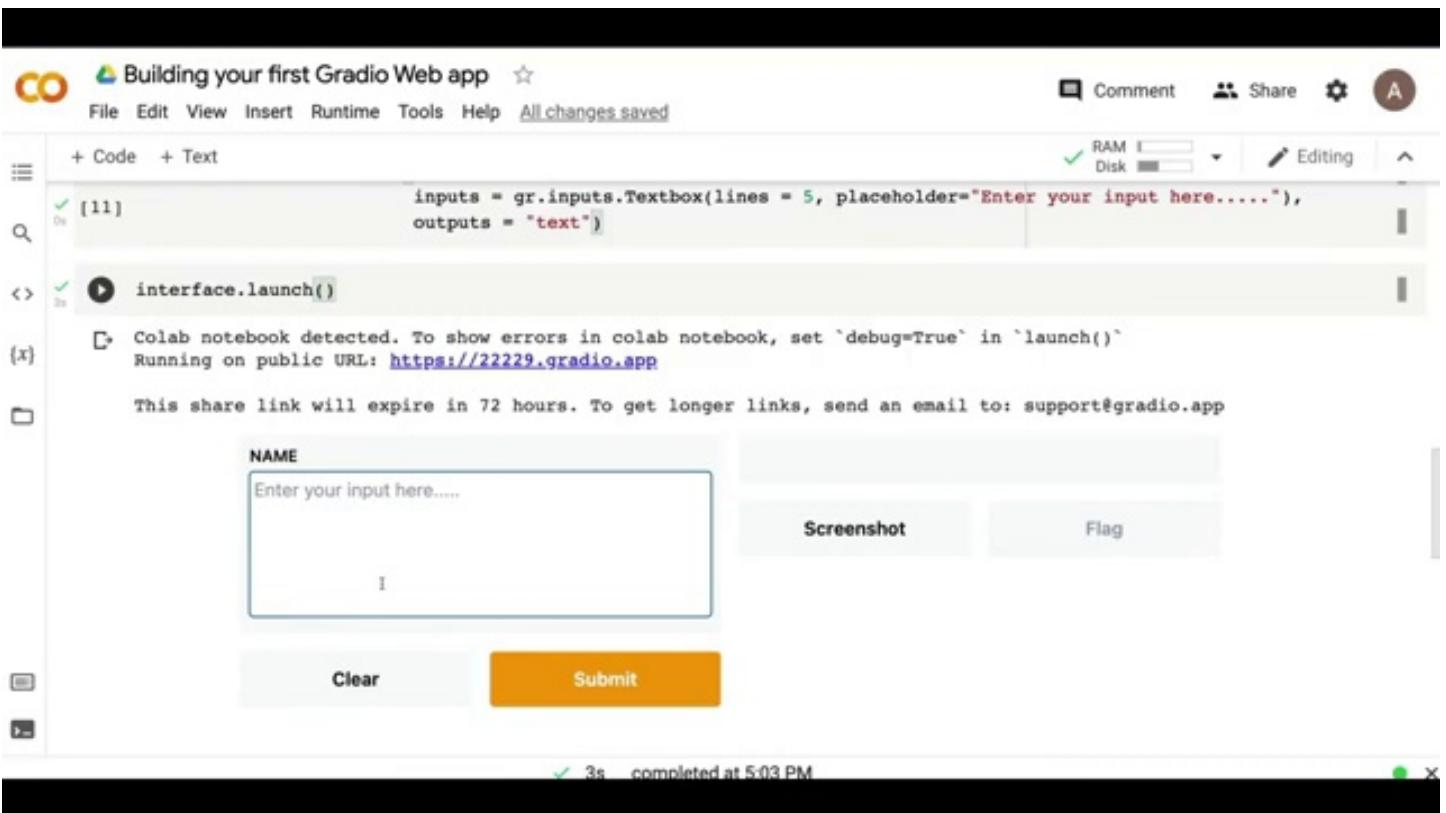
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 553.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://j1668.gradio.app>

RAM Disk Editing

```
[11] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 5, placeholder="Enter your input here....."),
                             outputs = "text")
```

(x) interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

3s completed at 5:03 PM



**Timestamp: 554.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 555.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://j1668.gradio.app>

RAM Disk Editing

Q interface = gr.Interface(fn = hello\_world,  
inputs = gr.inputs.Textbox(lines = 1, placeholder="Enter your input here....."),  
outputs = "text")

(x) [13] interface.launch()

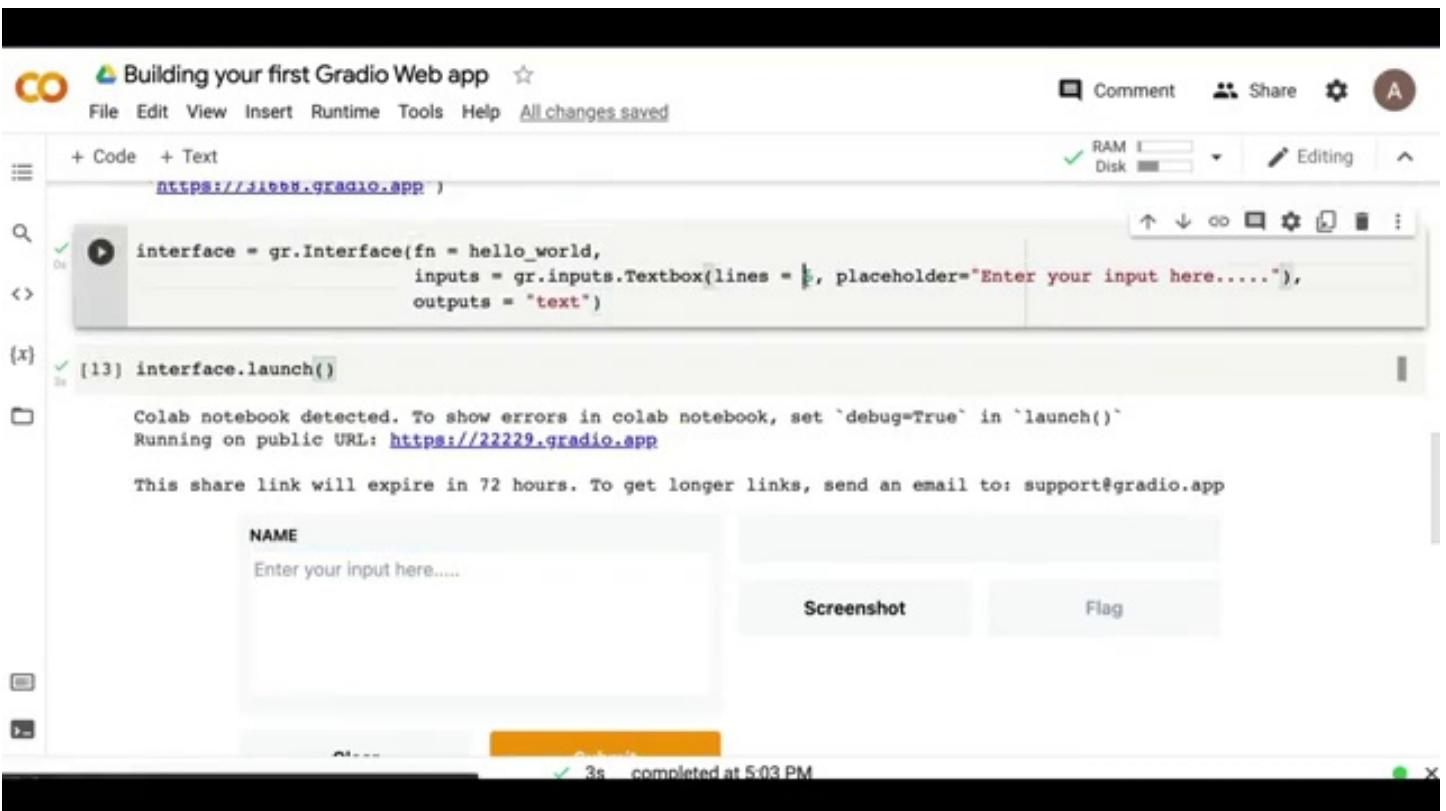
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

3s completed at 5:03 PM



**Timestamp: 556.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://j1668.gradio.app>

RAM Disk Editing

```
interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                         outputs = "text")
```

(x) [13] interface.launch()

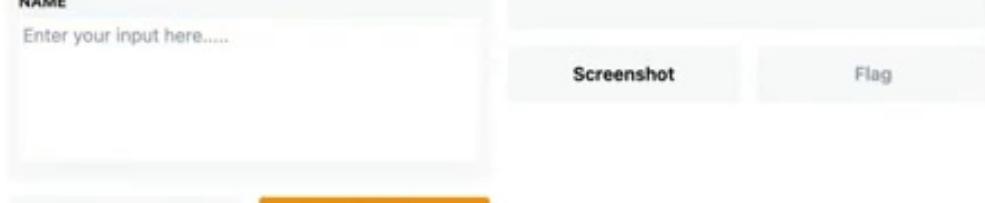
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://22229.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 557.00 seconds**



**Timestamp: 558.00 seconds**



**Timestamp: 559.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
interface.launch()
...
Colab notebook detected. To show errors in colab notebook, set 'debug=True' in 'launch()'
```

{x}

RAM Disk Editing

```
> Executing (1s) C > ls > setup > url r > url > o > ... > call > https > do > m > send > endh > send > s > co > wrin > cr > do hand ... X
```

**Timestamp: 560.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

```
interface = gr.Interface(fn = hello_world,
    inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
    outputs = "text")
```

(x) interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

Executing (2s) Cell > launch() > setup\_tunnel() > create\_tunnel() > connect() > start\_client() > wait() > wait()

**Timestamp: 561.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
interface = gr.Interface(fn = hello_world,
                         inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                         outputs = "text")
```

+ Code + Text

```
interface.launch()
```

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

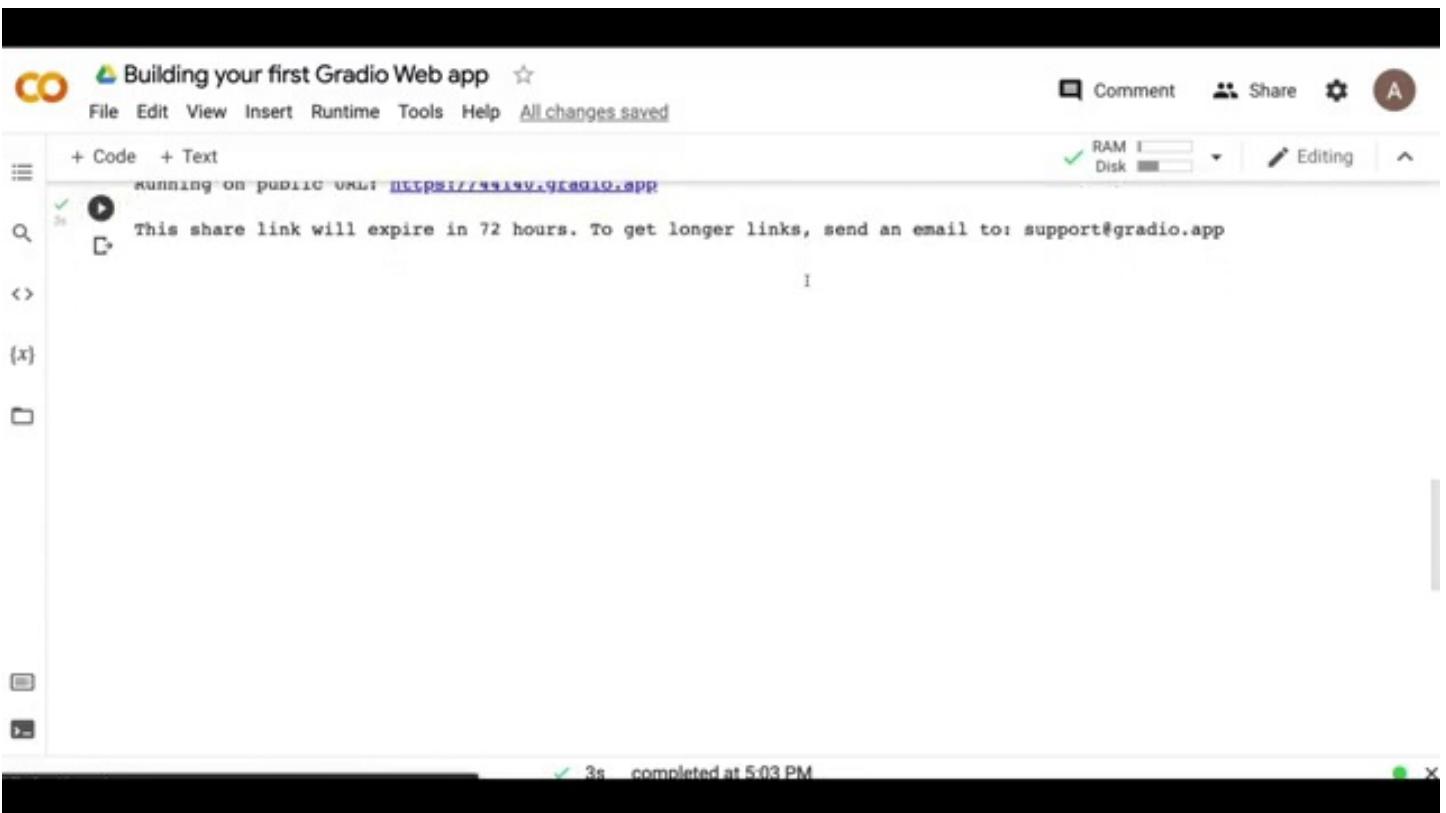
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

RAM Disk Editing

3s completed at 5:03 PM

The screenshot shows the Gradio web application interface. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. Below the navigation is a toolbar with icons for 'Comment', 'Share', 'Settings', and a user profile. The main area has tabs for '+ Code' and '+ Text', with '+ Code' currently selected. The code block contains Python code for creating a Gradio interface with a text input and text output. A note indicates that a Colab notebook is detected, and it provides a public URL for the app. A message at the bottom says the share link will expire in 72 hours. The bottom status bar shows '3s completed at 5:03 PM'.

**Timestamp: 562.00 seconds**



**Timestamp: 563.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Running on public URL: <https://44140.gradio.app>

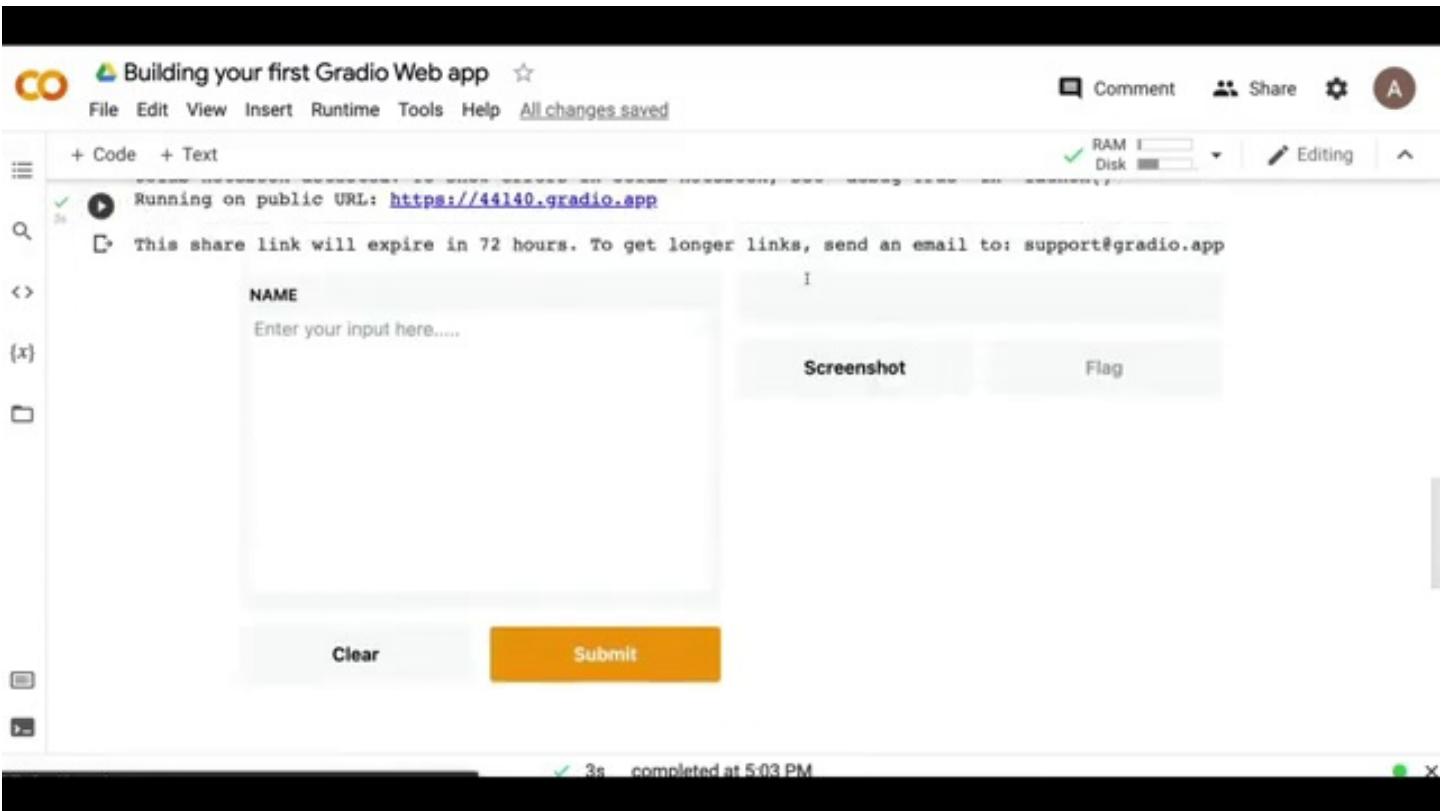
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 564.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
  'http://127.0.0.1:7860/',
  'https://31668.gradio.app')

interface = gr.Interface(fn = hello_world,
                        inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                        outputs = "text")

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 565.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

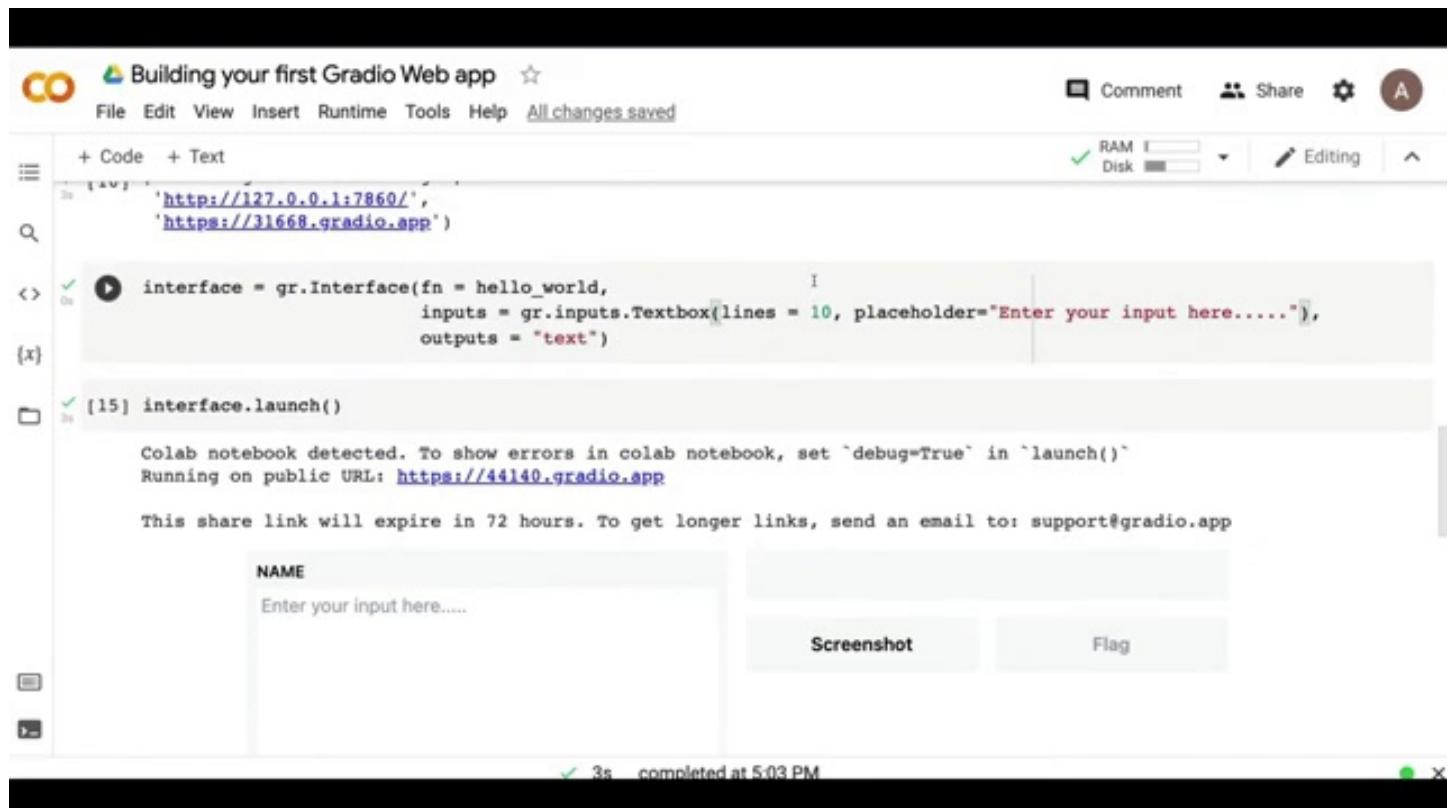
What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 566.00 seconds**



**Timestamp: 567.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
 3a  'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

<>  interface = gr.Interface(fn = hello_world,
    inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
    outputs = "text")

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 568.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
 3a  'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

<>  interface = gr.Interface(fn = hello_world,
    inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
    outputs = "text")

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

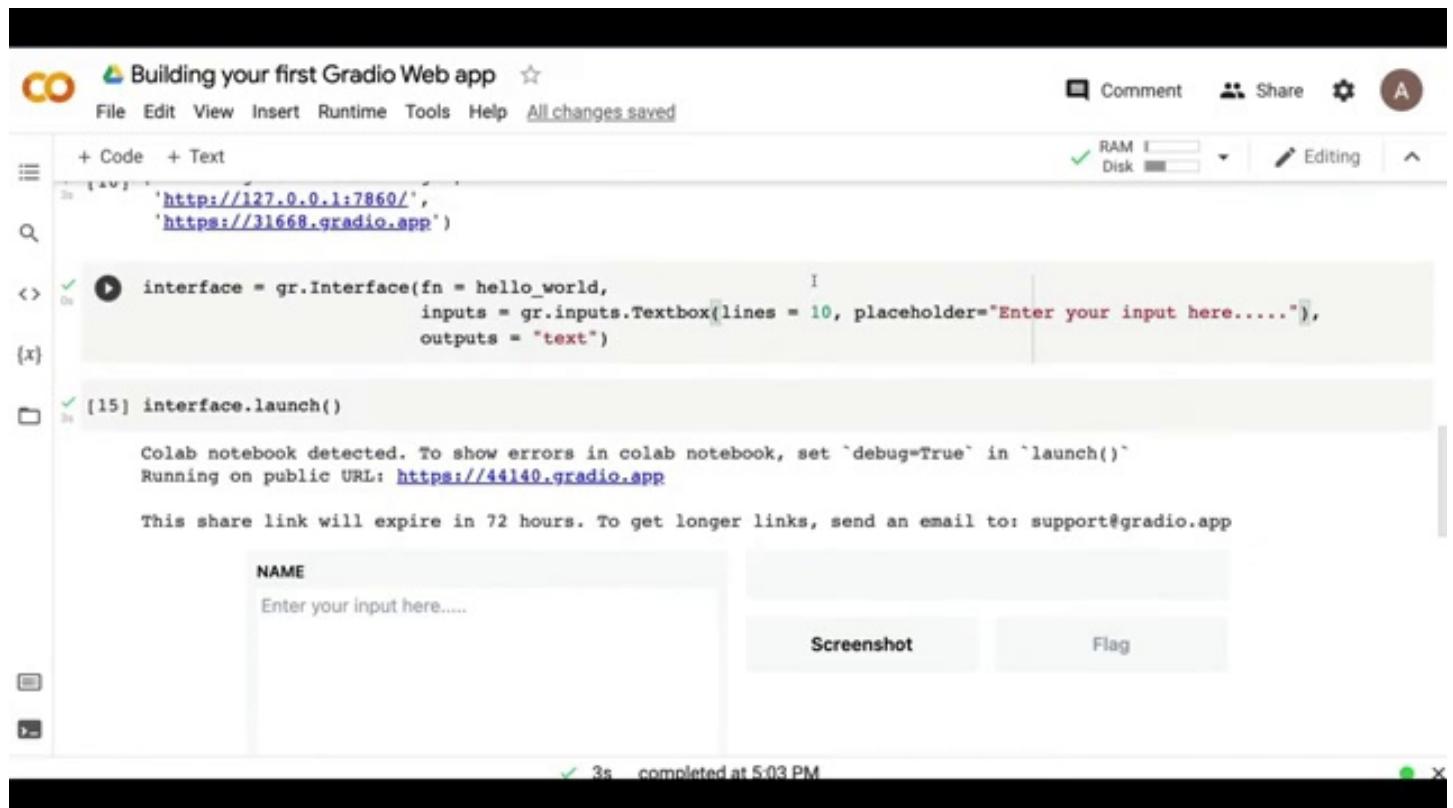
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 569.00 seconds**



**Timestamp: 570.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[15] interface.launch()

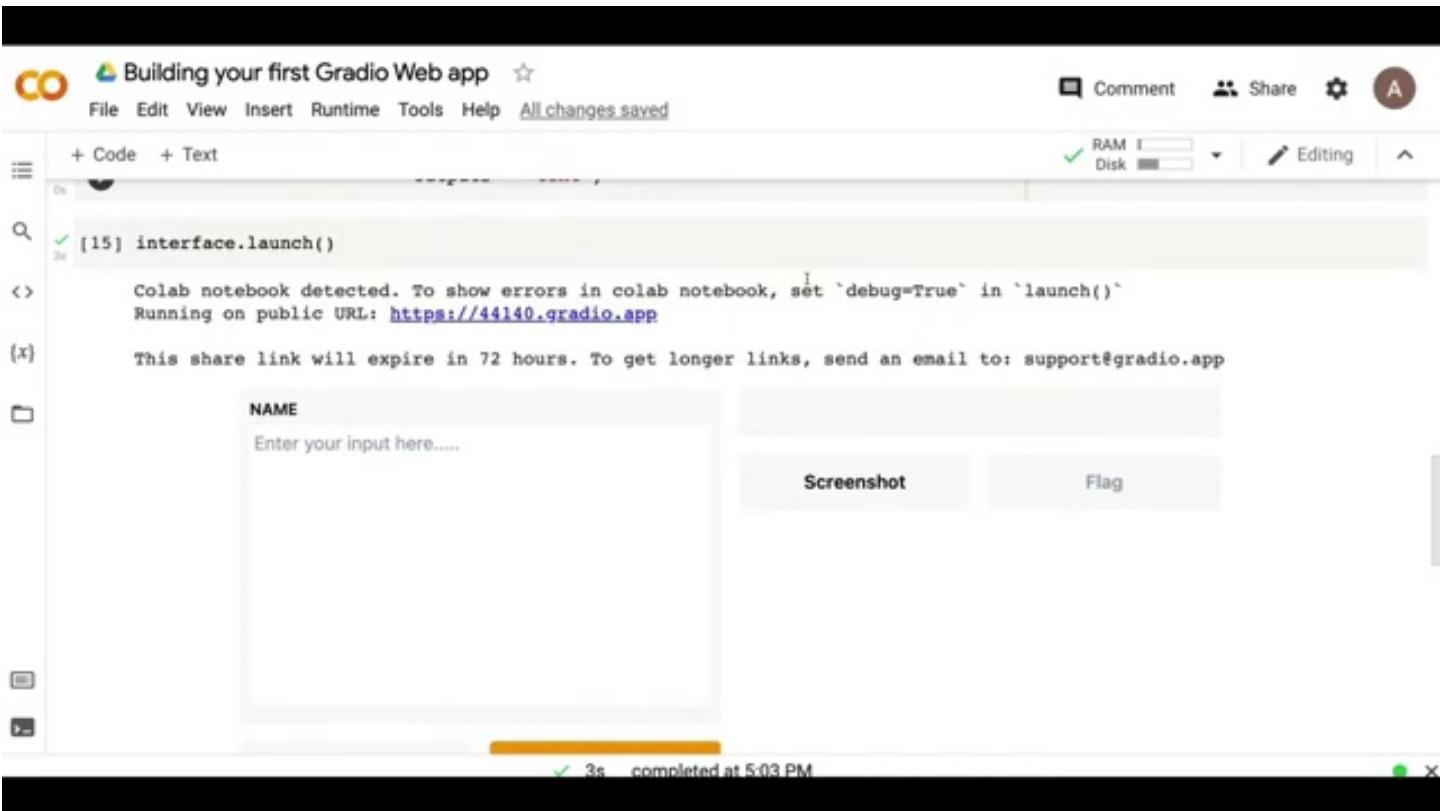
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here.....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 571.00 seconds**

 Building your first Gradio Web app star

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ interface.launch()

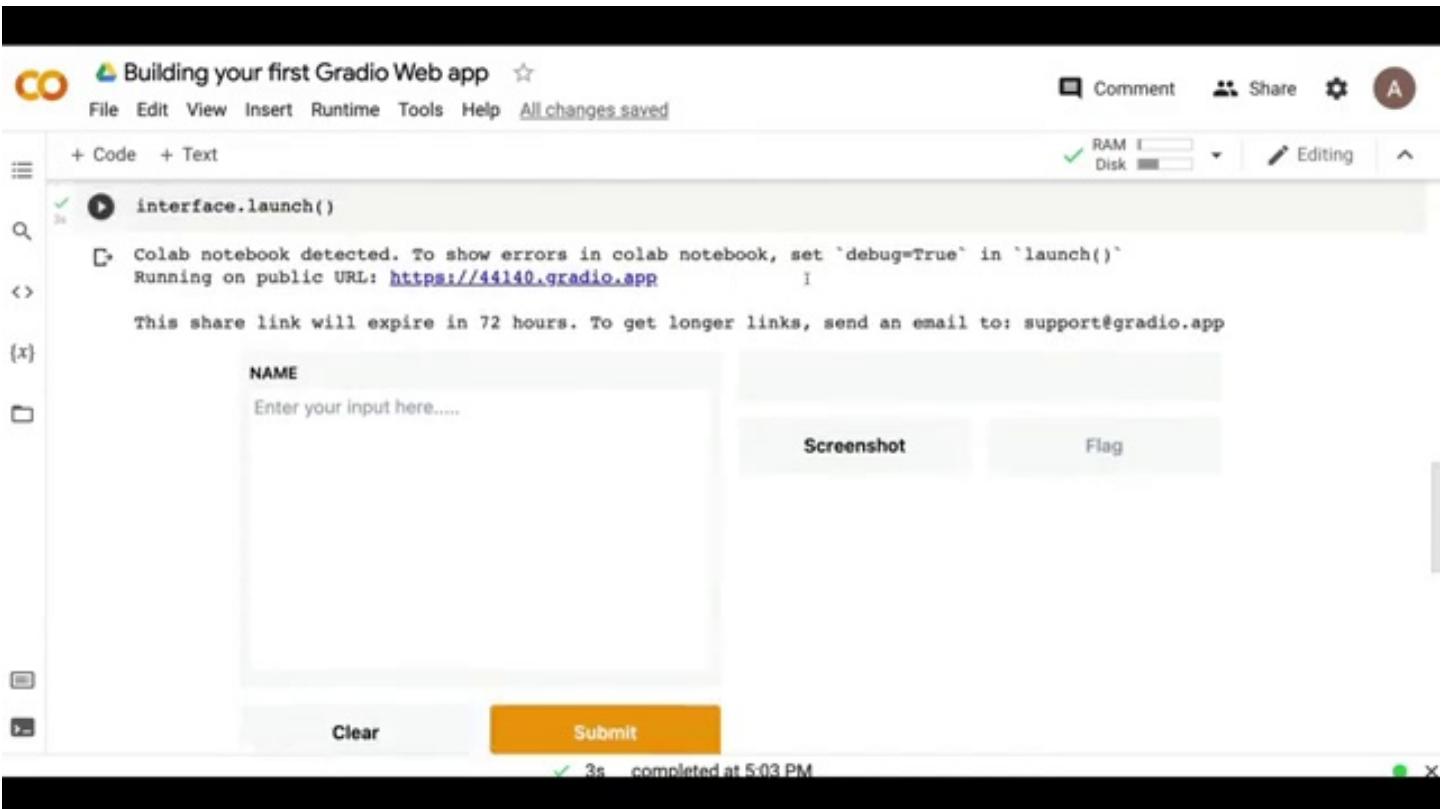
Colab notebook detected. To show errors in colab notebook, set "debug=True" in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

(x) NAME  
Enter your input here....

