Introduction to Data Science Programming in Python

J. Hathaway - Data Science Program Chair

Disclaimers

I am focusing on modern tools for data science in Python - Polars over Pandas, Plotly over Matplotlib, and Streamlit over Dash. These modern tools reflect the best of

- declarative programming (task focused programming)
- clean grammar (language abstraction that allows intuitive but complex actions)
- industry respect (all three tools are very popular for the quality and have rapid growth)

Agenda

Exemplify the data science process - Extract, Transform, Load, Analyze

1. Introduction and Set-up (30 minutes)
2. Polars for data munging (45 minutes)
3. Break (10 minutes)
4. Plotly for data visualization (45 minutes)
5. Streamlit for dashboards (45 minutes)

J. Hathaway

Data Scientists with ~20 years of industry experience and 8 years in Academia. Undergraduate degree in Economics (University of Utah) and a graduate degree in Statistics (BYU).

Checking our installation

1. Python Installed
2. VS Code Installed
3. Python VS Code Extension Installed
4. Python packages installed.

```
pip install polars[all] plotly streamlit
```

Introduction to Polars and Data Munging (speed)

Polars is a lightning fast DataFrame library/in-memory query engine. Its embarrassingly parallel execution, cache efficient algorithms and expressive API makes it perfect for efficient data wrangling, data pipelines, snappy APIs and so much more. Polars is about as fast as it gets, see the results in the H2O.ai benchmark.
Polars Website

Introduction to Polars and Data Munging (declarative API)

Polars functions (Contexts & Expressions) are human readable and they align with standard SQL methods as well as the very popular big data package PySpark.

Contexts

- **Selection:** `df.select([..])`, `df.with_columns([..])`
- **Filtering:** `df.filter()`
- **Groupby/Aggregation:** `df.groupby(..).agg([..])`

Expressions

```
new_df = df.select(
  pl.col("names").n_unique().alias("unique"),
  pl.approx_unique("names").alias("unique_approx"),
  (pl.col("nrs").sum() + 5).alias("nrs + 5"),
  pl.col("integers").cast(pl.Float32).alias("integers_as_floats"),
  pl.col("animal").str.n_chars().alias("letter_count")
)
```

Polars programming

Now let's practice using Polars with our installation of Python

1. Read data, melt data, and save as `.parquet` (01_read.py)
2. Munge data for visualization (02_munge.py)

Introduction to Parquet files

When users ask for data or when many institutions share data the files are usually some type of text file (`.txt`, `.csv`, `.tsv`) or an Excel file. The most ubiquitous format is `.csv`. However, these formats limit effective data handling. We want a format that stores small, reads fast, and maintains variable types. The Apache Arrow project facilitates the goal with the `.parquet` and `.feather` formats and their respective packages for leveraging those formats.

The following table from the openbridge blog provides a strong example of the benefits of these new formats.

Introduction to Data Visualization

Our eyes are drawn to colors and patterns. We can quickly identify red from blue, and squares from circles. Our culture is visual, including everything from art and advertisements to TV and movies. Data visualization is another form of visual art that grabs our interest and keeps our eyes on the message.

Advantages of data visualization:

- Easily sharing information.
- Interactively explore opportunities.
- Visualize patterns and relationships.

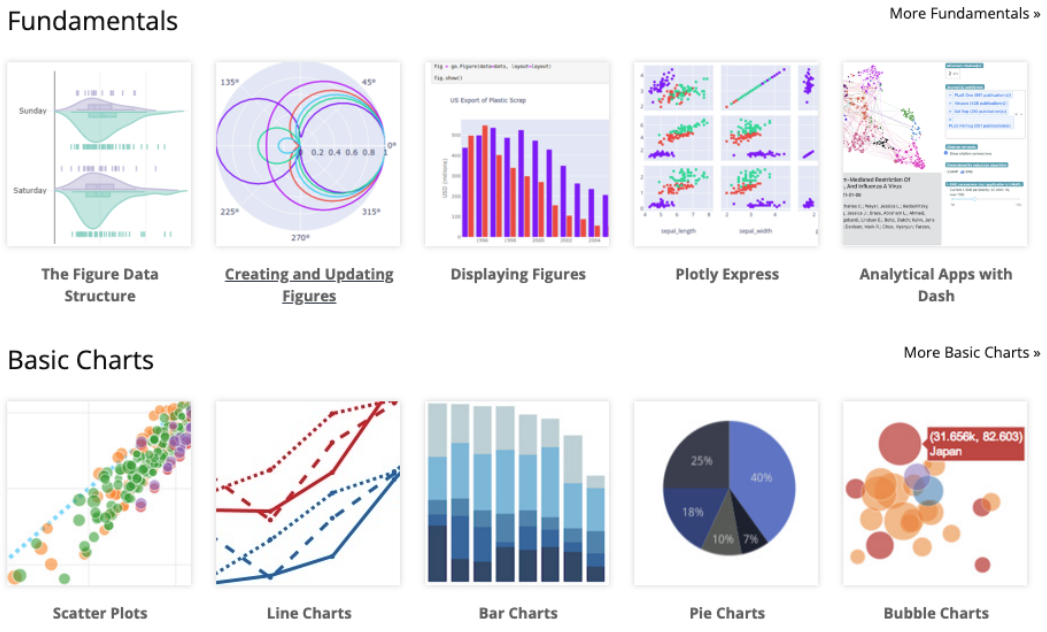
Tableau Reference

Disadvantages:

- Biased or inaccurate information.
- Correlation doesn't always mean causation.
- Core messages can get lost in translation.

