

MEAGAN LANG

CROPS IN SILICO
HACKATHON 2021

BUT FIRST...

NOTEBOOK PREP

PREP CHECKLIST

1. SIGN-UP FOR GITHUB

We will be using GitHub Issues to debug

2. OPEN THE PROJECT MATERIALS

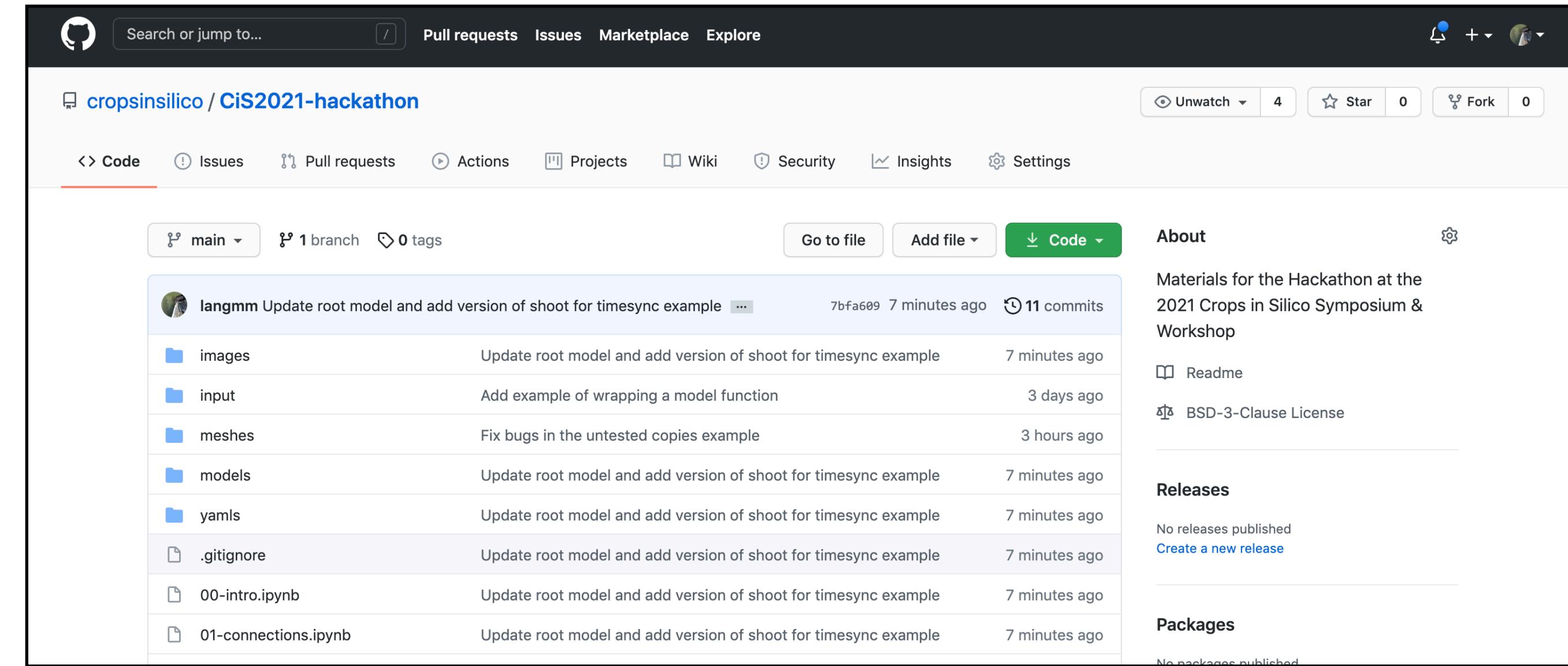
3. START A MYBINDER INSTANCE

We will be using Jupyter notebooks via MyBinder to avoid local installations (instructions for installing locally are found in the README)

Materials: <https://github.com/cropsinsilico/CiS2021-hackathon>

Launch binder
in a new
window/tab

Keep the
materials repo
open, we will
need it later



Materials: <https://github.com/cropsinsilico/CiS2021-hackathon>

Launch binder
in a new
window/tab

Keep the
materials repo
open, we will
need it later

cropsinsilico / CiS2021-hackathon

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

langmm Update root model and add version of shoot for timesync example ... 7bfa609 7 minutes ago 11 commits

images Update root model and add version of shoot for timesync example 7 minutes ago

input Add example of wrapping a model function 3 days ago

meshes Fix bugs in the untested copies example 3 hours ago

models Update root model and add version of shoot for timesync example 7 minutes ago

yaml Update root model and add version of shoot for timesync example 7 minutes ago

.gitignore Update root model and add version of shoot for timesync example 7 minutes ago

00-intro.ipynb Update root model and add version of shoot for timesync example 7 minutes ago

01-connections.ipynb Update root model and add version of shoot for timesync example 7 minutes ago

About

Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop

Readme

BSD-3-Clause License

Releases

No releases published

Create a new release

Packages

No packages published

SCROLL

README.md

CiS2021-hackathon

Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop

[launch binder](#)

Requirements

- Browser (tested on Google Chrome)
- Github Account

Materials: <https://github.com/cropsinsilico/CiS2021-hackathon>

Launch binder
in a new
window/tab

Keep the
materials repo
open, we will
need it later

cropsinsilico / CiS2021-hackathon

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

langmm Update root model and add version of shoot for timesync example ... 7bfa609 7 minutes ago 11 commits

images Update root model and add version of shoot for timesync example 7 minutes ago

input Add example of wrapping a model function 3 days ago

meshes Fix bugs in the untested copies example 3 hours ago

models Update root model and add version of shoot for timesync example 7 minutes ago

yaml Update root model and add version of shoot for timesync example 7 minutes ago

.gitignore Update root model and add version of shoot for timesync example 7 minutes ago

00-intro.ipynb Update root model and add version of shoot for timesync example 7 minutes ago

01-connections.ipynb Update root model and add version of shoot for timesync example 7 minutes ago

About

Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop

Readme

BSD-3-Clause License

Releases

No releases published

Create a new release

Packages

No packages published

README.md

CiS2021-hackathon

Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop

[launch binder](#)

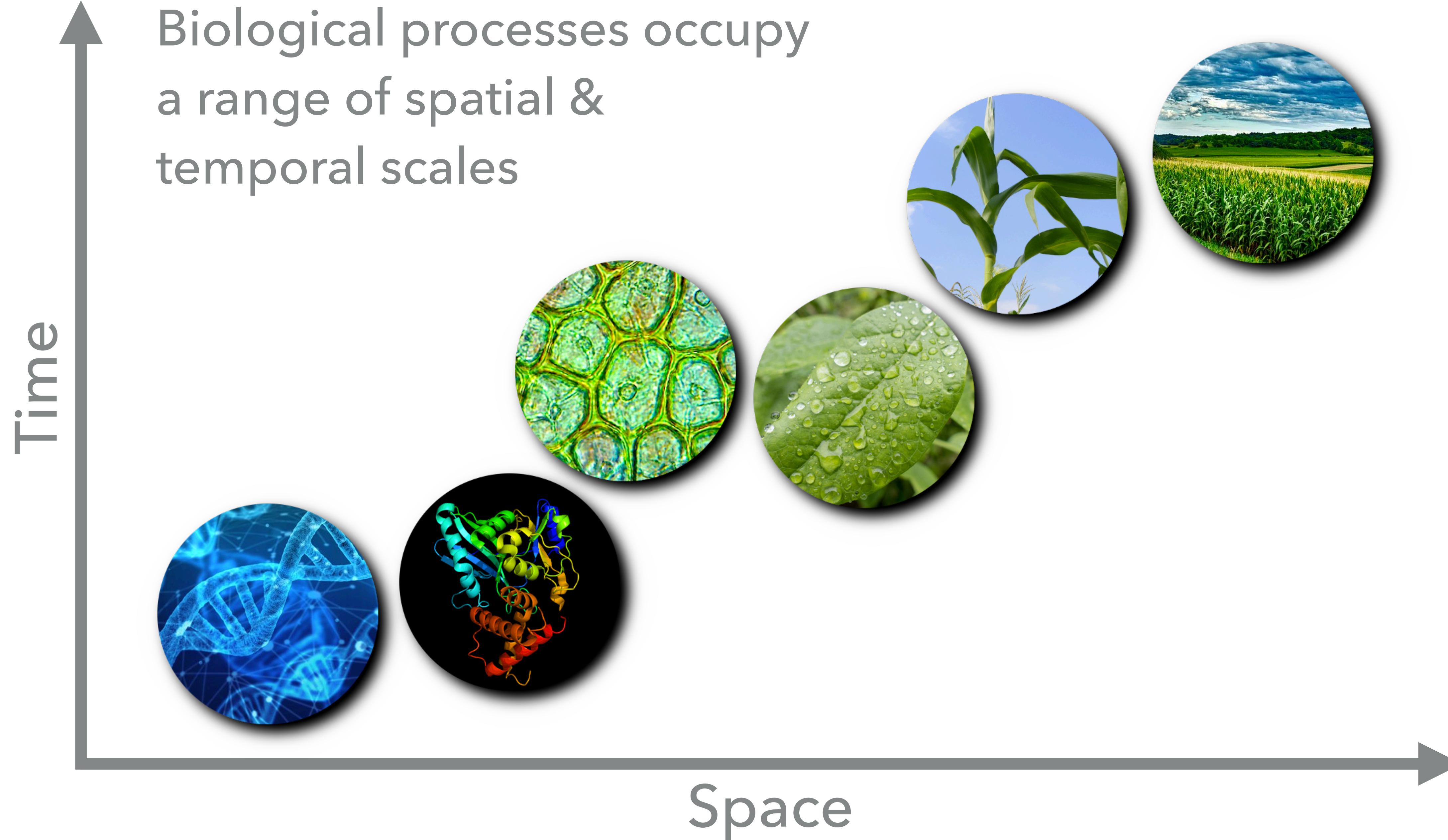
Requirements

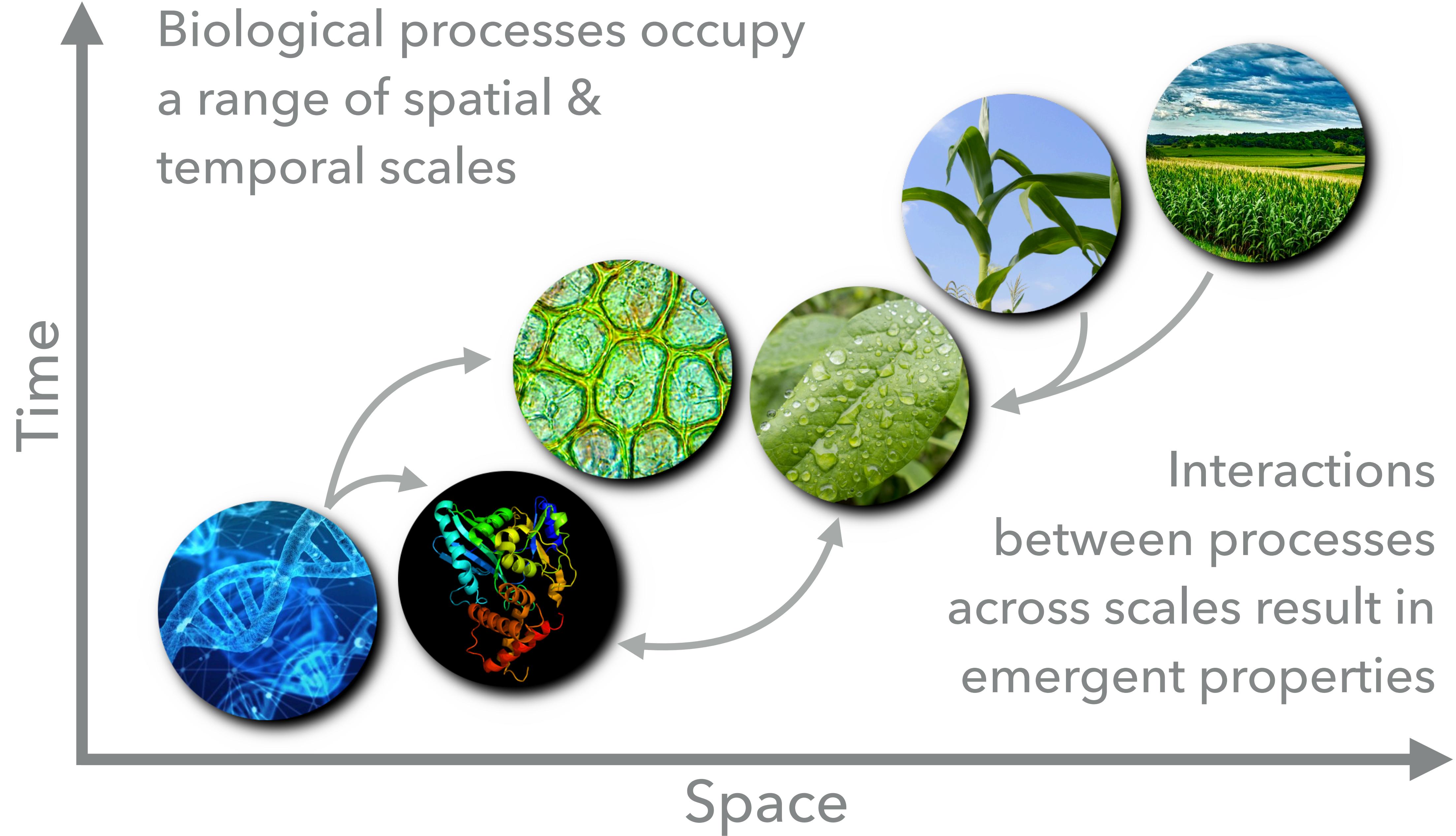
- Browser (tested on Google Chrome)
- Github Account

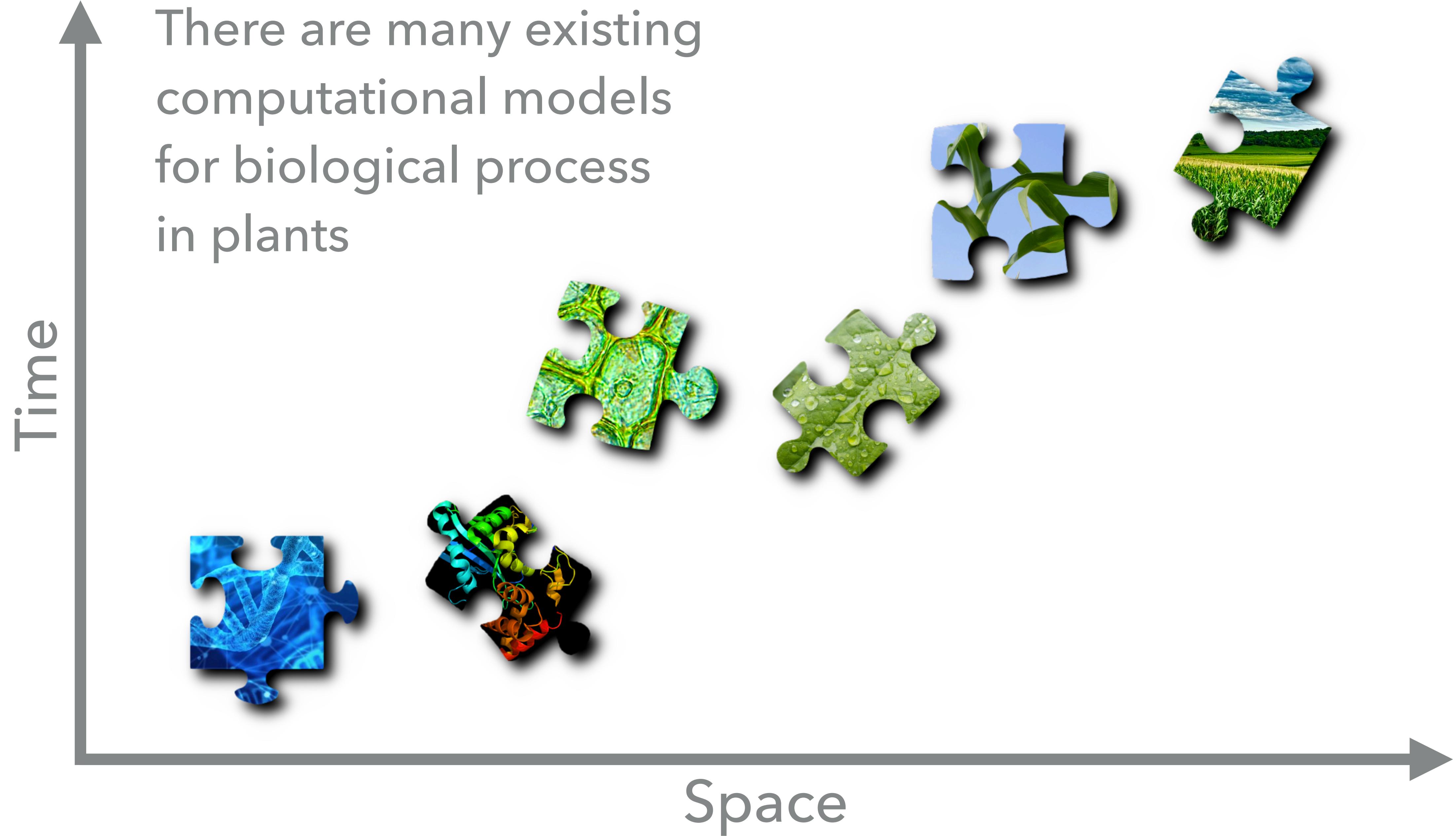
Materials: <https://github.com/cropsinsilico/CiS2021-hackathon>

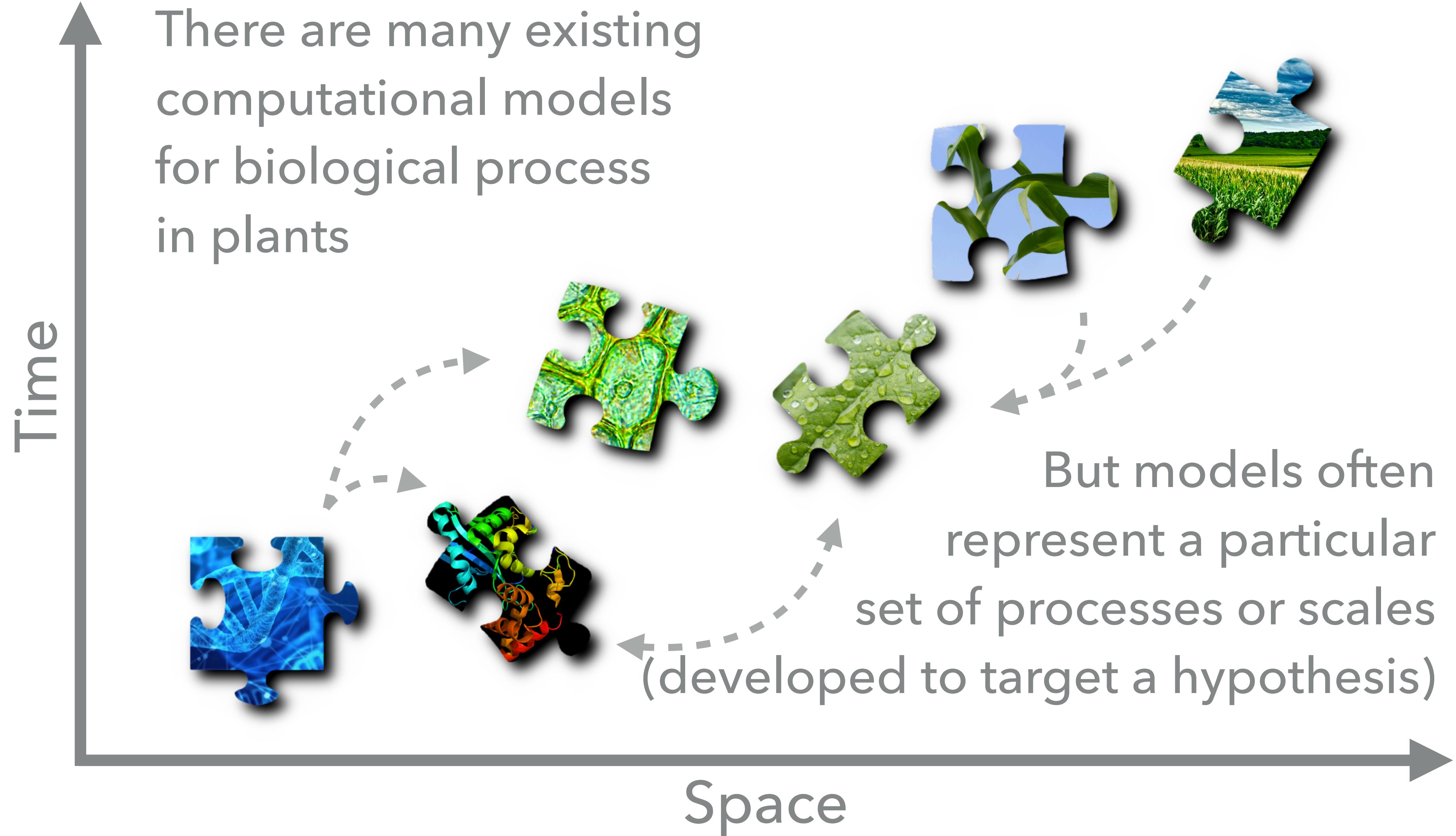
OK BACK TO THE...

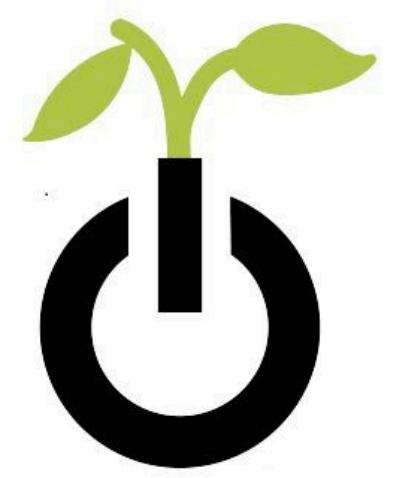
INTRODUCTION











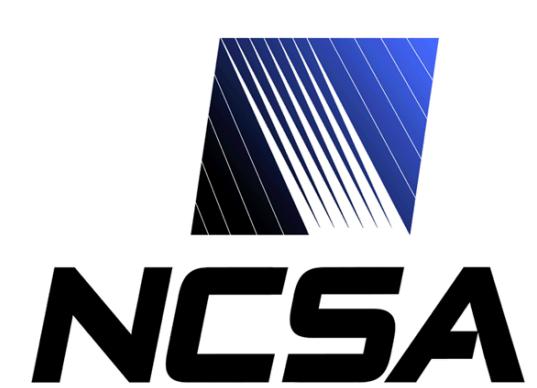
Crops *in silico*

I ILLINOIS

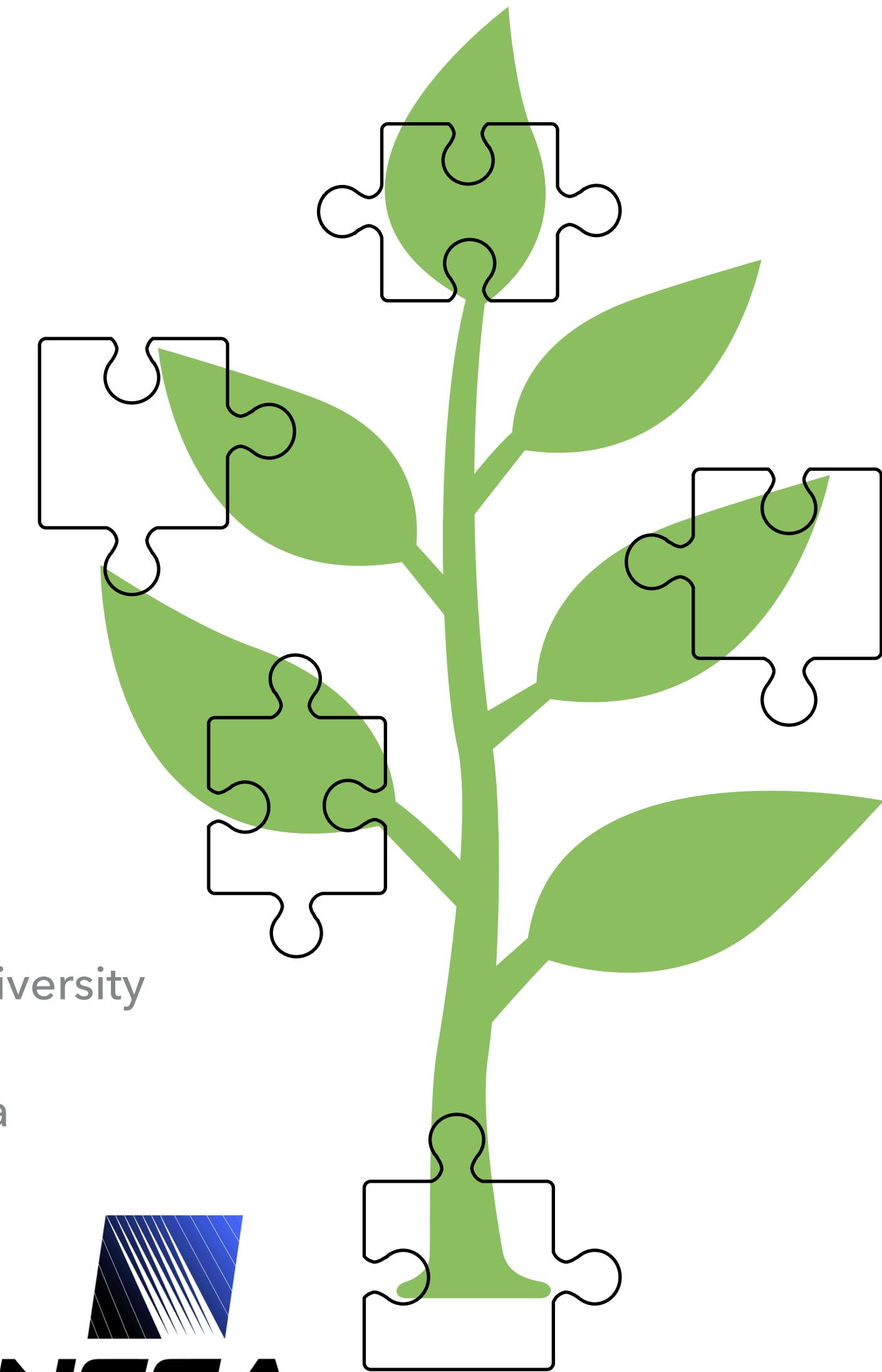
Funded by



FFAR



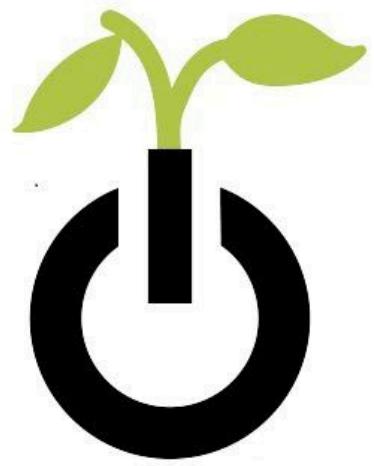
In partnership with
Oxford University
Pennsylvania State University
Purdue University
University of Nebraska





Models are pieces from
different puzzles

We need an adapter



Crops *in silico*¹



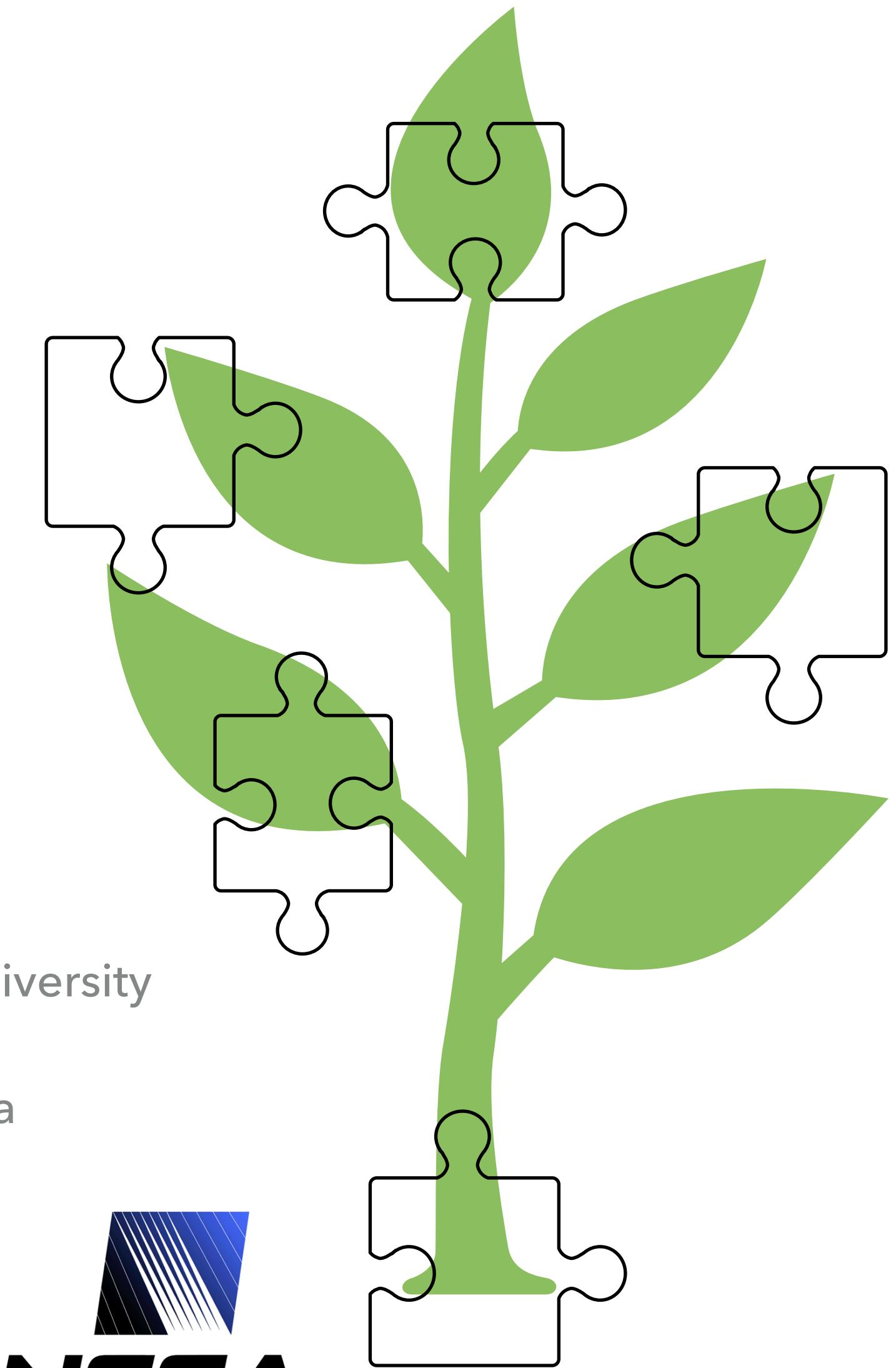
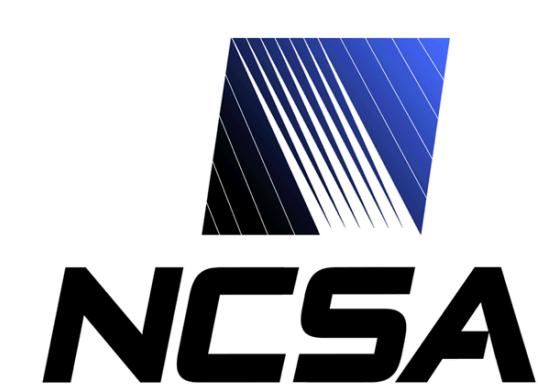
ILLINOIS

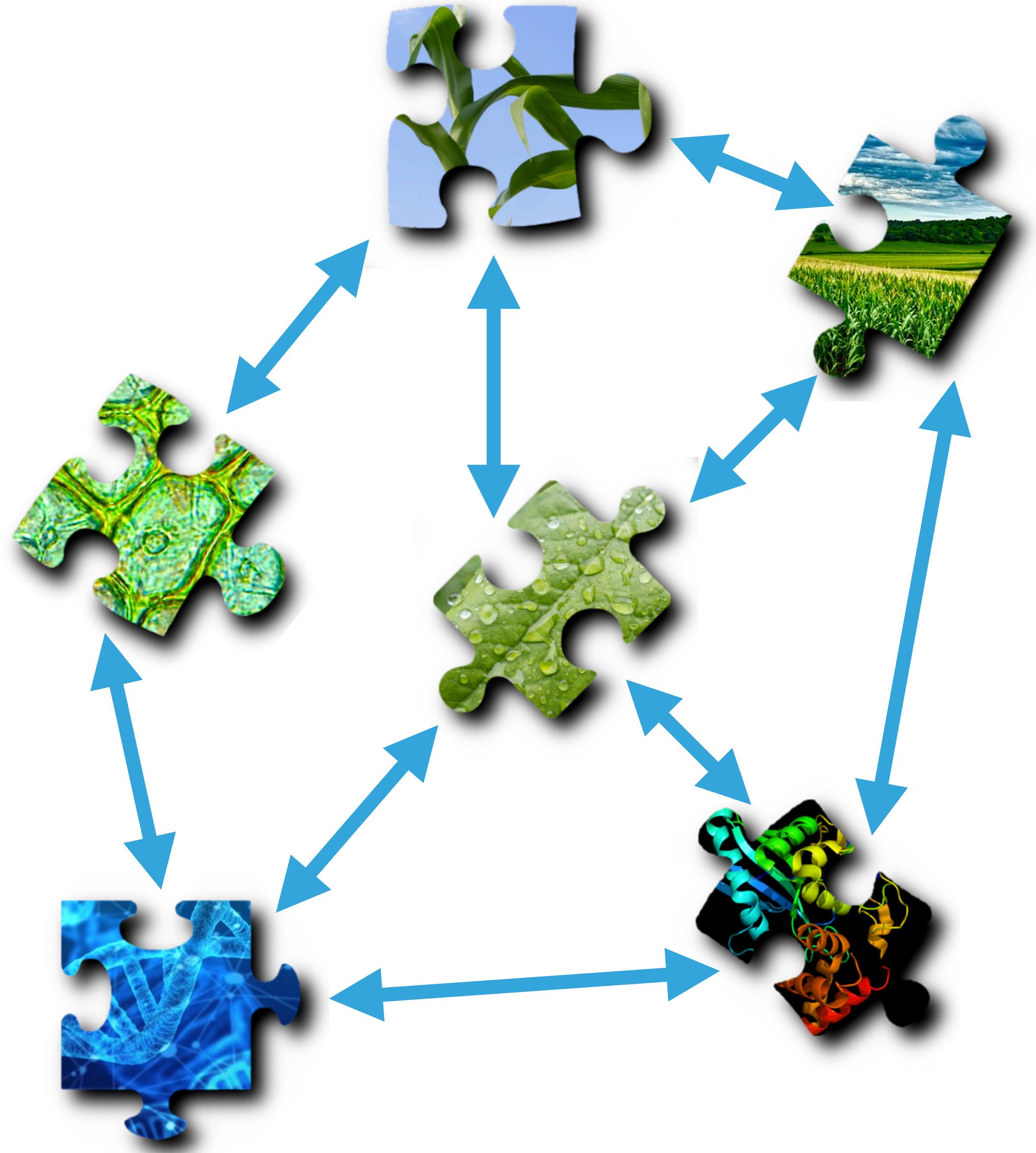
In partnership with
Oxford University
Pennsylvania State University
Purdue University
University of Nebraska

Funded by



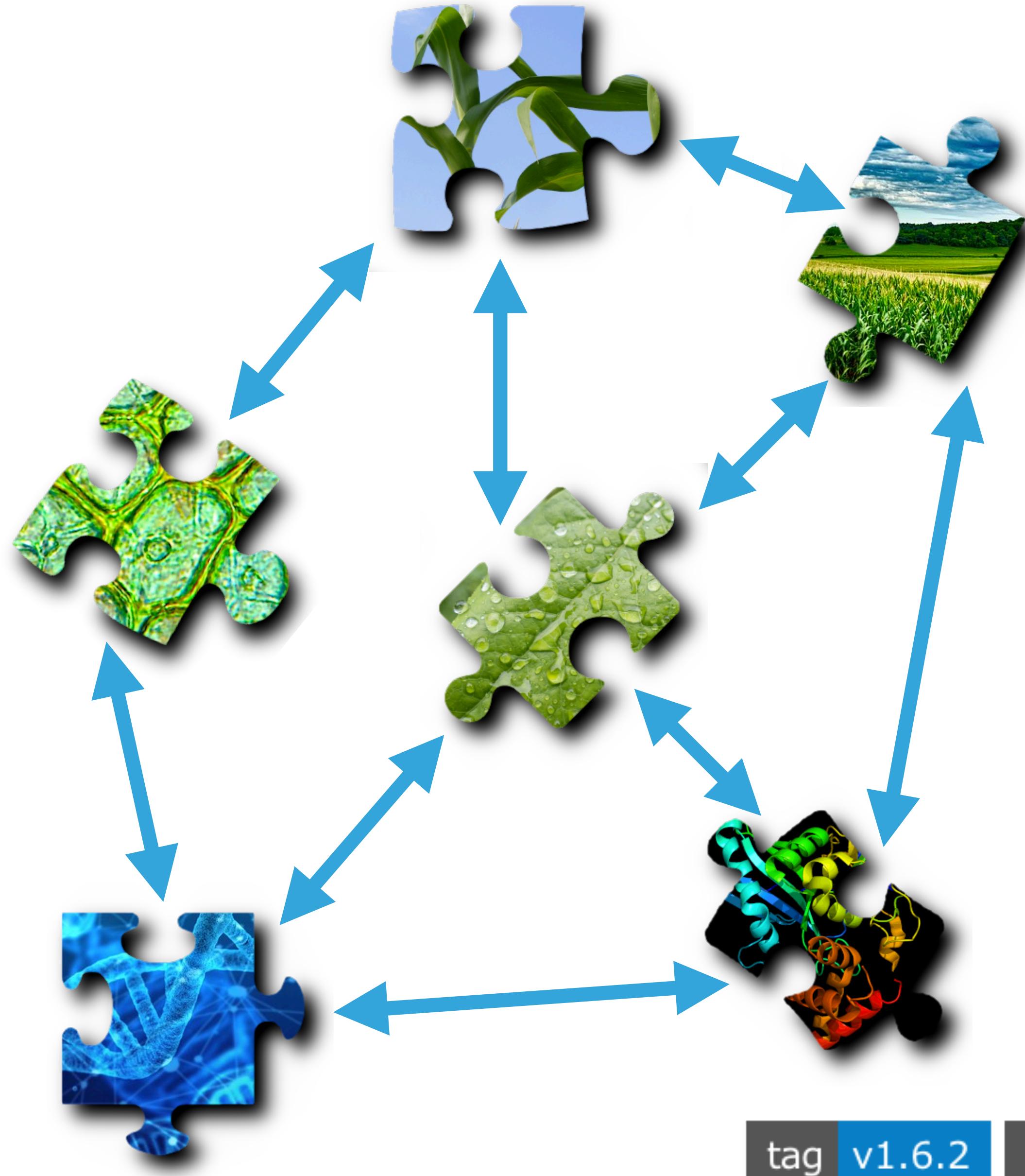
FFAR





YGGDRASIL:

OPEN SOURCE PYTHON
PACKAGE FOR
CONNECTING MODELS
ACROSS SCALES AND
LANGUAGES



YGGDRASIL:

OPEN SOURCE PYTHON
PACKAGE FOR
CONNECTING MODELS
ACROSS SCALES AND
LANGUAGES

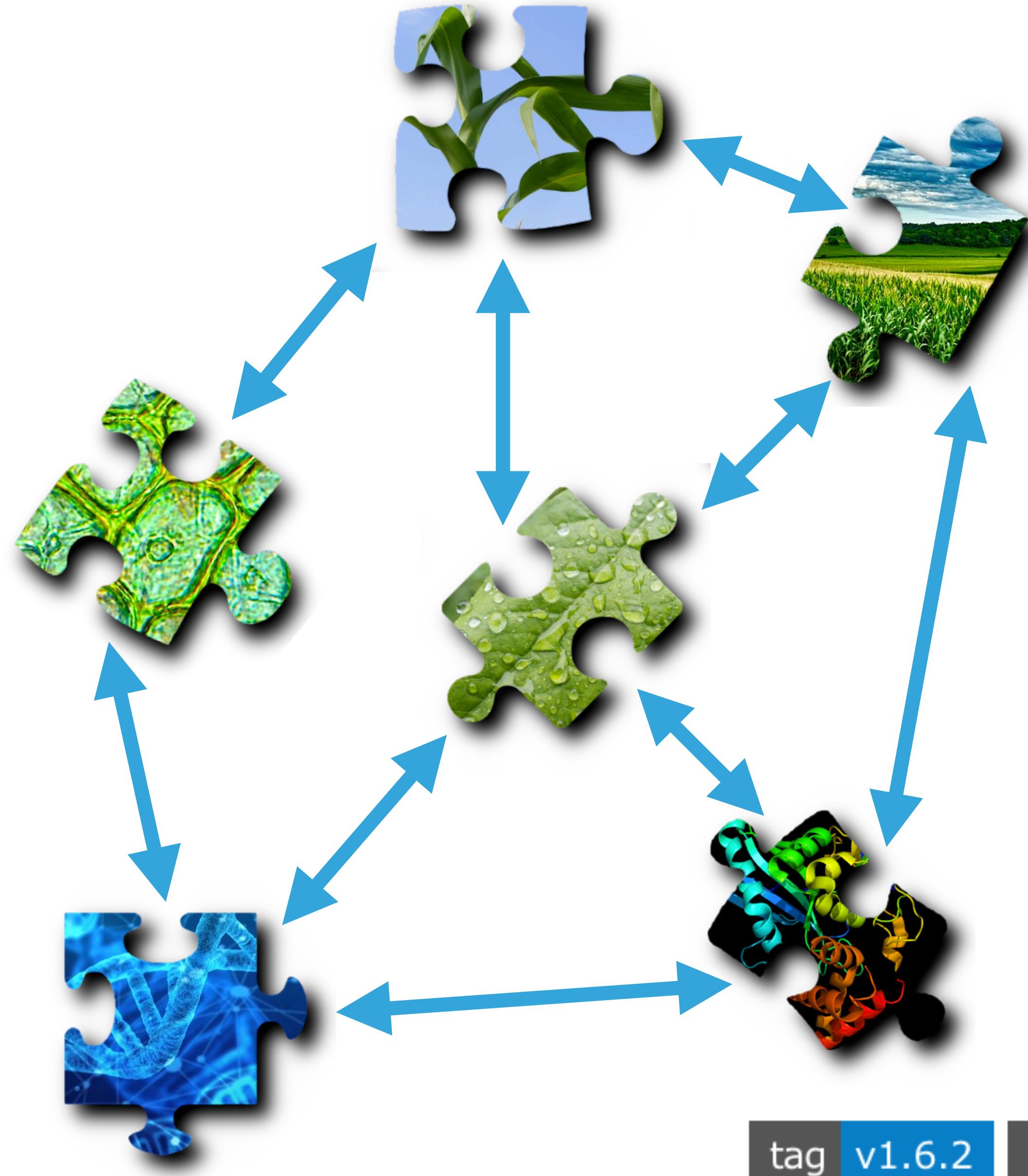
tag v1.6.2

pypi v1.6.2

build passing

coverage 100%

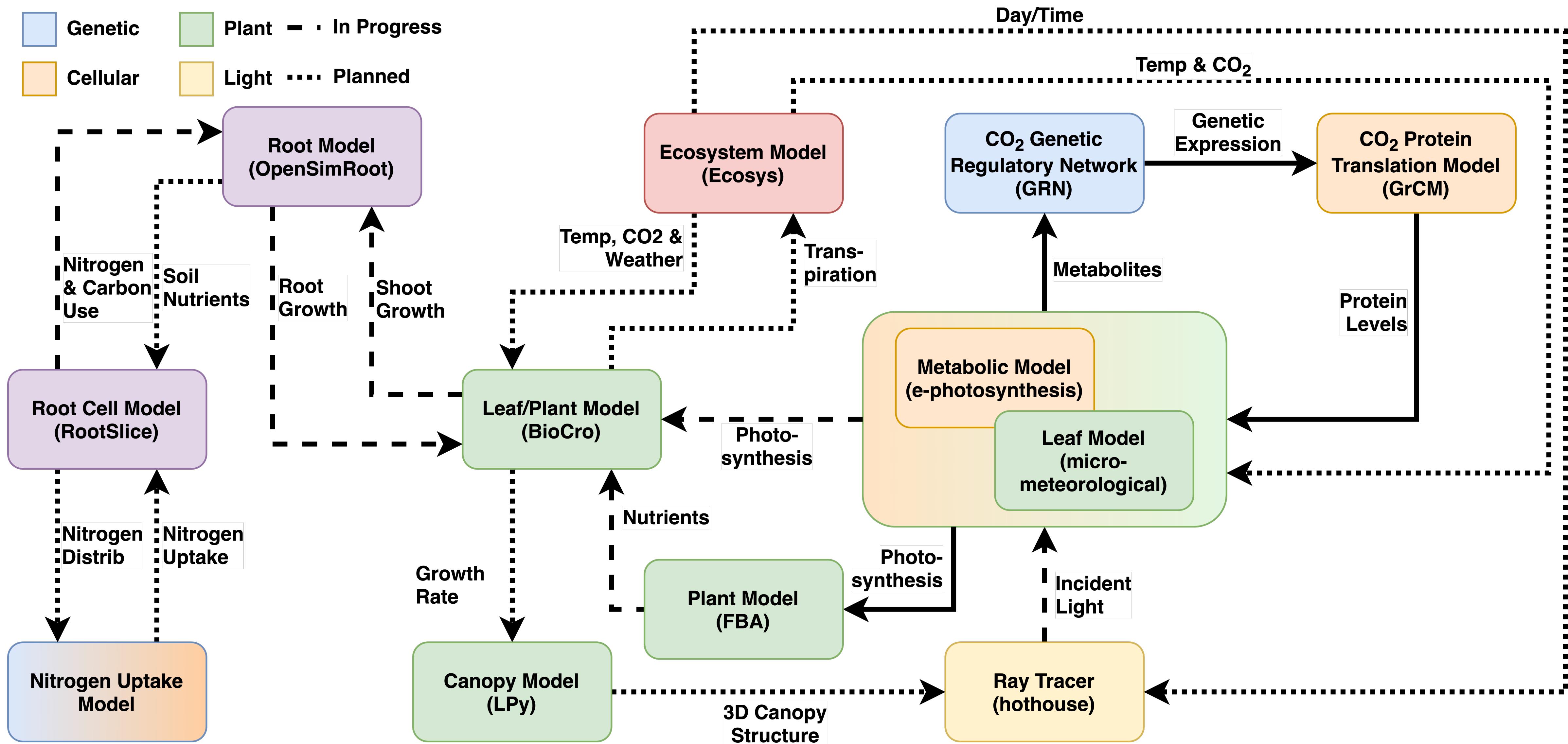
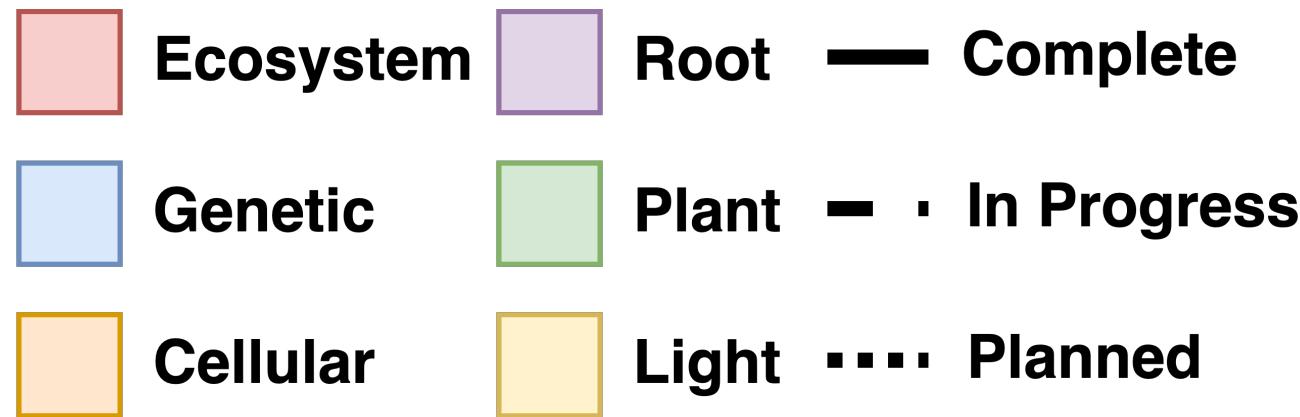
code style pep8

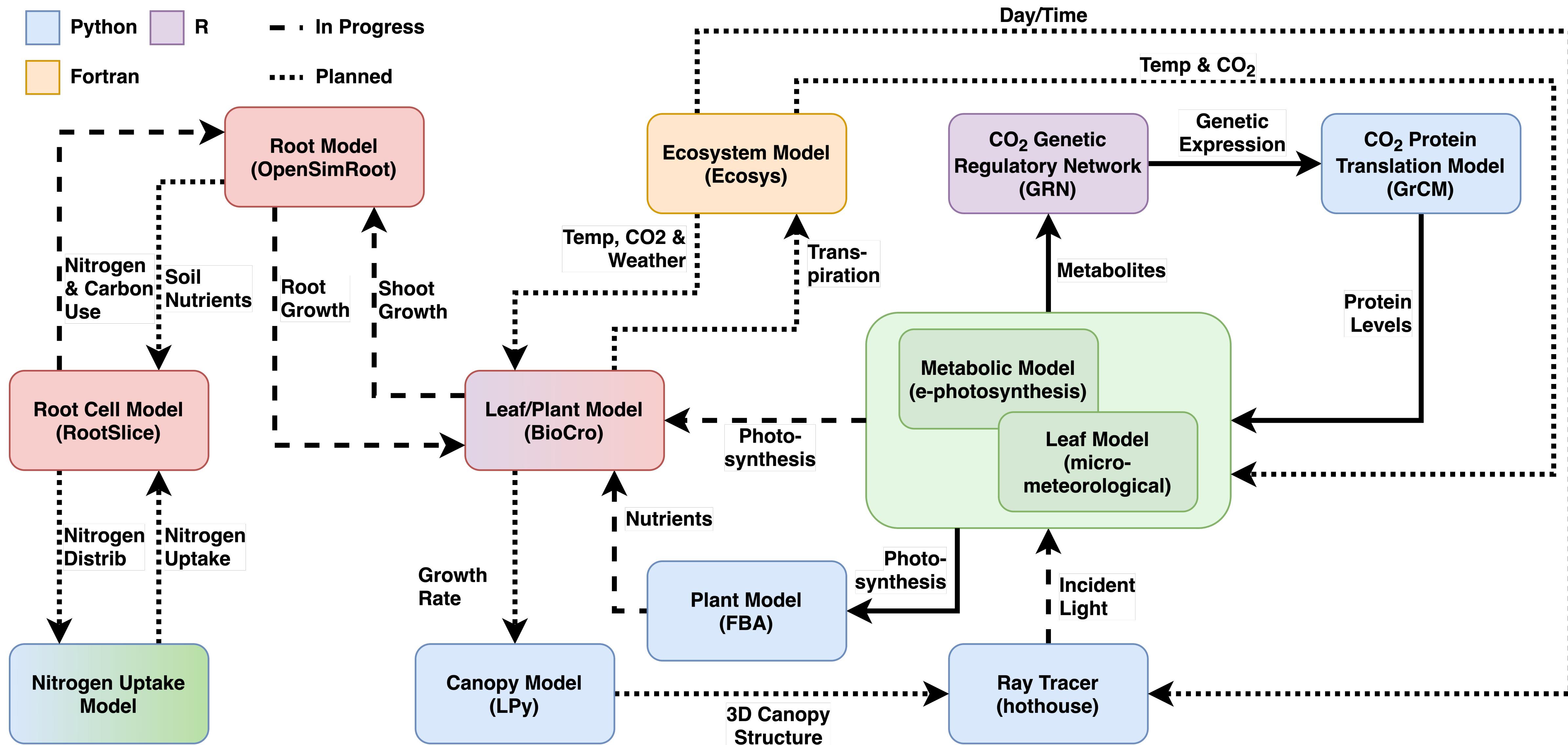
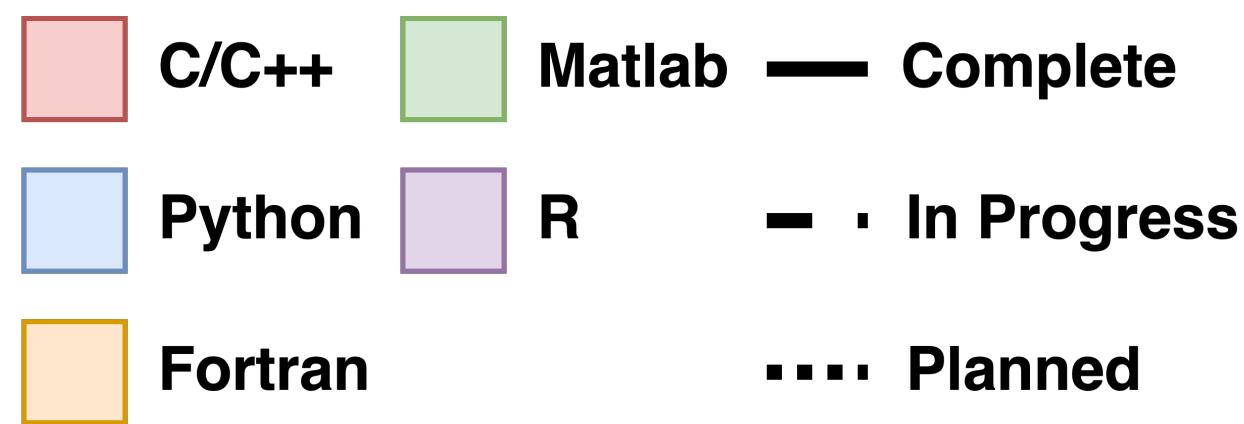


YGGDRASIL:

OPEN SOURCE PYTHON PACKAGE FOR CONNECTING MODELS ACROSS SCALES AND LANGUAGES

tag v1.6.2 pypi v1.6.2 build passing coverage 100% code style pep8
license BSD conda platforms linux-64 | win-64 | osx-64





LANGUAGES

PYTHON

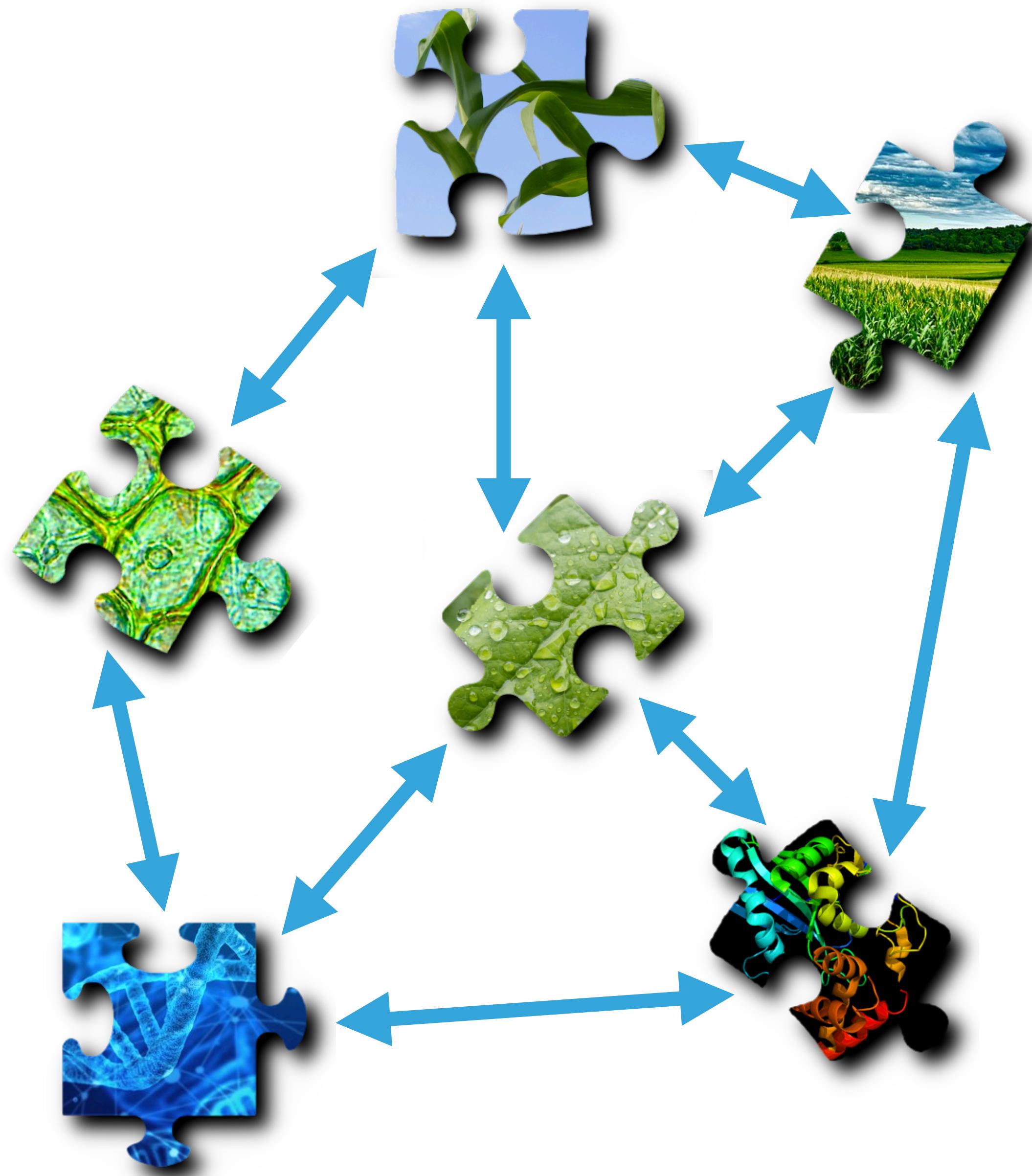
MATLAB

R

C/C++

FORTRAN ('03)





LANGUAGES

PYTHON

MATLAB

R

C/C++

FORTRAN ('03)

DOMAIN SPECIFIC LANGUAGES

SBML

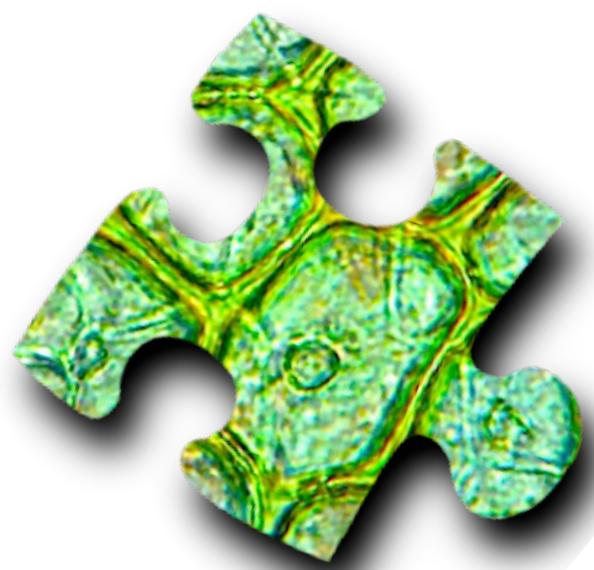
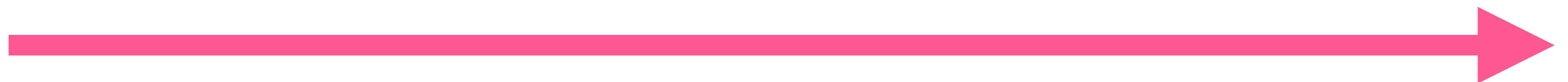
OPENSIMROOT XML

COMMUNICATION

PARALLEL EXECUTION



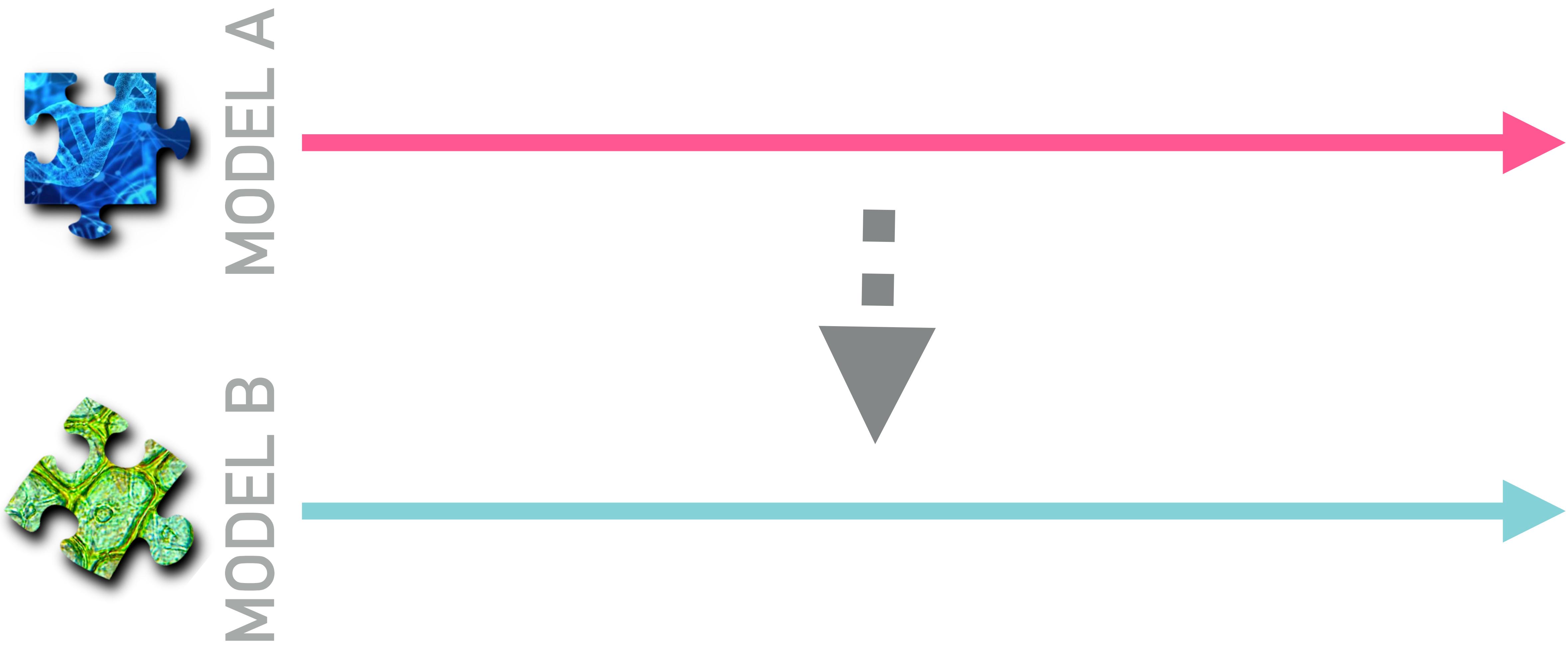
MODEL A



MODEL B



PARALLEL EXECUTION

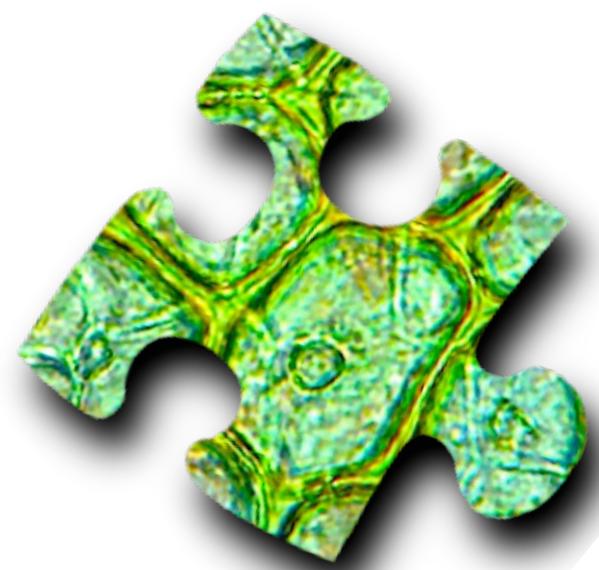


ASYNCHRONOUS COMMUNICATION

(SEND FIRST)



MODEL A



MODEL B

ASYNCHRONOUS COMMUNICATION

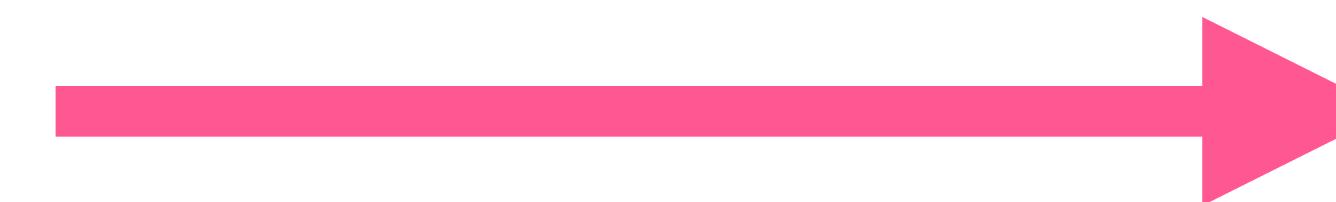
(SEND FIRST)



MODEL A



MODEL B



SEND

ASYNCHRONOUS COMMUNICATION

(SEND FIRST)



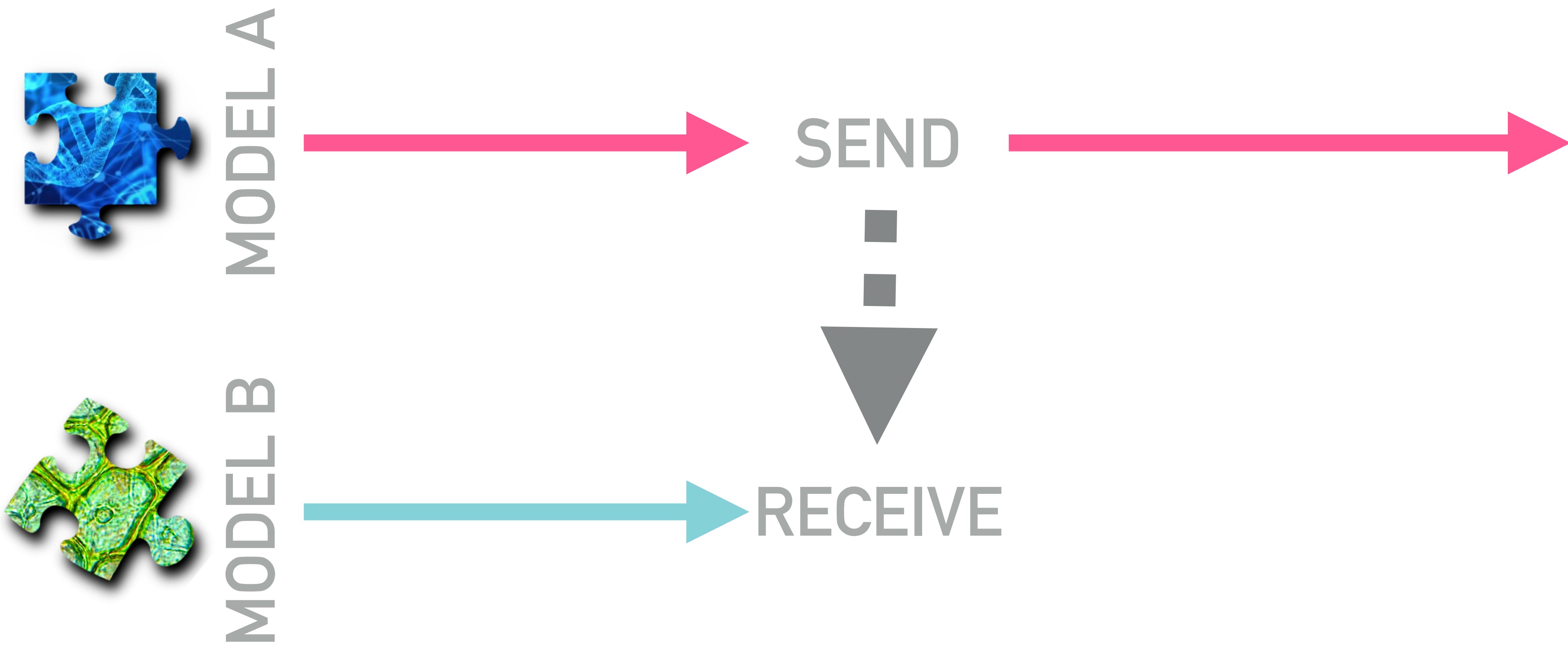
ASYNCHRONOUS COMMUNICATION

(SEND FIRST)



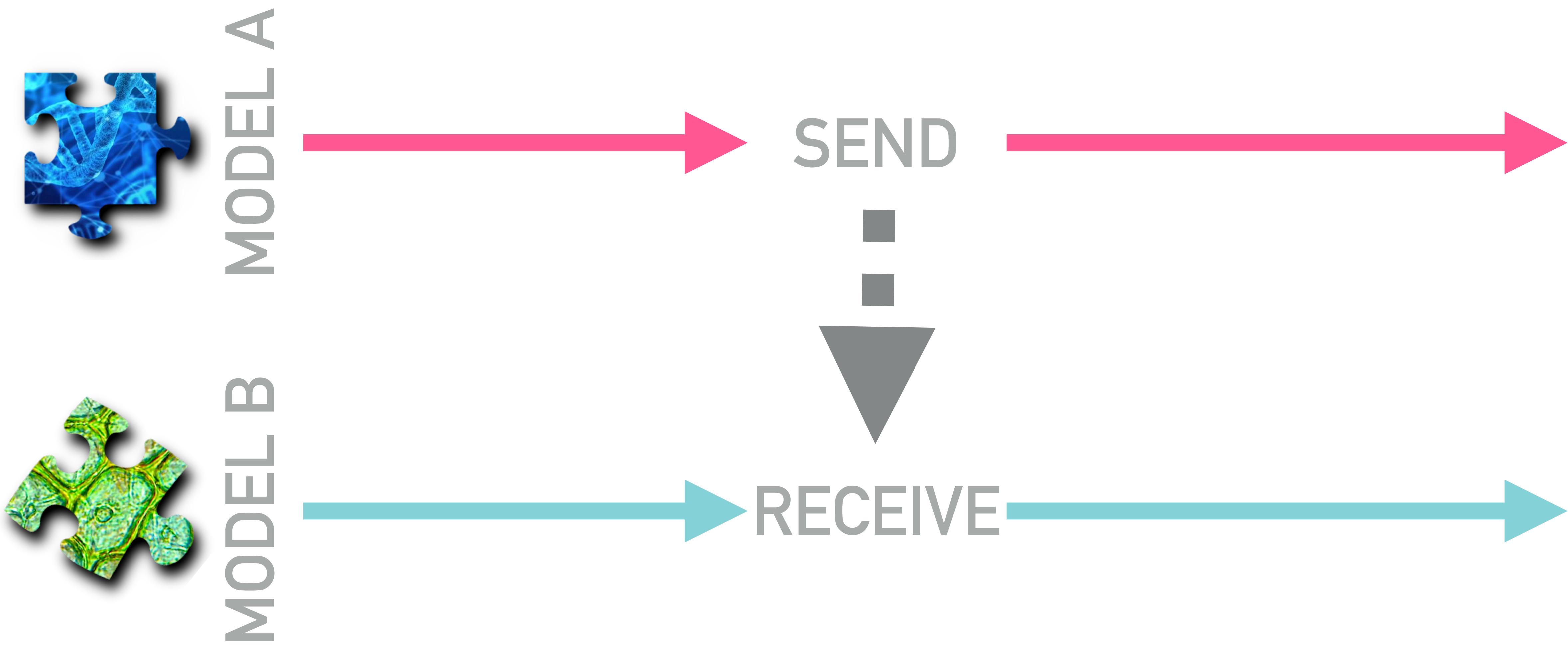
ASYNCHRONOUS COMMUNICATION

(SEND FIRST)



ASYNCHRONOUS COMMUNICATION

(SEND FIRST)

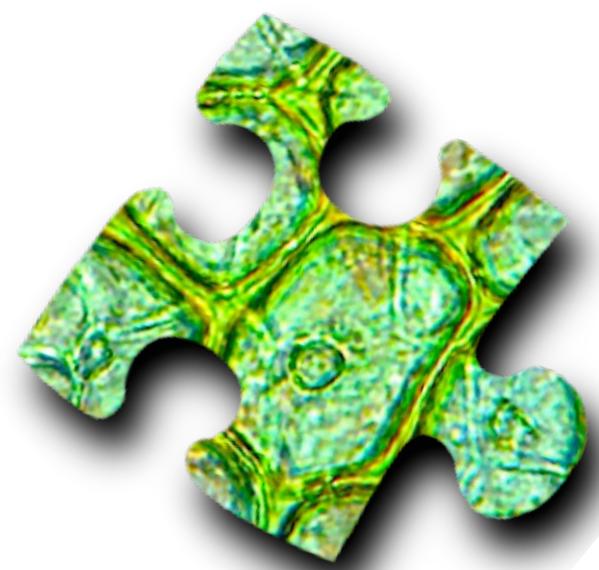


ASYNCHRONOUS COMMUNICATION

(RECEIVE FIRST)



MODEL A



MODEL B

ASYNCHRONOUS COMMUNICATION

(RECEIVE FIRST)



MODEL A



MODEL B



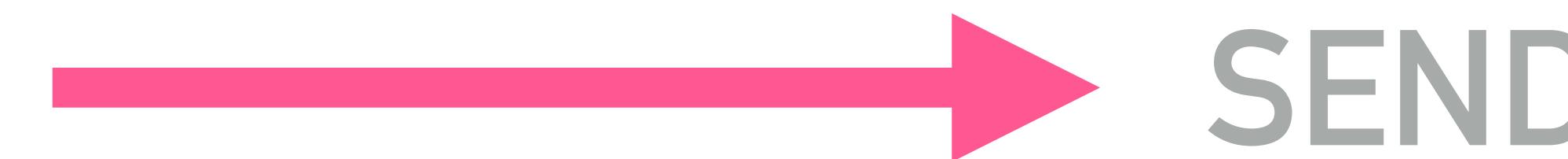
RECEIVE

ASYNCHRONOUS COMMUNICATION

(RECEIVE FIRST)