Screenshot Flag

Clear Submit ✓ 3s completed at 5:03 PM X



**Timestamp: 572.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[10] (<Flask 'gradio.networking'>,
      'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")

interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 573.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch()

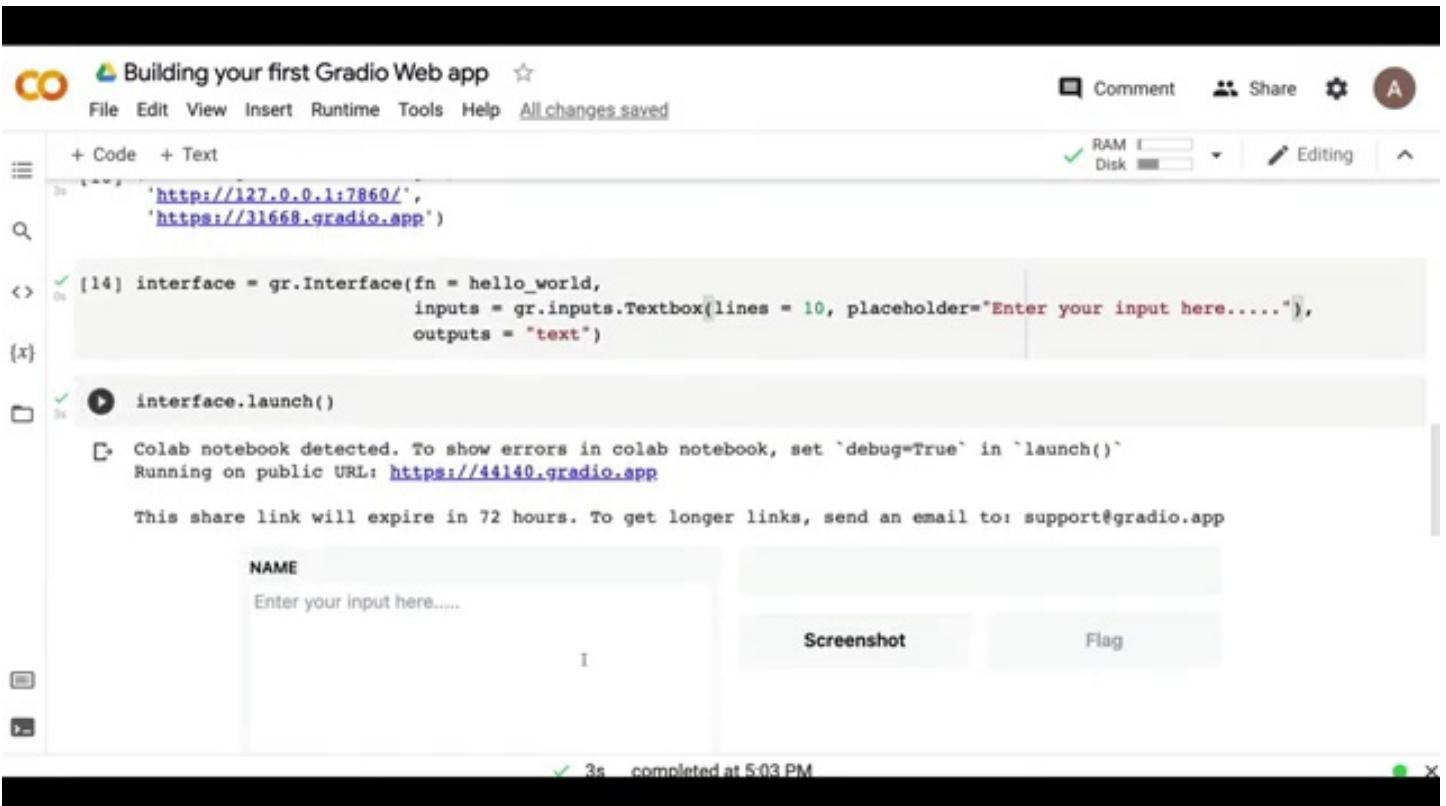
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 574.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch()

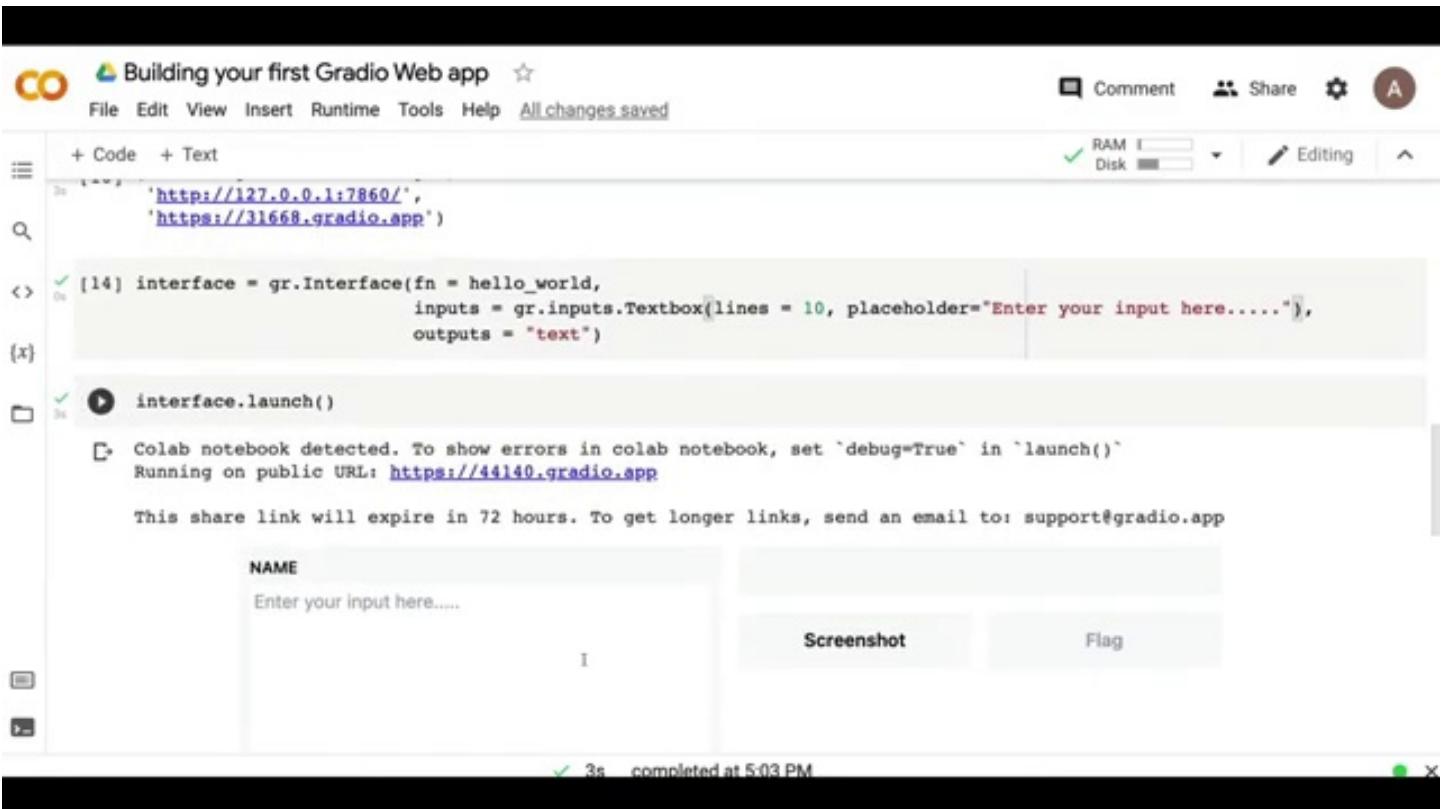
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 575.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch()

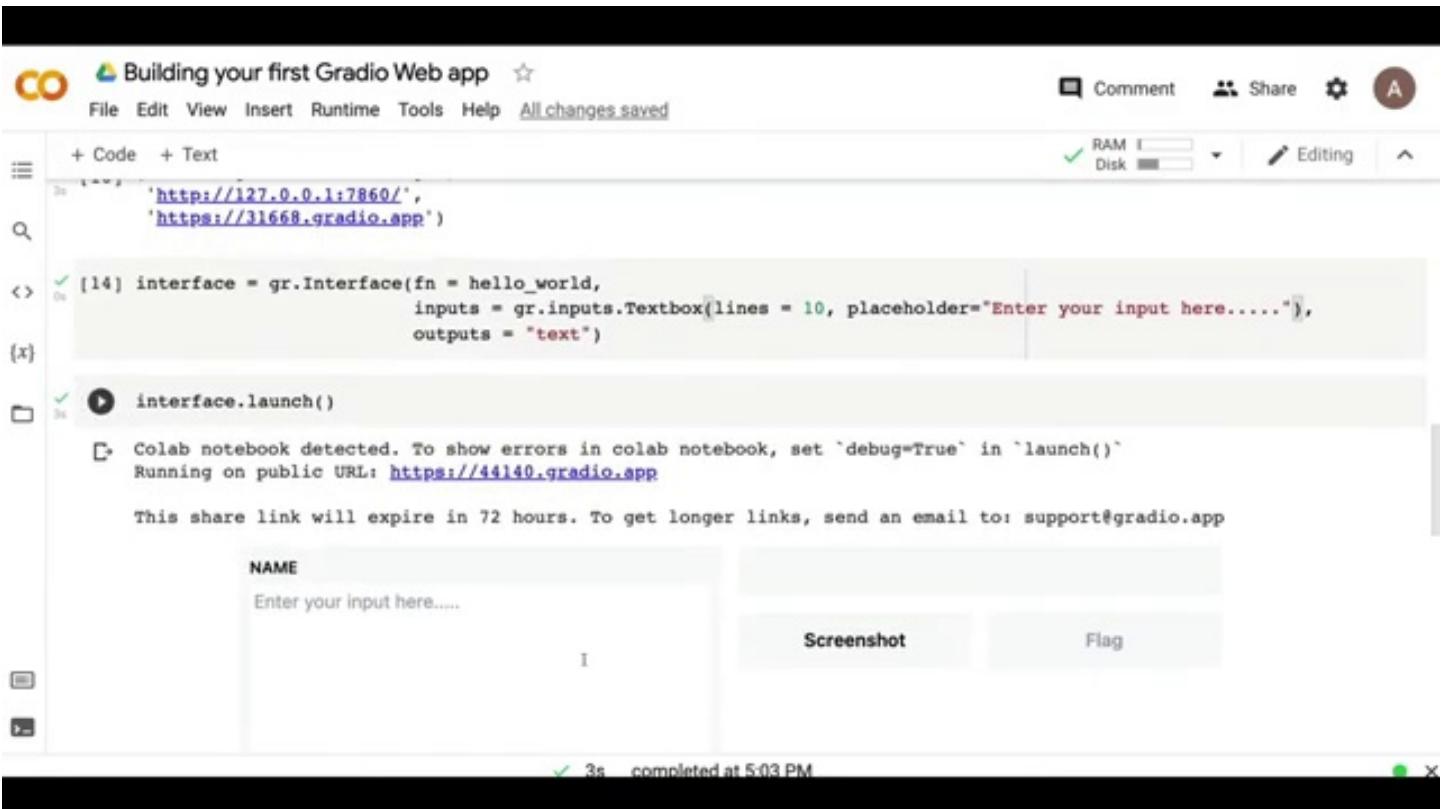
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 576.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 577.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
3s 14] 'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

<> [14] interface = gr.Interface(fn = hello_world,
                                inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                                outputs = "text")

{x} 3s interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 578.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
3s 14] 'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

<> [14] interface = gr.Interface(fn = hello_world,
                                inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                                outputs = "text")

{x} 3s interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 579.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share ⚙ A

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

interface.launch()

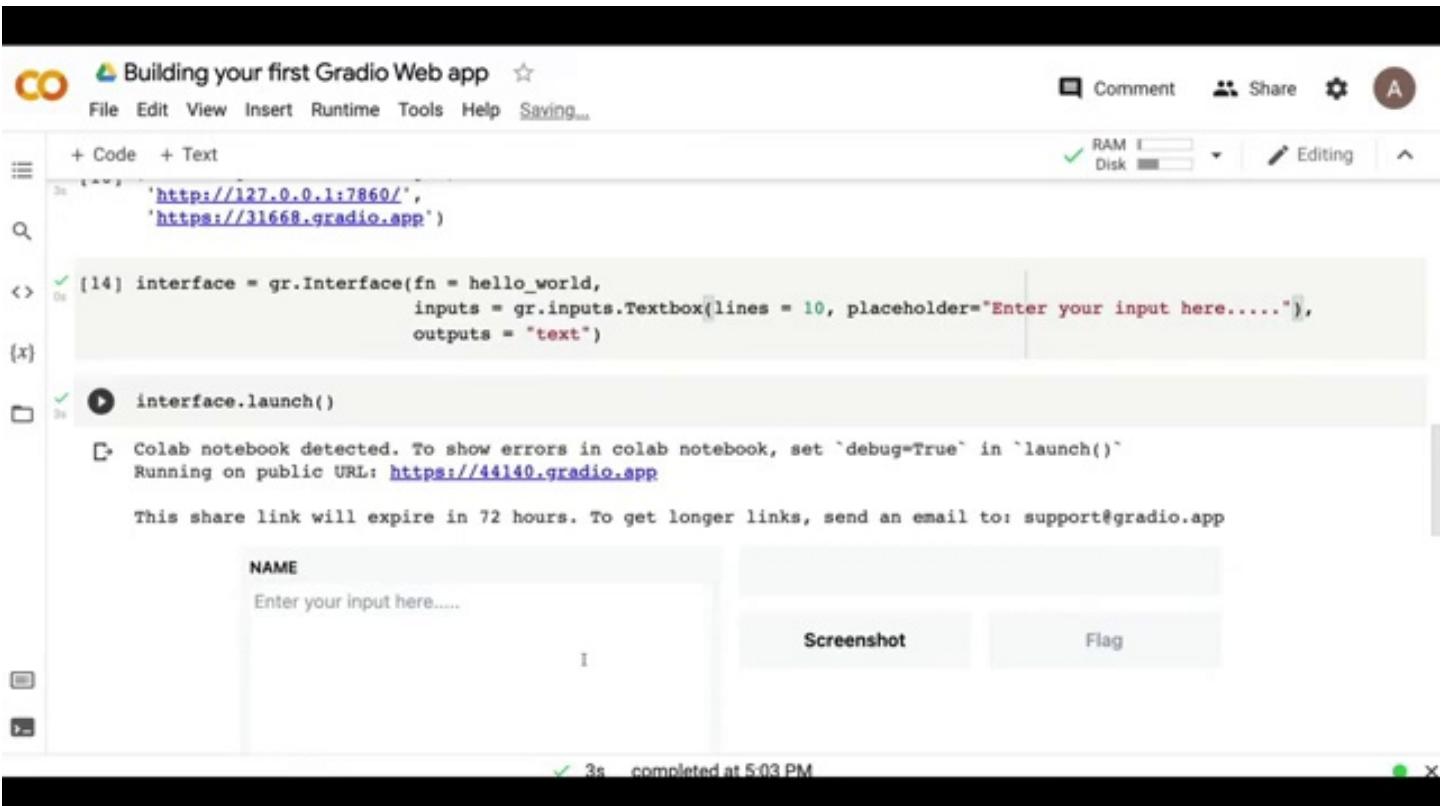
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 580.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

Comment Share A

+ Code + Text

RAM Disk Editing

```
[3]  'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 581.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[3]  'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 582.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

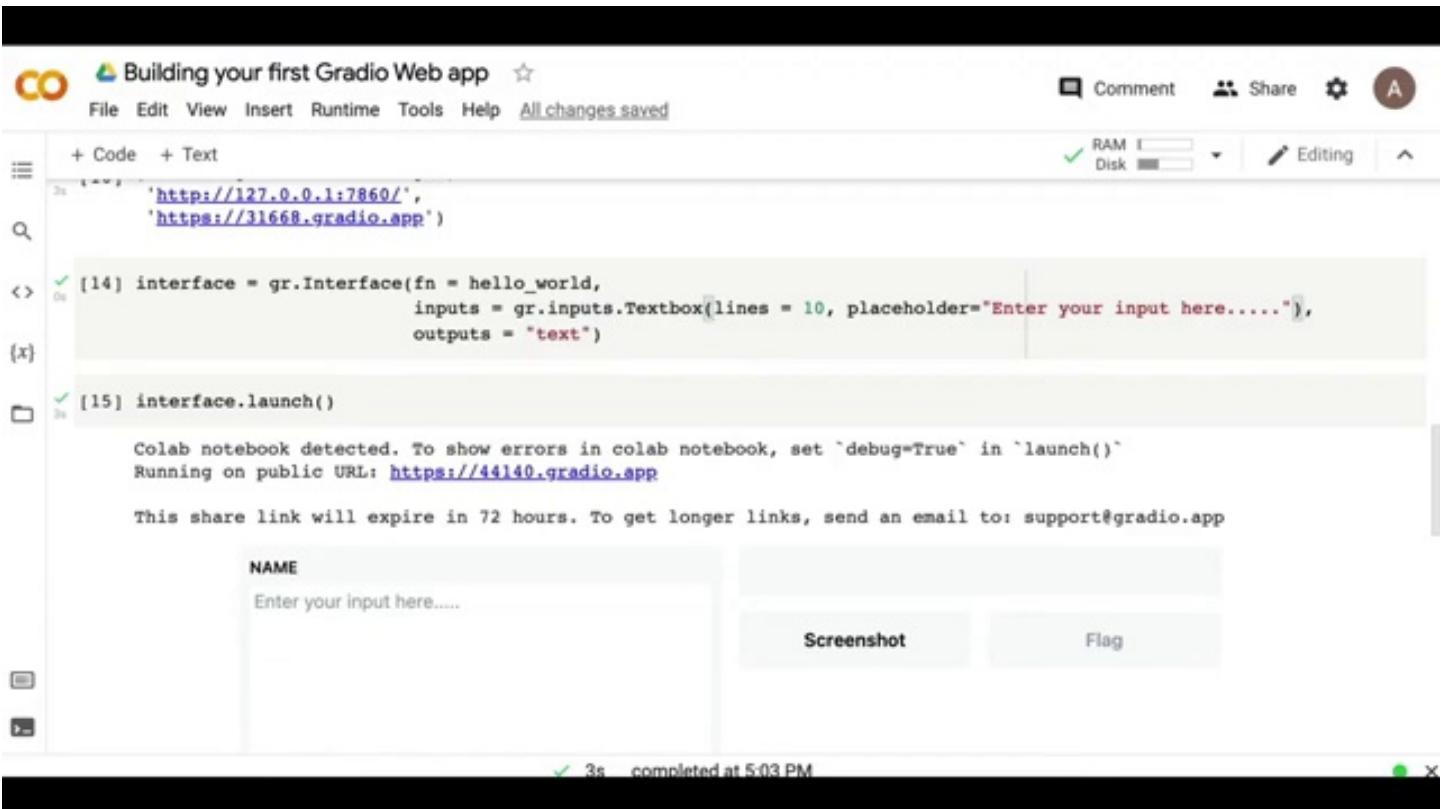
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 583.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

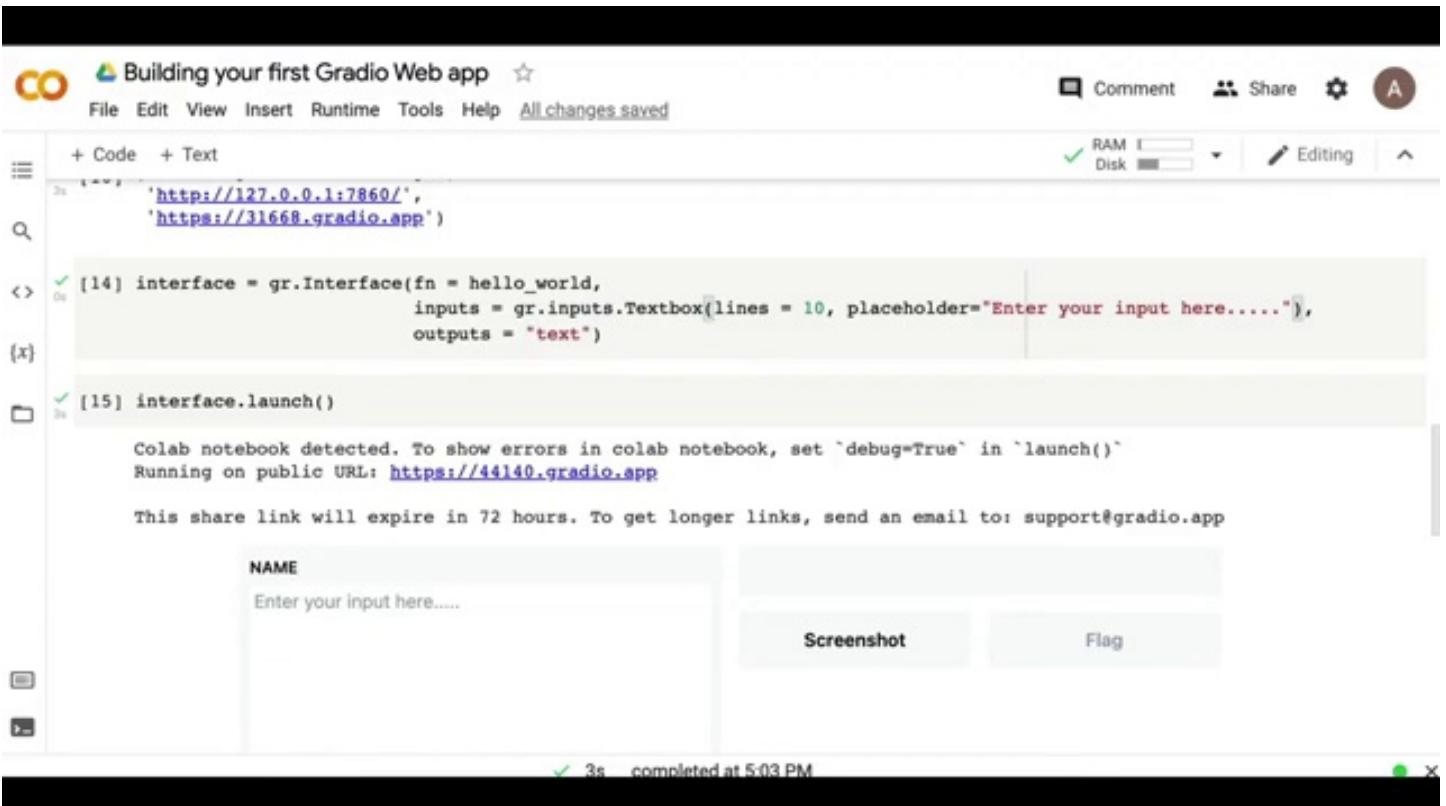
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 584.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

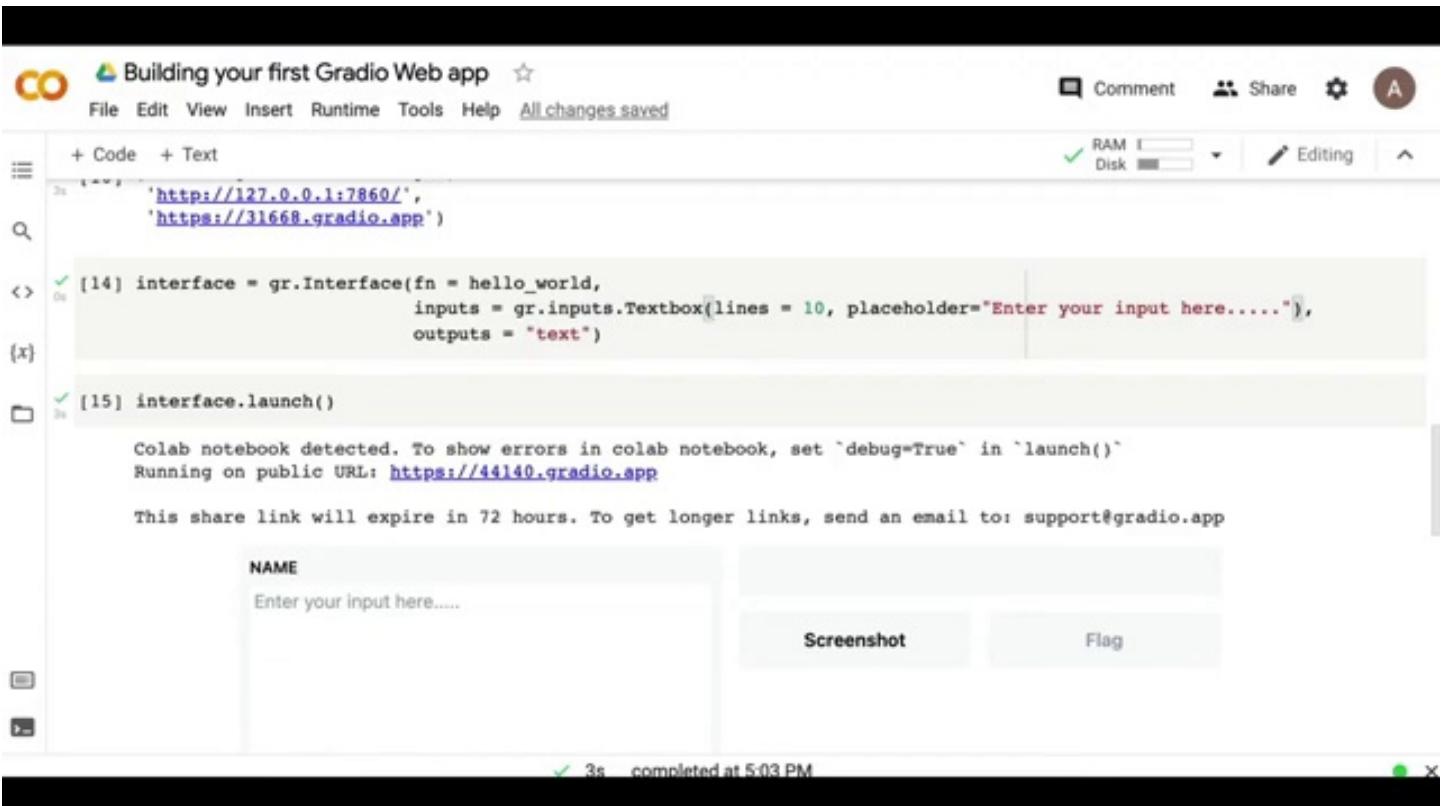
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 585.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

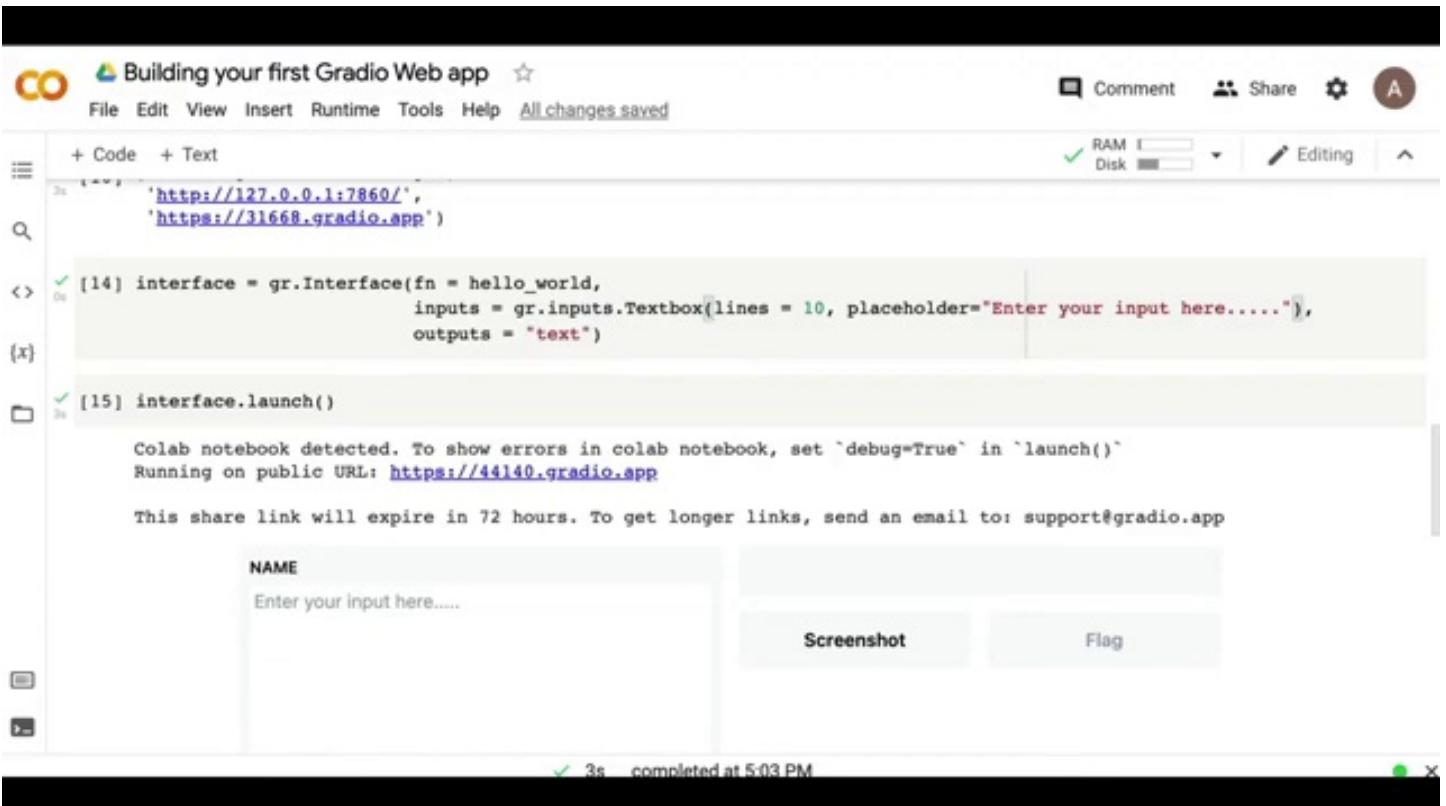
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 586.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

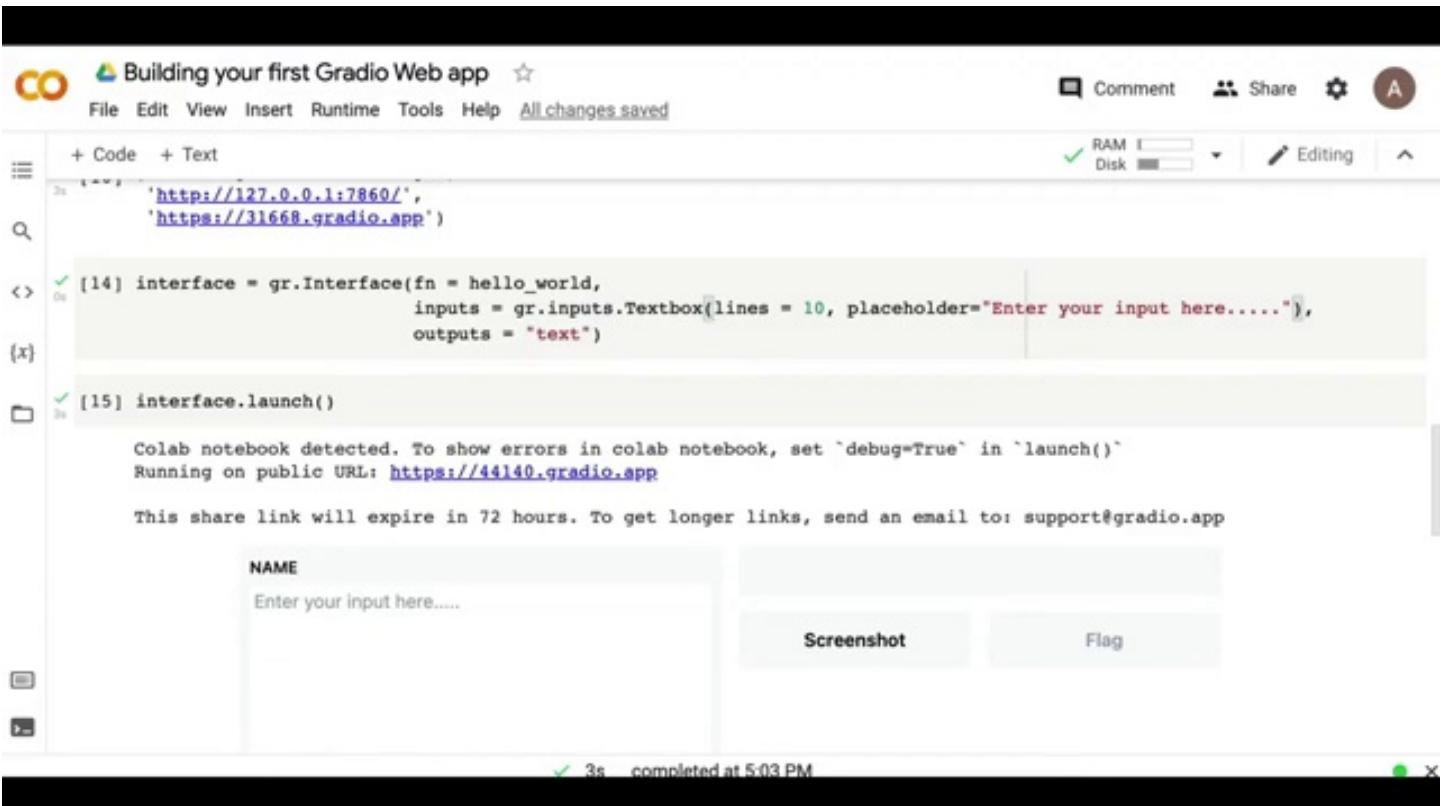
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 587.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn: the function to wrap`
- `inputs: the input component type(s)`
- `outputs: the output component type(s)`

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 588.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

