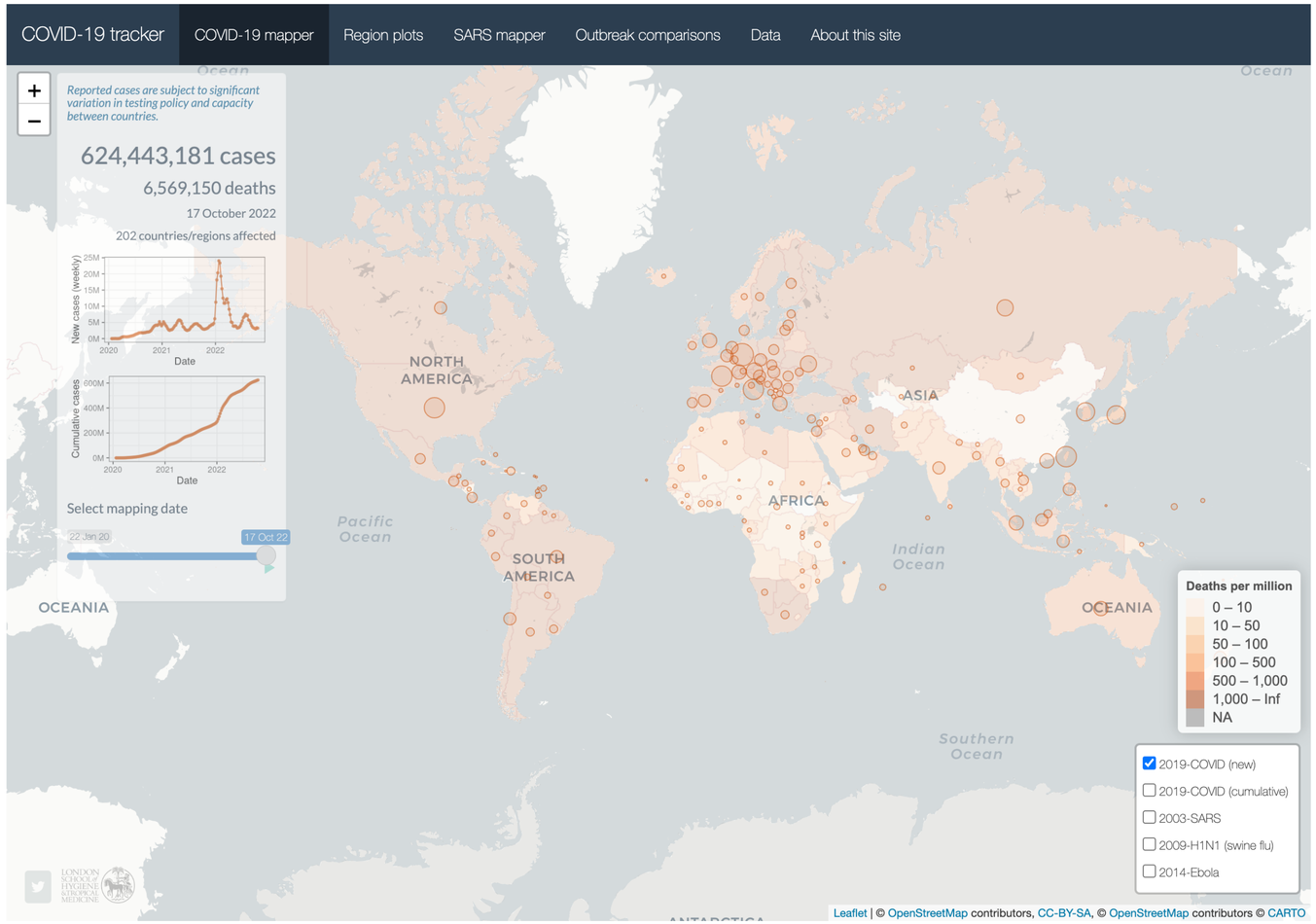
An Overview of Shiny for Python

Carson Sievert
Software Engineer, Shiny team @ Posit
Slides: bit.ly/py-shiny-epa

What is Shiny?

- A framework for building interactive web apps in R (and now Python!)
- Designed for data scientists
  - No HTML/CSS/JS required
  - ‘Ready to use’ components (e.g., dropdown, slider, etc)
  - Integration with pandas, matplotlib, Jupyter Widgets, etc
- ‘Magic’ decomposes into sound engineering principles
  - Reactive programming model scales efficiently & intuitively
  - UI can be “progressively enhanced” (i.e., sprinkle small amounts of HTML/CSS/JS, create custom components w/o complex frameworks, etc)

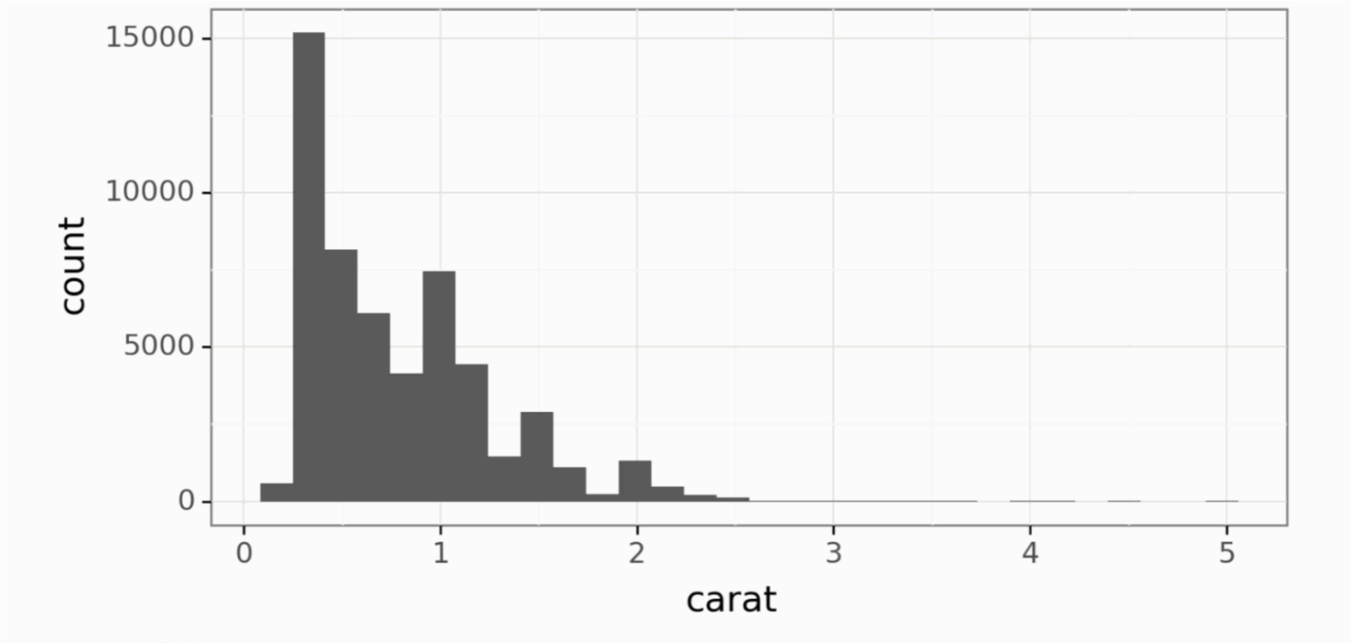


An Intro to Shiny for Python*, via R

* Currently in alpha

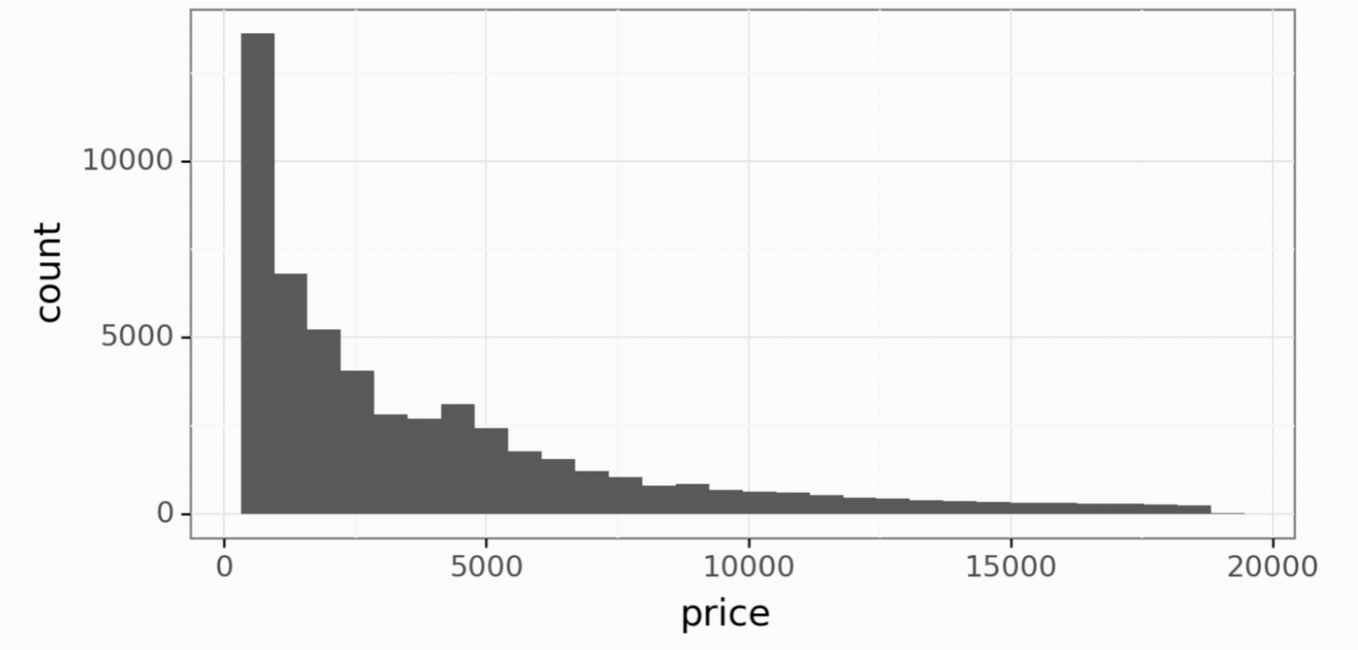
First, some R code

How about diamond price?



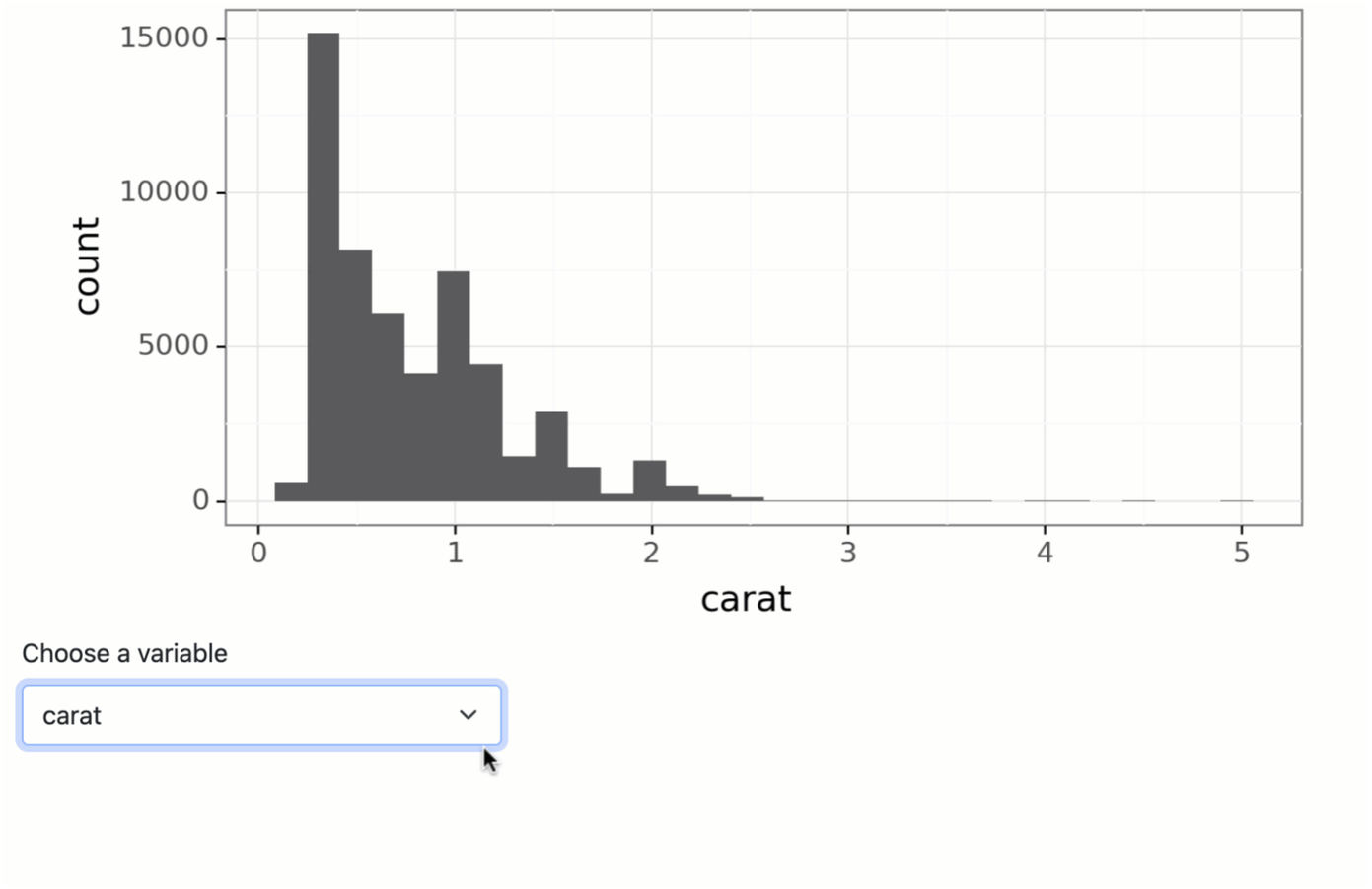

```
library(ggplot2)

x ← diamonds[["carat"]]

ggplot() +
  geom_histogram(aes(x = x)) +
  theme_bw(base_size = 16)
```

