porchlight

An open-source function management library for Python

(D. J.) Teal | They/Them

University of Maryland, College Park

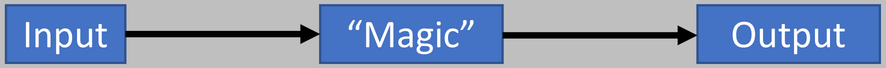
Software accessibility

- Application-Programming Interfaces (APIs)
  - Primary interface to your work
  - Enables reusability/extendibility
- That said:
  - Take time/planning to implement
  - Scientific code is not bound to an API
    - Can do science without it

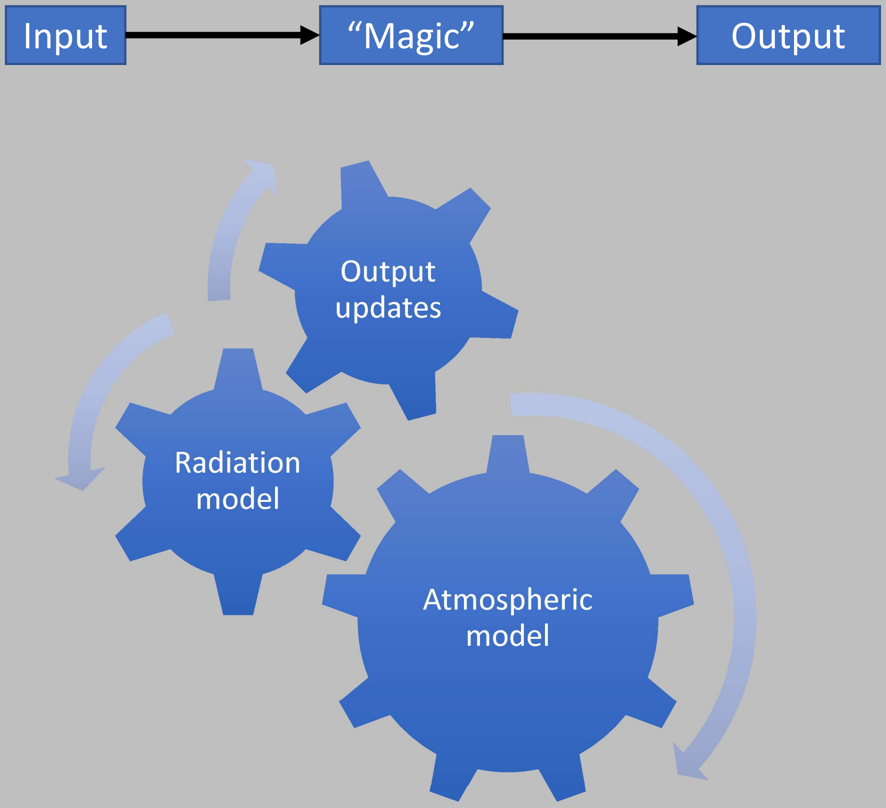
What is Porchlight?

- Open-source Python package
- Provides a mediator/adapter framework for arbitrary networks of python functions
- Helps make models, pipelines, software accessible

