CSE 5520 Homework 1

Student	Lynn Pepin ('Tristan Pepin')
	tmp13009 / 2079724
Due date	2021 / Sept / 7

This is Lynn Pepin's report for CSE 5520 homework 1. It is organized with code first, and then the microlab.

1. Hands-on Microlab

Let's create an interactive chart using `pygal`.

The file `lynnkit` will hold all the helper-functions and whatnot I use in this course. When the code is provided or trivial (e.g. a fibonacci generator), I don't include it in the notebook.

1.1 Plotting with PyGal

In [1]:
```
import pygal as pg
import lynnkit as lk
```

In [2]:
```
# generator for our fib vals
fib_generator = lk.fibgen()
fib_vals = [next(fib_generator) for _ in range(1000)]
print("Some of our vibonacci values", fib_vals[:10])
```

```
Some of our vibonacci values [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

In [3]:
```
# render our plot to an svg
bar_chart = pg.Bar()
bar_chart.add('Fibonacci', fib_vals[:10])
bar_chart.render_to_file('hw1_f1_10.svg')
```

In [4]:
```
# render our plot to an svg; this time with 50 fib values
bar_chart = pg.Bar()
bar_chart.add('Fibonacci', fib_vals[:50])
bar_chart.render_to_file('hw1_f1_50.svg')
```

1.2. Plotting with Plotly

In [5]:
```
import pandas as pd
import plotly.graph_objects as go
```

In [6]:
```
# Example code for the lab
# load the data
df = pd.read_csv('finance-charts-apple.csv')
df.columns = [col.replace('AAPL', '') for col in df.columns]

# create plotly figure
fig = go.Figure()
fig.add_trace(
    go.Scatter(
        x = df['Date'],
        y = df['.High']
    )
)

# update figure title
fig.update_layout(
    title_text="Time series with range sliders and selectors"
)

# add range slider
fig.update_layout(
    xaxis = dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1, label="1m", step="month", stepmode=
                dict(count=6, label="6m", step="month", stepmode=
                dict(count=1, label="YTD", step="year", stepmode=
                dict(count=1, label="1y", step="year", stepmode="
                dict(step="all")
            ])
        ),
        rangeslider=dict(visible=True),
        type="date"
    )
)
```

In [7]:

```
# plotting 10 fib values

fig = go.Figure()
fig.add_bar(
    x = list(range(10)),
    y = fib_vals[:10]
)


fig.update_layout(
    title_text="Fibonacci sequence values, as a bar chart"
)
```

In [8]:

```
# plotting 50 fib values

fig = go.Figure()
fig.add_bar(
    x = list(range(50)),
    y = fib_vals[:50]
)

fig.update_layout(
    title_text="Fibonacci sequence values, as a bar chart"
)
```

1.3 Host a graph with Dash

This is a tool by the makers of Plotly that provides a server for visualization in the browser.

In [9]:
```
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
external_stylesheets = ['bWLwgP.css']
```

In [10]:
```
# example from the lab
app = dash.Dash(
    __name__,
    external_stylesheets = external_stylesheets
)

# create data
fruits  = ['Apples', 'Oranges', 'Bananas', 'Apples', 'Oranges', '
amounts = [4, 1, 2, 2, 4, 5]
cities  = ['SF', 'SF', 'SF', 'Montreal', 'Montreal', 'Montreal']

df = pd.DataFrame(
    {
        'Fruit'  : fruits,
        'Amount' : amounts,
        'City'   : cities
    }
)

# instantiate figure
fig = px.bar(
    df,
    x='Fruit', y='Amount', color='City', barmode='group'
)

# populate app
app.layout = html.Div(
    children=[
```

```
        html.H1(children='Fruits'),
        html.Div(children=''' Fruits Amounts in San Francisco and
        dcc.Graph(
            id='example-graph',
            figure=fig
        )
    ])

# run server
#app.run_server(port=8050, host='localhost')
```

In [ ]:

```
# now let's do the fib vals
app = dash.Dash(
    __name__,
    external_stylesheets = external_stylesheets
)

fig_fib10 = go.Figure()
fig_fib10.add_bar(
    x = list(range(10)),
    y = fib_vals[:10]
)

fig_fib10.update_layout(
    title_text="Fibonacci sequence from 0 to 10"
)

fig_fib50 = go.Figure()
fig_fib50.add_bar(
    x = list(range(50)),
    y = fib_vals[:50]
)

fig_fib50.update_layout(
    title_text="Fibonacci sequence from 0 to 10"
)

app.layout = html.Div(
    children=[
        html.H1(children='The Wonderful World of Fibonacci Sequer
        html.Div(children=''' Here we'll see two graphs, showing
        dcc.Graph(
            id='fib10',
            figure=fig_fib10
        ),
        dcc.Graph(
            id='fib50',
            figure=fig_fib50
        )
    ])