```
[15] interface.launch()
```

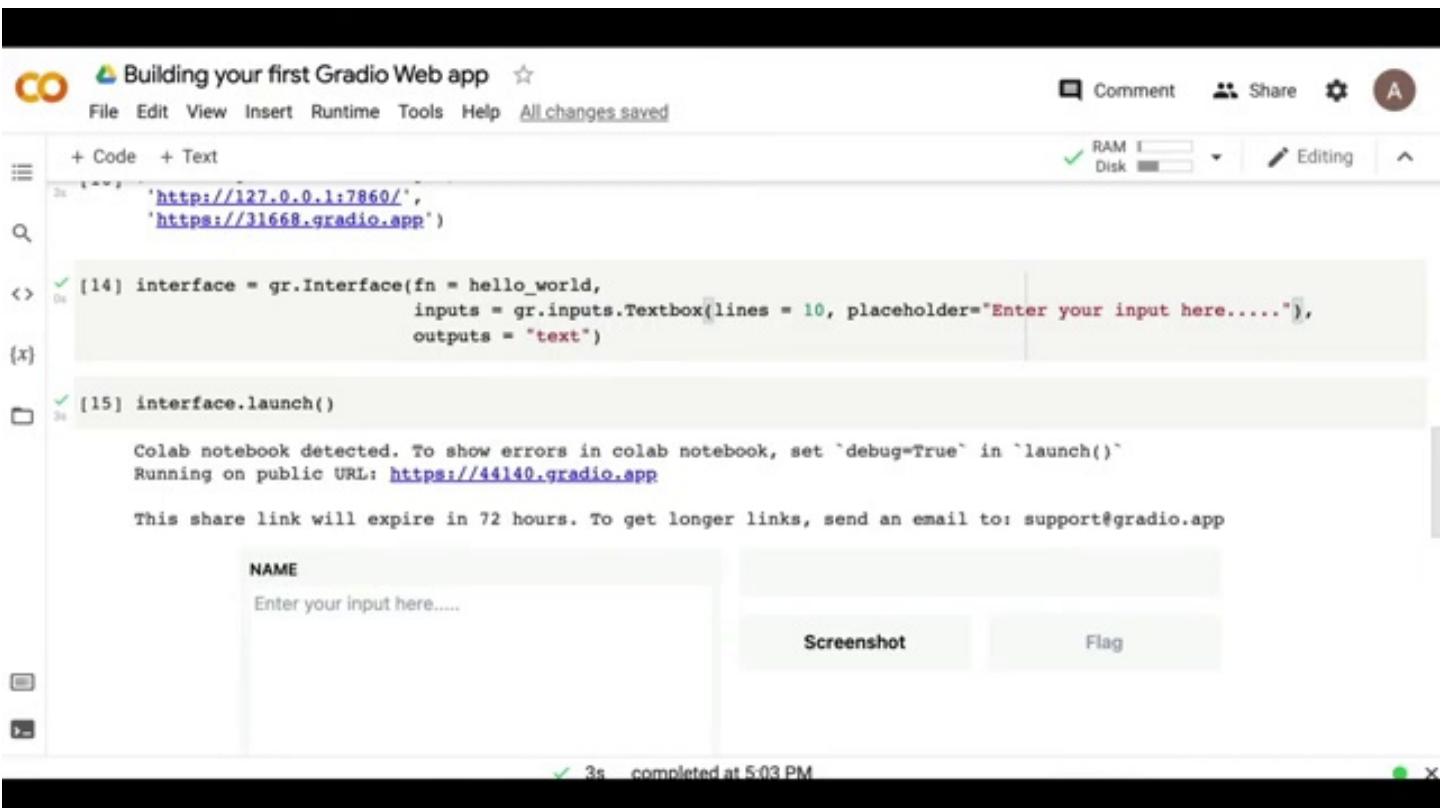
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 589.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

```
[15] interface.launch()
```

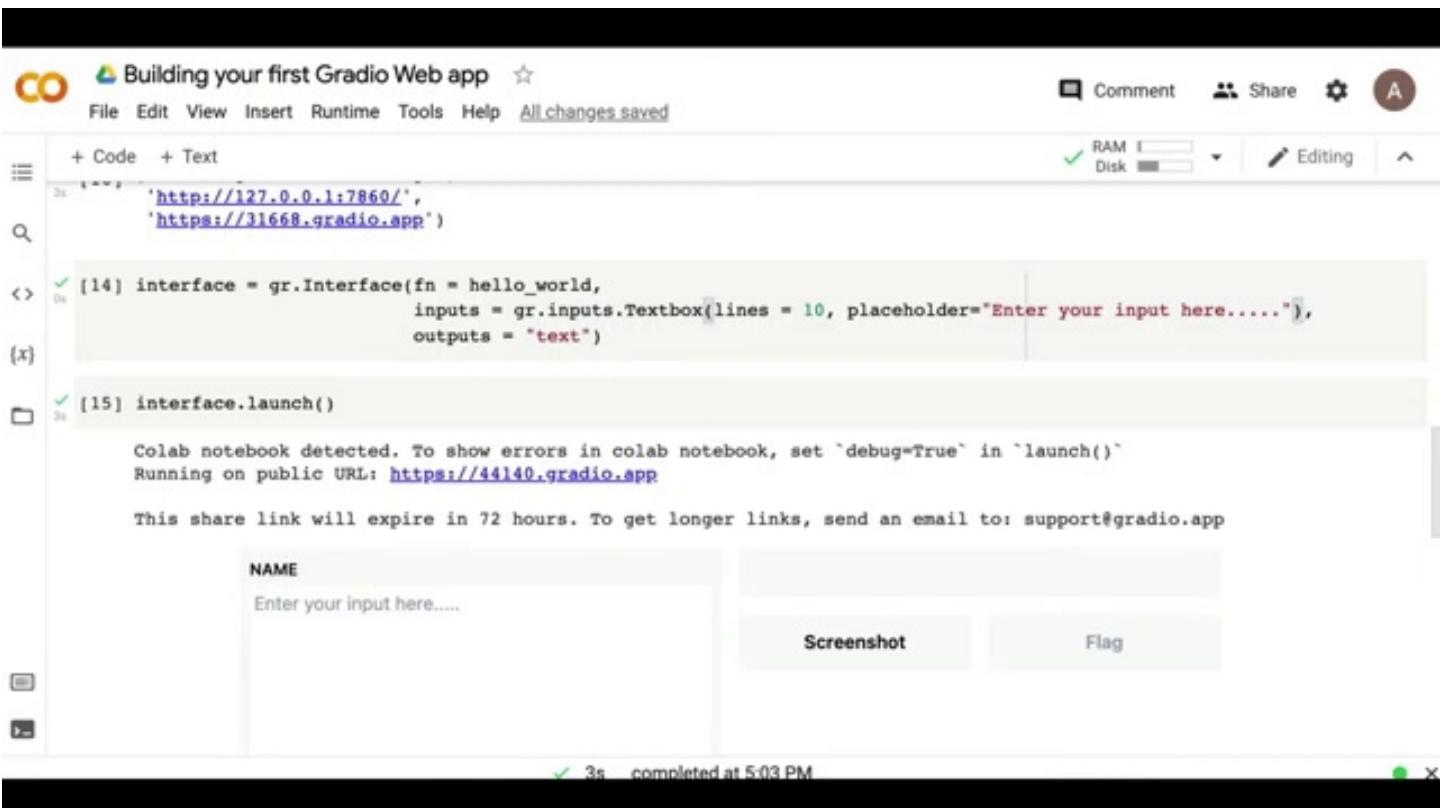
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 590.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

The screenshot shows a Jupyter Notebook cell with the following code:

```
[14] interface = gr.Interface(fn = hello_world,  
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),  
                             outputs = "text")  
  
[15] interface.launch()  
  
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: https://44140.gradio.app
```

The output of the cell includes a message about Colab detection, the public URL, and a note about share link expiration.

Below the code, there is a form with a placeholder "NAME" and a text input field labeled "Enter your input here....". There are two buttons: "Screenshot" and "Flag".

At the bottom, a progress bar indicates the task was completed in 3 seconds at 5:03 PM.

**Timestamp: 591.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

```
[15] interface.launch()
```

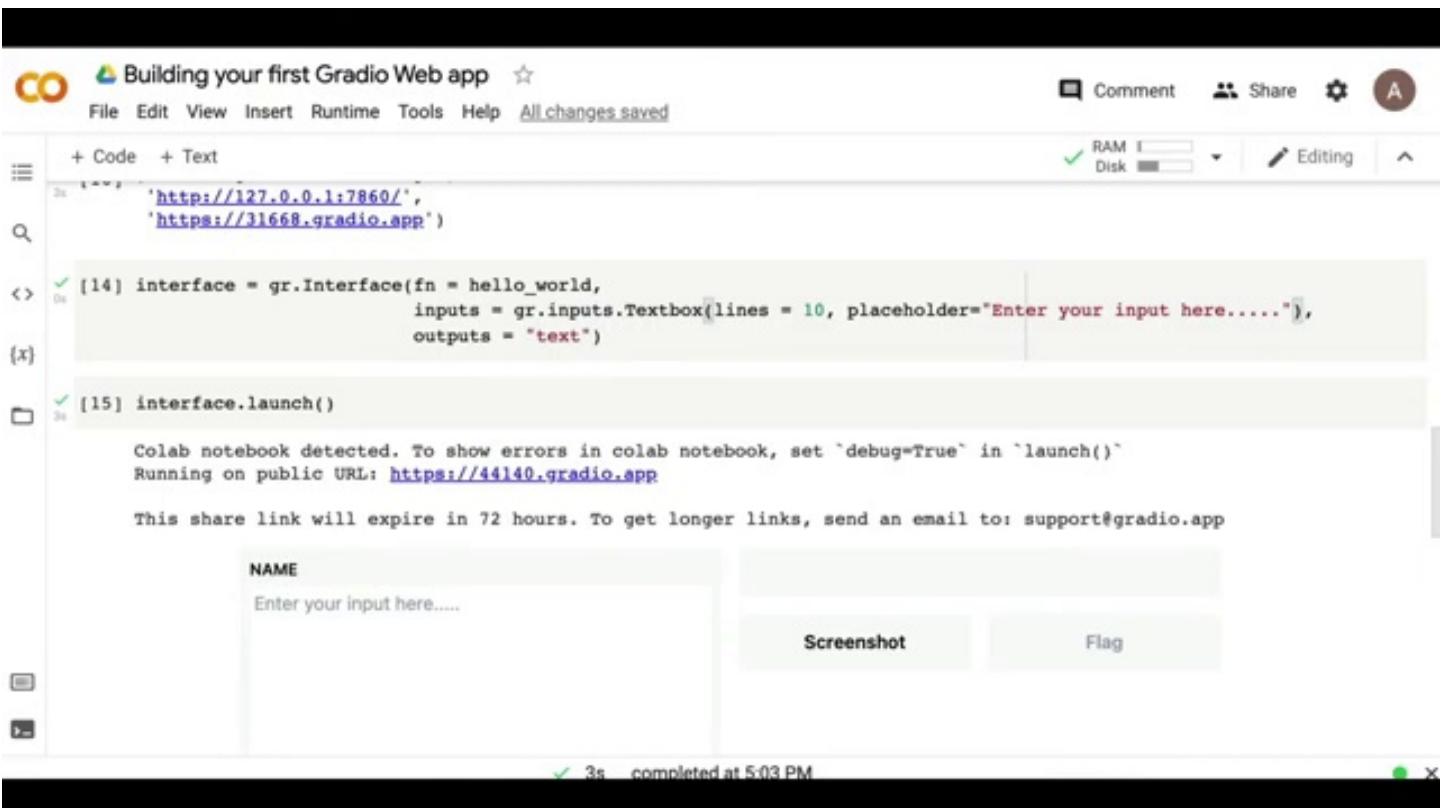
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 592.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

The screenshot shows a Jupyter Notebook cell with the following code:

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

Line 15 contains the command `interface.launch()`. Below the code, a message indicates that the Colab notebook is detected and provides a public URL: <https://44140.gradio.app>. A note states that the share link will expire in 72 hours and provides an email address for longer links: support@gradio.app.

Below the code cell, there is a form with a placeholder "NAME" and a text input field containing "Enter your input here....". There are two buttons: "Screenshot" and "Flag". At the bottom of the cell, it says "✓ 3s completed at 5:03 PM".

**Timestamp: 593.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
3s 14] 'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

<> [14] interface = gr.Interface(fn = hello_world,
                                inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                                outputs = "text")

(x) 3s interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 594.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
http://xx.y.y.y:17787 ,  
'https://31668.gradio.app')
```

(14) interface = gr.Interface(fn = hello\_world,  
inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),  
outputs = "text")

(x) interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

The screenshot shows a Jupyter Notebook cell with the following code:

```
http://xx.y.y.y:17787 ,  
'https://31668.gradio.app')
```

(14) interface = gr.Interface(fn = hello\_world,  
inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),  
outputs = "text")

(x) interface.launch()

A message indicates that a Colab notebook is detected and provides instructions for debugging. It also mentions a public URL: <https://44140.gradio.app>. A note states that the share link will expire in 72 hours and provides an email address for longer links.

The interface is displayed below the code cell, showing a text input field labeled "NAME" with the placeholder "Enter your input here....". There are "Screenshot" and "Flag" buttons next to the interface. At the bottom, it shows a status message: "✓ 3s completed at 5:03 PM".

**Timestamp: 595.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

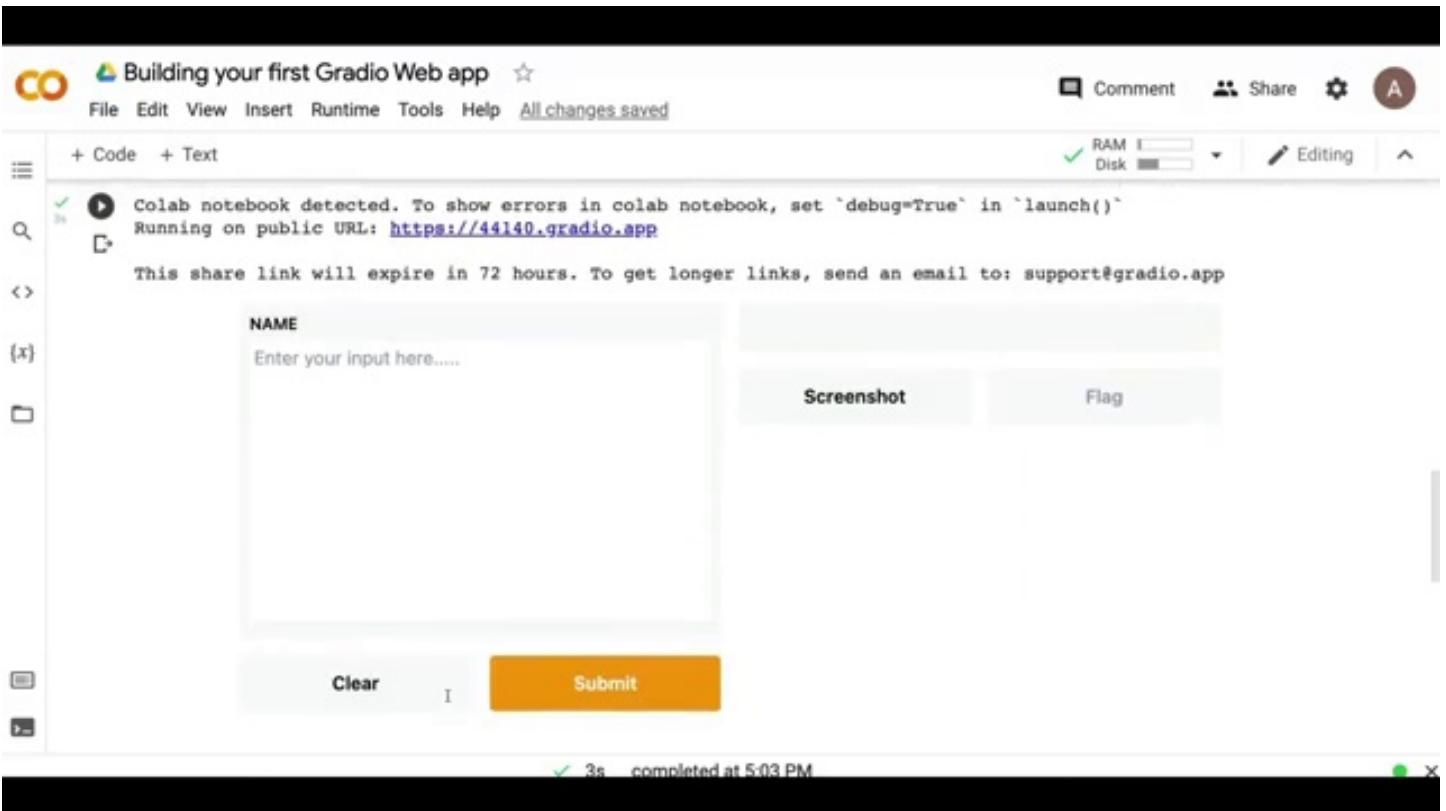
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
(x) Enter your input here.....

Screenshot Flag

Clear I Submit

✓ 3s completed at 5:03 PM



**Timestamp: 596.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

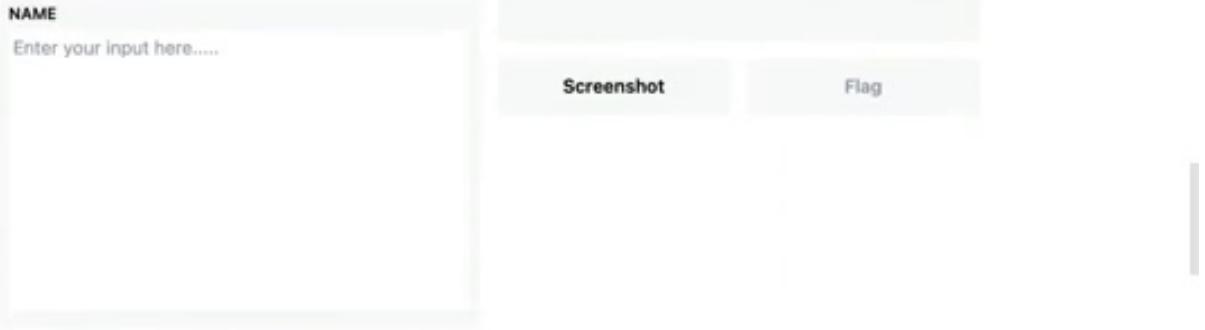
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
(x) Enter your input here.....

Screenshot Flag

Clear I Submit

✓ 3s completed at 5:03 PM



**Timestamp: 597.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

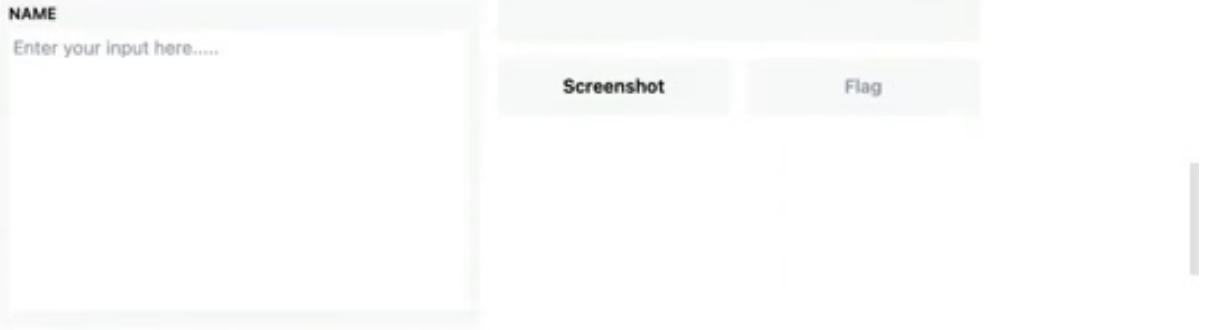
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
(x) Enter your input here.....

Screenshot Flag

Clear I Submit

✓ 3s completed at 5:03 PM



**Timestamp: 598.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 599.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

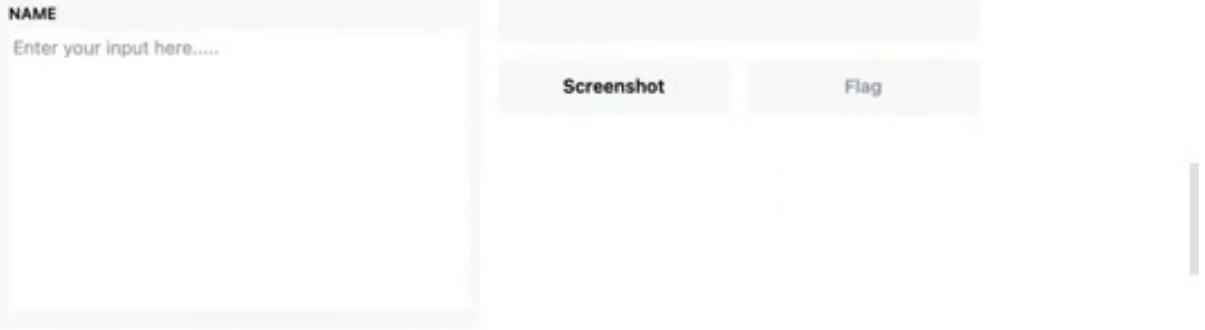
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
(x) Enter your input here.....

Screenshot Flag

Clear I Submit

✓ 3s completed at 5:03 PM



**Timestamp: 600.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

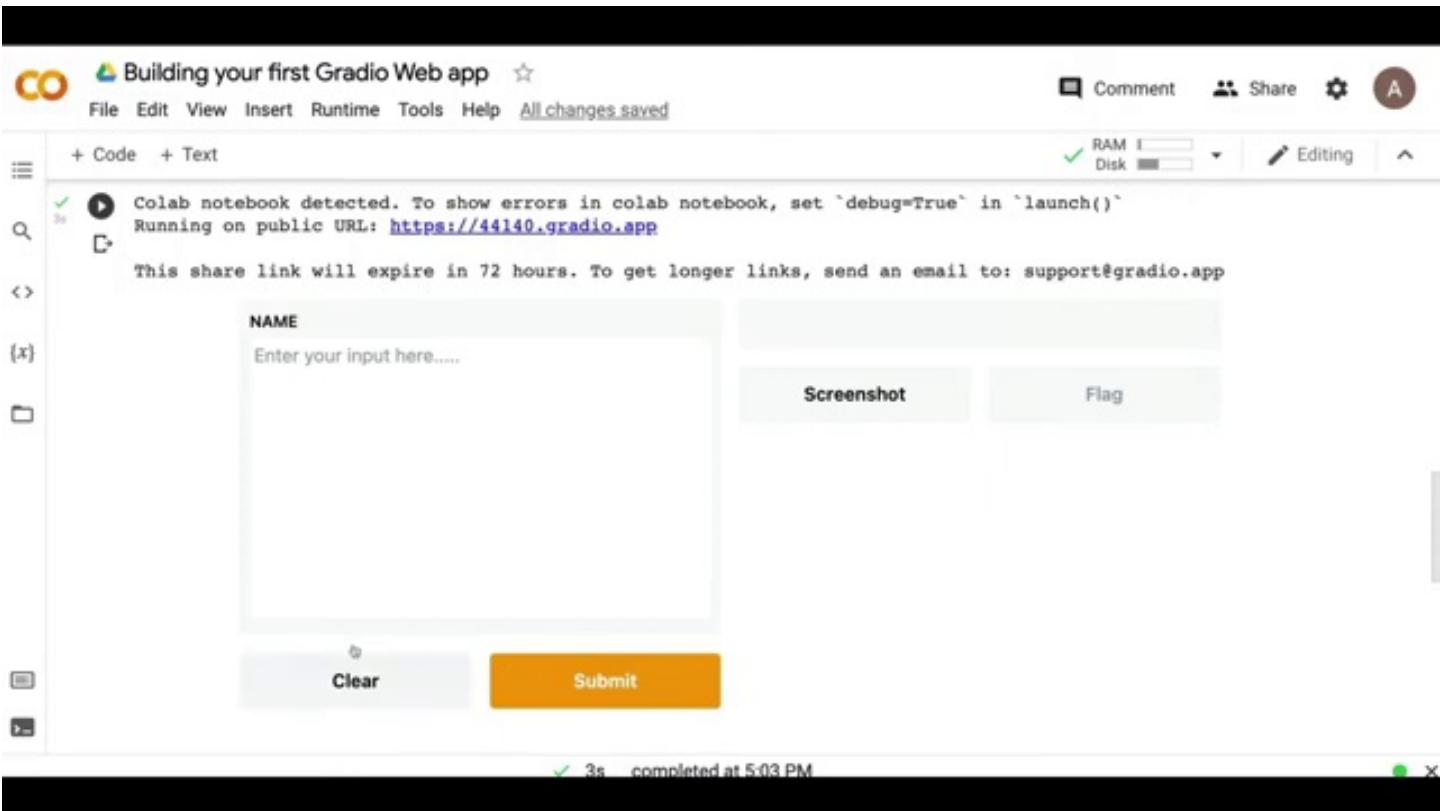
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
(x) Enter your input here.....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 601.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

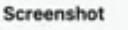
RAM Disk Editing

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

(x)  interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME	
Enter your input here....	 

