Dashboarding 101

FinTech

Lesson 6.3

Class Objectives

By the end of today's class, you will be able to:

- Explain what Panel is and its value in relation to data visualization.
- Explain Panel's role within the PyViz ecosystem.
- Design and develop a Panel dashboard layout using row, column, and tab objects.
- Understand how Panel extensions are used to enable interactivity and plugin integration.
- Integrate Plotly and hvPlot plots with Panel dashboard using Panel extensions.
- Explain the role of Bokeh in Panel dashboards.
- Deploy Bokeh server to host Panel dashboard.

Creating a Dashboard Is Different than Creating a Report

01

Reports are used to help visualize underlying data and enable users to perform their own analysis of the data.

02

Dashboards are used to consume data tailored to a specific reporting need. The reporting need will typically satisfy a business objective or evaluate performance based off of certain indicators.

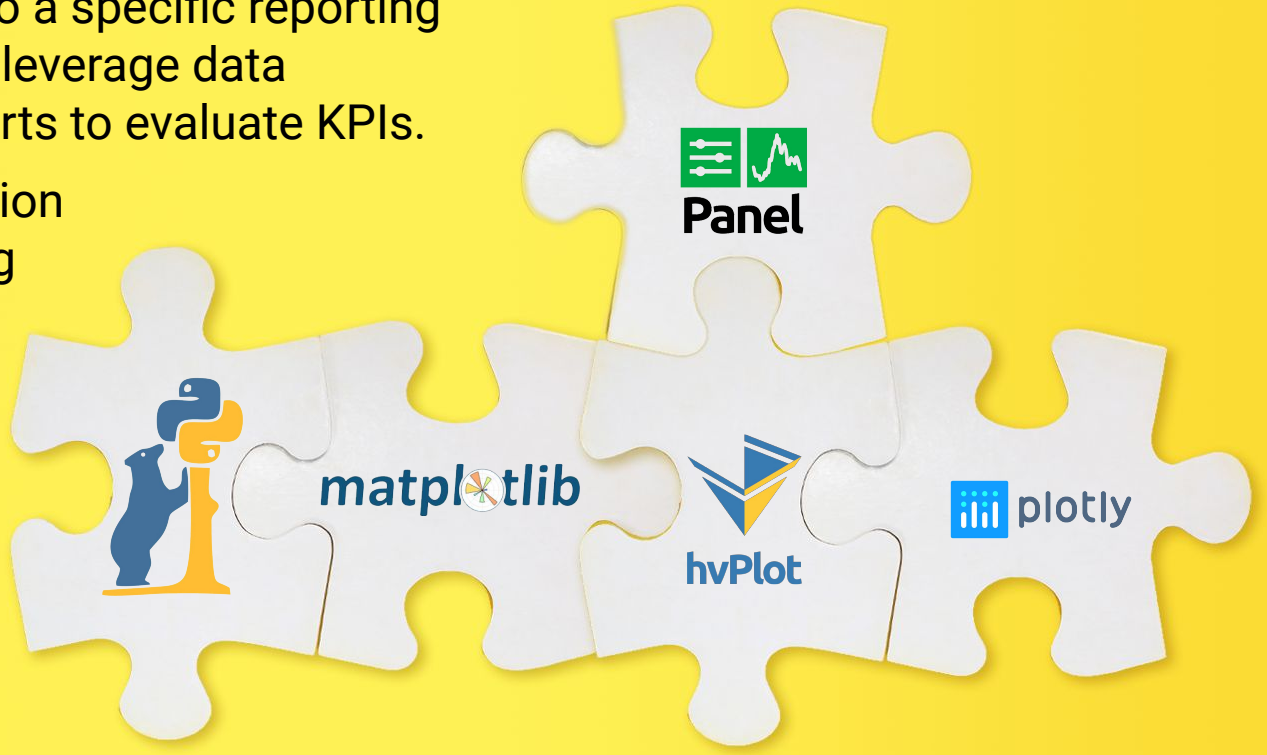
03

Dashboards can be used to consolidate multiple reports into one view. Structuring reports as parts of dashboards helps to create a robust reporting platform based off of a digital dashboard.

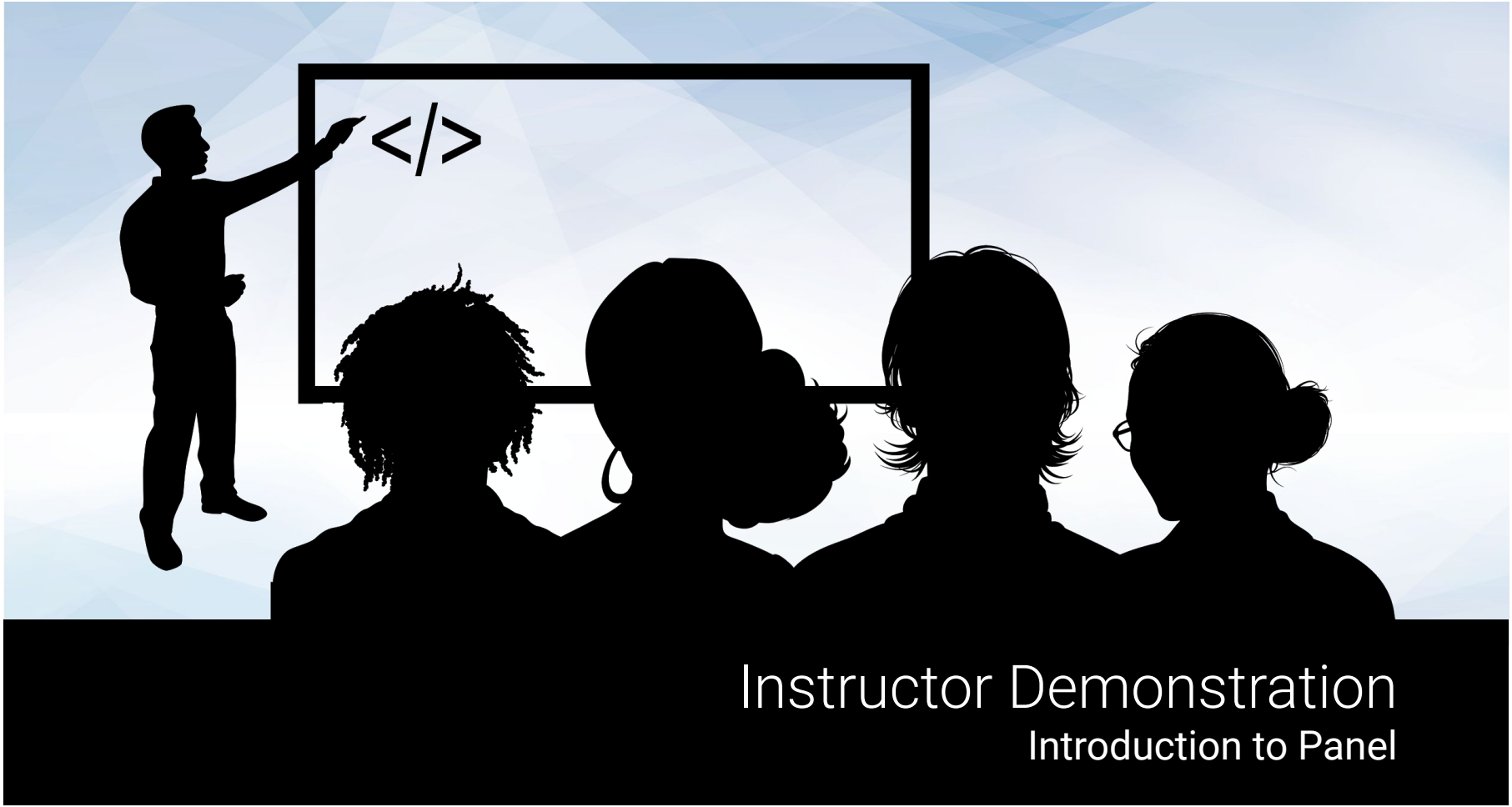
Panel: a PyViz Dashboard Package

Panel offers a one-stop shop for data visualization (curated to a specific reporting objective) where users leverage data visualizations and reports to evaluate KPIs.

Panel supports integration with each of the plotting libraries you've learned so far.



Panel



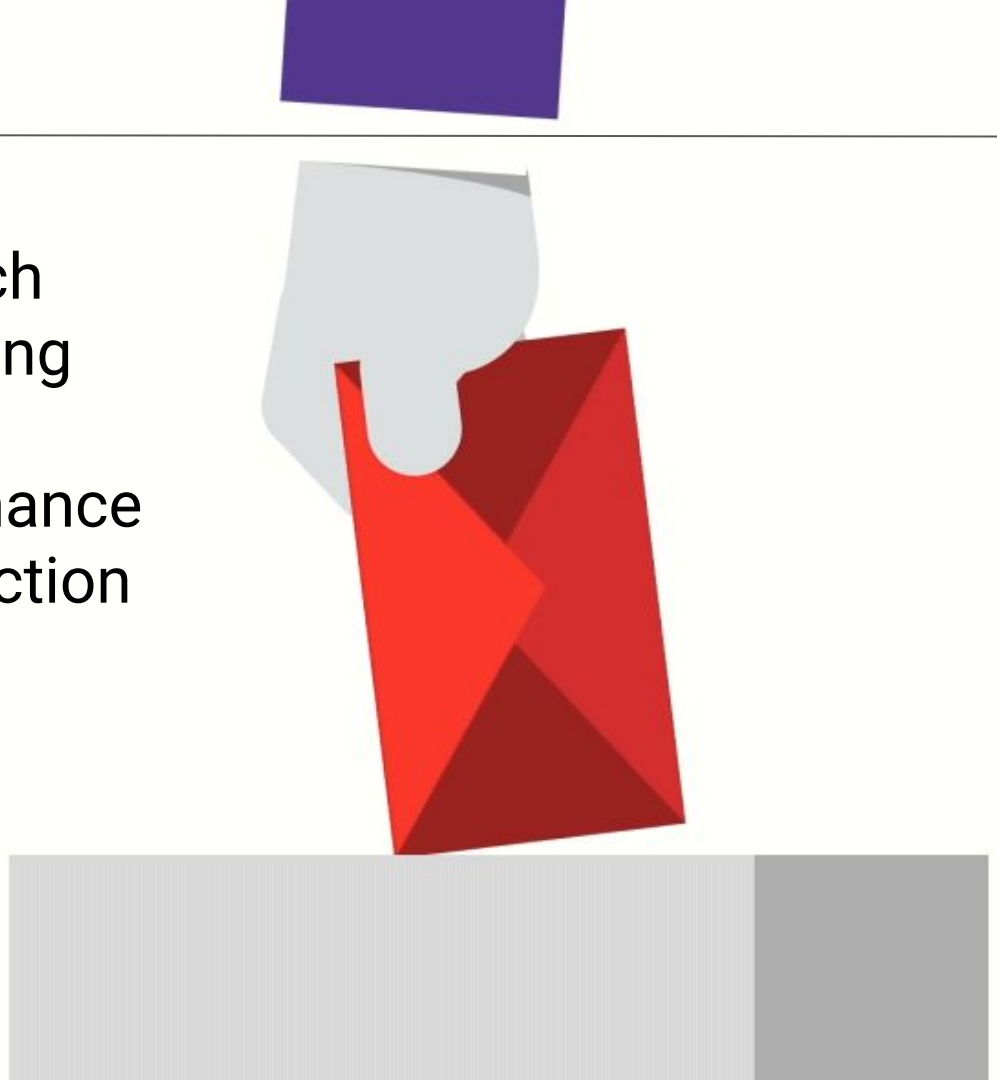
Instructor Demonstration

Introduction to Panel

Panel’s integration with PyViz has made it a highly valuable skill in analytics, as well as a staple in the Python data visualization ecosystem. Dashboards are currently being used to drive business decisions, track and monitor KPIs, and improve operational processes.

Panel Dashboarding

Digital dashboards have become a staple in the tech world. Dashboards are being used to visualize personal finance, company performance and revenue, and even election results/predictions.



Panel Dashboarding

01

Panel, a PyViz dashboarding package, has altered the data visualization arena by providing a programmatic way to easily create dashboards.

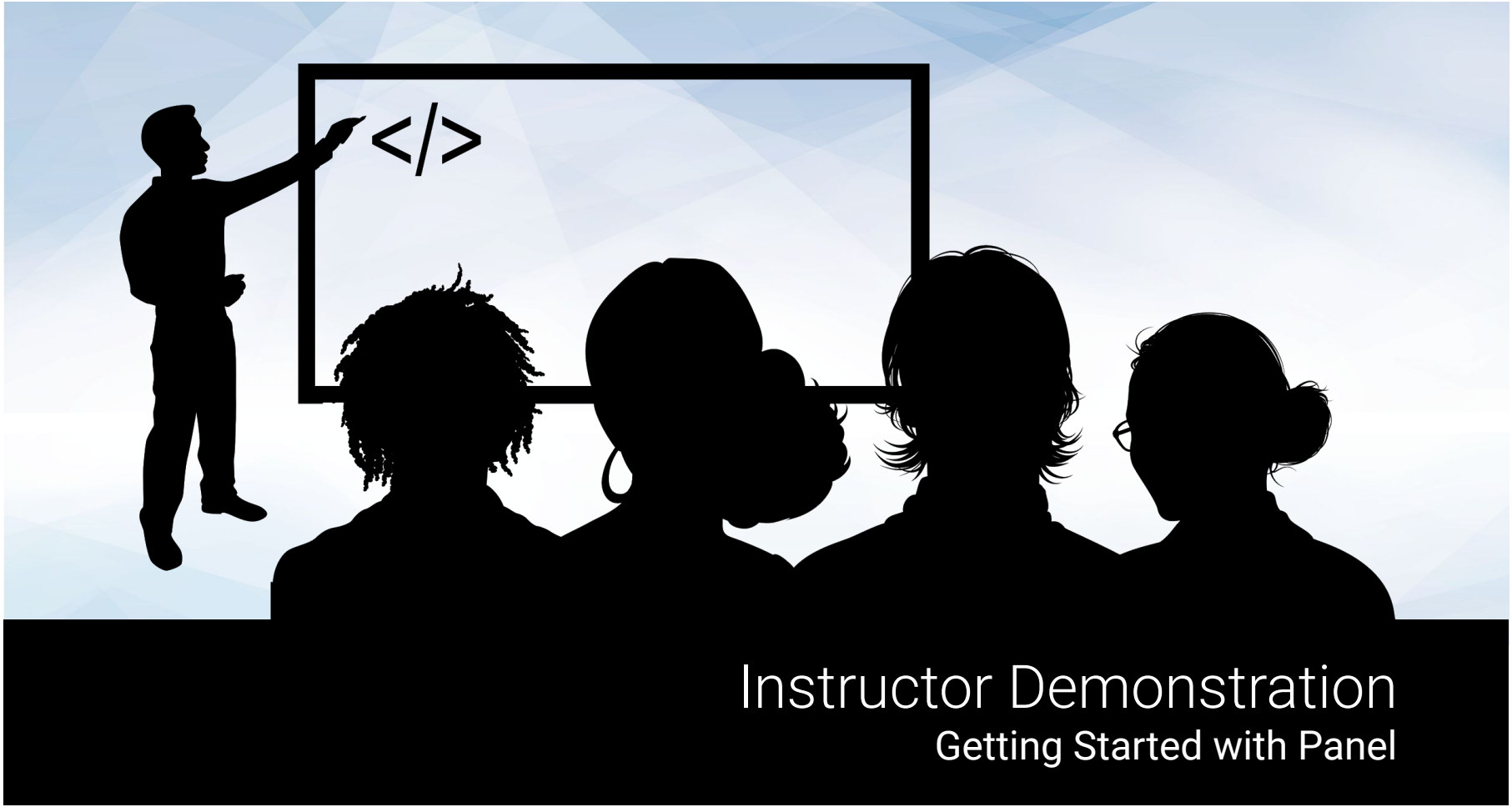
02

Panel's integration with PyViz has made it a highly valuable skill in analytics, as well as a staple in the Python data visualization ecosystem.

03

Panel allows data scientists and analysts to create and manage dashboards in the same environment they are completing development, using one technology stack. One developer can use the same technology to create both a data pipeline and dashboard visualizations for the data.

Getting Started with Panel



Instructor Demonstration

Getting Started with Panel

Getting Started with Panel

Panel can be imported into Python using the import command.

```
import panel as pn
```

Panel comes equipped with its own set of libraries, functions, and widgets. An example library is the `panel.interact` library.

```
from panel.interact import interact
from panel import widgets
```

When working with Panel in Jupyter Lab, the Jupyter Lab Panel plugin is required. The plugin can be activated using the `extension` command. If the extension is not activated, Panel objects will not render in Jupyter Lab.

```
pn.extension()
```

Getting Started with Panel

A commonly used Panel function is `panel.interact`. `interact` allows users to create widget UI controls to manipulate function parameters.

These parameter widgets provide a convenient and easy way for users to change the values populating the visual, which is extremely handy when completing impact analysis.

The Panel parameter widget bar will only appear if a function was passed that accepts parameters. If a function without parameters is passed to the interact function, the interact function will execute, but it will not provide the widget bar.

It's important to note that this parameter widget is an added benefit provided by Panel. Not all dashboarding technologies provide this convenient feature.

Panel with Python Data Structures

```
# Define function to choose a year
def choose_year(year):
    return year
```

```
# Create interactive Panel drop down list using Python List
list_of_years = ['2019','2018','2017','2016','2015']
interact(choose_year, year=list_of_years)
```

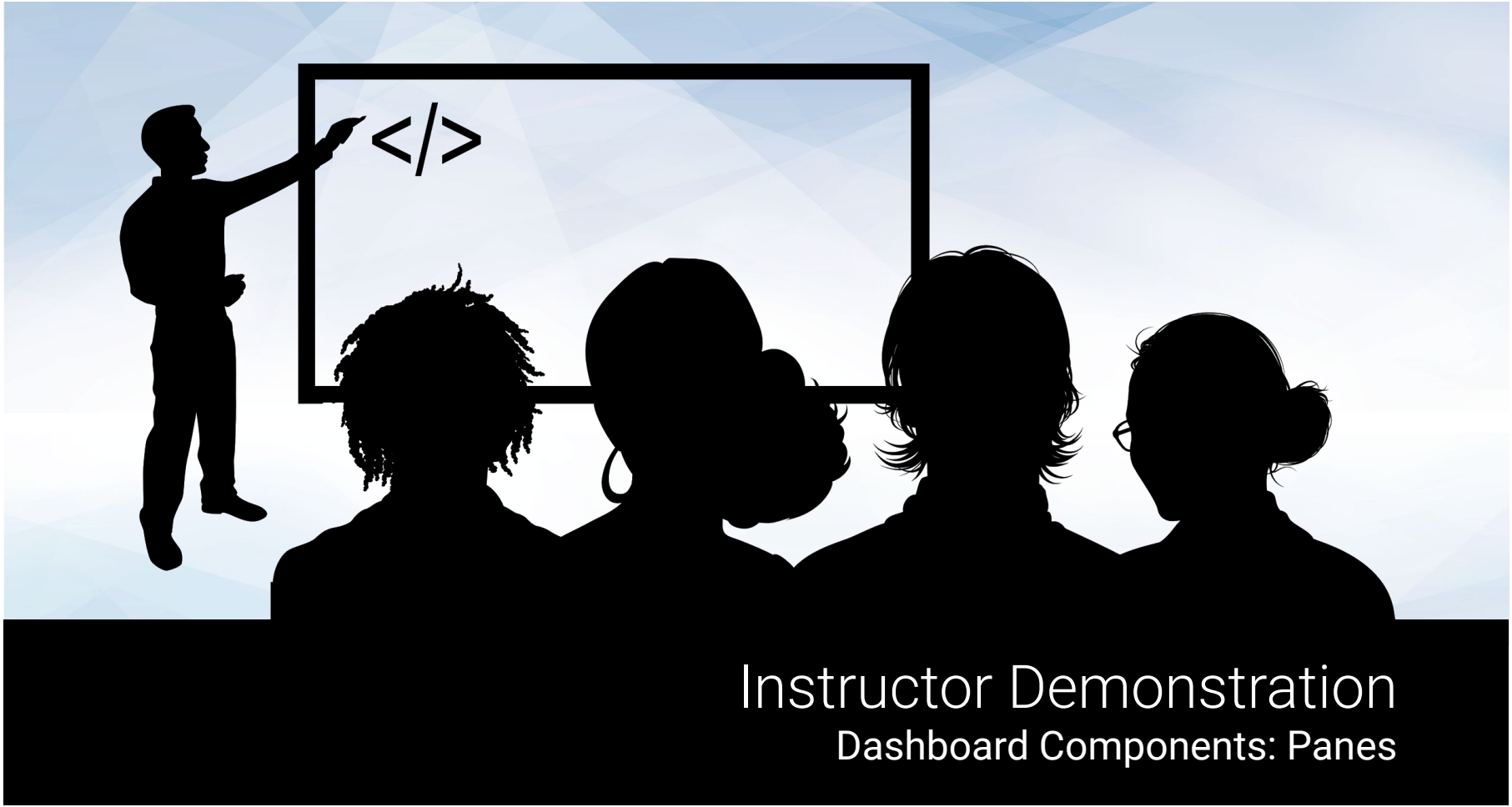

```
# Define function to choose a year
def choose_year(year):
    return year
```

```
# Create interactive Panel drop down list using Python List
list_of_years = ['2019','2018','2017','2016','2015']
interact(choose_year, year=list_of_years)
```



```
# Create interactive Panel drop down list using Python List
list_of_years_2 = ['2014','2013','2012','2011','2010']
interact(choose_year, year=list_of_years_2)
```

Dashboard Components



Instructor Demonstration

Dashboard Components: Panes

Dashboard Components (Panes)

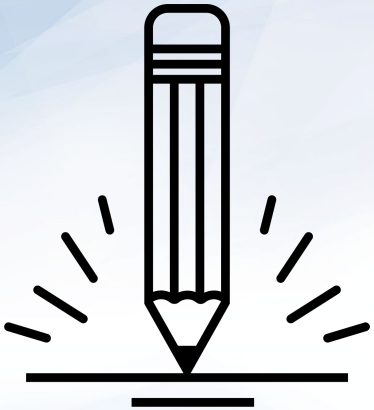
Dashboard components are used to create the overall layout and structure of a Panel dashboard. Dashboard components are what allow media to be rendered on a Panel dashboard.

Panel dashboards have two main components: panes and panels (rows, columns, and tabs).

Pane and panel objects work in same way, in the sense that they wrap around pre-existing objects (plots, text, html). Each specific media type will have a helper function that can be used to convert the content to a Panel object.

In order to display and interact with content on a Panel dashboard, the content must be converted to a Panel object. Once content is in a Panel pane object, the rendered content (i.e., plot, image, html, markdown, etc.) can be accessed and called with Panel.

Panel panes and panels are similar objects except panels are sub-components (rows, columns, and tabs).



Activity:

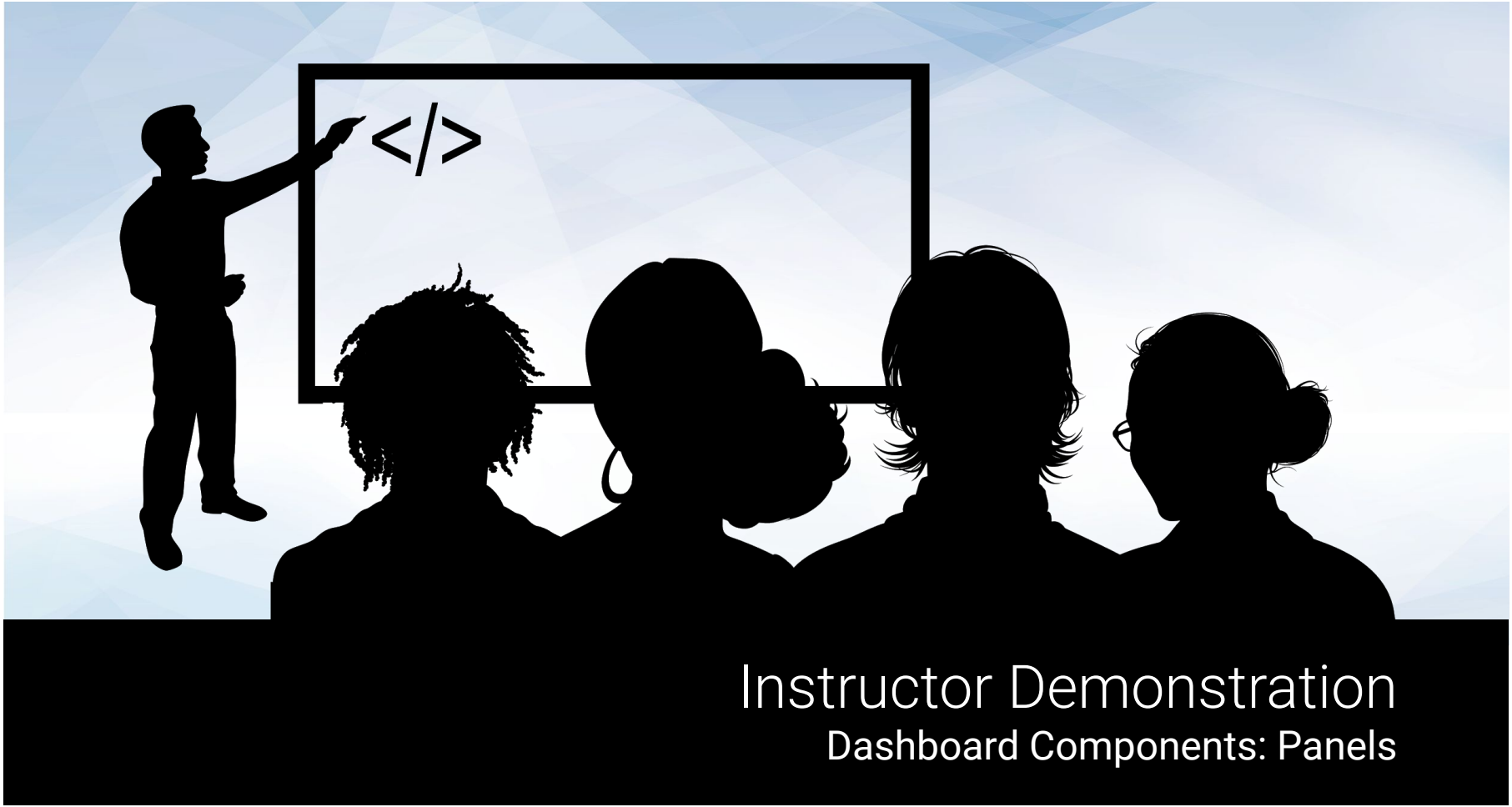
No Pane, No Gain

In this activity, you will create a Plotly plot and convert it to a Panel pane.

(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Time’s Up! Let’s Review.



Instructor Demonstration

Dashboard Components: Panels

Dashboard Components: Panels

Panel panels are similar to pane objects, except panels have a structure.

Panels can consist of rows, columns, and/or tabs. Rows, columns, and tabs all serve as storage mechanisms for media types. For example, a Plotly figure stored as a pane can be inserted into a row.

Rows, columns, and tabs are all list objects and operate similarly to Python lists. This means list functions (i.e., add, remove, etc.) can be used to add, remove, and update content.

Rows can be created to list components/media in a horizontal container. Row objects can contain different media types, such as Markdown language or different types of plots.

Dashboard Components: Panels

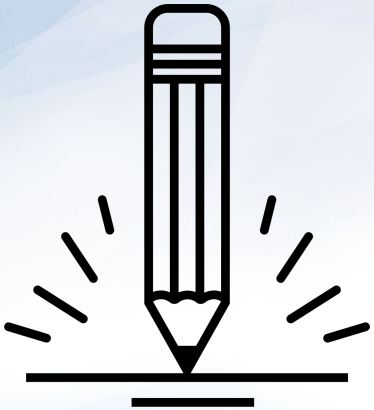
Panel panels are similar to pane objects, except panels have a structure.

Columns are similar to rows, except content is aligned in a vertical container. The `panel.Column` function can be used to align content in a vertical fashion. For example, the row object previously created could be added to a column that also includes Markdown headers.

```
# Create column using Markdown and row object
column = pn.Column(
    '# Alleghany, PA Real Estate Visualizations',
    '## Sales and Foreclosures',
    row)
column
```

Tabs organize components as selectable tabs. Tabs are an alternative to putting content in a horizontal or vertical container. Tabs allow one dashboard to have multiple reports/views into the data. Each tab could represent a different data set, plot, and/or analytic objective. Tabs are created using the `panel.Tabs` function.

```
# Create tabs
tabs = pn.Tabs(
    ("Correlations", scatter_plot),
    ("Time Series", bar_plot))
tabs
```



Activity: The Judge's Panel

In this activity, you will execute insert, update, and remove operations on Panel panels.

(Instructions sent via Slack.)

Suggested Time:
20 Minutes

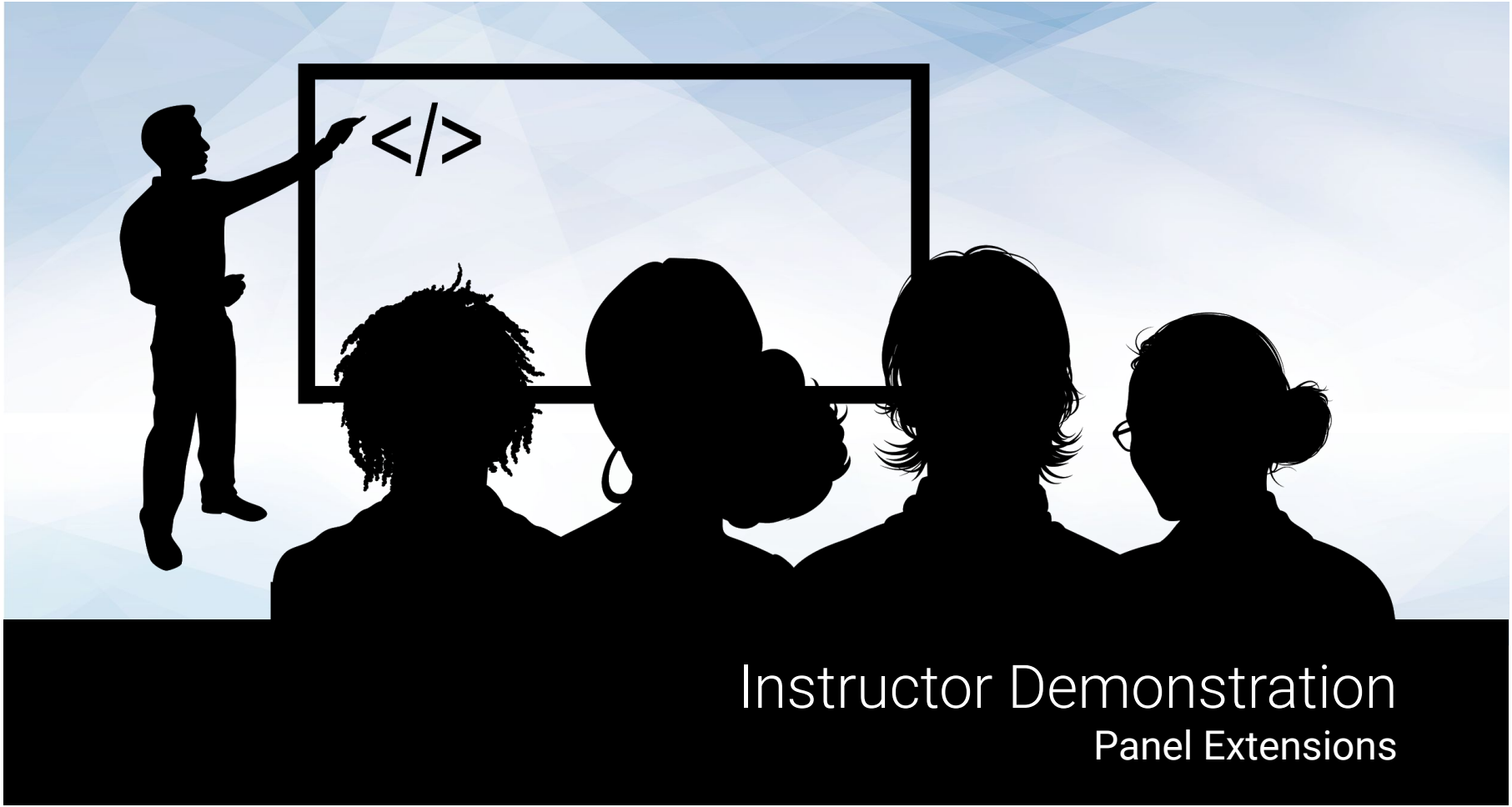
Time’s Up! Let’s Review.

Countdown timer

40:00

(with alarm)

Panel Extensions



Instructor Demonstration

Panel Extensions

Panel Extensions

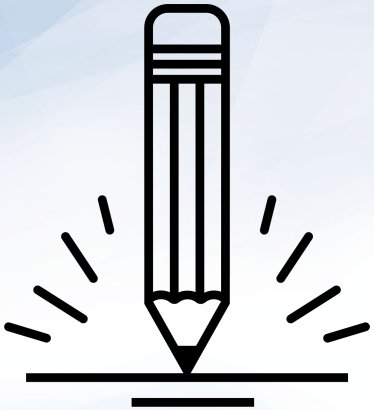
Panel supports a wide range of visualization technologies. Support for these technologies are managed by Panel extensions.

Panel supports a number of extensions, including Plotly, Bokeh, and Matplotlib. Extensions give Panel the ability to display and use content created by other technologies.

Each Panel extension has its own unique features and color schemes. There are also features that are shared across extensions. For this reason, multiple extensions may need to be specified if a dashboard leverages more than one technology (i.e., Plotly and Matplotlib).

By specifying the extension, multiple media types can be combined to create an informative and insightful dashboard.

Panel extensions are specified using the extension function. The extension function will load in the corresponding plugin, so that content produced by the technology can be rendered and used by Panel.



Activity: Extending Plotting

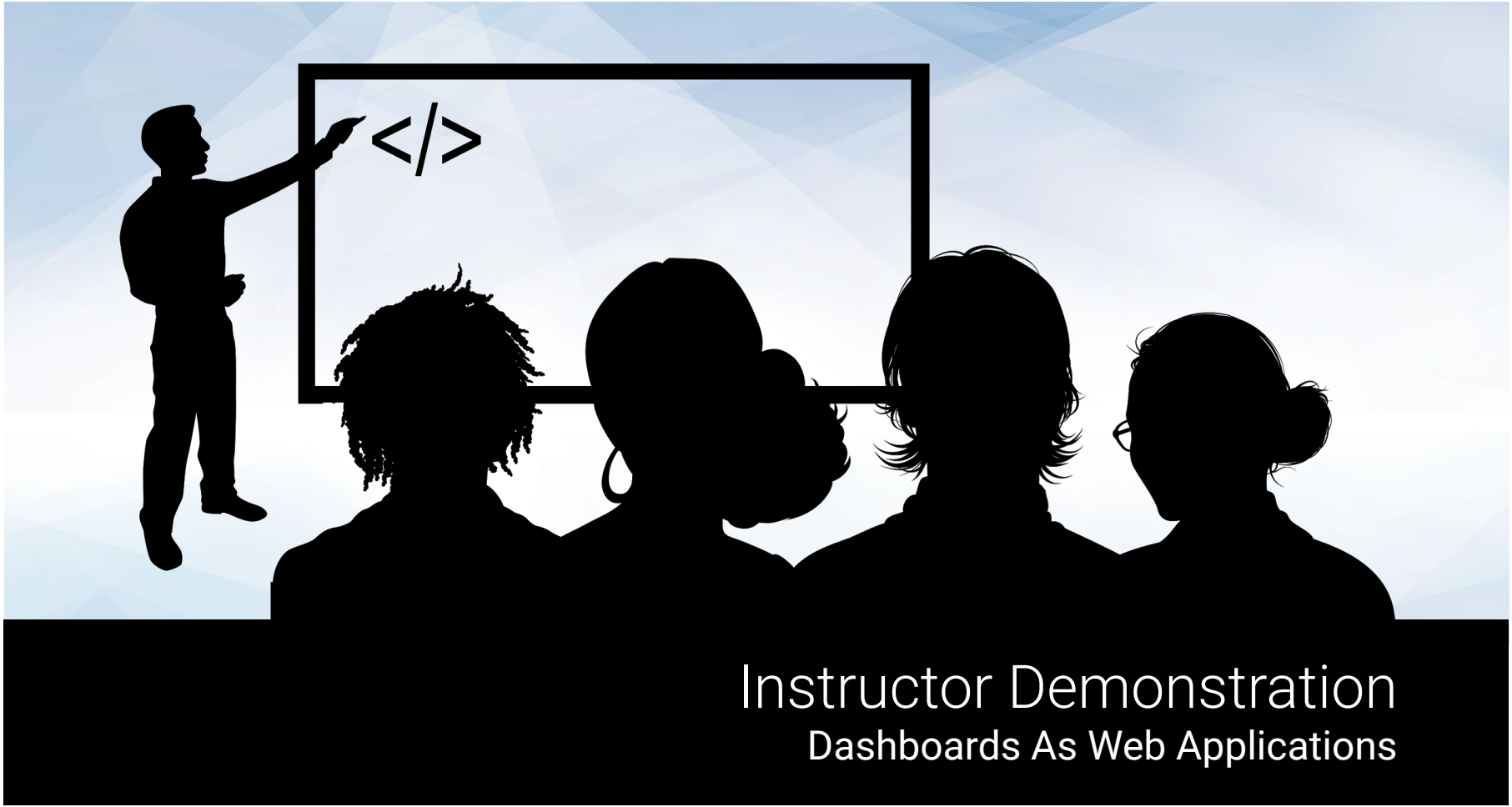
In this activity, you will create a dashboard that contains map visualizations and hvPlot composed plots.

(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Time’s Up! Let’s Review.

Dashboards As Web Applications



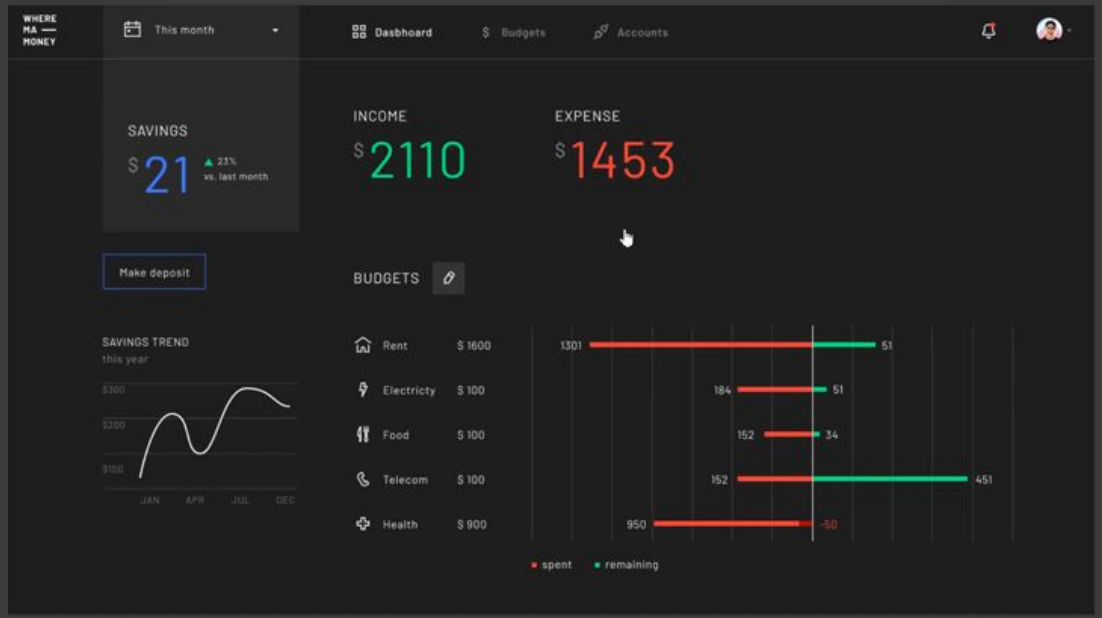
Instructor Demonstration

Dashboards As Web Applications

Dashboards As Web Applications

One of the advantages of using Panel for dashboarding is that Panel dashboards can be deployed as web apps: dashboard applications that are accessible over the internet rather than just in a Jupyter notebook. This means that any Jupyter notebook containing a Panel object can be deployed as its own independent web app.

Panel is equipped with a servable function that spins up a Bokeh server and launches the Jupyter notebook code as a web app. Even though the web app will use the code from the Jupyter notebook, the web app will only display the Panel object being rendered.



Activity:

Monte Carlo Dashboard Web App

In this activity, you will revisit simulations and use Monte Carlo to predict housing sales prices over the next 10 years.

(Instructions sent via Slack.)

Suggested Time:
15 Minutes

Time’s Up! Let’s Review.

Questions