app.run_server(port=8050, host='localhost')
```

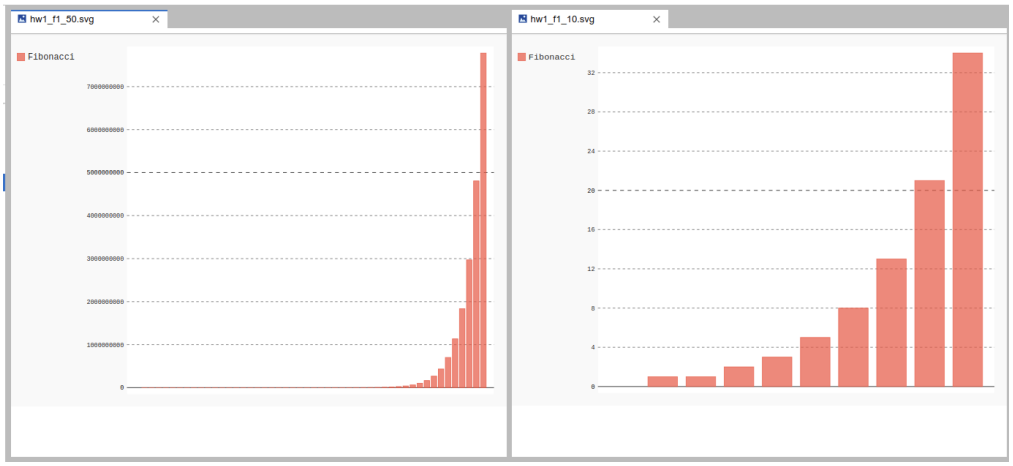
```
Dash is running on http://localhost:8050/

Dash is running on http://localhost:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a prod
uction deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://localhost:8050/ (Press CTRL+C to quit)
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /bWLwgP.css HTTP/1.1" 2
00 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash/deps/react@16.v1_21_0m1630541620.14.0.min.js HTTP/1.1" 200
-
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash/deps/polyfill@7.v1_21_0m1630541620.12.1.min.js HTTP/1.1" 2
00 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash/deps/prop-types@15.v1_21_0m1630541620.7.2.min.js HTTP/1.1"
200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash_html_components/dash_html_components.v1_1_4m1630541620.mi
n.js HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash/deps/react-dom@16.v1_21_0m1630541620.14.0.min.js HTTP/1.1"
200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash_core_components/dash_core_components-shared.v1_17_1m163054
1620.js HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash_core_components/dash_core_components.v1_17_1m1630541620.js
HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash/dash-renderer/build/dash_renderer.v1_21_0m1630541620.min.j
s HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-dependencies HTT
P/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-layout HTTP/1.1"
200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_favicon.ico?v=1.21.0
HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash_core_components/async-graph.js HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:24] "GET /_dash-component-suite
s/dash_core_components/async-plotlyjs.js HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:29] "GET /bWLwgP.css HTTP/1.1" 2
00 -
127.0.0.1 - - [07/Sep/2021 20:47:29] "GET /_dash-component-suite
s/dash_core_components/dash_core_components-shared.js.map HTTP/1.
1" 200 -
```