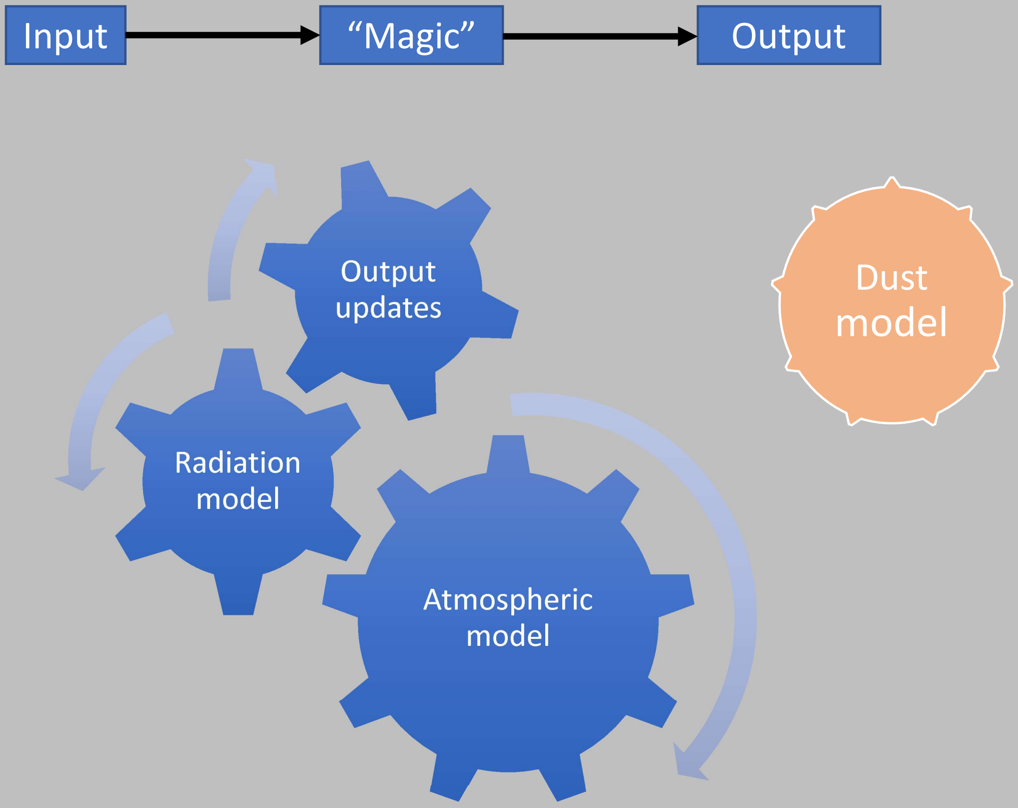
What makes a model/pipeline?



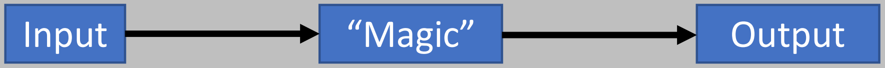
What makes a model/pipeline?

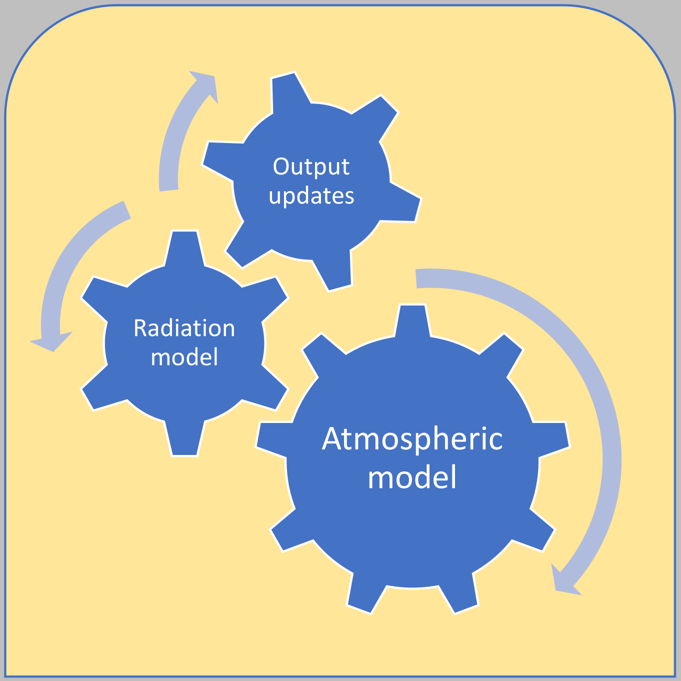


What makes a model/pipeline?



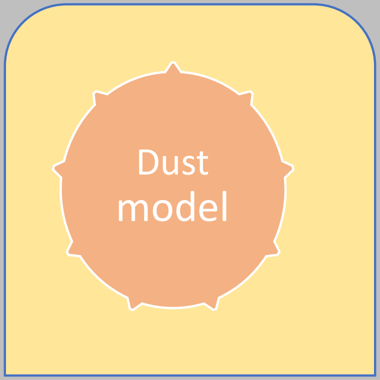
What makes a model/pipeline?