✓ 3s completed at 5:03 PM



**Timestamp: 602.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

(x)

```
[15] interface.launch()
```

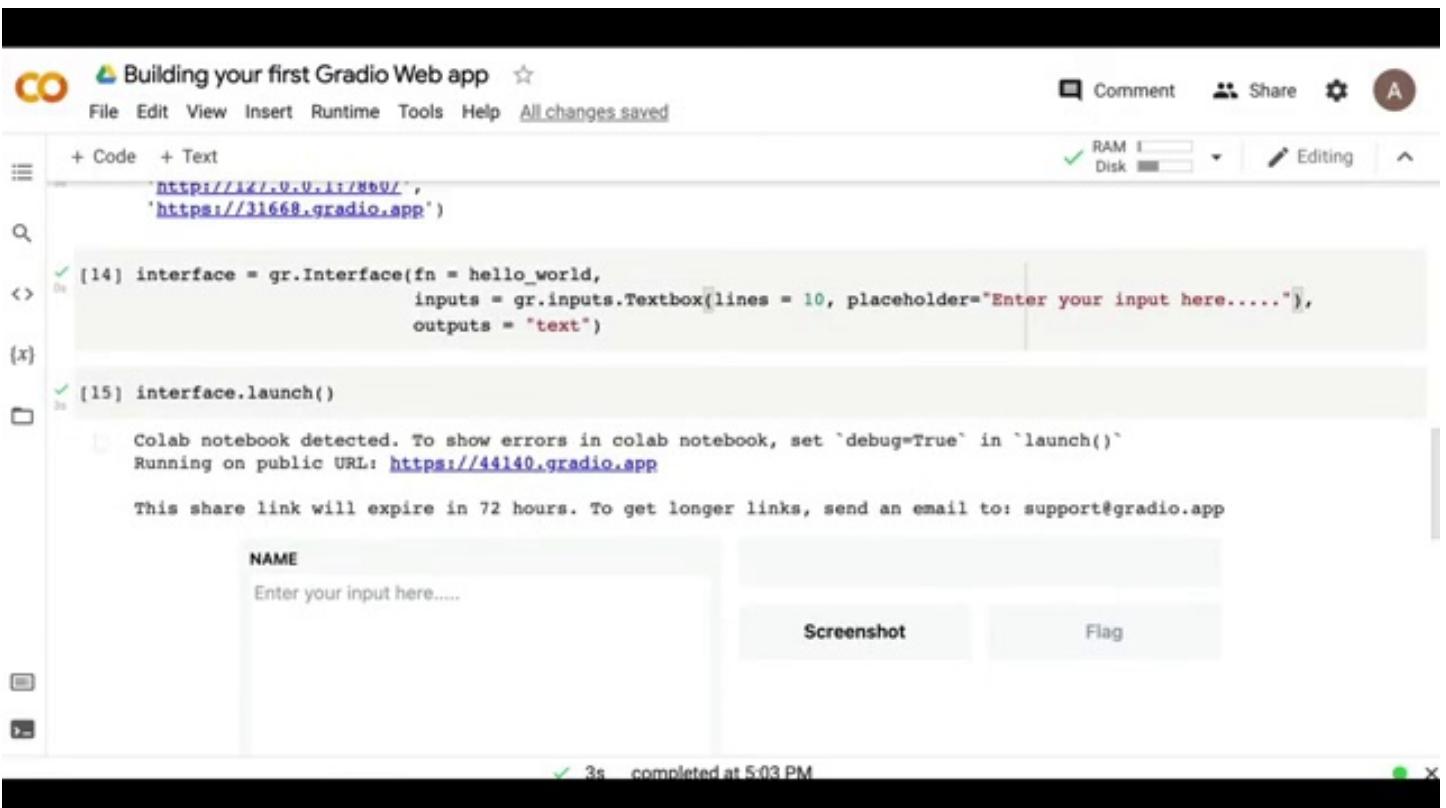
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 603.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 604.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

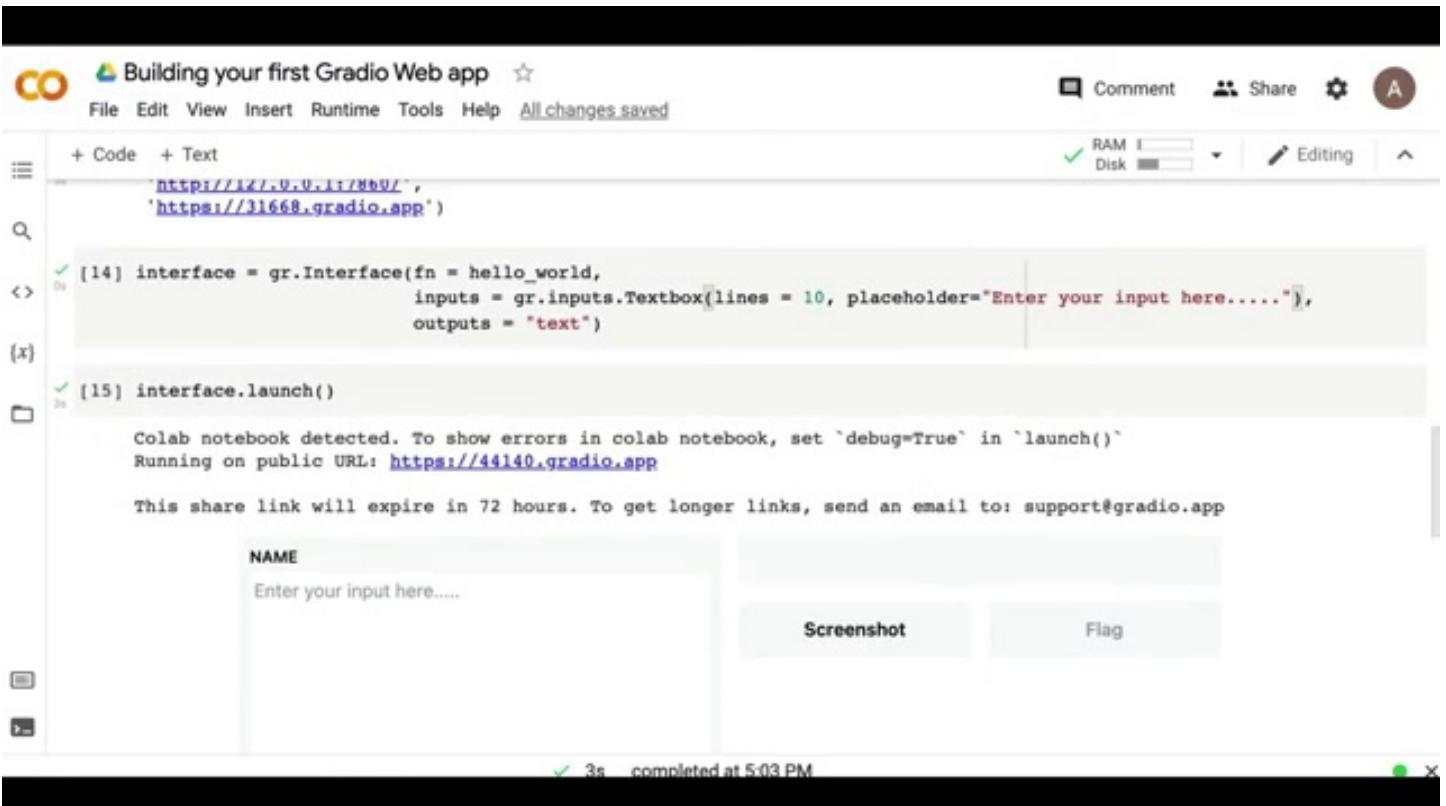
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 605.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

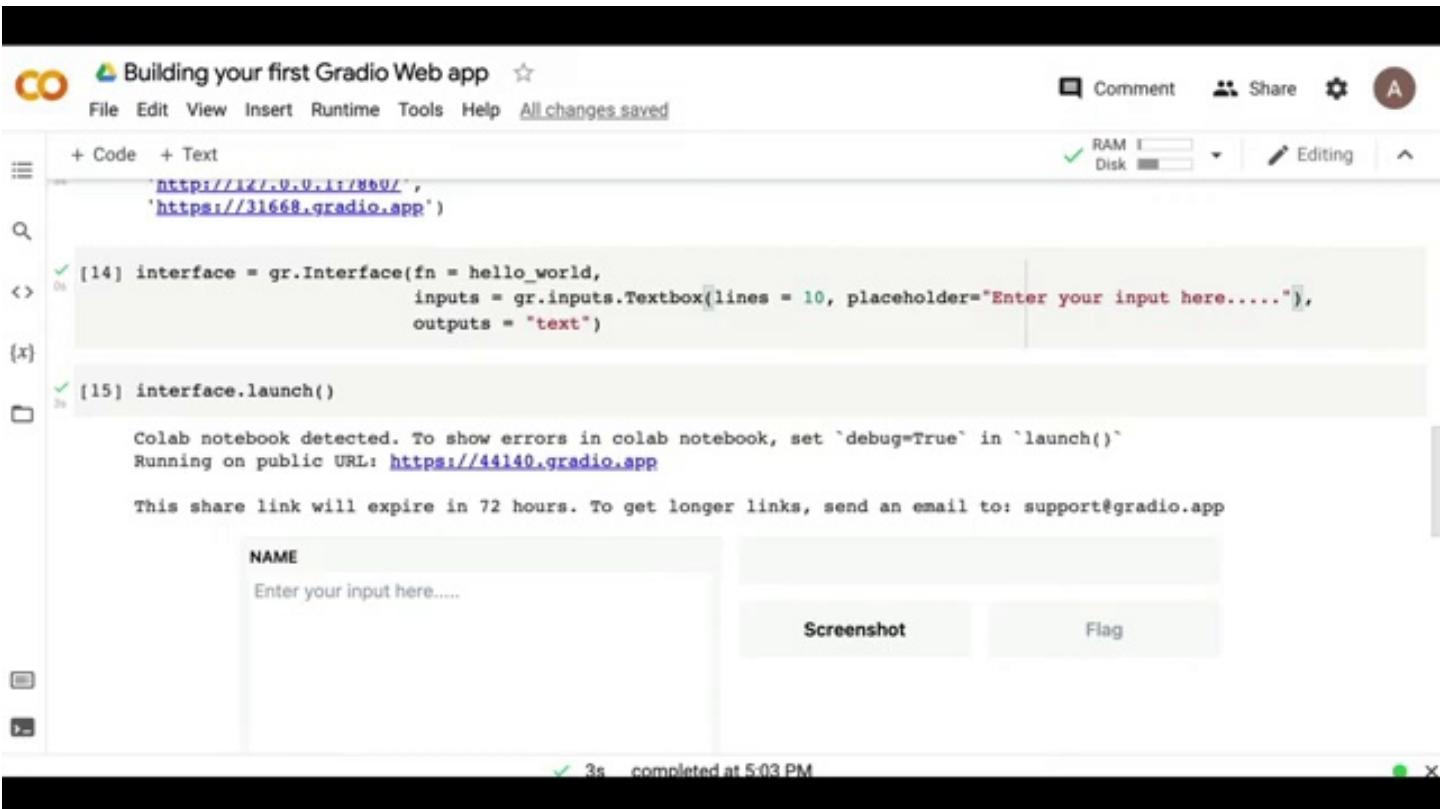
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 606.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

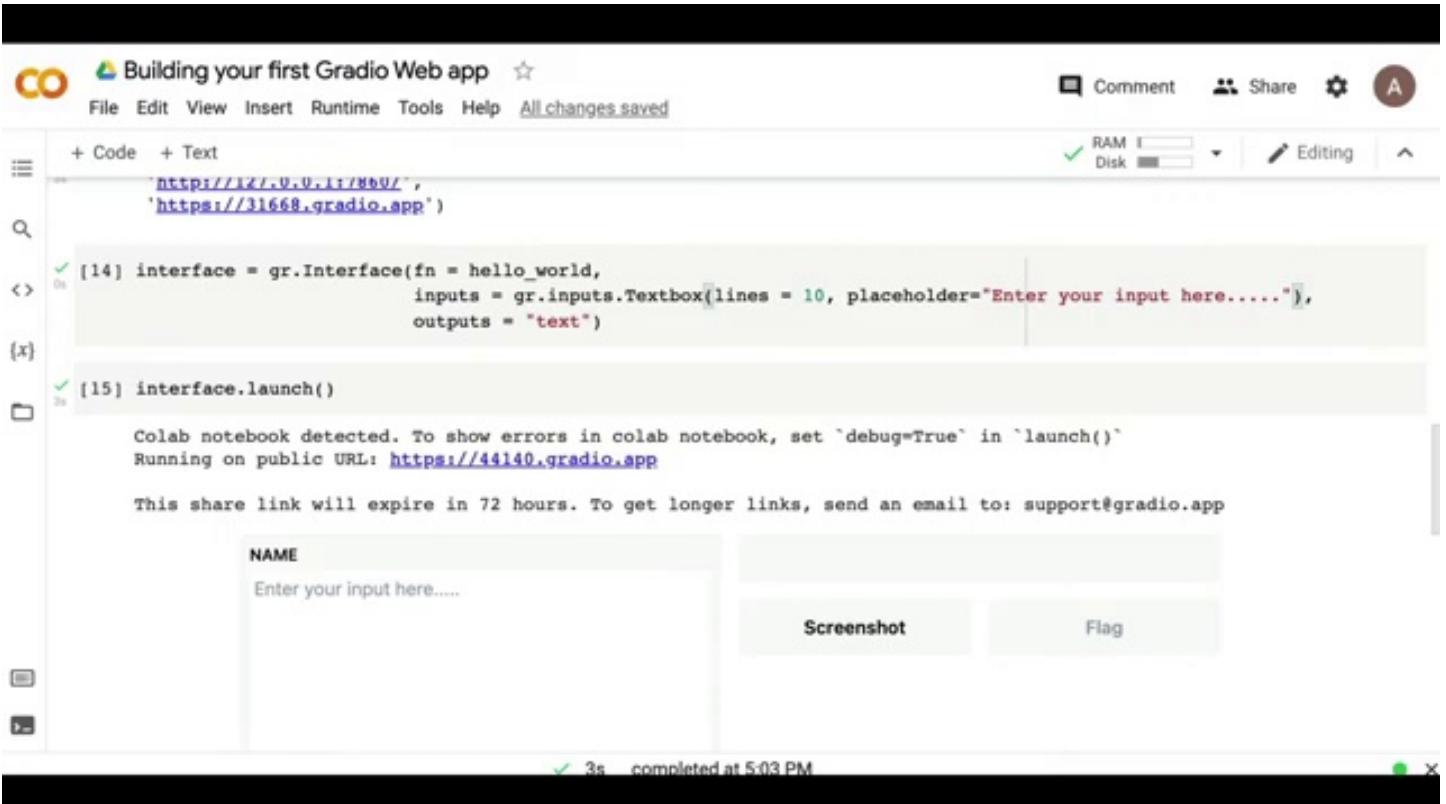
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 607.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

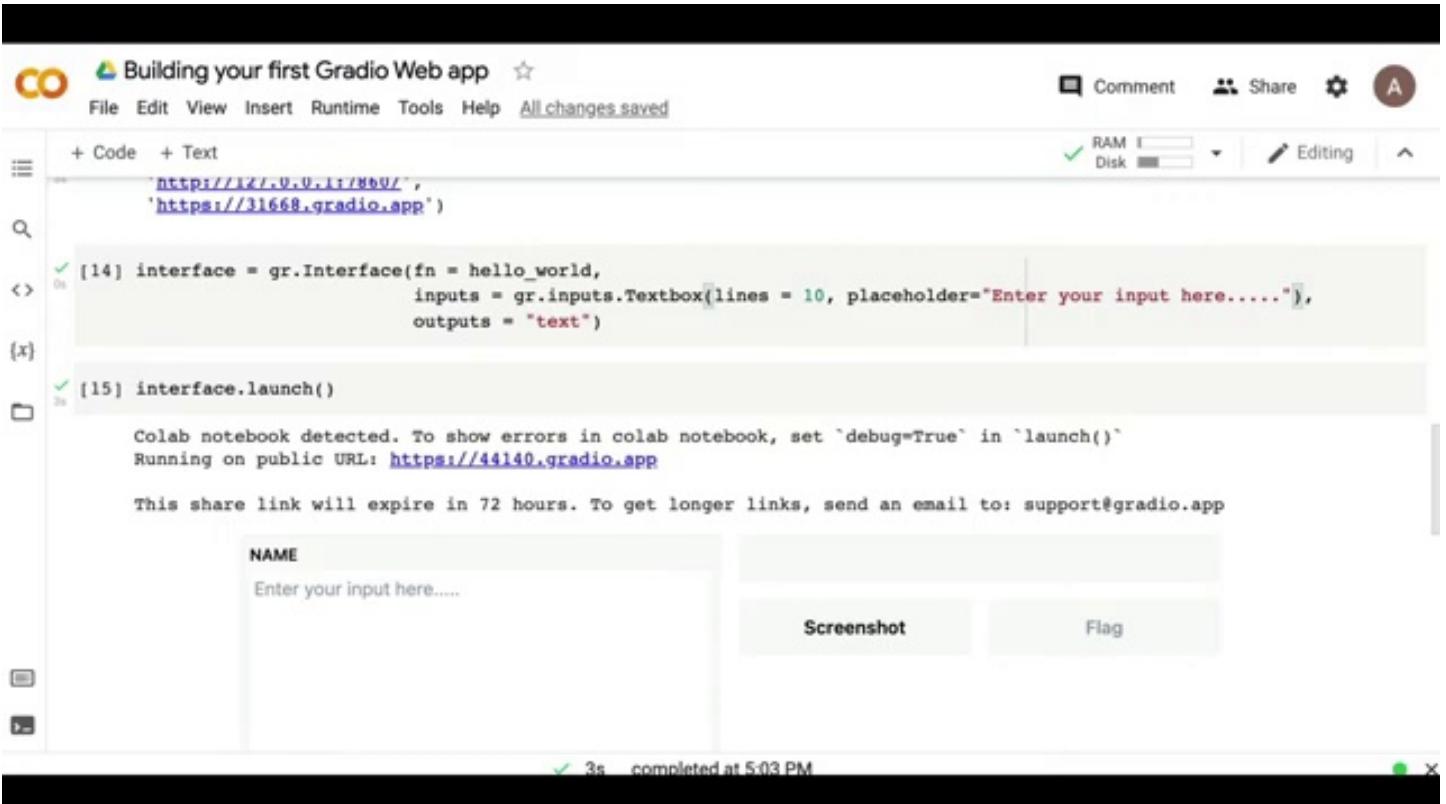
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 608.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

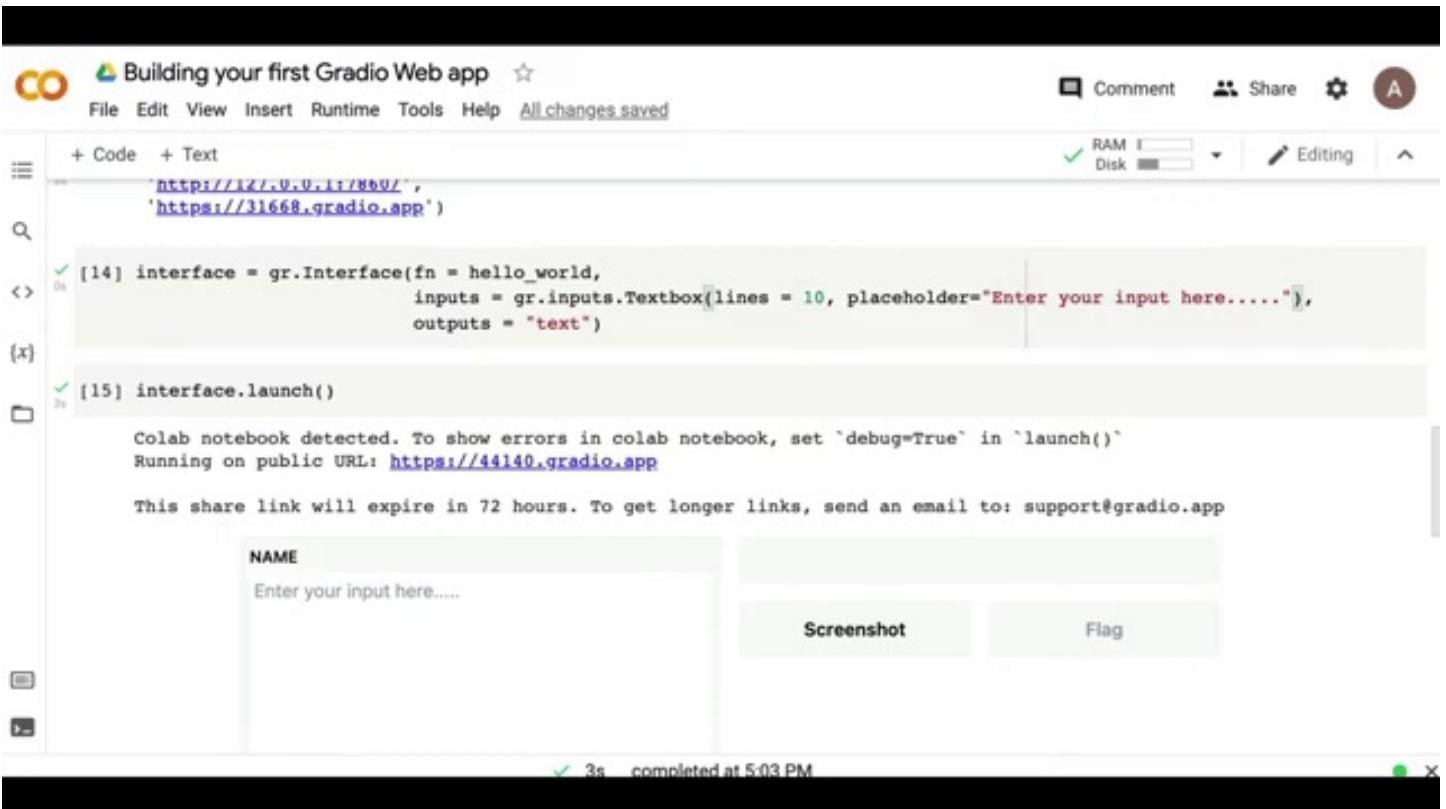
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 609.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 610.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

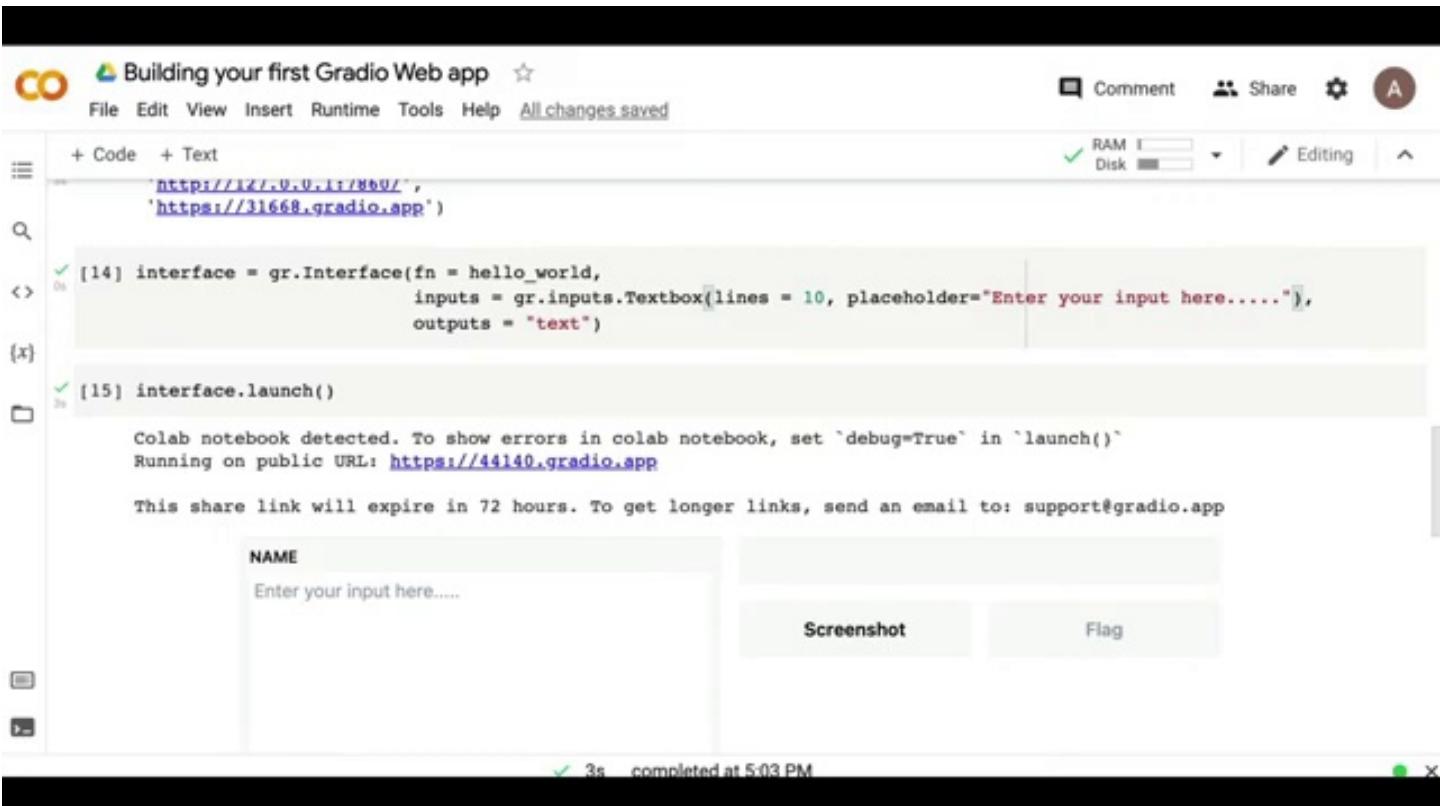
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 611.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

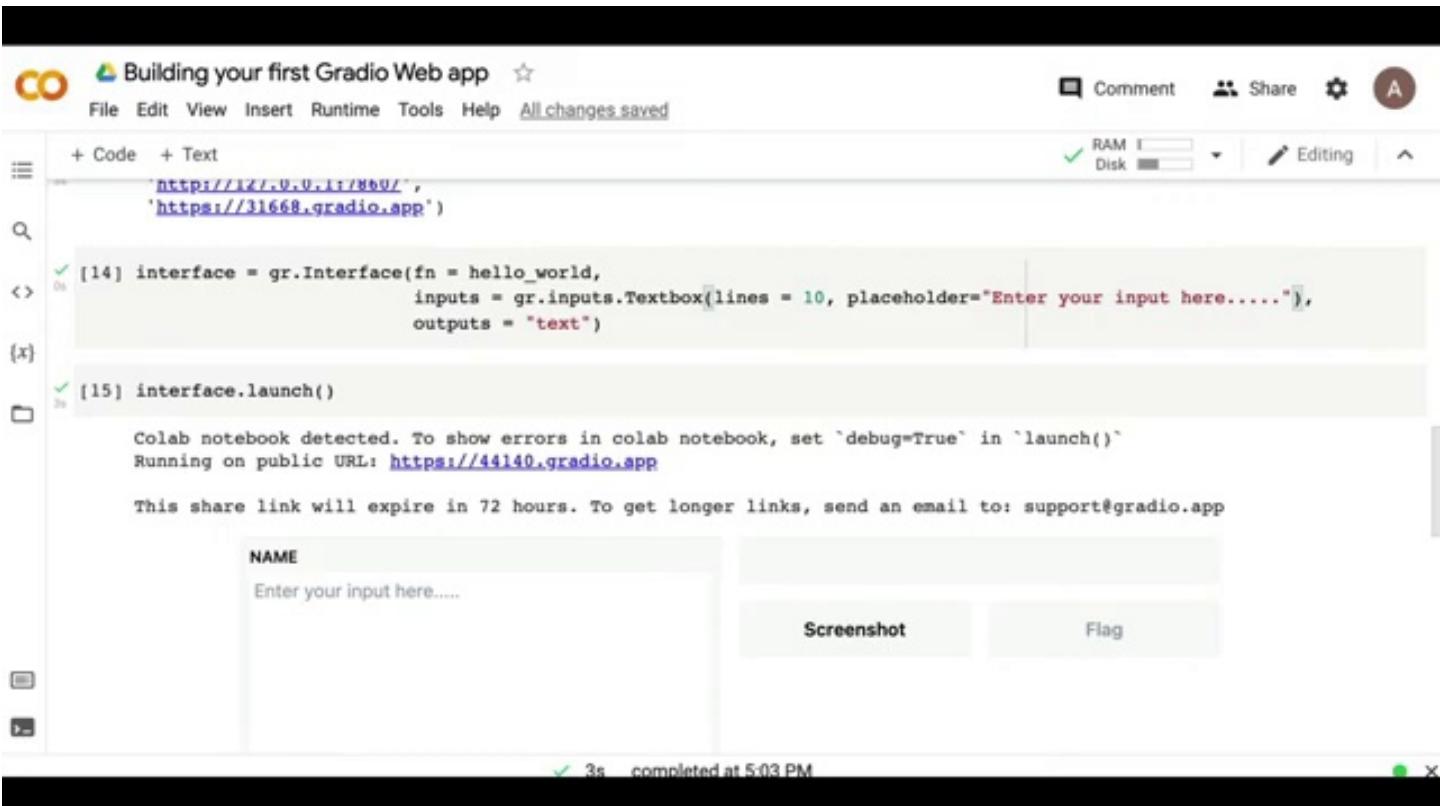
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 612.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

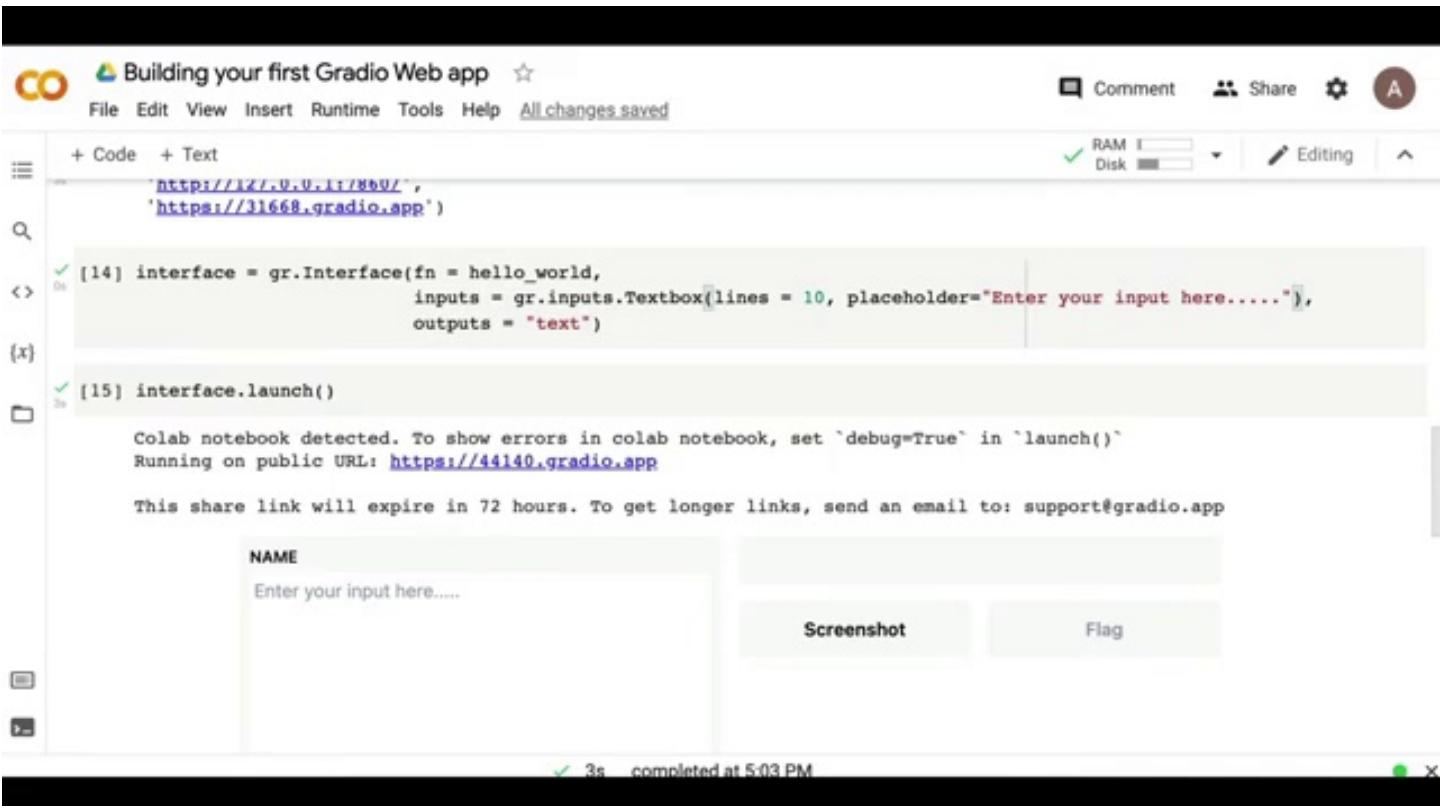
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 613.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

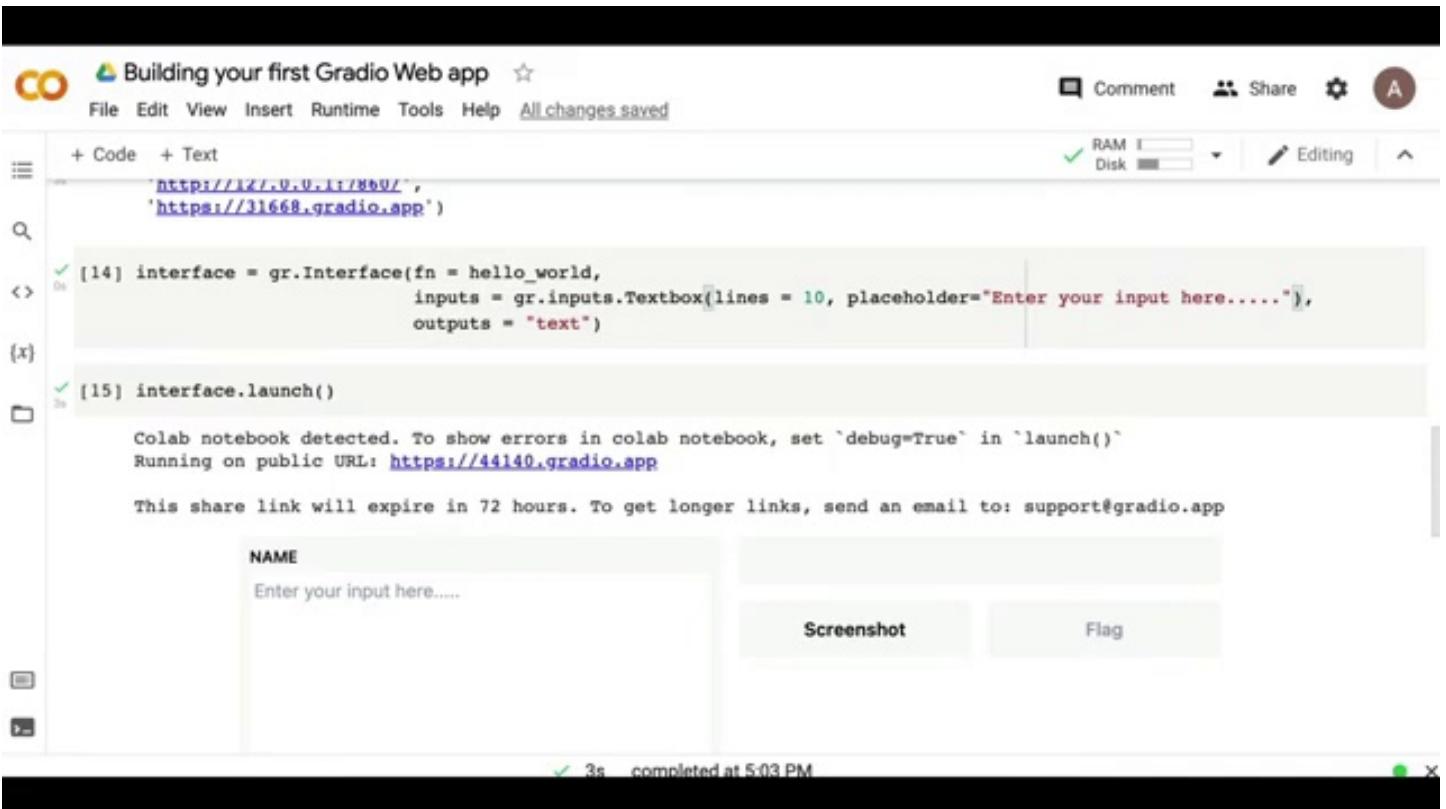
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 614.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

