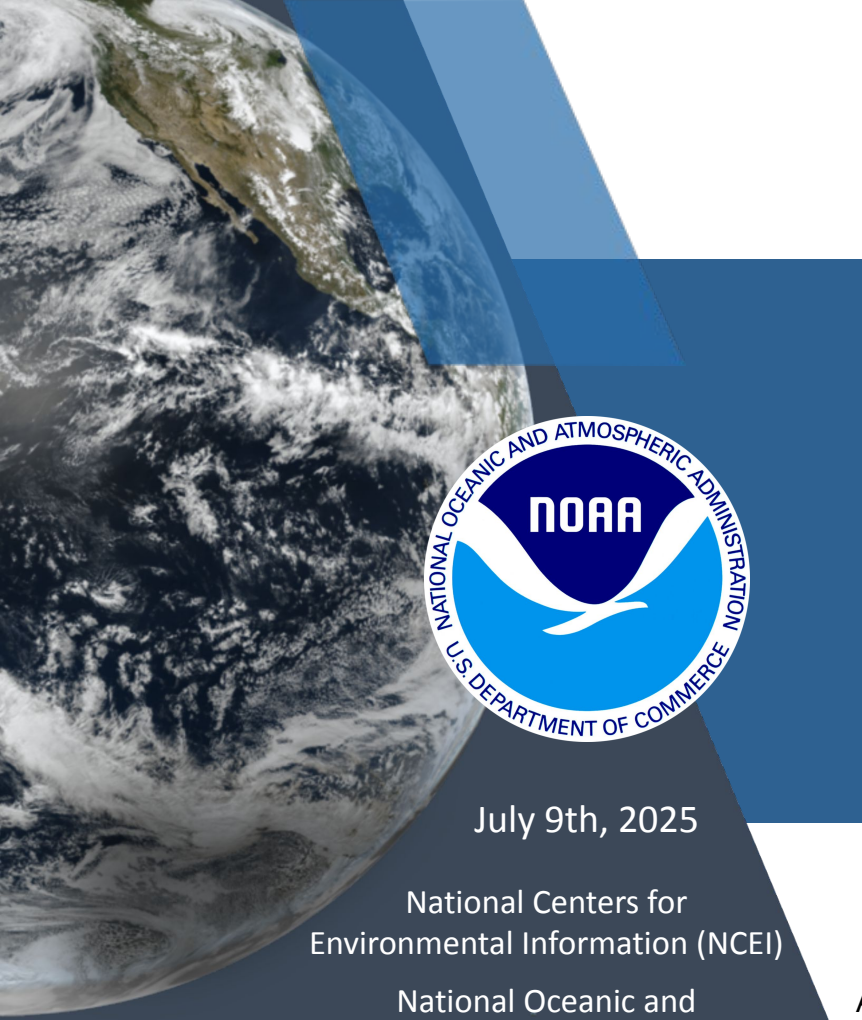


# Breaking Out of the Loop:

# Refactoring Legacy Software with Polars

*Teaching & Learning, Earth Science*

July 9th, 2025

National Centers for Environmental Information (NCEI)

National Oceanic and Atmospheric Administration (NOAA)

Author: Brodie Vidrine

# Agenda

1. About the Presenter
2. Who is this Talk For?
3. Objective
4. Legacy Code: Global Summary of the Month
5. Meeting Requirements with Python and Polars (Demo)
6. Recommend Reading & “Thanks!”

# About the Presenter

***Brodie Vidrine, Science & Technology Corporation***

- Federal Contractor for NOAA's National Centers for Environmental Information (NCEI)
- Hired 1 year ago as part of an NCEI code modernization initiative
- Background:  10+ years working at Ascend Math, a Java-based cloud application
- Team lead on NCEI's Global Summary of the Month (GSOM)

# Meet the Team

- **Chilling w/Polars** - Dr. Laura McBride, STC

  Posters, Wednesday 6-7pm

- **Beautiful PDFs w Pandas** - Tad Thurston, Riverside

  Posters, Wednesday, 6-7pm

- **From Legacy to Leading-Edge:**

  **Revamping NCEI Software for the Cloud Era** - Sarah Purpura, Riverside

  Room 315, Friday 11:25

# Who is this talk for?

- You're proficient in Python
- You're new to (or frustrated by) Polars or Panda
- You've got experience with object oriented programming
- You're not necessarily a database wizard or spreadsheet guru
- You're more interested in crunching data than analyzing it
- You have business requirements that are "impossible" to meet with Polars.

# Objective

It’s my hope that at the end of this talk you’ll walk out of here with a new set of strategies for tackling business requirements using native Polars expressions. (Of which, there are many Pandas equivalents.) If your requirements are STILL too complicated, I’ll show you how to create a custom JIT compiled function using Numba’s @guvectorize decorator.

