Institute
and Faculty
of Actuaries

IFoA Life Conference

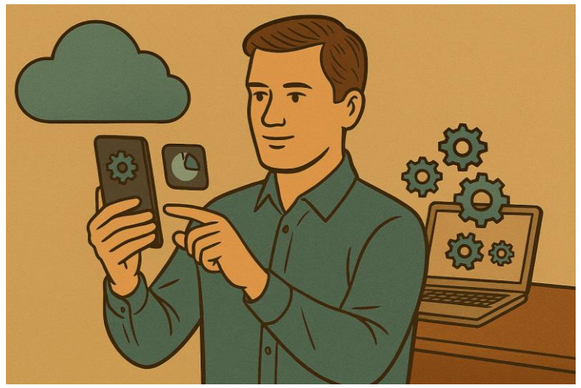
D7: Actuarial Cashflow Modelling with Python

Agenda

- **Culture**: Leading and influencing innovation
- **Open Source**: Introducing the heavylight python package
- **Idea to Implementation**: Our Journey within LBG

Setting the stage for success

Pace of change



?

1950

1990

2010

2020

2025

Building a change culture

Culture

Shared purpose
Trusting, safe environment
Regular challenge

Maximising Engagement

Setting guardrails for empowerment

Driving Output


DREAMWORK
STORY TELLING
SPRINTS
ME TO WE
GROWTH MINDSET
CLEAR PURPOSSE
TOGETHER WE ACHIEVE MORE

CHALLENGES TO EMBEDDING CULTURE

Python: the heavylight framework

Why Python?

Great language for everything else in the tech stack

- Data Processing
- Machine Learning (linear algebra)
- Process automation & pipelines
- Excel interaction
- API serving & distributed computing
- Version Control

Challenges:

- Out of box numerical performance poor
- No simple frameworks to deliver robust models

What makes actuarial models different?

Feature	
Assumption Tables	Actuarial models typically have 10+ tables covering economic assumptions, demographic assumptions, product parameters and related contracts like reinsurance.
Recursive Nature	All but the simplest actuarial models have a time dependency (time t depends on time t-1)
Explainability	Models are imperfect. Practitioners need to be able to understand and analyze the model.

