Applications of Python

May 24, 2023

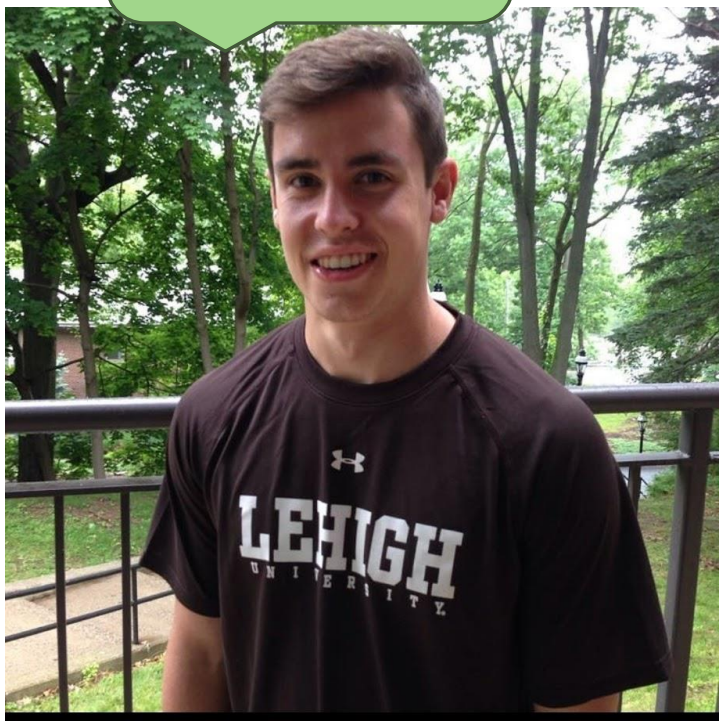
Washington Department of Health - Center for Data Science

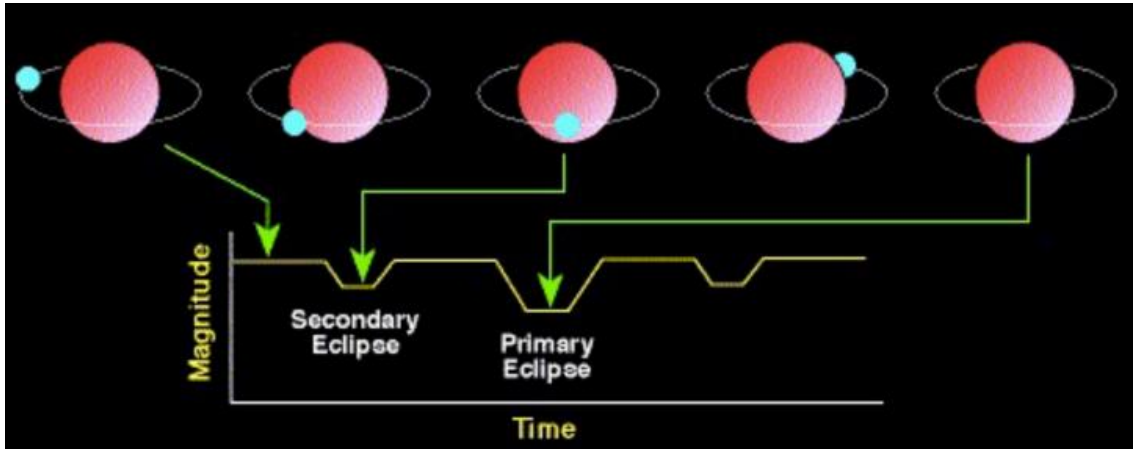
PJ Gibson, Mason Bushyeager, Joseph Korpela

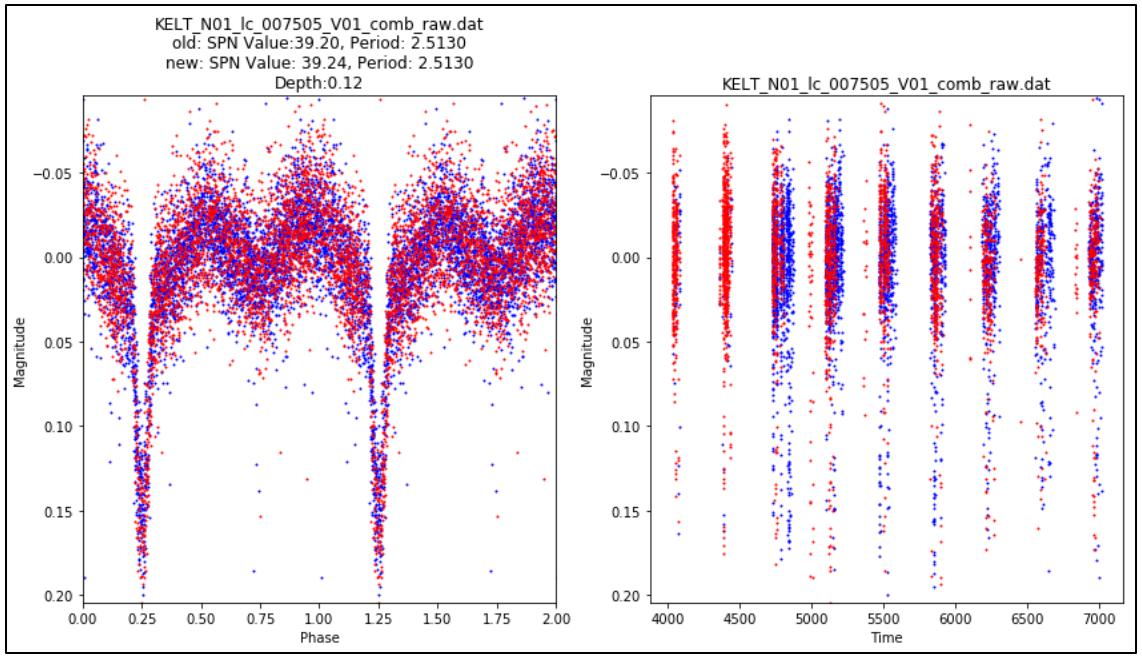
Presentation Outline

1. Purposes of Python
2. Data Analysis with Pandas
3. IDEs
4. Data Storage
5. Virtual Environments
6. Visualizations
7. Web-Scraping*
8. Machine Learning*
9. Learning Resources
10. Questions

What is a for-loop?