# Global Summary of the Month

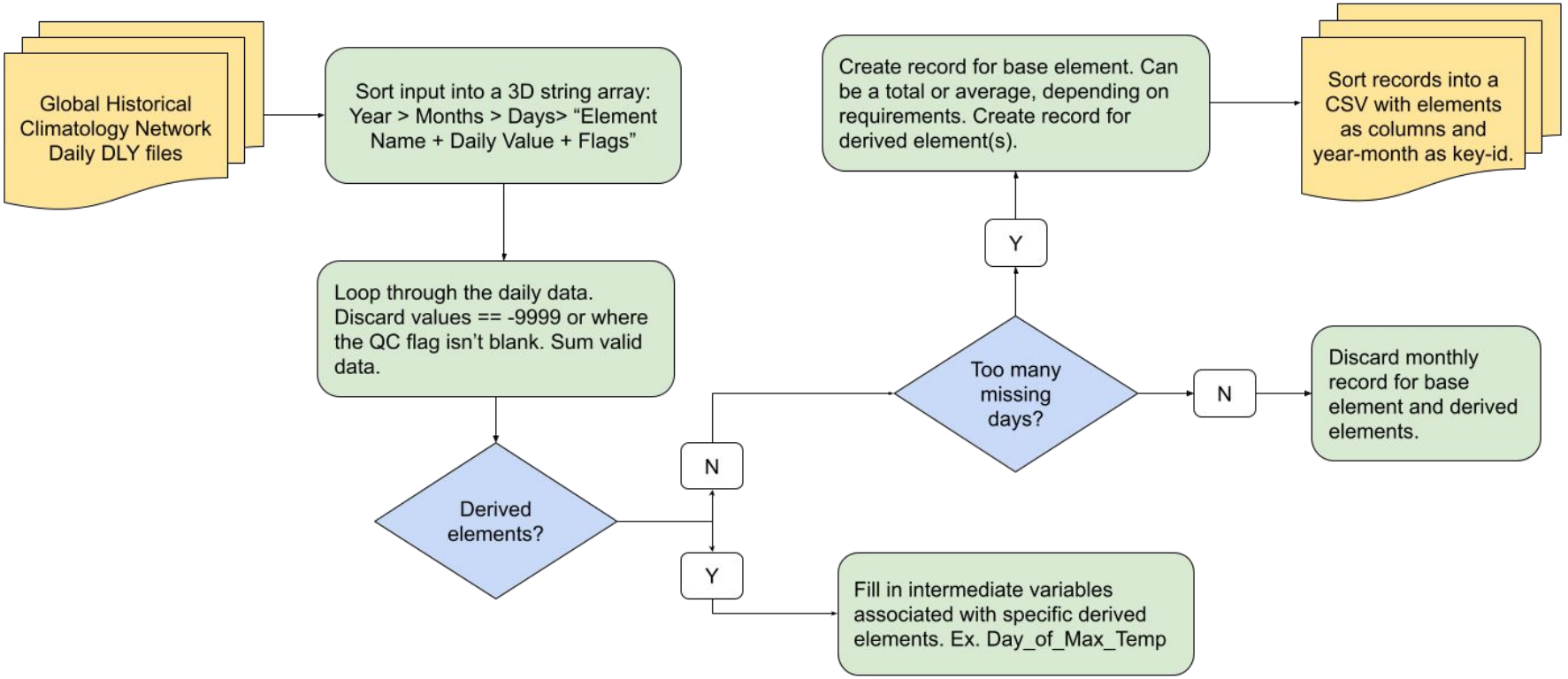
The Global Summary of the Month pulls daily data from over 129,000 land-surface weather stations from around the world and produces a monthly summary in CSV for each one.

You can browse or search through the CSVs over on NCEI's website. Each CSV can have up to 63 different weather elements. (Temperature Max, Total Precipitation in Centimeters, Direction of Fastest 5 Second Wind Gust, etc)

# Plan A

- Reduce 12,000 lines of Java to 2,000
- Remove obsolete code (discontinued features and elements)
- Integrate 8 new columns from NOAA's ASOS dataset
- Refactor using modern best practices
    - Ie, Slice up the 4 big ugly functions into a proper object-oriented program.
    - Use meaningful var names, docstrings, judicious commenting
- Set up CI/CD