Input: Temperature, Pressure

Output: Temperature, Pressure, Spectrum, Bulk Composition



Input: Temperature, Pressure, Progenitor Density

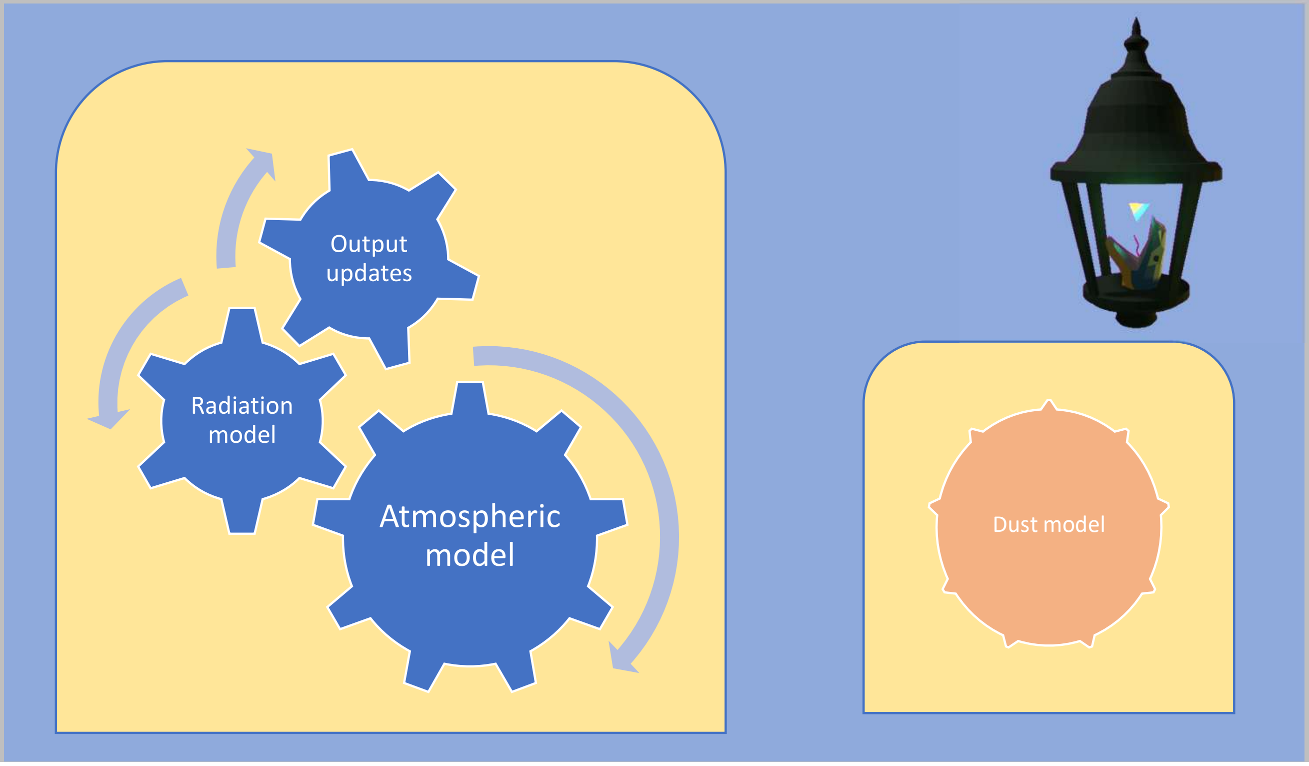
Output: Dust formation rate

What makes a model/pipeline?

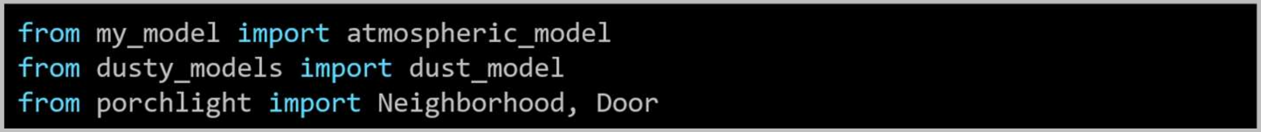
Input: Temperature, Pressure, Progenitor Density

$$\begin{tikzcd}\boxed{\text{Input}} \arrow[r] & \boxed{\text{“Magic”}} \arrow[r] & \boxed{\text{Output}}\end{tikzcd}$$