Introduction to **Plotly** for Data Visualization

The Plotly Python package leverages the plotly.js JavaScript library to enables Python users to create beautiful interactive web-based visualizations. Plotly.js is built on top of d3.js and stack.gl, Plotly.js is a high-level, declarative charting library. plotly.js ships with over 40 chart types, including 3D charts, statistical graphs, and SVG maps.



Plotly programming

Now let's practice using Plotly with our installation of Python

1. Plotly practice (03_begin_plotly.py)
2. Data visualization with our munged data (03_explore.py)

```
import plotly.express as px
df = px.data.iris() # iris is a pandas DataFrame
fig = px.scatter(df, x="sepal_width", y="sepal_length")
fig.show()
```

```
import plotly.express as px
df = px.data.gapminder().query("continent == 'Oceania'")
fig = px.line(df, x='year', y='lifeExp', color='country', markers=True)
fig.show()
```

Introduction to Dashboarding (Structured design)

A dashboard is a way of displaying various types of visual data in one place that let's the user focus on one general topic but explore questions within that topic.

The 5-second Test

- What question does the dashboard answer?
- Stick to 1 main question. If not possible, use Text widget
- Complete = Nothing can be taken out

Purpose-Built

- For the role
- For the expertise level
- For the screen resolution
- For the viewing approach.
- For the size of the environment

Layout

- Top-line header as summary
- Divide dashboard into sections. See next slide

Performance

- Large number of objects
- Long time period (1 month = 8650 data)
- Data Transformation
- Top-N. No reason to go beyond 1 day.

Scope

- Don't mix scope. Performance, Capacity, Configuration, etc.
- Don't mix monitoring and troubleshooting. Troubleshooting has many permutation as it depends on the problem.

Color & Content

- Blank → Good
  - When you expect nothing
- Red → highly utilized
- Black → wastage

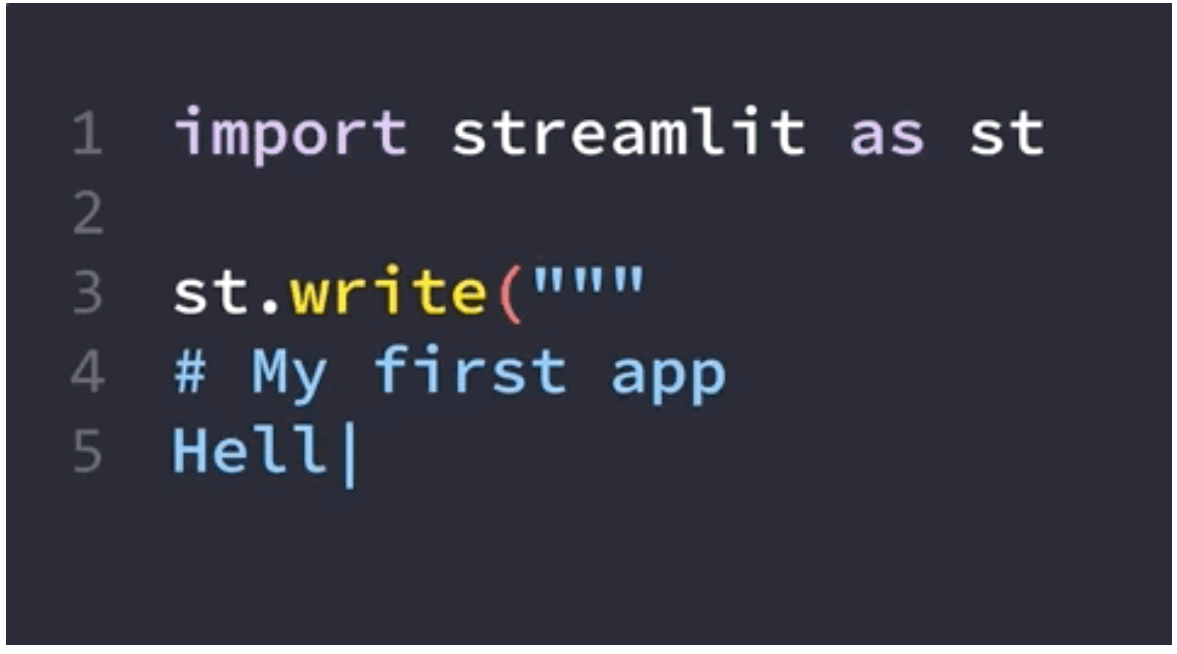
Introduction to Dashboarding (Audience)

A dashboard is a way of displaying various types of visual data in one place that let's the user focus on one general topic but explore questions within that topic.

[Reference])

Introduction to **Streamlit** dashboards

Streamlit turns data scripts into shareable web apps in minutes in pure Python. A faster way to build and share data apps with no front-end experience required.



Streamlit programming

Now let's practice using Streamlit with our installation of Python

1. Streamlit practice (04_dashboard.py)

```
import streamlit as st
import polars as pl

st.write("Here's our first attempt at using data to create a table:")
st.write(pl.DataFrame({
    'first column': [1, 2, 3, 4],
    'second column': [10, 20, 30, 40]
}))
```

Next Steps

1. Program your day (incorporate programming into your daily work)
2. Grow your skills -- BYU-I DS 250 Course and BYU-I CSE 450 Course
3. Challenge others -- Tidy Tuesday
4. Display your talent -- Github.com
5. Offer your services -- Find small projects to do for friends and contacts
6. Apply for jobs -- LinkedIn