Hamilton: Scalable, Portable, and Self-Documenting Dataflows in Python

PRESENTER:
Stefan Krawczyk

BACKGROUND: Research code is notorious for poor software craftsmanship. Multi-year projects regularly mean long lived code that passes through many hands, accumulating technical debt that causes more grant money than necessary to be spent fighting code bases.

Is there a better way?

Hamilton Paradigm
Instead of
```
df['col_c'] = df['col_a'] + df['col_b']
```
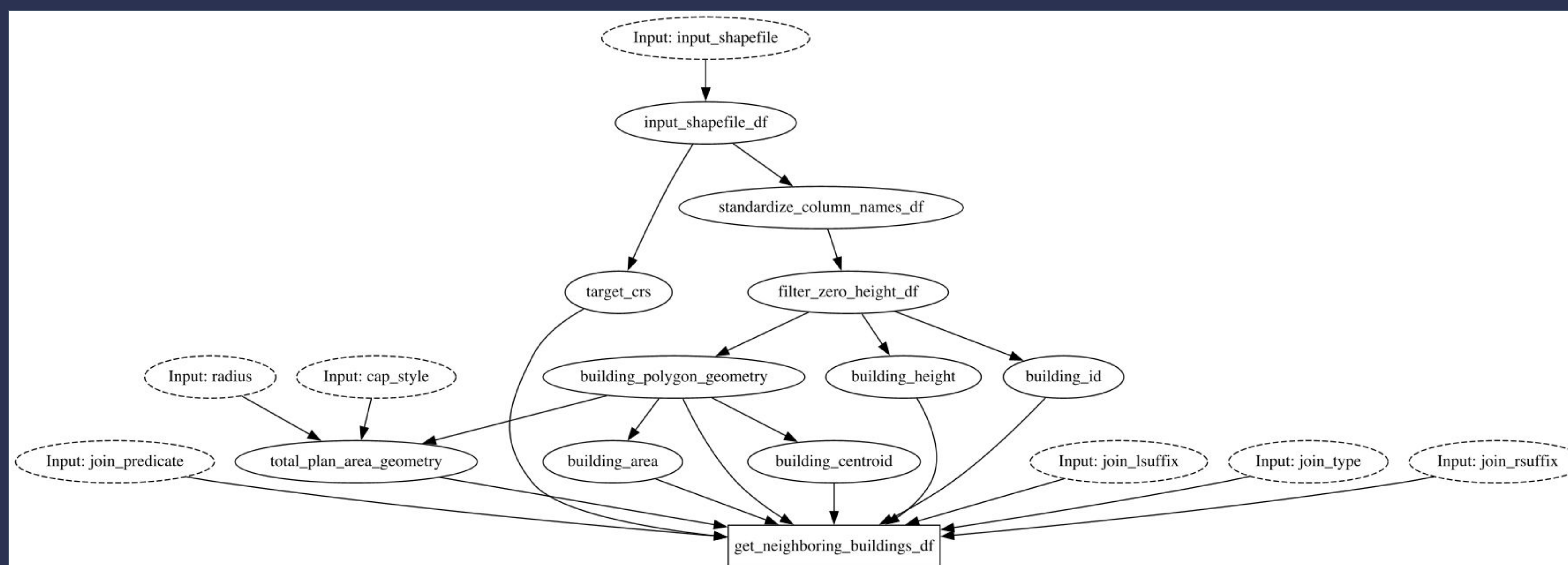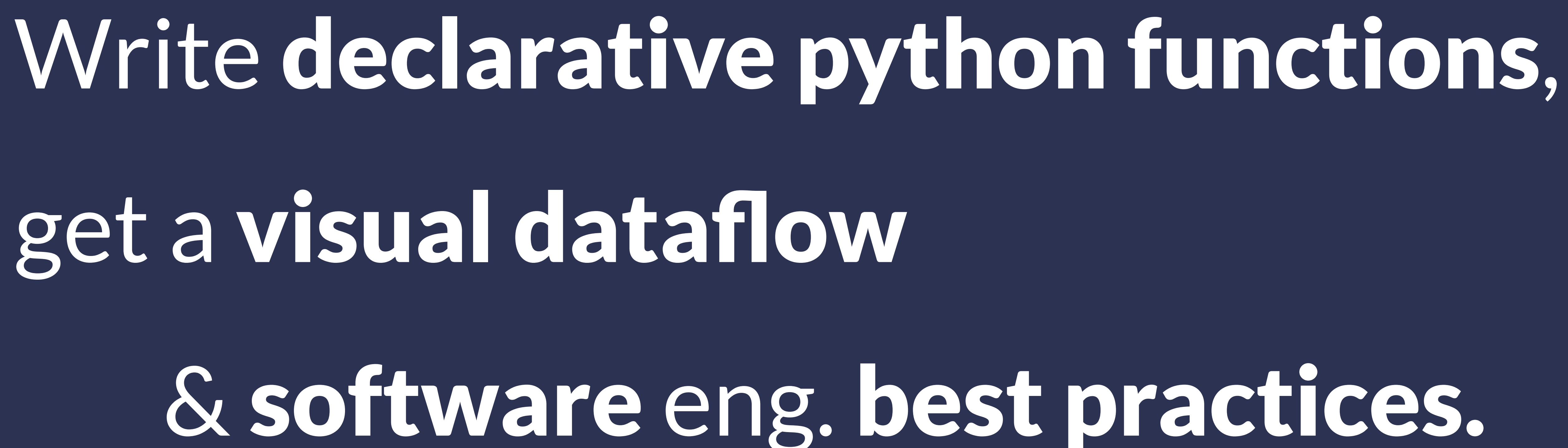### Write
```
def col_c(col_a: pd.Series,
          col_b: pd.Series) -> pd.Series:
    """Place for documentation"""
    return col_a + col_b
```
Process
1. Ditch writing procedural code in scripts.
2. Write declarative python functions.
3. Functions encode a dataflow via the way they are written.
4. Write "Drivers" that orchestrate python function execution.

Benefits
- "*Much easier to collaborate*" - Emily R.
- Researchers gain a standardized way to write code that enables collaborators to quickly ramp up and modify the project.
- Code is always:
  - unit testable
  - documentation friendly
  - visualizable
  - modified in a prescribed way
  - portable to other python contexts

Write **declarative python functions**, get a **visual dataflow** & **software** eng. **best practices.**

Functions encode lineage, i.e. a dataflow, that Hamilton can visualize for you.

Before → **After**

Scan for links

Before Hamilton
Worst Function: F (102)
Average complexity: C (19.64)
Test coverage: <5%

After Hamilton
Worst Function: B (7)
Average complexity: A (1.40)
Test Coverage: ~80%

Metrics were computed using https://github.com/rubik/radon & pytest

Hamilton Extras
- Lightweight data quality support
- Integrations for Ray, Dask, Pandas on Spark, PySpark.
- Ability to augment DAG Execution

Curious?
See: **www.tryhamilton.dev**

- Stefan Krawczyk   (DAGWorks)
  Elijah ben Izzy   (DAGWorks)
  Levi Sweet-Breu   (ORNL)
  Emily Rexer   (PNNL)
  Chris Vernon   (PNNL)
  Melissa Allen-Dumas (ORNL)