In [1]:
```
# setup notebook
# imports
import os
import numpy as np
import random
from pathlib import Path

import bokeh
from bokeh.layouts import gridplot
from bokeh.plotting import output_notebook
output_notebook() # set default; alternative is output_file()


# notebook formatting
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))

# fix RISE scollbar missing
from traitlets.config.manager import BaseJSONConfigManager
path = "C:\\Users\\chris\\.jupyter\\nbconfig"
cm = BaseJSONConfigManager(config_dir=path)

cm.update("livereveal", {
              "scroll": True,
});

# also check jupyter Cell menu/Toggle Scrolling
```

Bokeh

Meaning of Bokeh

- Japanese word **“bokeh”** used in photography to describe *blurring of the out-of-focus parts of an image.*

[James Drury example - Flickr](https://www.flickr.com/photos/james_drury/15923166238/)

- How do you pronounce this crazy word?
  - bouquet
  - bok-ah
  - **both are fine** (https://docs.bokeh.org/en/0.10.0/docs/faq.html#how-do-you-pronounce-bokeh)

Bokeh

- bokeh homepage (https://bokeh.pydata.org/en/latest/index.html)
- python library
- focused on interactive visualization
- targets browsers for presentation
- goals: elegant, concise, designed for large/streaming data

Two modes

- client based
- server

- Bokeh uses Python to create high level objects (plots, subplots, lines, etc).
- ...and then renders everything in Javascript
- ...for the browser to display

workflow:
data --> python --> Bokeh --> Bokeh ColumnDataSource--> BokehJS --> Javascript -->Browser --> your eyes

In [2]:
```
# bokeh makes it easy to work with your data
from bokeh.io import show, output_notebook
from bokeh.models import ColumnDataSource
from bokeh.palettes import Spectral6
from bokeh.plotting import figure

import bokeh.palettes  # http://docs.bokeh.org/en/1.3.2/docs/reference/palettes.html

output_notebook()

fruits = ["Apples", "Pears", "Nectarines", "Plums", "Grapes", "Strawberries"]
counts = [5, 3, 4, 2, 4, 6]
source = ColumnDataSource(data=dict(fruits=fruits, counts=counts, color=Spectral6))

p = figure(
    x_range=fruits,
    y_range=(0, 9),
    plot_height=350,
    plot_width=1000,
    title="Fruit Counts",
)

p.vbar(
    x="fruits",
    top="counts",
    width=0.9,
    color="color",
    legend_field="fruits",
    source=source,
)
p.xgrid.grid_line_color = None
p.legend.orientation = "horizontal"
show(p)
```

Interaction with Bokeh

...using ipywidgets in notebooks, plus...

1- Toolbar -- configure bar & change what you see
2- Select data points
3- Add hover actions
4- Link subplots and selections
5- Highlight data using the legend

Example 1 - interaction with ipywidgets

In [3]:
```
# simple interactivity example - using ipywidgets
# credit:  https://github.com/bokeh/bokeh/blob/1.3.4/examples/howto/notebook_comms/Jupyter%20Interactors.ipynb

from ipywidgets import interact
import numpy as np

from bokeh.io import push_notebook, show, output_notebook
from bokeh.plotting import figure

output_notebook()

# define evenly spaced data to plot trig functions
x = np.linspace(0, 2 * np.pi, 2000)
y = np.sin(x)

graph_function = figure(
    title="example - simple trig plot",
    plot_height=300,
    plot_width=900,
    y_range=(-10, 10),
    background_fill_color="#efefef",
)
r = graph_function.line(x, y, color="#8888cc", line_width=1.5, alpha=0.8)


def update(function, frequency=10, amplitude=2, phi=0):
    if function == "sin":
        func = np.sin
    elif function == "cos":
        func = np.cos
    elif function == "tan":
        func = np.tan
    r.data_source.data["y"] = amplitude * func(frequency * x + phi)
    push_notebook()


interact(
    update,
    function=["sin", "cos", "tan"],
    frequency=(0, 50),
    amplitude=(1, 10),
```

```
    phi=(0, 20, 0.5),
)
show(graph_function, notebook_handle=True)
# handle updates existing plot, only needed in jupyter notebook (not jupyterlab)
```

(https://bokeh.org) BokehJS 1.4.0 successfully loaded.

