1) **Dash** in Python is similar to Shiny in R. It allows interactivity in a browser window with heavy-duty computing underneath.
2) The website is https://dash.plot.ly, and it is an open source project of Plotly with an Enterprise version. Graphs are based on Plotly code.
3) Install with `python -m pip install dash` (or `python3` if python2.7 is your default).
4) Check out the gallery at https://dash.plot.ly/gallery to get an idea of what is possible.
5) The tutorial is at https://dash.plot.ly/getting-started.
6) A first Dash app (dash1.py)

```
# Learning dash
# H. Seltman, April 2019
# ref: https://dash.plot.ly/getting-started?
#         _ga=2.235236482.2020250661.1555942446-13138461.1554481284

import dash
import dash_core_components as dcc
import dash_html_components as html
import pandas as pd
import plotly.graph_objs as go

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)

colors = {
    'background': '#111111',
    'text': '#7FDBFF'
}

df = pd.read_csv(
    'https://gist.githubusercontent.com/chriddyp/' +
    '5d1ea79569ed194d432e56108a04d188/raw/' +
    'a9f9e8076b837d541398e999dcbac2b2826a81f8/' +
    'gdp-life-exp-2007.csv')


app.layout = \
    html.Div(
        style={'backgroundColor': colors['background'],
               'color': colors['text']},
        children=[
            html.H1('Hello Dash', style={'textAlign': 'center',
                    'color': '#DD00FF'}),

            html.Div(["Dash: A web application framework for Python.",
                      html.Br(),
                      "Now is the time..."
                      ]),
```

```
        dcc.Graph(
            id='life-exp-vs-gdp',
            figure={
                'data': [
                    go.Scatter(
                        x=df[df['continent'] == i]['gdp per capita'],
                        y=df[df['continent'] == i]['life expectancy'],
                        text=df[df['continent'] == i]['country'],
                        mode='markers',
                        opacity=0.7,
                        marker={'size': 15,
                                'line': {'width': 0.5,
                                'color': 'white'}},
                        name=i
                    ) for i in df.continent.unique()
                ],
                'layout': go.Layout(
                    xaxis={'type': 'log', 'title': 'GDP Per Capita'},
                    yaxis={'title': 'Life Expectancy'},
                    margin={'l': 40, 'b': 40, 't': 10, 'r': 10},
                    legend={'x': 0, 'y': 1},
                    hovermode='closest'
                )
            }  # end figure=
        )  # end Graph()
    ])  # end main Div in layout
```

7) Dash and Markdown

a. One or more children of any `html.Div()` in the Dash Layout can be a `dcc.Markdown()` element where the argument is a str or a list of str's containing basic Markdown (no LaTeX).

b. E.g., you can add this to your app:

```
markdown_text = '''
## My equation
See [My Homepage](http://www.stat.cmu.edu/~hseltman/Search.html).

It is:
*     Good
*     Useful
---

My equation is
α = 3 * β + γ  (Unicode is OK, but not LaTeX.)
'''
```

This shows up where you code `dcc.Markdown(markdown_text)`.

8) Other Dash Core Components (besides dcc.Graph)

a. These are like Shiny widgets.

b. **Dropdown box** (list of dictionaries, each with 'label' and 'value')

```
dcc.Dropdown(id='city-dropdown',
        options=[
            {'label': 'New York City', 'value': 'NYC'},
            {'label': 'Montreal', 'value': 'MTL'},
            {'label': 'San Francisco', 'value': 'SF'}
        ],
        value='MTL', multi=False)
```

Set 'multi' to True to allow the user to select multiple values

c. Slider

```
dcc.Slider(id='my-slider',
        min=0,  max=20, step=0.5, value=10,
        marks={0: '0', 3: '3', 10: '10', 20:'20'})
```

d. **Other dccs**: Checklist, ConfirmDialog, ConfirmDialogProvider, DatePickerRange, DatePickerSingle, Input, Interval, Link, Loading, Location, LogoutButton, RadioItems, RangeSlider, Store, Tab, Tabs, Textarea, Upload

e. **Other Graph Types**: area, bar, barpolar, box, candlestick, carpet, choropleth, cone, contour, contourcarpet, heatmap, histogram, pie, pointcloud, scatter3d, scattercarpet, scattergeo, scattergl, scattermapbox, scatterpolar, scatterpolargl, scatterternary, sunburst, violin, waterfall