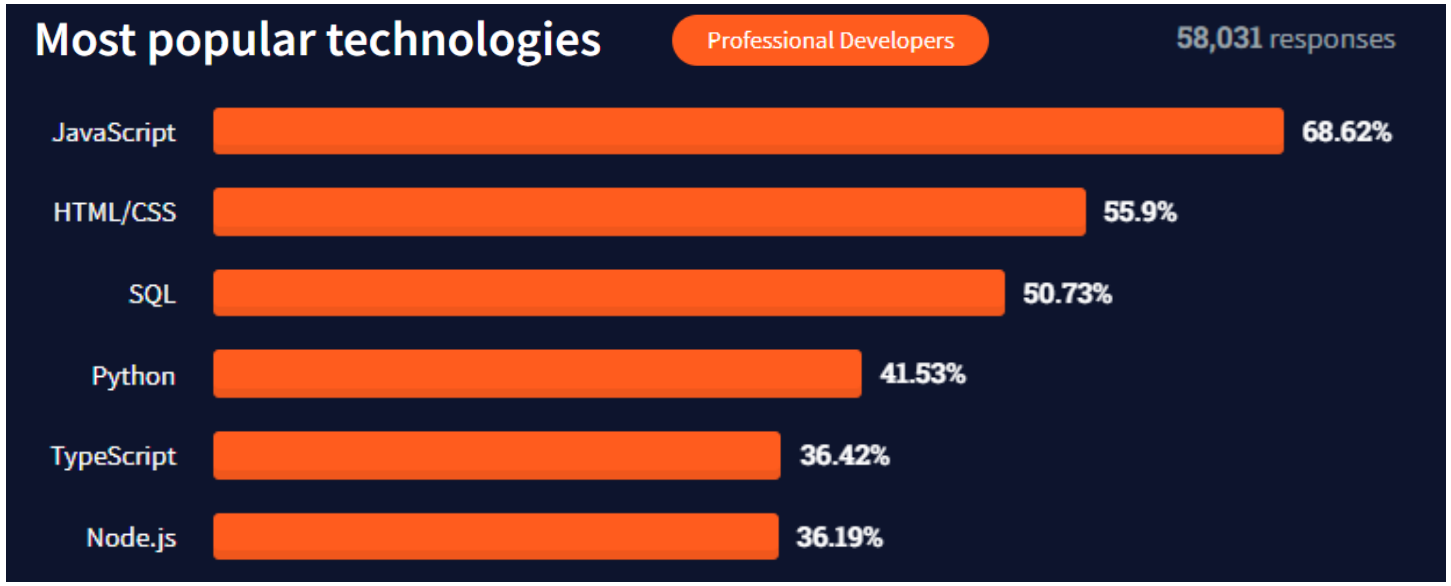
Purposes of Python

- General Purpose Language
- Data Analytics/Science
- Web Development
- Machine Learning
- Automation
- Parallel Compute

	Python	R	SAS
Open-Source	✓	✓	✗
Visualizations	3	1	2
Statistical Analysis	3	2	1
Community Support	1	2	3
Versatility	1	2	3
Ease-of-use	1	2	3

Python Selling points

- Changing field of data science
- Community
  - Stack Overflow - Professional Developers Survey 2021