Output: Temperature, Pressure, Spectrum, Bulk Composition, Dust Formation Rate

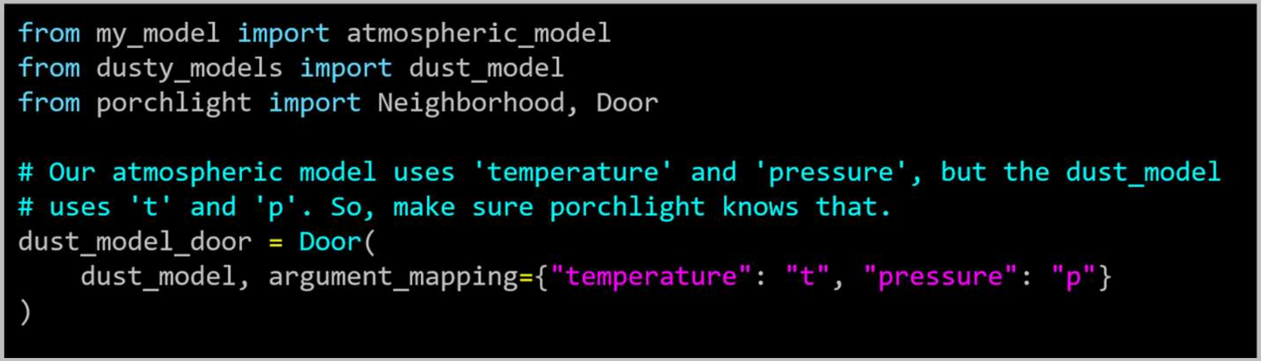


What does this really look like?



```
from my_model import atmospheric_model
from dusty_models import dust_model
from porchlight import Neighborhood, Door
```

What does this really look like?

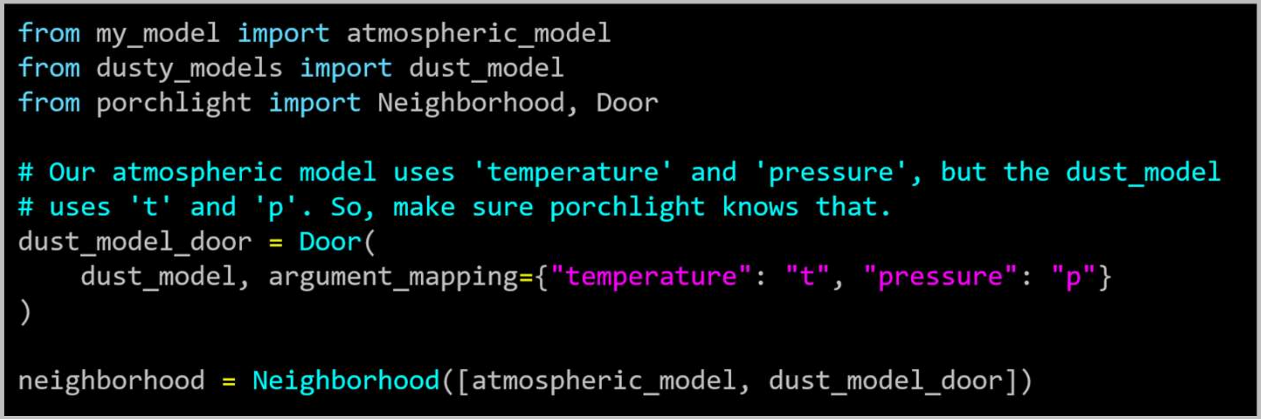


```python
from my_model import atmospheric_model
from dusty_models import dust_model
from porchlight import Neighborhood, Door

# Our atmospheric model uses 'temperature' and 'pressure', but the dust_model
# uses 't' and 'p'. So, make sure porchlight knows that.
dust_model_door = Door(
    dust_model, argument_mapping={"temperature": "t", "pressure": "p"}
)
```

- ‘Door’ is porchlight’s adapter for functions

What does this really look like?