MODEL A

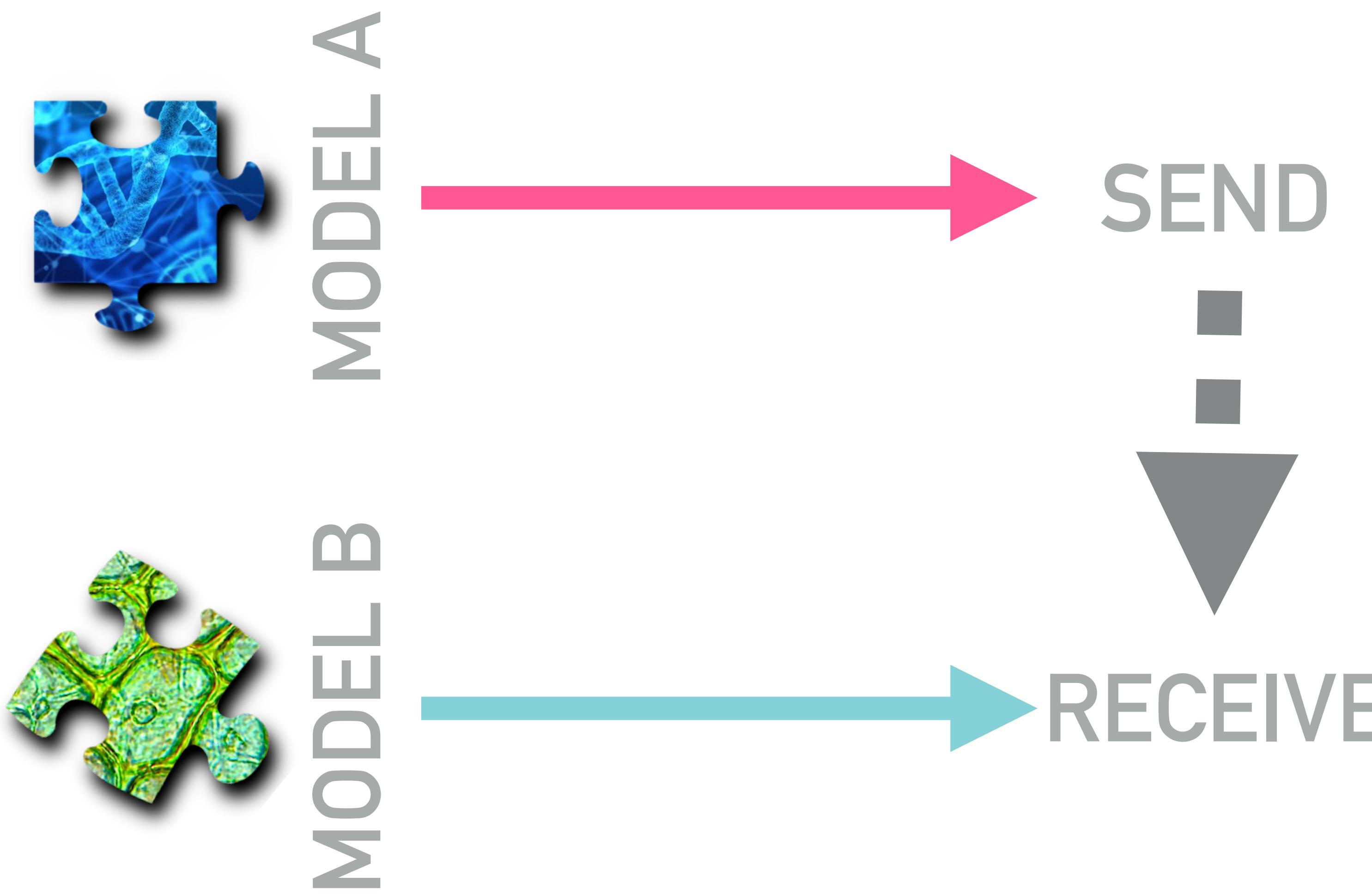


MODEL B



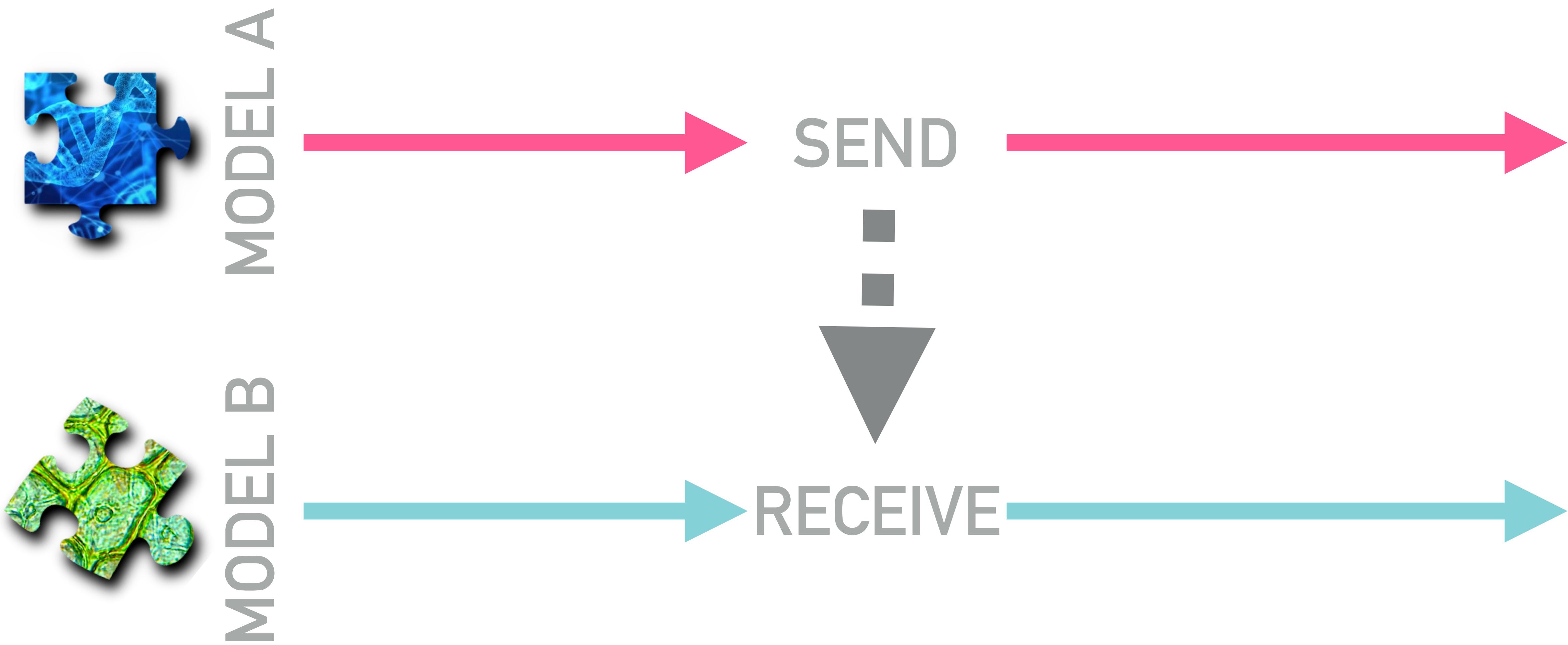
ASYNCHRONOUS COMMUNICATION

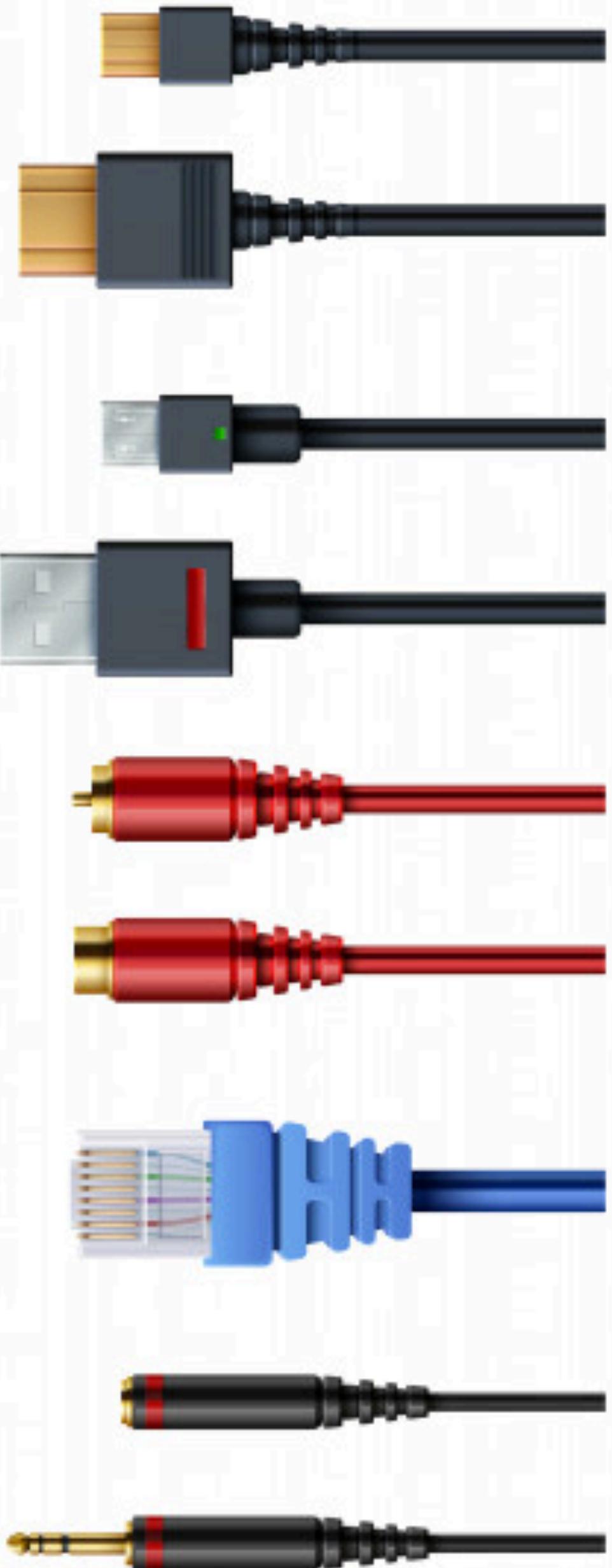
(RECEIVE FIRST)



ASYNCHRONOUS COMMUNICATION

(RECEIVE FIRST)





COMM:

COMMUNICATION OBJECT
ALLOWING MODELS TO
SEND/RECEIVE MESSAGES
TO/FROM OTHER MODELS

COMM CLASSES

INPUT

Receive messages from another model



OUTPUT

Send messages to another model



COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model



OUTPUT

INPUT



COMM CLASSES

INPUT

Receive messages from another model



OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses



CLIENT

Send requests to a server model and receive messages ("call" a server model)

COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses

CLIENT

Send requests to a server model and receive messages ("call" a server model)



CLIENT



SERVER

COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses

CLIENT

Send requests to a server model and receive messages ("call" a server model)



CLIENT



SERVER

COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

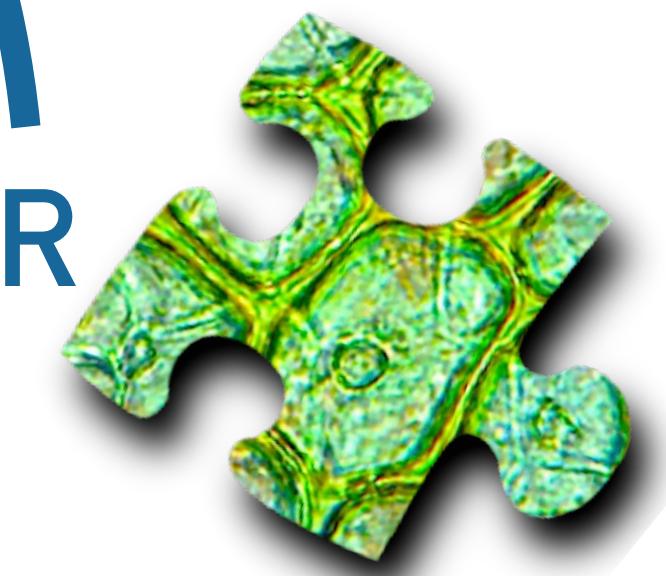
Receive requests from client models and send responses

CLIENT

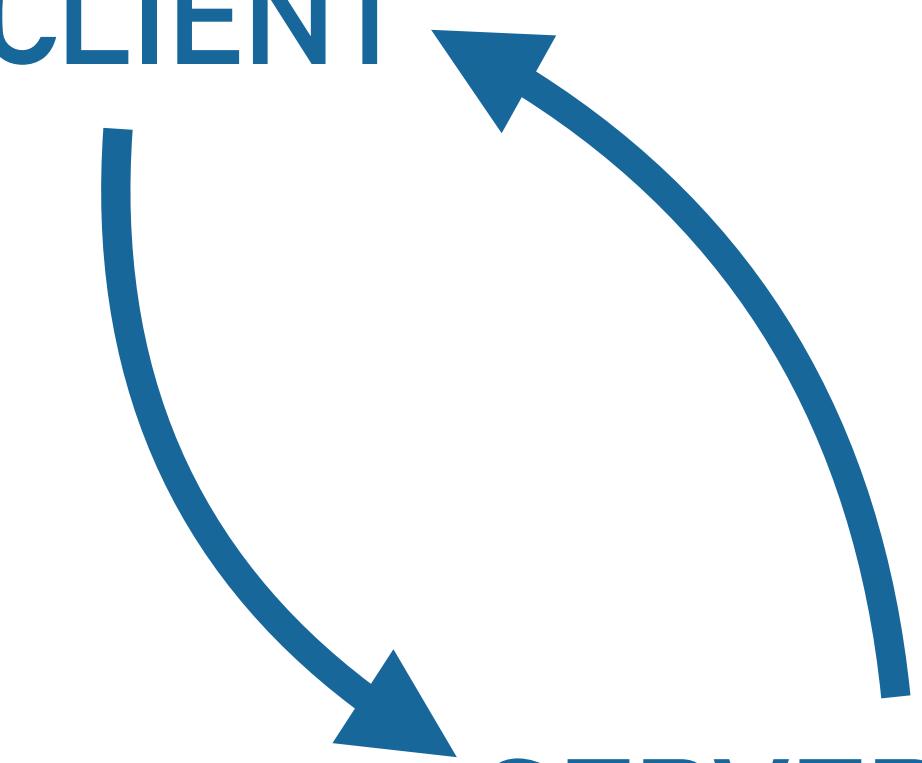
Send requests to a server model and receive messages ("call" a server model)



CLIENT



SERVER



COMM CLASSES

INPUT

Receive messages from another model



OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses



CLIENT

Send requests to a server model and receive messages ("call" a server model)

TIMESYNC

Send requests to a set of time-dependent models and receive time-dependent variables ("call" a time step synchronization)

COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses

CLIENT

Send requests to a server model and receive messages (“call” a server model)

TIMESYNC

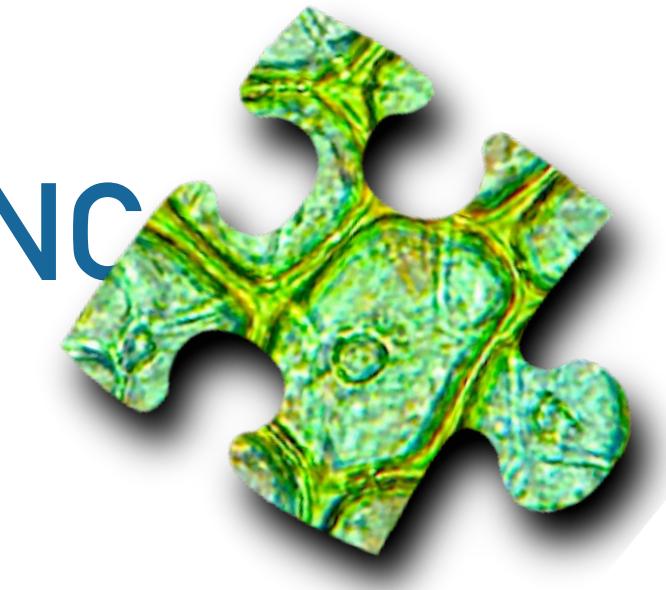
Send requests to a set of time-dependent models and receive time-dependent variables (“call” a time step synchronization)



TIMESYNC

SYNC

TIMESYNC



COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses

CLIENT

Send requests to a server model and receive messages ("call" a server model)

TIMESYNC

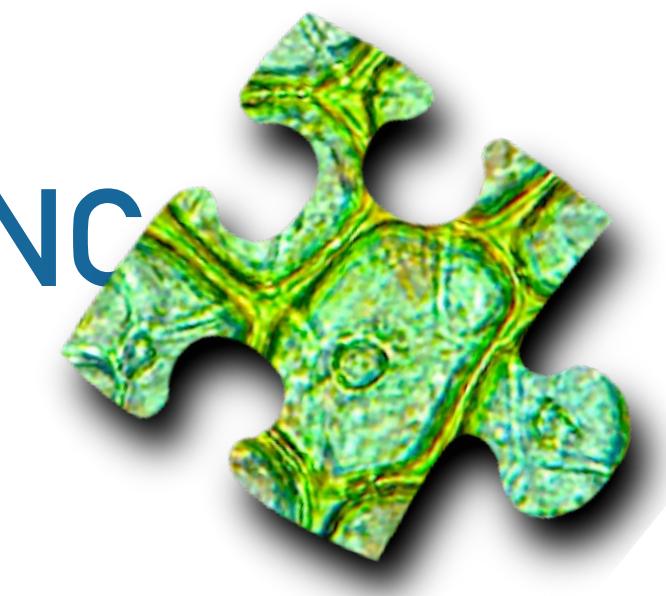
Send requests to a set of time-dependent models and receive time-dependent variables ("call" a time step synchronization)



TIMESYNC



SYNC



TIMESYNC

COMM CLASSES

INPUT

Receive messages from another model

OUTPUT

Send messages to another model

SERVER

Receive requests from client models and send responses

CLIENT

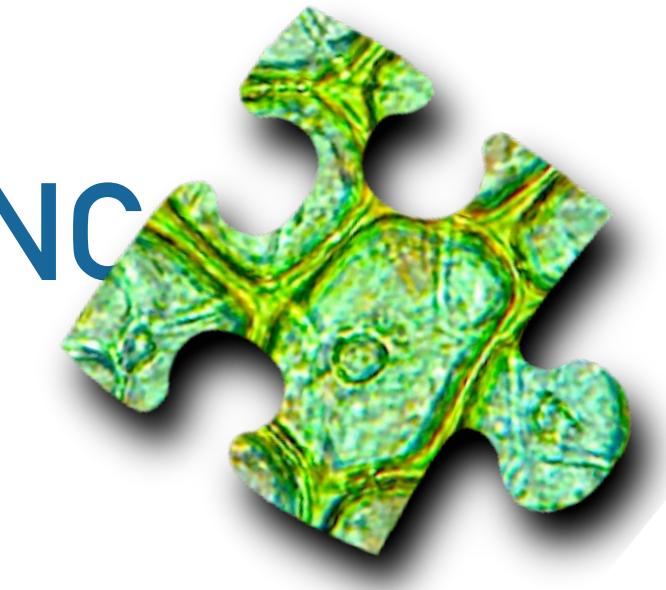
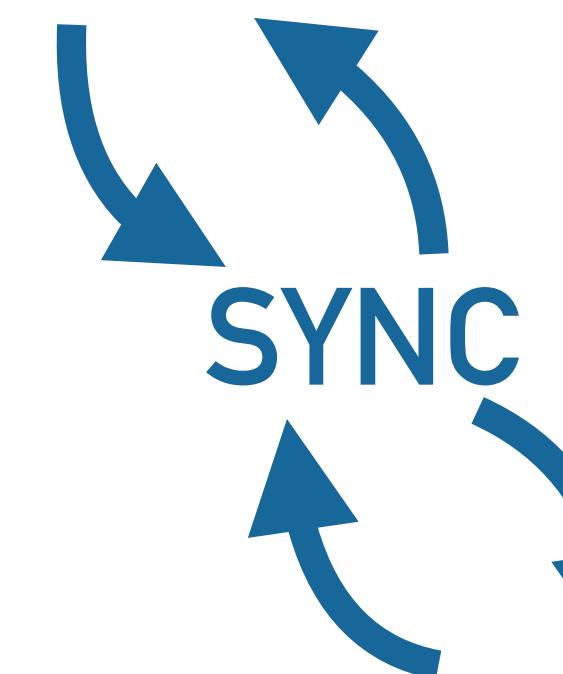
Send requests to a server model and receive messages ("call" a server model)

TIMESYNC

Send requests to a set of time-dependent models and receive time-dependent variables ("call" a time step synchronization)



TIMESYNC



TIMESYNC

COMM METHODS

IPC

Interprocess communication, only available on Unix (Linux & Mac)

ZEROMQ

Broker-less communication sockets via TCP, IPC, UDP, inproc, etc.; available on all OSs

RABBITMQ

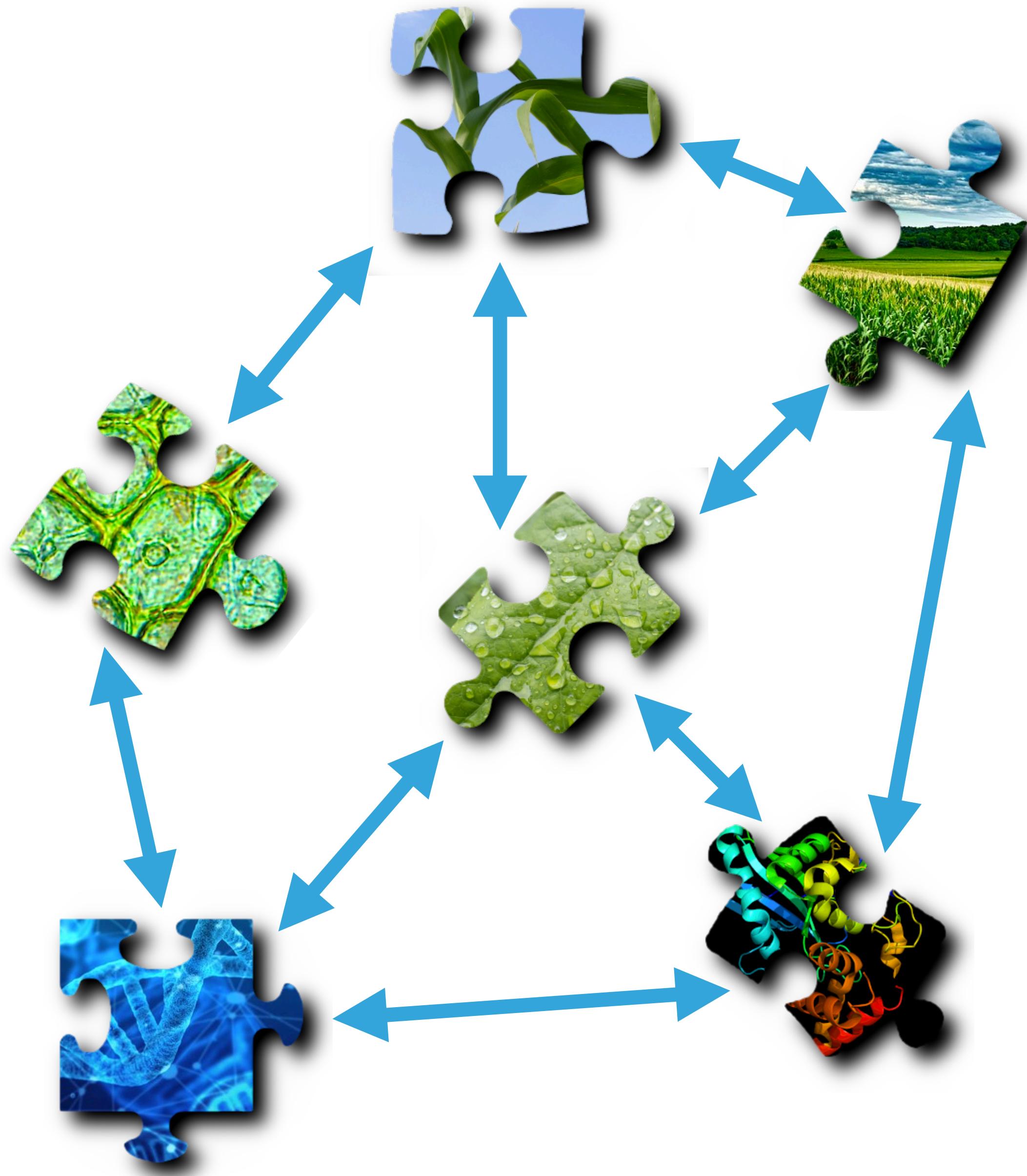
Brokered communication; requires a RabbitMQ server, but allows for more reliable communication with remote machines

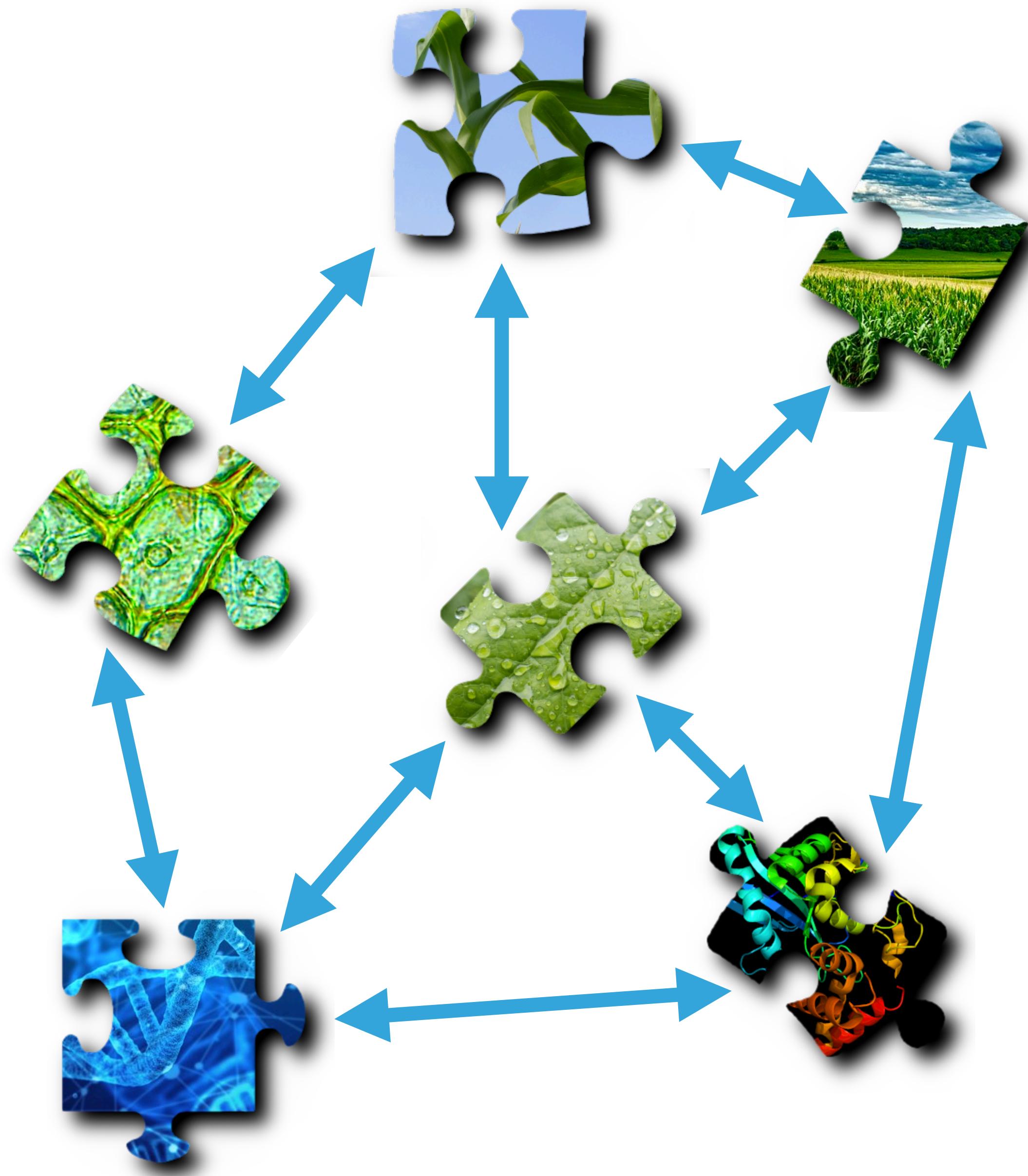
FILES

Slow due to interaction with the disk, but useful for input/output of parameters in a generic way that allows the same model to be used w/ files or other models

UNITS (AUTOMATED CONVERSION)

UNYT (PYTHON)
MATLAB SYMBOLIC UNITS
R UNITS





UNITS (AUTOMATED CONVERSION)

UNYT (PYTHON)

MATLAB SYMBOLIC UNITS

R UNITS

DATA FORMATS

SCALARS/ARRAYS

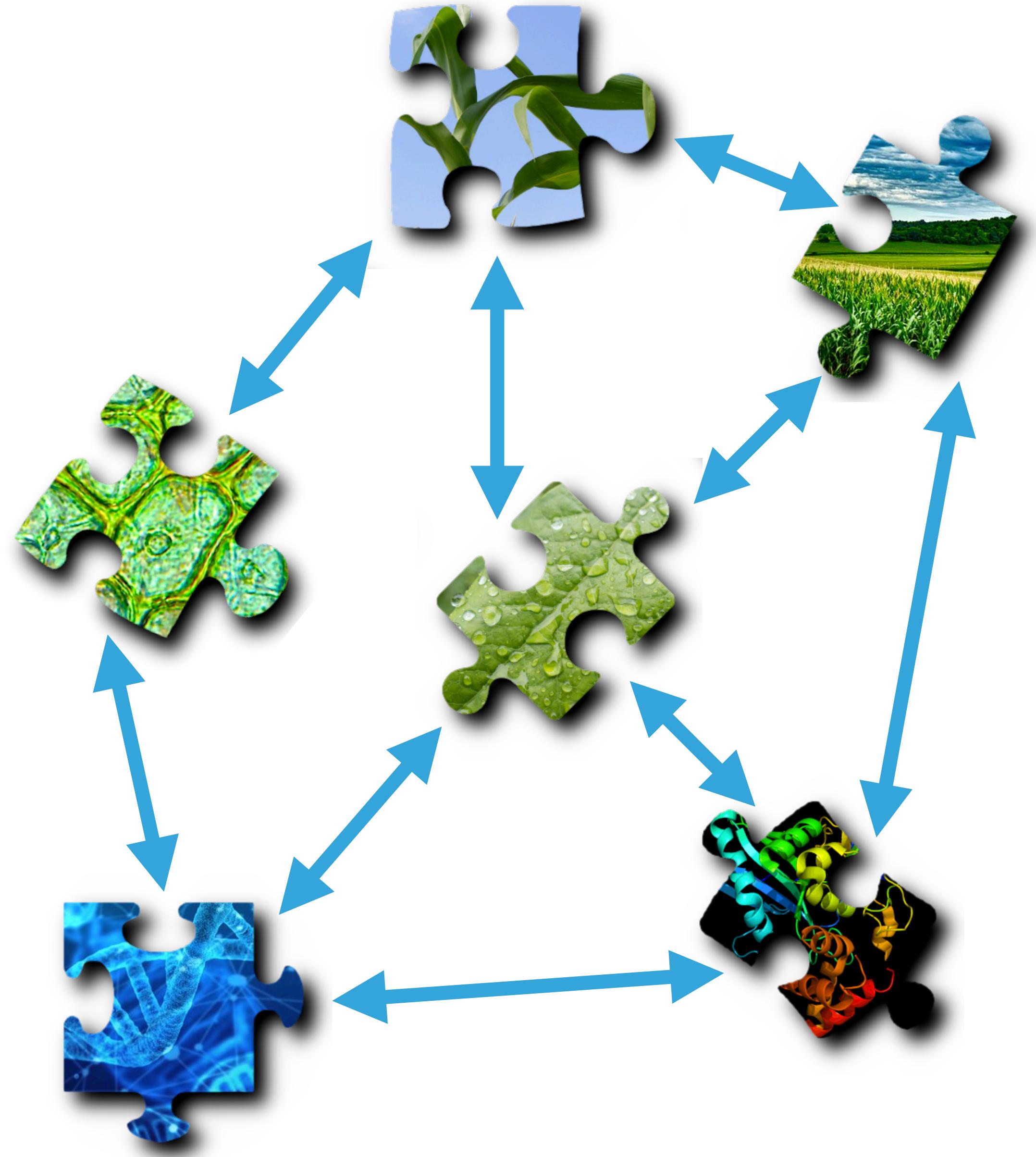
DELIMITED TABLES

PANDAS/R DATA FRAMES

PLY/OBJ

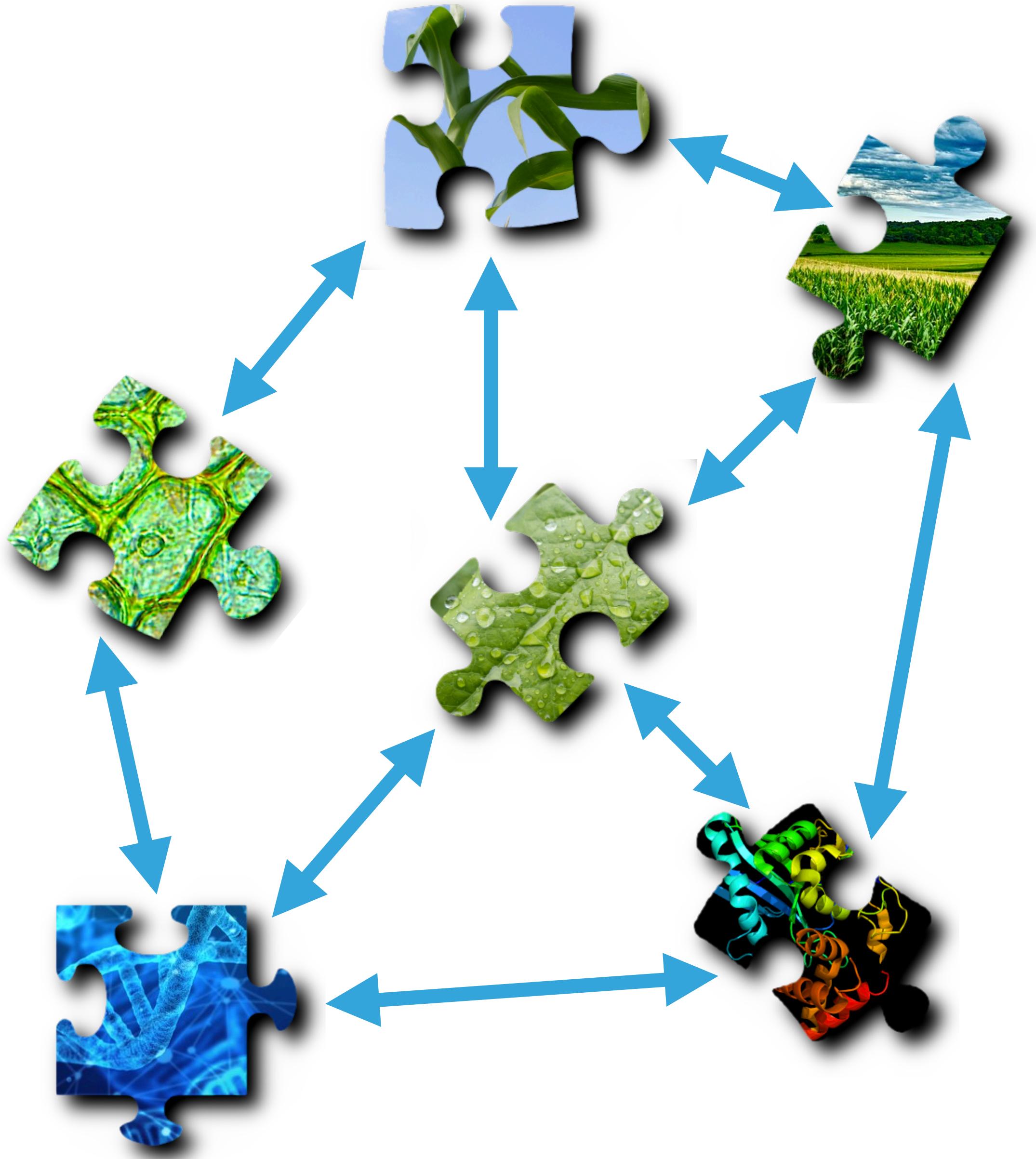
PYTHON PICKLES/
R .RDATA/MATLAB .MAT

EXECUTION



INTEGRATION:

NETWORK OF MODELS RUN USING YGGDRASIL



YAML SPECIFICATION:

INPUT FILE(S) TO
YGGDRASIL CONTAINING
INFO ON MODELS &
CONNECTIONS

YAML SPECIFICATION:

INPUT FILE(S) TO
YGGDRASIL CONTAINING
INFO ON MODELS &
CONNECTIONS

```
model:
  name: GrowthModel
  language: python
  args: ./src/growth.py
  inputs:
    - light
  outputs:
    - growth_rate
```

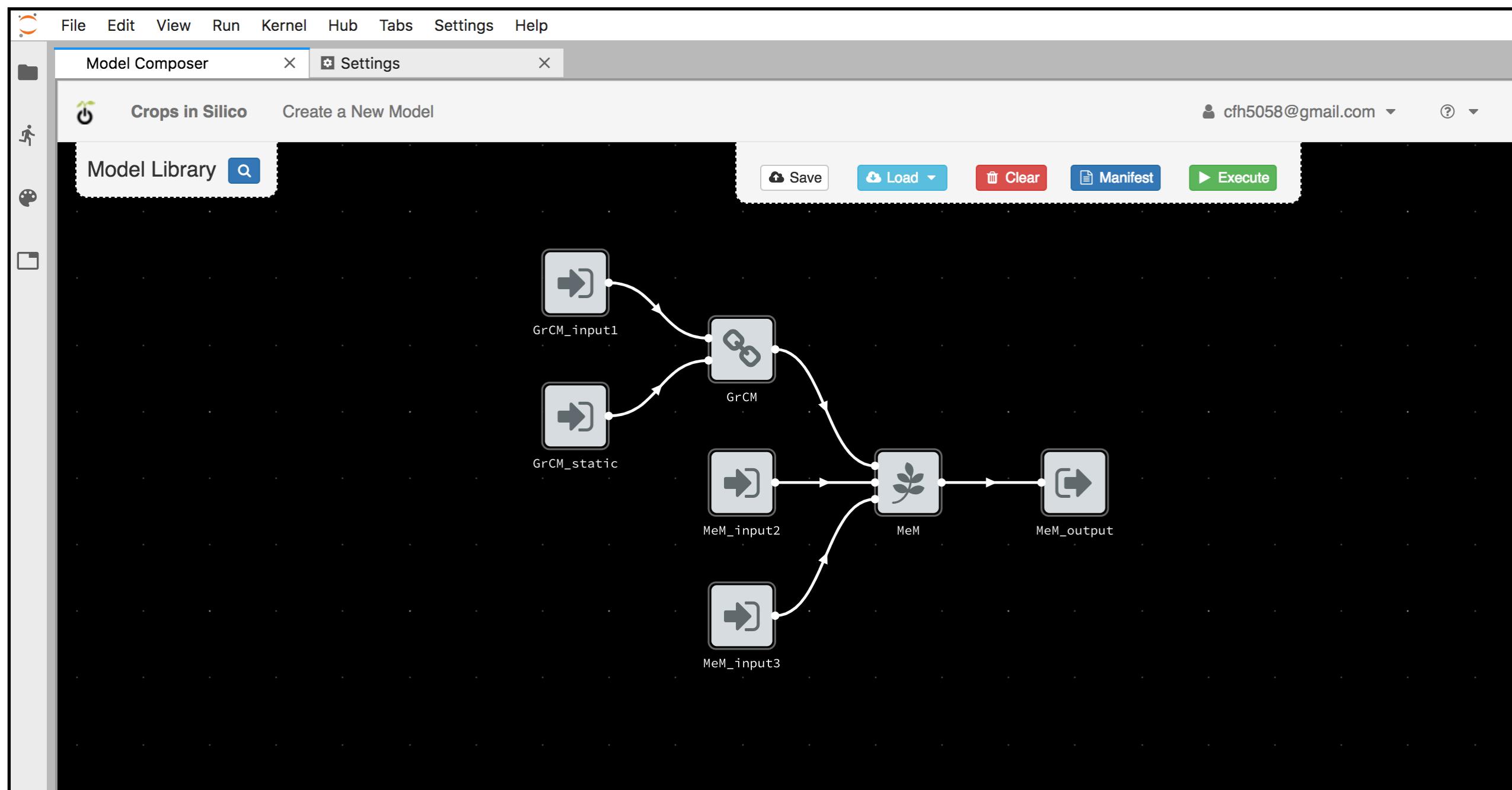
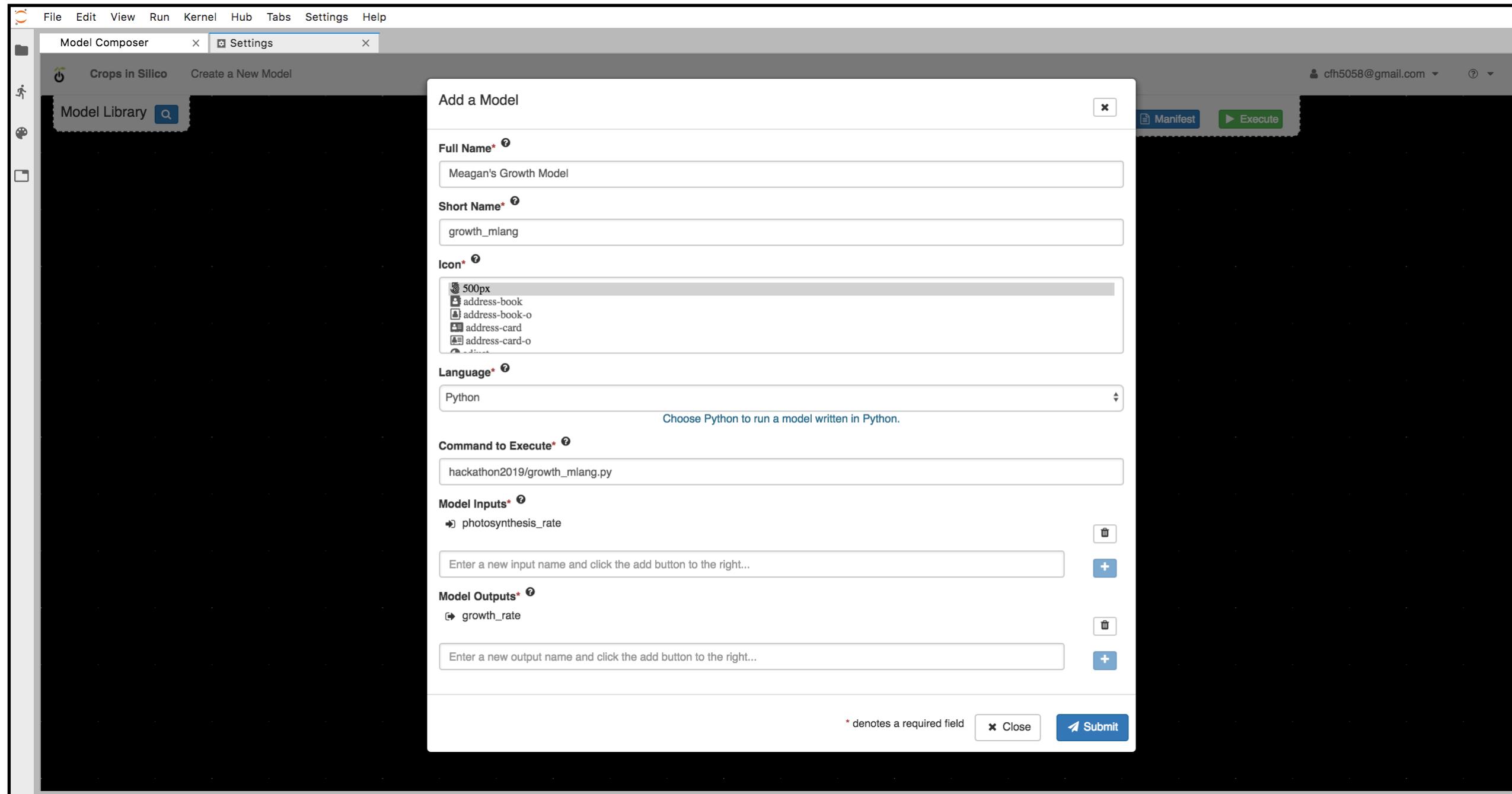
YAML SPECIFICATION:

INPUT FILE(S) TO
YGGDRASIL CONTAINING
INFO ON MODELS &
CONNECTIONS

```
model:  
  name: GrowthModel  
  language: python  
  args: ./src/growth.py  
  inputs:  
    - light  
  outputs:  
    - growth_rate  
  
connections:  
  - input: LightModel:light  
    output: GrowthModel:light  
  - input: growth_rate  
    output: ./Output/growth.txt  
    filetype: table  
    field_names: growth_rate
```

YAML SPECIFICATION:

INPUT FILE(S) TO
YGGDRASIL CONTAINING
INFO ON MODELS &
CONNECTIONS



GUI:
GRAPHICAL USER
INTERFACE IN
DEVELOPMENT FOR
YGGDRASIL CAN
GENERATE THE YAML

DEBUGGING

GITHUB ISSUES

Screenshot of a GitHub repository page for `cropsinsilico / CiS2021-hackathon`.