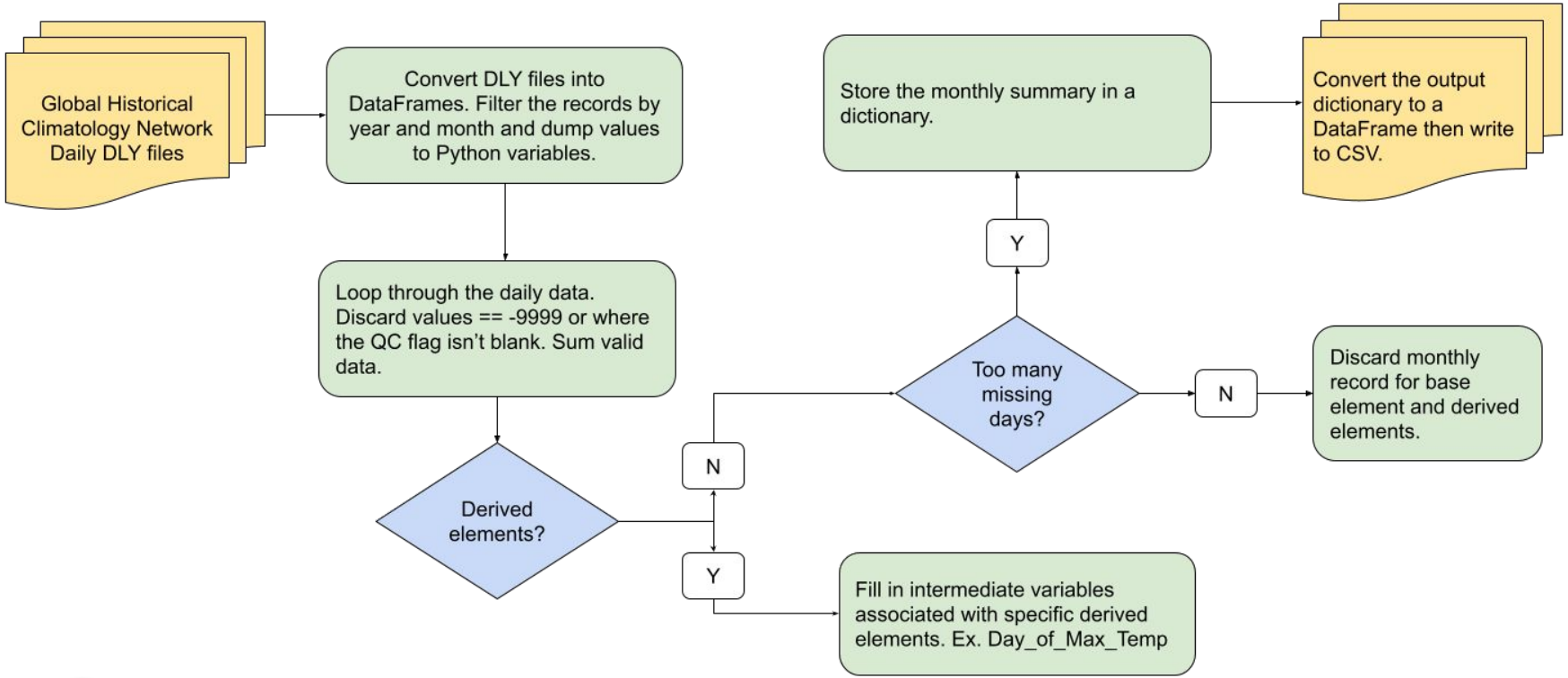
# Plan A - Flow


Global Historical Climatology Network Daily DLY files
Sort input into a 3D string array: Year > Months > Days> "Element Name + Daily Value + Flags"
Loop through the daily data. Discard values == -9999 or where the QC flag isn't blank. Sum valid data.
Derived elements?
N
Y
Fill in intermediate variables associated with specific derived elements. Ex. Day_of_Max_Temp
Too many missing days?
N
Discard monthly record for base element and derived elements.
Y
Create record for base element. Can be a total or average, depending on requirements. Create record for derived element(s).
Sort records into a CSV with elements as columns and year-month as key-id.

# Plan B

- Convert 3,000 lines of Java to Python
- Integrate Polars. That'll make it fast!
- Make sure climate scientists can read and maintain the code after the contract is up.

# Plan B - Flow


Global Historical Climatology Network Daily DLY files
Convert DLY files into DataFrames. Filter the records by year and month and dump values to Python variables.
Loop through the daily data. Discard values == -9999 or where the QC flag isn't blank. Sum valid data.
Derived elements?
N
Y
Fill in intermediate variables associated with specific derived elements. Ex. Day_of_Max_Temp
Too many missing days?
N
Discard monthly record for base element and derived elements.
Y
Store the monthly summary in a dictionary.
Convert the output dictionary to a DataFrame then write to CSV.

# Business Requirements

- Pull data from NCEI DLYs (flat files)
- Validate every day by confirming value != -9999 and QC flag ==””.
- Remove records where too many days are missed.
- Create derived elements from base elements. Ex. Hottest Day of the Month from Max Temperature.
- Create monthly summary required for element (average, total, etc)
- Create a CSV with weather elements as columns and year-month as the key index.
- Include an attribute column for each element including a comma separated string with flags like “days missed” and “trace reading”.