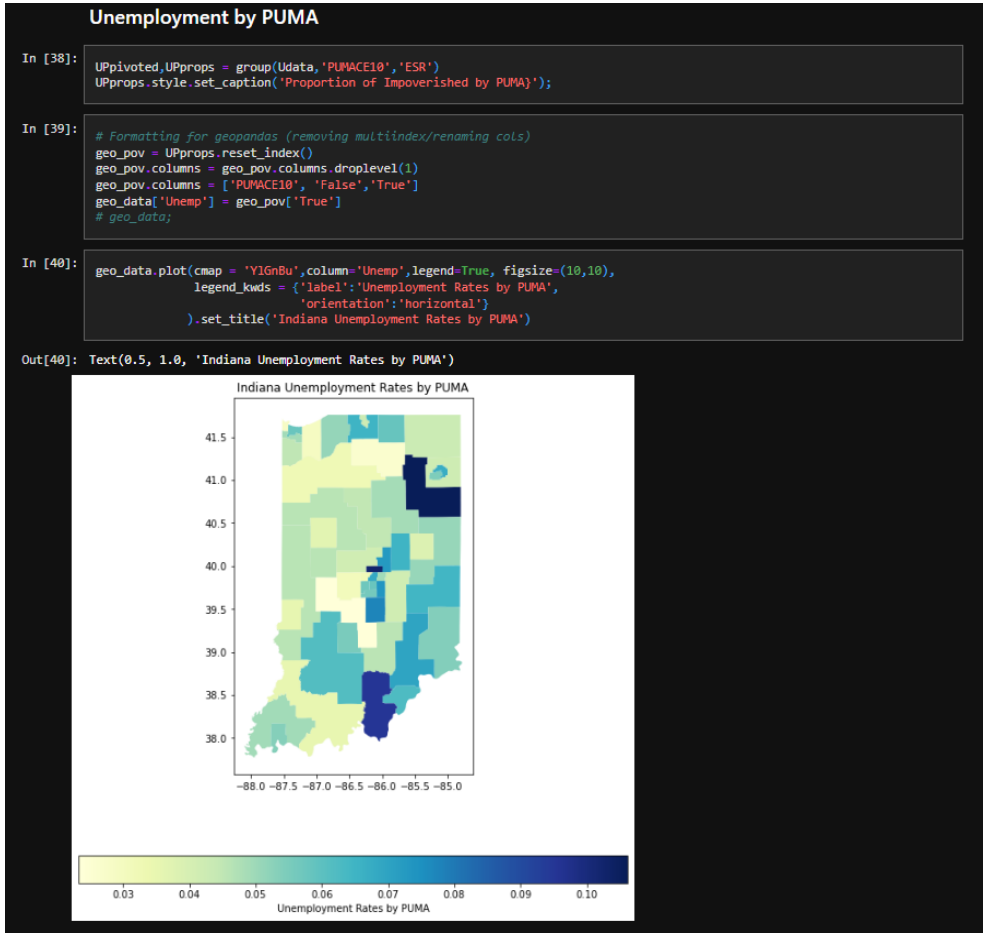


IDEs, Data Analysis, and Data Storage

Coding in Python

Three common methods

1. Scripts/CLI
2. (Jupyter) Notebooks
   - Documentable
   - Executable chunks of code
3. IDEs (e.g., VS Code or RStudio)
   - Debugging
   - Project Management
   - Versioning (Git)
   - Other extensions



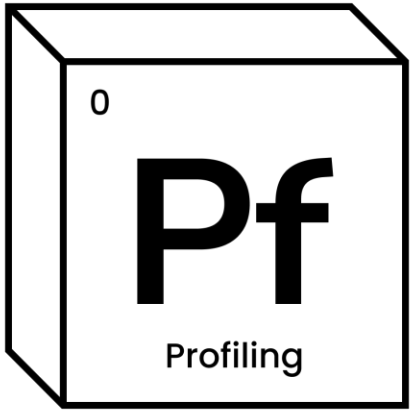
Data Analysis in Python

Two Primary Packages

- Pandas
- Polars

Data Profiling

- ydata-profiling



Sample Data

	PATIENT	DESCRIPTION	BASE_COST	REASONDESCRIPTION
0	487474a8-8433-4fa8-a12b-31f66b2d981a	amLODIPine 5 MG / Hydrochlorothiazide 12.5 MG ...	263.49	Hypertension
1	487474a8-8433-4fa8-a12b-31f66b2d981a	amLODIPine 5 MG / Hydrochlorothiazide 12.5 MG ...	263.49	Hypertension
2	487474a8-8433-4fa8-a12b-31f66b2d981a	amLODIPine 5 MG / Hydrochlorothiazide 12.5 MG ...	263.49	Hypertension
3	487474a8-8433-4fa8-a12b-31f66b2d981a	amLODIPine 5 MG / Hydrochlorothiazide 12.5 MG ...	263.49	Hypertension
4	487474a8-8433-4fa8-a12b-31f66b2d981a	amLODIPine 5 MG / Hydrochlorothiazide 12.5 MG ...	263.49	Hypertension

Fake patient medication data

VSCode and Pandas

```
# Selecting the patient column
df['PATIENT']
```

```
0       487474a8-8433-4fa8-a12b-31f66b2d981a
1       487474a8-8433-4fa8-a12b-31f66b2d981a
2       487474a8-8433-4fa8-a12b-31f66b2d981a
```