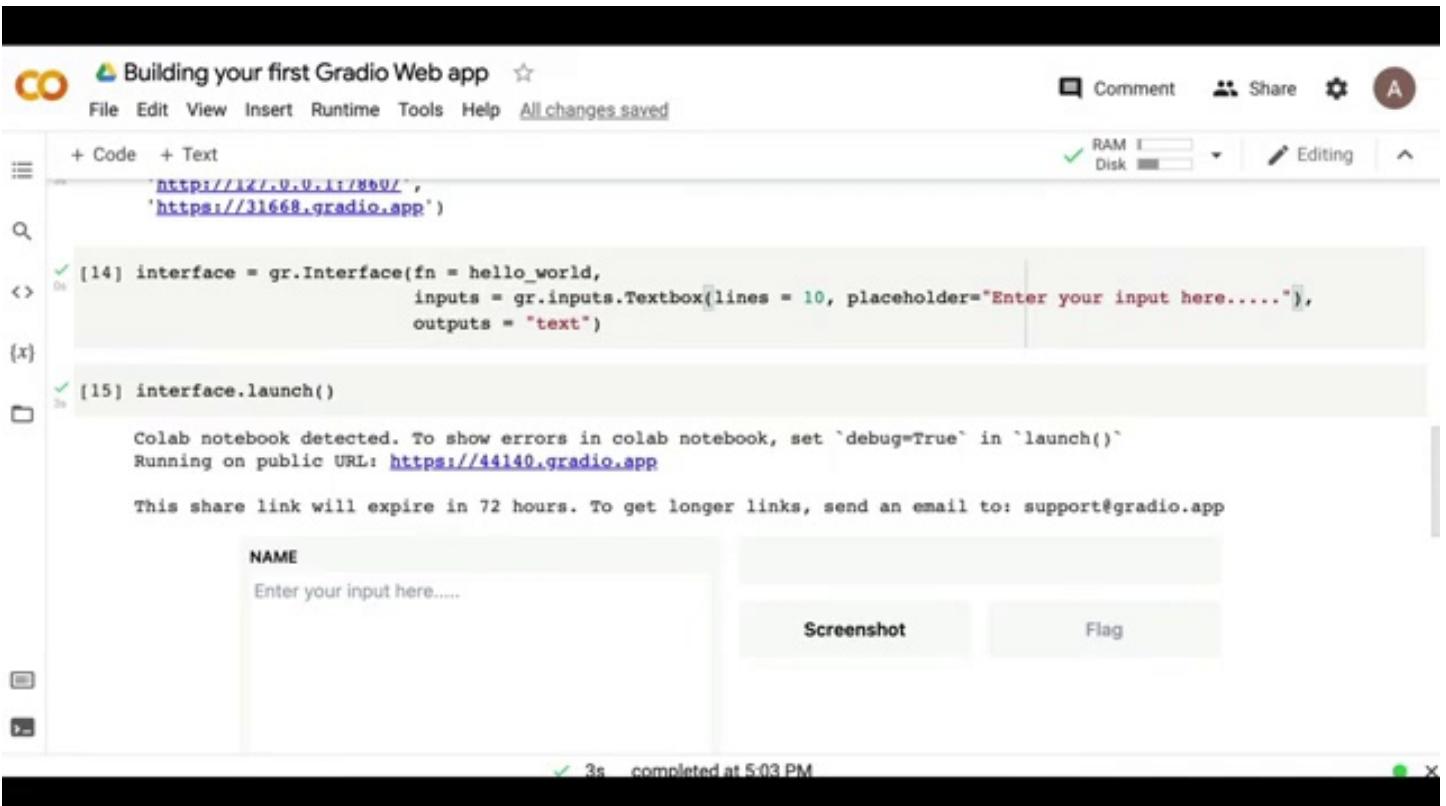
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 615.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

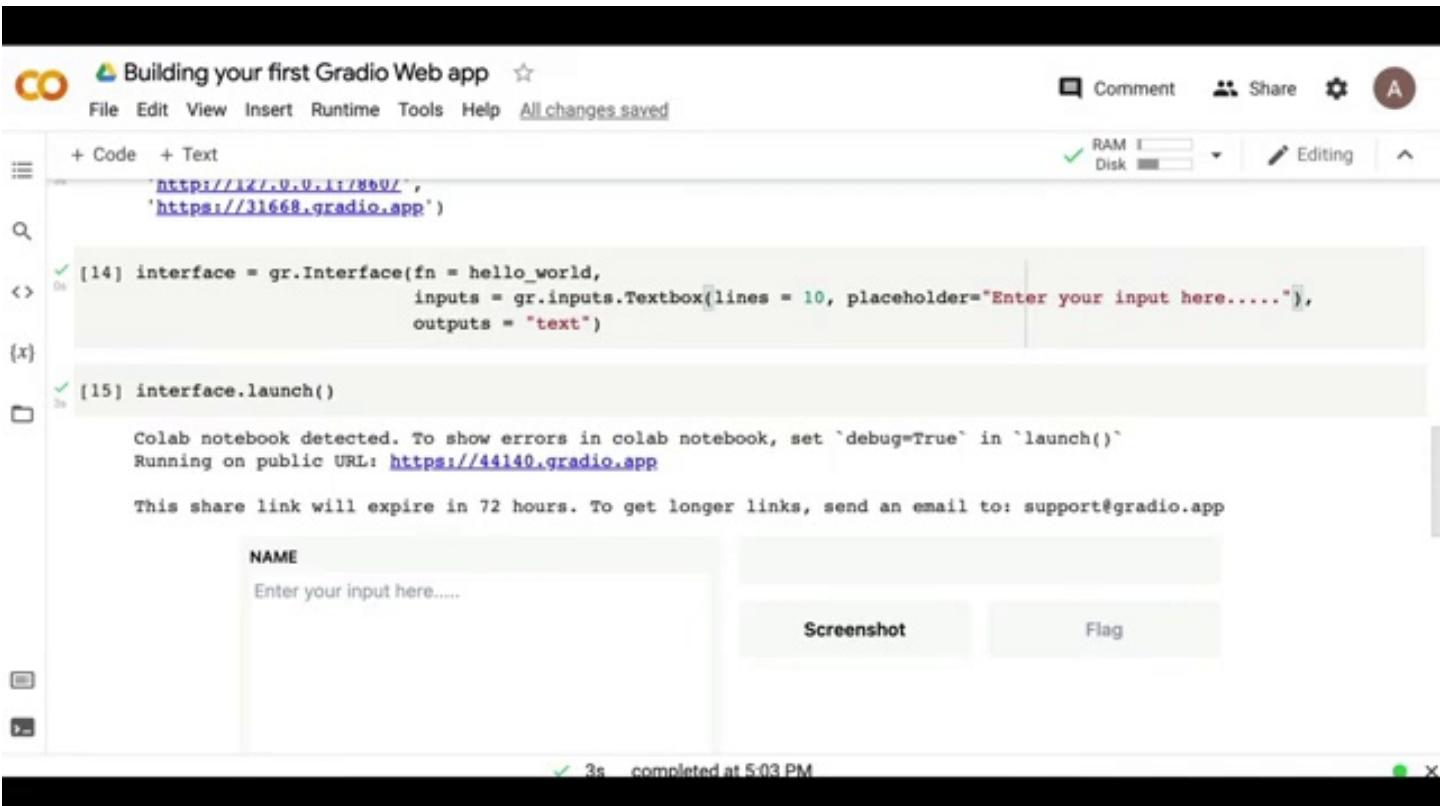
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 616.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

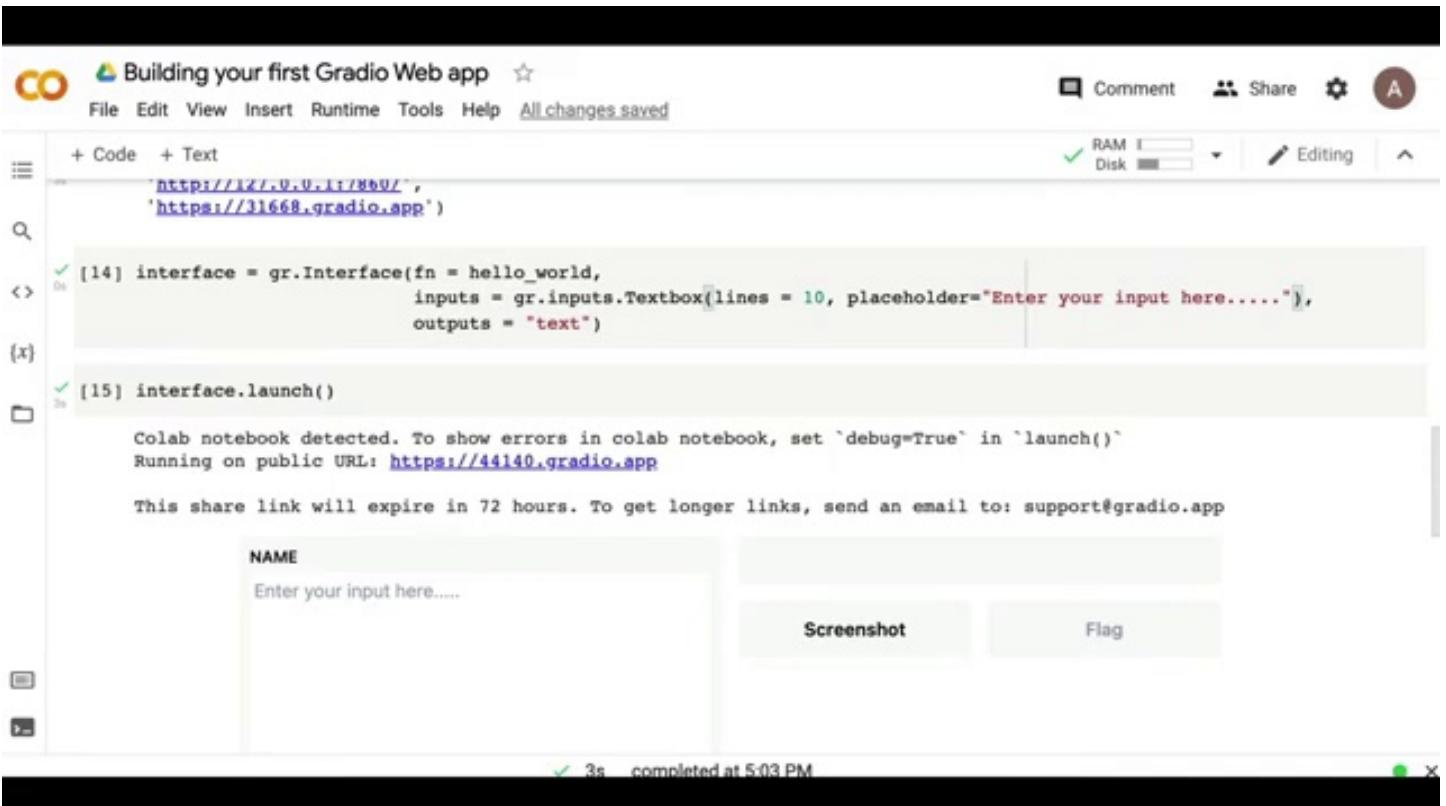
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 617.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

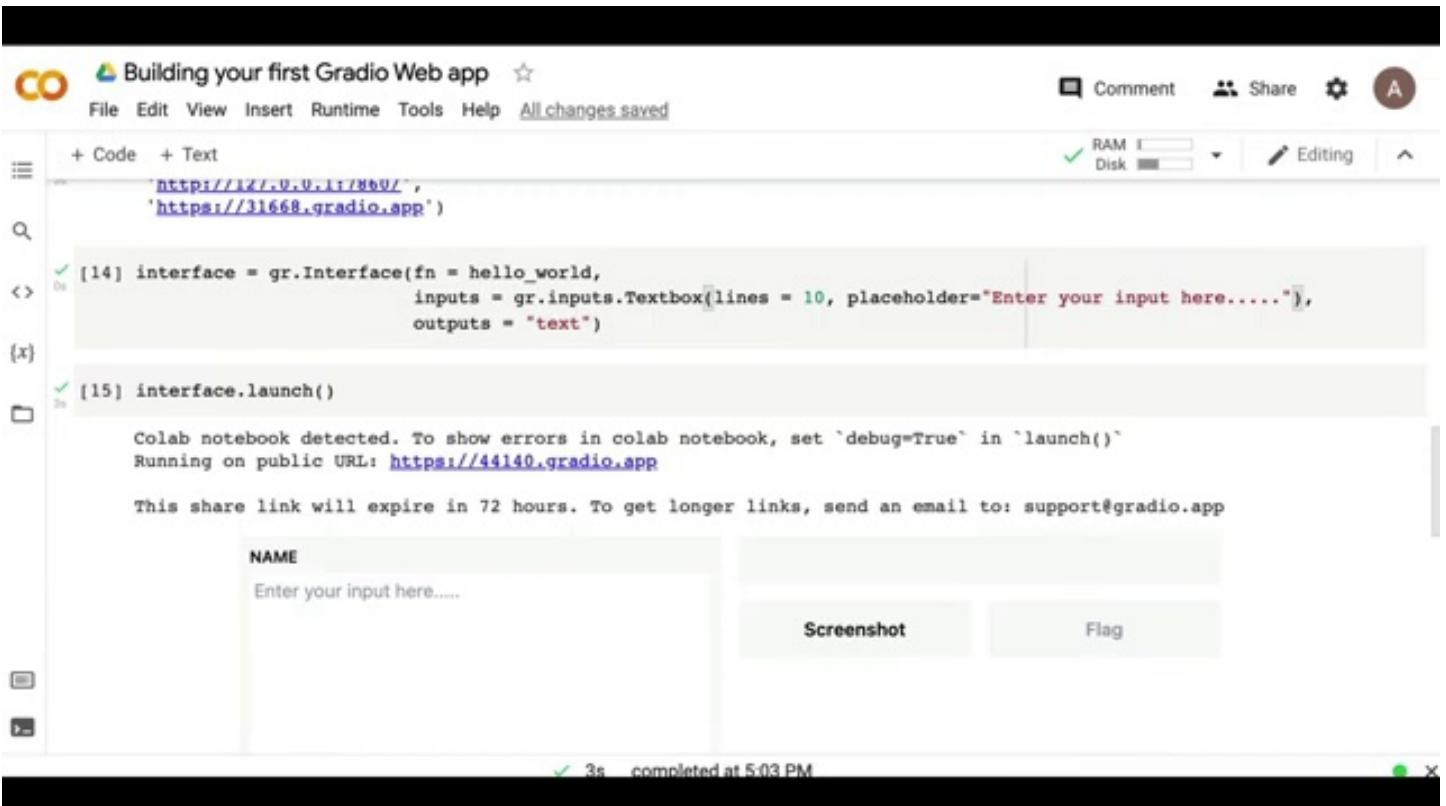
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 618.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

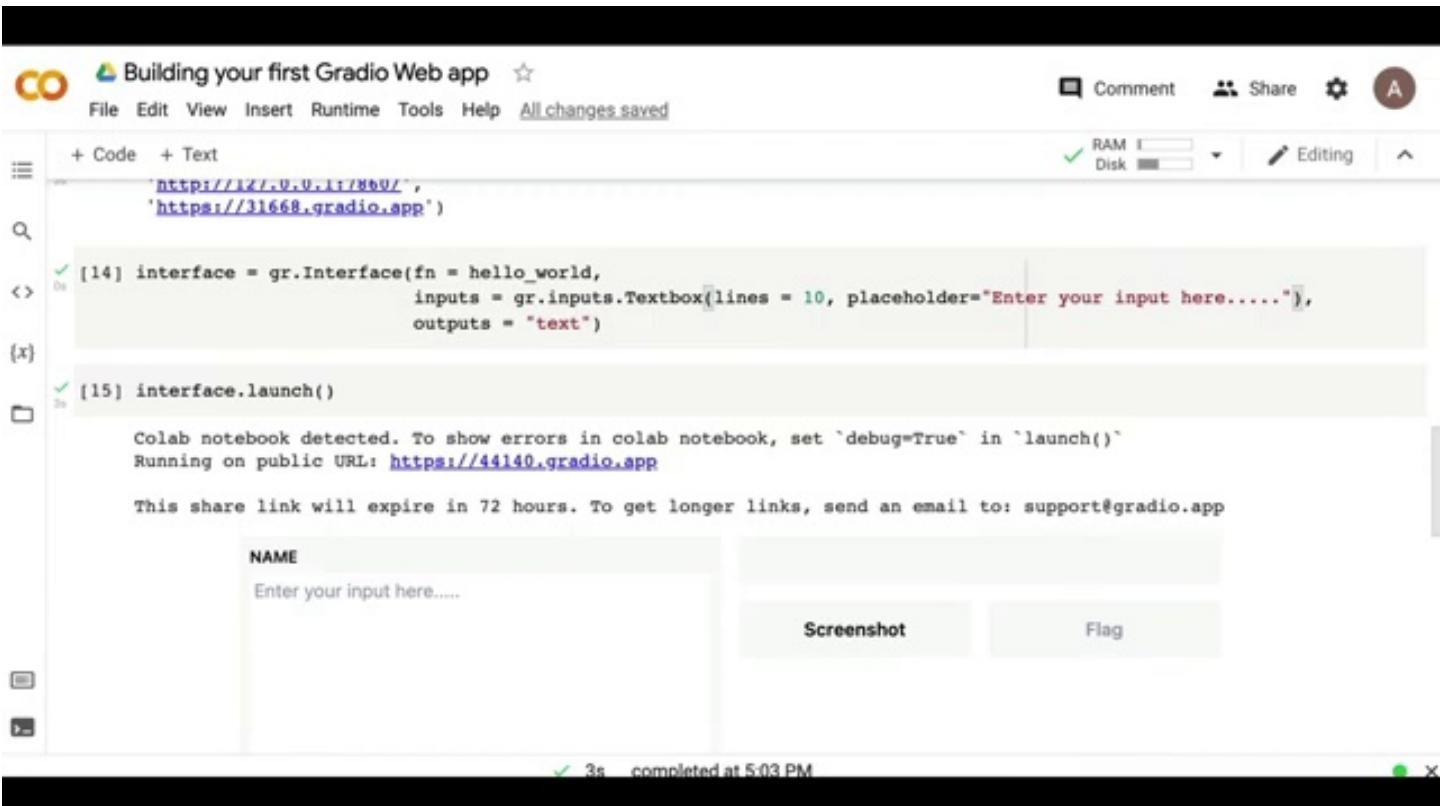
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 619.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

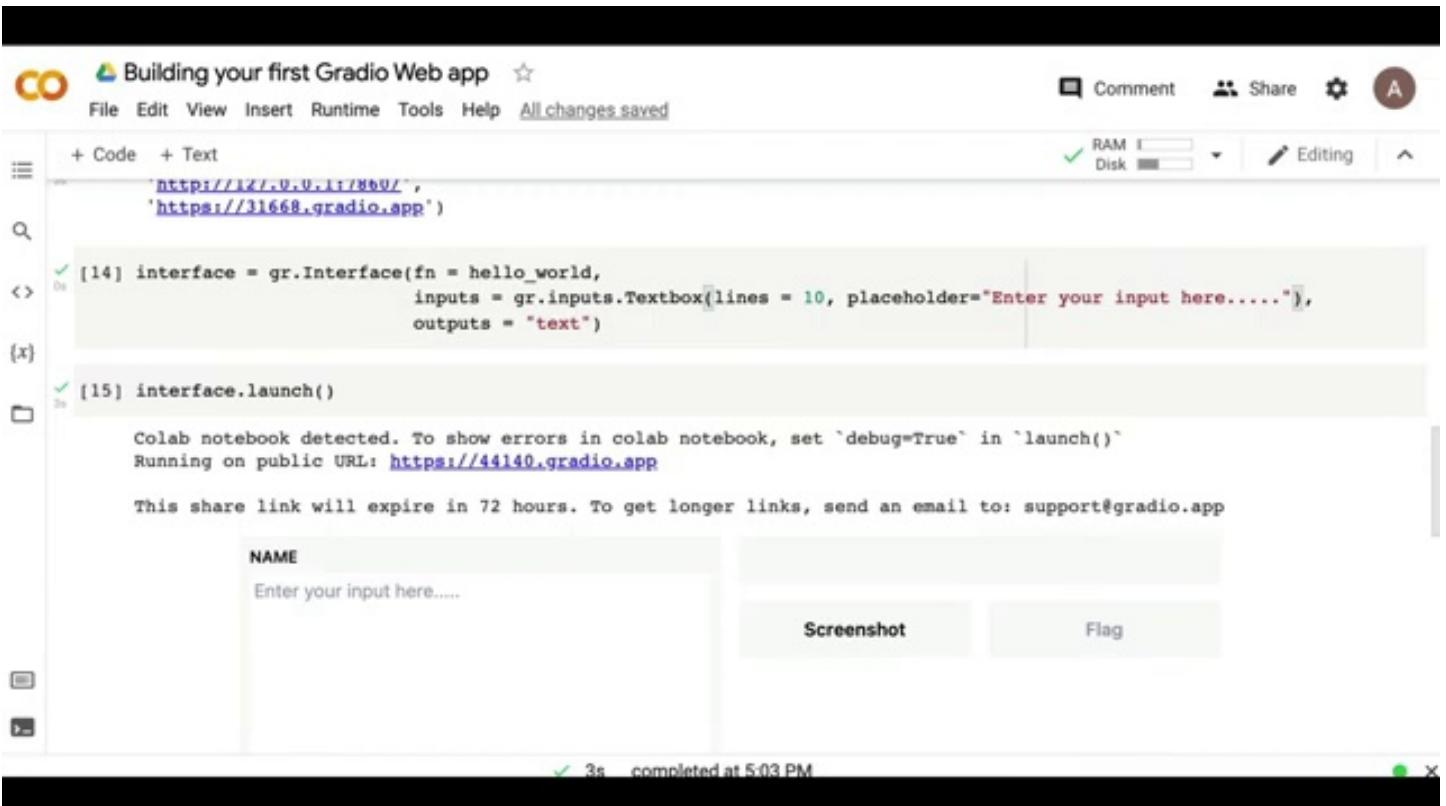
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 620.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 621.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

The screenshot shows a Colab notebook interface. At the top, there's a toolbar with icons for Comment, Share, Settings, and a profile picture. Below the toolbar, the menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message 'All changes saved'. The main area contains two tabs: '+ Code' and '+ Text'. The 'Code' tab is active, displaying Python code for creating a Gradio interface and launching it. The code uses the 'gr' library to define a function 'hello\_world' and create a 'Textbox' input with 10 lines and a placeholder 'Enter your input here.....'. It then creates a 'text' output. Finally, it calls 'interface.launch()' to run the app. A note indicates that the Colab notebook is detected, and an error will be shown if 'debug=True' is not set in 'launch()'. The public URL for the app is provided: <https://44140.gradio.app>. A message states that the share link will expire in 72 hours. Below the code, there's a text input field labeled 'NAME' with the placeholder 'Enter your input here....'. At the bottom right, there are 'Screenshot' and 'Flag' buttons. A progress bar at the bottom indicates the task was completed in 3 seconds at 5:03 PM.

**Timestamp: 622.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

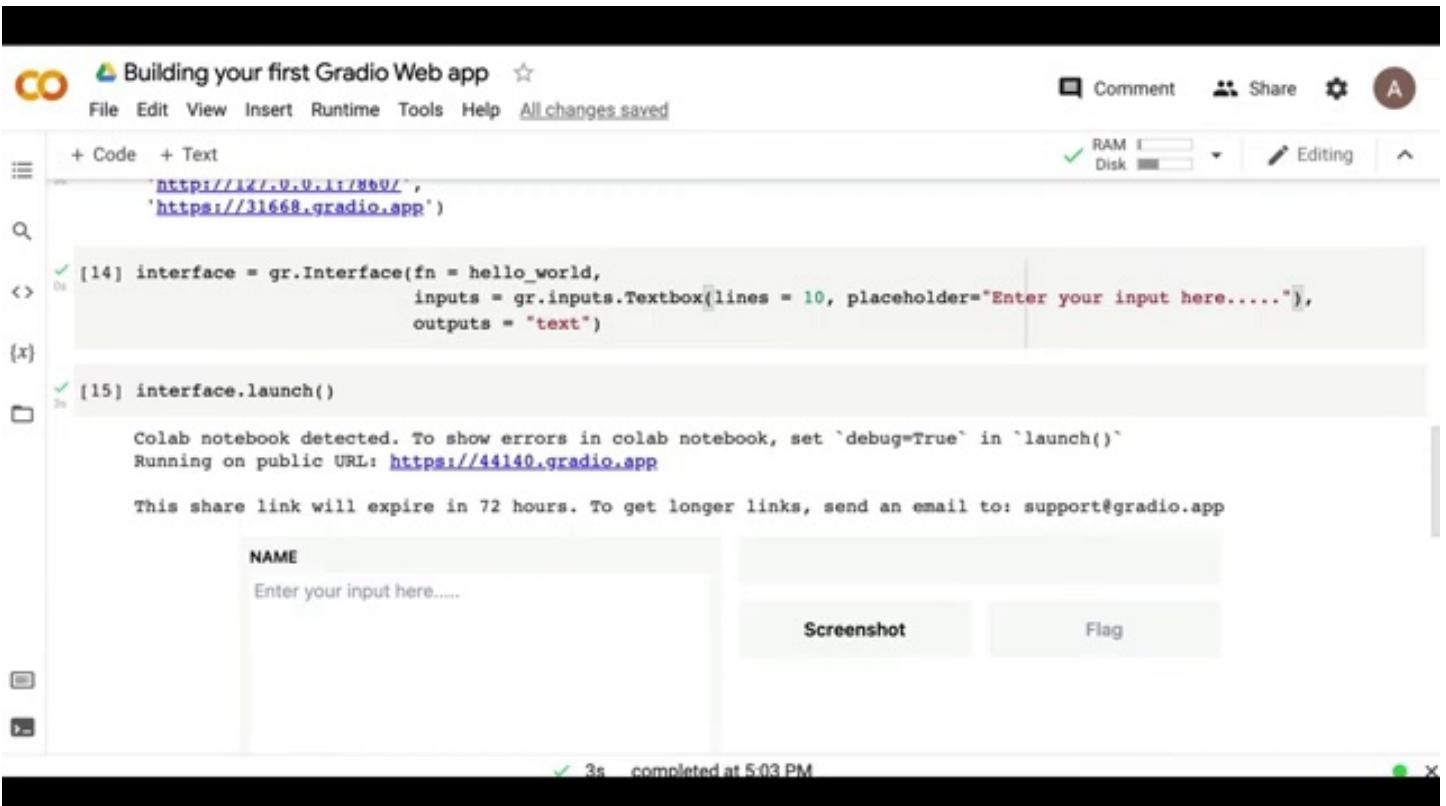
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 623.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

{x}

```
[15] interface.launch()
```

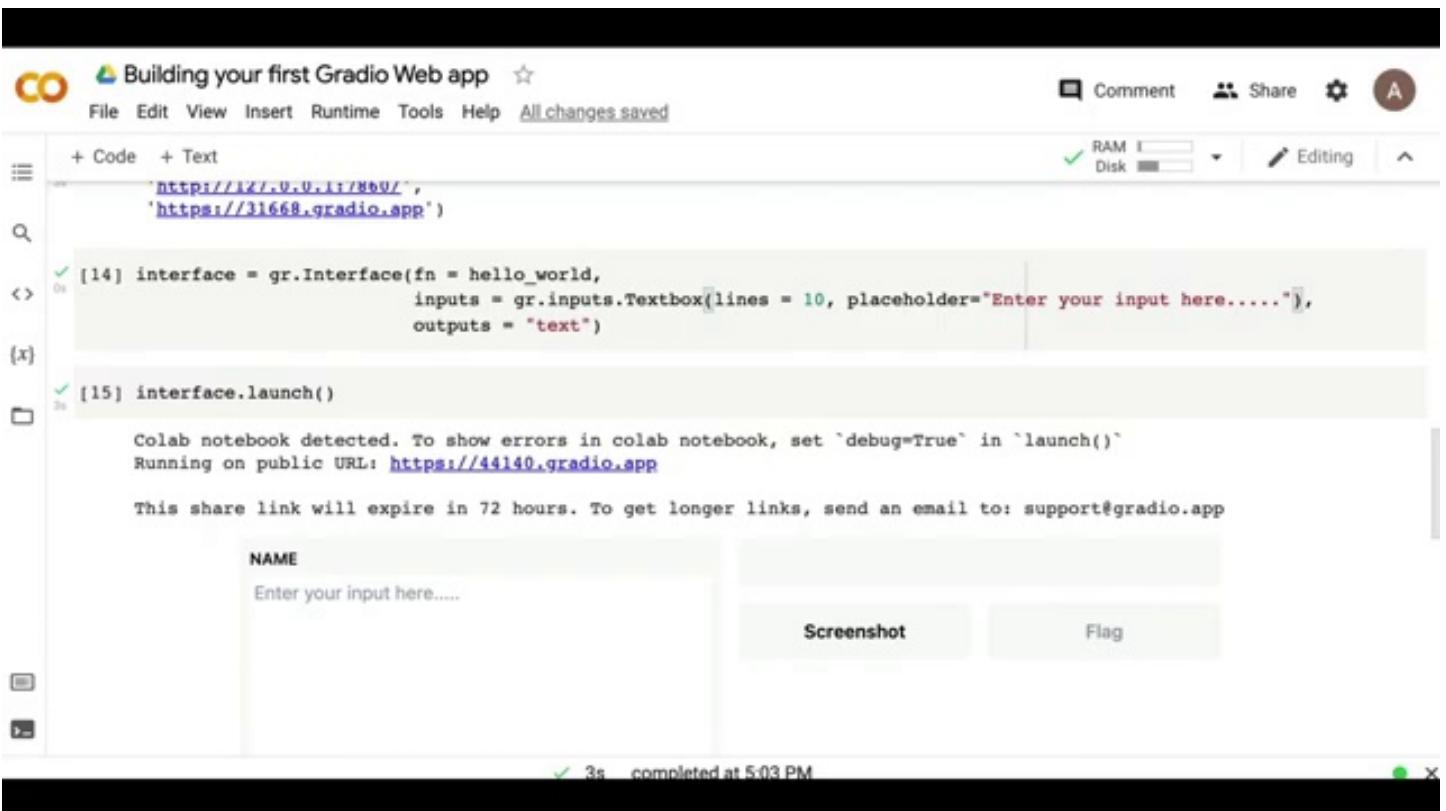
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 624.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[14] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")
```

[15] interface.launch()

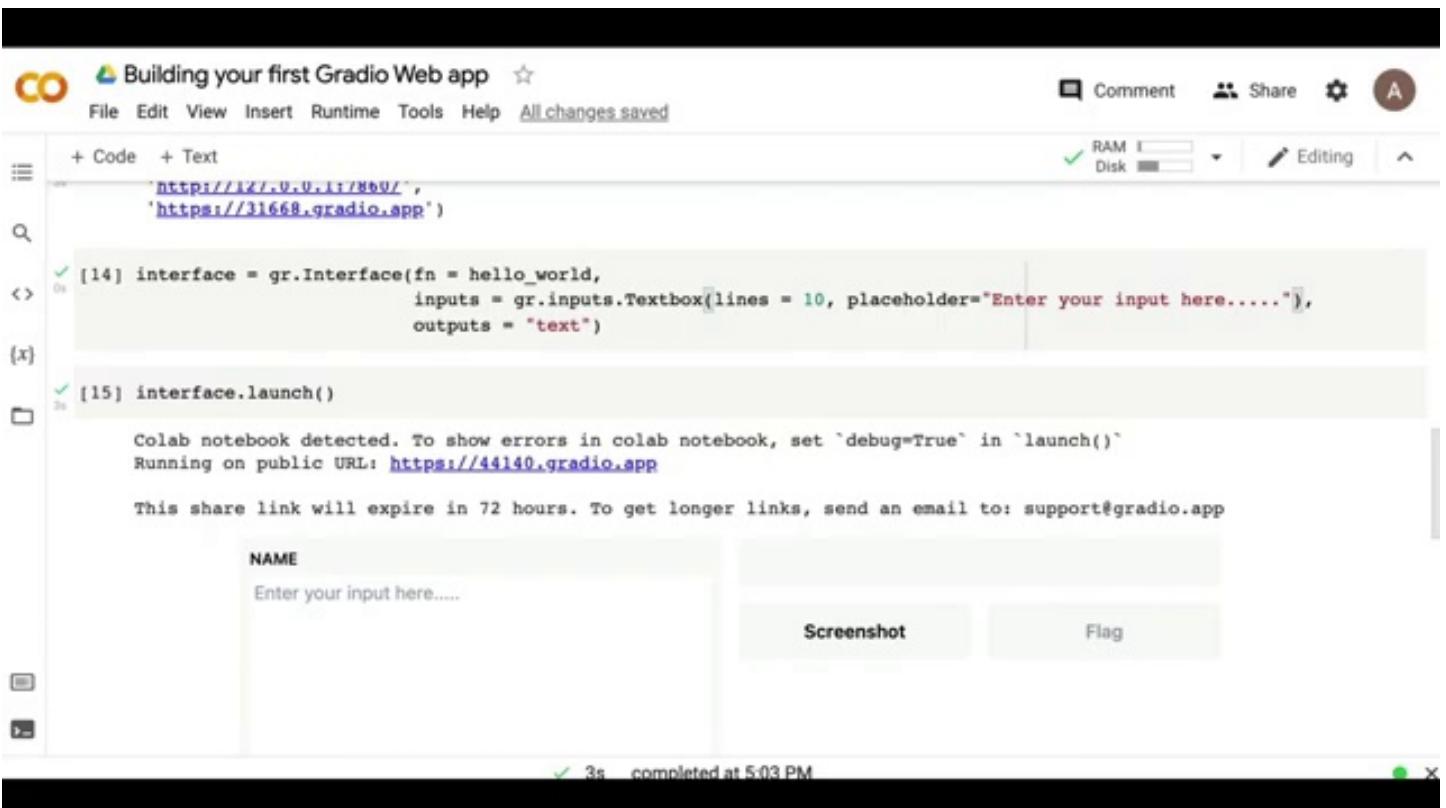
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 625.00 seconds**

## Customizing Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

NAME  
Name Here...

Screenshot

Flag

Clear

Submit

## Multiple Inputs and Outputs

Let's say we had a much more complex function, with multiple inputs and outputs. In the example below, we have a function that takes a string, boolean, and number, and returns a string and number. Take a look how we pass a list of input and output components.

```
import gradio as gr
```

**Timestamp: 626.00 seconds**

TEMPERATURE

0

Clear

Submit

We simply wrap the components in a list. Furthermore, if we wanted to compare multiple functions that have the same input and return types, we can even pass a list of functions for quick comparison.

I

### Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

**Timestamp: 627.00 seconds**

We simply wrap the components in a test. Furthermore, if we wanted to compare multiple functions that have the same input and return types, we can even pass a list of functions for quick comparison.

## Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepio_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepio_img = input_img.dot(sepio_filter.T)
    sepio_img /= sepio_img.max()
    return sepio_img

iface = gr.Interface(sepio, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```



**Timestamp: 628.00 seconds**

We simply wrap the components in a test. Furthermore, if we wanted to compare multiple functions that have the same input and return types, we can even pass a list of functions for quick comparison.

## Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sepio(input_img):
    sepio_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepio_img = input_img.dot(sepio_filter.T)
    sepio_img /= sepio_img.max()
    return sepio_img

iface = gr.Interface(sepio, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```



**Timestamp: 629.00 seconds**

We simply wrap the components in a list. Furthermore, if we wanted to compare multiple functions that have the same input and return types, we can even pass a list of functions for quick comparison.

### Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sephia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

The screenshot shows a Gradio interface window. On the left, there is a large dashed rectangular area labeled "INPUT IMAGE" with the instruction "Drop Image Here" and "Click to Upload" below it. To the right of this area is a "Screenshot" button and a "Flag" button. The overall background is white, and the interface has a clean, modern look.

**Timestamp: 630.00 seconds**

We simply wrap the components in a list. Furthermore, if we wanted to compare multiple functions that have the same input and return types, we can even pass a list of functions for quick comparison.

### Working with Images

Let's try an image to image function. When using the `Image` component, your function will receive a numpy array of your specified size, with the shape `(width, height, 3)`, where the last dimension represents the RGB values. We'll return an image as well in the form of a numpy array.

```
import gradio as gr
import numpy as np

def sephia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "(image")  
iface.launch()
```

The screenshot shows a Gradio interface window. On the left, there is a large dashed rectangular area labeled "INPUT IMAGE" with the instruction "Drop Image Here" and "Click to Upload" below it. To the right of this area is a processing button labeled "Process". At the bottom right of the window, there are two small buttons: "Screenshot" and "Flag".

**Timestamp: 631.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

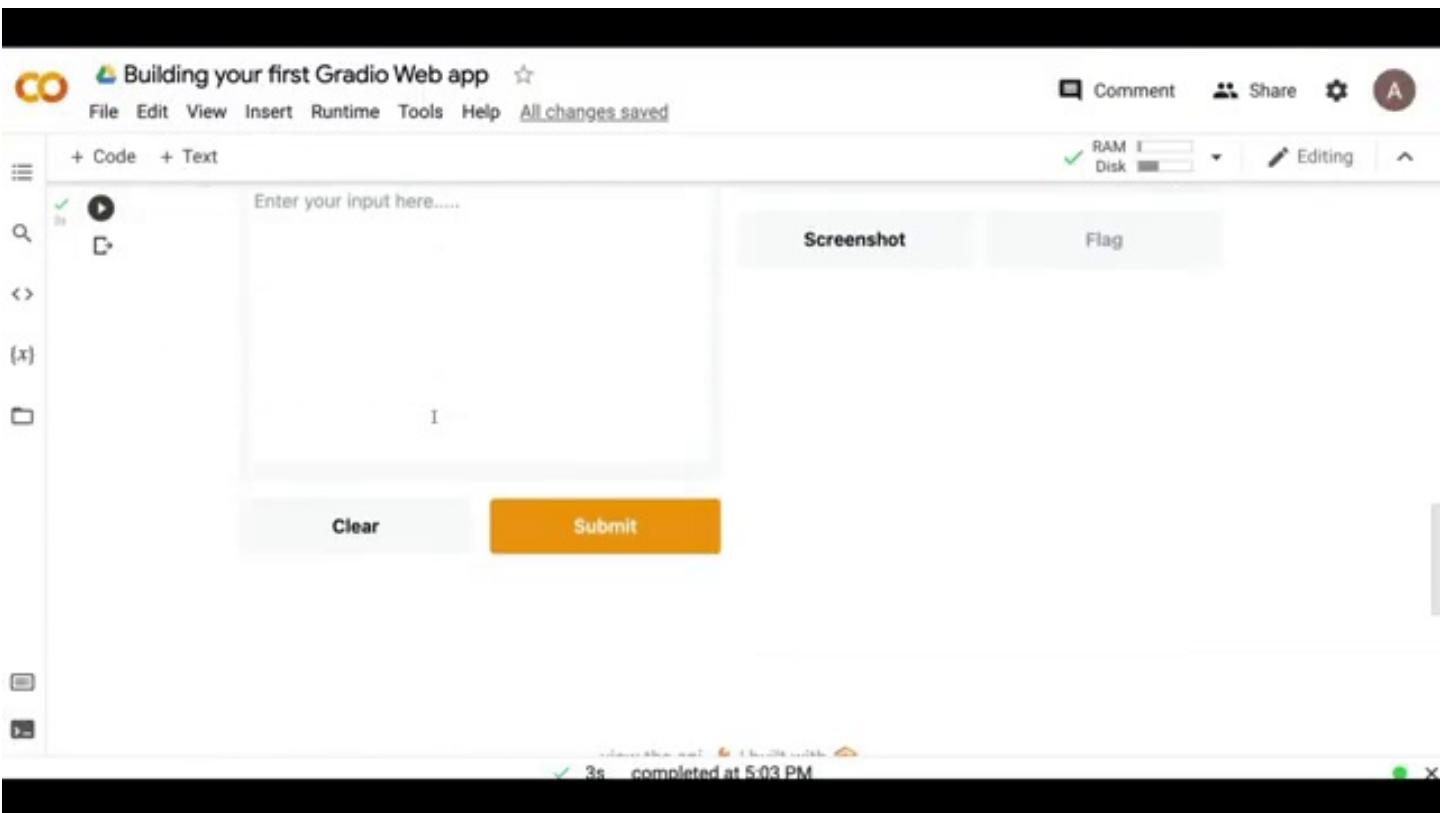
What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

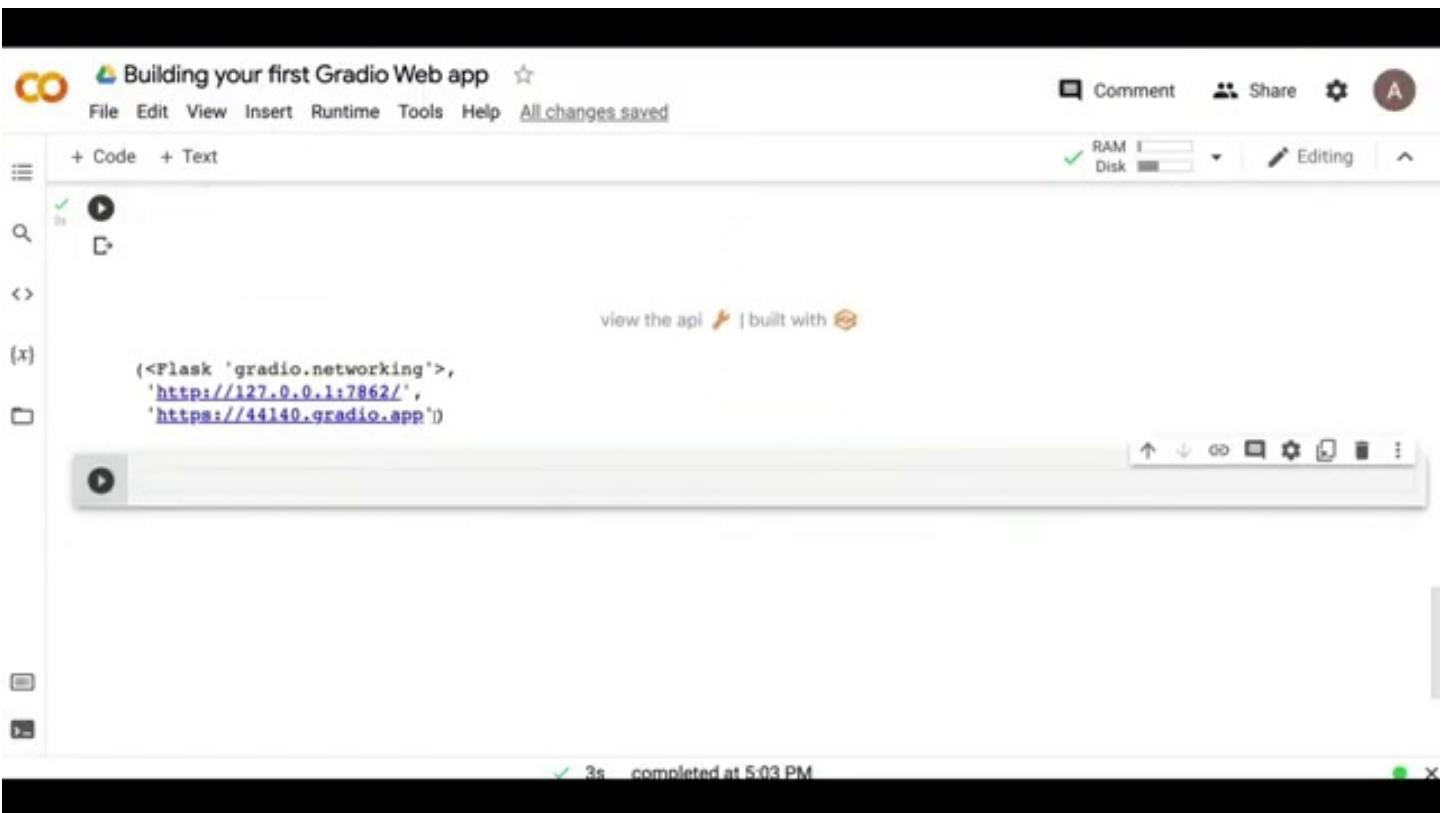
def greet(name):
    return "Hello " + name + "!"

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 632.00 seconds**



**Timestamp: 633.00 seconds**



**Timestamp: 634.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15]

view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')
```

3s completed at 5:03 PM



**Timestamp: 635.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'
```

{x} import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM

**Timestamp: 636.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 637.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 638.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'}
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

✓ 3s completed at 5:03 PM

**Timestamp: 639.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'}
```

{x} import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 640.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

✓ 3s completed at 5:03 PM

**Timestamp: 641.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 642.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#)

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 643.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 644.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 645.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 646.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 647.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 648.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'}
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sepia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sepia\_img = input\_img.dot(sepia\_filter.T)  
 sepia\_img /= sepia\_img.max()  
 return sepia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

✓ 3s completed at 5:03 PM

**Timestamp: 649.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] view the api | built with

```
(<Flask 'gradio.networking'>,
 'http://127.0.0.1:7862/',
 'https://44140.gradio.app')

(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()

✓ 3s completed at 5:03 PM
```

**Timestamp: 650.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'
```

(x) import gradio as gr  
import numpy as np

def sephia(input\_img):  
 sephia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sephia\_img = input\_img.dot(sephia\_filter.T)  
 sephia\_img /= sephia\_img.max()  
 return sephia\_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 651.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'}
```

(x) import gradio as gr  
import numpy as np

def sepia(input\_img):  
 sephia\_filter = np.array([[.393, .769, .189],  
 [.349, .686, .168],  
 [.272, .534, .131]])  
 sephia\_img = input\_img.dot(sephia\_filter.T)  
 sephia\_img /= sephia\_img.max()  
 return sephia\_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 652.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'}
```

(x) import gradio as gr  
import numpy as np

```
def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img
```

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()

✓ 3s completed at 5:03 PM

The screenshot shows a software interface for building a Gradio web application. At the top, there's a navigation bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'All changes saved'. To the right of the status are 'Comment', 'Share', and a user profile icon. Below the navigation is a toolbar with icons for RAM (green checkmark), Disk (grey), and various editing functions like 'Editing'. On the left, there's a sidebar with icons for file operations like '+ Code' and '+ Text', search ('Q'), and a refresh/copy/paste section. The main area is a code editor with a syntax-highlighted Python script. The script defines a 'sepia' function that takes an input image, applies a sepia filter (a 3x3 matrix), and returns the processed image. It then creates a Gradio interface named 'iface' using this function and launches it. A status bar at the bottom indicates the process completed successfully in 3 seconds at 5:03 PM.

**Timestamp: 653.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 654.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 655.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] view the api | built with

```
(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 656.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] view the api | built with

```
(x) import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           ...
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 657.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,  
'http://127.0.0.1:7862/'  
'https://44140.gradio.app'
```

(x)

```
import gradio as gr  
import numpy as np  
  
def sepia(input_img):  
    sepia_filter = np.array([[.393, .769, .189],  
                           [.349, .686, .168],  
                           [.272, .534, .131]])  
    sepia_img = input_img.dot(sepia_filter.T)  
    sepia_img /= sepia_img.max()  
    return sepia_img  
  
iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
iface.launch()
```

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 658.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[15] view the api | built with

```
<Flask 'gradio.networking'>,  
'http://127.0.0.1:7862/'  
'https://44140.gradio.app'
```

(x)

```
import gradio as gr  
import numpy as np  
  
def sepia(input_img):  
    sepia_filter = np.array([[.393, .769, .189],  
                           [.349, .686, .168],  
                           [.272, .534, .131]])  
    sepia_img = input_img.dot(sepia_filter.T)  
    sepia_img /= sepia_img.max()  
    return sepia_img  
  
iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")  
  
iface.launch()
```

RAM Disk Editing

3s completed at 5:03 PM

**Timestamp: 659.00 seconds**

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

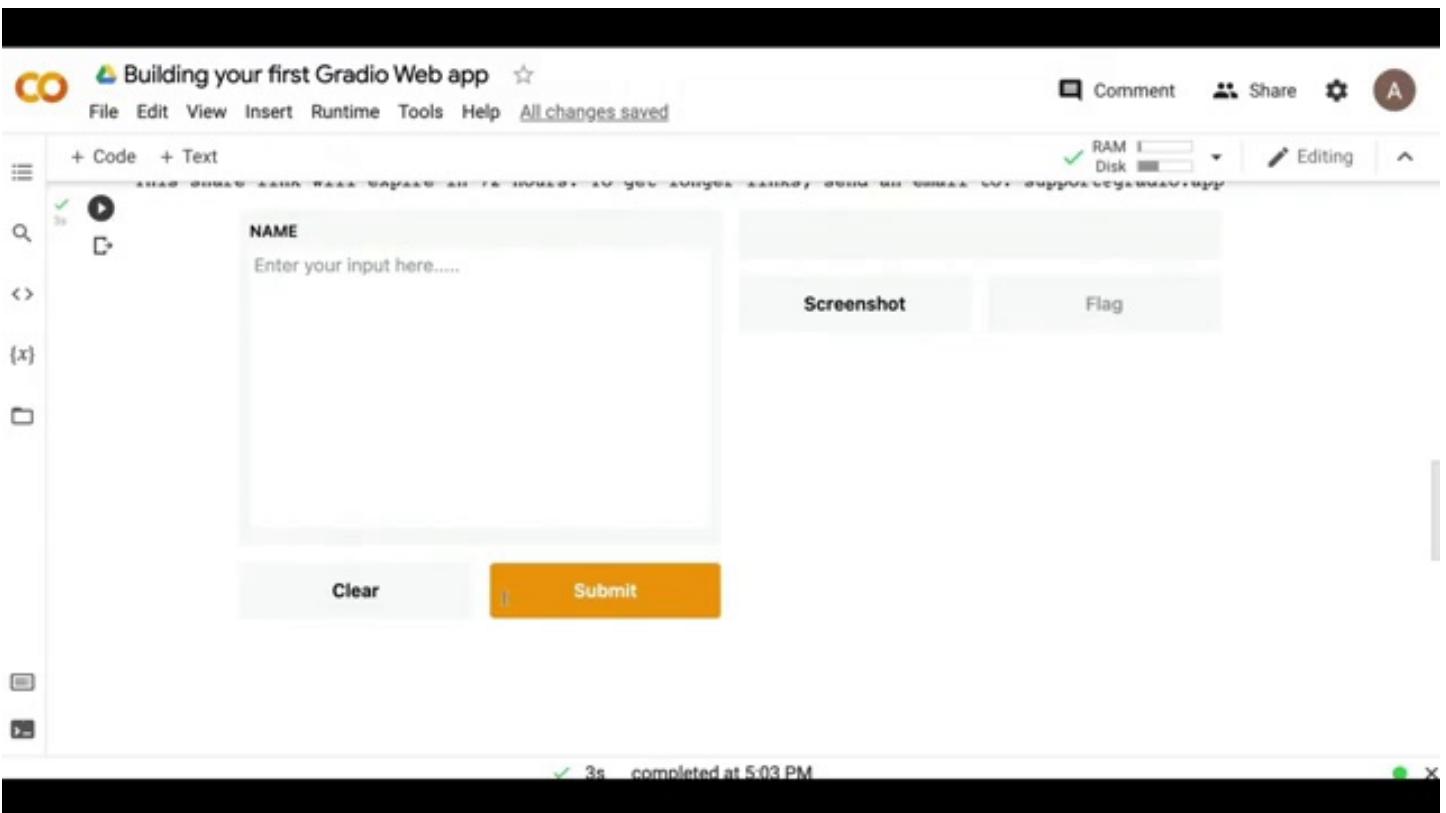
RAM Disk ✓ Editing A

NAME  
Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM



**Timestamp: 660.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[10] [  
↳ (<Flask 'gradio.networking',  
'http://127.0.0.1:7860/',  
'https://31668.gradio.app')  
]

{x} [11] interface = gr.Interface(fn = hello\_world,  
inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),  
outputs = "text")

[15] interface.launch()  
Colab notebook detected. To show errors in colab notebook, set "debug=True" in "launch()"  
Running on public URL: <https://44140.gradio.app>  
This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 661.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

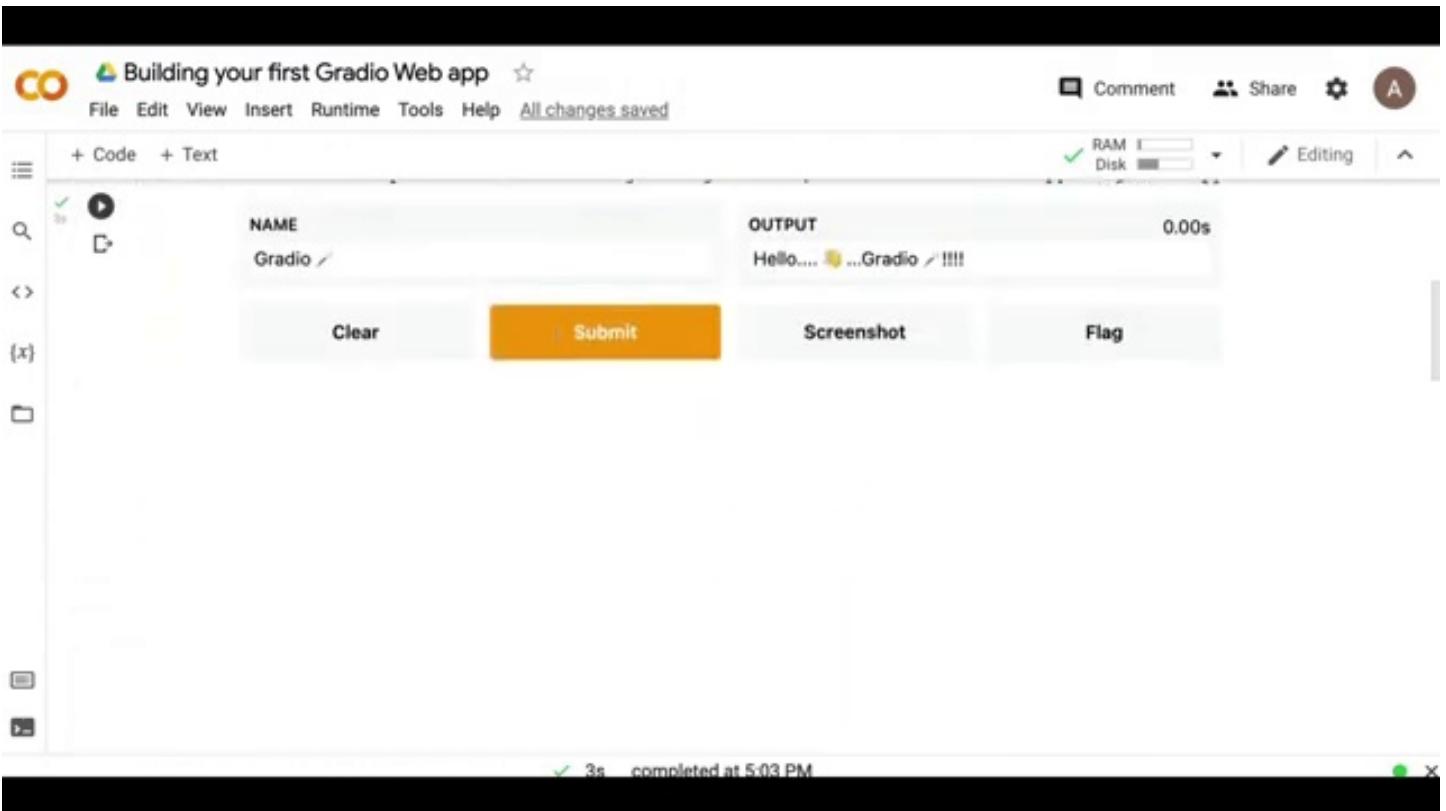
RAM Disk Editing A

NAME OUTPUT 0.00s

Gradio Hello.... 🎉 ...Gradio ✨ !!!!

Clear Submit Screenshot Flag

3s completed at 5:03 PM



**Timestamp: 662.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] ! pip install gradio

<> ✓ [2] import gradio as gr

{x} ✓ [3] def hello\_world(name):  
0s return "Hello.... 🎉 ..."+name + "!!!!"

✓ [4] hello\_world("llittlecoder")

'Hello.... 🎉 ...llittlecoder!!!!'

✓ [5] interface = gr.Interface(fn = hello\_world, inputs = 'text', outputs = "text")

✓ [6] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://31668.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to support@gradio.app

✓ 3s completed at 5:03 PM

RAM Disk Editing

**Timestamp: 663.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

```
[1] (<Flask 'gradio.networking'>,
     'http://127.0.0.1:7860/',
     'https://31668.gradio.app')

[2] [14] interface = gr.Interface(fn = hello_world,
                                inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                                outputs = "text")

[3] [15] interface.launch()

Colab notebook detected. To show errors in colab notebook, set "debug=True" in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME
Enter your input here.....
Screenshot Flag
```

✓ 3s completed at 5:03 PM X

**Timestamp: 664.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
[10] (<Flask 'gradio.networking'>,
      'http://127.0.0.1:7860/',
      'https://31668.gradio.app')

[11] interface = gr.Interface(fn = hello_world,
                             inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),
                             outputs = "text")

[12] interface.launch()

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://44140.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM

**Timestamp: 665.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 666.00 seconds**

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

[10] (<Flask 'gradio.networking'>,  
      '<https://127.0.0.1:7860/>',  
      '<https://31668.gradio.app>' )

(x) interface = gr.Interface(fn = hello\_world,  
                                  inputs = gr.inputs.Textbox(lines = 10, placeholder="Enter your input here....."),  
                                  outputs = "text")

[15] interface.launch()

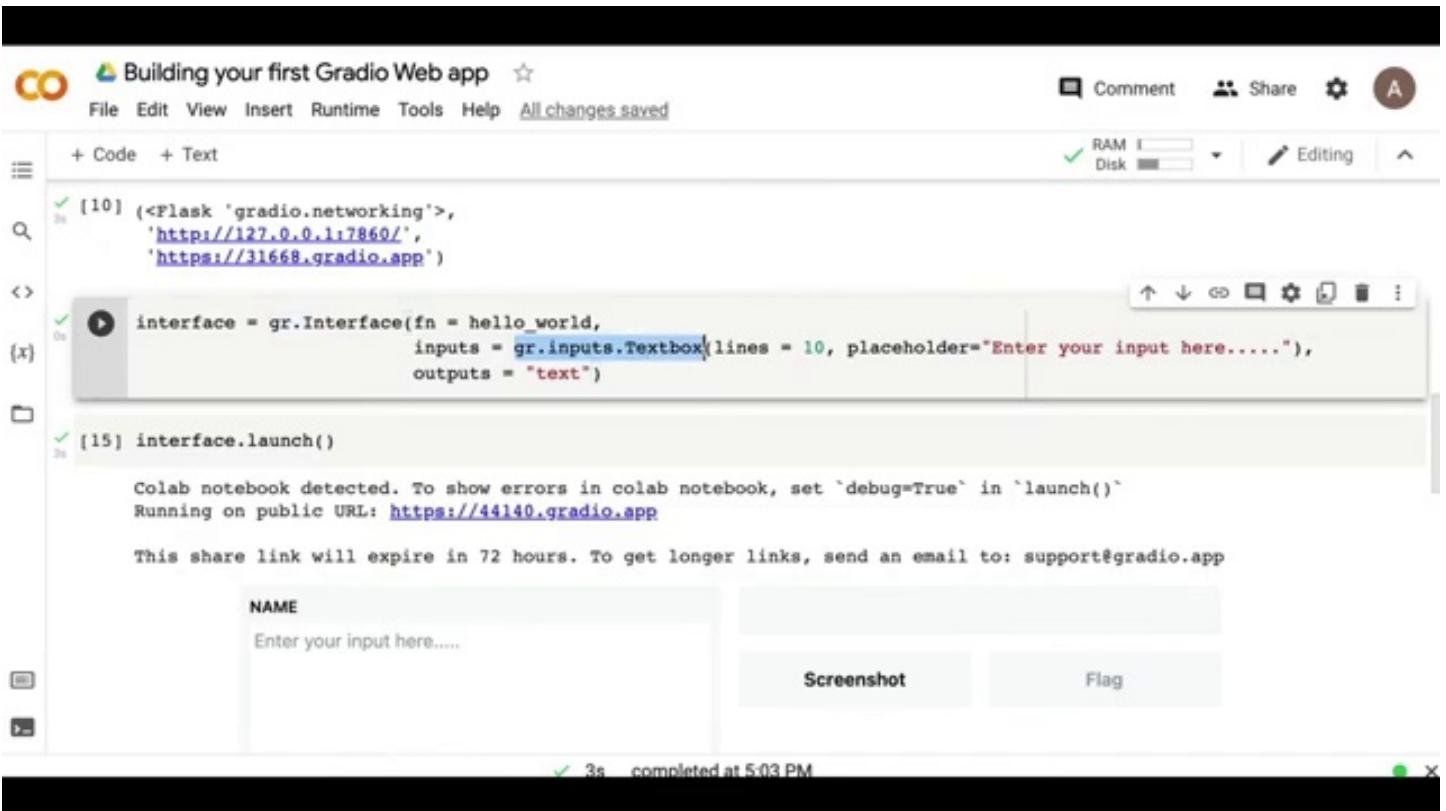
Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://44140.gradio.app>

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app

NAME  
Enter your input here....