First, some R code

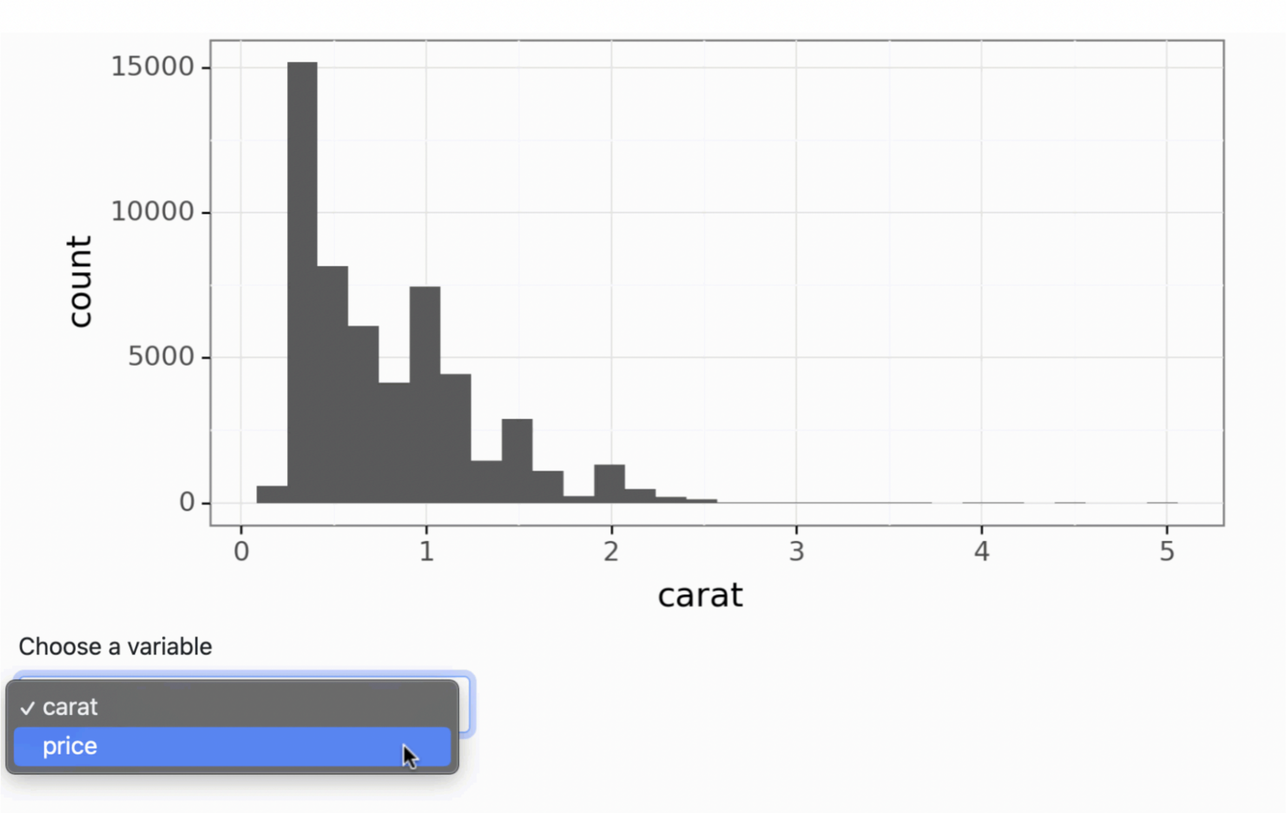



```
library(ggplot2)

x ← diamonds[[“price"]]

ggplot() +
  geom_histogram(aes(x = x)) +
  theme_bw(base_size = 16)
```

Wouldn’t it be nice?



Shiny for R




```r
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotOutput(“p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← renderPlot({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

Shiny for R

```
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotOutput(“p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← renderPlot({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

User interface of inputs and outputs

Shiny for R

```
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotOutput("p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← renderPlot({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

Logic for generating output from input

Shiny for R

```
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotOutput("p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← renderPlot({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

Reactively read input value

Shiny for R

```
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotOutput("p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← renderPlot({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

And render as a plot

Shiny for R

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  plotOutput("p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server <- function(input, output) {
  output$p <- renderPlot({
    x <- diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

Located above the dropdown

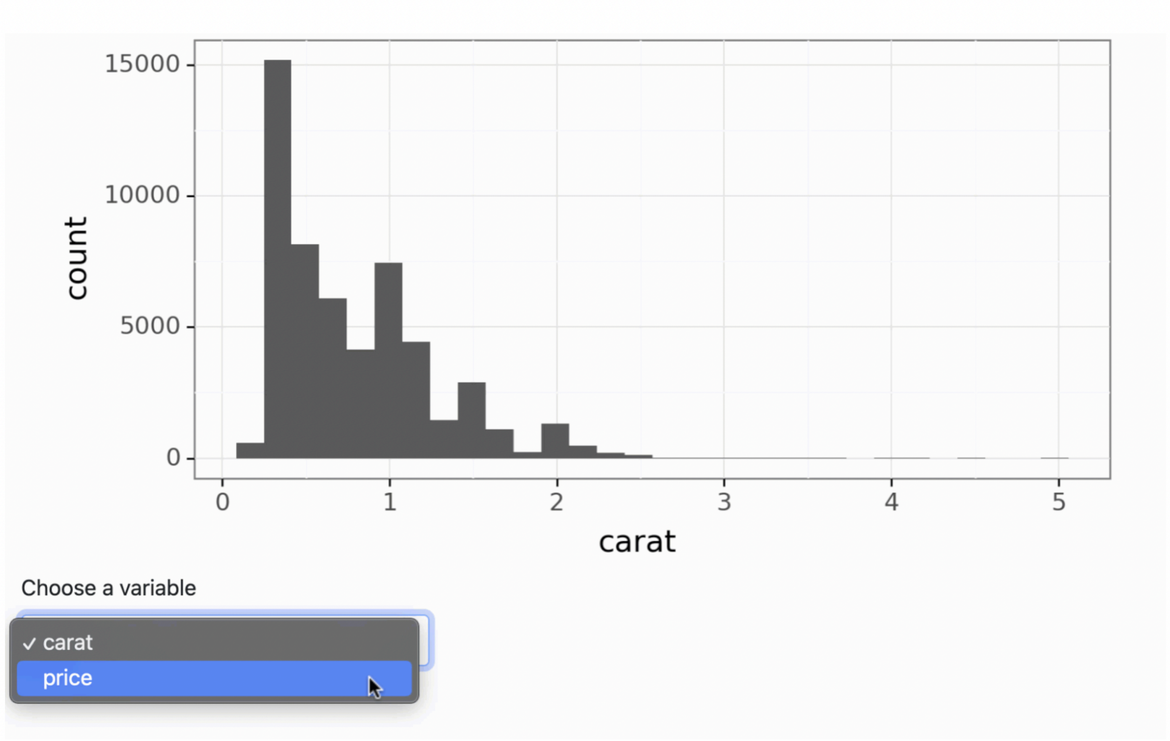
Shiny for R

```
library(shiny)
library(ggplot2)

ui ← fluidPage(
  plotly::plotlyOutput("p"),
  selectInput(
    "var", "Choose a variable",
    choices = c("carat", "price")
  )
)

server ← function(input, output) {
  output$p ← plotly::renderPlotly({
    x ← diamonds[[input$var]]
    ggplot() +
      geom_histogram(aes(x = x)) +
      theme_bw(base_size = 16)
  })
}

shinyApp(ui, server)
```

Many (3rd party) outputs available

That’s enough R...time for Python!

Shiny for Python




```python
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

User interface of inputs and outputs

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

Logic for generating output from input

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

Reactively read input value

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

And render as a plot

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

Located above the dropdown

Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

What about other inputs?