The repository has 4 issues, 0 stars, and 1 fork.

Code navigation buttons: Go to file, Add file, Code (selected).

Recent commits by `langmm`:

Commit	Message	Time Ago
66929e5	Update environment to trigger rebuild	12 hours ago
.github	Create issue_template.md	20 hours ago
images	Add images and use script directly to activate conda	13 hours ago
input	Add example of wrapping a model function	9 days ago
meshes	Fix bugs in the untested copies example	6 days ago
models	Update root model and add version of shoot for timesync example	6 days ago
yamls	Update root model and add version of shoot for timesync example	6 days ago
.gitignore	Update root model and add version of shoot for timesync example	6 days ago
00-intro.ipynb	Update images in notebooks	13 hours ago

About section:

- Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop
- Readme
- BSD-3-Clause License

Releases section:

- No releases published
- Create a new release

Packages section:

GITHUB ISSUES

Screenshot of a GitHub repository page for [cropsinsilico / CiS2021-hackathon](#).

The repository has 2 branches and 0 tags.

The Code tab is selected, indicated by a red circle.

The Issues tab is circled in red.

Recent commits:

Author	Commit Message	Time Ago
langmm	Update environment to trigger rebuild	12 hours ago
	.github Create issue_template.md	20 hours ago
	images Add images and use script directly to activate conda	13 hours ago
	input Add example of wrapping a model function	9 days ago
	meshes Fix bugs in the untested copies example	6 days ago
	models Update root model and add version of shoot for timesync example	6 days ago
	yamls Update root model and add version of shoot for timesync example	6 days ago
	.gitignore Update root model and add version of shoot for timesync example	6 days ago
	00-intro.ipynb Update images in notebooks	13 hours ago

About

Materials for the Hackathon at the 2021 Crops in Silico Symposium & Workshop

[Readme](#)

[BSD-3-Clause License](#)

Releases

No releases published

[Create a new release](#)

Packages

GITHUB ISSUES

The screenshot shows the GitHub Issues page for the repository `cropsinsilico/CiS2021-hackathon`. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. On the right side of the header are icons for notifications, a plus sign, and a user profile. Below the header, there are buttons for Unwatch (with 4 notifications), Star (0 stars), Fork (1 fork), and a link to the repository's page.

The main navigation bar below the header includes links for Code, Issues (which is selected and highlighted in red), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

A prominent callout box in the center of the page informs new contributors about GitHub's "Label issues and pull requests for new contributors" feature, stating that GitHub will help potential first-time contributors discover issues labeled with "good first issue". A "Dismiss" button is located in the top right corner of this box.

Below the callout box are filters for "Filters" (set to "is:issue is:open"), a search bar, and buttons for "Labels" (9), "Milestones" (0), and "New issue".

The main content area features a large exclamation mark icon and the heading "Welcome to issues!". A descriptive text explains that issues are used to track todos, bugs, feature requests, and more, and encourages users to create an issue.

GITHUB ISSUES

The screenshot shows the GitHub Issues page for the repository `cropsinsilico/CiS2021-hackathon`. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. On the right side of the header are icons for notifications, a plus sign, and a user profile. Below the header, there are buttons for Unwatch (with 4 notifications), Star (0 stars), Fork (1 fork), and a link to the repository's page.

The main content area features a navigation bar with links for Code, Issues (which is selected and highlighted with a red underline), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A modal window titled "Label issues and pull requests for new contributors" is displayed, stating that GitHub will help potential first-time contributors discover issues labeled with "good first issue". There is a "Dismiss" button in the top right corner of the modal.

Below the navigation bar, there are filters set to "is:issue is:open", a "Labels" section (9 labels), a "Milestones" section (0 milestones), and a prominent green "New issue" button. The "New issue" button is circled in red.

The main content area contains a large "Welcome to issues!" message with a circular icon containing an exclamation mark above it. Below this, a descriptive text explains that issues are used to track todos, bugs, feature requests, and more, and encourages users to create an issue.

GITHUB ISSUES

Screenshot of a GitHub repository page for `cropsinsilico / CiS2021-hackathon`.

The page shows the following navigation bar:

- Search or jump to... (with a search icon)
- Pull requests
- Issues (selected)
- Marketplace
- Explore

On the right side, there are icons for Unwatch (with a count of 4), Star (with a count of 1), and Fork (with a count of 1).

The main content area displays the issue creation form:

- Title:** (Input field)
- Write** and **Preview** buttons.
- Context (Environment):**
 - MyBinder instance
 - Local install
 - Docker container
- OS:** (Text input field)
- Browser:** (Text input field)
- Type of Issue:**
 - Jupyter notebook failed to open

On the right side of the form, there are configuration sections with gear icons:

- Assignees:** No one—assign yourself
- Labels:** None yet
- Projects:** None yet
- Milestone:** No milestone
- Linked pull requests:** Successfully merging a pull request may close this issue.

GITHUB ISSUES

Screenshot of the GitHub Issues interface for the repository `cropsinsilico / CiS2021-hackathon`.

The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. On the right, there are notifications, a user profile, and buttons for Unwatch (with 4 notifications), Star (1 star), Fork (1 fork), and Settings.

The main navigation bar below shows tabs for Code, Issues (selected), Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings.

The issue creation form on the left contains fields for Title, Write, Preview (circled in red), Context (Environment) (checkboxes for MyBinder instance, Local install, Docker container), OS:, Browser:, Type of Issue (checkbox for Jupyter notebook failed to open), and a file upload area with a placeholder for a screenshot.

The right sidebar displays settings for Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), Milestone (No milestone), and Linked pull requests (Successfylly merging a pull request may close this issue). There are also sections for Comments, Activity, and Recent changes.

GITHUB ISSUES

The screenshot shows a GitHub repository page for 'cropsinsilico / CiS2021-hackathon'. The 'Issues' tab is active. A new issue is being created, with the title field containing 'Title'. The issue body contains the following placeholder text:

```
<!-- Provide some information about how you are accessing the hackathon materials. -->
<!-- To mark a check box, replace the space inside the brackets with an X, e.g. [X] -->
## Context (Environment)
* [ ] MyBinder instance
* [ ] Local install
* [ ] Docker container

<!-- What operating system are you on? (e.g. Windows, Mac, Linux) -->
#### OS:

<!-- What web browser are you using to access the notebooks? -->
#### Browser:

<!-- Tell us what type of issue you are having -->
```

On the right side, there are configuration sections for Assignees, Labels, Projects, Milestone, and Linked pull requests. The 'Assignees' section says 'No one—assign yourself'. The 'Labels' section says 'None yet'. The 'Projects' section says 'None yet'. The 'Milestone' section says 'No milestone'. The 'Linked pull requests' section says 'Successfully merging a pull request may close this issue.' and 'None yet'.



Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. We will be using Github Issues to track problems encountered during the hackathon.
- Try launching a mybinder instance by clicking on this icon (or the link below). It may take a few moments to initialize. If you encounter an error, open an issue and try with another browser. If you still cannot launch the binder, follow the instruction [here](#) for downloading and installing the materials locally.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [yggdrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)

OTHER RESOURCES



Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. We will be using Github Issues to track problems encountered during the hackathon.
- Try launching a mybinder instance by clicking on this icon (or the link below). It may take a few moments to initialize. If you encounter an error, open an issue and try with another browser. If you still cannot launch the binder, follow the instruction [here](#) for downloading and installing the materials locally.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [yggdrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)

OTHER RESOURCES

README.md

 launch binder

Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. This will help you track your progress and share your work with others.
- Try launching a mybinder instance by clicking on this  icon. It may take a few moments to initialize. If you encounter an error, open an issue and try to reproduce it. If you're having trouble launching the binder, follow the instruction [here](#) for downloading and installing the required software.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [yggdrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)

yggdrasil
1.6.1

Search docs

CONTENTS:

- Overview
- Installation
- Getting started
- Formatted I/O
- Server/Client I/O
- Autowrapping Model Functions
- Notes on Autowrapping C/C++ Model Functions
- Conditional I/O
- Transformed I/O
- Timestep Synchronization
- YAML Files
- Configuration Files
- Units
- C-Style Format Strings

Debugging

- Tips for Debugging
- Possible Errors
- OpenMP Threading in Models
- Examples

OTHER RESOURCES

[View page source](#)

Debugging

Tips for Debugging

1. *Look at the full output.* The final error raised by yggdrasil may not contain all of the information provided by errors that were raised within a model due to limitations of error forwarding between the different languages. It is important to look at the full output from a failed run. Usually the first error encountered or the error raised within the model's language will be the most relevant and be the most useful for debugging.
2. *Check for known errors.* The list below includes several errors that have already been encountered by yggdrasil users and the method used to solve the issue.
3. *Turn on debugging log messages.* This will increase the number of log messages greatly and help you track down any issues. Debug messages can be enabled by setting the `ygg` and `client` debug options in your config file to `DEBUG` (see [Configuration Options](#) for details on the location of the user config file and additional logging options).
4. *Trace the flow of data.* Use the debug messages to trace the flow of data from one model to the next and determine where the failure is occurring.
5. *Check `|yggdrasil| summary`.* yggdrasil includes a command line utility, `ygginfo` that will print out relevant information about yggdrasil, the languages it supports, and the operating system. This information can be useful for running down conflicting dependencies or determining why yggdrasil thinks a language is or isn't installed. Additional information about the system can be displayed by adding the `--verbose` flag, including the current conda environment information (if you are using a conda environment) and information about the current R installation (if R is installed). This information should be included in any Github issues opened related to bugs in order to help us assist you.

DEMO TIME!



Starting repository: [cropsinsilico/CiS2021-hackathon/HEAD](#)

You can connect with the Binder community in the [Jupyter community forum](#).

Build logs

[hide](#)

libgfortran-ng-9.3.0	22 KB	#####	100%
gitdb-4.0.7	46 KB	#####	100%
r-jsonlite-1.7.2	462 KB	#####	100%
smmap-3.0.5	22 KB	#####	100%
czmq-4.2.1	540 KB	#####	100%
networkx-2.5.1	1.2 MB	#####	100%
gcc_linux-64-9.3.0	23 KB	#####	100%
openjpeg-2.4.0	444 KB	#####	100%
r-rappdirs-0.3.3	50 KB	#####	100%

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) [⟳](#)

<input type="checkbox"/>	0	▼	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images				33 minutes ago	
<input type="checkbox"/>	 input				33 minutes ago	
<input type="checkbox"/>	 meshes				33 minutes ago	
<input type="checkbox"/>	 models				33 minutes ago	
<input type="checkbox"/>	 yaml				33 minutes ago	
<input type="checkbox"/>	 00-intro.ipynb				33 minutes ago	457 kB
<input type="checkbox"/>	 01-connections.ipynb				33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb				33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb				33 minutes ago	3.56 kB

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) 

<input type="checkbox"/> 0	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images		33 minutes ago	
<input type="checkbox"/>	 input		33 minutes ago	
<input type="checkbox"/>	 meshes		33 minutes ago	
<input type="checkbox"/>	 models		33 minutes ago	
<input type="checkbox"/>	 yaml		33 minutes ago	
<input checked="" type="checkbox"/>	 00-intro.ipynb		33 minutes ago	457 kB
<input type="checkbox"/>	 01-connections.ipynb		33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb		33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb		33 minutes ago	3.56 kB



Introduction

(NOTE: This notebook is intended for use with the slides found [here](#)).

This is a Jupyter notebook. It allows us to run code (in this case Python) alongside text in different "cells". This cell is a markdown cell that can display text and html, the next cell is a code cell.

In the code cells (prefixed by `In []:`), you can assign variables, perform calculations or call external functions/classes. You can run code cells by selecting the cell (so that a blue or green box appears around it) and then clicking the run button (located at the top of the page) or pressing `Shift+Enter` together. Then a number will appear inside the brackets indicating the order of when the cell was executed.

Output from the cell will be displayed below it with the `Out [#]:` prefix where the number in the brackets indicates the input cell that generated it.

```
In [ ]: x = 1
y = 3
z = (x + y)**3
z
```

Any Python code can be used, and we can import external packages as well just like in Python scripts. Cells can also use any variables created in any previously executed cell. The cell below imports some tools that will be used in the rest of this notebook.



TEXT
CELL

Introduction

(NOTE: This notebook is intended for use with the slides found [here](#)).

This is a Jupyter notebook. It allows us to run code (in this case Python) alongside text in different "cells". This cell is a markdown cell that can display text and html, the next cell is a code cell.

In the code cells (prefixed by `In []:`), you can assign variables, perform calculations or call external functions/classes. You can run code cells by selecting the cell (so that a blue or green box appears around it) and then clicking the run button (located at the top of the page) or pressing `Shift+Enter` together. Then a number will appear inside the brackets indicating the order of when the cell was executed.

Output from the cell will be displayed below it with the `Out [#]:` prefix where the number in the brackets indicates the input cell that generated it.

```
In [ ]: x = 1
y = 3
z = (x + y)**3
z
```

Any Python code can be used, and we can import external packages as well just like in Python scripts. Cells can also use any variables created in any previously executed cell. The cell below imports some tools that will be used in the rest of this notebook.



Introduction

(NOTE: This notebook is intended for use with the slides found [here](#)).

This is a Jupyter notebook. It allows us to run code (in this case Python) alongside text in different "cells". This cell is a markdown cell that can display text and html, the next cell is a code cell.

In the code cells (prefixed by `In []:`), you can assign variables, perform calculations or call external functions/classes. You can run code cells by selecting the cell (so that a blue or green box appears around it) and then clicking the run button (located at the top of the page) or pressing `Shift+Enter` together. Then a number will appear inside the brackets indicating the order of when the cell was executed.

Output from the cell will be displayed below it with the `Out [#]:` prefix where the number in the brackets indicates the input cell that generated it.

In []:

```
x = 1
y = 3
z = (x + y)**3
z
```

CODE
CELL

Any Python code can be used, and we can import external packages as well just like in Python scripts. Cells can also use any variables created in any previously executed cell. The cell below imports some tools that will be used in the rest of this notebook.

NOTEBOOK INTRO

Code cells can contain any valid Python code

Run cells by holding shift and pressing enter
(shift + enter)

In []:

```
x = 1
y = 3
z = (x + y)**3
z
```

Code cells can contain any valid Python code

Run cells by holding shift and pressing enter
(shift + enter)

In [1]:

```
x = 1
y = 3
z = (x + y)**3
z
```

Out[1]: 64

Code cells can contain any valid Python code
Run cells by holding shift and pressing enter
(shift + enter)

In [1]:

```
x = 1
y = 3
z = (x + y)**3
z
```

Out[1]: 64

Output appears below
Number in bracket is the order of execution
("∗" indicates the cell is still running)

We need some tools!

trimesh - package for loading/displaying meshes in the notebook

yggdrasil - the method for running yggdrasil integration

In [2]:

```
from yggdrasil import tools # Displaying syntax
from yggdrasil.runner import run # Running integ.
import trimesh # Load & display 3D meshes
```

We need some tools!

trimesh - package for loading/displaying meshes in the notebook

yggdrasil - the method for running yggdrasil integration

In [2]: `from yggdrasil import tools # Displaying syntax
from yggdrasil.runner import run # Running integ.
import trimesh # Load & display 3D meshes`

We need some tools!

trimesh - package for loading/displaying meshes in the notebook

yggdrasil - the method for running yggdrasil integration

In [2]: `from yggdrasil import tools # Displaying syntax
from yggdrasil.runner import run # Running integ.
import trimesh # Load & display 3D meshes`

No output, so nothing appears below

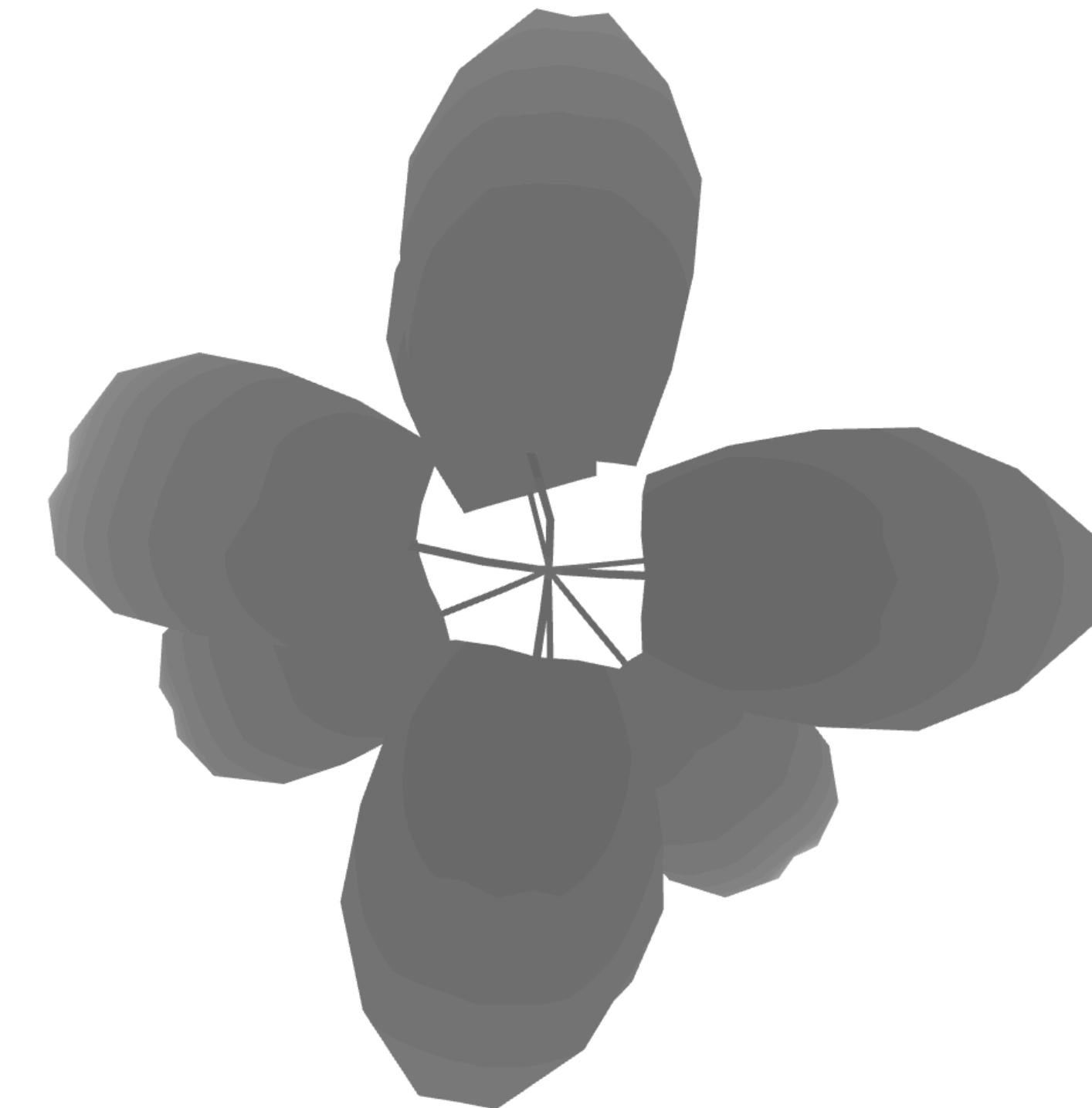
Lets load & display some 3D mesh data

```
In [ ]: fname = 'meshes/plants-2.obj'  
mesh = trimesh.load_mesh(fname)  
mesh.show()
```

Lets load & display some 3D mesh data

```
In [3]: fname = 'meshes/plants-2.obj'  
mesh = trimesh.load_mesh(fname)  
mesh.show()
```

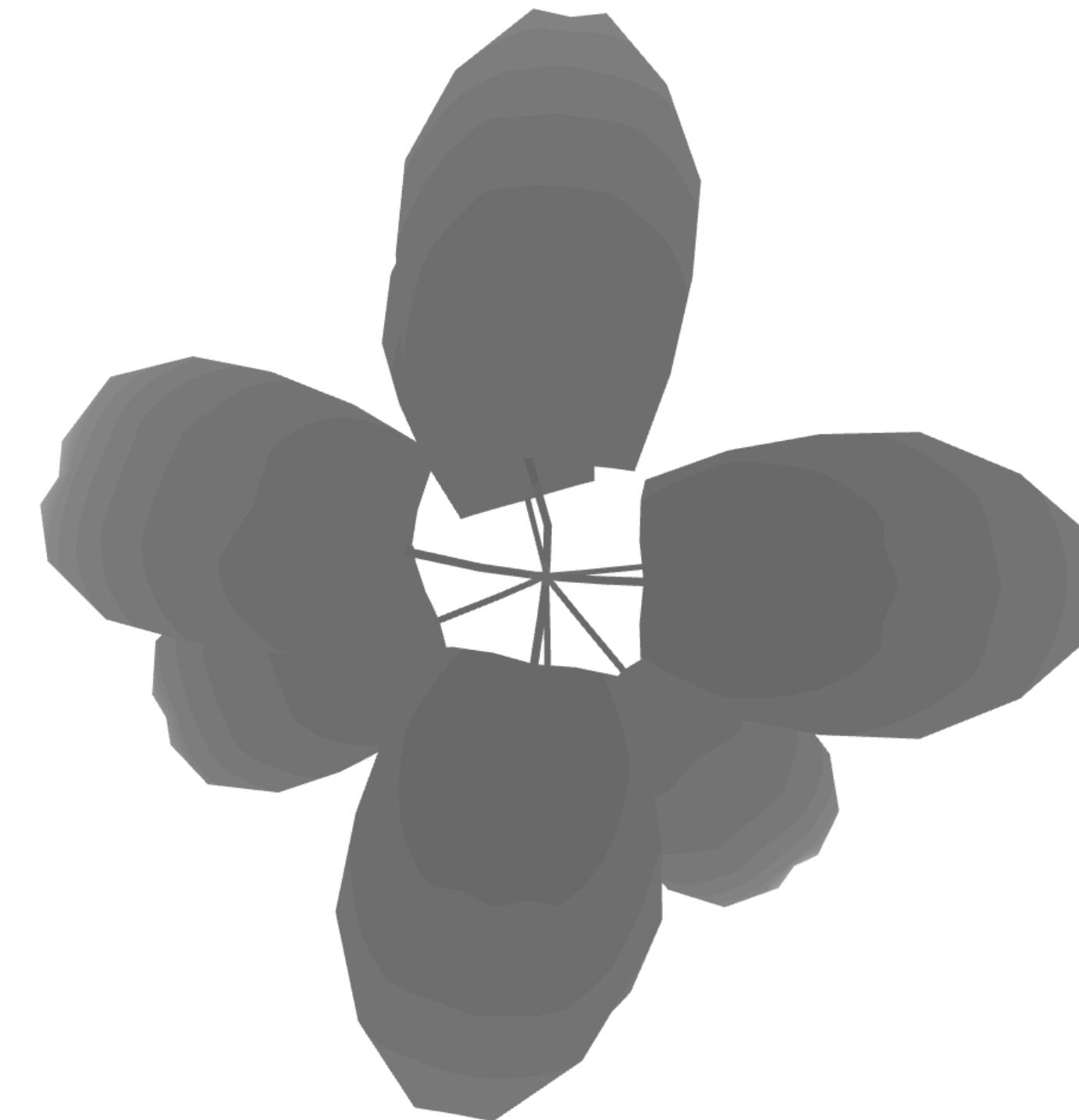
Out[3]:



Lets load & display some 3D mesh data

```
In [3]: fname = 'meshes/plants-2.obj'  
mesh = trimesh.load_mesh(fname)  
mesh.show()
```

Out[3]:



Click and drag
to move mesh

INTEGRATING MODELS AS FUNCTIONS

```
In [4]: tools.display_source('models/light_v0.py', number_lines=True)

file: models/light_v0.py
=====
1: import numpy as np
2: from yggdrasil import units
3:
4:
5: def light(doy, height):
6:     """Compute the intensity of light.
7:
8:     Args:
9:         doy (float): Day of year.
10:        height (float): Distance from ground in cm.
11:
12:    Returns:
13:        float: Intensity of light in ergs cm^-2 s^-1.
14:
15:    """
16:    # Define parameters that are static across a run
17:    amplitude = units.add_units(80.0, 'ergs cm^-3 s^-1')
18:    doy_offset = units.add_units(0.0, 'days')
19:
20:    # Calculate intensity
21:    intensity = (
22:        amplitude * height *
23:        (1.0 + np.sin(2.0 * np.pi * (doy - doy_offset) /
24:                      units.add_units(365.0, 'days'))))
25:
26:    return intensity
```

```
In [5]: tools.display_source('yamls/light_v0_python.yml', number_lines=True)
```

```
file: yamls/light_v0_python.yml
=====
1: model:
2:   name: light
3:   language: python
4:   args: ../models/light_v0.py
5:   function: light
```

```
In [5]: tools.display_source('yamls/light_v0_python.yml', number_lines=True)
```

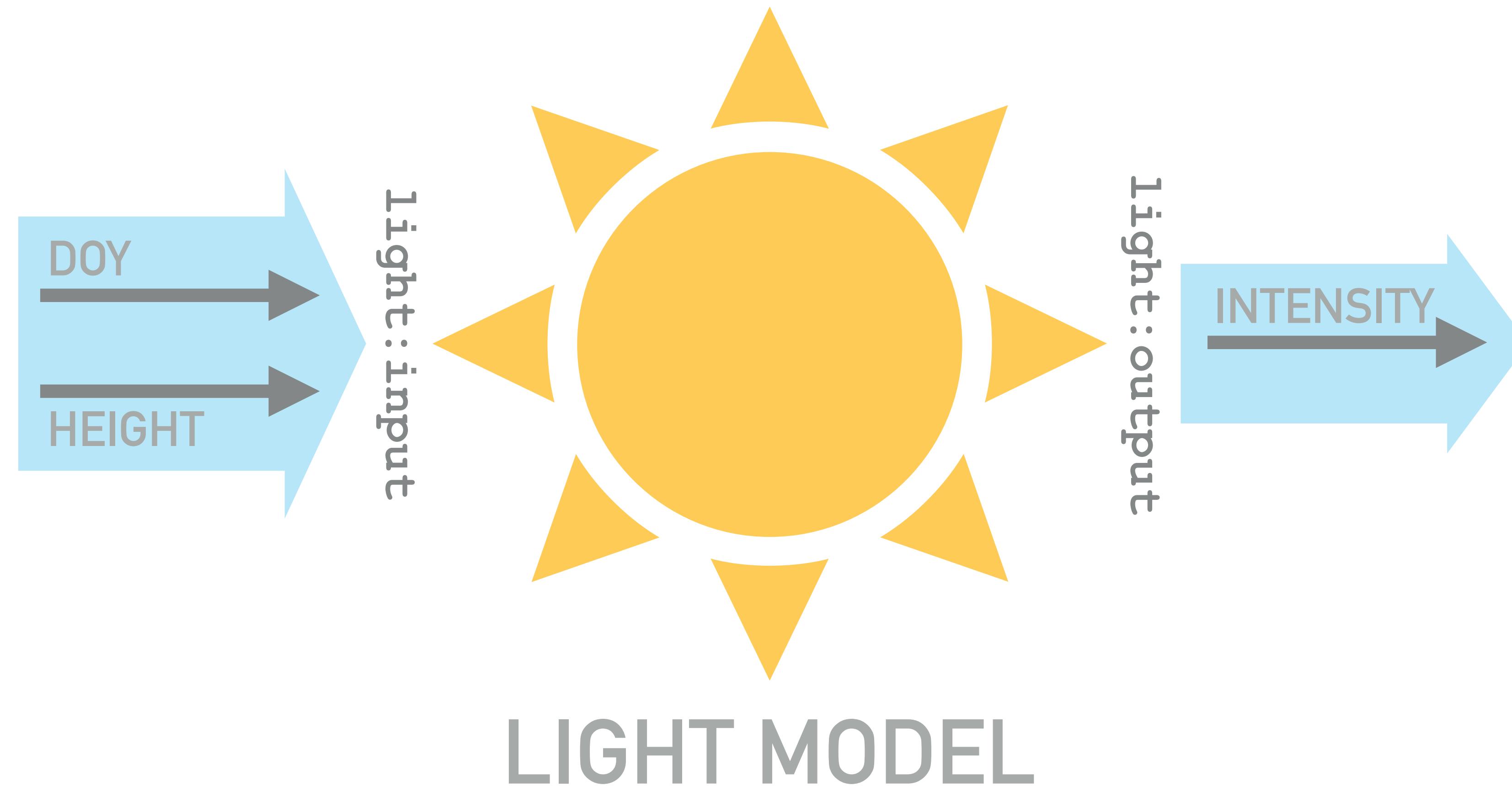
```
file: yamls/light_v0_python.yml
=====
1: model:
2:   name: light
3:   language: python
4:   args: ../models/light_v0.py
5:   function: light
```

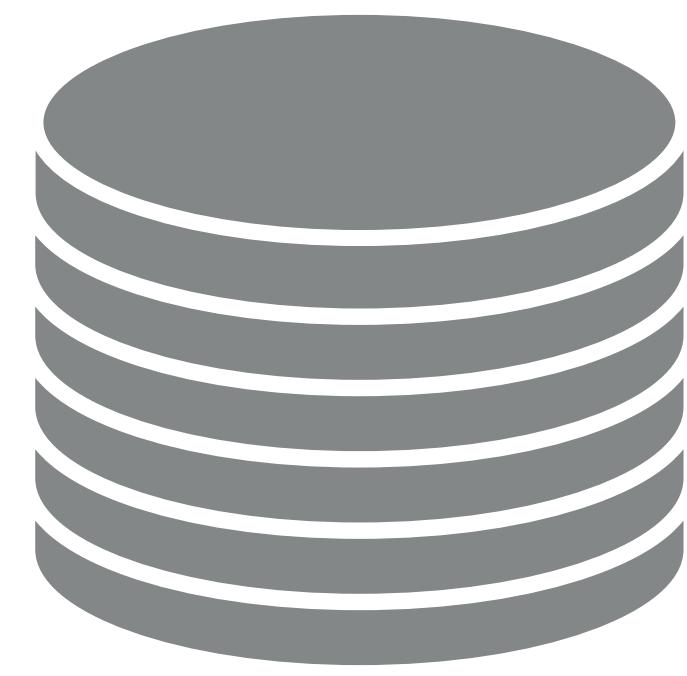
YAMLs provide info needed to run model

function - name of the function that yggdrasil should wrap

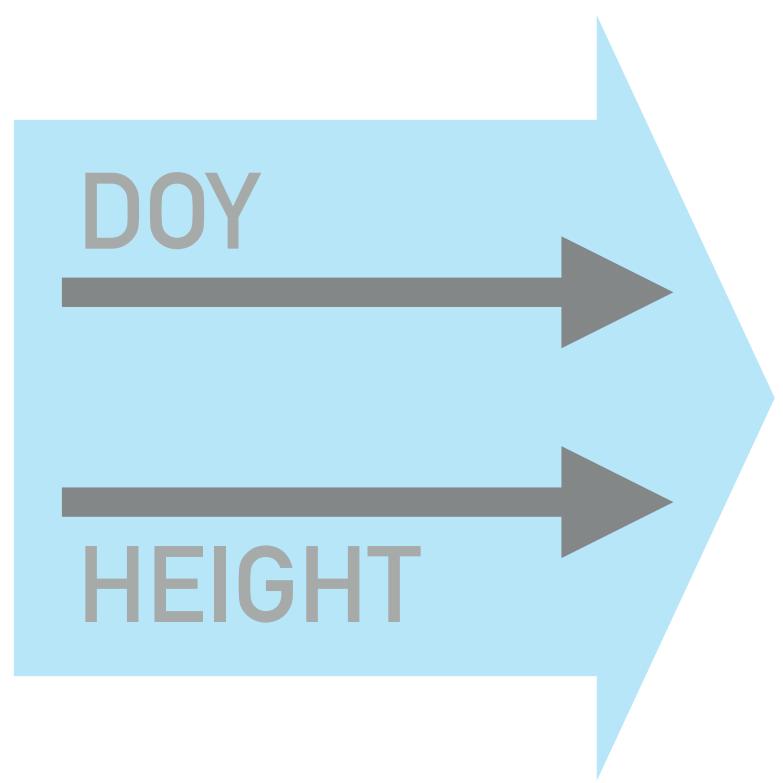


LIGHT MODEL





`input/light_v0.txt`

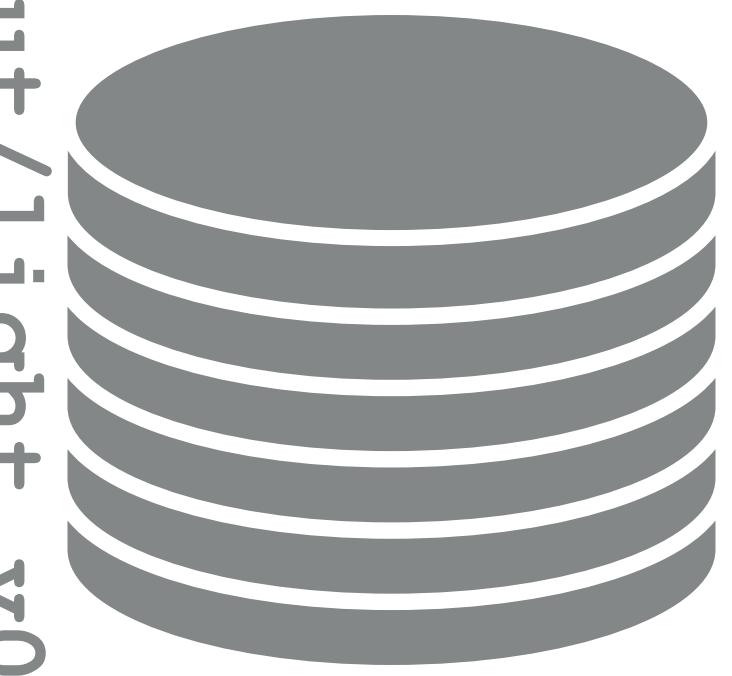


LIGHT MODEL

`light:output`

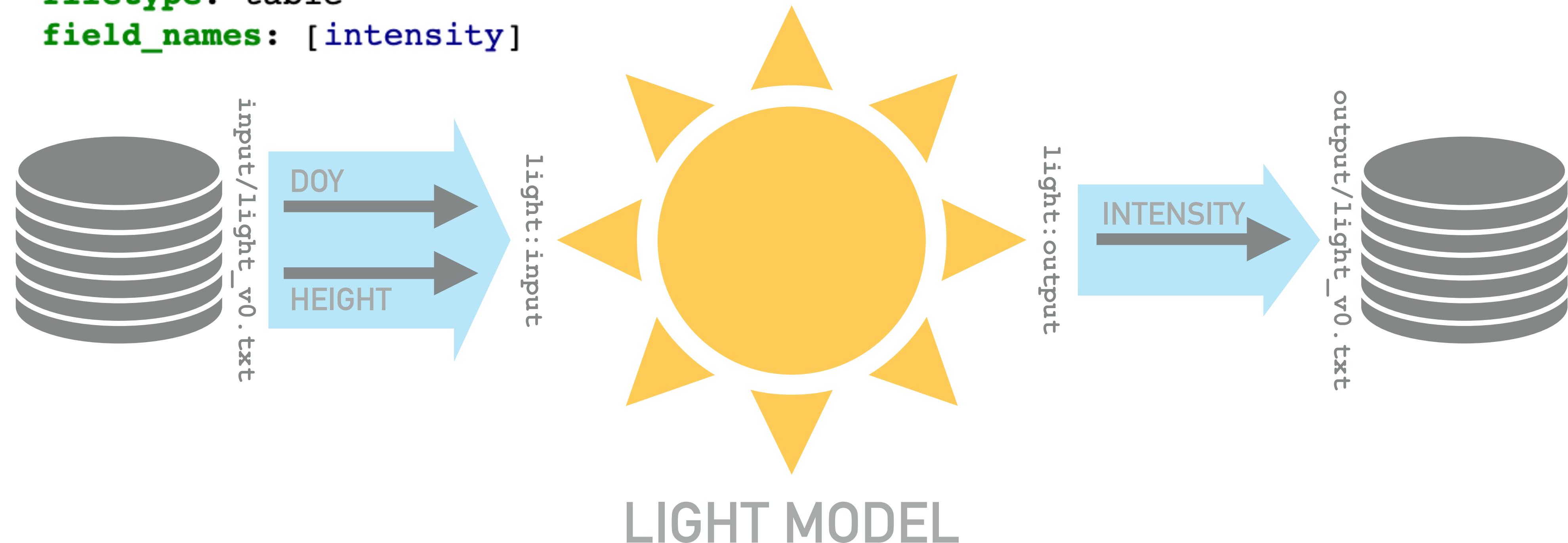


`output/light_v0.txt`



```
In [6]: tools.display_source('yamls/connections_v0.yml', number_lines=True)
```

```
file: yamls/connections_v0.yml
=====
1: connections:
2:   - input:
3:     name: ../input/light_v0.txt
4:     filetype: table
5:   output: light:input
6:   - input: light:output
7:     output:
8:       name: ../output/light_v0.txt
9:       filetype: table
10:      field_names: [intensity]
```



```
In [7]: run(['yamls/light_v0_python.yml', 'yamls/connections_v0.yml'], production_run=True)
```

```
INFO:88383:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg  
_light_v0.py  
End of input from temp_doy.  
INFO:88383:runner.waitModels[553]:YggRunner(runner): light finished running.  
INFO:88383:runner.waitModels[559]:YggRunner(runner): light finished exiting.  
INFO:88383:runner.waitModels[573]:YggRunner(runner): All models completed  
INFO:88383:runner.run[374]:YggRunner(runner):           init      0.000001  
INFO:88383:runner.run[374]:YggRunner(runner):           load drivers  0.309536  
INFO:88383:runner.run[374]:YggRunner(runner):           start drivers 0.091199  
INFO:88383:runner.run[374]:YggRunner(runner):           run models   5.400952  
INFO:88383:runner.run[374]:YggRunner(runner):           at exit     0.023104  
INFO:88383:runner.run[376]:YggRunner(runner): =====  
INFO:88383:runner.run[377]:YggRunner(runner):           Total      5.824792
```

```
In [7]: run(['yamls/light_v0_python.yml', 'yamls/connections_v0.yml'], production_run=True)
```

```
INFO:88383:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg  
_light_v0.py  
End of input from temp_doy.  
INFO:88383:runner.waitModels[553]:YggRunner(runner): light finished running.  
INFO:88383:runner.waitModels[559]:YggRunner(runner): light finished exiting.  
INFO:88383:runner.waitModels[573]:YggRunner(runner): All models completed  
INFO:88383:runner.run[374]:YggRunner(runner): init 0.000001  
INFO:88383:runner.run[374]:YggRunner(runner): load drivers 0.309536  
INFO:88383:runner.run[374]:YggRunner(runner): start drivers 0.091199  
INFO:88383:runner.run[374]:YggRunner(runner): run models 5.400952  
INFO:88383:runner.run[374]:YggRunner(runner): at exit 0.023104  
INFO:88383:runner.run[376]:YggRunner(runner): =====  
INFO:88383:runner.run[377]:YggRunner(runner): Total 5.824792
```

```
In [8]: tools.display_source('output/light_v0.txt', number_lines=True)
```

```
file: output/light_v0.txt  
=====  
1: # intensity  
2: # erg/(cm**2*s)  
3: # %g  
4: 0  
5: 40.6885  
6: 82.7537  
7: 168.259  
8: 342.017  
9: 434.386  
10: 617.737  
11: 896.166
```

```
In [9]: tools.display_source('models/light_v0.cpp', number_lines=True)
tools.display_source('yamls/light_v0_cpp.yml', number_lines=True)
run(['yamls/light_v0_cpp.yml', 'yamls/connections_v0.yml'], production_run=True)
tools.display_source('output/light_v0.txt', number_lines=True)
```

```
file: models/light_v0.cpp
=====
1: #define _USE_MATH_DEFINES // Required to use M_PI with MSVC
2: #include <math.h>
3:
4: /**
5:  @brief Compute the intensity of light.
6:
7:  @param[in] doy Day of year.
8:  @param[in] height Distance from ground in cm.
9:
10: @returns intensity Intensity of light in ergs cm^-2 s^-1.
11: */
12: double light(double doy, double height) {
13:     // Define parameters that are static across a run
14:     double amplitude = 80.0;
15:     double doy_offset = 0.0;
16:
17:     // Calculate intensity
18:     double intensity = amplitude * height * (1.0 + sin(2.0 * M_PI * (doy - doy_offset) / 365));
19:
20:     return intensity;
21: }
```

```
In [9]: tools.display_source('models/light_v0.cpp', number_lines=True)
tools.display_source('yamls/light_v0_cpp.yml', number_lines=True)
run(['yamls/light_v0_cpp.yml', 'yamls/connections_v0.yml'], production_run=True)
tools.display_source('output/light_v0.txt', number_lines=True)
```

```
file: yamls/light_v0_cpp.yml
=====
1: model:
2:   name: light
3:   language: c++
4:   args: ../models/light_v0.cpp
5:   function: light
6:   inputs:
7:     - name: input
8:       vars: [doy, height]
9:       datatype:
10:         type: array
11:         items:
12:           - type: float
13:             units: day
14:           - type: float
15:             units: cm
16:   output:
17:     - name: output
18:       datatype:
19:         type: float
20:         units: ergs/(cm**2*s)
```

```
In [9]: tools.display_source('models/light_v0.cpp', number_lines=True)
tools.display_source('yaml/light_v0_cpp.yml', number_lines=True)
run(['yaml/light_v0_cpp.yml', 'yaml/connections_v0.yml'], production_run=True)
tools.display_source('output/light_v0.txt', number_lines=True)
```