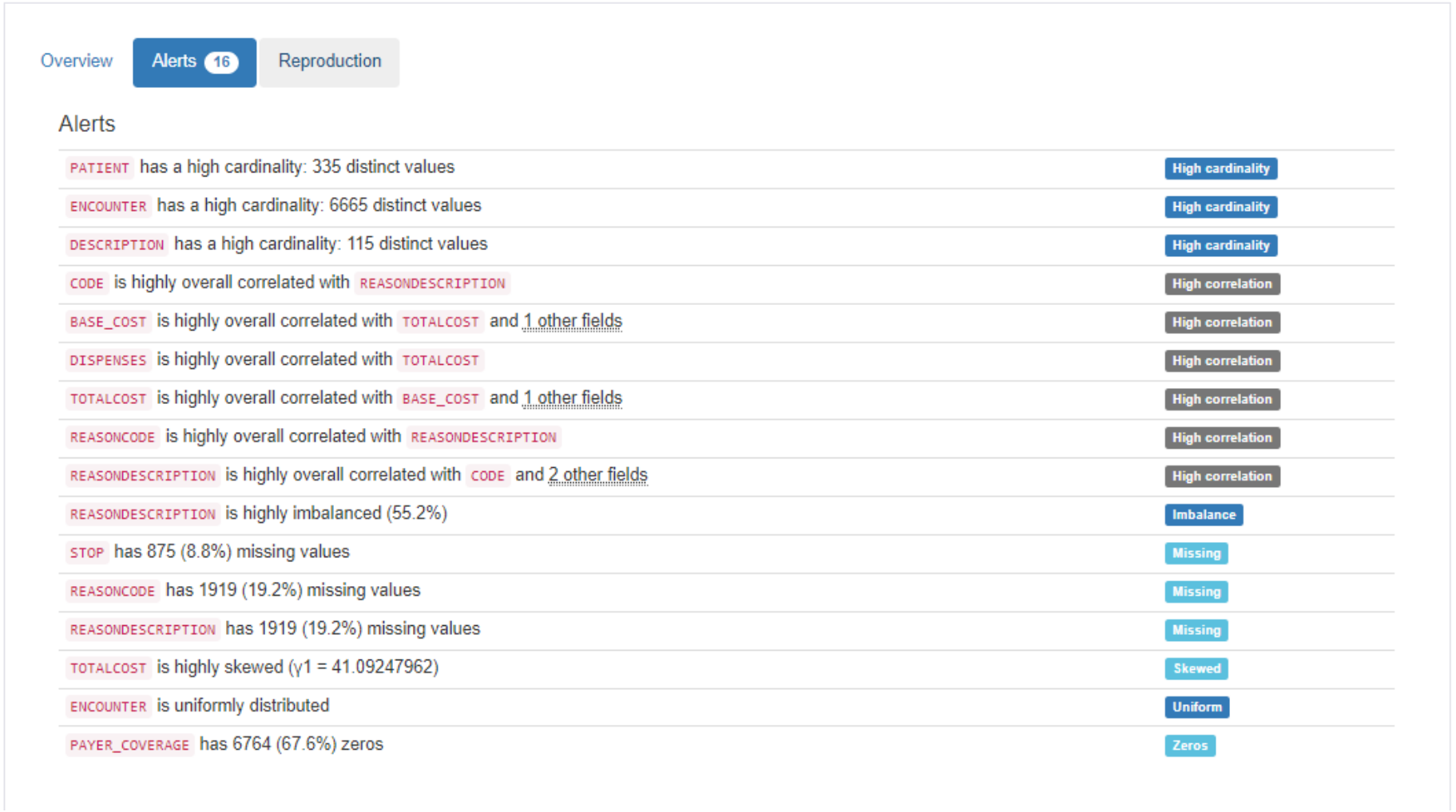
```
# Getting the count of people with a particular affliction and sorting by most frequent
df.where(df["REASONDESCRIPTION"].notnull()).groupby("REASONDESCRIPTION")["PATIENT"].count().sort_values(ascending=False)
```

```
REASONDESCRIPTION
Hypertension                                    3174
Diabetes                                        2732
```

```
# Demonstrating the use of pandasql and SQL in Pandas
sqldf('select REASONDESCRIPTION, count(PATIENT) from df where REASONDESCRIPTION IS NOT NULL GROUP BY REASONDESCRIPTION ORDER BY count(PATIENT) DESC')
```

	REASONDESCRIPTION	count(PATIENT)
0	Hypertension	3174
1	Diabetes	2732

ydata-profiling



Overview | Alerts 16 | Reproduction

Alerts

PATIENT has a high cardinality: 335 distinct values	High cardinality
ENCOUNTER has a high cardinality: 6665 distinct values	High cardinality
DESCRIPTION has a high cardinality: 115 distinct values	High cardinality
CODE is highly overall correlated with REASONDESCRIPTION	High correlation
BASE_COST is highly overall correlated with TOTALCOST and 1 other fields	High correlation
DISPENSES is highly overall correlated with TOTALCOST	High correlation
TOTALCOST is highly overall correlated with BASE_COST and 1 other fields	High correlation
REASONCODE is highly overall correlated with REASONDESCRIPTION	High correlation
REASONDESCRIPTION is highly overall correlated with CODE and 2 other fields	High correlation
REASONDESCRIPTION is highly imbalanced (55.2%)	Imbalance
STOP has 875 (8.8%) missing values	Missing
REASONCODE has 1919 (19.2%) missing values	Missing
REASONDESCRIPTION has 1919 (19.2%) missing values	Missing
TOTALCOST is highly skewed (γ1 = 41.09247962)	Skewed
ENCOUNTER is uniformly distributed	Uniform
PAYER_COVERAGE has 6764 (67.6%) zeros	Zeros

Data Access & Storage in Python

Files

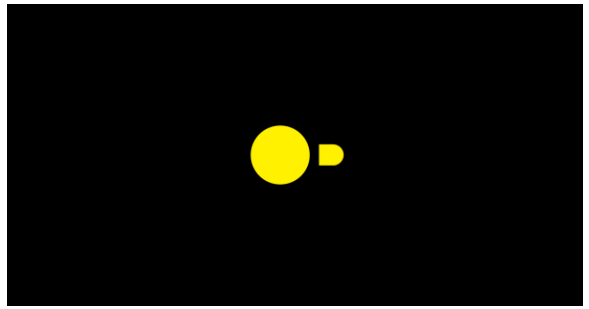
- CSV, JSON, Excel, etc...

Databases

- External
  - MySQL, MongoDB, etc...
- In-Memory
  - SQLite and DuckDB

APIs

- Twitter, Census, etc...



Virtual Environments and Data Visualization

Virtual Environments in Python

What are they?