+ performant

The heavylight Platform

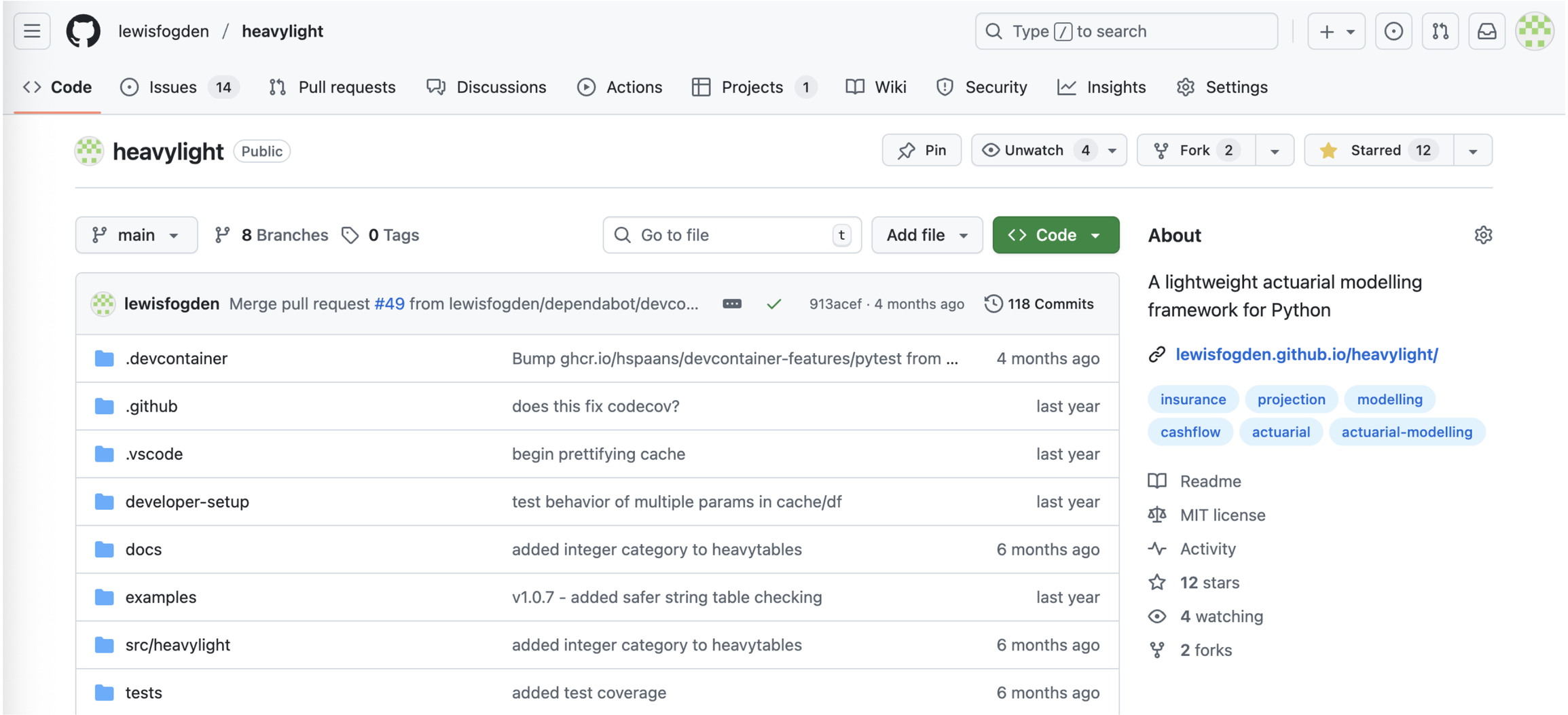
Provides:

- **Model** class for defining models & controlling execution.
- **Table** class: vectorized tables, handles multiple keys, including ranges.
- **Example** models with Excel spreadsheet replica.
- Direct export to pandas dataframes & Excel (`.df`).
- Documentation! https://lewisfogden.github.io/heavylight/

Benefits:

- Free: No usage restrictions (MIT licence)
- Open Source: install from pypi, extensible
- Lightweight: sub 1k lines of code.
- Tested: 88% code coverage

Open Source: Your invitation to Use / Steal / Improve



Sample Model Walkthrough

Heavylight Model Walkthrough

Heavylight includes an example model implementation in both Python and Excel.

Caveat: This model was defined primarily to allow the Actuarial Open Source group to write comparable models in different programming languages & packages for methodology and performance benchmarking.

https://lewisfogden.github.io/heavylight/#loading-from-templates-examples

Model Walkthrough: Model Definition (~100 L.O.C.)

```python
import heavylight
import numpy as np

class TermAssurance(heavylight.Model):
    def t(self, t):
        return t

    def net_cf(self, t):
        return self.premiums(t) - self.claims(t) - self.expenses(t)
```

```python
    def q_x_12(self, t):
        return 1 - (1 - self.q_x_rated(t))**(1/12)

    def q_x(self, t):
        return self.basis["mort_table"][self.age(t), self.duration(t), self.data["smoker_status"]]
```

Source: https://github.com/lewisfogden/heavylight/blob/main/examples/std_model_numpy/protection_model.py

Model Walkthrough: Define The Basis

```
from protection_model import TermAssurance, Basis, make_test_data
```

```
basis = Basis(
        cost_inflation_pa=0.03,
        initial_expense=200,
        expense_pp= 20,
        lapse_rate_pa=0.1,
        mort_table=heavylight.Table.read_csv(r"basis/q_x_generic.csv"),
        forward_rates=heavylight.Table.read_csv(r"basis/forward_rates.csv"),
        )
```

```
basis.mort_table.df.tail()
```

	age|int_bound	duration|int_bound	smoker_status|str	q_x|float
871	86	5	S	0.058366
872	87	5	S	0.065550
873	88	5	S	0.073738
874	89	5	S	0.083082
875	90	5	S	0.093763