```
127.0.0.1 - - [07/Sep/2021 20:47:29] "GET /_dash-component-suite
s/dash_core_components/async-graph.js.map HTTP/1.1" 200 -
127.0.0.1 - - [07/Sep/2021 20:47:29] "GET /_dash-component-suite
s/dash_html_components/dash_html_components.min.js.map HTTP/1.1"
200 -
127.0.0.1 - - [07/Sep/2021 20:47:29] "GET /_dash-component-suite
s/dash_core_components/dash_core_components.js.map HTTP/1.1" 200
-
```

2. Screenshots of running code

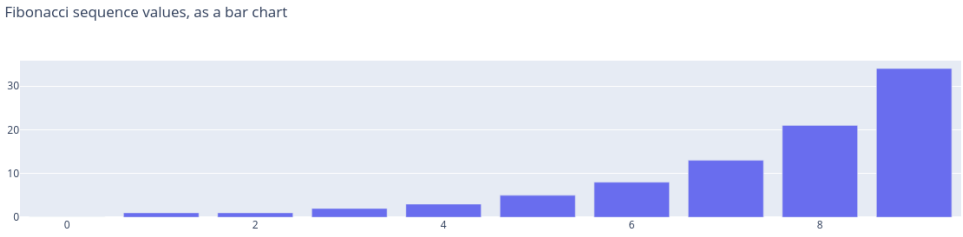
2.1. Pygal

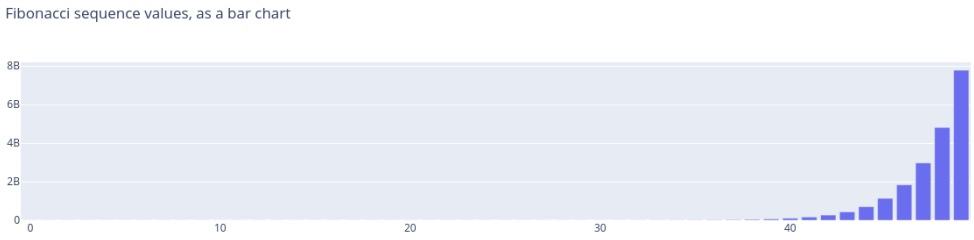
PyGal screenshot of two SVGs, with Fibonacci sequence values from 1 to 10, and from 1 to 50.



2.2. Plotly

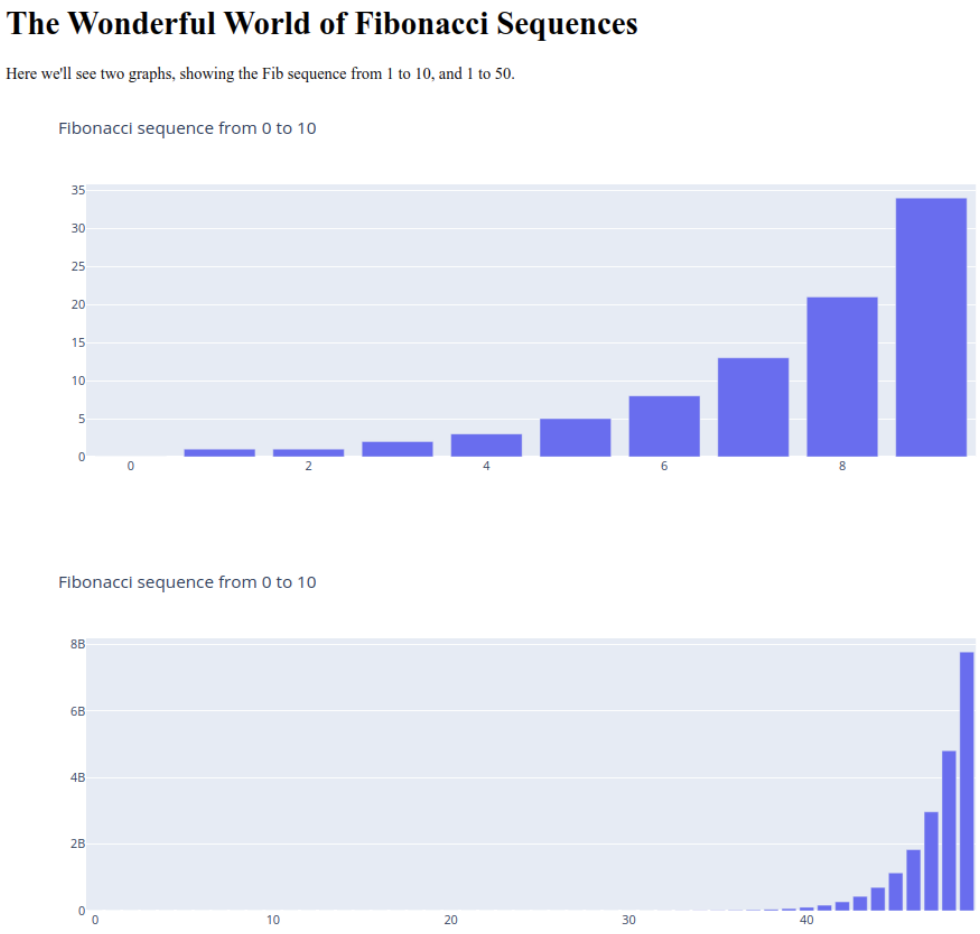
As above, so below, for Plotly





2.3. Dash

As above, so below, published in-browser using Dash.



Addendum: Code from `lynnkit`

I put extra code into `lynnkit`. I wrote a Fibonacci generator, which I am very proud of, so I have copied the pertinent code here.

```
def fibgen():
    """Provides a generator yielding the fibonacci
sequence

    :yields: int
    :returns: An iterator which yields the i-th value
of the Fibonacci sequence
        for each i-th call of next() on an instance
of fibgen
    :rtype: Iterator[int]

    >>> f = fibgen()
    >>> next(f)
    0
    >>> next(f)
    1
    >>> next(f)
    1
    >>> next(f)
    2
    >>> next(f)
    3
    """

    vals = [0, 1]
    ii = 0
    while True:
        yield vals[ii%2]
        vals[ii%2] += vals[(ii+1)%2]
        ii += 1
```

In [ ]: