

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=3, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

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=3, placeholder="Name Here..."),
    outputs="text")
iface.launch()
```

```
import gradio as gr\n\n\ndef greet(name):\n    return \"Hello \" + name + \"!\"\n\niface = gr.Interface(fn=greet, inputs=\"text\", outputs=\"text\")\niface.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 text input field with the placeholder "Type your name here". 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 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



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=1, placeholder="Name Here..."),
    outputs="text")
```

regression training with a single command, either from the command line or in a script.

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:

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

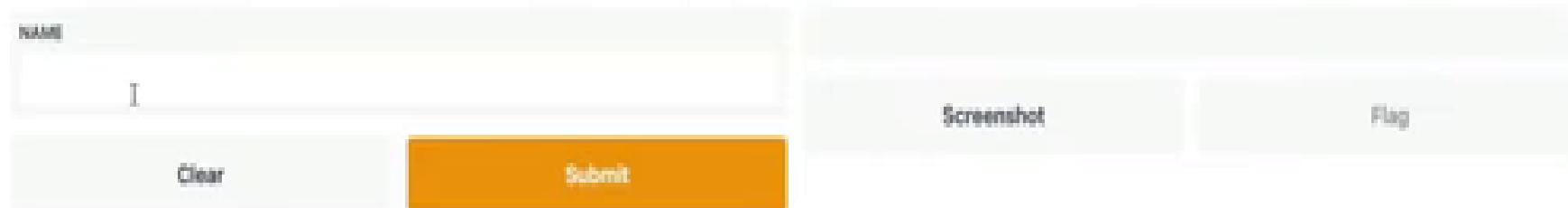
Name Here...

Screenshot

Flag

```
import gradio as gr\n\n\ndef greet(name):\n    return \"Hello \" + name + \"!!!\"\n\niface = gr.Interface(fn=greet, inputs=\"text\", outputs=\"text\")\niface.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:

- `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?

- 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 image shows a Gradio interface window. At the top, there's a title bar with the word 'Name'. Below it is a text input field with a placeholder 'Name Here...'. At the bottom of the window is a large, prominent orange button labeled 'Submit'.



Screenshot

Flag

- **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?

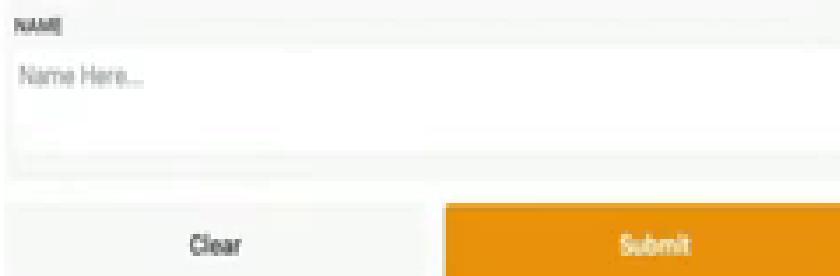
Customizable Components

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 image shows a screenshot of a Gradio interface. It features a single-line text input field with the placeholder text "Name Here...". Below the input field are two buttons: "Clear" on the left and "Submit" on the right, both in white text on a dark background.



- `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, there is a text input field with the placeholder text "Name Here...". Below the input field is a large, solid orange rectangular button with the word "Submit" centered in white text. To the left of the input field, there is a small, faint "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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing

RAM Disk ✓ Editing

interface = gr.Interface(fn=hello_world, inputs='text', outputs='text')

interface.launch()

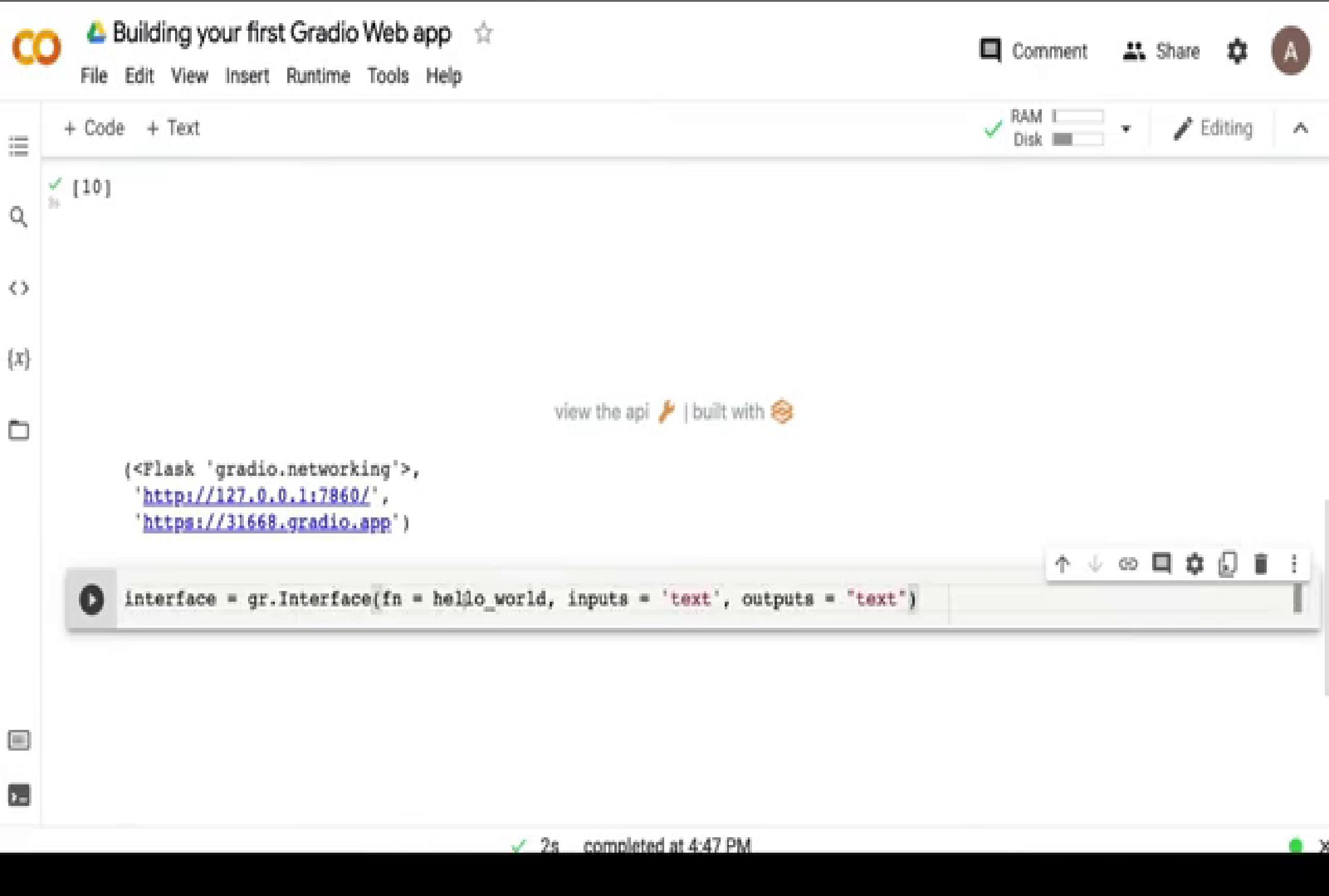
Colab notebook detected. To show errors in colab notebook, set "debug=True" in "launch()"
Running on public URL: <https://31668.gradio.app>

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



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://31660.gradio.app')
```

interface = gr.Interface(fn = hello_world,
 inputs = 'text',
 outputs = "text")

↑ ↓ ⏪ ⏩ ⏴ ⏵ ⏷ ⏸ ⏹ ⏺ !

✓ 2s completed at 4:47 PM

Building your first Gradio Web app ☆

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	0.00s
Gradio /	Hello... 🎉 ...Gradio / !!!!	

Clear Submit Screenshot Flag

✓ 2s completed at 4:47 PM

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_wo
 inputs = gr.i
 outputs = "text")

RAM Disk ✓ Editing

module

{} inputs
{} inspect
{} interface
{} interpretation
ip_address
Interface
get_input_instance
load_interface
get_output_instance
quantify_difference_in_label

Comment Share A

2s completed at 4:47 PM

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 🎨

```
(<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:

| -Textbox(label="None")

✓ 0s completed at 5:03 PM



 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

 Comment  Share  

+ Code + Text

RAM Disk   Editing 

```
[11] interface = qr.Interface(fn = hello_world,
                            inputs = qr.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.gadio.app>

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



Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

This session ends in 1 day, 14 hours. To get average stats, send an email to support@gradioapp.com

✓ RAM: 100% Disk: 10%

Editing

NAME

Enter your input here....

Screenshot Flag

Clear Submit

✓ 3s completed at 5:03 PM

Comment Share A

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ RAM: [] Disk: [] Editing

[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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

<https://21999.gradio.app/>

```
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://21999.gradio.app/>

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

Comment Share ⚙ A

RAM Disk ✓ Editing ^

3s completed at 5:03 PM

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()'

RAM Disk

Editing

Executing (1): Cell > launch() > setup_tunnel() > create_tunnel() > connect() > start_client() > wait() > wait()

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing A

```
✓ ① 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) C_>[run_>url_>hei_>recu_>recu_>sen_>sen_>unloc_>make recu_>netresoon_>ben_>read stat_>readin_>recy in_>res_ X

Building your first Gradio Web app ☆

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

Comment Share A

☰

☰

☰

☰

☰

☰

☰

☰

☰

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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing ^

```
20:  '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

I

3s completed at 5:03 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM: [██████] Disk: [██████]

Editing

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

Clear Submit

✓ 3s completed at 5:03 PM

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

To simply import components and test functionality, or 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()
```



Screenshot

Flag

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')

❶ import gradio as gr
❷ import numpy as np

❸ 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

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

❺ iface.launch()
```

RAM: ✓ Disk: [] Editing

Comment Share ⚙ A

3s completed at 5:03 PM

 Building your first Gradio Web app 

File Edit View Insert Runtime Tools Help All changes saved

 Comment Share A

+ Code + Text

RAM: 1 Disk: 1 Editing ▲

15

view the apollo built with 

```
(<Flask 'qradio.networking'>
  'http://127.0.0.1:7862/',
  'https://44140.qradio.apn')
```

A set of small, light-gray navigation icons located at the bottom of the page. From left to right, they include: a left arrow, a right arrow, a double left arrow, a double right arrow, a magnifying glass, a square, a diamond, a vertical bar, a horizontal bar, and a vertical ellipsis.

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

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) 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

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

Imported

```
import gradio as qr
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 = qr.Interface(sepia, gr.inputs.Image(shape=(200, 200)), "image")

iface.launch()
```

3s completed at 5:03 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing ^

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

Screenshot Flag

Drop Image Here

- OR -

Click to Upload

Clear Submit

✓ 3s completed at 5:05 PM

00:05:49

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM: [] ✓ Disk: [] Editing

INPUT IMG



✓ X Screenshot Flag

Clear Submit

✓ 3s completed at 5:05 PM

Comment Share A

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share ⚙ A

+ Code + Text

✓ RAM: 1 Disk: 1 Editing ^

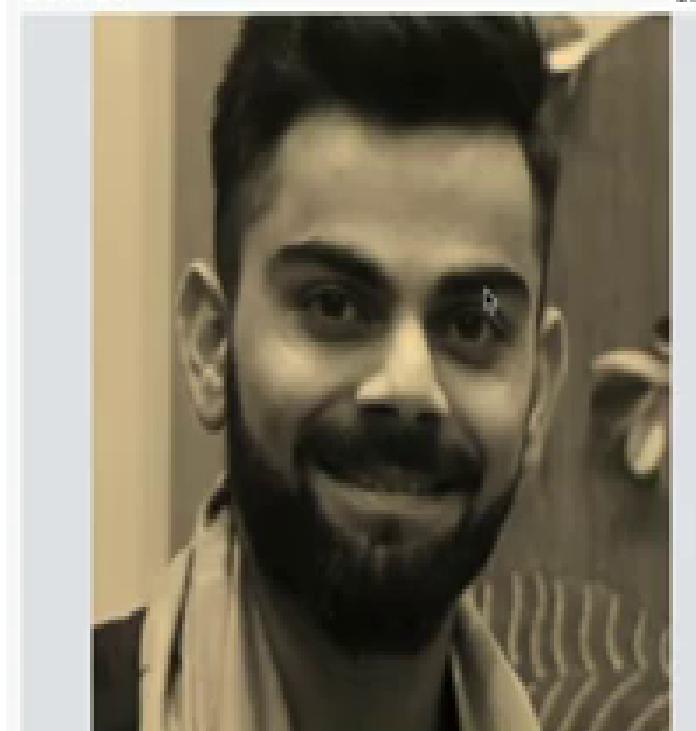
Running on public URL: <https://15052.gradio.app>

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

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

Comment Share ⚙ A

+ Code + Text

RAM: [] Disk: [] Editing

```

❶ import gradio as gr
import numpy as np

❷ 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

❸ 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

 Building your first Gradio Web app 

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM: 1 Disk: 1 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

The image shows a software interface for image processing. On the left, under the heading "INPUT IMG", there is a small thumbnail of a scene with a white wall, a brown door, and a dark object. To its right is a large, blurry thumbnail labeled "OUTPUT". Above the output thumbnail, the text "0.01s" indicates the processing time. At the bottom right of the interface, there is a red "X" button.

✓ 35 completed at 5:05 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing ^

[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

 Building your first Gradio Web app 

File Edit View Insert Runtime Tools Help All changes saved

 Comment Share A

+ 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@radio.ac

✓ 35 completed at 5:05 PM

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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

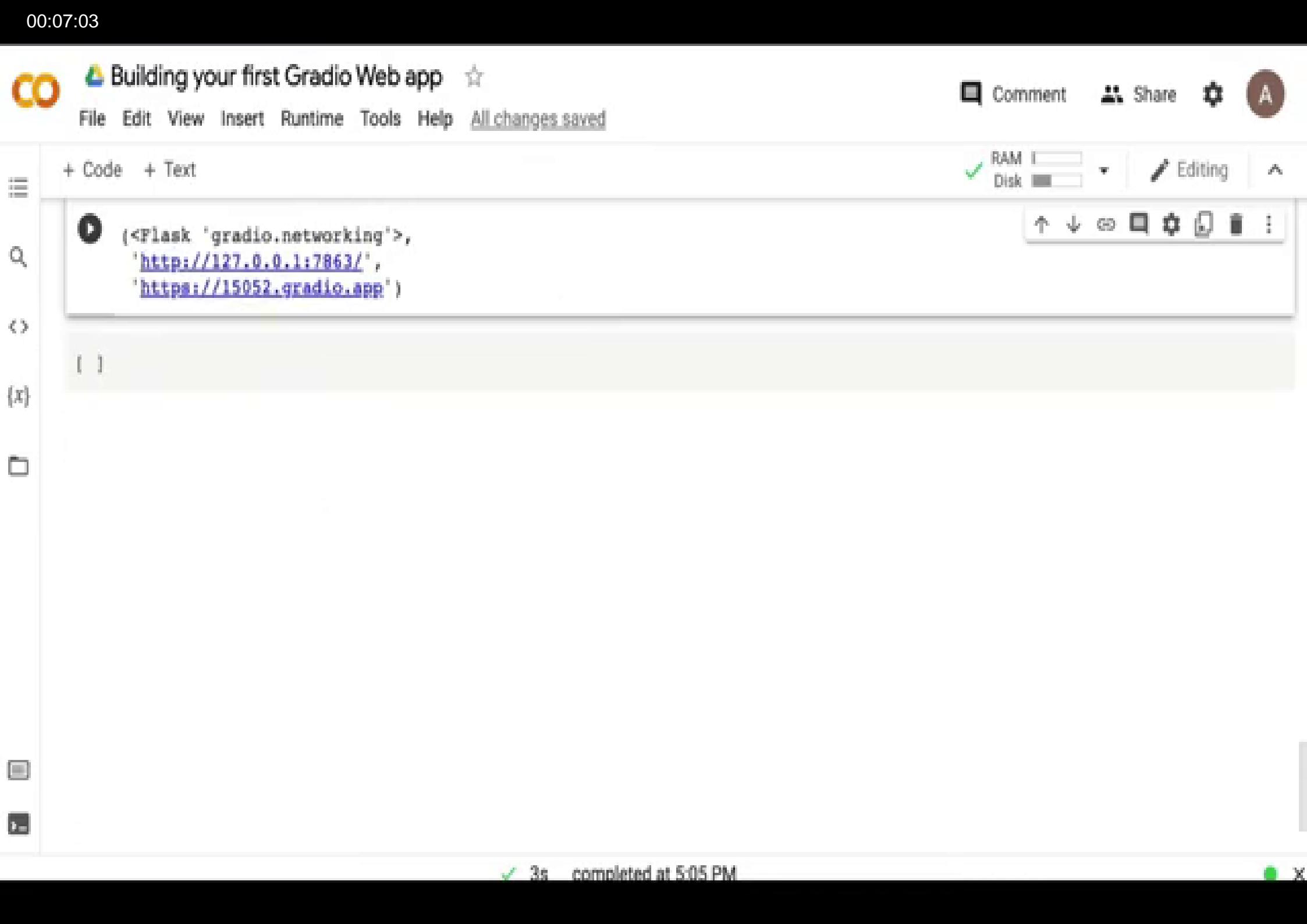
RAM Disk ✓ Editing

Q A

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

()

3s completed at 5:05 PM



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.polyfit(np.polyfit([0, 1, 2], row, 2)), 0, sales_data)
    , projected_months = np.repeat(np.expand_dims(
        np.arange(0, 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": "John", "Jan": 12, "Feb": 14, "Mar": 18}, {"Name": "Alice", "Jan": 14, "Feb": 17, "Mar": 21}, {"Name": "Sara", "Jan": 8, "Feb": 9.5, "Mar": 12}])
),
[
    "dataframe",
    "plot",
    "numpy"
],
description="Please enter the names of your employees along with their Jan, Feb, and Mar sales values. The script will then generate a plot showing projected sales for the next 12 months.")
```

00:07:09

```

regression_values = np.apply_along_axis(lambda row:
    np.array(np.polyfit(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, "21": 21}, {"Name": "Sara", "4": 4, "9.5": 9.5, "12": 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
)
iface.launch()

```

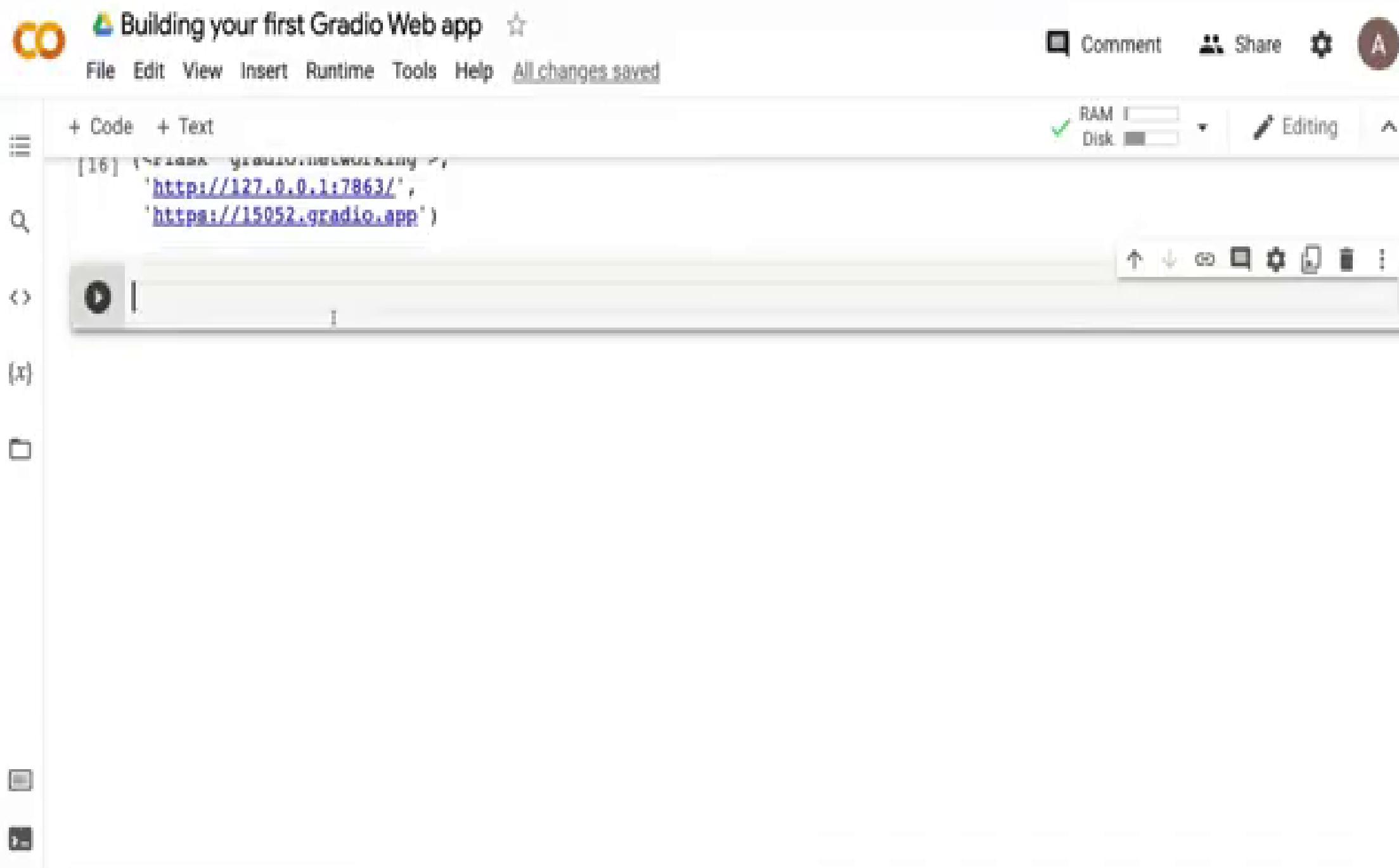
Enter sales figures for employees to predict sales trajectory over year.

EMPLOYEE DATA

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

Screenshot

Flag





 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

 Comment Share

+ Code + Text

RAM: [] Disk: []  Editing 

[view the API](#) | [DRAFT WITH ME](#)

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

```
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
```

✓ 35 completed at 5:05 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

Comment Share ⚙ A

+ Code + Text

✓ RAM: [] Disk: [] Editing



```
import gradio as qr
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 = qr.Interface(sales_projections,
    qr.inputs.DataFrame(
        headers=["Name", "Jan Sales", "Feb Sales", "Mar Sales"],
        default=[["Jon", 12, 14, 18], ["Alice", 14, 17, 2], ["Sana", 8, 9.5, 12]]]
```

✓ 3s completed at 5:05 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk ✓ Editing

```

0
    np.array(np.poly1d(np.polyfit([0,1,2], row, 2))), 0, sales_data)
projected_months = np.repeat(np.expand_dims(
    np.arange(1,12), 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 = qr.Interface(sales_projections,
    qr.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()

```

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

RAM Disk ✓ Editing

0

```

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."

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

 Comment  Share  

+ 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, "Alice": 14, "Sana": 18}, {"Alice": 14, "Sana": 17, "Jon": 2}, {"Sana": 8, "Jon": 9.5, "Alice": 12}]]),
    [
        "dataframe",
        "plot",
        "numpy"
    ],

```

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing

Comment Share A

0 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, "Alice": 14, "Sana": 18}, {"Alice": 14, "Sana": 17, "Jon": 2}, {"Sana": 8, "Jon": 9.5, "Alice": 12}]]
),
[
 "dataframe",
 "plot",
 "numpy"

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM: [] Disk: [] Editing

Comment Share A

```
0
    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()
```

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



Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing

0

```
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
```

iiface = qr.Interface(sales_projections,
qr.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()

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk Editing

```
[ ] "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 (2s) Cell > launch() > setup_tunnel() > create_tunnel() > connect() > start_client() > wait() > wait()

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM Disk ✓ Editing ^

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

3s completed at 5:09 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ 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://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

Clear Submit

OUTPUT 1 0.04s

✓ 3s completed at 5:09 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ 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://31333.gradio.app>

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

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

The chart displays a line graph with three data series: Jon (blue), Alice (orange), and Sana (green). The Y-axis is labeled "Sales" with ticks at 1000 and 1500. The X-axis is labeled "Time" with a tick at "3s". Sana's sales start at 8 and end at 12, while Jon's and Alice's sales remain constant at 12 and 14 respectively.

3s completed at 5:09 PM

 Building your first Gradio Web app

File Edit View Insert Runtime Tools Help Saving...

+ Code + Text

 Comment  Share 

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 = qr.Interface(sales_projections,
    qr.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.")
```

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ RAM: [] Disk: [] Editing

0) iface.launch()

Colab notebook detected. To show errors in colab notebook, set "debug=True" in "launch()"
Running on public URL: <https://31311.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 a line graph with three data series: Jon (blue), Alice (orange), and Sana (green). The Y-axis represents sales values ranging from 0 to 150. The X-axis represents the months Jan, Feb, and Mar. The legend indicates the color mapping for each individual. The chart shows that Sana's sales increased significantly from January to March, while Jon and Alice maintained relatively stable sales levels.

150s · 3s completed at 5:09 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing

0) iface.launch()

Colab notebook detected. To show errors in colab notebook, set "debug=True" in "launch()"
Running on public URL: <https://31111.gradio.app>

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

OUTPUT 3

x	Jon	Alice	Sana
0	-100	100	200
1	-150	150	250
2	-200	200	300
3	-250	250	350
4	-300	300	400
5	-350	350	450
6	-400	400	500
7	-450	450	550
8	-500	500	600

✓ 3s completed at 5:09 PM

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text RAM: [██████] Disk: [██████] Editing

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

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
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

Building your first Gradio Web app

File Edit View Insert Runtime Tools Help

+ Code + Text

Comment Share ⚙ A

RAM: [] Disk: [] Editing ^

```

0 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, 100, 18], ["Alice", 14, 17, 2], ["Sana", 8, 9.5, 12]])
),
[
    "dataframe",
    "plot",
    "numpy"
]

```

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM: [██████████] Disk: [██████]

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

...

(x)

Executing (0s) C_>laun...>version ch...>a...>requ...>requ...>se...>se...>unio...> make req...>defrespo...>ben...>read sta...>readin...>recv[...]>req...> X

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

Comment Share

+ 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	Sai	Feb	Sai	Mar	Sai

3s completed at 5:09 PM

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

✓ RAM: [██████] Disk: [██████] Editing

```

 0
    "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.

	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

 Building your first Gradio Web app 

File Edit View Insert Runtime Tools Help

+ Code + Text RAM: [██████] ✓ Disk: [██████]

 0 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

Clear
Submit

OUTPUT 1 0.02s

Name	Jan Sales	Feb Sales	Mar Sales
Jon	12	100	18

 3s completed at 5:09 PM 

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

Clear Submit

OUTPUT 1

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

0.02s

OUTPUT 2

The chart displays a line graph with three data series: Jon (blue), Alice (orange), and Sana (green). The Y-axis represents sales values ranging from 1500 to 2500. The X-axis represents time periods. Sana's sales show a sharp increase from January to March, starting at 8 and ending at 12. Alice's sales remain relatively stable around 17. Jon's sales fluctuate between 12 and 100.

3s completed at 5:09 PM

 Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

Comment Share ⚙ A

+ Code + Text

RAM: [██████] Disk: [██████] Editing ▲

```
0    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]]
    ),
    [
        "dataframe",
        "plot",
        "numpy"
    ],
    description="Enter sales figures for employees to predict sales trajectory over year."
```

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM: [██████████] Disk: [██████]

Editing

0

```
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", 0, 9.5, 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

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()"

Executing (1s) C_>launch->setup_tf->url_requ->urlooo->op_>op_->call_ch->https_a->do_a->defrespo->be_->read_st_->readi_->recy_L->rea_->X

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM: [██████] Disk: [██████]

Editing

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

EMPLOYEE DATA

	Name	Jan	Sai	Feb	Sai	Mar	Sai
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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help

+ Code + Text

RAM: [██████████] Disk: [██████]

Editing

```

    "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 0.0%

	Name	Jan	Sai	Feb	Sai	Mar	Sai
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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ 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

The graph displays three data series: Jon (blue line), Alice (orange line), and Sana (green line). The Y-axis represents Sales (Sal) ranging from -500 to 2500. The X-axis represents Months from 0 to 8. Sana's sales show a strong positive linear trend, starting at approximately 800 in month 0 and reaching about 2200 in month 8. Alice's sales show a slight downward trend, starting around -400 in month 0 and leveling off near -500 by month 8. Jon's sales remain relatively flat, starting at 12 in month 0 and ending at approximately -100 in month 8.

3s completed at 5:10 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk ✓ Editing ^

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 three data series: Jon (blue line), Alice (orange line), and Sana (green line). The Y-axis represents Sales (0 to 250) and the X-axis represents Months (0 to 8). Sana's sales show a strong positive linear trend from approximately 100 in month 0 to 250 in month 8. Alice's sales show a slight negative trend from approximately -50 in month 0 to -100 in month 8. Jon's sales remain relatively flat, starting at 12 and ending at 18.

Clear Submit

Screenshot Flag

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()
```

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

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

✓ RAM: [] Disk: [] Editing

0 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()

iiface = 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."
)
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

✓ 3s completed at 5:10 PM

Building your first Gradio Web app ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM: [██████████] Disk: [██████]

Editing

0

```

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(1,12), 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()

```