- Isolate installed Python packages per project

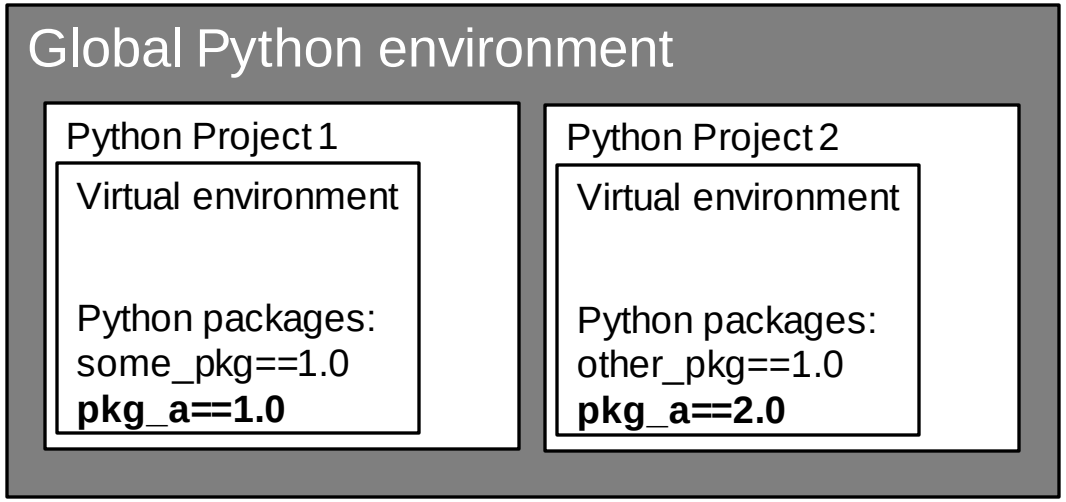
Why use them?

- Avoid dependency conflicts, example on right:
  - some_pkg 1.0 incompatible w/ pkg_a 2.0
  - other_pkg 1.0 incompatible w/ pkg_a 1.0
- Improve reproducibility

When should you use them?

- Always (one per software project)

Avoid Dependency Conflicts



Improve Reproducibility

```
# Export dependencies as text file
$ pip freeze > requirements.txt

# Import dependencies from text file
$ pip install -r requirements.txt
```

Virtual Environments: Examples


venv

- Python's built-in module for virtual environments
- Use virtual environments via CLI

Anaconda

- Scientific computing platform that includes Python
- Use virtual environments via GUI and CLI

Docker

- Encapsulates entire OS vs. just Python dependencies

venv CLI example

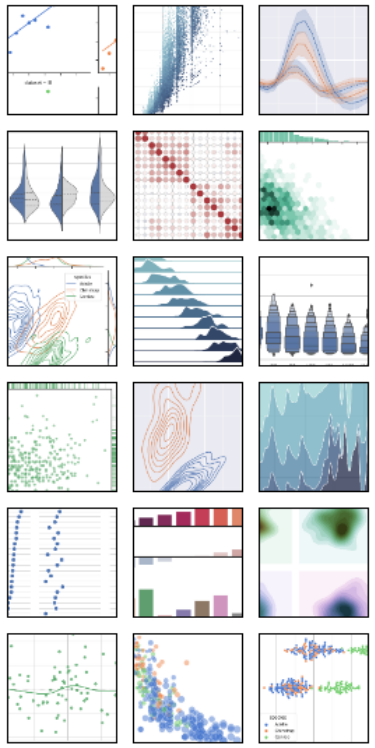
```
# Create new virtual environment
USER@pcname $ python3 -m venv example-env

# Active (switch on)
USER@pcname $ . example-env/Scripts/activate

# Deactivate (switch off)
(example-env) USER@pcname $ deactivate
```

Anaconda CLI example

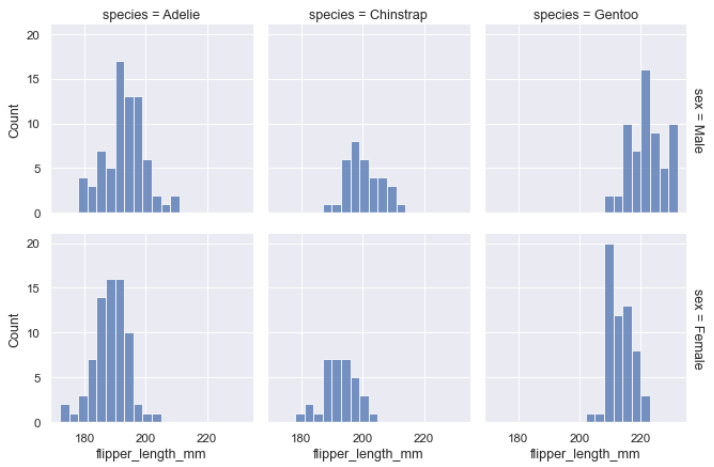
```
# Create new virtual environment
USER@pcname $ conda create --name example-env

# Activate (switch on)
USER@pcname $ conda activate example-env

# Deactivate (switch off)
(example-env) USER@pcname $ conda deactivate
```

Data Viz: Examples of Static Charts

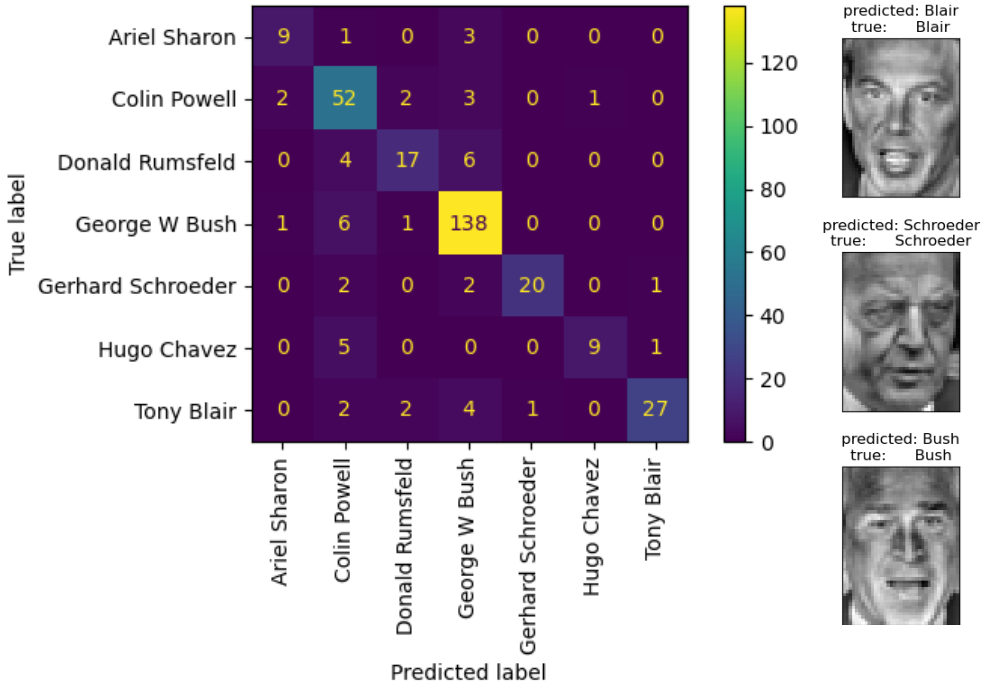
Quickly create various plots with **seaborn**



```
import seaborn as sns

sns.set_theme(style="darkgrid")
df = sns.load_dataset("penguins")
sns.displot(
    df, x="flipper_length_mm", col="species", row="sex",
    binwidth=3, height=3, facet_kws=dict(margin_titles=True),
)
```

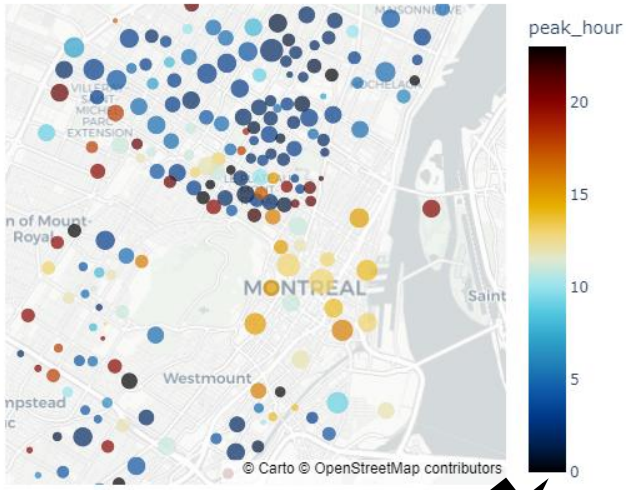


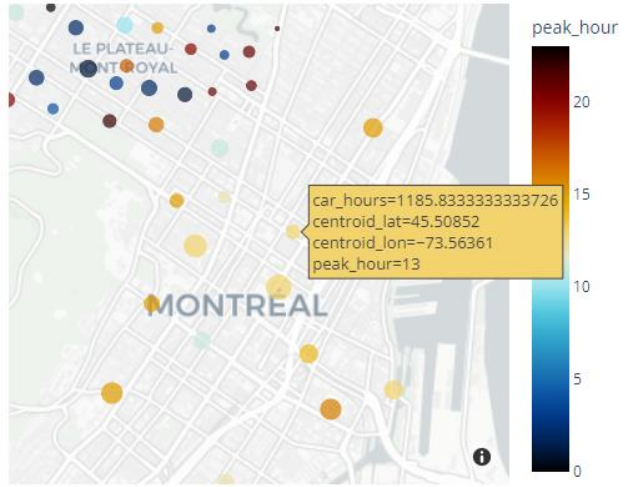
Visualize ML results with **matplotlib**



Data Viz: Interactive Charts w/ Plotly

```
import plotly.express as px
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length", color="species", symbol="species")
fig.show()
```





Data Viz: Dashboards w/ Shiny

Shiny for Python

Share

Password input

Outputs

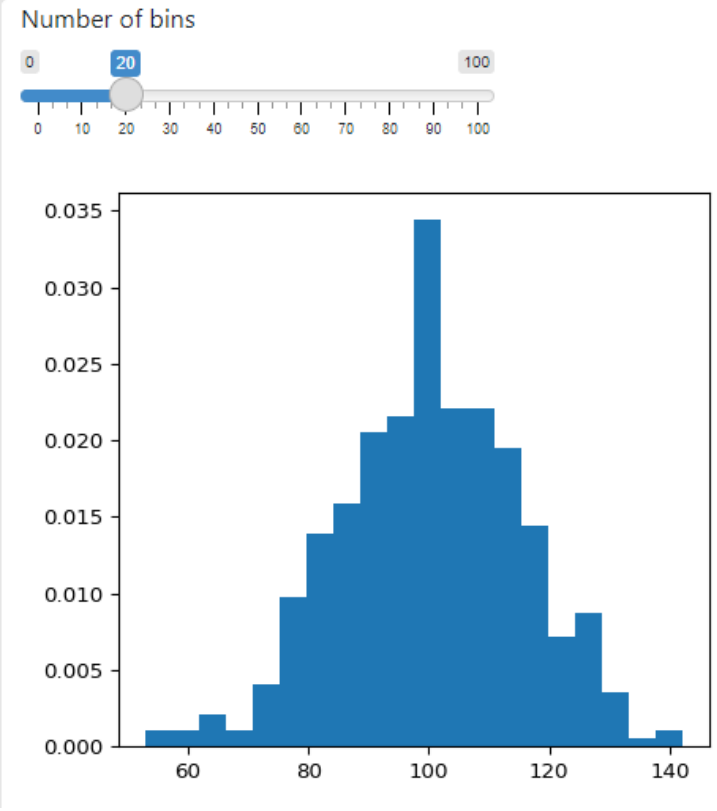
- Text output
- Verbatim text output
- UI output
- Plot output
- Table output

