# (A Brief) Introduction to Bokeh

Interactive Data Visualisation in Python

## Overview

- What is Bokeh?
- Walk through examples exploring functionality
- Deployment options
- Tips
- Wrap up

## The Python Visualisation Landscape

- https://speakerdeck.com/jakevdp/pythons-visualization-landscape-pycon-2017

## What is Bokeh?

- Data Visualisation Library for Python
- Web first
- Supports static charts or fully interactive visualisation web apps
- Write Python - use the web for data vis without writing html, css, and js
- Functionality similar to products like Tableau or Power BI

## Building basic charts

- Start by defining an output file
- Define some data (Bokeh expects a sequence of values)
- Create a figure. This is like an empty canvas
- Add glyphs (also known as renderers) to the figure
- Generate an output file and open in browser

## What happens?

- Bokeh has two main components:
    - Bokeh - python library for defining charts
    - BokehJS - javascript library for rendering charts in browser
- Bokeh generates an HTML document
- Bokeh generates json definition of chart which includes data and chart settings, and embeds this in the document
- BokehJS uses the json to render chart and maps data onto the canvas

## Styling visualisations

- Bokeh has an extensive range of visual properties that can be set
- Can modify visual properties of objects such as Axes, Glyphs, Legends, etc
- Properties include: size, fill colors, line colors, alphas (transparency), fonts, etc
- Styles can be set with keyword args at object creation or by modifying attributes after
- Support for themes in certain scenarios
- Includes a large collection of built-in palettes
- Reduce the “data ink”

## Layouts

- Support for grid based layouts
- Bokeh provides Row and Column objects and able to define a layout as nested lists of plots
- Layouts also support a range of ‘sizing modes’ which includes support for responsive charts
- Most Layout objects inherit from LayoutDOM, check docs for sizing and other properties

## Working with data

- Bokeh expects equal length sequences
- Bokeh has limited support for data transformation, this is left to the user
- Bokeh has a representation of a data source called **ColumnDataSource**, a mapping of column names to lists of values
- When **ColumnDataSource** is used this enables advanced features such as streaming, data sharing between plots, and filtering
- **ColumnDataSource** expects a **dict** type but can also accept pandas dataframes
- **ColumnDataSource** can be provided to a glyph/renderer

## Interactivity introduction

- Delivered through a client - server model
- Uses websockets for two way communication between client and server
- Create our script then in the terminal/command line start bokeh server via the command:

```
bokeh serve file.py
```

- Bokeh Server renders visualisations etc.
- Bokeh Server listens for events on the front end and responds with new data
- bokehjs re-draws visualisations with new data

## Building interactivity

- Build interactivity through widgets and events on a visualisation e.g. changes of value
- Create a widget then register a callback function
- Callback is executed when associated event occurs
- Callback has access to widget’s values (and other information) and uses these to fetch new data and send this to the front-end (client) for rendering

## Advanced interactivity - Bokeh Apps

- Same design but offers finer control and flexibility
- When running as an app bokeh expects a given directory structure
- Can customise the default template and application resources such as css
- Includes Server Lifecycle Hooks for executing code at certain server events e.g. when new session created
- Start bokeh server from terminal/command line but pass in the directory name (not script name):

```
bokeh serve app-name
```

## Sharing and Deployment

- Static charts are (generally) self contained files and can be distributed as email attachments, via shared directories, hosted, etc.
- Interactive visualisations are more work
- Bokeh Server is not designed to be fully capable web server. Small internal network loads are ok
- Official docs recommend reverse proxy traffic to Bokeh Server via dedicated HTTP server e.g. nginx

## Tips & Tricks 1

- Use Browser debugging tools (Web Inspector) to help debugging
- Avoid too much data wrangling/organisation in the visualisation code. Offload e.g. to SQL or stored procedure, or process in advance of time
- More visualisations = more code. Take advantage of Python’s features organising a vis into packages or modules of higher level components e.g. datasources.py, widgets.py, etc
- Close database connections explicitly if working with pandas (or use with context manager)!

## Tips & Tricks 2

- Navigating the docs:
  - User Guide - best place for how to
  - Reference - best place for looking up visual properties you can modify
- Bokeh has changed a lot since pre 1.0. Older solutions on SO etc. may not work, consider filtering search results by time
- While not essential some knowledge of HTML, CSS, and JavaScript goes a long way with Bokeh

## Worth mentioning

- Holoviews: Higher level visualisation API. Offers less used visualisations . Built on top of Bokeh and others
- Plot.ly: Very similar to Bokeh i.e. web first, interactive, very glossy
- Food for thought:
  - https://stackoverflow.blog/2019/07/16/google-looker-salesforce-tableau-bi-open-source-alternatives/