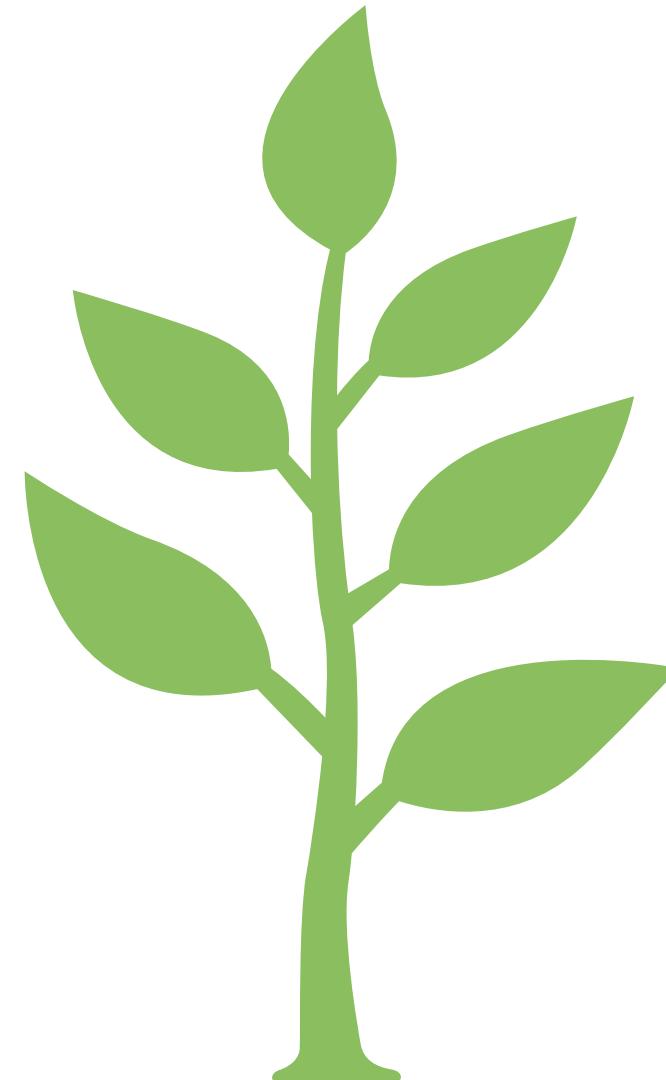
```
INFO:91854:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg_light_v0_cpp_clang++x_clang++x.out
End of input from &doy, &height.
INFO:91854:runner.waitModels[553]:YggRunner(runner): light finished running.
INFO:91854:runner.waitModels[559]:YggRunner(runner): light finished exiting.
INFO:91854:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:91854:runner.run[374]:YggRunner(runner):           init      0.000001
INFO:91854:runner.run[374]:YggRunner(runner):           load drivers   3.856998
INFO:91854:runner.run[374]:YggRunner(runner):           start drivers  0.148603
INFO:91854:runner.run[374]:YggRunner(runner):           run models    0.533877
INFO:91854:runner.run[374]:YggRunner(runner):           at exit       0.075296
INFO:91854:runner.run[376]:YggRunner(runner): =====
INFO:91854:runner.run[377]:YggRunner(runner):           Total      4.614775

file: output/light_v0.txt
=====
1: # intensity
2: # ergs/(cm**2*s)
3: # %g
4: 0
5: 40.6885
6: 82.7537
7: 168.259
8: 342.017
9: 434.386
10: 617.737
11: 896.166
```

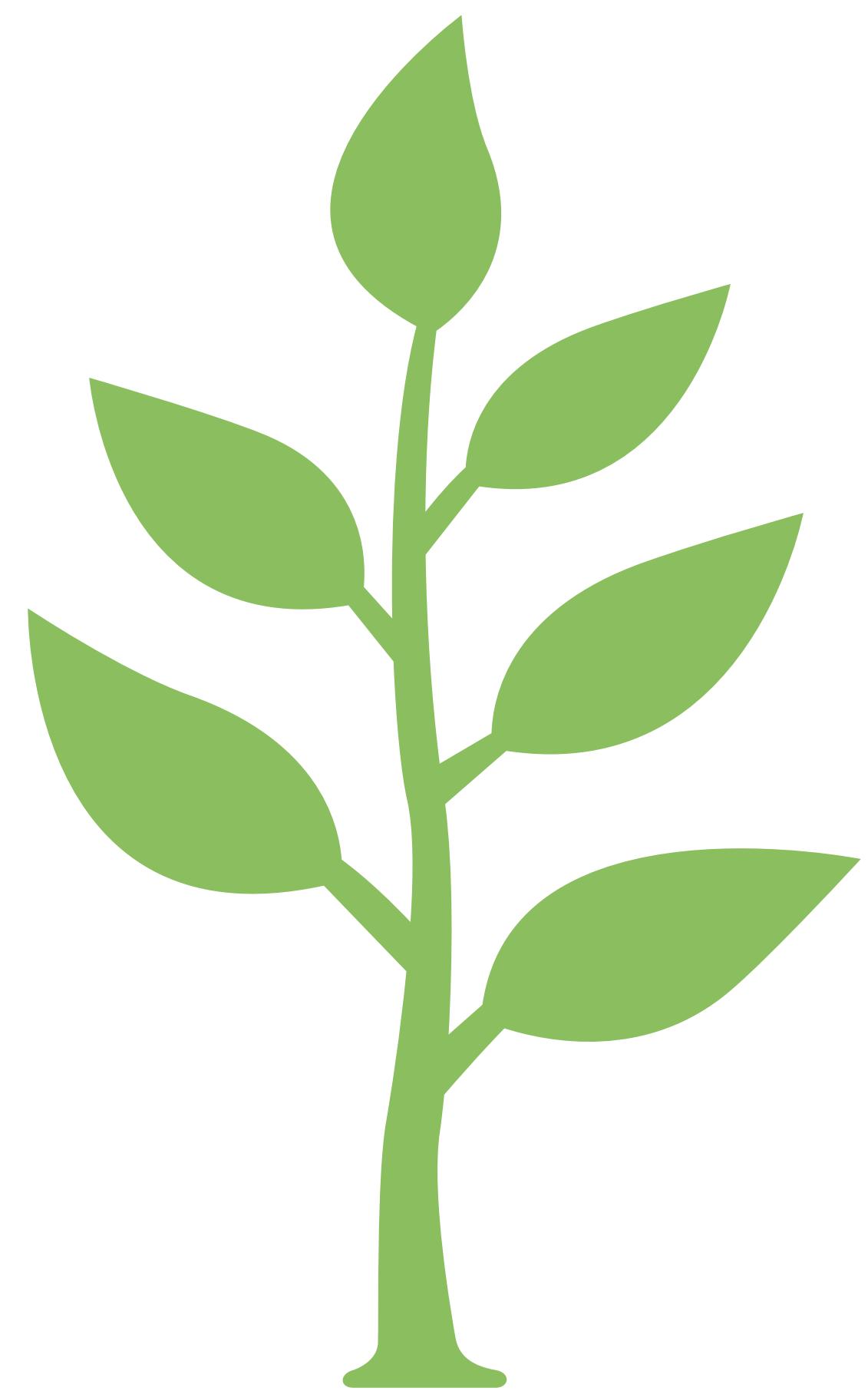
**INTEGRATING MODELS
VIA INTERFACE**

```
In [12]: tools.display_source('models/shoot_v0.py', number_lines=True)
```

```
file: models/shoot_v0.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: # Continue simulation until time limit is reached
26: while t <= tmax:
27:
28:     # Compute the scale factor
29:     # (pretend this is a biologically complex calculation)
30:     scale = mass / 4.5e4
31:
32:     # Grow the shoot
33:     # (pretend this is a biologically complex calculation)
34:     mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
35:     mass += mass * scale
36:
37:     # Save mesh for this timestep
38:     filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
39:     with open(filename_mesh, 'w') as fd:
40:         mesh.export(fd, 'obj')
41:
42:     # Advance time step
43:     t += tstep
44:     i += 1
```



SHOOT
MODEL



SHOOT
MODEL

```
In [13]: tools.display_source('yamls/shoot_v0.yml', number_lines=True)
run('yamls/shoot_v0.yml', production_run=True)
```

```
file: yamls/shoot_v0.yml
=====
```

```
1: model:
2:   name: shoot
3:   language: python
4:   args: [./models/shoot_v0.py, 0.0, 48.0, 6.0]
```

```
INFO:91854:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/shoot_v0.py 0.0 48.0 6.0
INFO:91854:runner.waitModels[553]:YggRunner(runner): shoot finished running.
INFO:91854:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:91854:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:91854:runner.run[374]:YggRunner(runner):           init      0.000001
INFO:91854:runner.run[374]:YggRunner(runner):           load drivers  0.000918
INFO:91854:runner.run[374]:YggRunner(runner):           start drivers 0.042213
INFO:91854:runner.run[374]:YggRunner(runner):           run models    0.856992
INFO:91854:runner.run[374]:YggRunner(runner):           at exit       0.000524
INFO:91854:runner.run[376]:YggRunner(runner): =====
INFO:91854:runner.run[377]:YggRunner(runner):           Total      0.900648
```

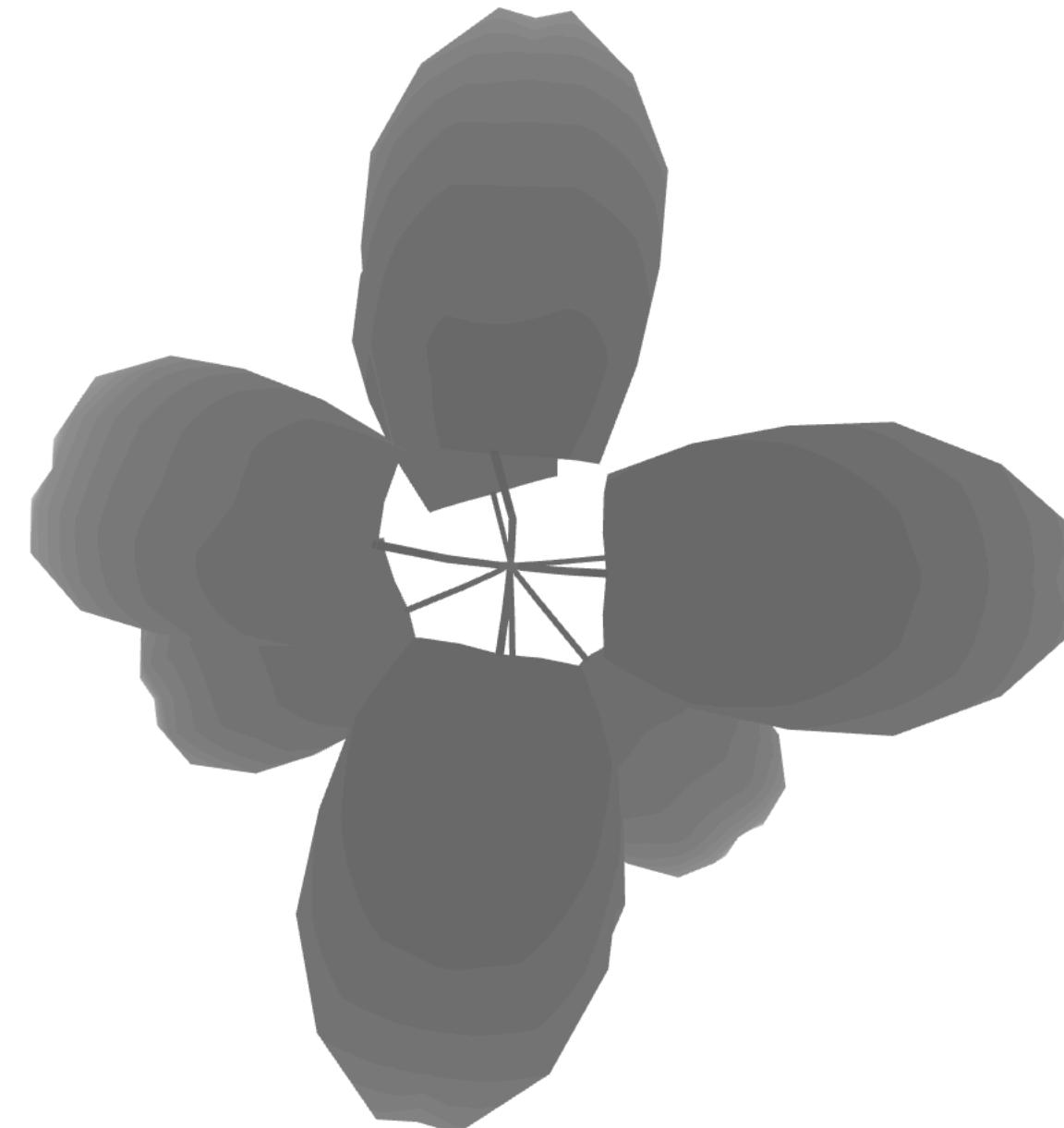
```
In [13]: tools.display_source('yaml/shoot_v0.yml', number_lines=True)
run('yaml/shoot_v0.yml', production_run=True)
```

```
file: yaml/shoot_v0.yml
=====
1: model:
2:   name: shoot
3:   language: python
4:   args: [./models/shoot_v0.py, 0.0, 48.0, 6.0]
```

```
INFO:91854:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/shoot_v0.py 0.0 48.0 6.0
INFO:91854:runner.waitModels[553]:YggRunner(runner): waiting for 1 models
INFO:91854:runner.waitModels[559]:YggRunner(runner): waiting for 1 models
INFO:91854:runner.waitModels[573]:YggRunner(runner): waiting for 1 models
INFO:91854:runner.run[374]:YggRunner(runner): running 1 models
INFO:91854:runner.run[376]:YggRunner(runner): running 1 models
INFO:91854:runner.run[377]:YggRunner(runner): running 1 models
```

```
In [14]: mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.show()
```

Out[14]:



```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42:             flag = height_out.send(
43:                 [units.add_units(t, 'hrs'),
44:                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45:             if not flag:
46:                 raise Exception("Error sending height to output")
47:
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

Code to call yggdrasil inside "if blocks" so that model runs exactly the same without yggdrasil

```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42:             flag = height_out.send(
43:                 [units.add_units(t, 'hrs'),
44:                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45:             if not flag:
46:                 raise Exception("Error sending height to output")
47:
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

Import yggdrasil functions and connect to the channel that will be listed in the YAML.

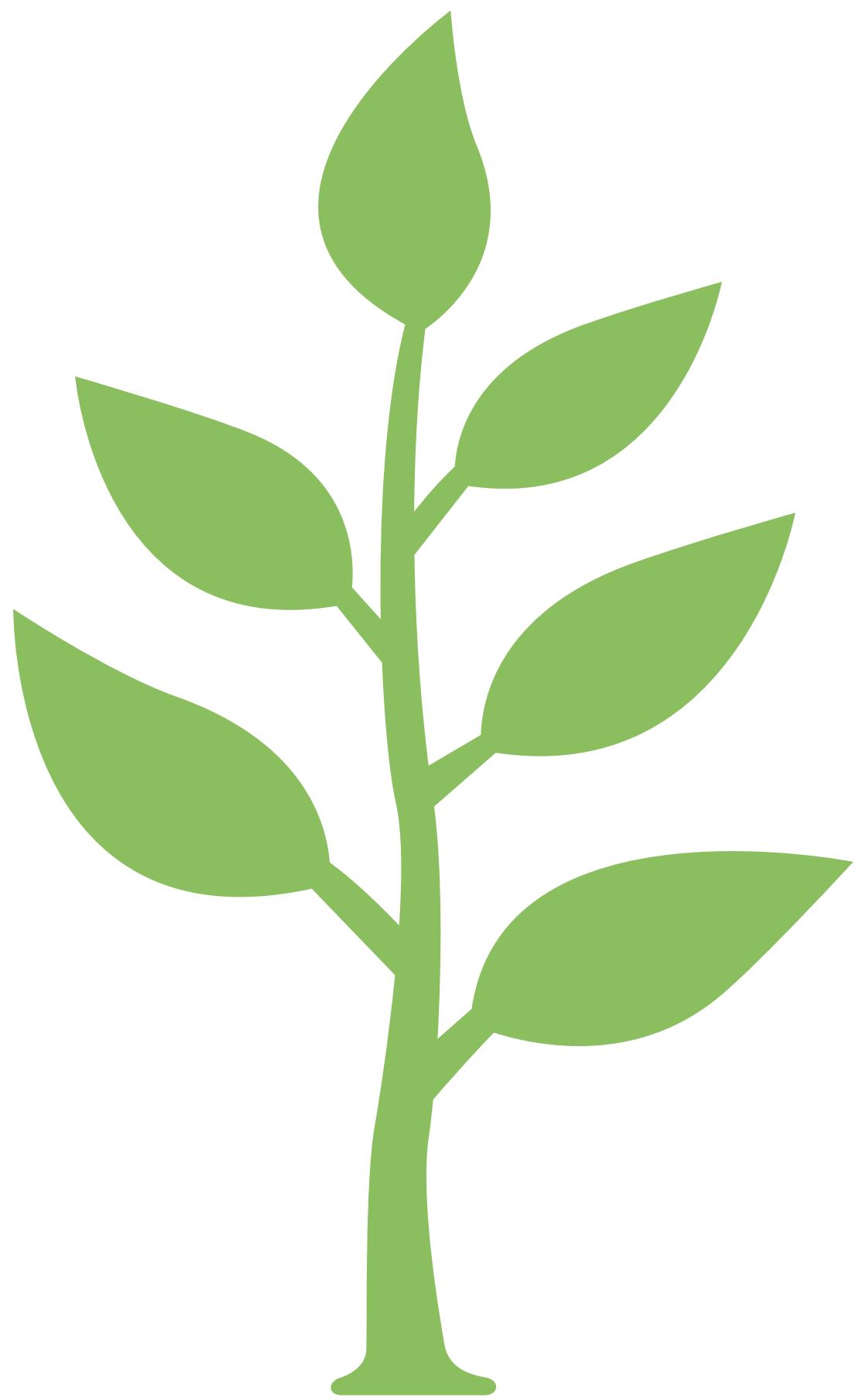
Code to call yggdrasil inside “if blocks” so that model runs exactly the same without yggdrasil

```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42: +             flag = height_out.send(
43: +                 [units.add_units(t, 'hrs'),
44: +                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45: +             if not flag:
46: +                 raise Exception("Error sending height to output")
47: +
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

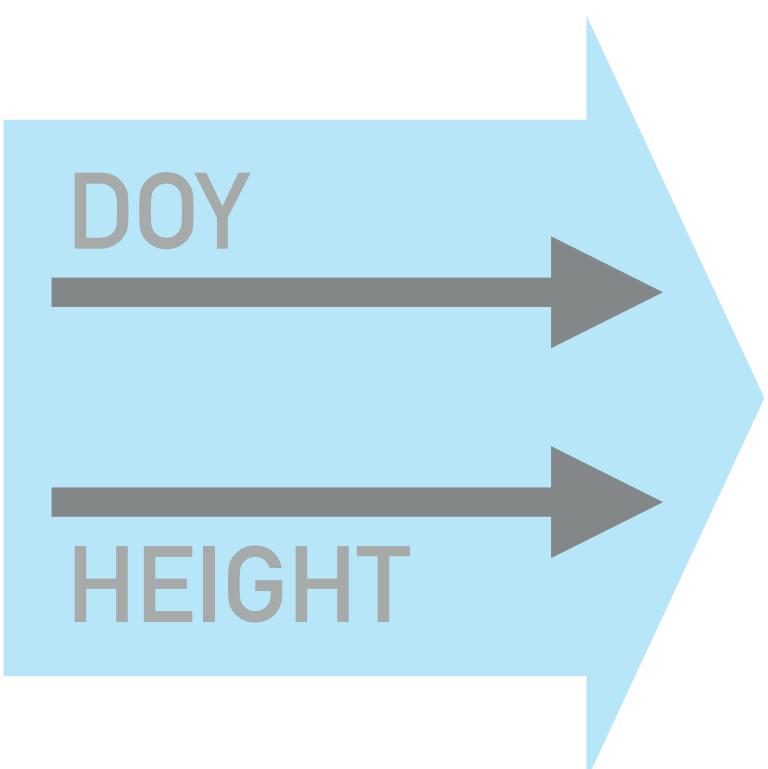
Send height to output channel

Code to call yggdrasil inside "if blocks" so that model runs exactly the same without yggdrasil

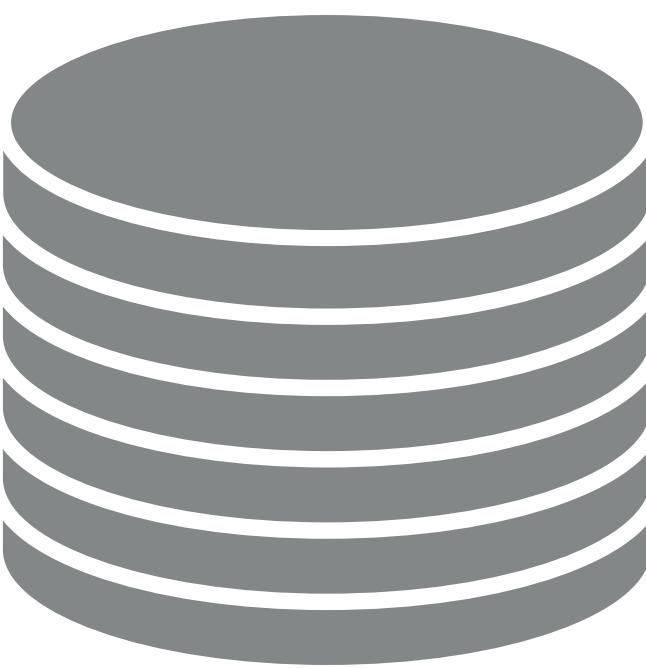


SHOOT
MODEL

height

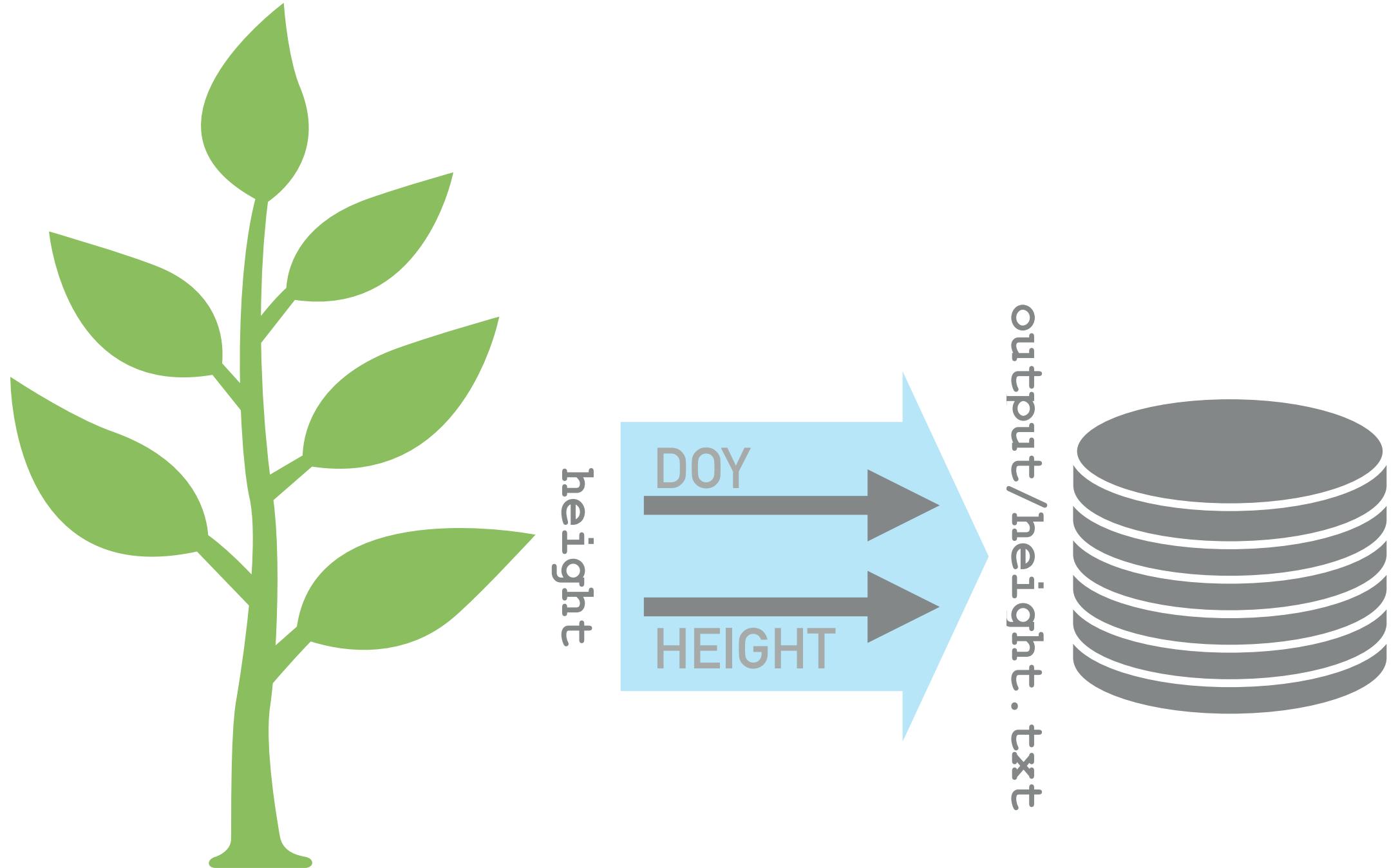


output/height.txt



```
In [16]: tools.display_source_diff('yamls/shoot_v0.yml', 'yamls/shoot_v1.yml', number_lines=True)
```

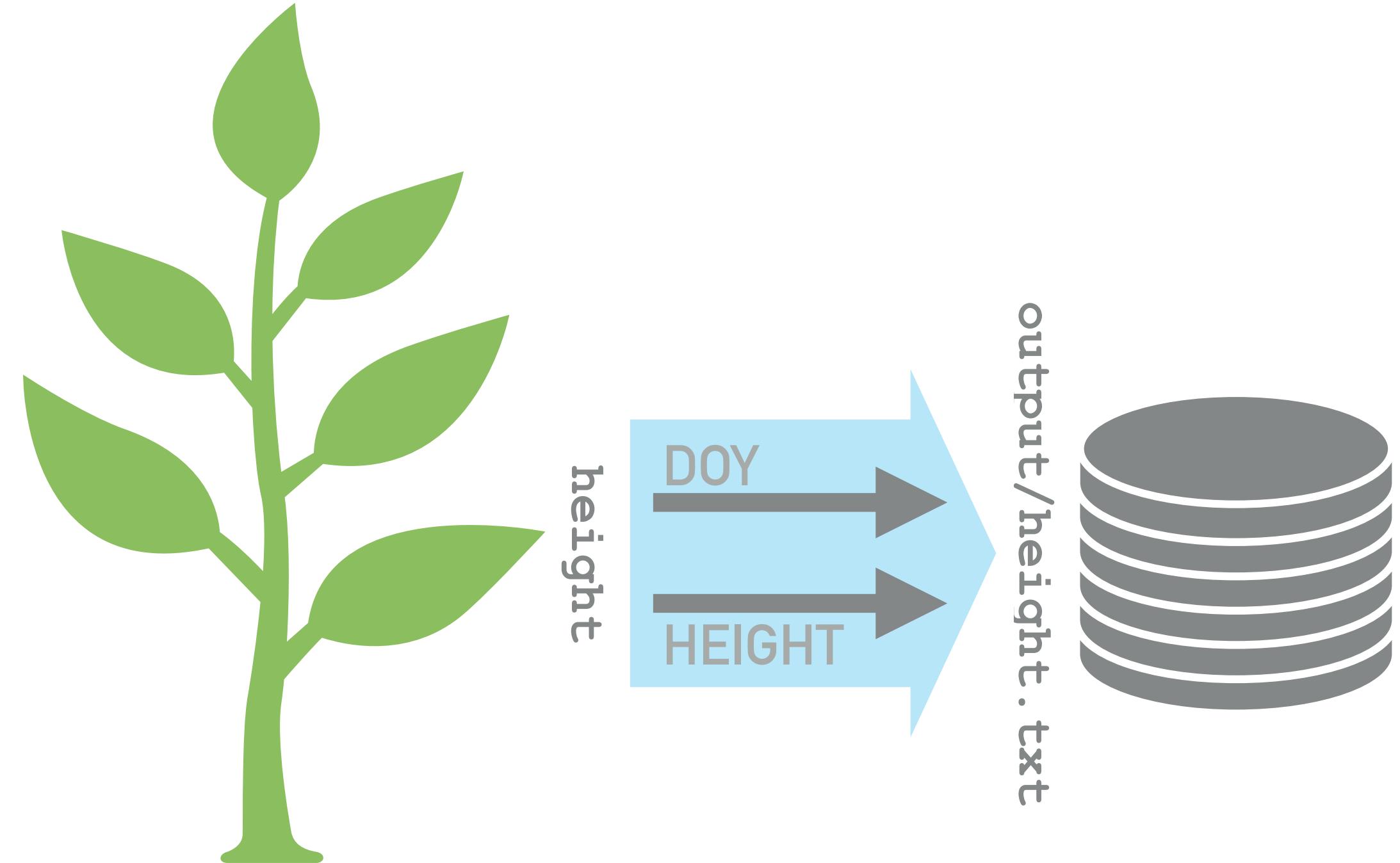
```
file1: yamls/shoot_v0.yml
file2: yamls/shoot_v1.yml
=====
1:   model:
2:     name: shoot
3:     language: python
4:     - args: [./models/shoot_v0.py, 0.0, 48.0, 6.0]
5:     ?
6:
7: +   args: [./models/shoot_v1.py, 0.0, 48.0, 6.0]
8: +
9: +   ?
10: +
11: +   outputs:
12: +     - name: height
13: +       default_file:
14: +         name: ../output/height.txt
15: +       filetype: table
```



SHOOT
MODEL

```
In [16]: tools.display_source_diff('yamls/shoot_v0.yml', 'yamls/shoot_v1.yml', number_lines=True)
```

```
file1: yamls/shoot_v0.yml
file2: yamls/shoot_v1.yml
=====
1:   model:
2:     name: shoot
3:     language: python
4:     - args: [../models/shoot_v0.py, 0.0, 48.0, 6.0]
5:     ?
6:
7: +   args: [../models/shoot_v1.py, 0.0, 48.0, 6.0]
8: +
9: +   ?
10: +
11: +   outputs:
12: +     - name: height
13: +       default_file:
14: +         name: ../output/height.txt
15: +       filetype: table
```



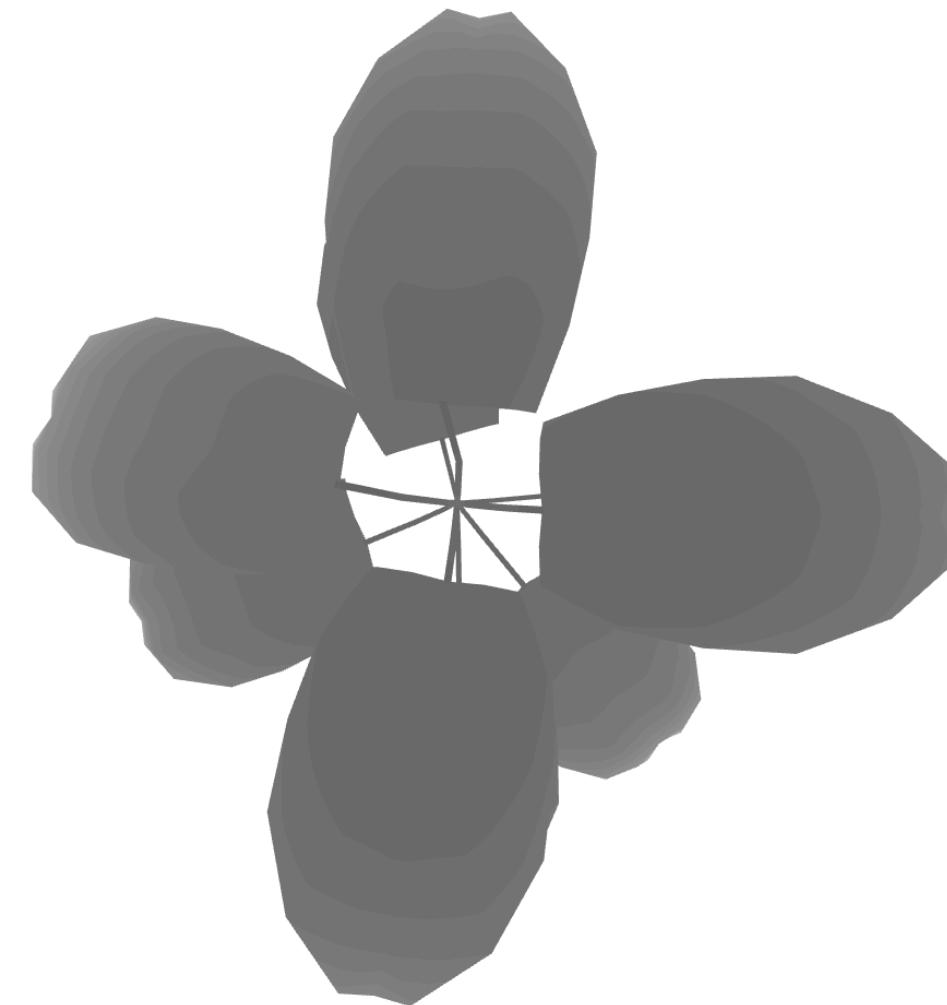
Declare one output with a default file
that is only used if no other connection
connects to it.

SHOOT
MODEL

```
In [17]: run(['yamls/shoot_v1.yml'], production_run=True)
```

```
INFO:91854:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/sho  
ot_v1.py 0.0 48.0 6.0  
INFO:91854:runner.waitModels[553]:YggRunner(runner): shoot finished running.  
INFO:91854:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.  
INFO:91854:runner.waitModels[573]:YggRunner(runner): All models completed  
INFO:91854:runner.run[374]:YggRunner(runner): init 0.000001  
INFO:91854:runner.run[374]:YggRunner(runner): load drivers 0.006655  
INFO:91854:runner.run[374]:YggRunner(runner): start drivers 0.060119  
INFO:91854:runner.run[374]:YggRunner(runner): run models 5.849015  
INFO:91854:runner.run[374]:YggRunner(runner): at exit 0.003885  
INFO:91854:runner.run[376]:YggRunner(runner): =====  
INFO:91854:runner.run[377]:YggRunner(runner): Total 5.919675
```

```
In [18]: mesh = trimesh.load_mesh('output/mesh_008.obj')  
mesh.show()  
out[18]:
```