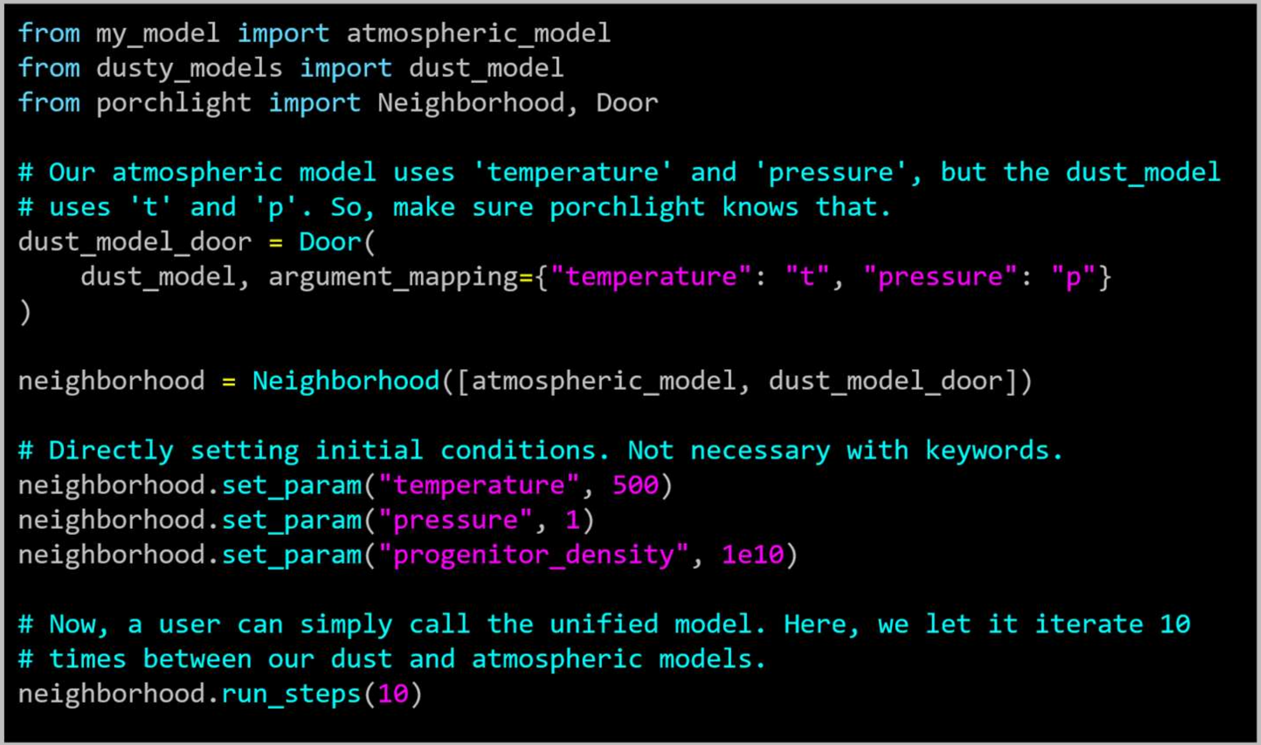
```python
from my_model import atmospheric_model
from dusty_models import dust_model
from porchlight import Neighborhood, Door

# Our atmospheric model uses 'temperature' and 'pressure', but the dust_model
# uses 't' and 'p'. So, make sure porchlight knows that.
dust_model_door = Door(
    dust_model, argument_mapping={"temperature": "t", "pressure": "p"}
)

neighborhood = Neighborhood([atmospheric_model, dust_model_door])
```

- ‘Door’ is porchlight’s adapter for functions
- Now we have an API (via neighborhood)
- Only 5 statements here
- Could:
    - Set parameters to constant
    - Restrict parameter spaces

What does this really look like?



```python
from my_model import atmospheric_model
from dusty_models import dust_model
from porchlight import Neighborhood, Door

# Our atmospheric model uses 'temperature' and 'pressure', but the dust_model
# uses 't' and 'p'. So, make sure porchlight knows that.
dust_model_door = Door(
    dust_model, argument_mapping={"temperature": "t", "pressure": "p"}
)

neighborhood = Neighborhood([atmospheric_model, dust_model_door])

# Directly setting initial conditions. Not necessary with keywords.
neighborhood.set_param("temperature", 500)
neighborhood.set_param("pressure", 1)
neighborhood.set_param("progenitor_density", 1e10)

# Now, a user can simply call the unified model. Here, we let it iterate 10
# times between our dust and atmospheric models.
neighborhood.run_steps(10)
```

- 'Door' is porchlight's adapter for functions
- Now we have an API (via neighborhood)
- Only 5 statements here
- Could:
  - Set parameters to constant
  - Restrict parameter spaces

Where do I find it?

https://github.com/teald/porchlight

- Publicly available on GitHub
  - Actively updated
  - Open source
- Happy to respond to bugs, questions, ideas!

What else can it do?

- Set parameters to be constant
- Runtime failure conditions
  - E.g., negative temperature
- Parameter mapping
  - f(p1, p2) -> f(x, y)