9) **Callbacks** (Combines the reactive and render functions of Shiny)

a. You write a function that takes input values from input widgets (dccs) and returns value(s) that change output widgets or html elements. Use the `@app.callback(Output, Input)` decorator connects your function to the widgets.

b. Example:

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Input(id='my-id', value='initial value', type='text'),
    html.Div(id='my-div')
])


@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')])
def update_output_div(input_value):
    return 'You entered "{}"'.format(input_value)
```

c. The use of `Output()` and `Input()` is required to construct the full arguments for the decorator. There are also `State()` and `Event()` functions. The `State()` function is analogous to `isolate()` in Shiny; the values are passed to the function, but changes in them don't trigger the function. `Event('go-button', 'click')`, e.g., can be used to make the function reactive to an event.

d. Whenever a mentioned input or state changes, the callback function is called.

e. Other callback examples include updating the children of a component to display new text, the figure of a dcc.Graph component to display new data, the style of a component, and the available options of a dcc.Dropdown component.

f. **Slider example** (dashSlider.py)

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output
import pandas as pd
import plotly.graph_objs as go

df = pd.read_csv('https://raw.githubusercontent.com/plotly/'
                 'datasets/master/gapminderDataFiveYear.csv')

app = dash.Dash(__name__)

app.layout = html.Div([
    dcc.Graph(id='graph-with-slider'),
    dcc.Slider(
        id='year-slider',
        min=df['year'].min(), max=df['year'].max(),
        value=df['year'].min(), step=None,
        marks={str(year): str(year) for year in
               df['year'].unique()})
])

@app.callback(
    Output('graph-with-slider', 'figure'),
    [Input('year-slider', 'value')])
def update_figure(selected_year):
    filtered_df = df[df.year == selected_year]
    traces = []
    for i in filtered_df.continent.unique():
        df_by_continent = \
            filtered_df[filtered_df['continent'] == i]
        traces.append(go.Scatter(
            x=df_by_continent['gdpPercap'],
            y=df_by_continent['lifeExp'],
            text=df_by_continent['country'],
            mode='markers',
            opacity=0.7,
            marker={
                'size': 15,
                'line': {'width': 0.5, 'color': 'white'}
            },
            name=i
        ))
```

```
            return {
                'data': traces,
                'layout': go.Layout(
                    xaxis={'type': 'log', 'title': 'GDP Per Capita'},
                    yaxis={'title': 'Life Expectancy',
                           'range': [20, 90]},
                    margin={'l': 40, 'b': 40, 't': 10, 'r': 10},
                    legend={'x': 0, 'y': 1},
                    hovermode='closest'
                )
            }  # end defining returned dictionary ('figure')
```

10) Chained callbacks

The output of one callback function can be the input of another callback function. This can be used to create dynamic UIs where one input component updates the available options of the next input component.

```
all_options = {
    'America': ['New York City', 'San Francisco', 'Cincinnati'],
    'Canada': [u'Montréal', 'Toronto', 'Ottawa']
}
app.layout = html.Div([
    dcc.RadioItems(
        id='countries-dropdown',
        options=[{'label': k, 'value': k} for k in all_options.keys()],
        value='America'
    ),
    html.Hr(),
    dcc.RadioItems(id='cities-dropdown'),
    html.Hr(),
    html.Div(id='display-selected-values')
])


@app.callback(
    Output('cities-dropdown', 'options'),
    [Input('countries-dropdown', 'value')])
def set_cities_options(selected_country):
    return [{'label': i, 'value': i} for i in
all_options[selected_country]]


@app.callback(
    Output('cities-dropdown', 'value'),
    [Input('cities-dropdown', 'options')])
def set_cities_value(available_options):
    return available_options[0]['value']


@app.callback(
    Output('display-selected-values', 'children'),
    [Input('countries-dropdown', 'value'),
     Input('cities-dropdown', 'value')])
def set_display_children(selected_country, selected_city):
    return u'{} is a city in {}'.format(
        selected_city, selected_country)
```

11) Full example

a. Title: Zika Outbreak Explorer
b. Author: Charley Ferrari
c. App Link: https://zika-crossfilter.herokuapp.com/
d. Code Link: https://github.com/charleyferrari/zika-crossfilter/