```
In [19]: tools.display_source('output/height.txt')
```

```
file: output/height.txt  
=====
```

#	hr	m
0	77.2603	
6	80.6941	
12	84.4399	
18	88.5415	
24	93.0513	
30	98.0321	
36	103.561	
42	109.73	
48	116.656	

NEW NOTEBOOK!
(BREAK TIME)

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) [⟳](#)

<input type="checkbox"/>	0	▼	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images				33 minutes ago	
<input type="checkbox"/>	 input				33 minutes ago	
<input type="checkbox"/>	 meshes				33 minutes ago	
<input type="checkbox"/>	 models				33 minutes ago	
<input type="checkbox"/>	 yaml				33 minutes ago	
<input type="checkbox"/>	 00-intro.ipynb				33 minutes ago	457 kB
<input type="checkbox"/>	 01-connections.ipynb				33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb				33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb				33 minutes ago	3.56 kB

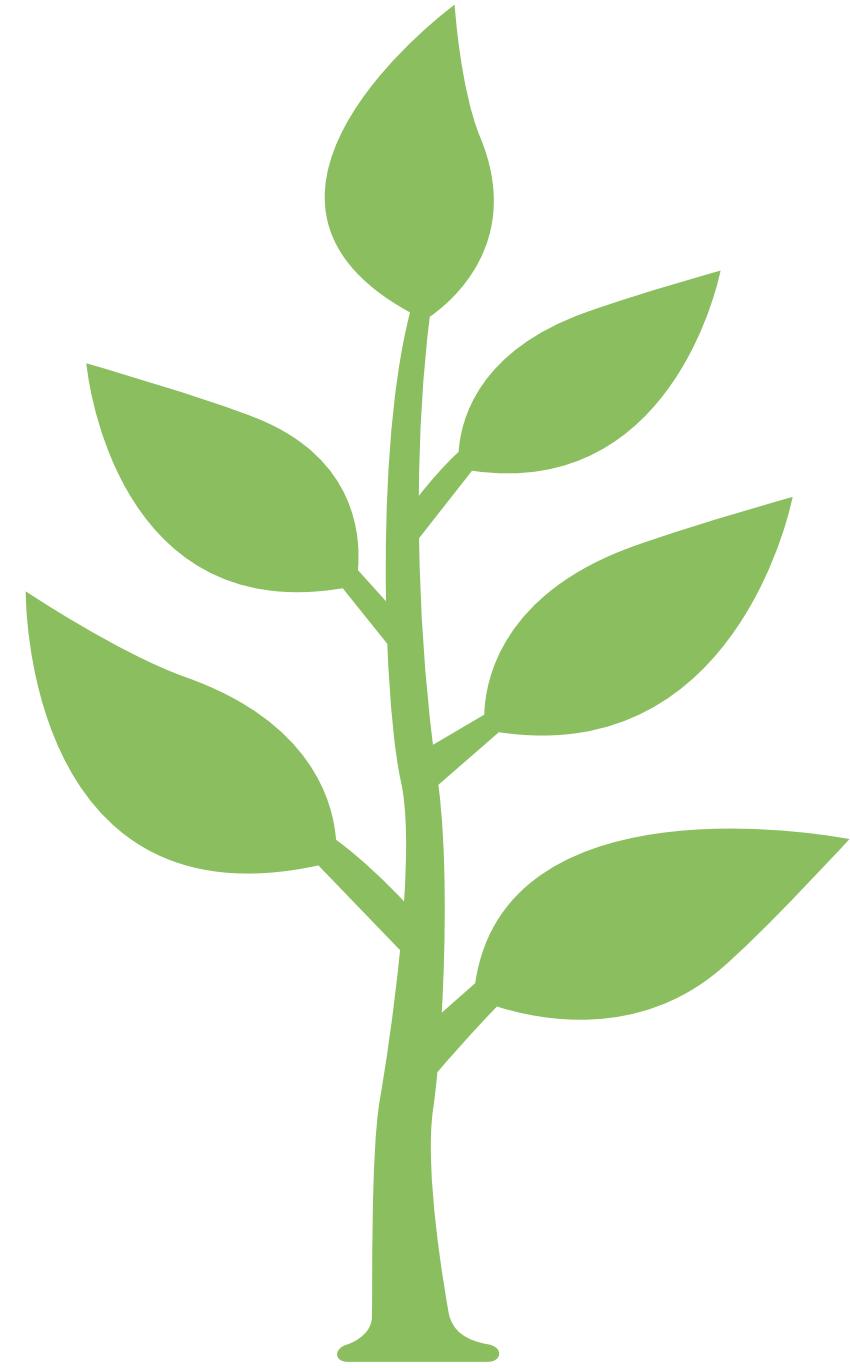
[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) 

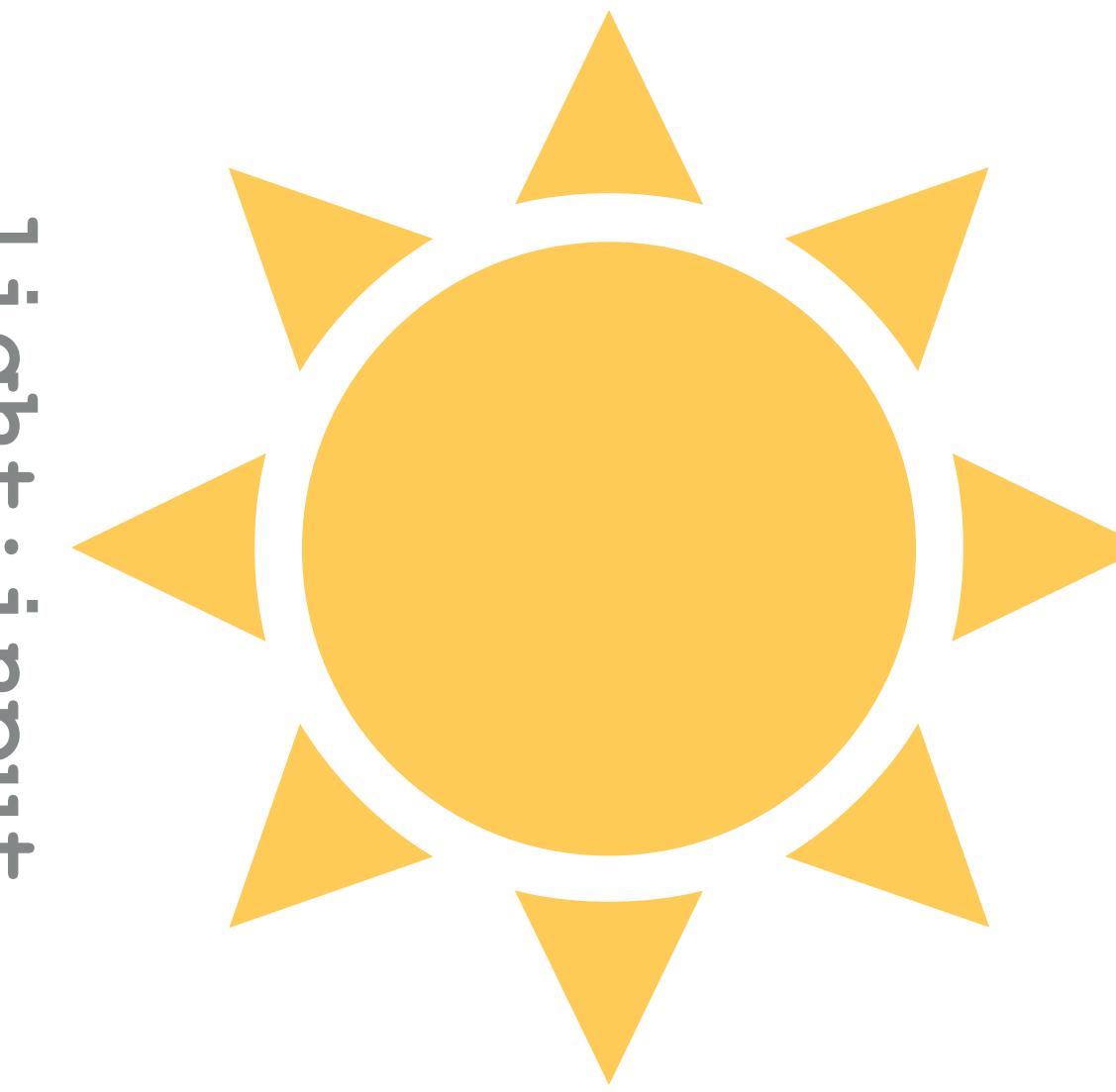
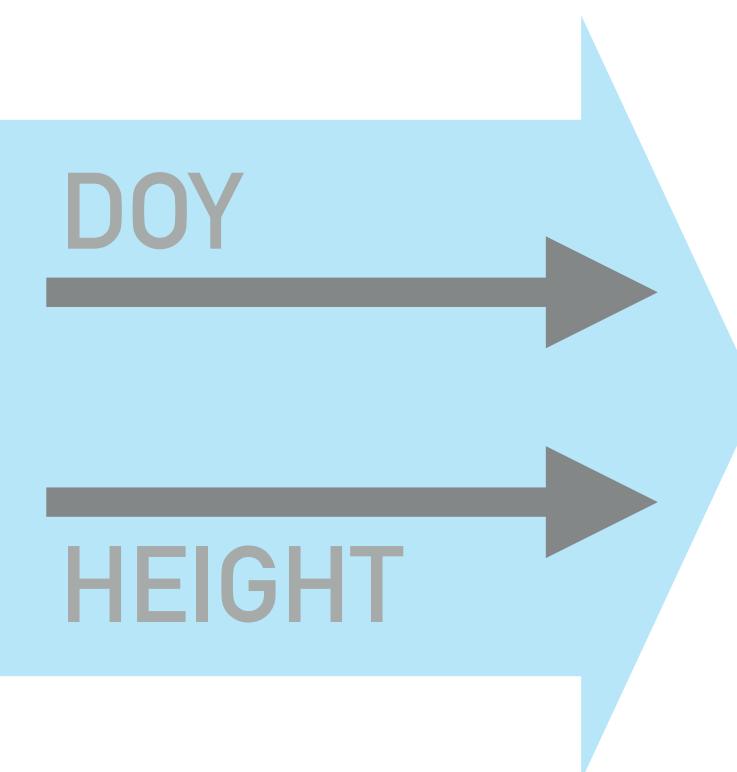
<input type="checkbox"/> 0	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images		33 minutes ago	
<input type="checkbox"/>	 input		33 minutes ago	
<input type="checkbox"/>	 meshes		33 minutes ago	
<input type="checkbox"/>	 models		33 minutes ago	
<input type="checkbox"/>	 yaml s		33 minutes ago	
<input type="checkbox"/>	 00-intro.ipynb		33 minutes ago	457 kB
<input checked="" type="checkbox"/>	 01-connections.ipynb		33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb		33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb		33 minutes ago	3.56 kB

**ONE WAY
MODEL-TO-MODEL
CONNECTION**



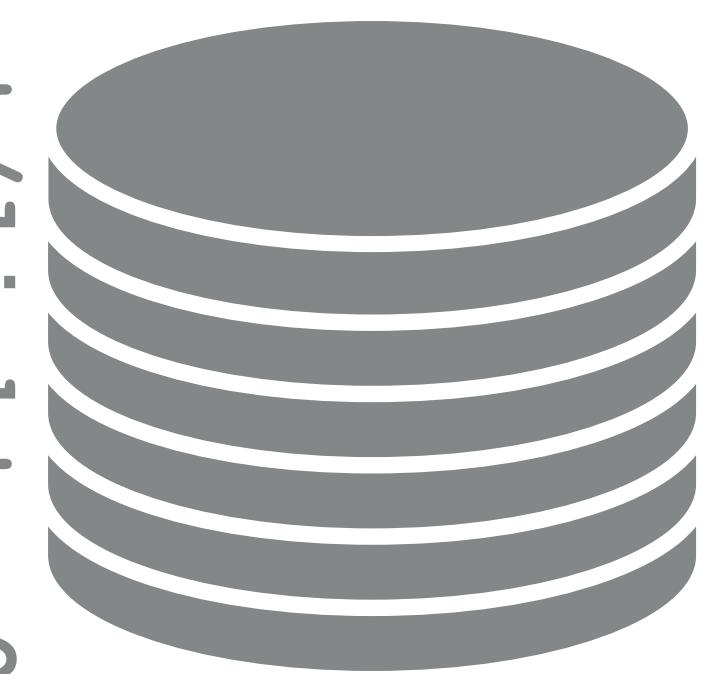
SHOOT
MODEL

shoot:height



LIGHT MODEL

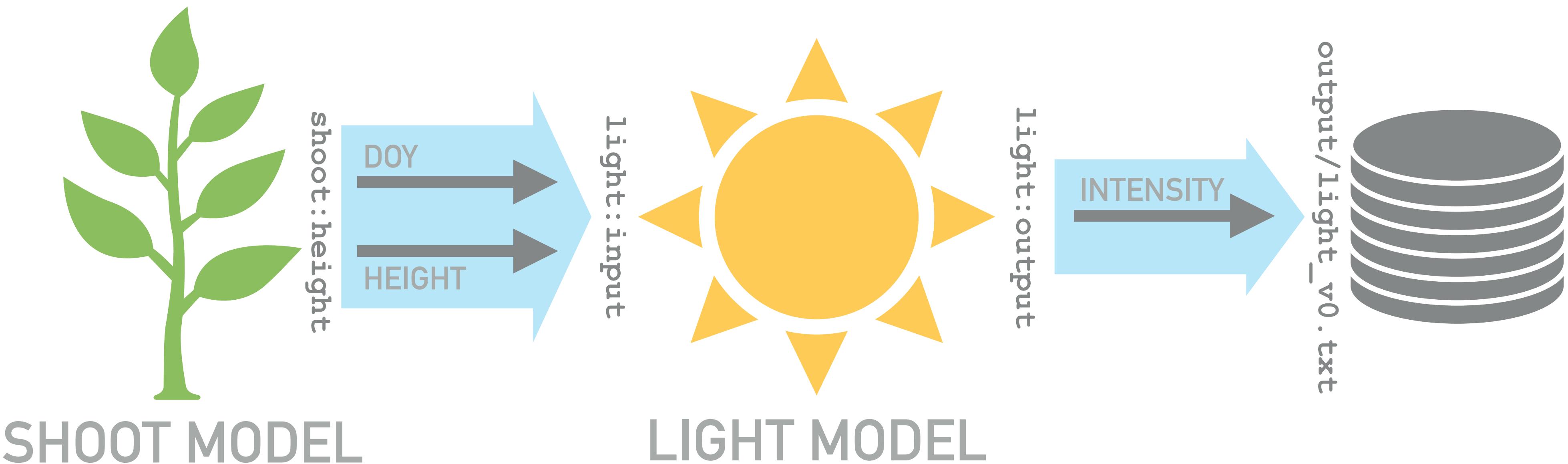
light:output



output/light_v0.txt

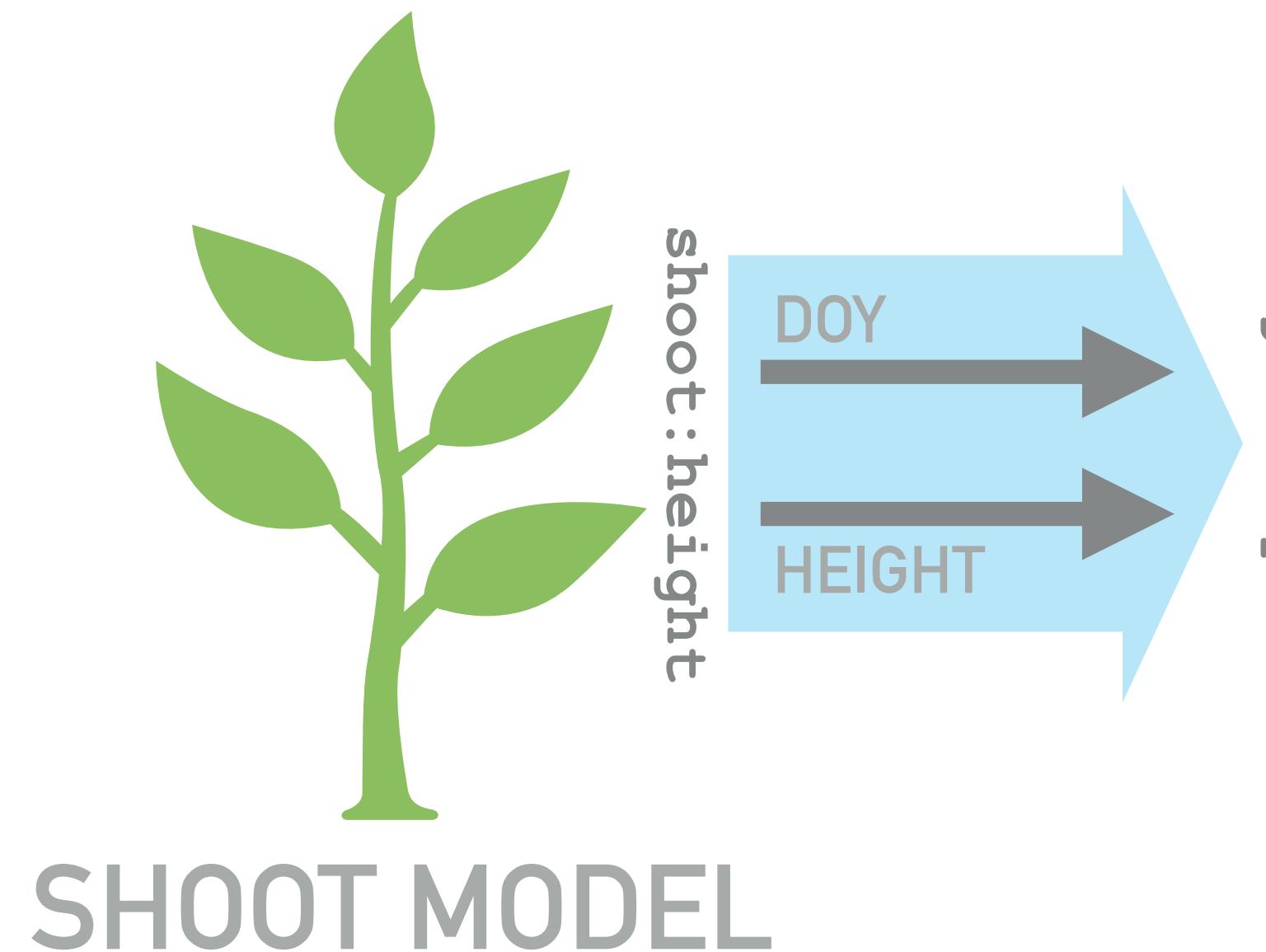
```
In [2]: tools.display_source_diff('yamls/connections_v0.yml', 'yamls/connections_v1.yml', number_lines=True)
```

```
file1: yamls/connections_v0.yml
file2: yamls/connections_v1.yml
=====
1:   connections:
2:     - input:
3:     - input: shoot:height
4:     ?
5:       ++++++
6:
7:     - name: ../input/light_v0.txt
8:     - filetype: table
9:     output: light:input
10:    - input: light:output
11:    output:
12:      name: ../output/light_v0.txt
13:      filetype: table
14:      field_names: [intensity]
```



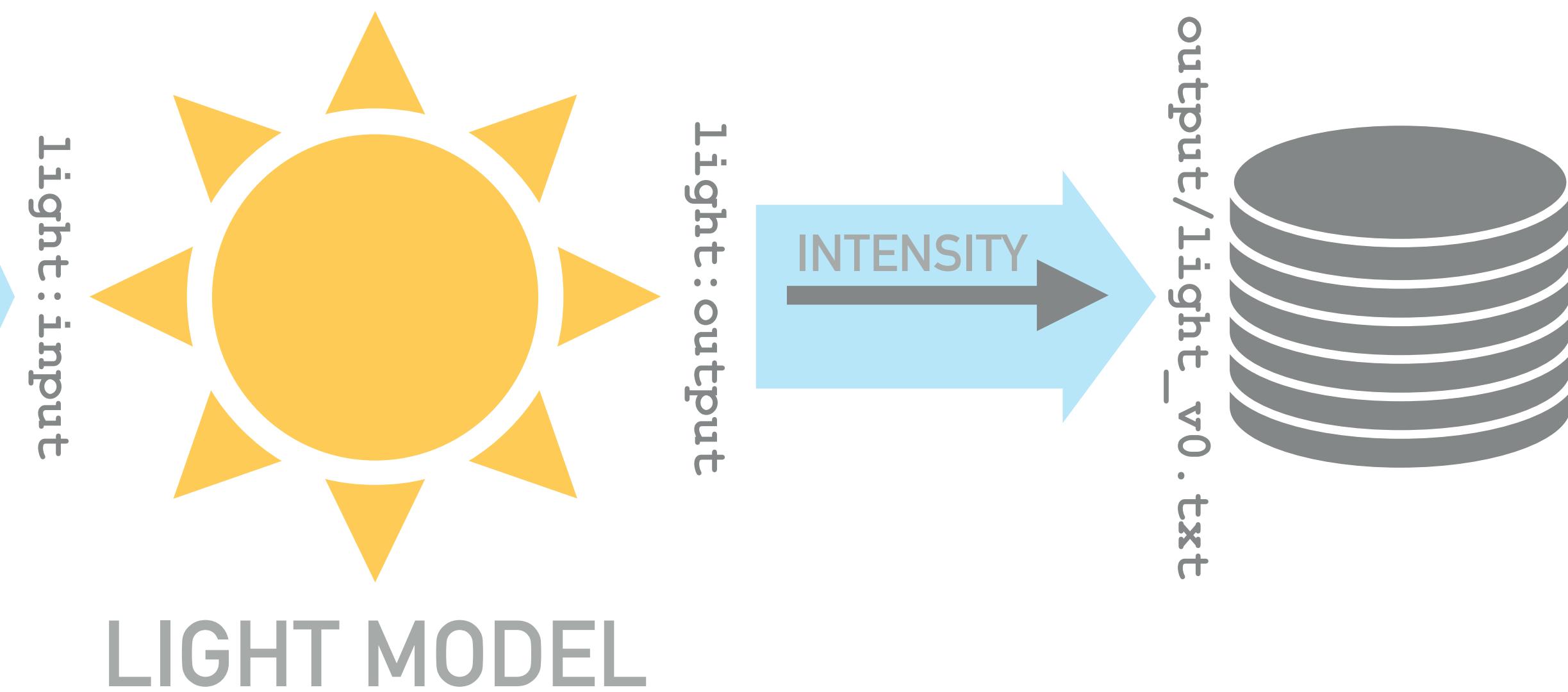
```
In [2]: tools.display_source_diff('yamls/connections_v0.yml', 'yamls/connections_v1.yml', number_lines=True)
```

```
file1: yamls/connections_v0.yml
file2: yamls/connections_v1.yml
=====
1:   connections:
2:     - input:
3:       - input: shoot:height
4:       ?
5:         ++++++
6:
7:         - name: ../input/light_v0.txt
8:         - filetype: table
9:           output: light:input
10:          - input: light:output
11:            output:
12:              name: ../output/light_v0.txt
13:              filetype: table
14:              field_names: [intensity]
```



Connect light input with shoot output
channel "height"

The prefix "shoot:" indicates the
height channel for the shoot model

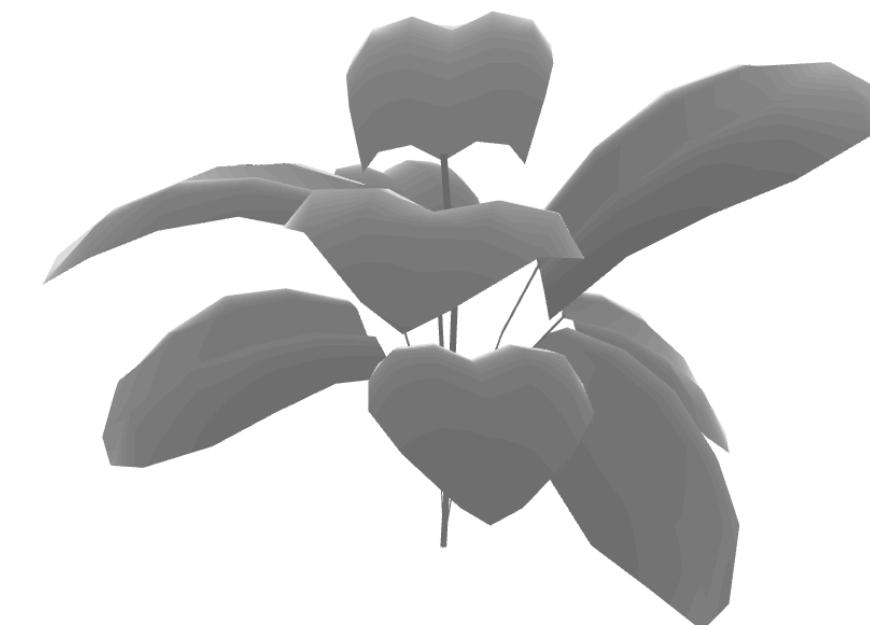


```
In [3]: run(['yamls/light_v0_python.yml', 'yamls/shoot_v1.yml', 'yamls/connections_v1.yml'], production_run=True)
```

```
INFO:93696:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg  
_light_v0.py  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/sho  
ot_v1.py 0.0 48.0 6.0  
End of input from temp_doy.  
INFO:93696:runner.waitModels[553]:YggRunner(runner): shoot finished running.  
INFO:93696:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.  
INFO:93696:runner.waitModels[553]:YggRunner(runner): light finished running.  
INFO:93696:runner.waitModels[559]:YggRunner(runner): light finished exiting.  
INFO:93696:runner.waitModels[573]:YggRunner(runner): All models completed  
INFO:93696:runner.run[374]:YggRunner(runner): init 0.000000  
INFO:93696:runner.run[374]:YggRunner(runner): load drivers 0.282206  
INFO:93696:runner.run[374]:YggRunner(runner): start drivers 0.090545  
INFO:93696:runner.run[374]:YggRunner(runner): run models 6.751301  
INFO:93696:runner.run[374]:YggRunner(runner): at exit 0.021096  
INFO:93696:runner.run[376]:YggRunner(runner): =====  
INFO:93696:runner.run[377]:YggRunner(runner): Total 7.145148
```

```
In [4]: mesh = trimesh.load_mesh('output/mesh_008.obj')  
mesh.show()
```

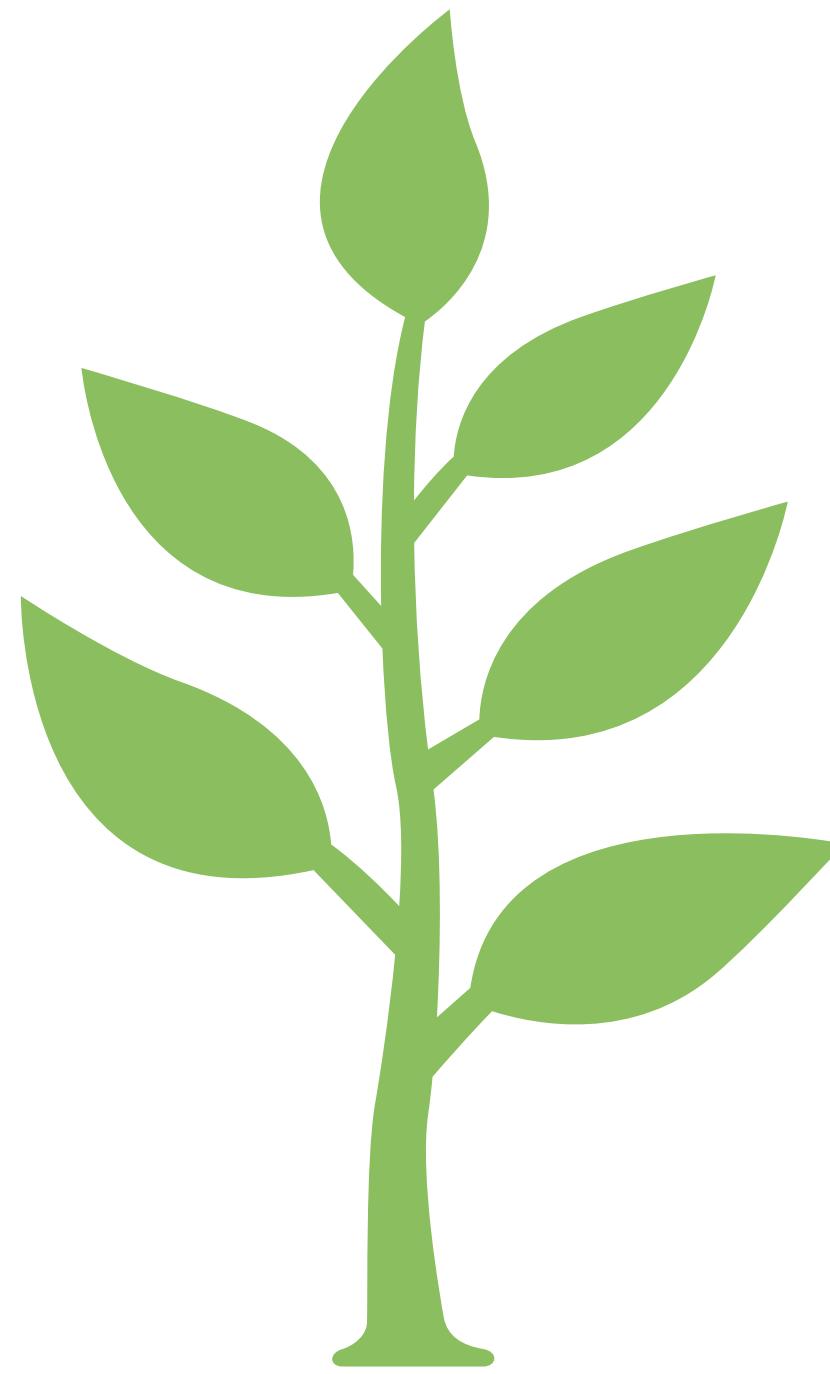
```
Out[4]:
```



```
In [5]: tools.display_source('output/light_v0.txt')
```

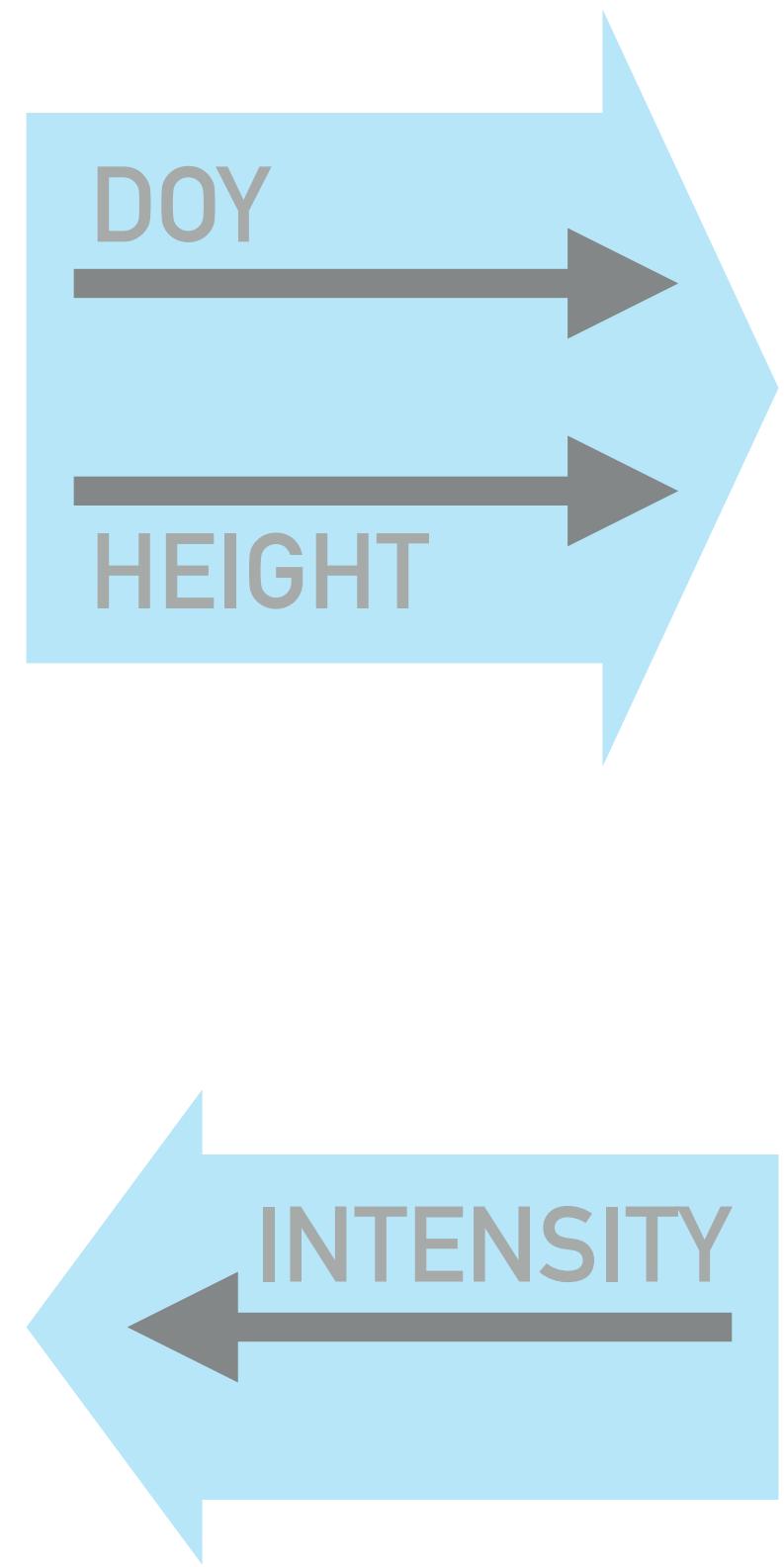
```
file: output/light_v0.txt  
=====  
# intensity  
# erg/(cm**2*s)  
# %g  
618082  
648331  
681333  
717477  
757224  
801131  
849874  
904281  
965376
```

REMOTE PROCEDURE CALL (RPC)

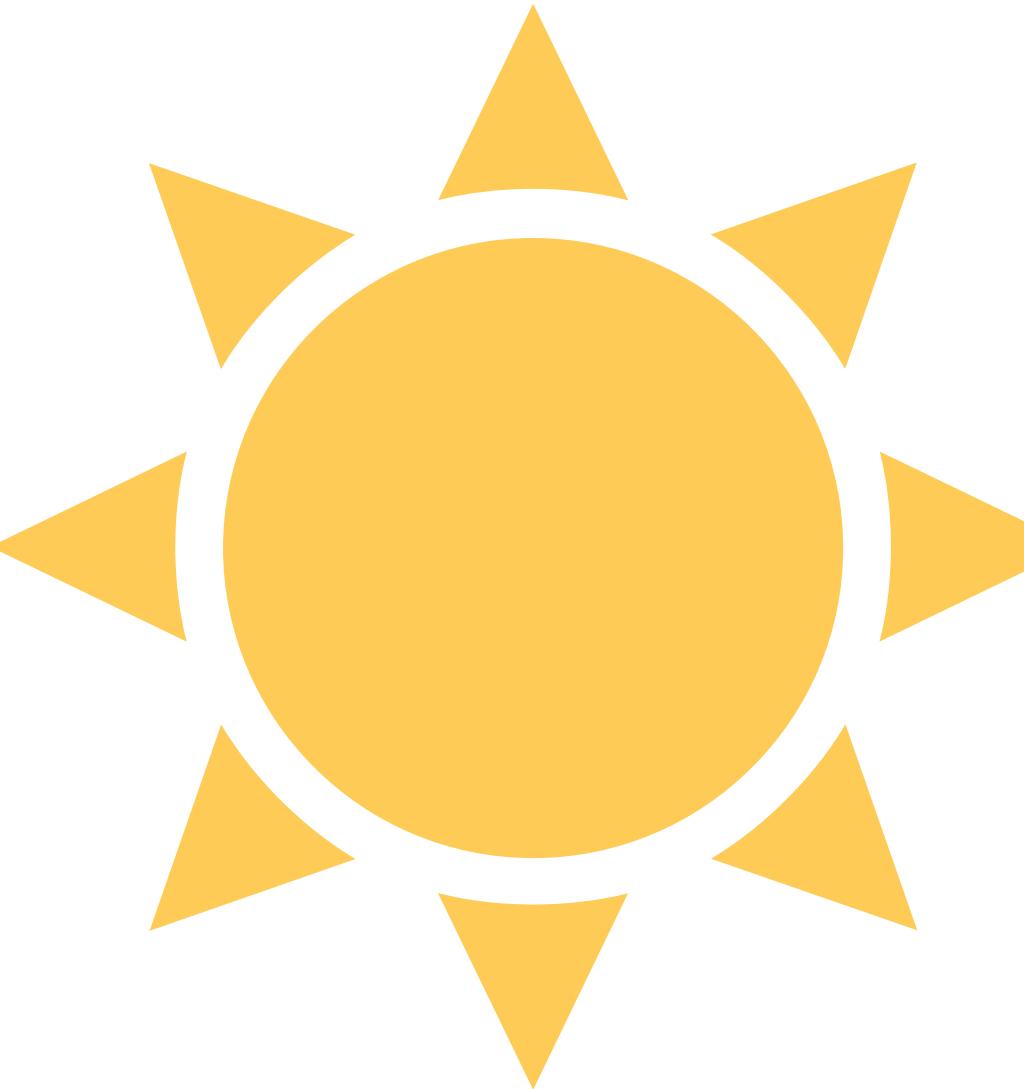


SHOOT
MODEL

light_shoot



light:input light:output



LIGHT
MODEL

```
In [6]: tools.display_source_diff('models/shoot_v1.py', 'models/shoot_v2.py', number_lines=True)
```

```
file1: models/shoot_v1.py
file2: models/shoot_v2.py
=====
1: import os
2: import trimesh
3: import argparse
...
28: # If the model is running as part of an yggdrasil integration, import
29: # the relevant yggdrasil routines and use the interface routine to
30: # complete the connection defined in the YAML
31: if with_yggdrasil:
32:     from yggdrasil import units
33:     from yggdrasil.languages.Python.YggInterface import YggOutput
34:     from yggdrasil.languages.Python.YggInterface import YggRpcClient
35:
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39:         # If running as part an yggdrasil integration, send the time and
40:         # maximum height of the mesh to the height channel with units
41:         if with_yggdrasil:
42:             flag = height_out.send(
43:                 flag, intensity = light_rpc.call(
44:                     [units.add_units(t, 'hrs'),
45:                      units.add_units(max(mesh.vertices[:, 2]), 'm')]))
46:
47:             # Compute the scale factor
48:             # Compute the scale factor using intensity, stripping units
49:             # of the result to allow use with trimesh
50:             # (pretend this is a biologically complex calculation)
51:             # (pretend this is a biologically complex calculation)
52:             scale = units.get_data(
53:                 units.add_units(mass, 'g') * intensity /
54:                 units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
55:         else:
56:             # Compute the scale factor
57:             # (pretend this is a biologically complex calculation)
58:             scale = mass / 4.5e4
59:             scale = mass / 4.5e4
? +++++
```

```
In [6]: tools.display_source_diff('models/shoot_v1.py', 'models/shoot_v2.py', number_lines=True)
```

```
file1: models/shoot_v1.py
file2: models/shoot_v2.py
=====
1: import os
2: import trimesh
3: import argparse
...
28: # If the model is running as part of an yggdrasil integration, import
29: # the relevant yggdrasil routines and use the interface routine to
30: # complete the connection defined in the YAML
31: if with_yggdrasil:
32:     from yggdrasil import units
-     from yggdrasil.languages.Python.YggInterface import YggOutput
?
^---^
33: +     from yggdrasil.languages.Python.YggInterface import YggRpcClient
?
^-----^

-     height_out = YggOutput('height')
?
^-----^
34: +     light_rpc = YggRpcClient('light_shoot')
?
^-----^
^-----^

35:
36: # Continue simulation until time limit is reached
37: while t <= tmax:
38:
39:     # If running as part an yggdrasil integration, send the time and
40:     # maximum height of the mesh to the height channel with units
41:     if with_yggdrasil:
-         flag = height_out.send(
?
        ^-----^
42: +         flag, intensity = light_rpc.call(
?
        +-----+
        ^-----^
        ^-----^
43:             [units.add_units(t, 'hrs'),
44:              units.add_units(max(mesh.vertices[:, 2]), 'm')])

...

```

Replace the interface function

```
In [6]: tools.display_source_diff('models/shoot_v1.py', 'models/shoot_v2.py', number_lines=True)
```