Out[3]: <Bokeh Notebook handle for **In[3]**>

Example 2 - Bokeh toolbar and interactions

4 categories

- **Gestures**
  -- Pan/Drag Tools
  -- Click/Tap Tools
  -- Scroll/Pinch Tools
- **Actions** (e.g. save, reset)
- **Inspectors** report extra info -- Hovertool -- Crosshair
- **Edit tools** (e.g. various drawing tools)

docs - bokeh toolbar (https://docs.bokeh.org/en/latest/docs/user_guide/tools.ht0ml)

In [4]:
```
from bokeh.plotting import figure, output_notebook, show

x = [x for x in range(0, 11)]
y = [9, 8, 7, 5, 4, 6, 8, 3, 2, 0, 1]

chart = figure(
    title="simple line chart (with default toolbar)",
    x_axis_label="x axis!",
    y_axis_label="y axis",
    toolbar_location="right",
)

chart.xaxis.axis_label_text_font_size = "18pt"
chart.yaxis.axis_label_text_font_size = "18pt"

chart.line(x, y, line_width=2)

show(chart)

# review interactive tools
```

In [5]:
```
# adding tools and annotations
from bokeh.models import BoxAnnotation

low_box = BoxAnnotation(top=5, bottom=2, fill_alpha=0.1, fill_color="green")

# create new chart using different object from prior cell
chart2 = figure(
    title="simple chart with annotations",
    x_axis_label="x axis!",
    y_axis_label="y axis",
    toolbar_location="right",
)

chart2.xaxis.axis_label_text_font_size = "18pt"
chart2.yaxis.axis_label_text_font_size = "18pt"

chart2.circle(x, y, size=15, color="#AA3939")
chart2.add_layout(low_box)

show(chart2)
```

Introducing Bokeh ColumnDataSource object

- Bokeh class
- Bokeh can plot directly from Pandas DataFrames AND
- ColumnDataSource helps maps names of columns to sequences or arrays

```
from bokeh.models import ColumnDataSource

data = {"x": [1,2,3,4], "y": [11, 22, 33, 44], labels=["alpha", "bravo", "charlie", "delta"]}

source = ColumnDataSource(data,  color=Colorblind)
```

why use this?

- easier to share data + selections between plots and subplots
- efficient graphing of streaming - bokeh only sends new data to plots
- offloads work to the browser (e.g. colormapping)

- color examples: https://docs.bokeh.org/en/latest/docs/reference/palettes.html (https://docs.bokeh.org/en/latest/docs/reference/palettes.html)

Example 3 - Hover Tools

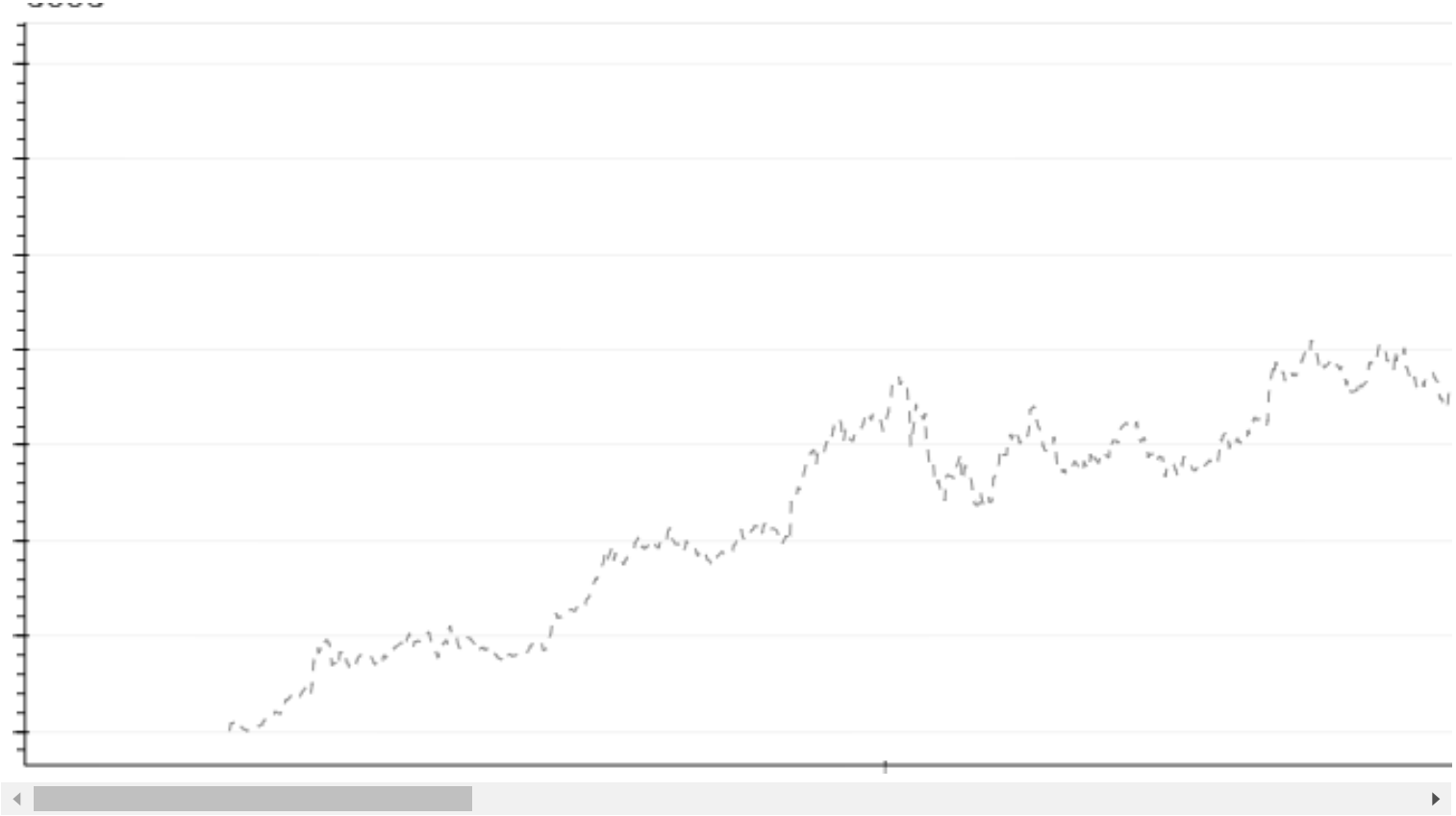
Hover tools provide additional insight in the context of your graph

In [6]:
```
# download bokeh sample data - will download to $HOME/.bokeh/data - and create directories if ne
cessary
bokeh_data_path = Path.home() / ".bokeh" / "data" / "US_Counties.csv"
if not Path.exists(bokeh_data_path):
    print("data not downloaded")
    bokeh.sampledata.download()

from bokeh.plotting import figure
from bokeh.models import ColumnDataSource
from bokeh.models.tools import HoverTool
from bokeh.io import show

# sample data includes:  AAPL, FB, GOOG, IBM, MSFT; change import symbol
from bokeh.sampledata.stocks import GOOG

tmp = GOOG

tmp["adj close"] = GOOG["adj_close"]

tmp["date"] = np.array(
    GOOG["date"], dtype=np.datetime64
)  # convert date strings to real datetimes

p = figure(
    x_axis_type="datetime", title="GOOG", plot_height=400, sizing_mode="stretch_width"
)
p.xgrid.grid_line_color = None
p.ygrid.grid_line_alpha = 0.5
p.xaxis.axis_label = "Year"
p.yaxis.axis_label = "Closing Price (USD)"

p.line(
    "date",
    "adj close",
    source=ColumnDataSource(data=tmp),
    line_dash="dashed",
    line_color="grey",
)


hover_tooltip = HoverTool(
    tooltips=[
```

```
        ("date", "@date{%F}"),
        ("close", "$@{adj close}{%0.2f}"),  # use @{ } for field names with spaces
        ("volume", "@volume{0.00 a}"),
    ],
    # use bokeh formatter classes
    formatters={
        "date": "datetime",  # use "datetime" formatter for "date" field
        "adj close": "printf",  # use "printf" formatter for "adj close" field
        # use default "numeral" formatter for other fields
    },
    # display a tooltip whenever the cursor is vertically in line with a glyph
    mode="vline",
)

p.add_tools(hover_tooltip)

show(p)
```

