**Data Visualization with Python**

**Week 4 – Create Web-based application containing Dashboards**

**Plotly | Dash:**

The [plotly Python library](https://plotly.com/python/) is an interactive, [open-source](https://plotly.com/python/is-plotly-free) plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

Built on top of the Plotly JavaScript library ([plotly.js](https://plotly.com/javascript/)), plotly enables Python users to create beautiful interactive web-based visualizations that can be displayed in Jupyter notebooks, saved to standalone HTML files, or served as part of pure Python-built web applications using Dash.

Dash is a Python framework for building web analytic applications. It runs on top of flask plotly.js and react.js.

Dash is well suited for building data visualization apps with highly customized user interfaces.

**React.JS**

React is a free and open-source front-end JavaScript library for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies. React can be used to develop single-page, mobile, or server-rendered applications with frameworks like Next.js.

**Panel:**

Panel works with visualizations from Bokeh, Matplotlib, HoloViews, and many other Python plotting libraries, making them instantly viewable, either individually or when combined with interactive widgets that control them.

Panel works equally well in Jupyter Notebooks for creating quick data exploration tools or as a standalone deployed app in dashboards and allows you to easily switch between those contexts as needed. Voila turns Jupyter notebooks into standalone web applications.

**Voila:**

Voila turns Jupyter notebooks into standalone web applications. It's compatible with separate layout tools like Jupyter-flex or templates like voila-vuetify.

**Streamlit:**

Streamlit can easily turn data scripts into shareable web apps with three main principles:

* embrace Python scripting,
* treat widgets as variables,
* and reuse data and computation.

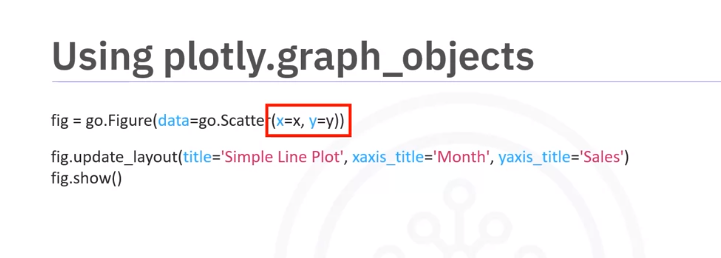
**Bokeh:**

Bokeh is a plotting library, widget, and app library. It acts as a server for both plots and dashboards. Panel, which is one of the web-based dashboarding tools, is built on Bokeh.

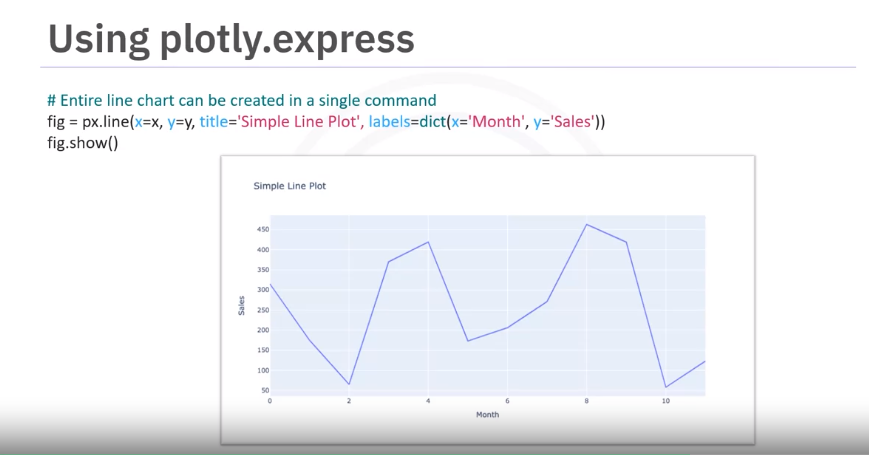
**Ipywidgets** provides a wide array of Jupyter compatible widgets and an interface supported by many Python libraries.

**Plotly Sub-Modules:**

* **Plotly Graph Objects** 
  + Plotly Graph Objects is the low-level interface to figures, traces and layout.
  + The Plotly Graph Objects module provides an **automatically generated hierarchy** of classes, figures, traces, and layout called graph objects that are used for representing figures with a top-level class Plotly.graph\_objects.Figure.



* **Plotly Express** 
  + Plotly Express is a high level wrapper for Plotly. It's a recommended starting point for creating the most common figures provided by Plotly. Because of it's simple syntax, i**t uses graph objects internally.**



**Plotly.express syntax:**

*px.<graphtype>(x=<xaxis value source>,y=<y-axis value source>,title=<appropriate title as a string>)*

The numpy random seed is a numerical value that generates a new set or repeats pseudo-random numbers. The value in the numpy random seed saves the state of randomness. If we call the seed function using value 1 multiple times, the computer displays the same random numbers.

With the seed reset (every time), the same set of numbers will appear every time.

If the random seed is not reset, different numbers appear with every invocation:

## **Sunburst Charts:**

Sunburst charts represent hierarchical data in the form of concentric circles. Here the innermost circle is the root node which defines the parent, and then the outer rings move down the hierarchy from the center. They are also called radial charts. We can use them to plot

**Plotly Dash:**

Dash is an open-source, **user interface** Python library for creating reactive web-based applications. It's both enterprise-ready and a first-class member of Plotly’s open-source tools.

Dash applications are web servers running Flask and communicating JSON packets over HTTP requests.

It's easy to build **graphical user interfaces(GUI)** using Dash as it abstracts all technologies required to make the applications.

Dash is declarative and reactive.

It can be rendered in a web browser and deployed to servers.

It provides a simple reactive decorator for binding code to UI. They are inherently mobile and cross-platform-ready.

**Dash Components:**

1. Dash Core Components *(import dash\_core\_components as dcc)*
2. Dash HTML components *(import dash\_html\_components as html)*

<https://realpython.com/python-dash/>

# Summary: Creating Dashboards with Plotly and Dash

Congratulations! You have completed this module. At this point in the course, you know:

* Dash is an Open-Source User Interface Python library for creating reactive, web-based applications.
* It is easy to build Graphical User Interfaces using Dash as it abstracts all technologies required to make the applications.
* There are two components of Dash: Core and HTML components.
* The dash\_core\_components describe higher-level interactive components generated with JavaScript, HTML, and CSS through the React.js library.
* The dash\_html\_components library has a component for every HTML tag.
* A callback function is a python function that is automatically called by Dash whenever an input component's property changes.
* The @app.callback decorator decorates the callback function in order to tell Dash to call it whenever there is a change in the input component value.
* The callback function takes input and output components as parameters and performs operations to return the desired result for the output component.