Other input components

input_select()

Select an item

Choice A

input_numeric()

Enter a number

8

input_text()

Enter text

Hello, world!

input_checkbox_group()

Checkbox group

- [x] Choice A
- [ ] Choice B

input_radio_buttons()

Radio buttons

- (•) Choice A
- ( ) Choice B

input_date()

Date input

2022-10-18

« October 2022 »

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

input_switch()

Freeze output

input_slider()

Slider

0 10 20

0 2 4 6 8 10 12 14 16 18 20

Other UI stuff

input_action_button()

Dismiss

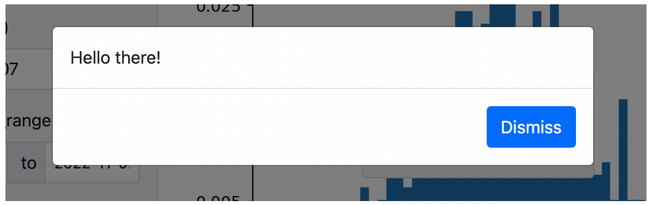
download_button()

Download

input_file()

Browse... No file selected

modal_show()

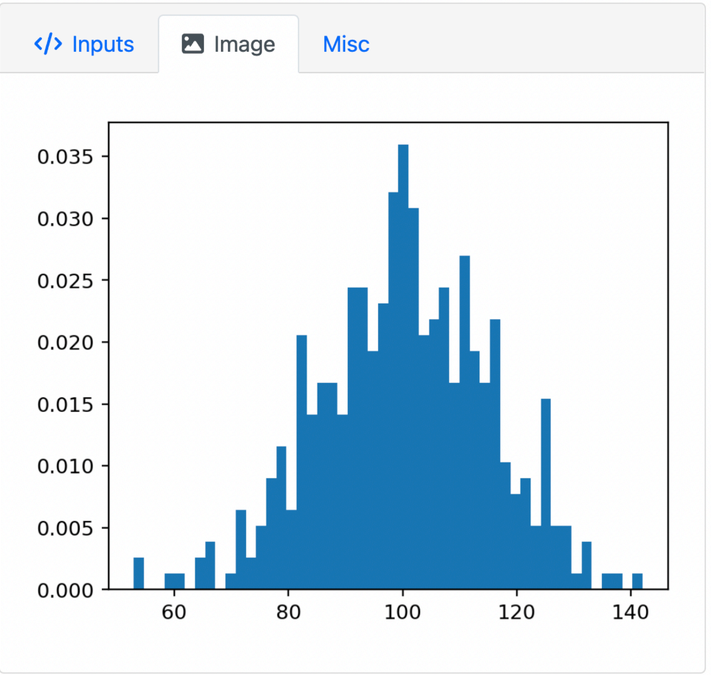


notification_show()

A notification!

Computing

navset_tab_card()



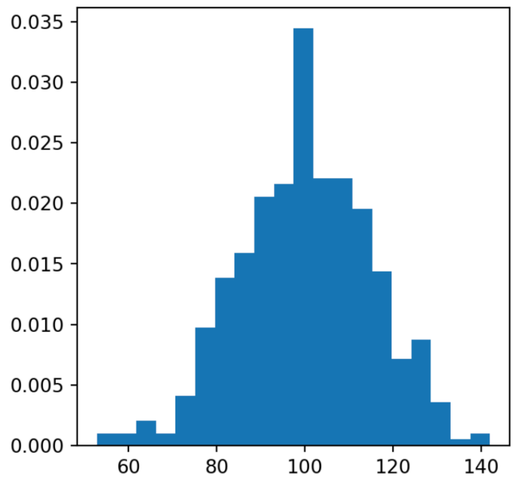
Shiny for Python

```
from shiny import ui, App, render
from plotnine import ggplot, geom_histogram, aes, theme_bw
import pandas as pd

diamonds = pd.read_csv("diamonds.csv")

app_ui = ui.page_fluid(
  ui.output_plot("p"),
  ui.input_select(
    "var", "Choose a variable", [“carat", “price"]
  )
)

def server(input, output, session):
    @output
    @render.plot
    def p():
        return (
            ggplot(diamonds)
            + geom_histogram(aes(x=input.var()), bins=30)
            + theme_bw(base_size=16)
        )

app = App(app_ui, server)
```

What about other outputs?

Other output components

`output_plot()`



`output_table()`

species	island	bill_length_mm	bill_depth_mm
Adelie	Torgersen	39.1	18.7
Adelie	Torgersen	39.5	17.4
Adelie	Torgersen	40.3	18.0
Adelie	Torgersen	nan	nan
Adelie	Torgersen	36.7	19.3
Adelie	Torgersen	39.3	20.6
Adelie	Torgersen	38.9	17.8
Adelie	Torgersen	39.2	19.6
Adelie	Torgersen	34.1	18.1
Adelie	Torgersen	42.0	20.2
Adelie	Torgersen	37.8	17.1
Adelie	Torgersen	37.8	17.3

`output_text()`

Hello, world!