In [7]:
```
# download bokeh sample data - will download to $HOME/.bokeh/data - and create directories if ne
cessary
bokeh_data_path = Path.home() / ".bokeh" / "data" / "US_Counties.csv"
if not Path.exists(bokeh_data_path):
    print("data not downloaded")
    bokeh.sampledata.download()

from bokeh.plotting import figure
from bokeh.models import ColumnDataSource
from bokeh.models.tools import HoverTool
from bokeh.io import show

# sample data includes:  AAPL, FB, GOOG, IBM, MSFT; change import symbol
from bokeh.sampledata.stocks import GOOG

tmp2 = GOOG

tmp2["adj close"] = GOOG["adj_close"]

tmp2["date"] = np.array(
    GOOG["date"], dtype=np.datetime64
)  # convert date strings to real datetimes

p2 = figure(
    x_axis_type="datetime", title="GOOG", plot_height=400, sizing_mode="stretch_width"
)
p2.xgrid.grid_line_color = None
p2.ygrid.grid_line_alpha = 0.5
p2.xaxis.axis_label = "Year"
p2.yaxis.axis_label = "Closing Price (USD)"

p2.line(
    "date",
    "adj close",
    source=ColumnDataSource(data=tmp2),
    line_dash="dashed",
    line_color="grey",
)

p2.circle(
    "date",
    "adj close",
```

```
    name="red_circle",
    source=ColumnDataSource(data=tmp2),
    size=12,
    fill_color="grey",
    hover_fill_color="firebrick",
    fill_alpha=0,
    hover_alpha=0.2,
    line_color=None,
    hover_line_color="white",
)

p2.add_tools(HoverTool(tooltips=[None], names=["red_circle"], mode="hline"))

show(p2)
```



Example 4 - Interactive Linked Brushing & Results Export

Three main points:

1. Linking plots is easy & useful
2. Hover tools provide additional insights
3. Helpful error messages

Warning - JavaScript ahead!

In [8]:
```
# Generate linked plots
# show helpful error messages
# table and save button
# show hover tool

from random import random

from bokeh.io import output_notebook  # prevent opening separate tab with graph
from bokeh.io import show

from bokeh.layouts import row
from bokeh.layouts import grid
from bokeh.models import CustomJS, ColumnDataSource
from bokeh.models import Button  # for saving data
from bokeh.models.widgets import DataTable, DateFormatter, TableColumn
from bokeh.models import HoverTool
from bokeh.plotting import figure

from bokeh.resources import INLINE

bokeh.io.output_notebook(INLINE)
```

(https://bokeh.org)
BokehJS 1.4.0 successfully loaded.

In [9]:
```
# !if not error - break and fix - demo error msg
# show bokeh code
# show helpful error messages
# demo linked plots
# demo table
# demo save selected results to file
# demo hover tool

# create data
x = [random() for x in range(500)]
y = [random() for y in range(500)]

# create first subplot
plot_width = 400
plot_height = 400

s1 = ColumnDataSource(data=dict(x=x, y=y))
fig01 = figure(
    plot_width=plot_width,
    plot_height=plot_height,
    tools=["lasso_select", "reset", "save"],
    title="Select Here",
)
fig01.circle("x", "y", source=s1, alpha=0.6)

# create second subplot
s2 = ColumnDataSource(data=dict(x=[], y=[]))

# demo smart error msg:  `box_zoom`, vs `BoxZoomTool`
fig02 = figure(
    plot_width=400,
    plot_height=400,
    x_range=(0, 1),
    y_range=(0, 1),
    tools=["box_zoom", "wheel_zoom", "reset", "save"],
    title="Watch Here",
)

# corrected code - reference
# fig02 = figure(plot_width=plot_width, plot_height=plot_height, x_range=(0, 1), y_range=(0, 1),
#            tools=["pan", "box_zoom", "wheel_zoom", "reset"], title="Watch Here")
```

```
fig02.circle("x", "y", source=s2, alpha=0.6, color="firebrick")

# create dynamic table of selected points
columns = [
    TableColumn(field="x", title="X axis"),
    TableColumn(field="y", title="Y axis"),
]

table = DataTable(
    source=s2,
    columns=columns,
    width=400,
    height=600,
    sortable=True,
    selectable=True,
    editable=True,
)
```

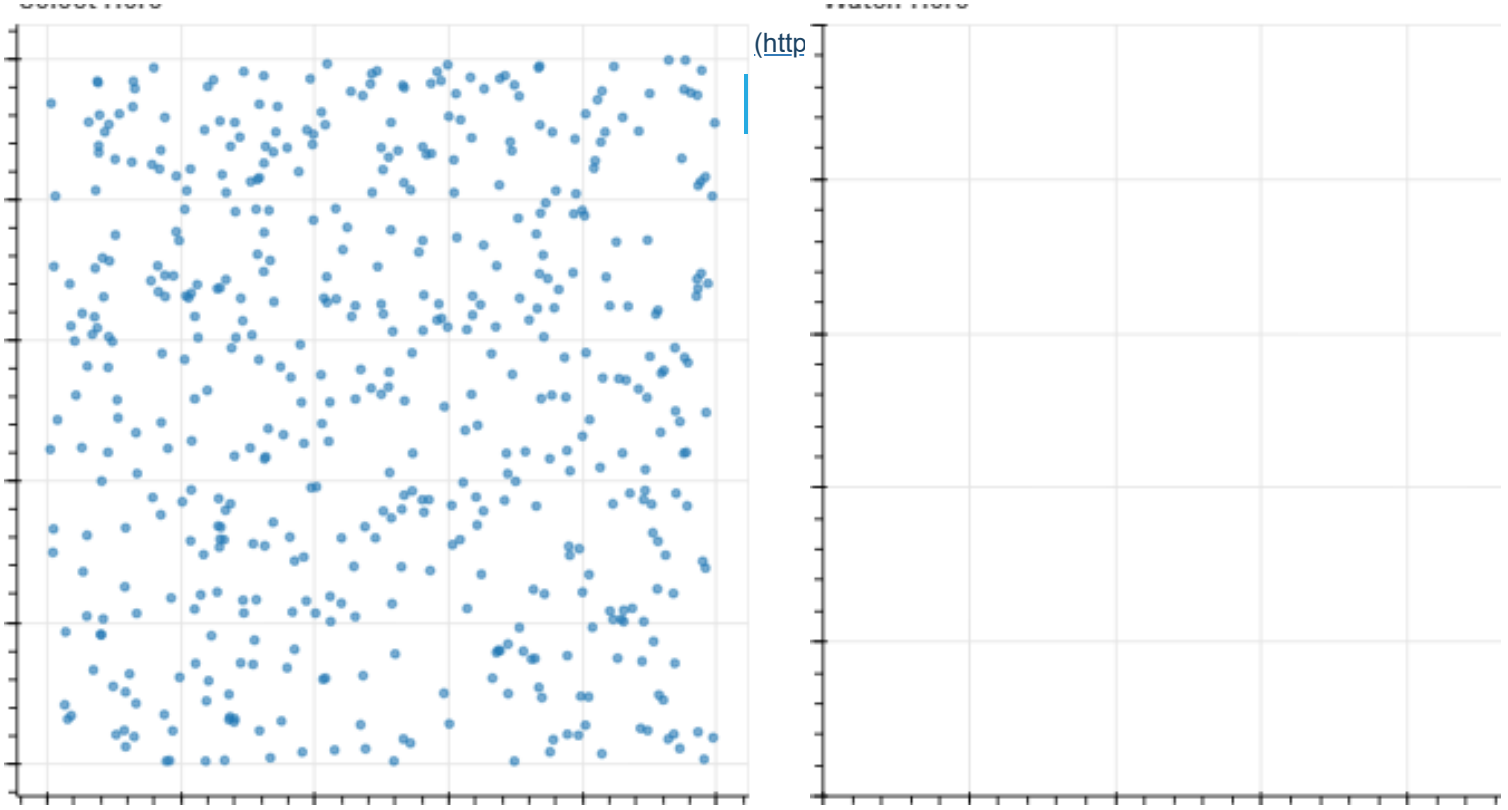
In [10]:
```
# fancy javascript to link subplots
# js pushes selected points into ColumnDataSource of 2nd plot
s1.selected.js_on_change(
    "indices",
    CustomJS(
        args=dict(s1=s1, s2=s2, table=table),
        code="""
        var inds = cb_obj.indices;
        var d1 = s1.data;
        var d2 = s2.data;
        d2['x'] = []
        d2['y'] = []
        for (var i = 0; i < inds.length; i++) {
            d2['x'].push(d1['x'][inds[i]])
            d2['y'].push(d1['y'][inds[i]])
        }
        s2.change.emit();
        table.change.emit();
    """,
    ),
)
```