Layout

- Shinyswatch *Visual themes*
- Sidebar
- Two columns

app.py

```python
import matplotlib.pyplot as plt
import numpy as np
from shiny import App, render, ui

app_ui = ui.page_fluid(
    ui.input_slider("n", "Number of bins", 0, 100, 20),
    ui.output_plot("plot"),
)


def server(input, output, session):
    @output
    @render.plot(alt="A histogram")
    def plot():
        np.random.seed(19680801)
        x = 100 + 15 * np.random.randn(437)

        fig, ax = plt.subplots()
        ax.hist(x, input.n(), density=True)
        return fig


app = App(app_ui, server, debug=True)

```



Web-scraping

Libraries

Scrapy

Considerations

- Pre-built APIs
- Permissions
- Request volume
- Code flexibility

Web-Scraping Walkthrough

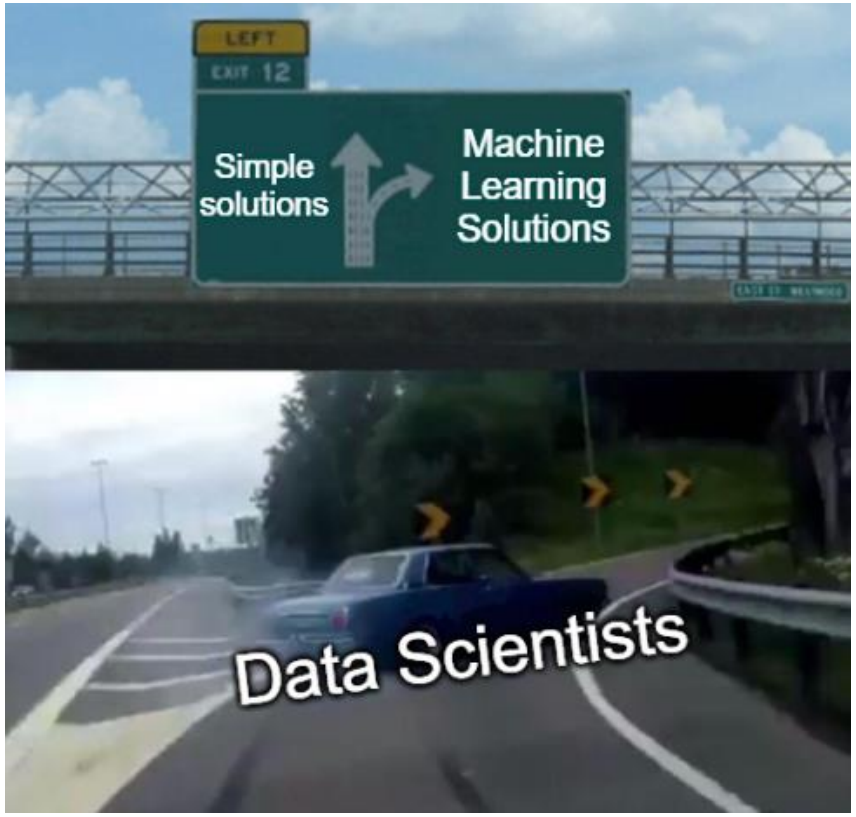
Github Notebook Link

Machine Learning

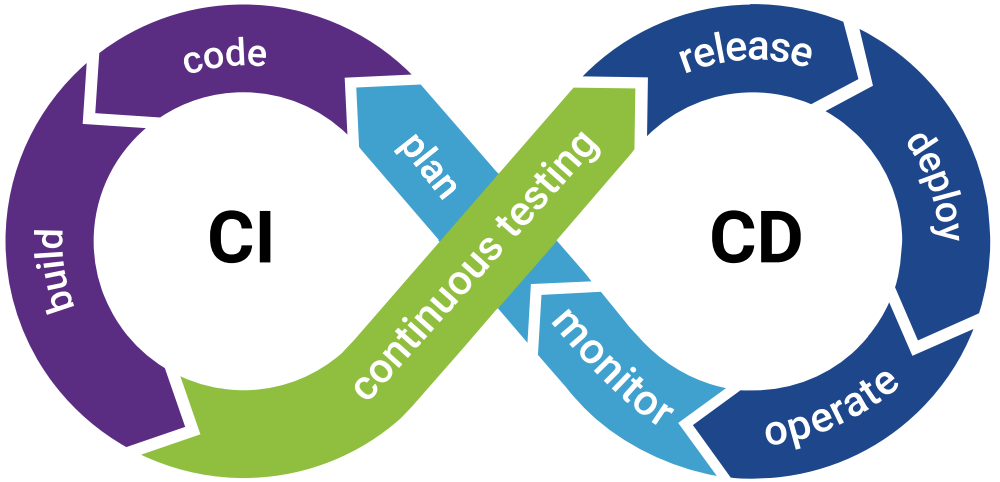
scikit learn

TensorFlow

PyTorch



Machine Learning (cont)



Machine Learning Application

- Record Linkage
- COVID: (-) ELRs ↔ (+) ELRs
- **Objective:**
  - ML model on par with human manual reviewers

Machine Learning Walkthrough

Github Notebook Link

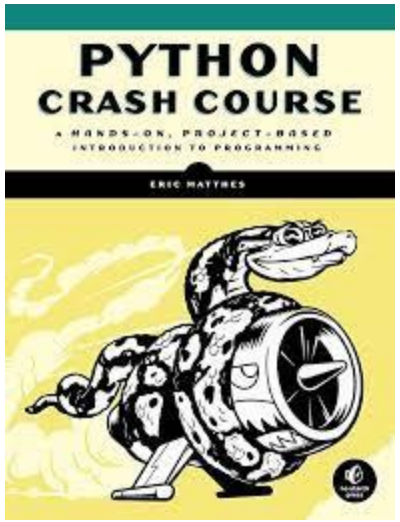
Learning Python

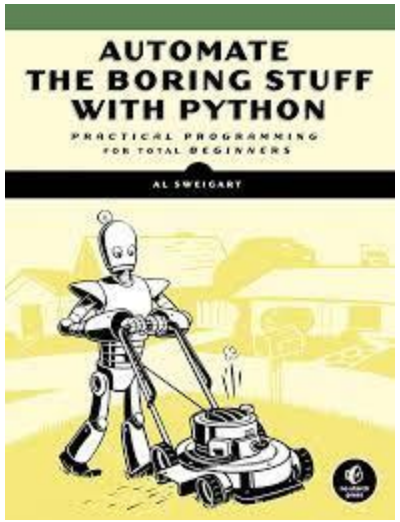
Early Learning

- Video Classes (free via audit)
  - edX – CS50 Introduction to Computer Science
  - Coursera – Programming for Everybody
- FreeCodeCamp
- Books
  - Python Crash Course – Eric Matthes
  - Automate the Boring Stuff – Al Sweigert

Continued Learning

- Kaggle
- DataCamp (paid)
- Projects



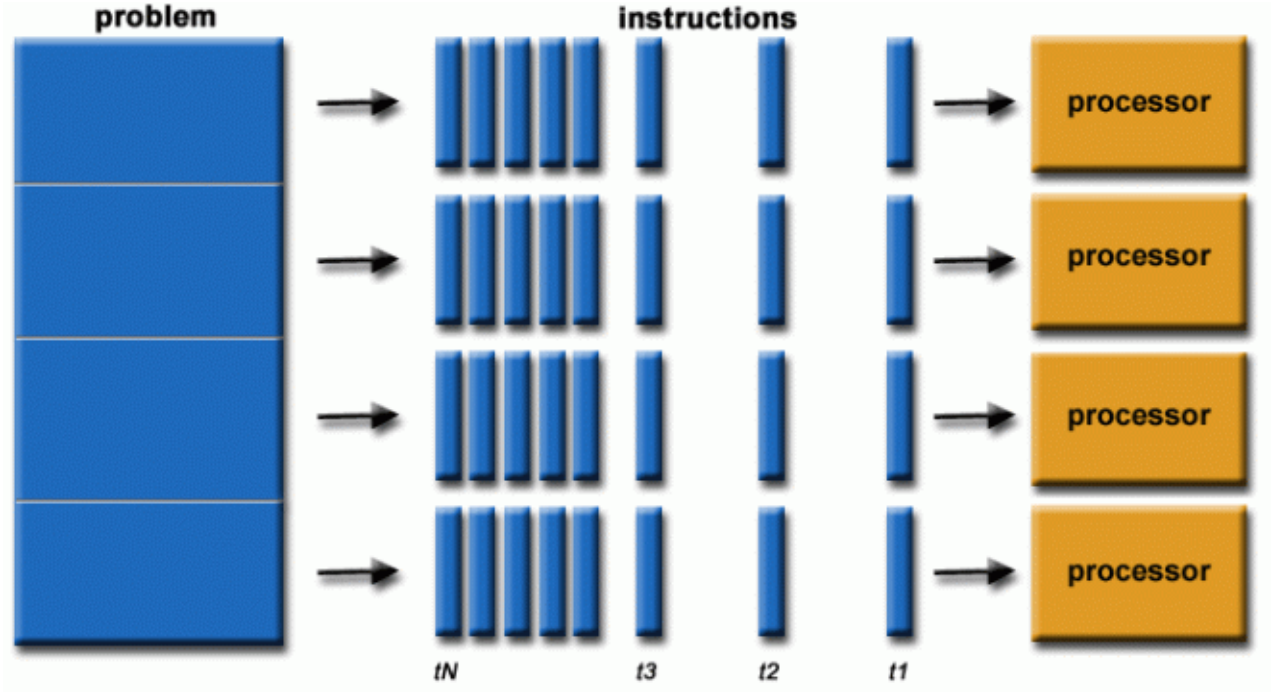


Questions?

Github Presentation Repo - **https://github.com/DOH-PJG1303/Applications-of-Python**

Scaling with Python

- Pyspark
- Dask
- Ray
- Joblib
- Multiprocessing
- Cloud Services



Scaling with Pyspark

Benefits

- Easy transition
- Easy to read
- No Databricks, no problem

databricks

Example

Negative Labs Process:

- Originally SAS, >10 hrs
- Now Python, <2 hrs

References (hyperlinked)

Slide 4: Numpy, Pandas, Django, Scikit-Learn, PySpark, TensorFlow
Slide 6: ChatGPT, Stack Overflow Survey
Slide 8: Jupyter, VSCode, RStudio, Git, GitHub
Slide 9: Pandas, Polars, ydata-profiling
Slide 10: Synthea Data
Slide 11: PandasSQL
Slide 13: Excel, MySQL, MongoDB, DuckDB, SQLite, Twitter, Census
Slide 16: Python, Anaconda, Docker
Slide 17: Seaborn, Scikit-learn
Slide 18: Plotly
Slide 19: Shiny for Python
Slide 20: Requests, Selenium, Scrapy
Slide 22: Scikit-Learn, TensorFlow, PyTorch, Meme Generator
Slide 23: Garbage-in-garbage-out image, CICD image
Slide 24: RL Blocking paper
Slide 26: edX Course, Coursera Course, FreeCodeCamp, Automate the Boring Stuff, Python Crash Course, Kaggle, DataCamp
Slide 28: Parallel Computing
Slide 29: Databricks