`output_text_verbatim()`

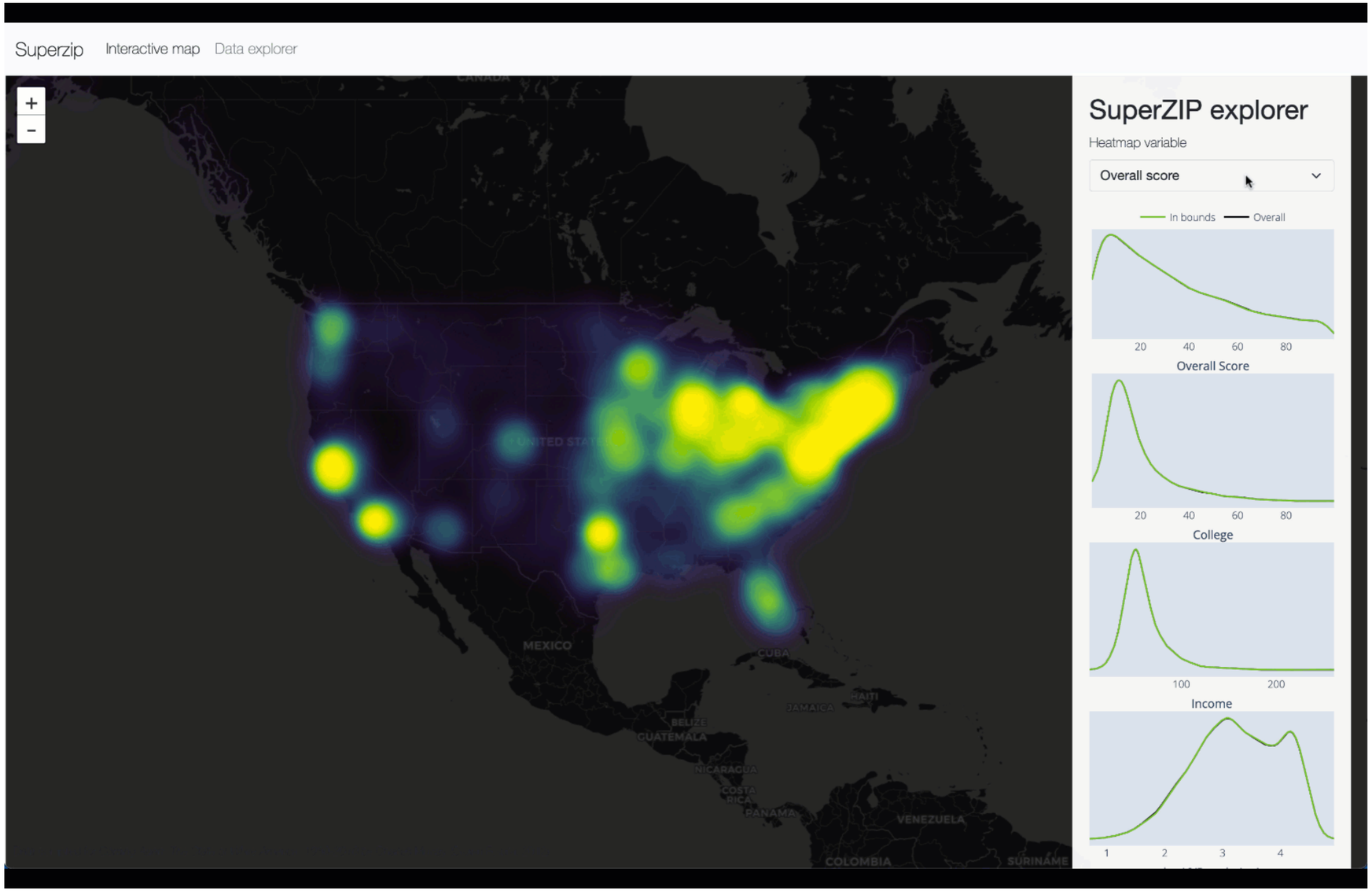
```
Hello, world!
```

`output_ui()`

</>

shinywidgets: Jupyter widgets in Shiny

Jupyter widgets (e.g., plotly, altair, bokeh, ipyleaflet, etc.) that implement ipywidgets protocol are usable in Shiny via the shinywidgets package.



Other reactive building blocks

Python	R	Purpose
`@reactive.Calc()`	`reactive()`	Reactive calculations
`@reactive.Effect()`	`observe()`	Reactive side-effects
`@reactive.event()`	`bind_event()`	Reactive events (e.g., trigger an action on click)
`reactive.isolate()`	`isolate()`	Don’t take a reactive dependency

For more, see: shiny.rstudio.com/py/docs/reactive-programming.html
mastering-shiny.org/reactivity-intro.html

Deployment platforms

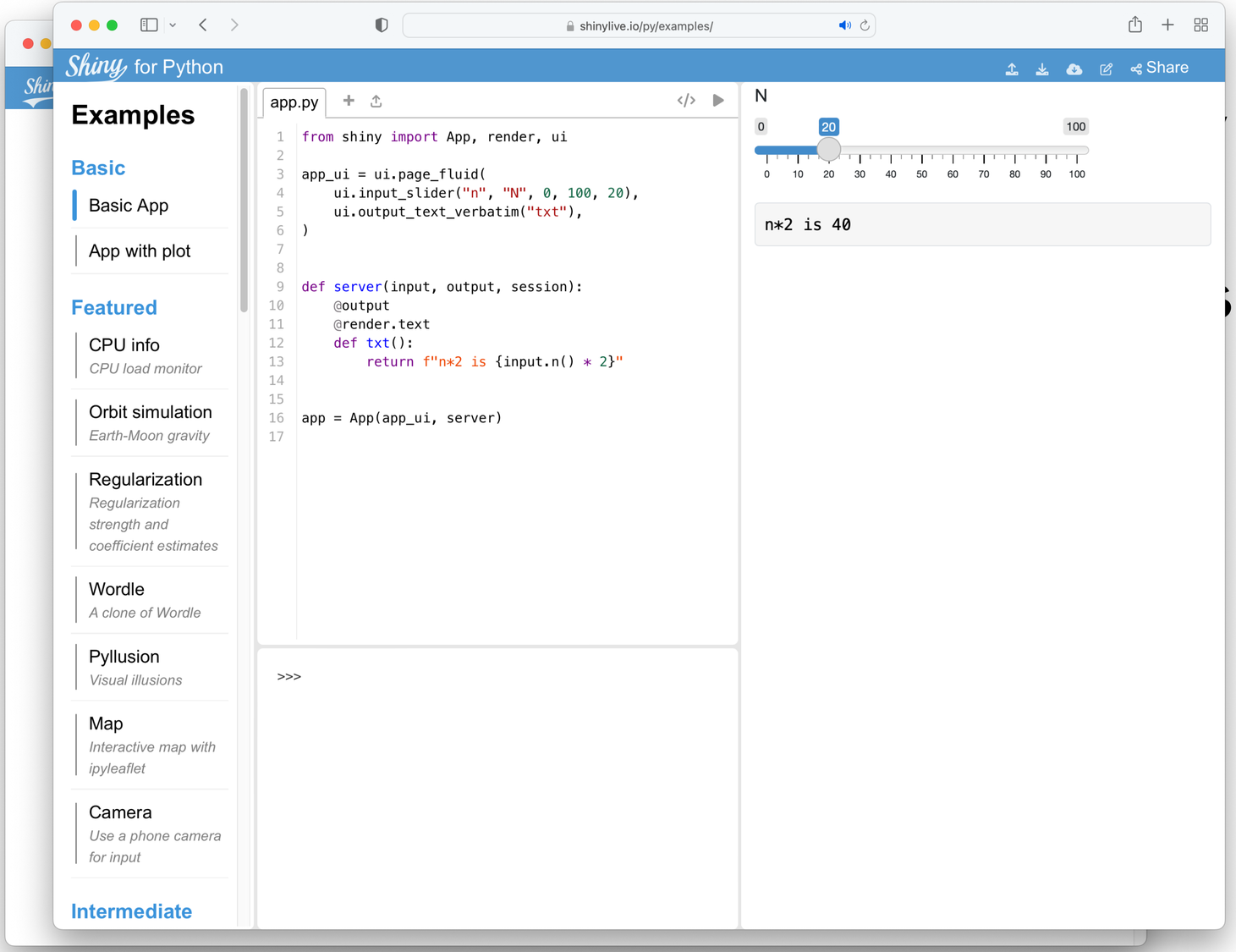
- **shinyapps.io** (managed cloud hosting platform)
- **Posit (aka RStudio) Connect**
- **Shiny Server** (open source)
- *In theory*, any hosting platform that supports FastAPI applications, like Heroku
  - *In practice*, we couldn't get Heroku to work reliably due to issues with sticky sessions. If you figure it out, please let us know!

For more, see: shiny.rstudio.com/py/docs/deploy.html

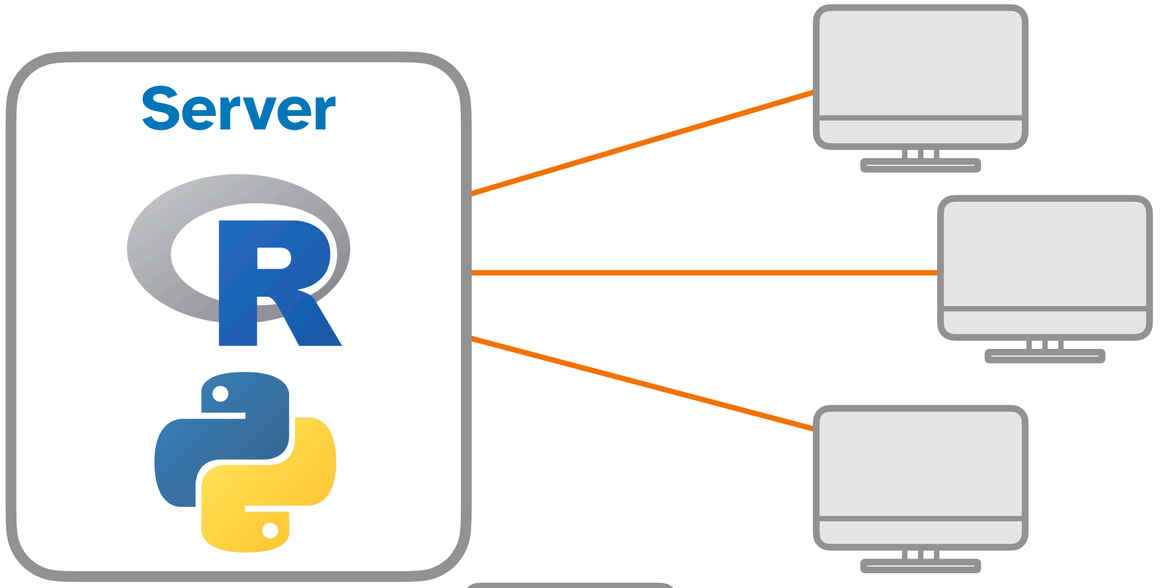
Shinylive: Shiny without a server

Shinylive: Shiny running in the browser

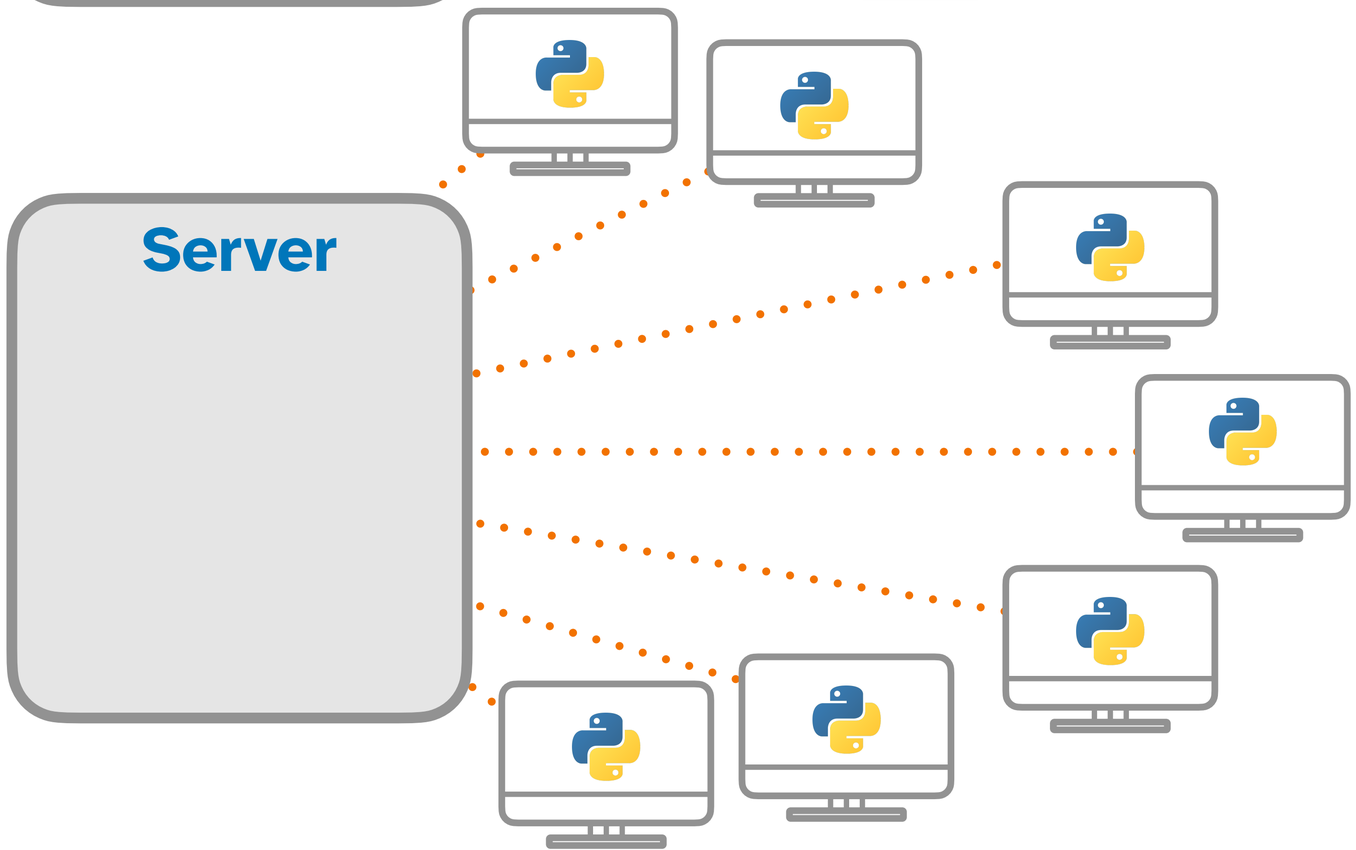
- **WebAssembly** (wasm): binary format for programs that run in the web browser
- **Pyodide**: Python compiled to WebAssembly
- **Shinylive**: Shiny running on Pyodide, in the web browser
- Can be deployed to static web hosting services like GitHub Pages, Netlify, Amazon S3



Traditional Shiny app deployment



Shinylive app deployment



Drawbacks and Limitations

- Shinylive is still experimental!
- Not all packages are available in Pyodide.
- Code and data are sent to the web browser: No secrets!
- Not good for large amounts of data.
- Download size: for a very basic application, browser will download about 13 MB.
    - Browser caching speeds up subsequent visits.

If Shinylive doesn't work for your application, you always have the option to do a traditional Shiny app deployment.

Thank you!

Shiny for Python

shiny.rstudio.com/py

Jupyter Widgets in Shiny

github.com/rstudio/py-shinywidgets

shinydashboard

github.com/jcheng5/py-shinydashboard