```
file1: models/shoot_v1.py
file2: models/shoot_v2.py
=====
1: import os
2: import trimesh
3: import argparse
...
28: # If the model is running as part of an yggdrasil integration, import
29: # the relevant yggdrasil routines and use the interface routine to
30: # complete the connection defined in the YAML
31: if with_yggdrasil:
32:     from yggdrasil import units
33:     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     from yggdrasil.languages.Python.YggInterface import YggRpcClient
35:
36: # Continue simulation until time limit is reached
37: while t <= tmax:
38:
39:     # If running as part an yggdrasil integration, send the time and
40:     # maximum height of the mesh to the height channel with units
41:     if with_yggdrasil:
42:         flag = height_out.send(
43:             [units.add_units(t, 'hrs'),
44:              units.add_units(max(mesh.vertices[:, 2]), 'm')])
45:
46:     # Compute the scale factor
47: +     scale = units.get_data(
48:         units.add_units(mass, 'g') * intensity /
49:         units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
50: +     else:
51:         # Compute the scale factor
52:         scale = mass / 4.5e4
53:     # (pretend this is a biologically complex calculation)
54: +     scale = mass / 4.5e4
55: ? +++++
```

Replace the send with a call

```
In [6]: tools.display_source_diff('models/shoot_v1.py', 'models/shoot_v2.py', number_lines=True)
```

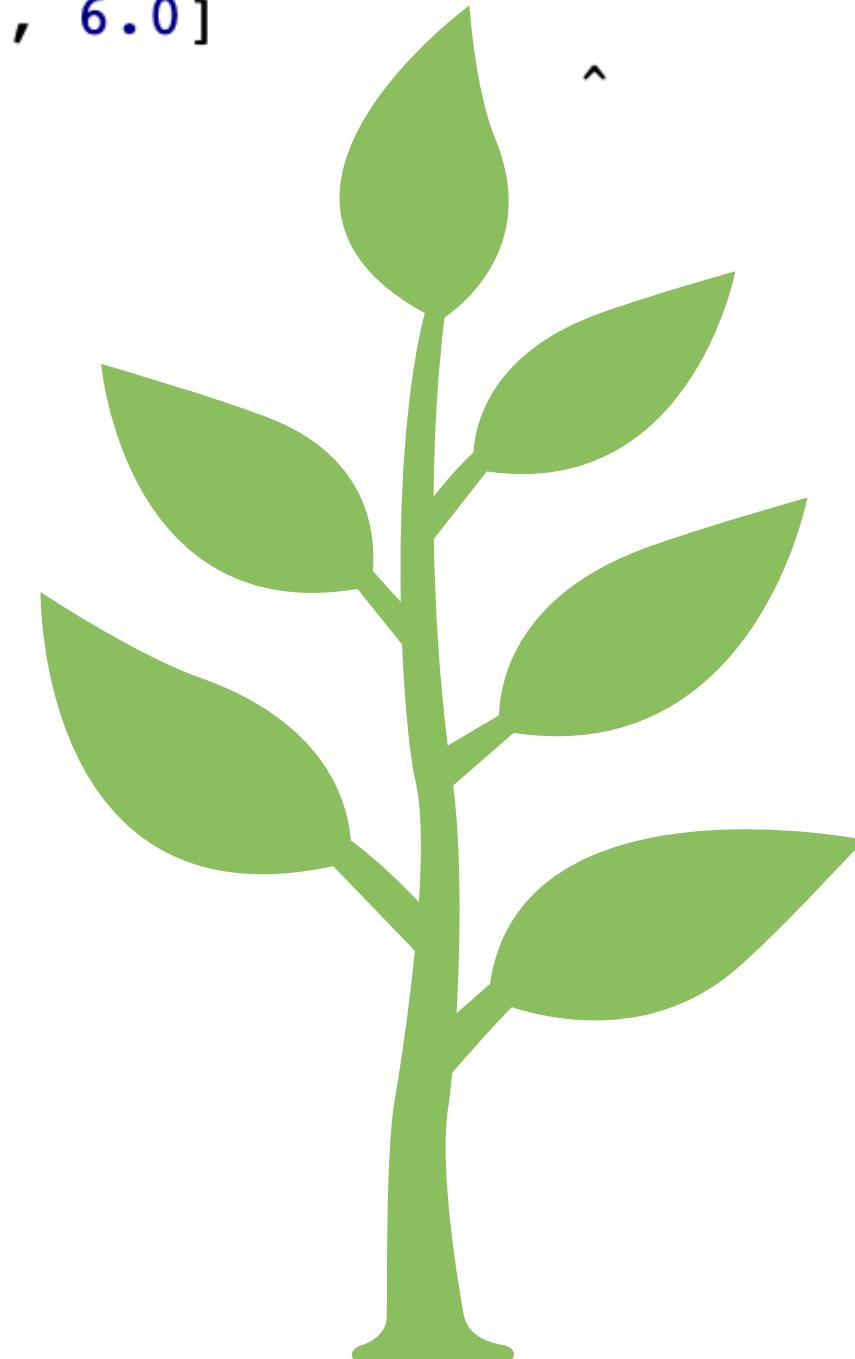
```
file1: models/shoot_v1.py
file2: models/shoot_v2.py
=====
1: import os
2: import trimesh
3: import argparse
...
28: # If the model is running as part of an yggdrasil integration, import
29: # the relevant yggdrasil routines and use the interface routine to
30: # complete the connection defined in the YAML
31: if with_yggdrasil:
32:     from yggdrasil import units
33:     from yggdrasil.languages.Python.YggInterface import YggOutput
34:     from yggdrasil.languages.Python.YggInterface import YggRpcClient
35:
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39:         # If running as part an yggdrasil integration, send the time and
40:         # maximum height of the mesh to the height channel with units
41:         if with_yggdrasil:
42:             flag = height_out.send(
43:                 flag, intensity = light_rpc.call(
44:                     [units.add_units(t, 'hrs'),
45:                      units.add_units(max(mesh.vertices[:, 2]), 'm')]))
46:
47:             # Compute the scale factor
48:             # Compute the scale factor using intensity, stripping units
49:             # of the result to allow use with trimesh
50:             # (pretend this is a biologically complex calculation)
51:             # (pretend this is a biologically complex calculation)
52:             scale = units.get_data(
53:                 units.add_units(mass, 'g') * intensity /
54:                 units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
55:         else:
56:             # Compute the scale factor
57:             # (pretend this is a biologically complex calculation)
58:             scale = mass / 4.5e4
59:             scale = mass / 4.5e4
? +++++
```

Change how scale is computed when
yggdrasil used

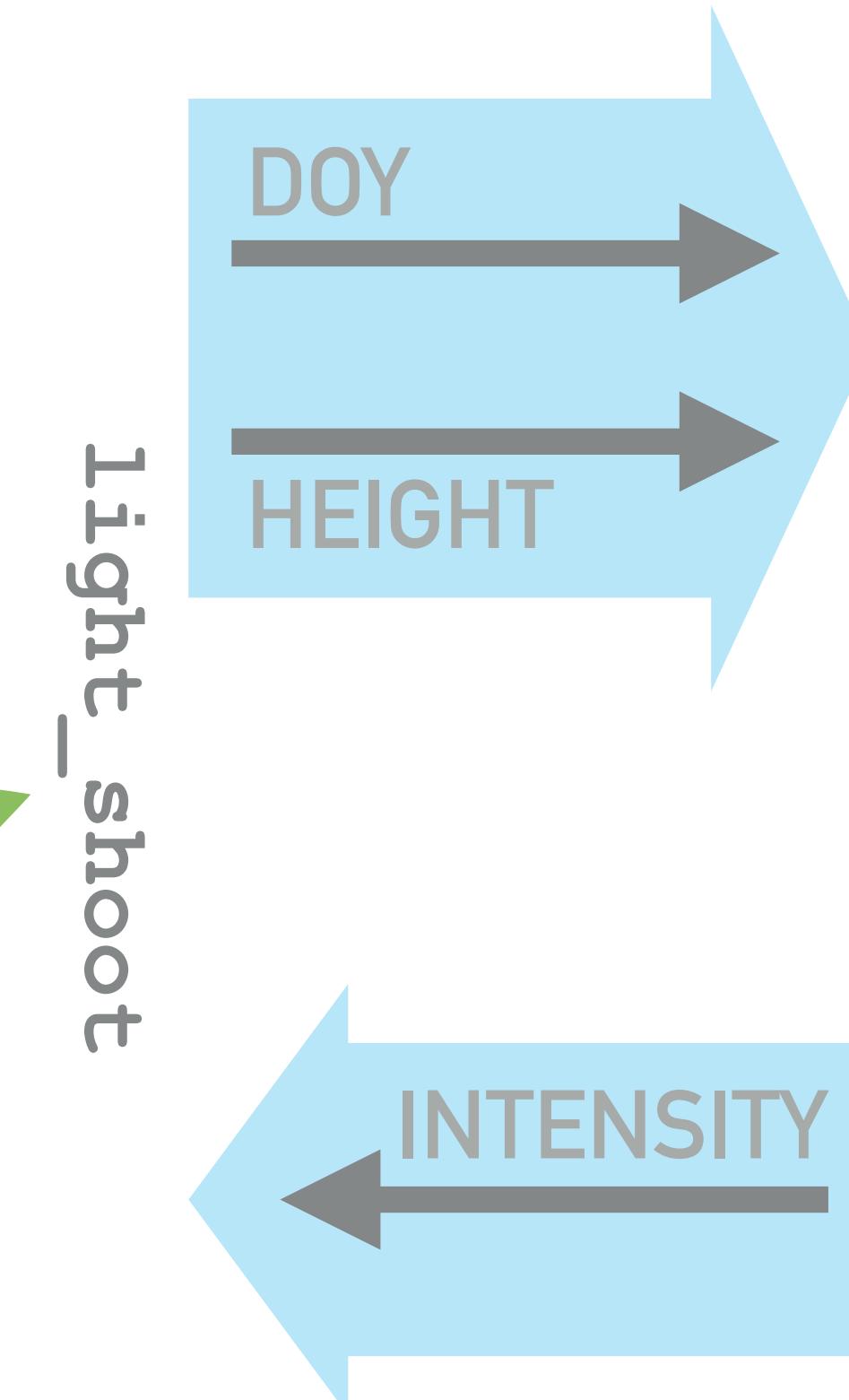
```
In [7]: tools.display_source_diff('yamls/shoot_v1.yml', 'yamls/shoot_v2.yml', number_lines=True)
tools.display_source_diff('yamls/light_v0_python.yml', 'yamls/light_v1_python.yml', number_lines=True)
```

```
file1: yamls/shoot_v1.yml
file2: yamls/shoot_v2.yml
=====
1:   model:
2:     name: shoot
3:     language: python
4:     - args: [../models/shoot_v1.py, 0.0, 48.0, 6.0]
5:     ?
6:
7: 4: +   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
8:  ?
9:
10: -   outputs:
11: -     - name: height
12: ?   ----
13: 5: +   client_of: light
14: ?     +--+ ^^^
15:
16: -     default_file:
17: -       name: ../output/height.txt
18: -       filetype: table
19:
20: file1: yamls/light_v0_python.yml
21: file2: yamls/light_v1_python.yml
=====
```

1: model:
2: name: shoot
3: language: python
4: - args: [../models/shoot_v1.py, 0.0, 48.0, 6.0]
5: ?
6:
7: 4: + args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
8: ?
9:
10: - outputs:
11: - - name: height
12: ? ----
13: 5: + client_of: light
14: ? +--+ ^^^
15:
16: - default_file:
17: - name: ../output/height.txt
18: - filetype: table
19:
20: file1: yamls/light_v0_python.yml
21: file2: yamls/light_v1_python.yml
=====



SHOOT
MODEL



light:input light:output

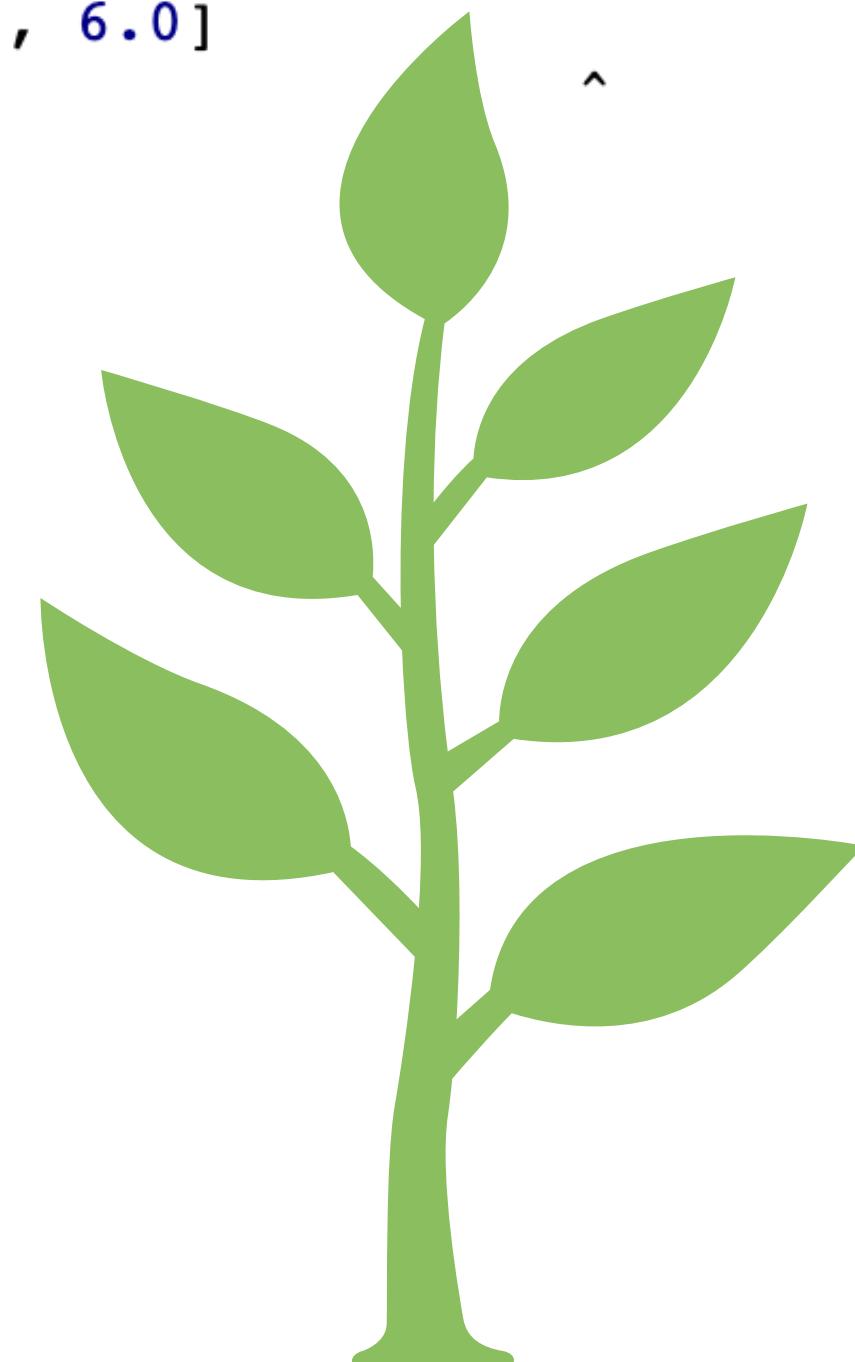


LIGHT
MODEL

```
In [7]: tools.display_source_diff('yamls/shoot_v1.yml', 'yamls/shoot_v2.yml', number_lines=True)
tools.display_source_diff('yamls/light_v0_python.yml', 'yamls/light_v1_python.yml', number_lines=True)
```

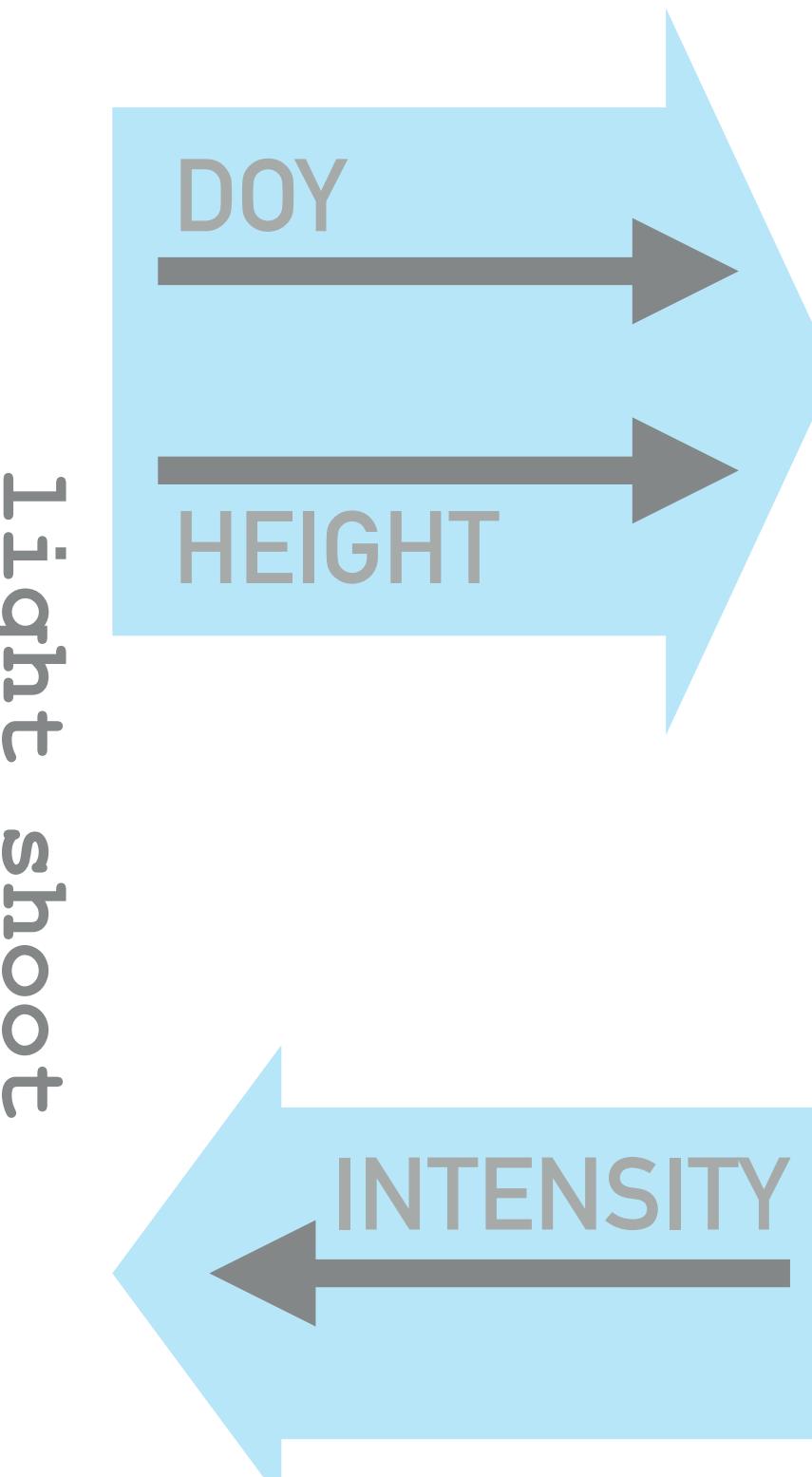
```
file1: yamls/shoot_v1.yml
file2: yamls/shoot_v2.yml
=====
1:   model:
2:     name: shoot
3:     language: python
4:     args: [../models/shoot_v1.py, 0.0, 48.0, 6.0]
5:     ?
6:
7: 4: +   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
8:    ?
9:
10: -   outputs:
11: -     - name: height
12:    ?   ----
13: 5: +   client_of: light
14:    ?   +--+ ^^^
15:    -
16: -     default_file:
17: -       name: ../output/height.txt
18: -       filetype: table
```

```
file1: yamls/light_v0_python.yml
file2: yamls/light_v1_python.yml
=====
1:   model:
2:     name: light
3:     language: python
4:     args: ../models/light_v0.py
5:     function: light
6: +   is_server: true
```



SHOOT
MODEL

Remove output & add "client_of" to
shoot model



light_shoot

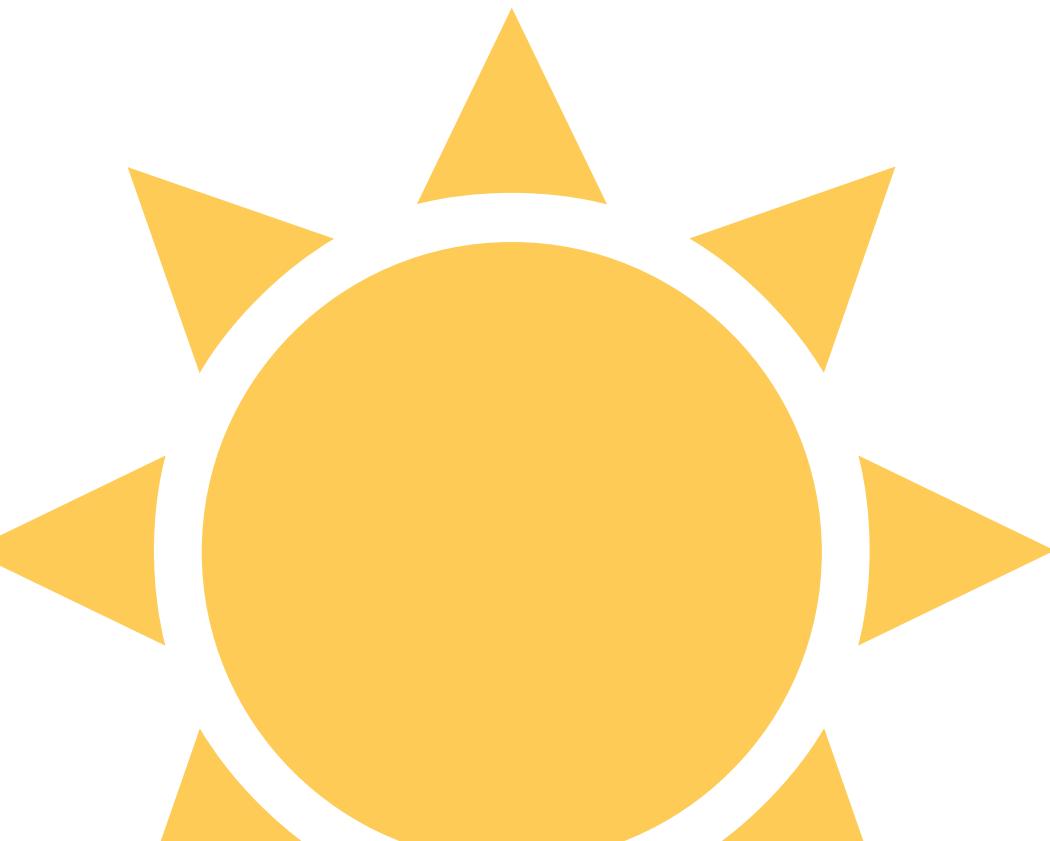
DOY

HEIGHT

INTENSITY

light:input

light:output

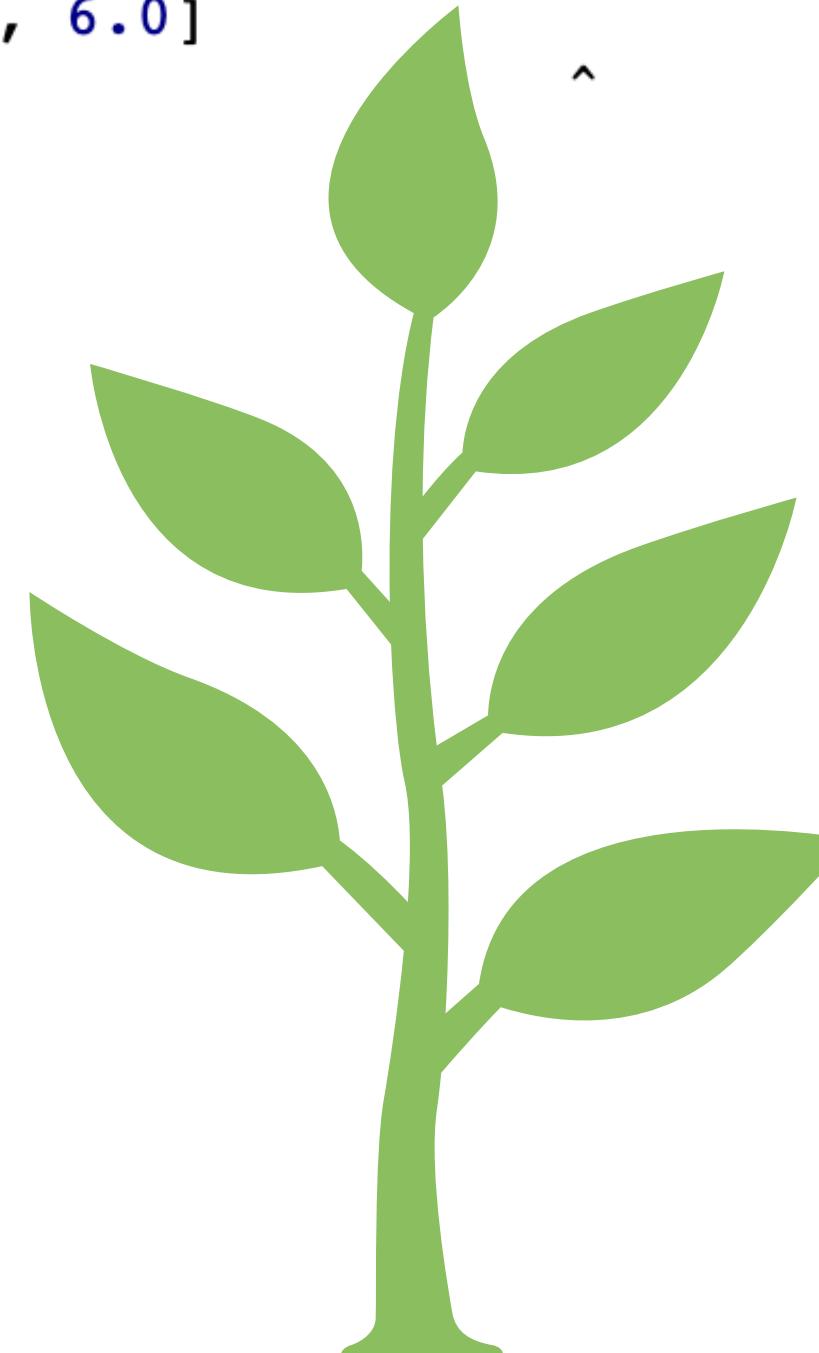


LIGHT
MODEL

```
In [7]: tools.display_source_diff('yamls/shoot_v1.yml', 'yamls/shoot_v2.yml', number_lines=True)
tools.display_source_diff('yamls/light_v0_python.yml', 'yamls/light_v1_python.yml', number_lines=True)
```

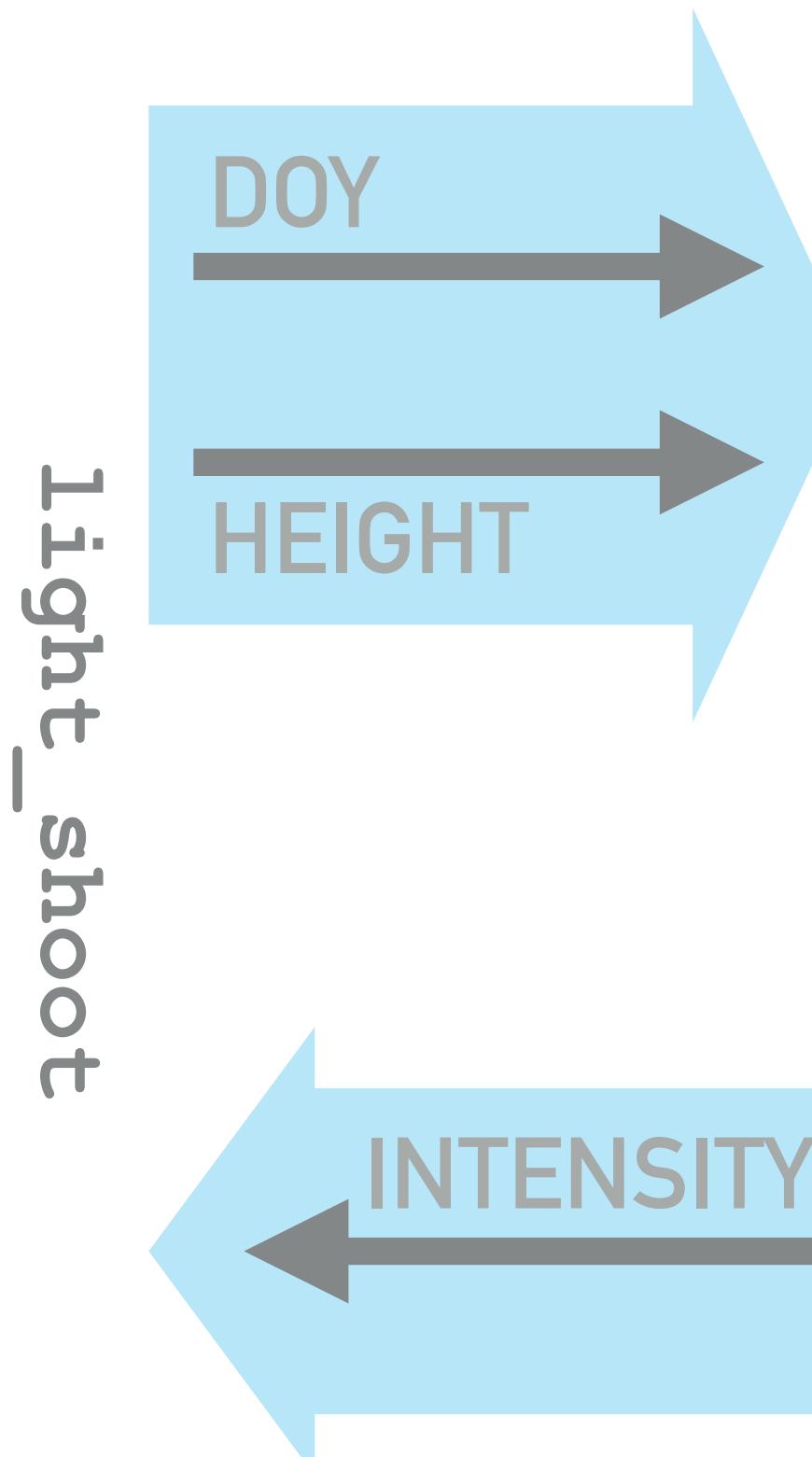
```
file1: yamls/shoot_v1.yml
file2: yamls/shoot_v2.yml
=====
1:   model:
2:     name: shoot
3:     language: python
4:     args: [../models/shoot_v1.py, 0.0, 48.0, 6.0]
5:     ?
6:
7:     +
8:       args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
9:       ?
10:
11:      -
12:        outputs:
13:          - name: height
14:          ?
15:          ----
16:          ^
17:          ^
18:          ^^^
19:          ^^^^
20:          ^^^^
21:          ^
22:
23:      +
24:        client_of: light
25:        +
26:          ++
27:          ^^^
28:          ^
29:
30:          -
31:            default_file:
32:              name: ../output/height.txt
33:              filetype: table
34:
35:
36: file1: yamls/light_v0_python.yml
37: file2: yamls/light_v1_python.yml
=====
```

1: model:
2: name: light
3: language: python
4: args: ../models/light_v0.py
5: function: light
6: + is_server: true

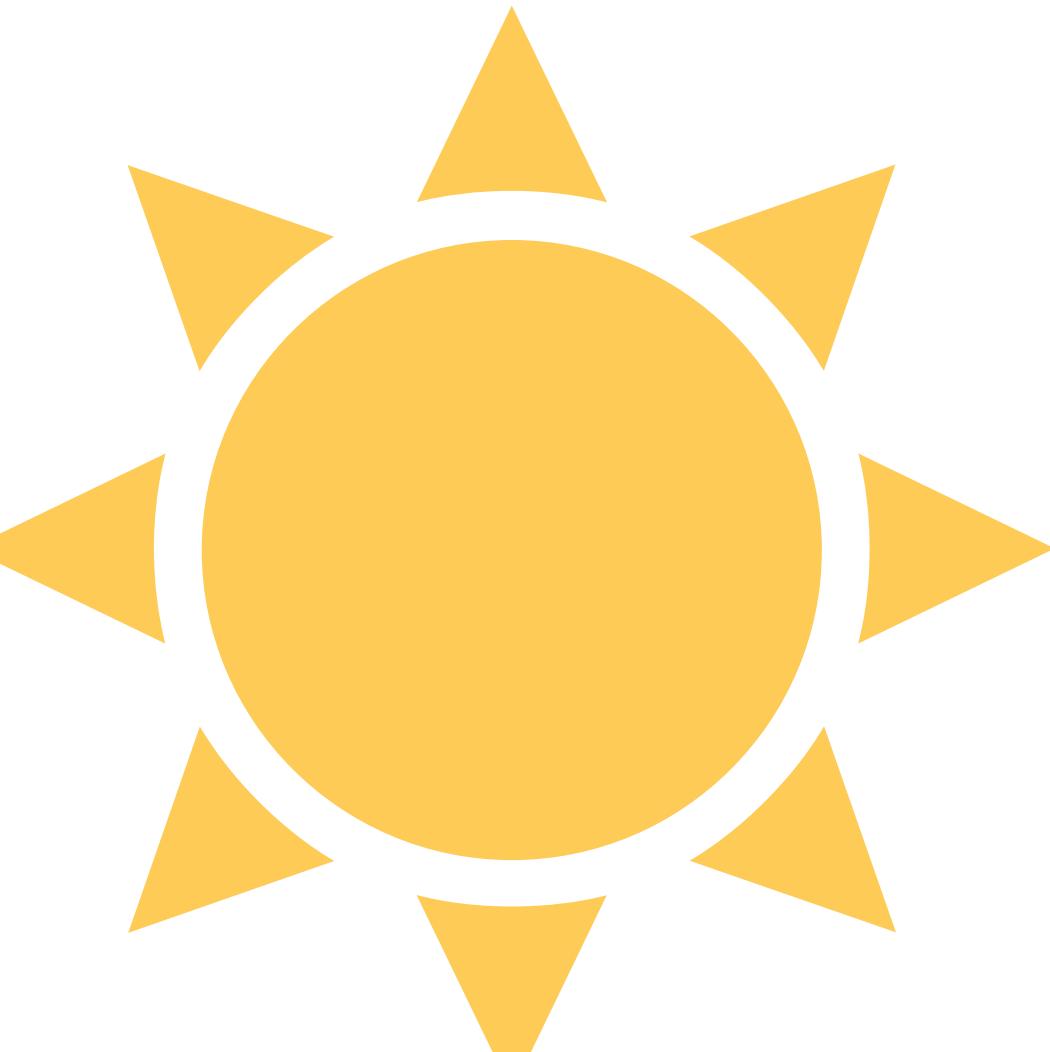


SHOOT
MODEL

Add "is_server" to light model



light:input light:output

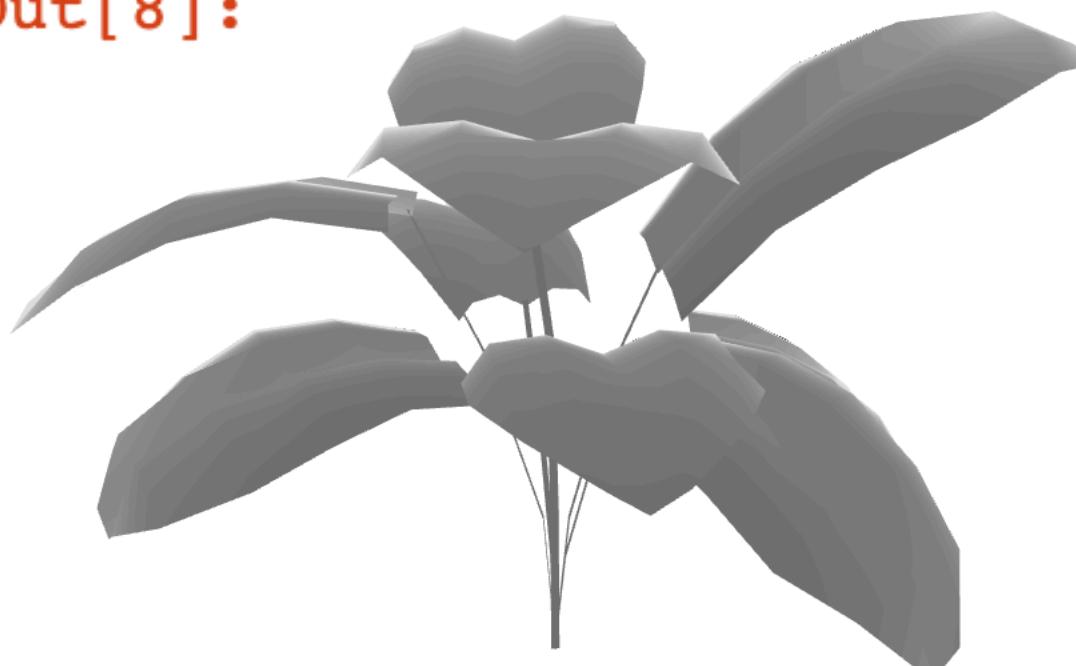


LIGHT
MODEL

```
In [8]: run(['yamls/light_v1_python.yml', 'yamls/shoot_v2.yml'], production_run=True)
mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.show()
```

```
INFO:93696:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/ygg
_light_v0.py
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/sho
ot_v2.py 0.0 48.0 6.0
End of input from temp_doy.
INFO:93696:runner.waitModels[553]:YggRunner(runner): light finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): light finished exiting.
INFO:93696:runner.waitModels[553]:YggRunner(runner): shoot finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:93696:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:93696:runner.run[374]:YggRunner(runner):
                           init      0.000001
INFO:93696:runner.run[374]:YggRunner(runner):
                           load drivers   0.010089
INFO:93696:runner.run[374]:YggRunner(runner):
                           start drivers  0.090419
INFO:93696:runner.run[374]:YggRunner(runner):
                           run models    7.769536
INFO:93696:runner.run[374]:YggRunner(runner):
                           at exit       0.044337
INFO:93696:runner.run[376]:YggRunner(runner): =====
INFO:93696:runner.run[377]:YggRunner(runner):           Total      7.914382
```

Out[8]:



SPLITTING RPC CALLS

```
In [9]: tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v2_split.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v2_split.py
=====
...
39:     # If running as part an yggdrasil integration, send the time and
40:     # maximum height of the mesh to the height channel with units
41:     if with_yggdrasil:
42:         # Send request to the light model
43:         flag, intensity = light_rpc.call(
44:             [units.add_units(t, 'hrs'),
45:              units.add_units(max(mesh.vertices[:, 2]), 'm')])
46:     if not flag:
47:         raise Exception("Error calling the light model.")
48:     raise Exception("Error sending request to the light model.")
+
49:     # Calculations that don't rely on the output from the light model
50:     # can be run here in parallel with the light model calculations
51:
52:     # Receive response from the light model
53:     flag, intensity = light_rpc.recv()
54:     if not flag:
55:         raise Exception("Error receiving response from the light model.")

...
```

```
In [9]: tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v2_split.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v2_split.py
=====
...
39:     # If running as part an yggdrasil integration, send the time and
40:     # maximum height of the mesh to the height channel with units
41:     if with_yggdrasil:
42:         # Send request to the light model
43:         flag, intensity = light_rpc.call(
44:             [units.add_units(t, 'hrs'),
45:              units.add_units(max(mesh.vertices[:, 2]), 'm')])
46:         if not flag:
47:             raise Exception("Error calling the light model.")
48:         ...
49:     # Calculations that don't rely on the output from the light model
50:     # can be run here in parallel with the light model calculations
51:     ...
52:     # Receive response from the light model
53:     flag, intensity = light_rpc.recv()
54:     if not flag:
55:         raise Exception("Error receiving response from the light model.")

...