Model Walkthrough: Generate Test Data (10k pols)

```
data = make_test_data(10000) # pick a number between 0 and 50000
df_data = pd.DataFrame(data.__dict__)
df_data
```

	policy_id	shape	sum_assured	annual_premium	term_y	init_pols_if	age_at_entry	smoker_status	extra_mortality
0	1	decreasing	225967.692715	4431.637748	11	1.0	21	N	0.291420
1	2	level	666298.423555	1592.564020	29	1.0	26	S	0.120836
2	3	level	769192.905100	3903.994207	20	1.0	34	S	0.107373
3	4	decreasing	174969.781598	9757.793687	29	1.0	28	S	0.009343
4	5	decreasing	53419.700584	6124.362583	18	1.0	23	S	0.978356
...	...	...	...	...	...	...	...	...	...
9995	9996	decreasing	876231.345685	513.074395	22	1.0	29	N	0.812249
9996	9997	level	584437.739989	4998.904289	18	1.0	38	N	0.463928
9997	9998	decreasing	122135.493639	6836.321887	29	1.0	25	S	0.985490
9998	9999	level	310710.747667	2262.765133	22	1.0	24	S	0.574462
9999	10000	level	858452.776413	7822.344793	13	1.0	25	N	0.937209

Model Walkthrough: Run the Model

```
# run the model
model_result = TermAssurance(data = data, basis = basis, proj_len=12*31)
```

```
# copy data and append the result as final column
df_result = df_data.copy()
df_result["pv_net_cf_t0"] = model_result.pv_net_cf_t0()
df_result.tail(5)
```

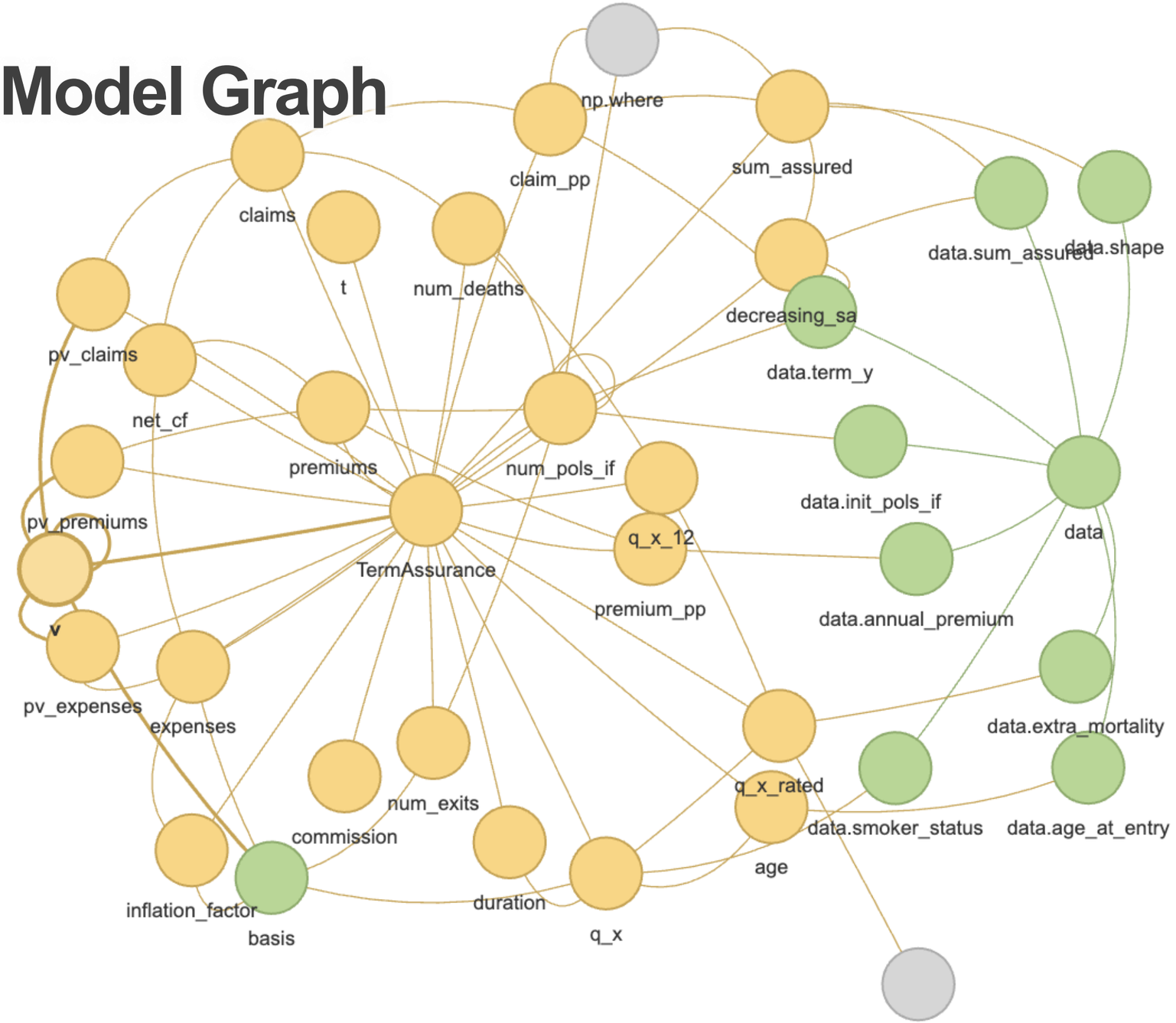
	policy_id	shape	sum_assured	annual_premium	term_y	init_pols_if	age_at_entry	smoker_status	extra_mortality	pv_net_cf_t0
9995	9996	decreasing	876231.345685	513.074395	22	1.0	29	N	0.812249	-1501.390276
9996	9997	level	584437.739989	4998.904289	18	1.0	38	N	0.463928	31878.435497
9997	9998	decreasing	122135.493639	6836.321887	29	1.0	25	S	0.985490	53826.123164
9998	9999	level	310710.747667	2262.765133	22	1.0	24	S	0.574462	13796.082226
9999	10000	level	858452.776413	7822.344793	13	1.0	25	N	0.937209	47522.047118

Model Walkthrough: View / Export Cashflows

```
df_index(model_result, 0) # to_clipboard() / to_excel() etc.
```

	t	net_cf	pv_net_cf	premium_pp	sum_assured	decreasing_sa	claim_pp	inflation_factor	v	premiums	...	expenses
0	0	163.345840	163.345840	369.303146	2.259677e+05	2.259677e+05	225967.692715	1.000000	1.000000	369.303146	...	200.000000
1	1	358.538199	358.372452	369.303146	2.248113e+05	2.248113e+05	224811.289647	1.000000	0.999538	366.065105	...	1.652053
2	2	355.424669	355.096129	369.303146	2.236483e+05	2.236483e+05	223648.348108	1.000000	0.999076	362.855456	...	1.637568
3	3	352.338343	351.849926	369.303146	2.224788e+05	2.224788e+05	222478.831128	1.000000	0.998614	359.673948	...	1.623210
4	4	349.278983	348.633564	369.303146	2.213027e+05	2.213027e+05	221302.701530	1.000000	0.998152	356.520336	...	1.608978
...	...	...	...	...	...	...	...	...	...	...	...	...
367	367	0.000000	0.000000	369.303146	-1.189066e+06	-1.189066e+06	0.000000	2.427262	0.664708	0.000000	...	0.000000
368	368	0.000000	0.000000	369.303146	-1.198223e+06	-1.198223e+06	0.000000	2.427262	0.664086	0.000000	...	0.000000
369	369	0.000000	0.000000	369.303146	-1.207432e+06	-1.207432e+06	0.000000	2.427262	0.663465	0.000000	...	0.000000
370	370	0.000000	0.000000	369.303146	-1.216693e+06	-1.216693e+06	0.000000	2.427262	0.662845	0.000000	...	0.000000
371	371	0.000000	0.000000	369.303146	-1.226007e+06	-1.226007e+06	0.000000	2.427262	0.662225	0.000000	...	0.000000

Explorable: Model Graph



Performance

Institute and Faculty of Actuaries

Python Performance

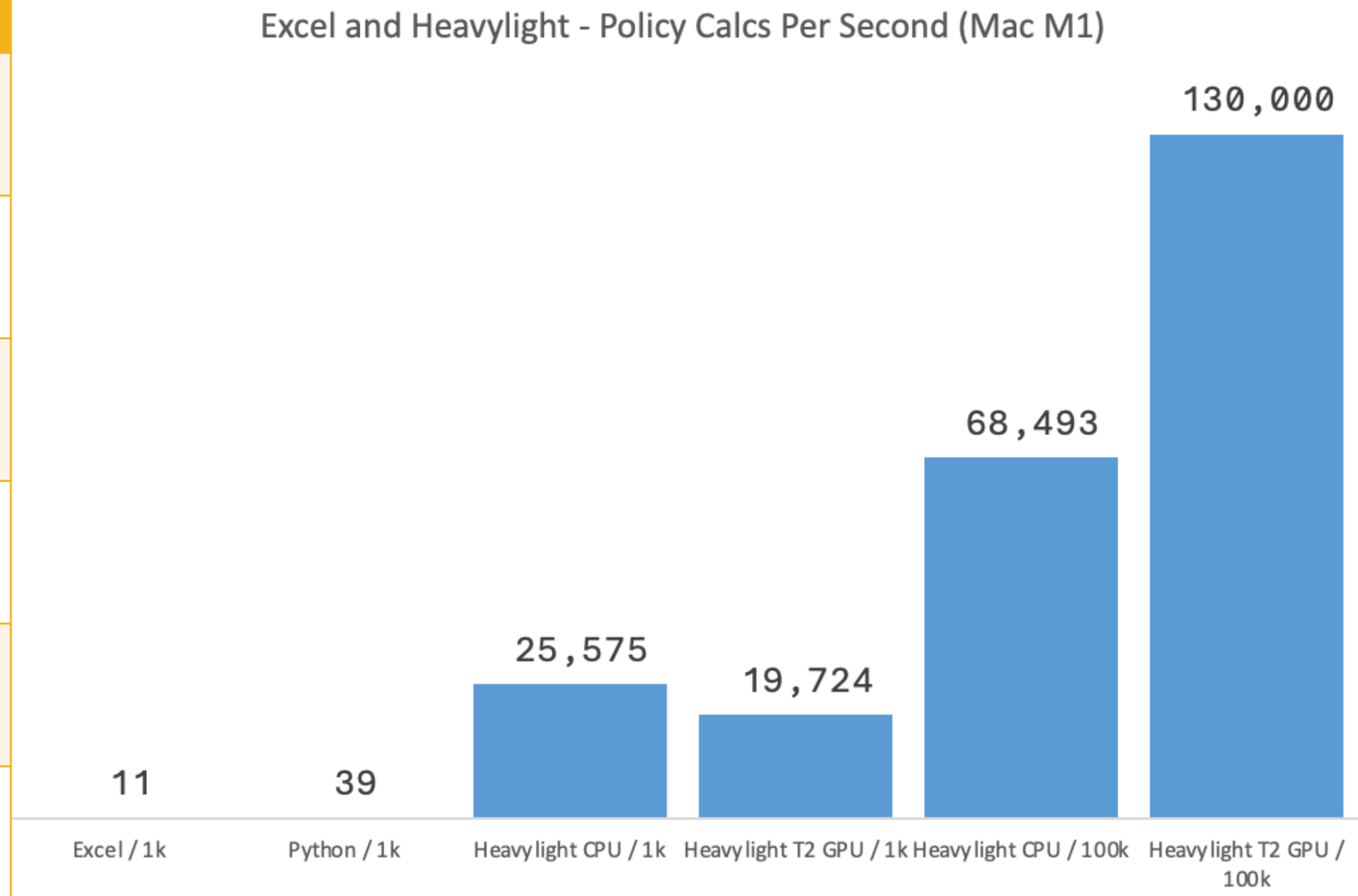
Python core performance is poor. If you write your python code based on Excel or other languages, you won't see much benefit.

Time to run 1000 policies	Time (secs)
Excel / VBA (Mac M1)	95
Python - naïve loop	26

Python Performance with Heavylight

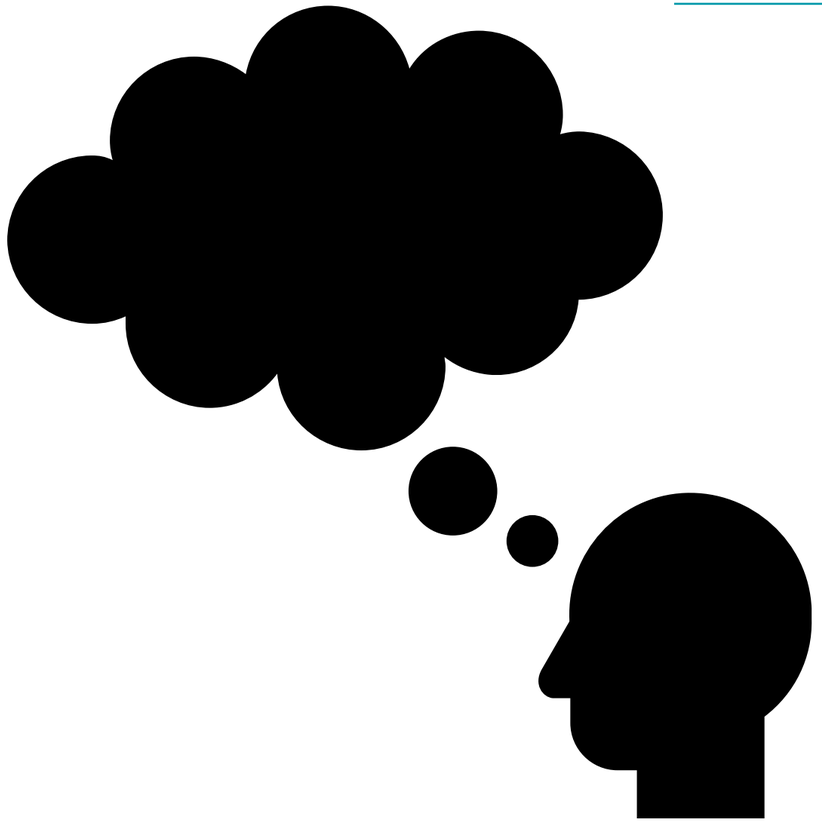
We can use vectorization and other techniques to massively improve performance.

Approach / Batch Size	pols/sec
Excel 1k	11
Python Core 1k	39
Heavylight CPU 1k	25,575
Heavylight T2 GPU 1k	19,724
Heavylight T2 CPU 100k	68,493
Heavylight T2 GPU 100k	130,000



The LBG Journey

LBG Journey: The challenge we set ourselves



To develop the team's Python modelling capabilities

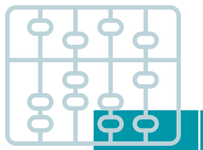
To increase pricing agility

To pass model governance requirements

LBG Journey: What Have We Delivered?

Upskilling

- Developed by new user of Python
- Improved product understanding and documentation
- LBG Finance award finalist



Functionality

- E2E in one language
- 17 assumptions tables / ~500 lines of code
- Typical run takes 1 sec per 1k model points

Governance

- Complies with LBG software engineering guardrails
- Tested and Independent internal validation completed

LBG Journey: What Have We Learned?

Developing Capability

Relatively easy to learn Python

SME availability

Building Models

Documentation availability

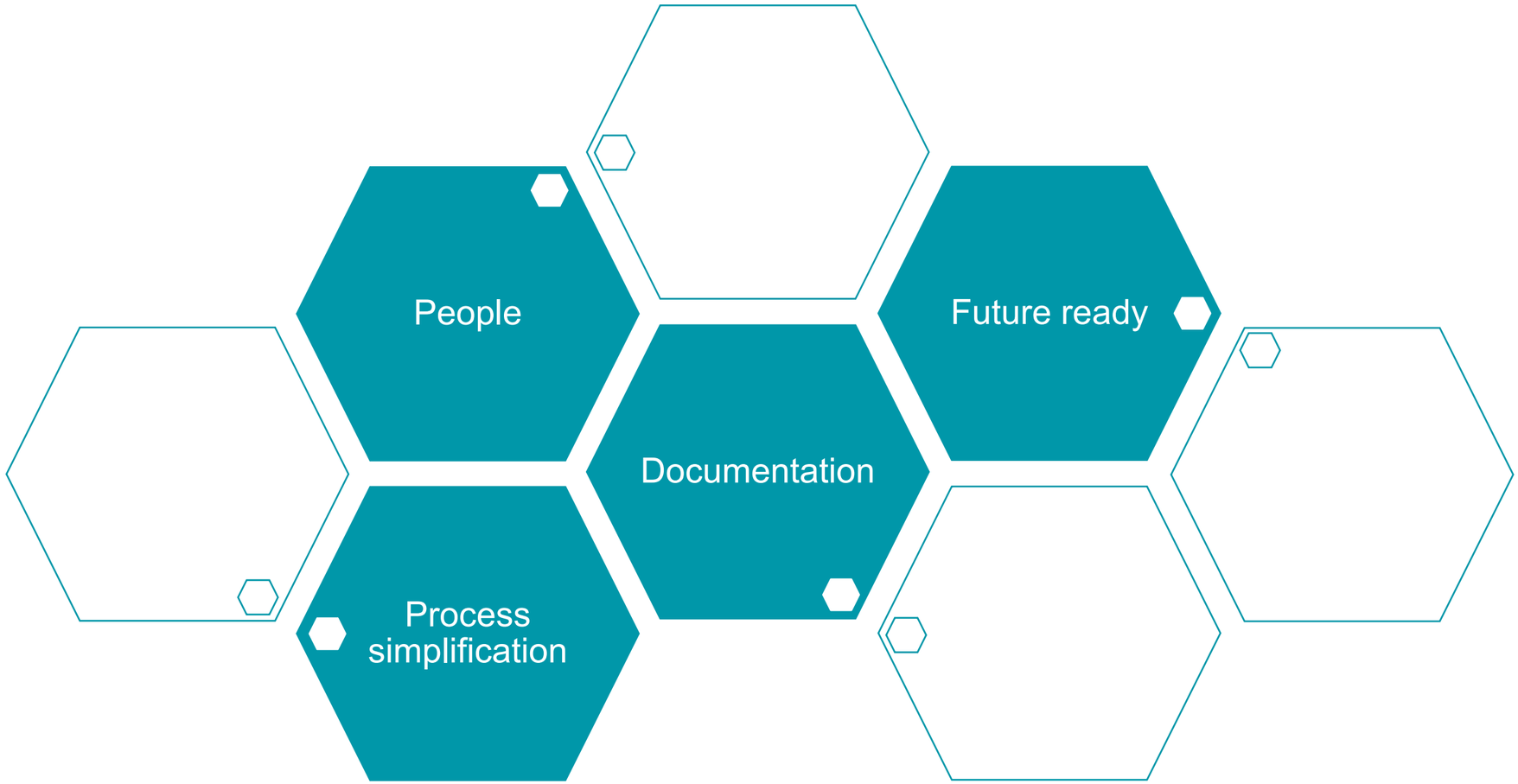
Assumption definition

People and Process

Design thinking essential

Model is a small component

What are the next opportunities?



Questions

Institute and Faculty of Actuaries

Lewis.Fogden@scottishwidows.co.uk

Chloe.Tervit@scottishwidows.co.uk