In [11]:
```
# create save button - saves selected datapoints to text file onbutton
savebutton = Button(label="Save", button_type="success")
savebutton.callback = CustomJS(
    args=dict(source_data=s1),
    code="""
        var inds = source_data.selected.indices;
        var data = source_data.data;
        var out = "x, y\\n";
        for (i = 0; i < inds.length; i++) {
            out += data['x'][inds[i]] + "," + data['y'][inds[i]] + "\\n";
        }
        var file = new Blob([out], {type: 'text/plain'});
        var elem = window.document.createElement('a');
        elem.href = window.URL.createObjectURL(file);
        elem.download = 'selected-data.txt';
        document.body.appendChild(elem);
        elem.click();
        document.body.removeChild(elem);
        """,
)
```

In [12]:
```
# add Hover tool
# define what is displayed in the tooltip
tooltips = [
    ("X:", "@x"),
    ("Y:", "@y"),
    ("static text", "static text"),
    (
        "image",
        """<div>
                            <img
                            src="./images/streaming_water_living_stills.gif" height="40" alt="im
age"
                            style="float: left; margin: 0px 15px 15px 0px; image-rendering: pixe
lated;"
                            border="2"
                            ></img>
                        </div>""",
    ),
]

fig02.add_tools(HoverTool(tooltips=tooltips))
```

In [13]:
```
# display results
# demo linked plots
# demo zooms and reset
# demo hover tool
# demo table
# demo save selected results to file


layout = grid([fig01, fig02, table, savebutton], ncols=3)

output_notebook()
show(layout)
```

(https://bokeh.org) BokehJS 1.4.0 successfully loaded.



In [14]:
```
import pandas as pd

data = (Path.cwd()).parent / "data" / "selected-data.txt"
df = pd.read_csv(data, sep=",")
df.head(5)
```

Out[14]:

	x	y
0	0.080379	0.021882
1	0.965410	0.912836
2	0.265163	0.289784
3	0.714565	0.565581
4	0.855827	0.972430

Server Based Plotting

Why use a server?

- Bokeh's reactive client-server model can trigger server-side code

- data >> local machine or notebook
  -- server supports automatic downsampling
  -- supporting tools like datashader can manage millions of points in realtime
- need to sync between server and browser:
  -- respond to browser events (python computations or db queries)
  -- automatically push updates the UI browser
  -- streaming (periodic, and async)

- create deployable apps (e.g. dashboard)

Bokeh resources

bokeh docs (https://docs.bokeh.org/en/latest/)

bokeh cheat sheet (credit: codecamp) (https://datacamp-community-prod.s3.amazonaws.com/f9511cf4-abb9-4f52-9663-ea93b29ee4b7)

Bokeh Server Examples (http://docs.bokeh.org/en/1.3.2/docs/gallery.html)

Quiz - Alberto Cairo

link to Example 4 - working with Streaming Data (http://localhost:8888/notebooks/notebooks/03_bokeh_with_streamin

In [ ]: