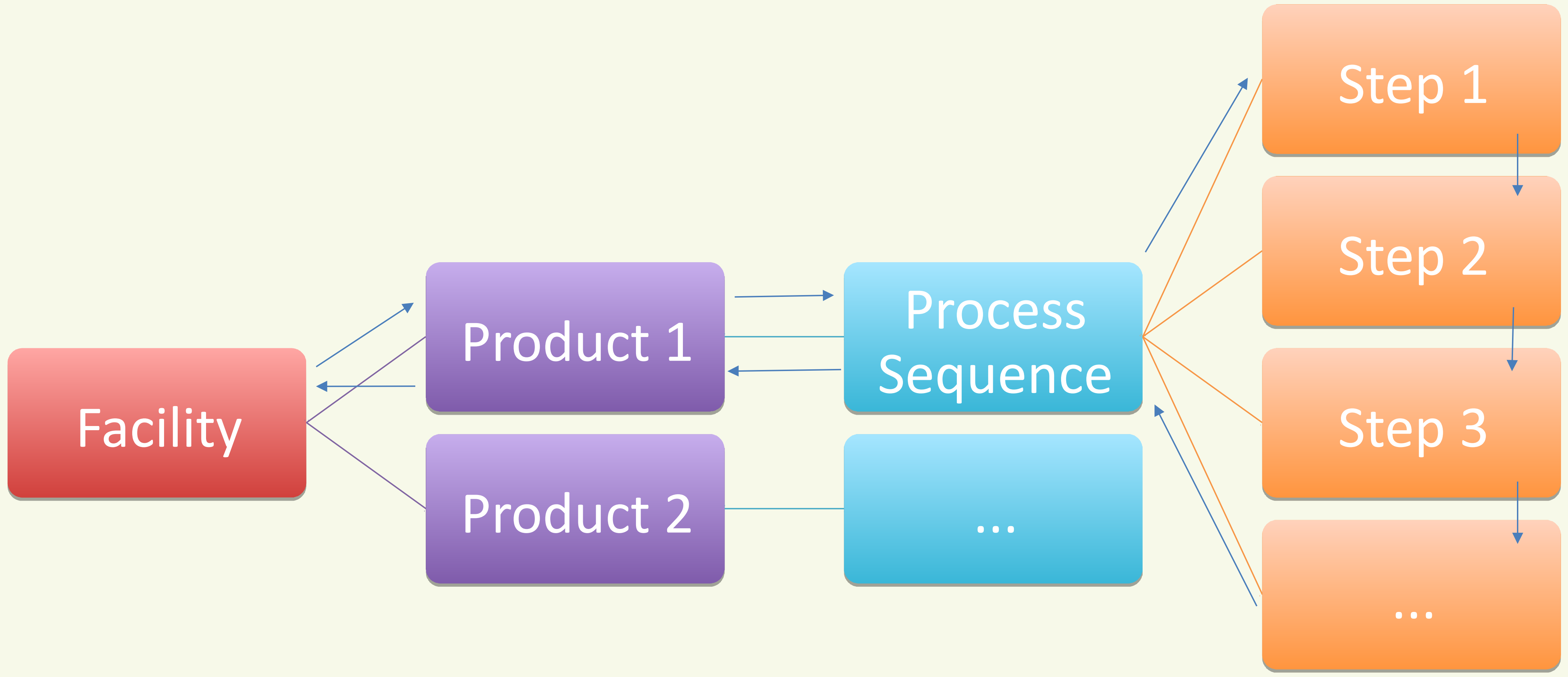
PyBioPharma: cost-effective drug production, and lessons learned

Jonathan Cooper
Head of Research Software Development
University College London

What is PyBioPharma?

1. An object-oriented Python framework for developing and analysing process economics models of biopharmaceutical manufacturing facilities
2. A specific such model of antibody production translated from the C# code

Simulation organisation



Key features

- Flexible configuration of different models: process sequences, equations, parameters
- Automatically catch (common) errors in model implementation
  - Units checking & conversion
  - Mechanisms for specifying parameters & outputs
- Automated tests for model & framework
- Parameters read from CSV and YAML files, kept in version control
  - Easy to see what has changed between experiments
  - A Microsoft Access database was used by C# code

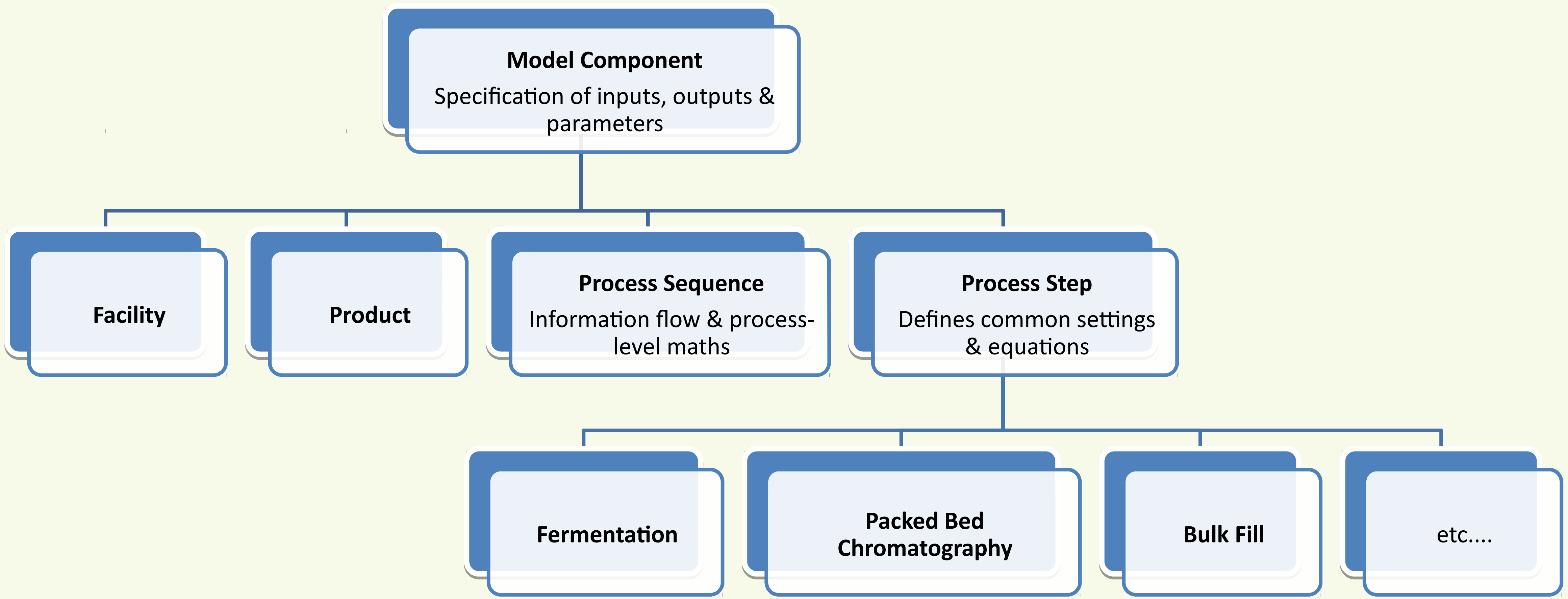
Key features

- Separation of concerns: minimal intertwining of model definition, multi-objective optimisation, and Monte Carlo sensitivity analysis
  - Includes a genetic algorithm optimiser based on deap
- Web browser interface with Jupyter Notebook, for researchers
  - Documented examples of typical analyses
  - Researchers can edit parameters and model equations in the browser

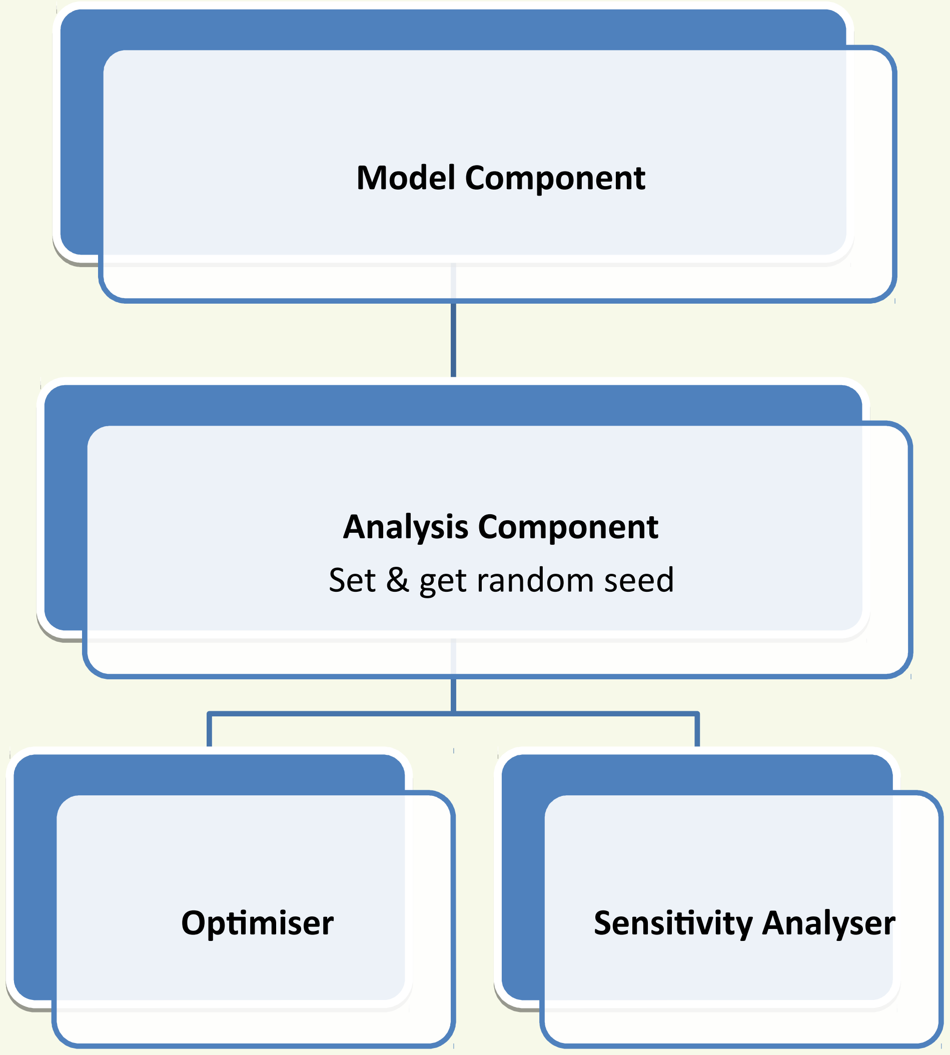
Key features

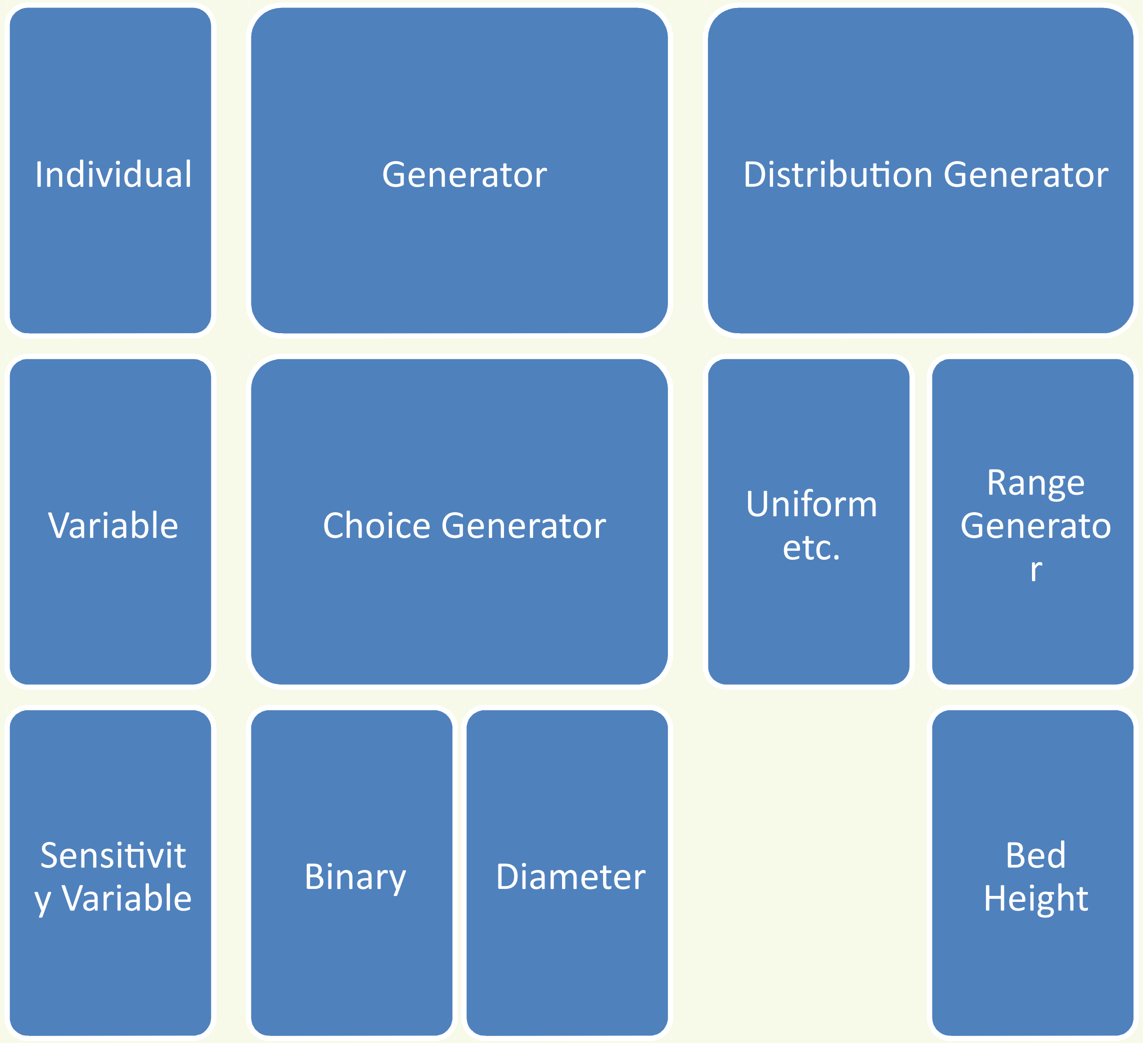
- Easy to build user interfaces for models
  - Everything is self-documenting
  - Minimal hand-customisation of UI needed for a new model
- A (Flask) web interface targeted at end users
  - Those who just want to input parameter values and run scenarios, rather than updating the model
  - Generated from the specifications of parameters, outputs, etc.
  - Uses a Celery task queue for running experiments

Class hierarchy


Model Component
Specification of inputs, outputs & parameters
Facility
Product
Process Sequence
Information flow & process-level maths
Process Step
Defines common settings & equations
Fermentation
Packed Bed Chromatography
Bulk Fill
etc....

Class hierarchy (analysis)





Model component features

- Specify `INPUTS`, `OUTPUTS`, and `PARAMETERS`
- Assignments to `self.inputs`, `self.outputs`, `self.parameters` are checked against these
- Method `load_parameters` reads values from YAML & CSV files
  - File name set by `param_filename` argument when creating the component
- The `name` constructor argument gives a label for the component

Specification examples

```
PARAMETERS = {
    'totalDemand': Q('kg',
                     'Demand for this product over the planning horizon (typically annual)'),
    'extraProduction': Q('dimensionless', 'Factor by which we multiply the clinical demand'),
    'nrBatches': Q('count', 'How many batches should be produced (theoretically).'
                   ' Can be set to 0 to compute this.'),
```

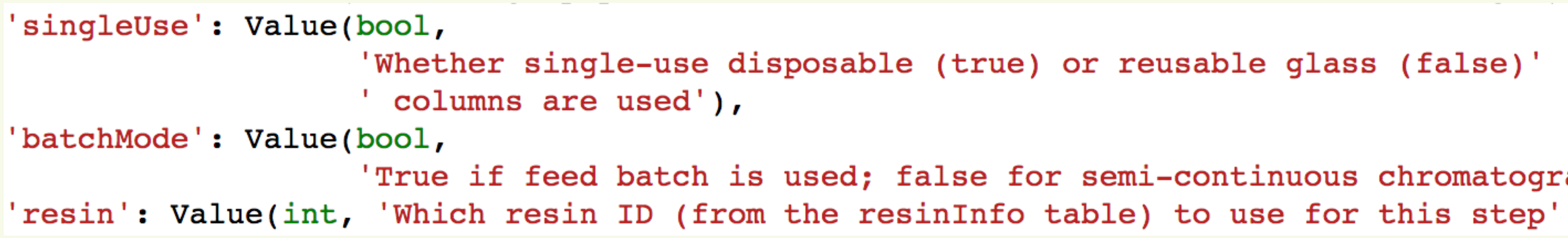
```
INPUTS = {
    'mass': Q('g', 'Total mass before this step'),
    'volume': Q('L', 'Volume before this step'),
    'water': Q('L', 'Amount of water used'),
```

```
OUTPUTS = {
    # Main output of interest
    'cogs': Q('GBP/g', 'Total cost of goods per gram for all batches'),
    # Info on what was produced
    'theoreticalNrBatches': Q('count', 'Estimate of how many batches will be produced'),
    'nrBatches': Q('count', 'How many batches were actually produced'),
```

```
self.outputs['mass'] = self.inputs['volume'] * self.parameters['titre']
self.outputs['volume'] = bp.round(self.inputs['volume'], units.litre)
```

Specification types

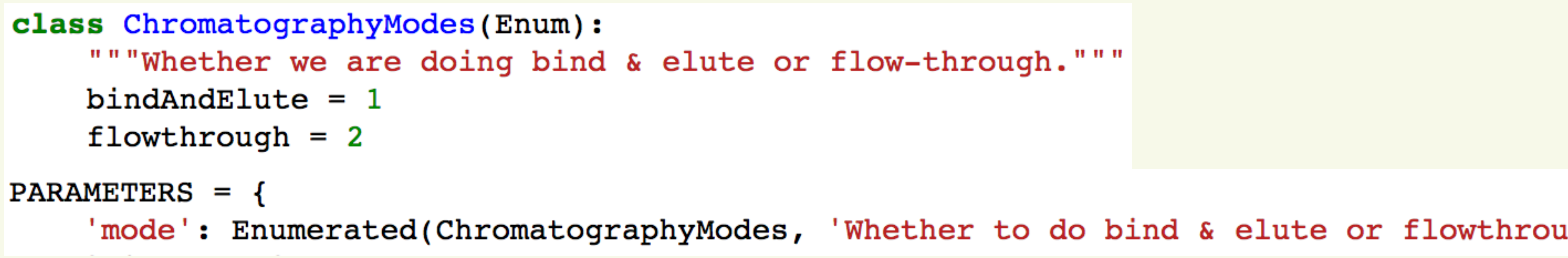
- Q: a quantity in the given units
- Value: a value of the specified type



```
'singleUse': Value(bool,
                   'Whether single-use disposable (true) or reusable glass (false)'
                   ' columns are used'),
'batchMode': Value(bool,
                   'True if feed batch is used; false for semi-continuous chromatogr
'resin': Value(int, 'Which resin ID (from the resinInfo table) to use for this step'
```

Specification types

- Q: a quantity in the given units
- Value: a value of the specified type
- Enumerated: values chosen from a list

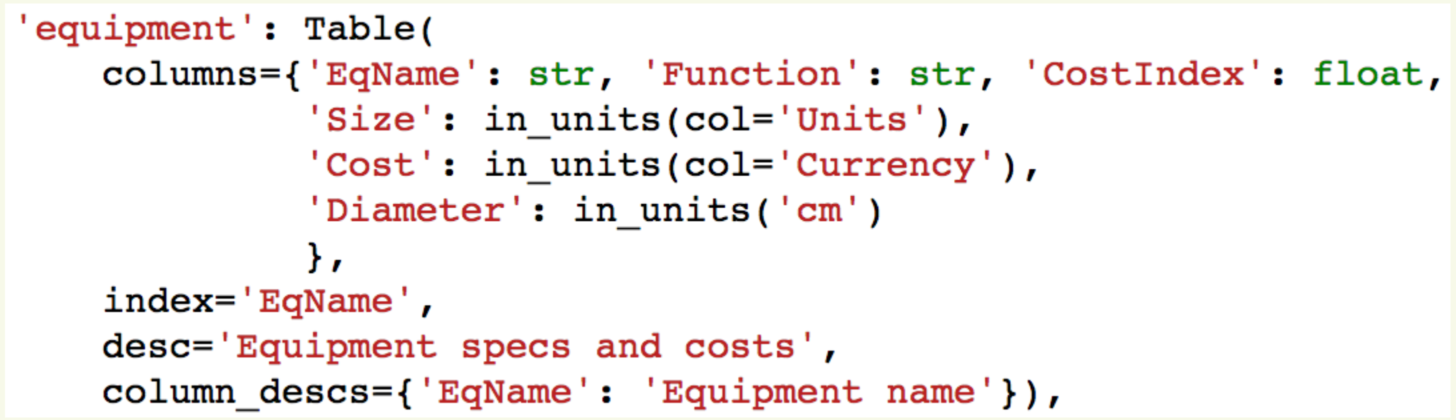


```
class ChromatographyModes(Enum):
    """Whether we are doing bind & elute or flow-through."""
    bindAndElute = 1
    flowthrough = 2
```

```
PARAMETERS = {
    'mode': Enumerated(ChromatographyModes, 'Whether to do bind & elute or flowthrou
```

Specification types

- Q: a quantity in the given units
- Value: a value of the specified type
- Enumerated: values chosen from a list
- Table: a table of parameters read from a CSV file



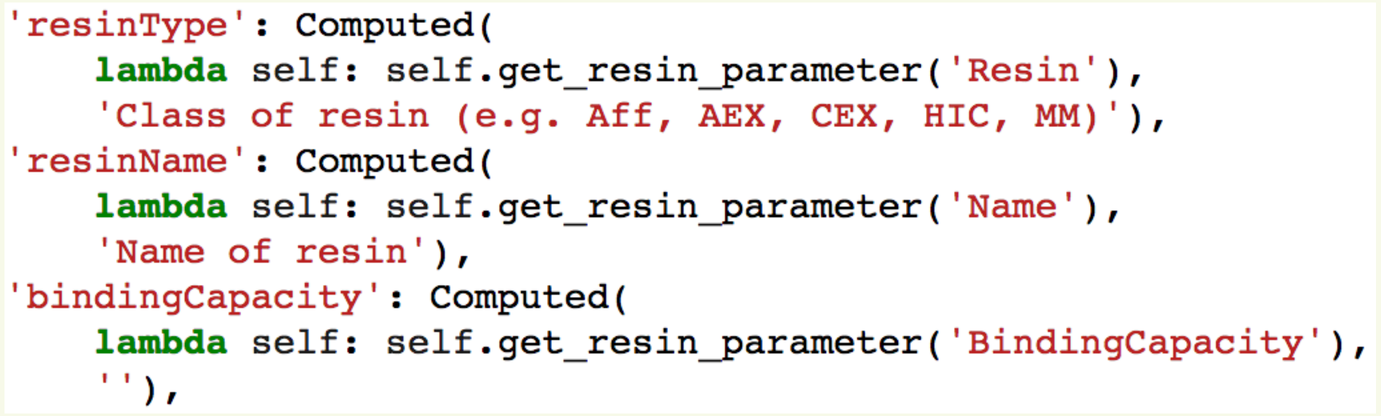
```
'equipment': Table(
    columns={'EqName': str, 'Function': str, 'CostIndex': float,
             'Size': in_units(col='Units'),
             'Cost': in_units(col='Currency'),
             'Diameter': in_units('cm')
             },
    index='EqName',
    desc='Equipment specs and costs',
    column_descs={'EqName': 'Equipment name'}),
```

Specification types

- Q: a quantity in the given units
- Value: a value of the specified type
- Enumerated: values chosen from a list
- Table: a table of parameters read from a CSV file
- Nested: a group of related parameters

Specification types

- Q: a quantity in the given units
- Value: a value of the specified type
- Enumerated: values chosen from a list
- Table: a table of parameters read from a CSV file
- Nested: a group of related parameters
- Computed: a parameter derived automatically from others by some function



```
'resinType': Computed(
    lambda self: self.get_resin_parameter('Resin'),
    'Class of resin (e.g. Aff, AEX, CEX, HIC, MM)'),
'resinName': Computed(
    lambda self: self.get_resin_parameter('Name'),
    'Name of resin'),
'bindingCapacity': Computed(
    lambda self: self.get_resin_parameter('BindingCapacity'),
    ''),
```

Live demo

- See web browser

Lessons learned

- Importance of ‘coal face’ user engagement throughout
  - For gathering requirements
  - For project sustainability
- Web interfaces always take longer than you think…

Possible future work

- Extract the model-agnostic parts of the framework into a standalone package
  - If anyone else wants this!
- Automatic tracking of provenance for outputs
  - The web interface does this for users, but we could use e.g. recipy for researches interacting via Jupyter
- Further examples of visualisations, results analyses, etc.

Questions

Over pizza & drinks!