Screenshot Flag

✓ 3s completed at 5:03 PM



**Timestamp: 667.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

RAM Disk Editing A

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM

**Timestamp: 668.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

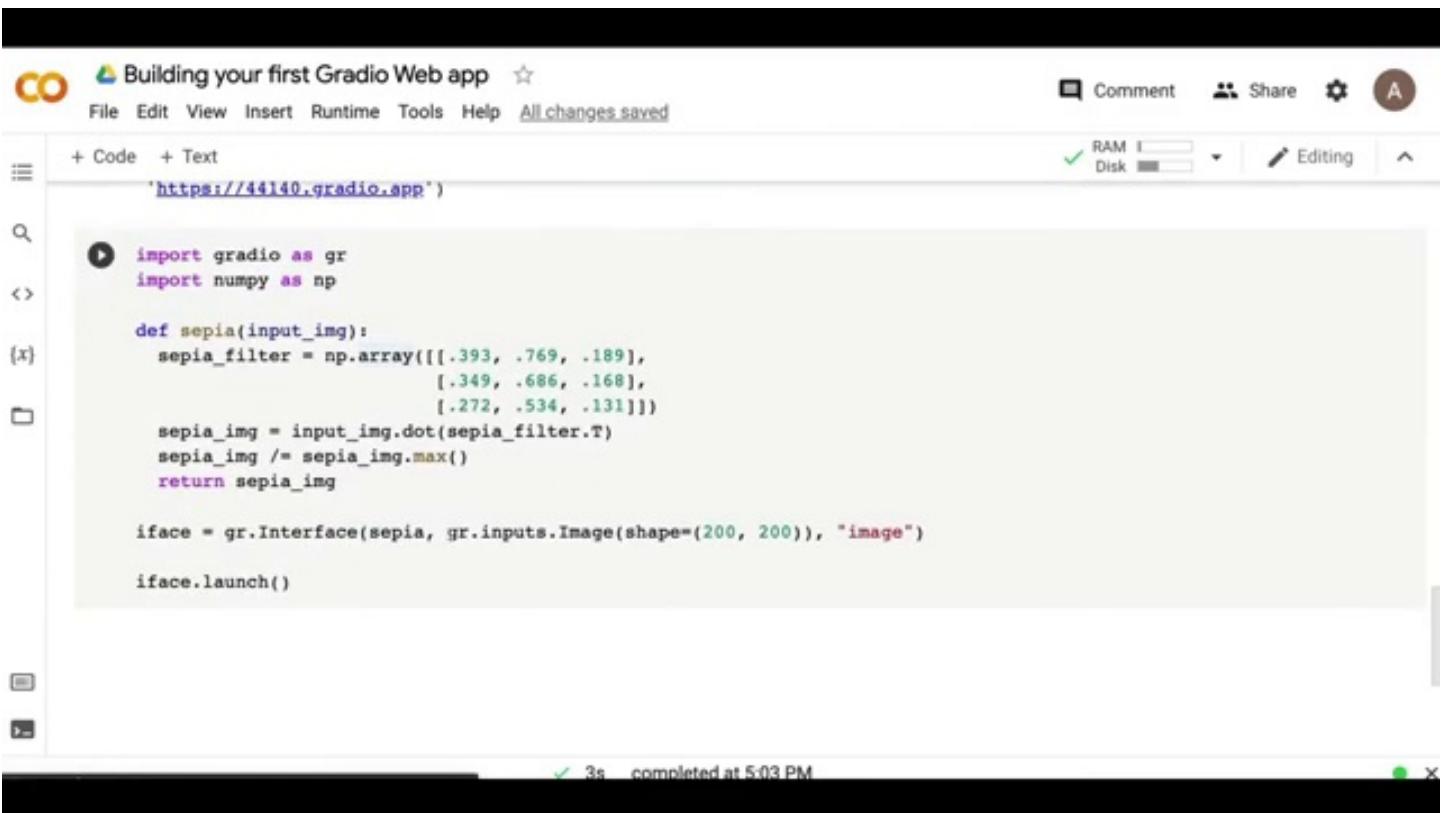
RAM Disk Editing A

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 669.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

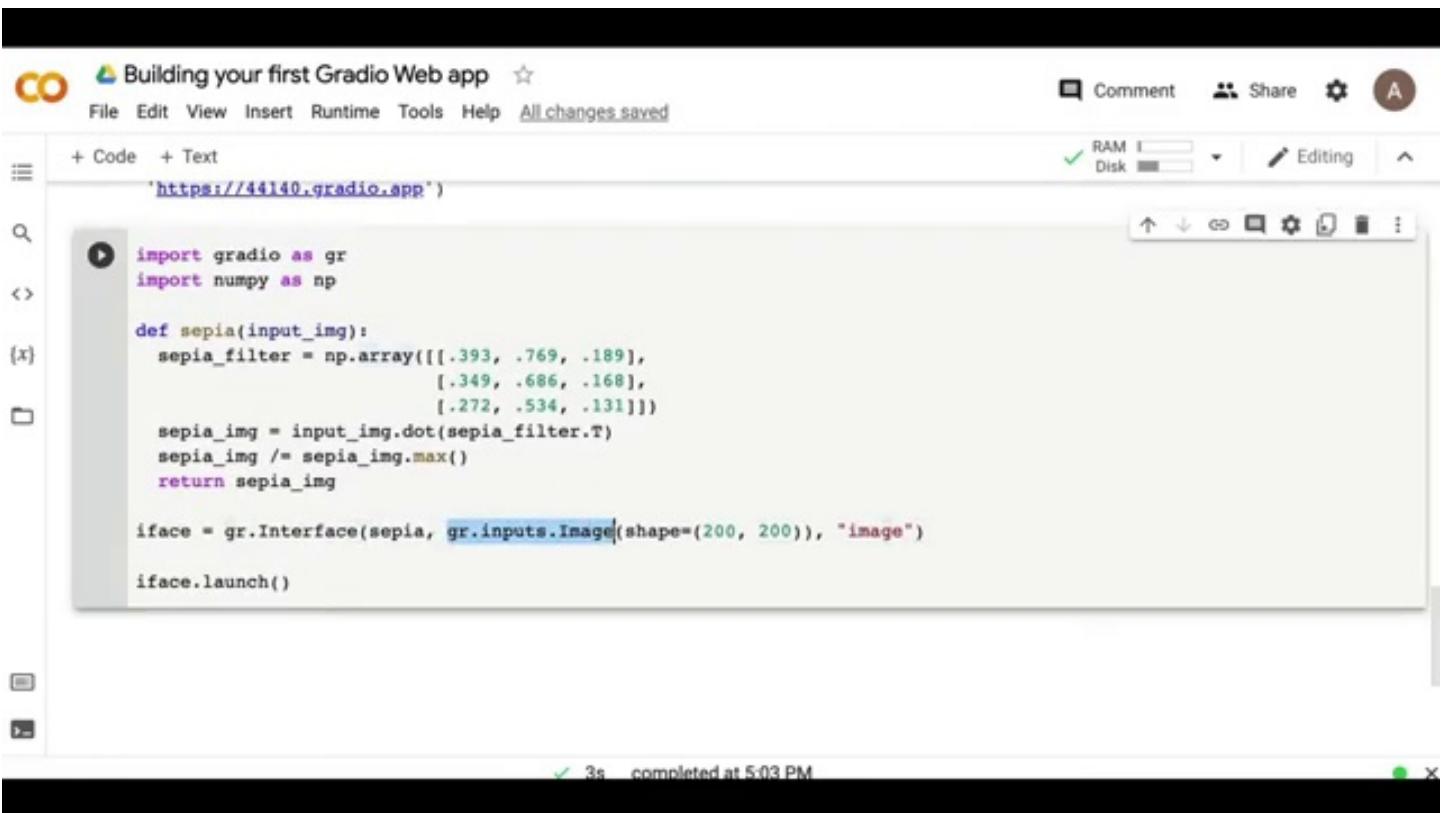
RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 670.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

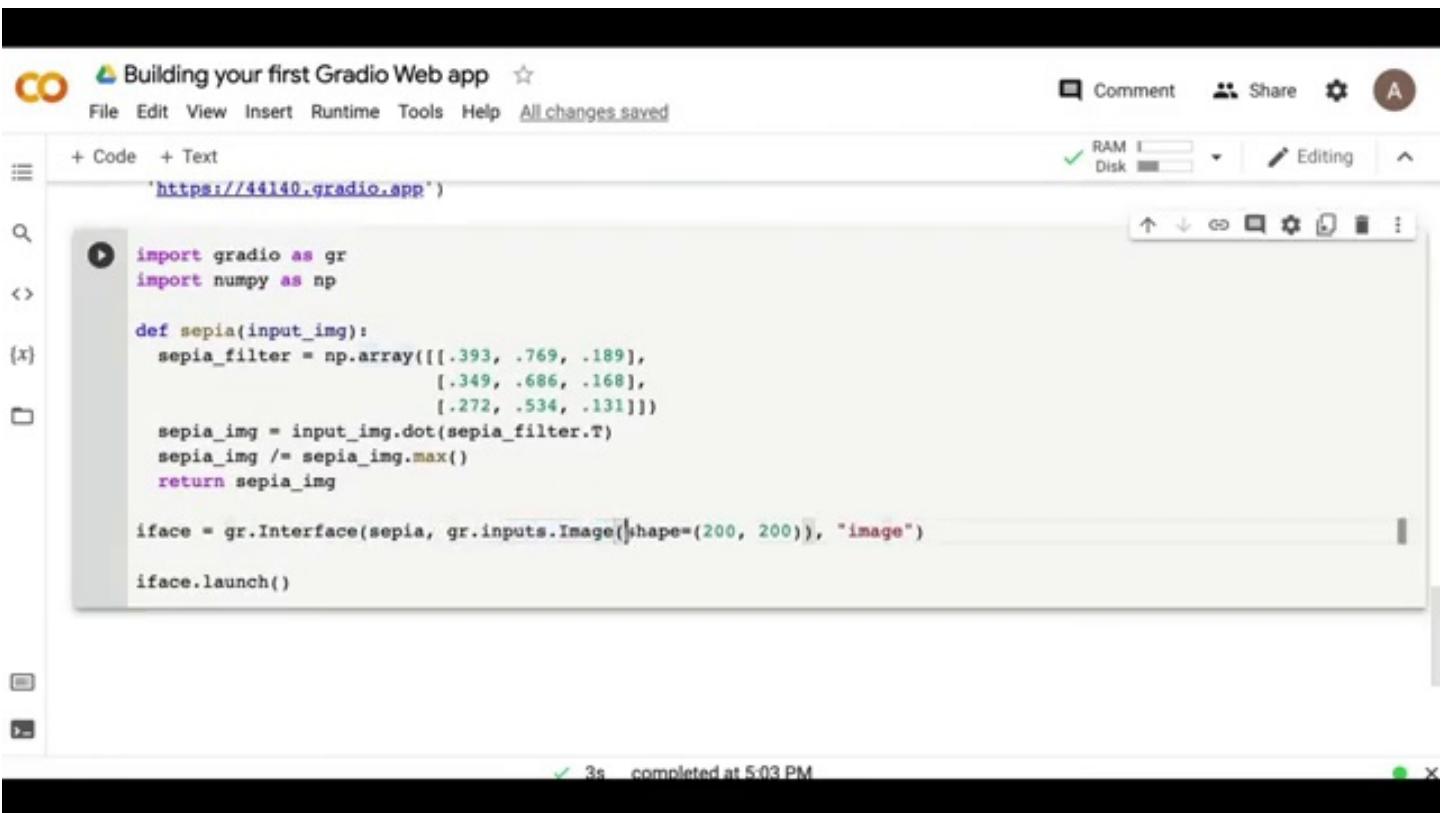
RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 671.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

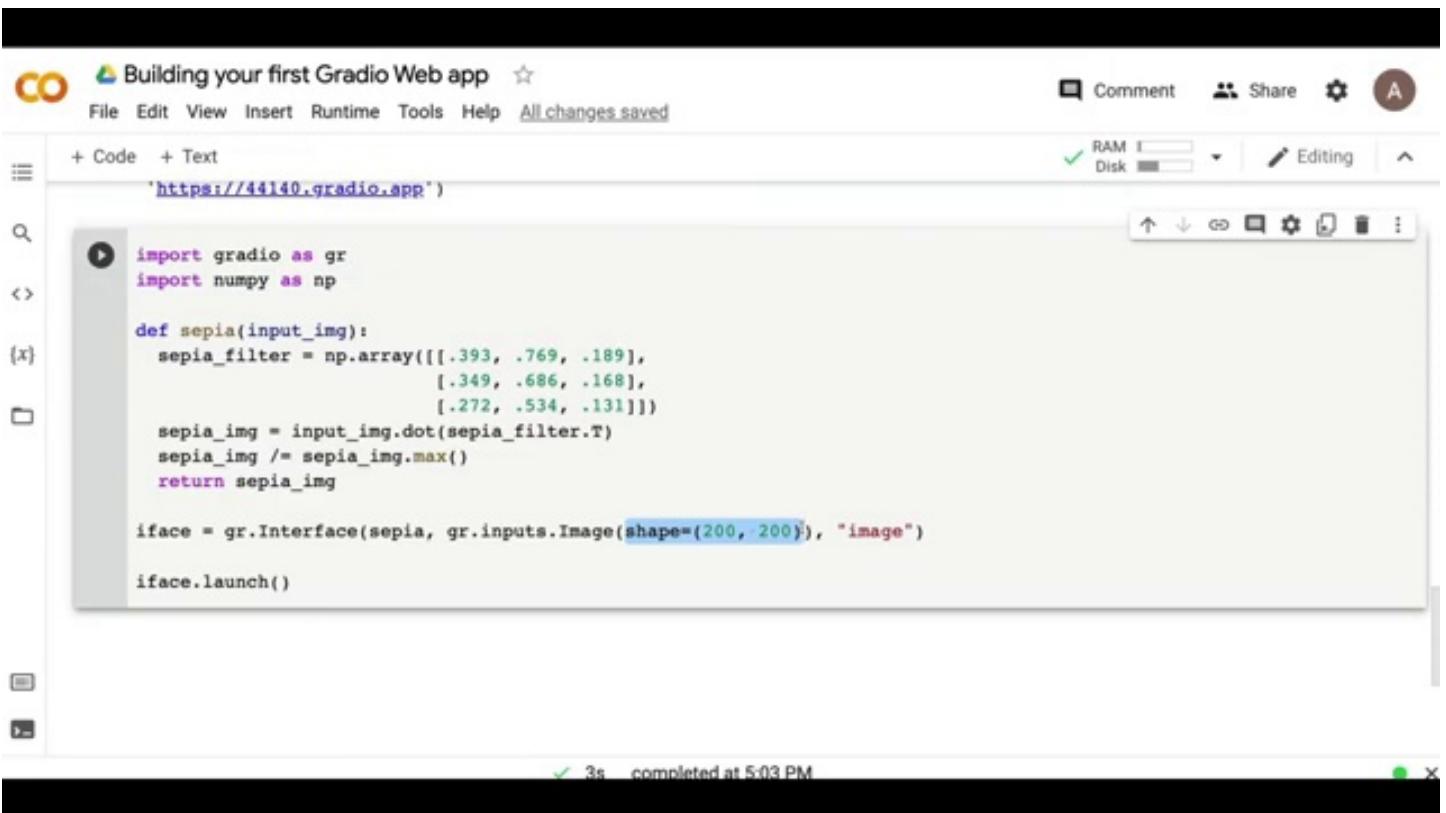
RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 672.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

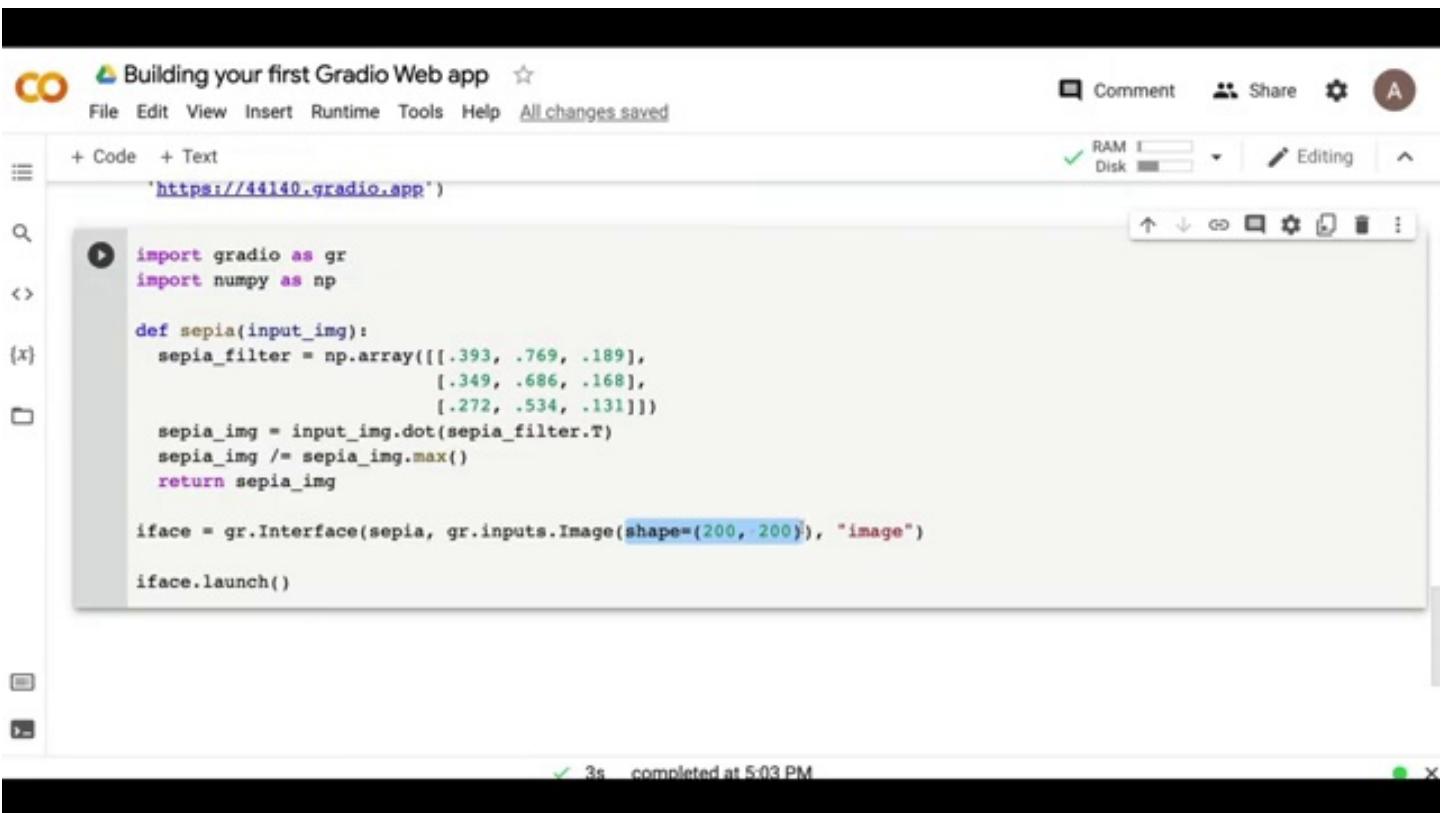
RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 673.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

✓ 3s completed at 5:03 PM

**Timestamp: 674.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

RAM Disk Editing

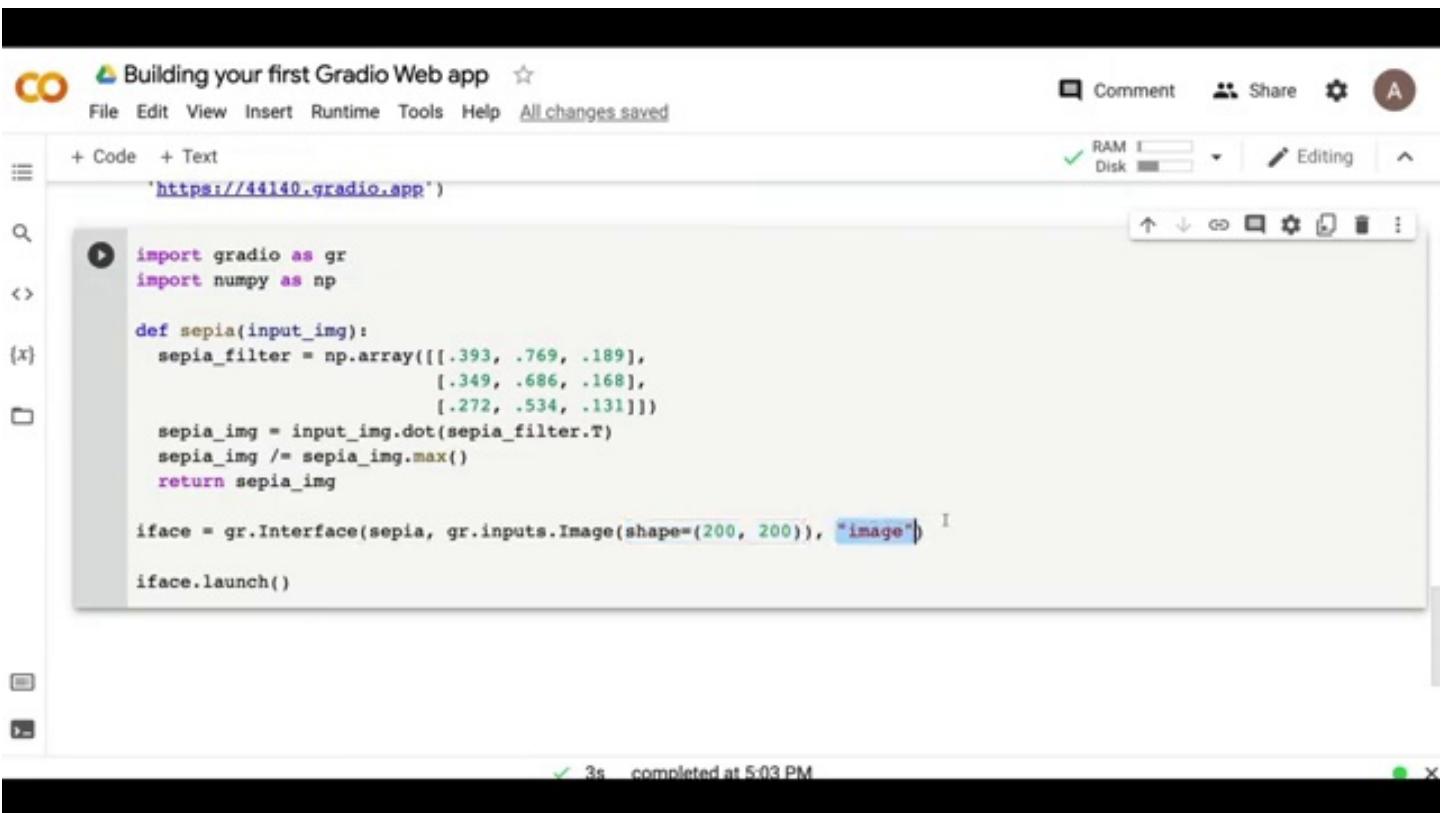
```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 675.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text <https://44140.gradio.app>

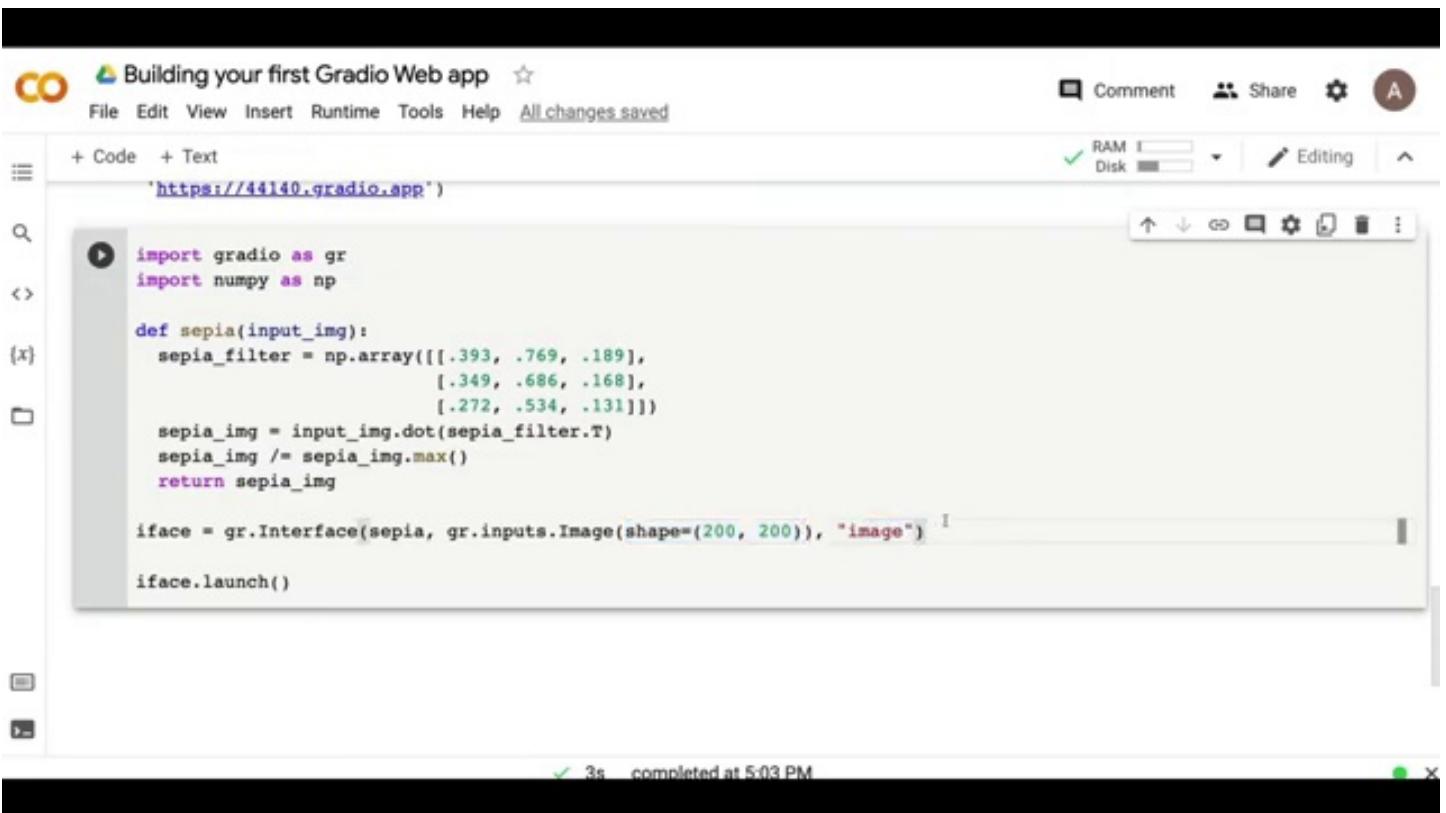
RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sephia_filter = np.array([[.393, .769, .189],
                             [.349, .686, .168],
                             [.272, .534, .131]])
    sephia_img = input_img.dot(sephia_filter.T)
    sephia_img /= sephia_img.max()
    return sephia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()
```

✓ 3s completed at 5:03 PM



**Timestamp: 676.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `f`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

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

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

**Timestamp: 677.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .168],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia_img /= sepia_img.max()
    return sepia_img

iface = gr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

Executing (0s)

**Timestamp: 678.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share A

+ Code + Text

RAM Disk Editing

view the api | built with

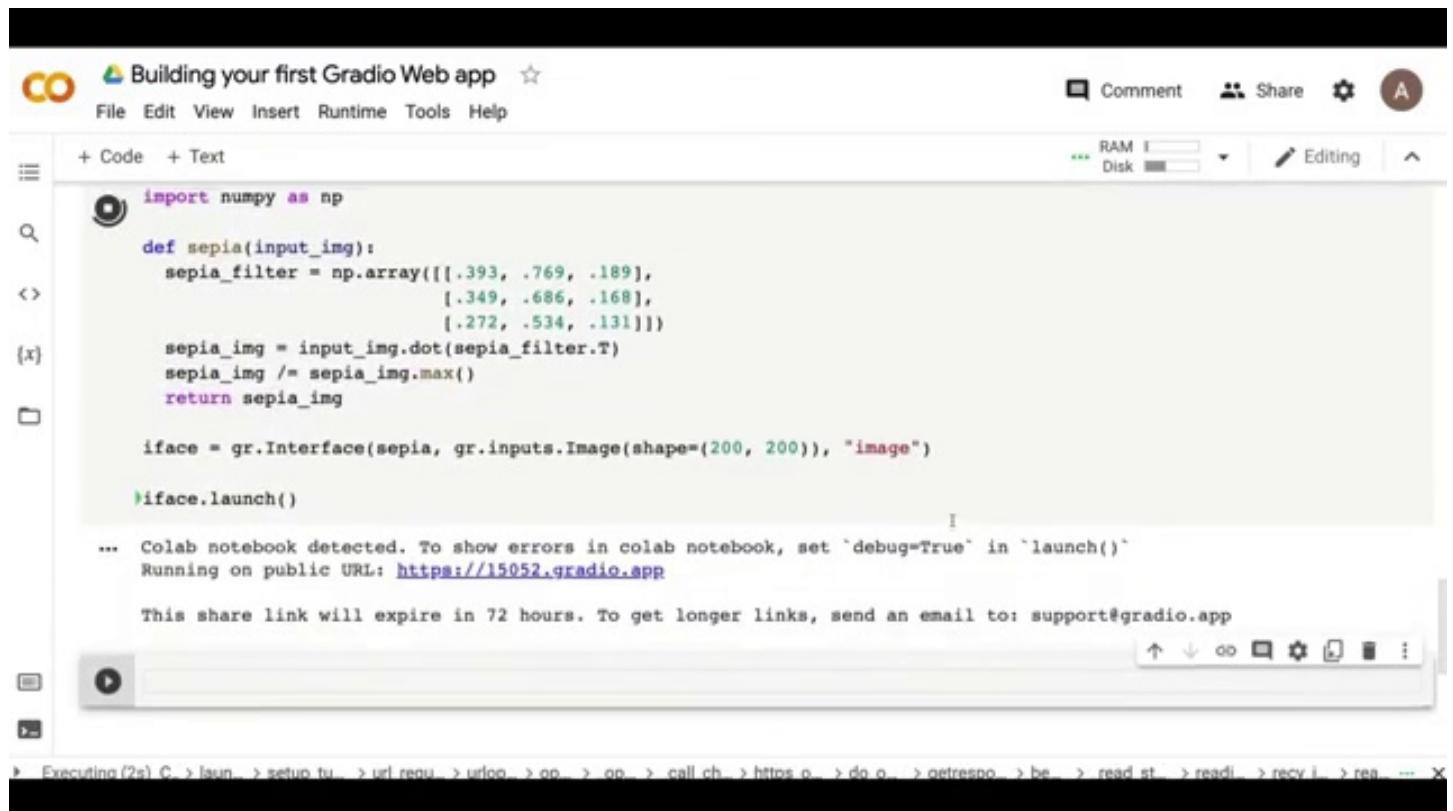
```
<Flask 'gradio.networking'>,
'http://127.0.0.1:7862/',
'https://44140.gradio.app'

import gradio as gr
import numpy as np

def sepia(input_img):
    sepia_filter = np.array([[.393, .769, .189],
                           [.349, .686, .169],
                           [.272, .534, .131]])
    sepia_img = input_img.dot(sepia_filter.T)
    sepia img /= sepia img.max()

> Executing (1s) C > lsun > setup_tf > url_renu > url > op > op > call cb > https o > do o > getresno > be > read st > readl > recy i > ren ... X
```

**Timestamp: 679.00 seconds**



**Timestamp: 680.00 seconds**

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

```
sephia_img /= sephia_img.max()
return sephia_img

iface = gr.Interface(sephia, gr.inputs.Image(shape=(200, 200)), "image")
iface.launch()

(x) Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`
Running on public URL: https://15052.gradio.app

This share link will expire in 72 hours. To get longer links, send an email to: support@gradio.app
```

RAM Disk Editing

Colab notebook detected. To show errors in colab notebook, set `debug=True` in `launch()`  
Running on public URL: <https://15052.gradio.app>

Executing (3s) C > laun... > url > hea... > requ... > requ... > seph... > seph... > urlo... > make\_requ... > netrespon... > hea... > read\_stat... > readin... > recy\_in... > res... ... X

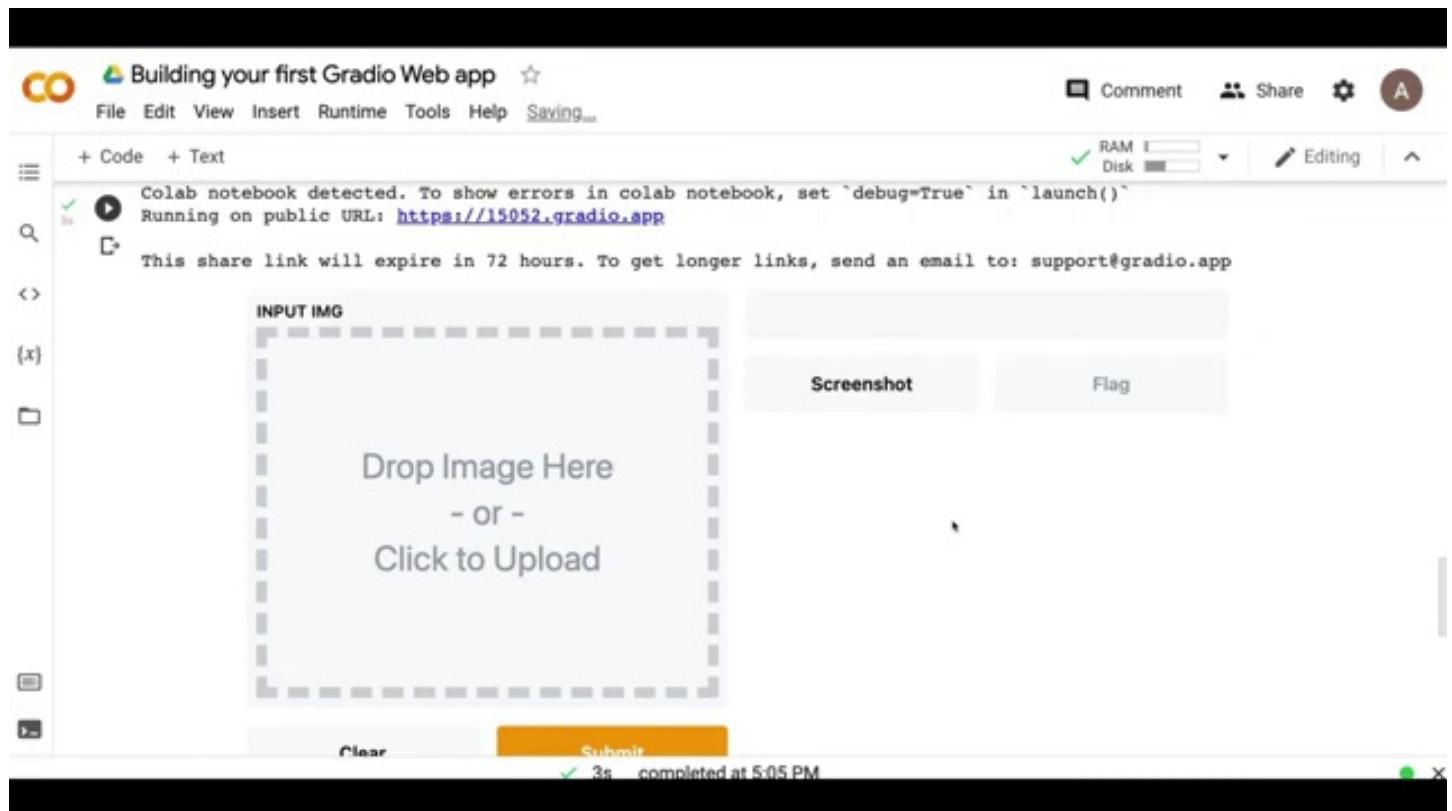
**Timestamp: 681.00 seconds**



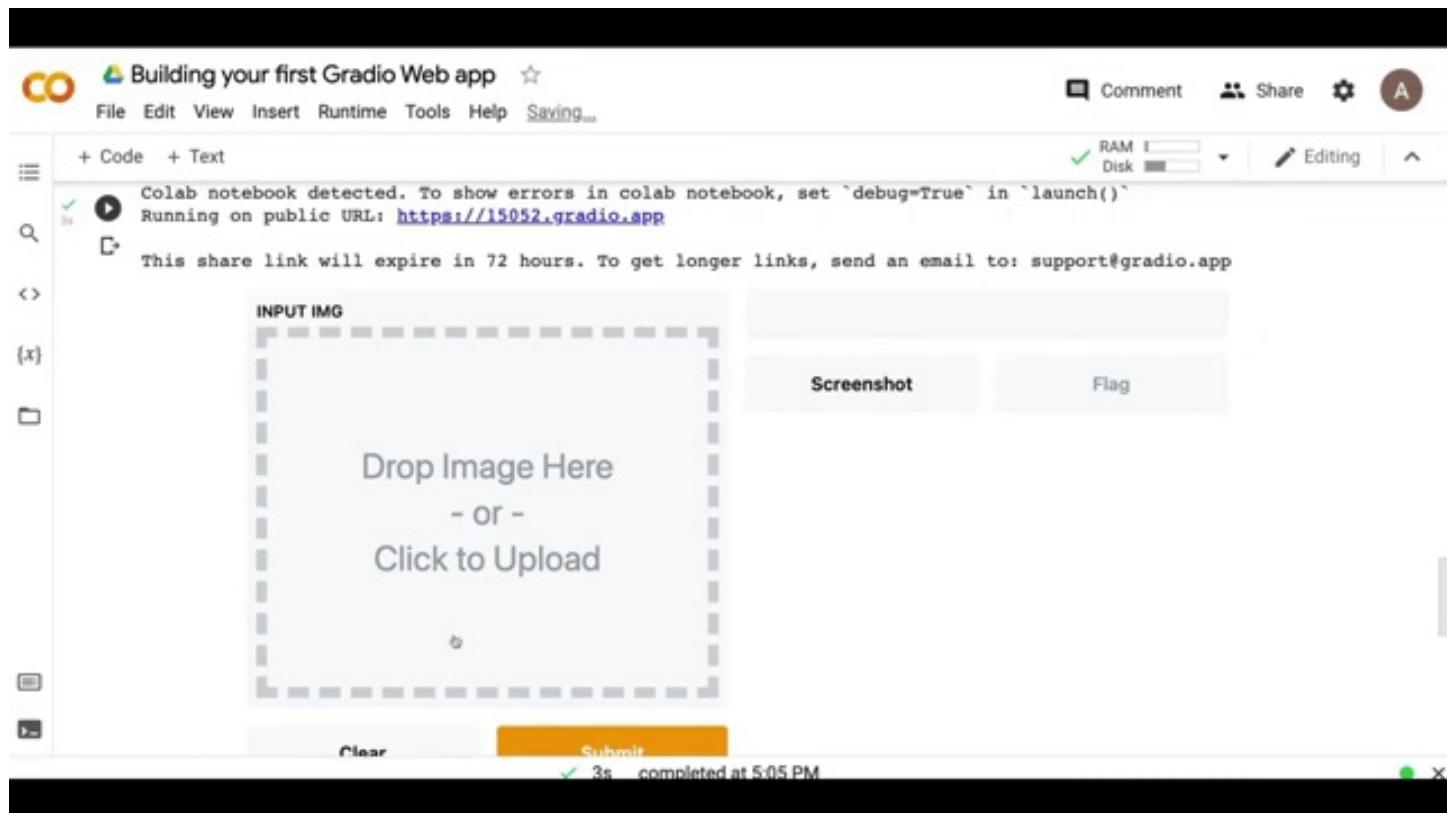
**Timestamp: 682.00 seconds**



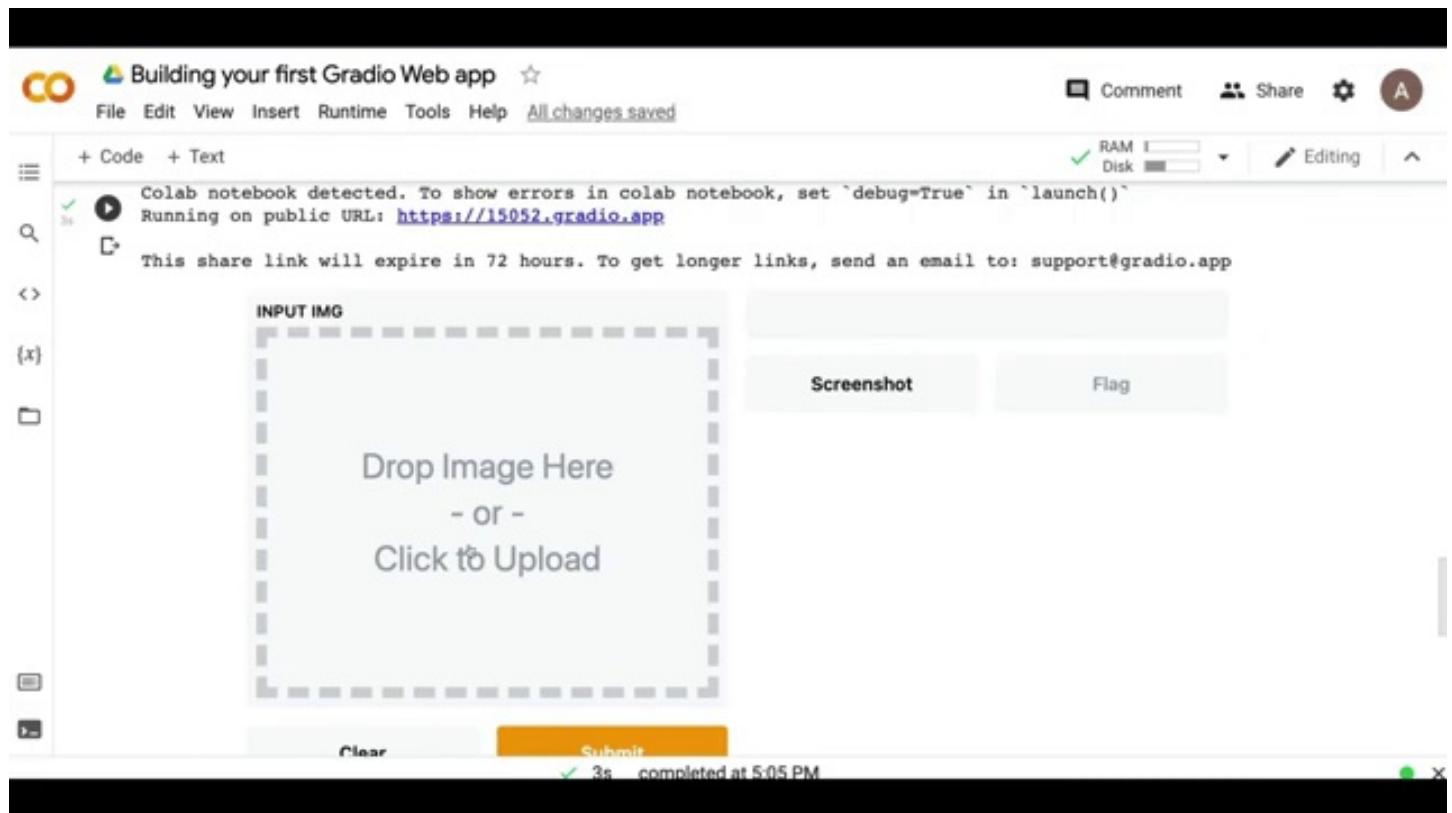
**Timestamp: 683.00 seconds**



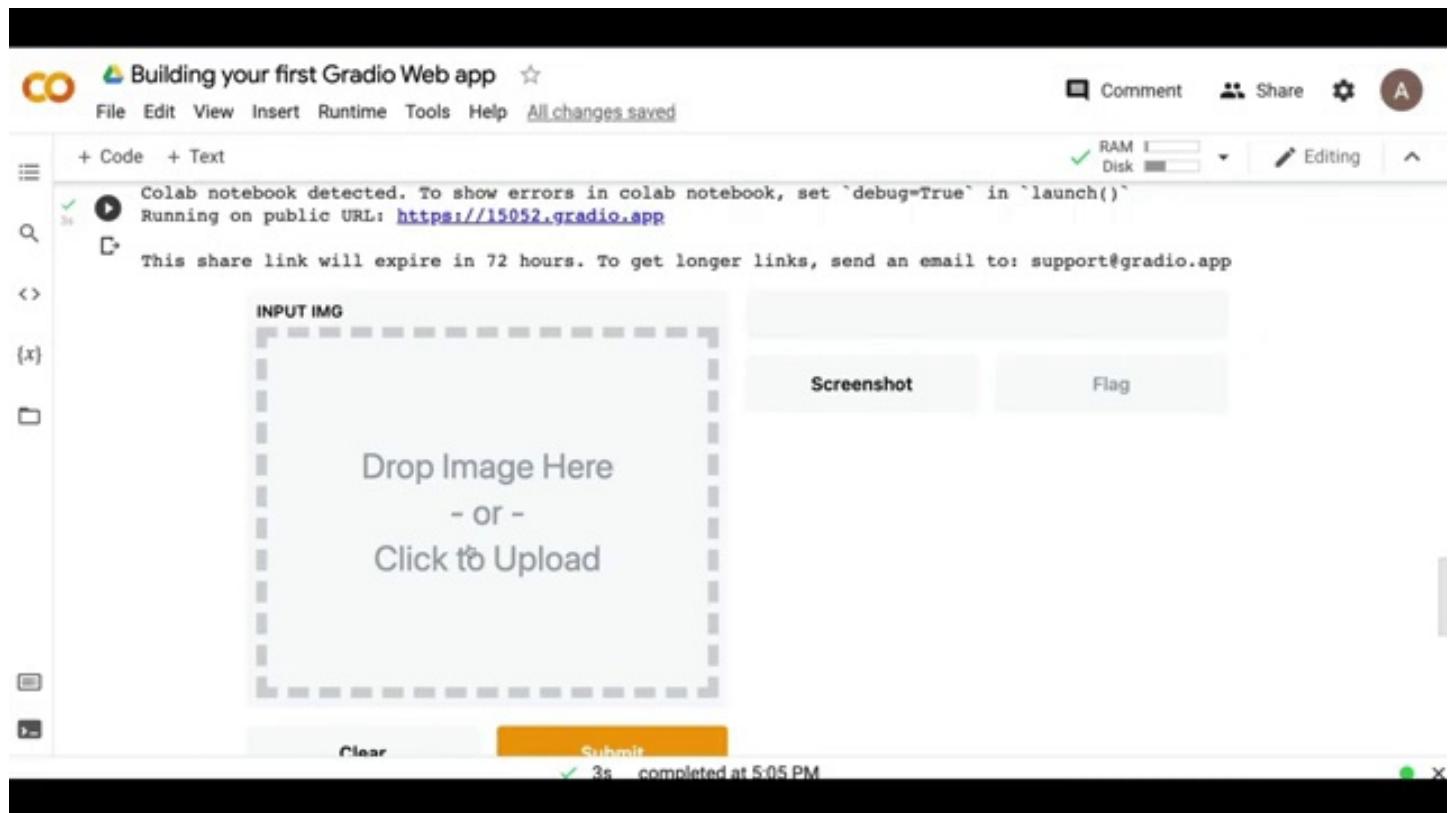
**Timestamp: 684.00 seconds**



**Timestamp: 685.00 seconds**



**Timestamp: 686.00 seconds**



**Timestamp: 687.00 seconds**

## The Interface

Gradio can wrap almost any Python function with an easy to use interface. That function could be anything from a simple tax calculator to a pretrained model.

The core `Interface` class is initialized with three parameters:

- `fn`: the function to wrap
- `inputs`: the input component type(s)
- `outputs`: the output component type(s)

With these three arguments, we can quickly create interfaces and `launch()` them. But what if you want to change how the UI components look or behave?

## Customizable Components

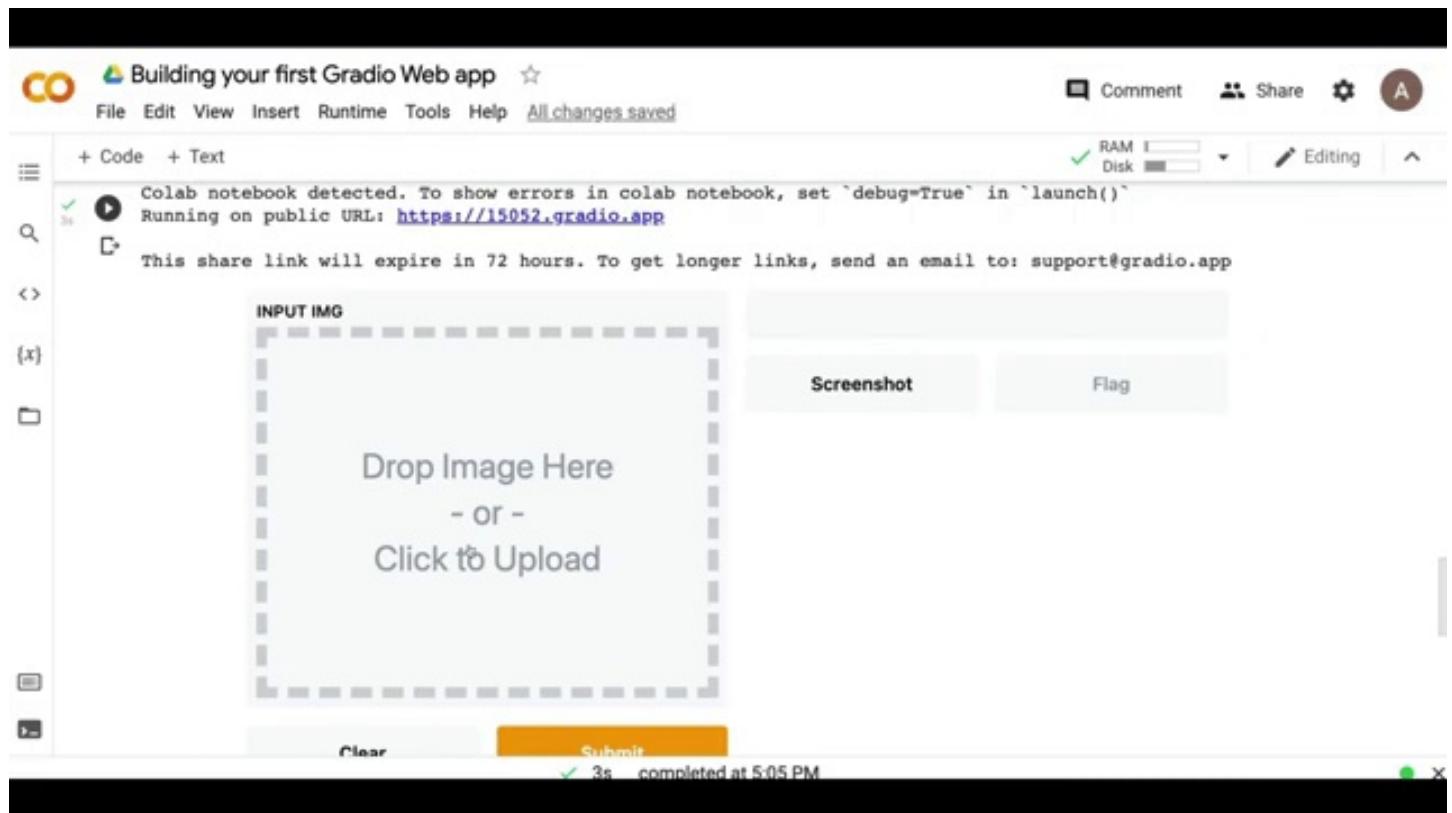
What if we wanted to customize the input text field - for example, we wanted it to be larger and have a text hint? If we use the actual input class for `Textbox` instead of using the string shortcut, we have access to much more customizability. To see a list of all the components we support and how you can customize them, check out the [Docs](#).

```
import gradio as gr

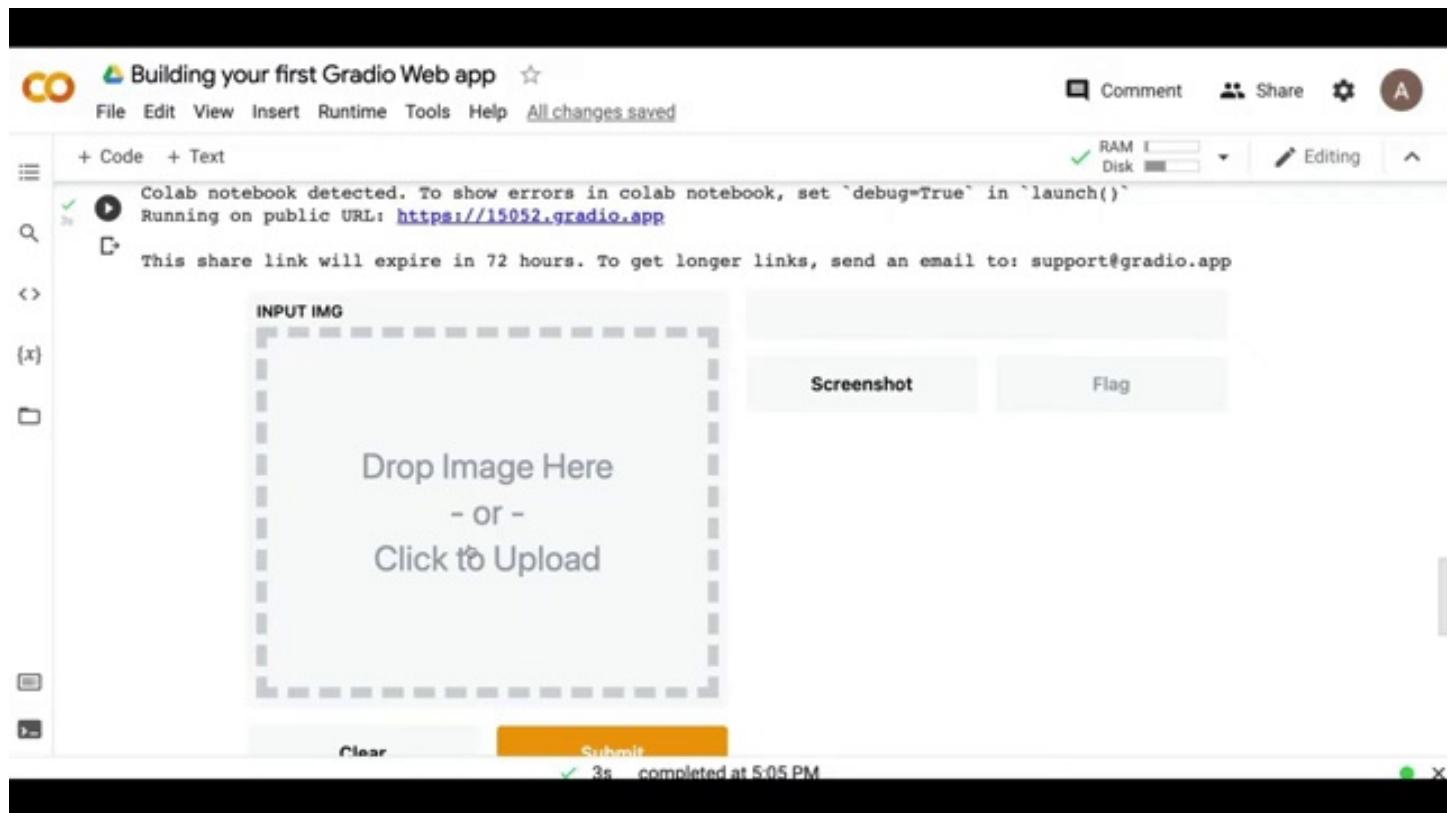
def greet(name):
    return "Hello " + name + "!"

iface = gr.Interface(
    fn=greet,
    inputs=gr.inputs.Textbox(lines=2, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

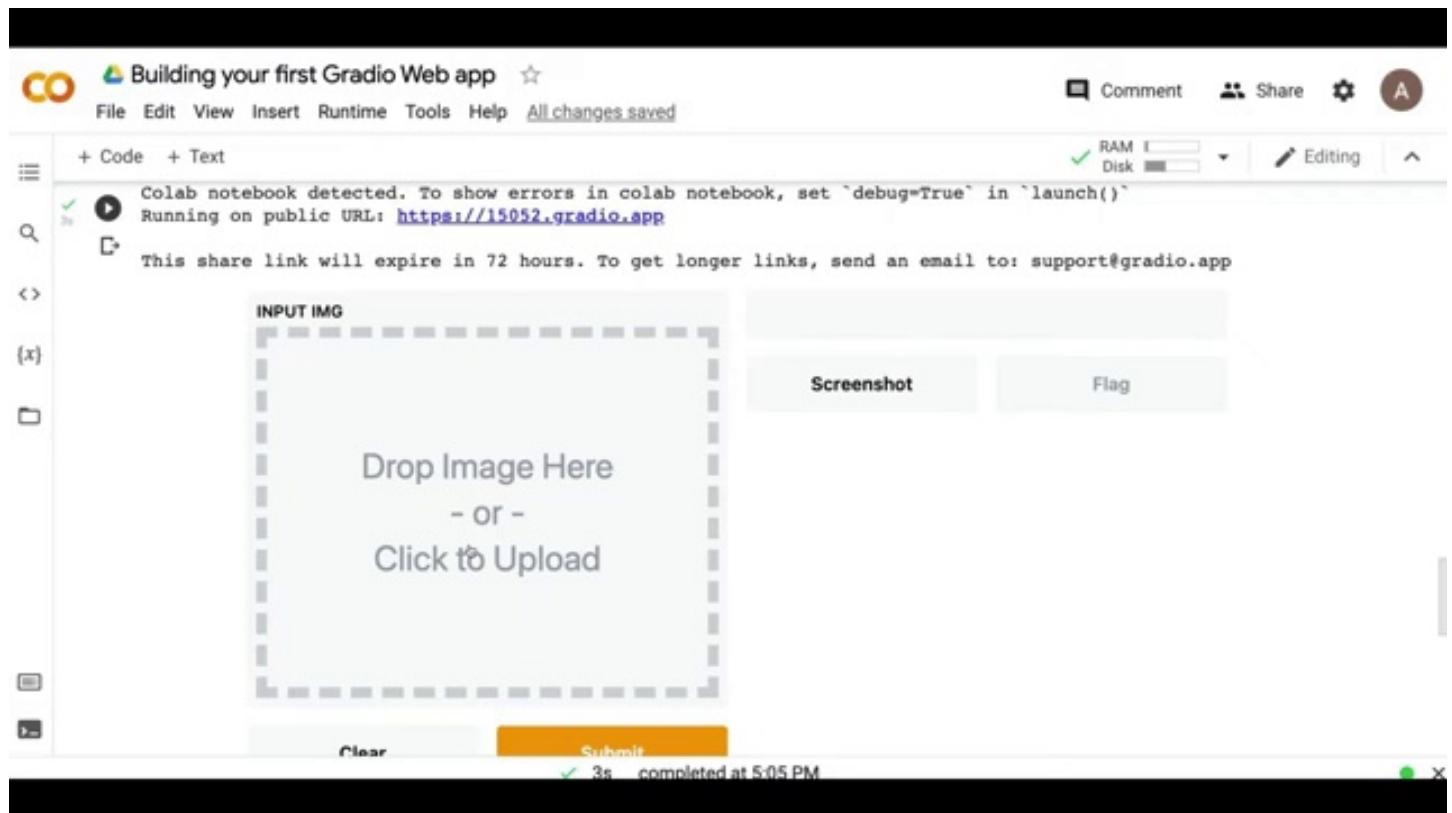
**Timestamp: 688.00 seconds**



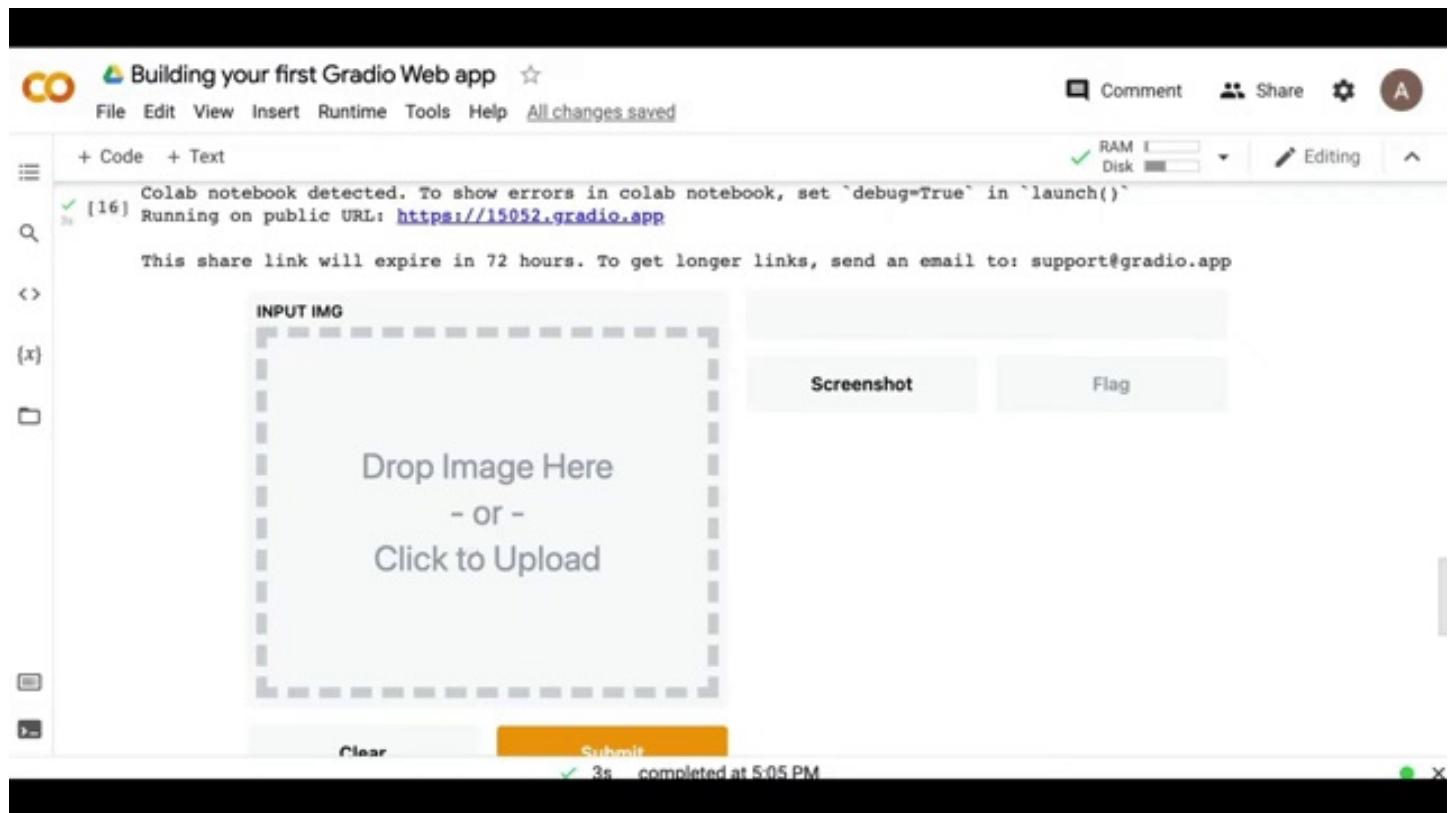
**Timestamp: 689.00 seconds**



**Timestamp: 690.00 seconds**



**Timestamp: 691.00 seconds**



**Timestamp: 692.00 seconds**