# Plan Z

By rethinking business requirements in Polars, we reduced processing time from 24 hours (current run time on prod server) to 3 hours (running on my Dell Inspiron). Today, I'd like to demo Plan Z for you:

# Recommend Reading & Thanks!

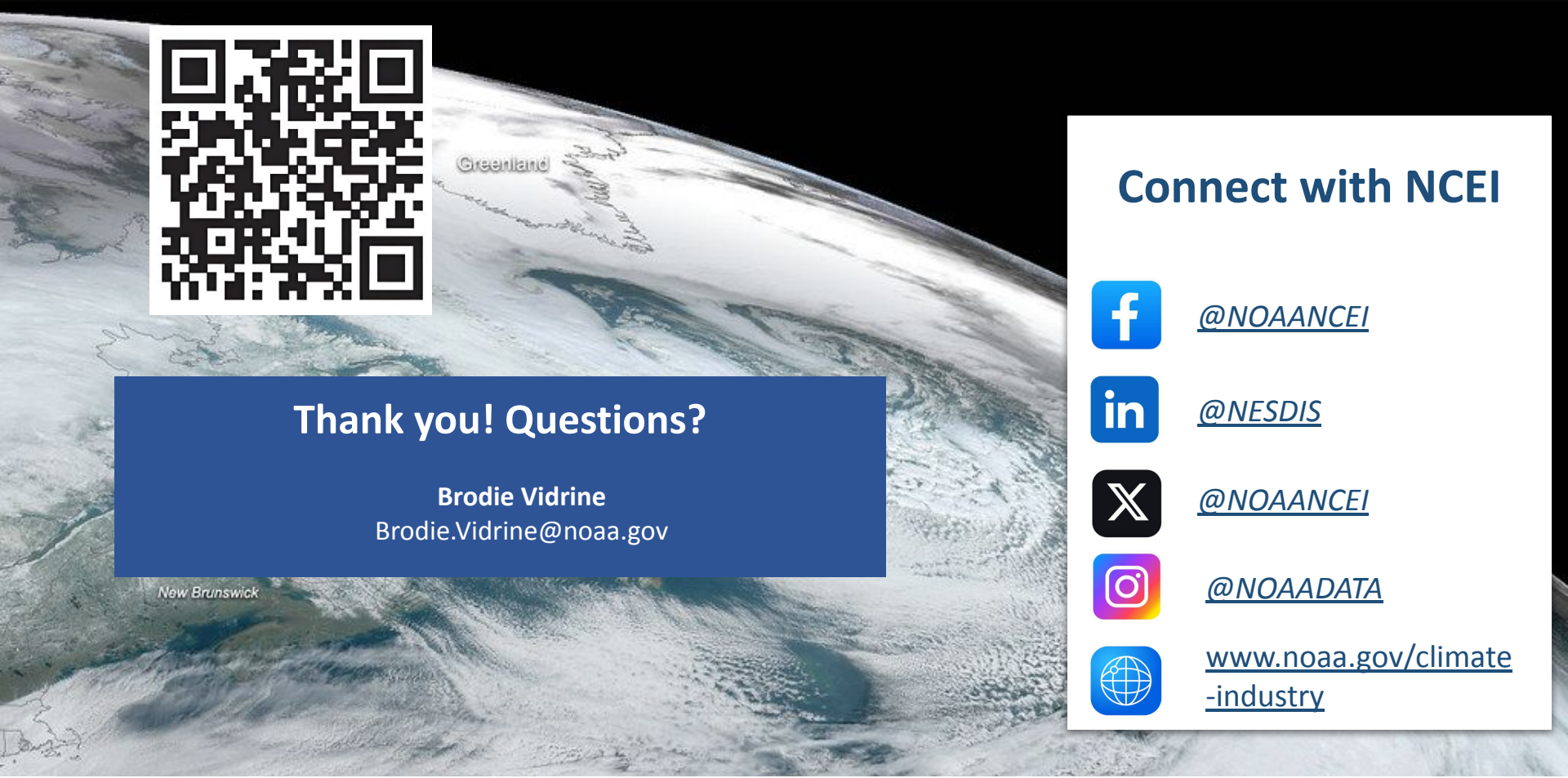
*There's this weird phenomenon where people write data programming code as if they hate themselves. - Kevin Heavey*

Modern Polars - Kevin Heavey

Search YouTube for Matt Harrison on Idiomatic Pandas

Questions about anything from the presentation?

Email me: brodie.vidrine@noaa.gov




## Thank you! Questions?

**Brodie Vidrine**
Brodie.Vidrine@noaa.gov

## Connect with NCEI

- *@NOAANCEI*
- *@NESDIS*
- *@NOAANCEI*
- *@NOAADATA*
- www.noaa.gov/climate-industry

RIVERSIDE CADMUS STC CASE AECOM isciences Deloitte. BLN24