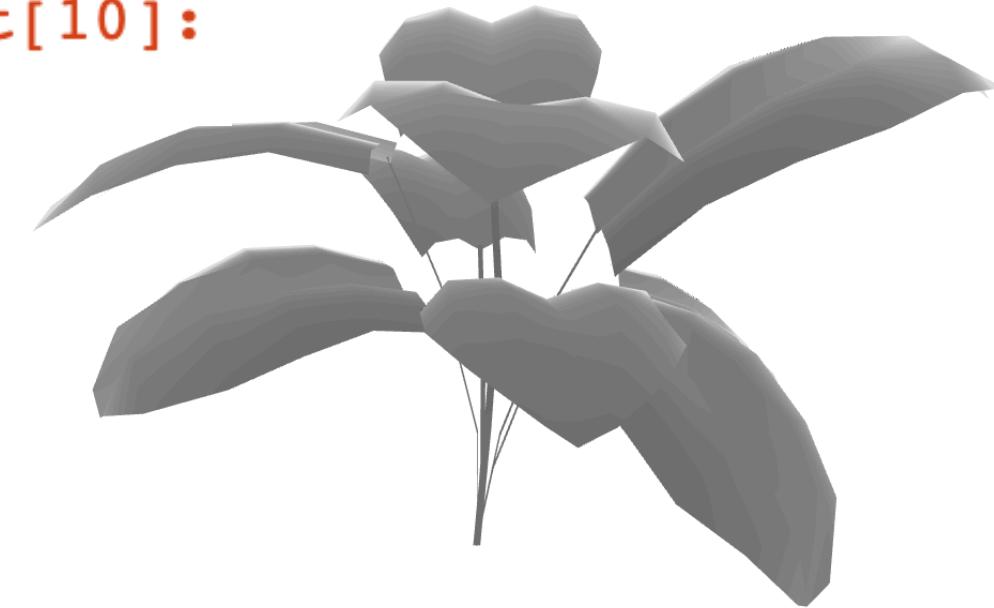
```

Calls can be split into
send & recv to enhance
parallelism

```
In [10]: run(['yamls/light_v1_python.yml', 'yamls/shoot_v2_split.yml'], production_run=True)
mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.show()
```

```
INFO:93696:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg
_light_v0.py
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/sho
ot_v2_split.py 0.0 48.0 6.0
End of input from temp_doy.
INFO:93696:runner.waitModels[553]:YggRunner(runner): light finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): light finished exiting.
INFO:93696:runner.waitModels[553]:YggRunner(runner): shoot finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:93696:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:93696:runner.run[374]:YggRunner(runner):
                           init      0.000001
INFO:93696:runner.run[374]:YggRunner(runner):
                           load drivers   0.009048
INFO:93696:runner.run[374]:YggRunner(runner):
                           start drivers  0.087974
INFO:93696:runner.run[374]:YggRunner(runner):
                           run models    7.865753
INFO:93696:runner.run[374]:YggRunner(runner):
                           at exit       0.044422
INFO:93696:runner.run[376]:YggRunner(runner): =====
INFO:93696:runner.run[377]:YggRunner(runner):
                           Total      8.007198
```

Out[10]:



DUPPLICATING MODELS

```
In [11]: tools.display_source_diff('models/shoot_v2_split.py', 'models/shoot_v2_copies.py', number_lines=True)
```

```
file1: models/shoot_v2_split.py
file2: models/shoot_v2_copies.py
=====
...
41:     # If running as part an yggdrasil integration, send the time and
42:     # maximum height of the mesh to the height channel with units
43:     if with_yggdrasil:
44:         -           # Send request to the light model
44:         +           # Send requests to the light model for each mesh vertex
44:         ?
45:         +           +
45:         for v in mesh.vertices[:, 2]:
46:             -           flag = light_rpc.send(
46:             +           flag = light_rpc.send(
46:             ? ++++
47:             -           [units.add_units(t, 'hrs'),
47:             +           [units.add_units(t, 'hrs'),
47:             ? ++++
48:             -           units.add_units(max(mesh.vertices[:, 2]), 'm'))
48:             +           units.add_units(v, 'm'))
48:             -           if not flag:
49:             +           if not flag:
49:             ? ++++
...

```

```
In [11]: tools.display_source_diff('models/shoot_v2_split.py', 'models/shoot_v2_copies.py', number_lines=True)
```

```
file1: models/shoot_v2_split.py
file2: models/shoot_v2_copies.py
=====
...
41:     # If running as part an yggdrasil integration, send the time and
42:     # maximum height of the mesh to the height channel with units
43:     if with_yggdrasil:
44:         -           # Send request to the light model
44: +           # Send requests to the light model for each mesh vertex
44: ?
45: +           for v in mesh.vertices[:, 2]:
45: -
46: +               flag = light_rpc.send(
46: ?       +++)
46: -
47: +                   [units.add_units(t, 'hrs'),
47: ?       +++)
47: -
48: +                   units.add_units(max(mesh.vertices[:, 2]), 'm'))
48: +                   units.add_units(v, 'm'))
48: -
49: +               if not flag:
49: +                   if not flag:
49: ?
49: ...

```

Send heights for each vertex to the light model


```
In [11]: tools.display_source_diff('models/shoot_v2_split.py', 'models/shoot_v2_copies.py', number_lines=True)
```

```
file1: models/shoot_v2_split.py
file2: models/shoot_v2_copies.py
=====
...
56: +         nvert = mesh.vertices.shape[0]
57: +         intensity = np.zeros(nvert, 'f8')
58: +         for iv in range(nvert):
59: -             flag, intensity = light_rpc.recv()
59: +             flag, v_intensity = light_rpc.recv()
      ? ++++
      ? ++++
      -         if not flag:
60: +             if not flag:
      ? ++++
      -             raise Exception("Error receiving response from the light model.")
61: +             raise Exception("Error receiving response from the light model.")
      ? ++++
      ? ++++
      -         if not units.has_units(intensity):
62: +             intensity = units.add_units(intensity,
63: +                                         units.get_units(v_intensity))
64: +             intensity[iv] = v_intensity
65: +             filename_light = os.path.join(_dir, f'../output/light_{i:03d}.pkl')
66: +             with open(filename_light, 'wb') as fd:
67: +                 pickle.dump(intensity, fd)
68: +
...

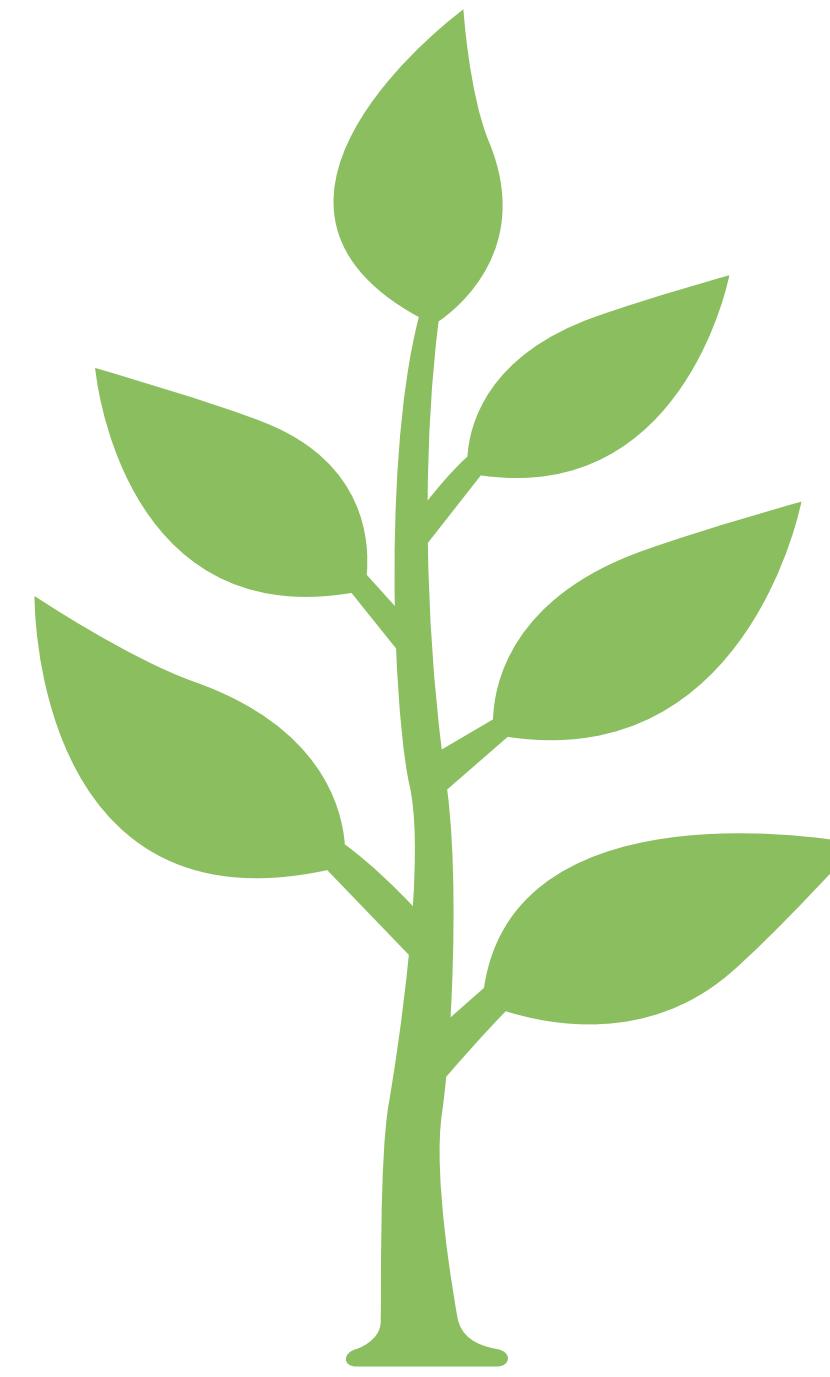
```

Receive intensities for each vertex

```
In [11]: tools.display_source_diff('models/shoot_v2_split.py', 'models/shoot_v2_copies.py', number_lines=True)
```

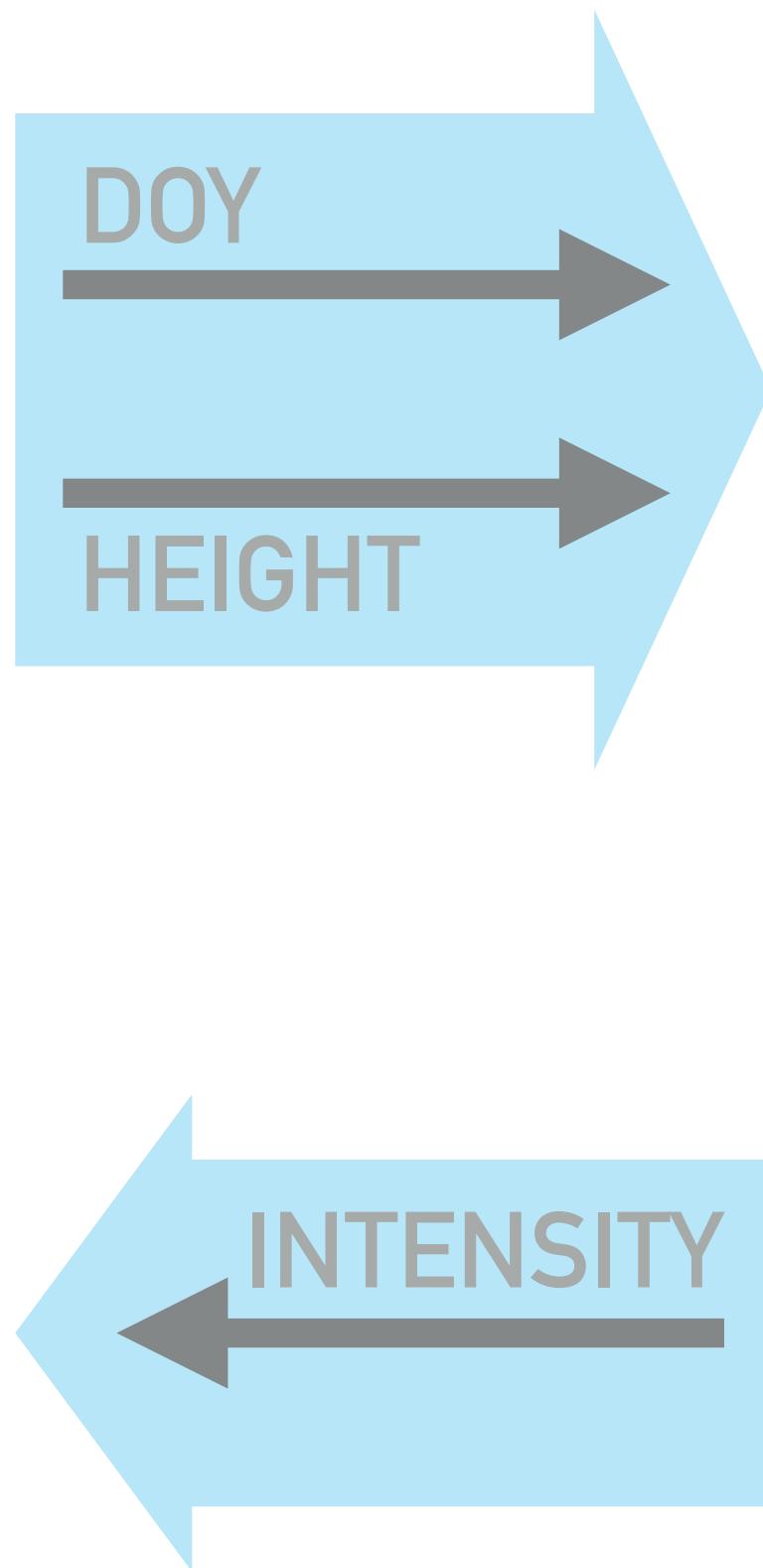
```
file1: models/shoot_v2_split.py
file2: models/shoot_v2_copies.py
=====
...
56: +         nvert = mesh.vertices.shape[0]
57: +         intensity = np.zeros(nvert, 'f8')
58: +         for iv in range(nvert):
59: -             flag, intensity = light_rpc.recv()
60: +             flag, v_intensity = light_rpc.recv()
61: ? ++++
62: -             if not flag:
63: +                 if not flag:
64: ? ++++
65: -                     raise Exception("Error receiving response from the light model.")
66: +                     raise Exception("Error receiving response from the light model.")
67: ? ++++
68: +         if not units.has_units(intensity):
69: +             intensity = units.add_units(intensity,
70: +                                         units.get_units(v_intensity))
71: +             intensity[iv] = v_intensity
72: +             filename_light = os.path.join(_dir, f'../output/light_{i:03d}.pkl')
73: +             with open(filename_light, 'wb') as fd:
74: +                 pickle.dump(intensity, fd)
```

Save intensities for each vertex to a file

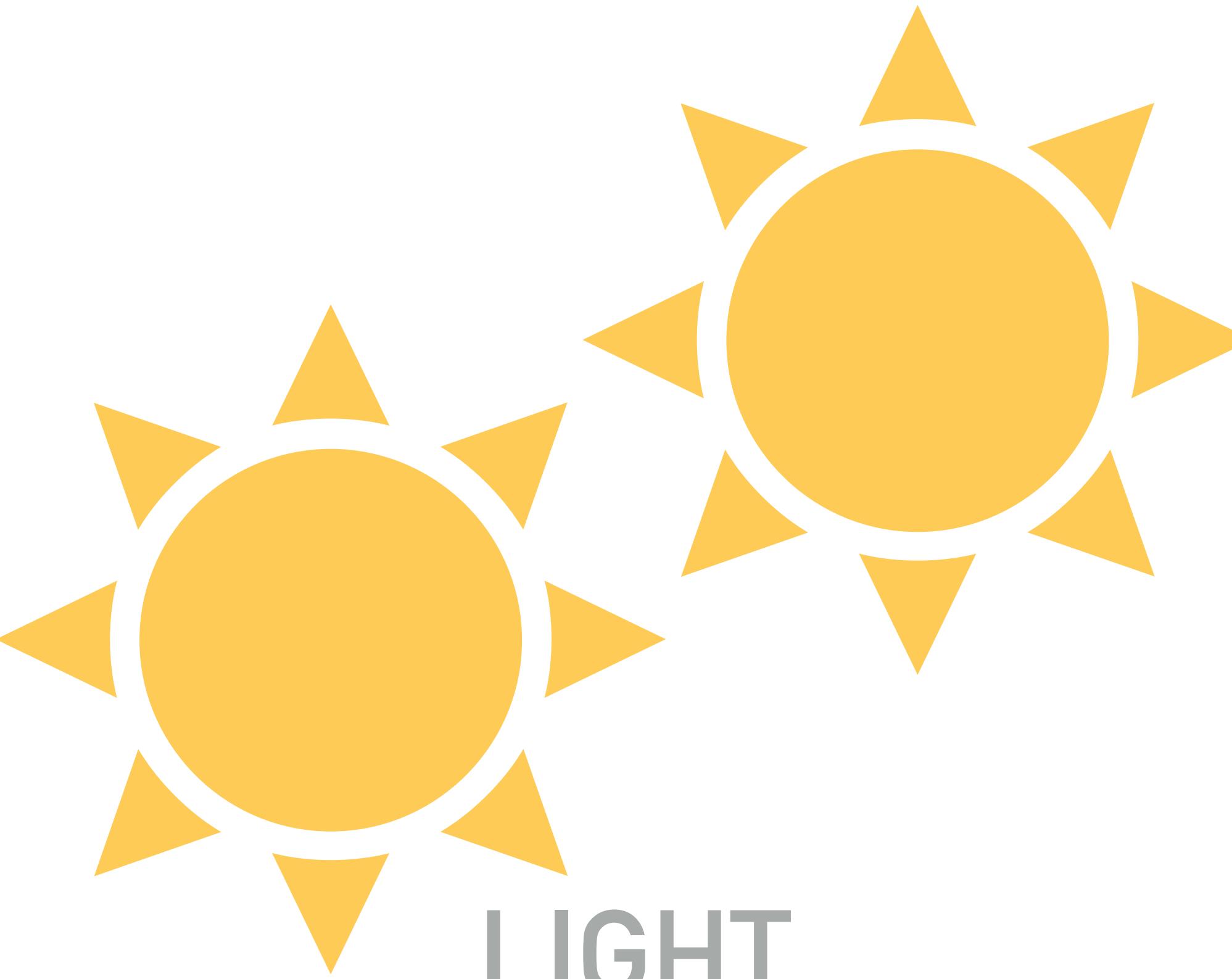


SHOOT
MODEL

light_shoot



light:input



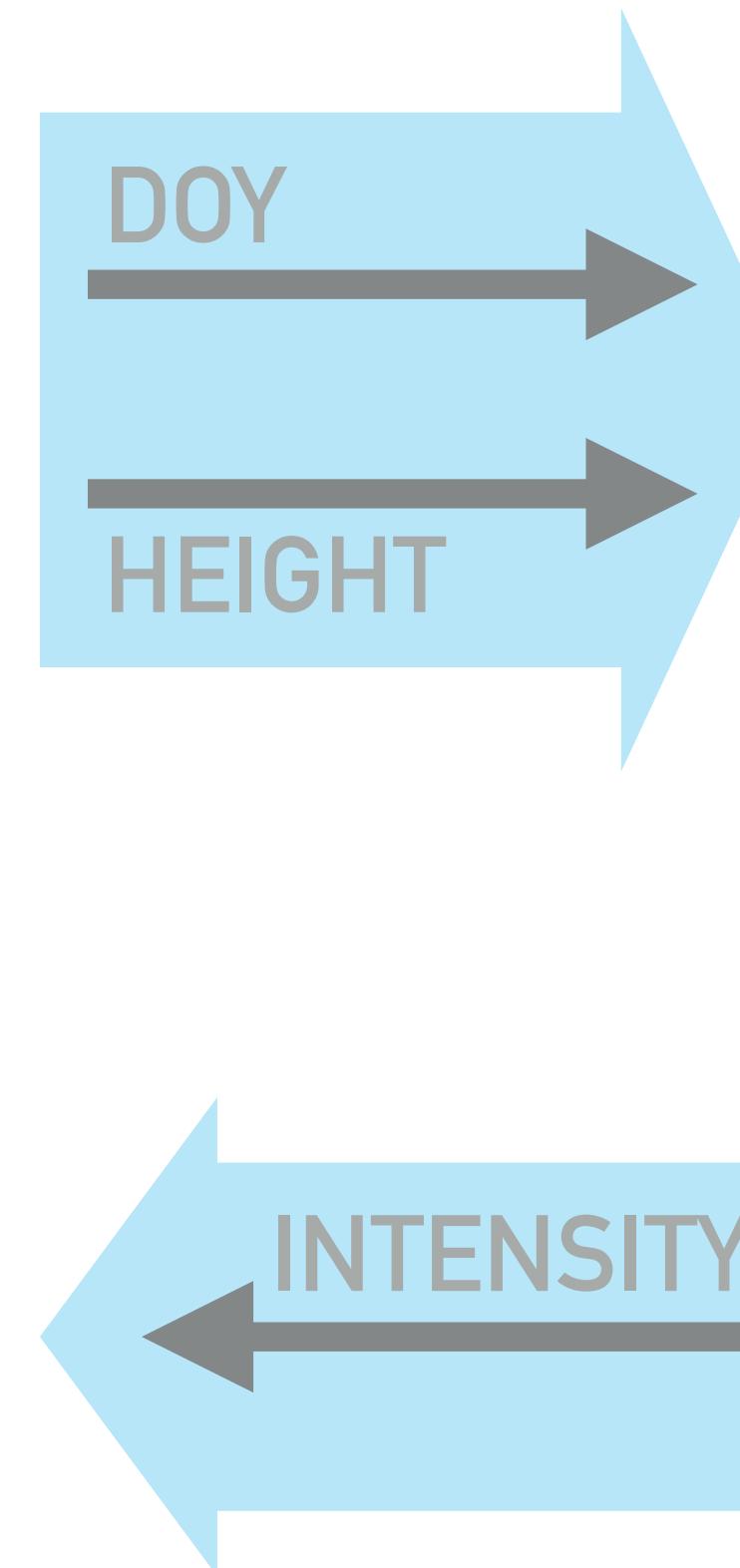
```
In [12]: tools.display_source_diff('yamls/light_v1_python.yml', 'yamls/light_v2_python.yml', number_lines=True)
```

```
file1: yamls/light_v1_python.yml
file2: yamls/light_v2_python.yml
=====
1:   model:
2:     name: light
3:     language: python
4:     args: ../models/light_v0.py
5:     function: light
6:     is_server: true
7: +   copies: 2
```

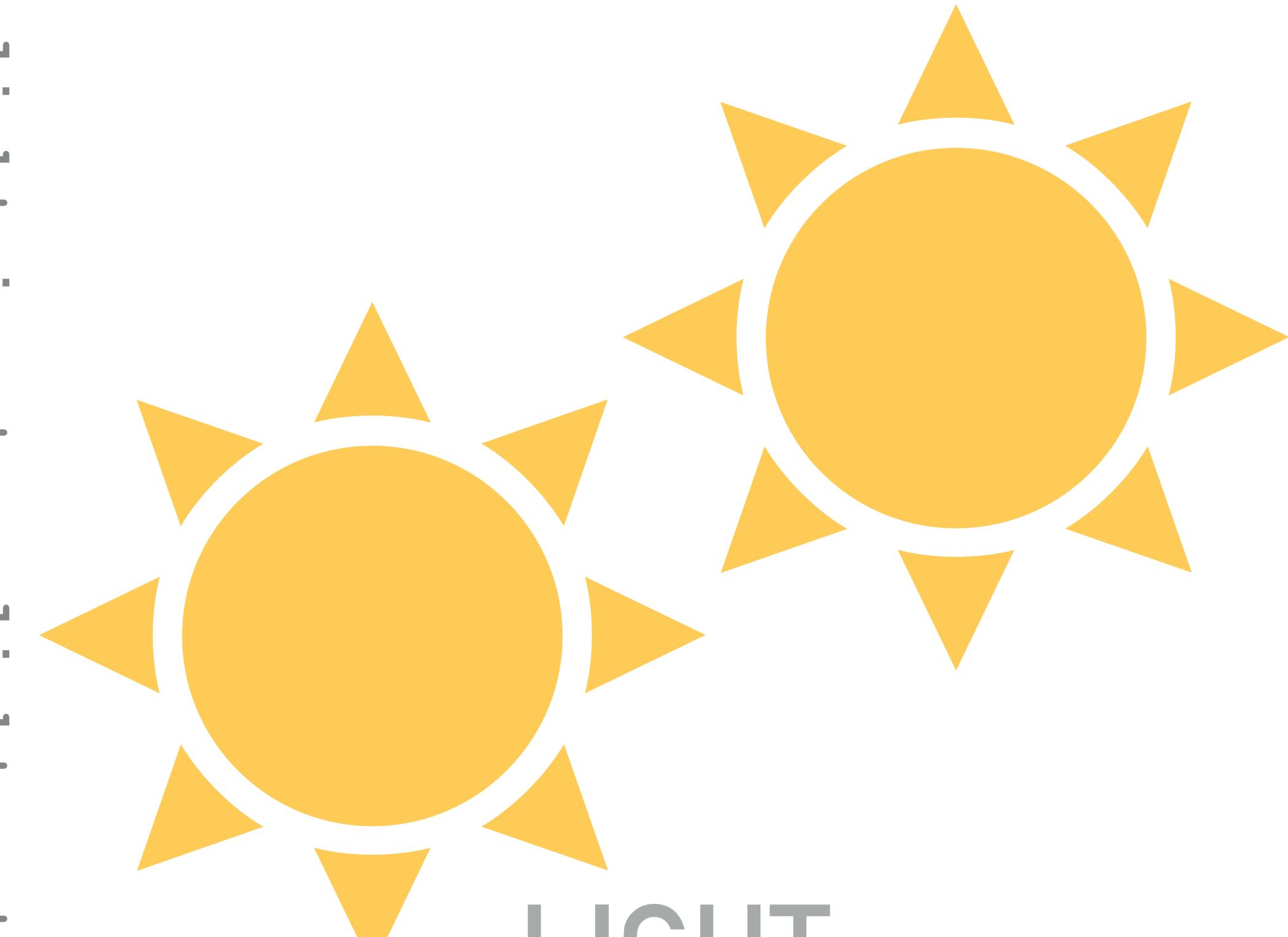


SHOOT
MODEL

light_shoot



light:input
light:output



LIGHT
MODELS

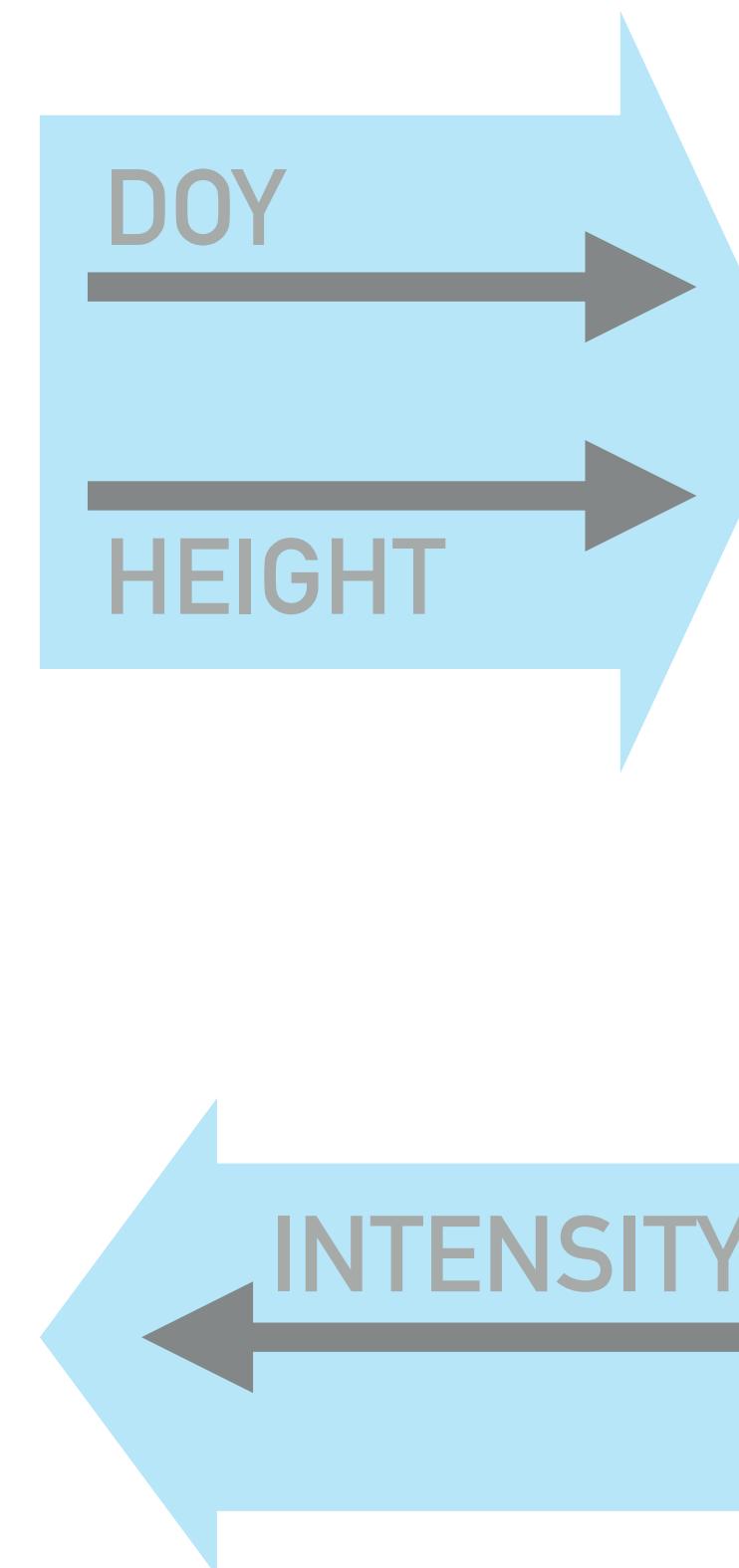
```
In [12]: tools.display_source_diff('yamls/light_v1_python.yml', 'yamls/light_v2_python.yml', number_lines=True)
```

```
file1: yamls/light_v1_python.yml
file2: yamls/light_v2_python.yml
=====
1:   model:
2:     name: light
3:     language: python
4:     args: ../models/light_v0.py
5:     function: light
6:     is_server: true
7: +   copies: 2
```



SHOOT
MODEL

light_shoot



Add “copies” to indicate multiple instances of a model should be run

light:input
light:output



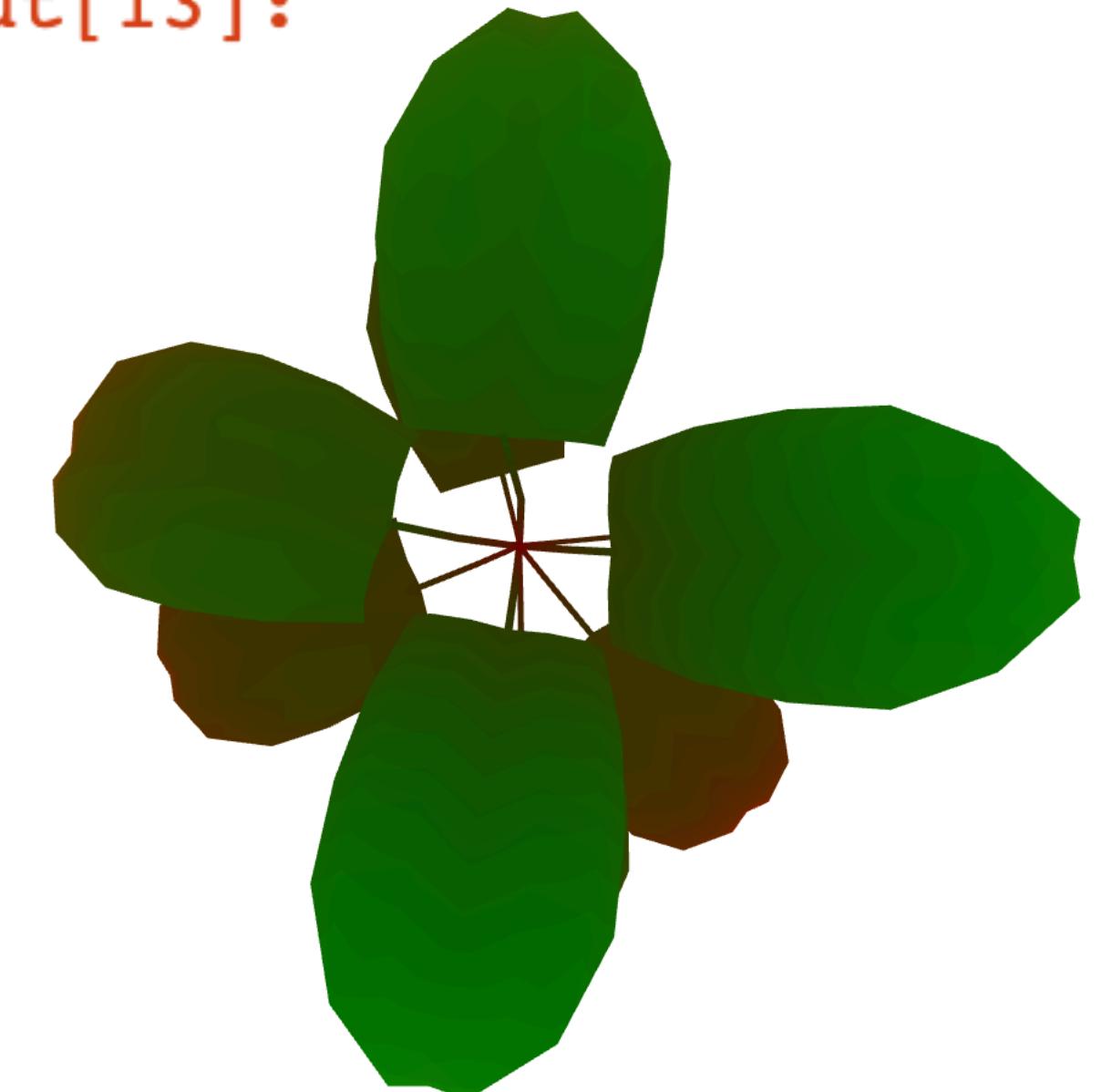
LIGHT
MODELS

```
In [13]: run(['yamls/light_v1_python.yml', 'yamls/shoot_v2_copies.yml'], production_run=True)
```

```
# Plot results w/ light intensity mapped to color
import pickle
with open('output/light_008.pkl', 'rb') as fd:
    light = pickle.load(fd)
mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.visual.vertex_colors = trimesh.visual.interpolate(light/max(light))
mesh.show()
```

```
INFO:93696:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg
_light_v0.py
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/sho
ot_v2_copies.py 0.0 48.0 6.0
End of input from temp_doy.
INFO:93696:runner.waitModels[553]:YggRunner(runner): light finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): light finished exiting.
INFO:93696:runner.waitModels[553]:YggRunner(runner): shoot finished running.
INFO:93696:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:93696:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:93696:runner.run[374]:YggRunner(runner):           init      0.000000
INFO:93696:runner.run[374]:YggRunner(runner):       load drivers   0.009468
INFO:93696:runner.run[374]:YggRunner(runner):     start drivers   0.086156
INFO:93696:runner.run[374]:YggRunner(runner):       run models  107.365030
INFO:93696:runner.run[374]:YggRunner(runner):      at exit     0.028054
INFO:93696:runner.run[376]:YggRunner(runner): =====
INFO:93696:runner.run[377]:YggRunner(runner):           Total  107.488708
```

Out[13]:

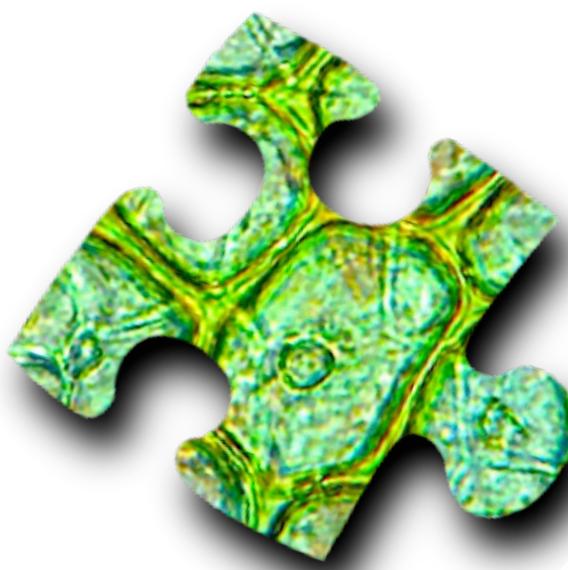
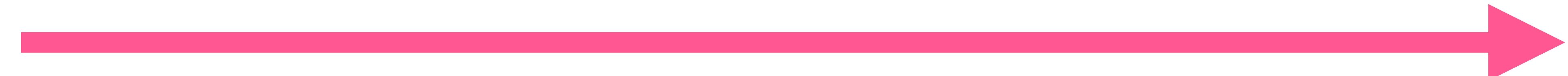


PARALLELISM ON MYBINDER

MyBinder instances only have access to 1 core (models run in serial)



MODEL A



MODEL B



PARALLELISM ON MYBINDER

MyBinder instances only have access to 1 core (models run in serial)



MODEL A



MODEL B



NEW NOTEBOOK!
(BREAK TIME)

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) [⟳](#)

<input type="checkbox"/>	0	▼	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images				33 minutes ago	
<input type="checkbox"/>	 input				33 minutes ago	
<input type="checkbox"/>	 meshes				33 minutes ago	
<input type="checkbox"/>	 models				33 minutes ago	
<input type="checkbox"/>	 yaml				33 minutes ago	
<input type="checkbox"/>	 00-intro.ipynb				33 minutes ago	457 kB
<input type="checkbox"/>	 01-connections.ipynb				33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb				33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb				33 minutes ago	3.56 kB

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#)

<input type="checkbox"/> 0	/	Name	Last Modified	File size
<input type="checkbox"/>	images		33 minutes ago	
<input type="checkbox"/>	input		33 minutes ago	
<input type="checkbox"/>	meshes		33 minutes ago	
<input type="checkbox"/>	models		33 minutes ago	
<input type="checkbox"/>	yaml		33 minutes ago	
<input type="checkbox"/>	00-intro.ipynb		33 minutes ago	457 kB
<input type="checkbox"/>	01-connections.ipynb		33 minutes ago	470 kB
<input type="checkbox"/>	02-timesync.ipynb		33 minutes ago	298 kB
<input type="checkbox"/>	03-misc.ipynb		33 minutes ago	3.56 kB

TIME STEP SYNCHRONIZATION

```
In [2]: tools.display_source('models/roots_v0.py', number_lines=True)
```

```
file: models/roots_v0.py
=====
1: import os
2: import argparse
3: import pickle
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate root growth over time.")
9: parser.add_argument('tmin', help='Starting time (in days)', type=float)
10: parser.add_argument('tmax', help='Ending time (in days)', type=float)
11: parser.add_argument('tstep', help='Time step (in days)', type=float)
12: args = parser.parse_args()
13: tmin = args.tmin
14: tmax = args.tmax
15: tstep = args.tstep
16:
17: # Set initial conditions
18: mass = 0.0
19: t = tmin
20: times = []
21: masses = []
22:
23: # Continue simulation until time limit is reached
24: while t <= tmax:
25:
26:     # Compute the scale factor
27:     # (pretend this is a biologically complex calculation)
28:     scale = 0.2
29:
30:     # Calculate mass for the time step
31:     # (pretend this is a biologically complex calculation)
32:     mass += t * scale
33:
34:     # Add mass & time to array
35:     times.append(t)
36:     masses.append(mass)
37:
38:     # Advance time step
39:     t += tstep
40:
41:     # Write the total mass array to output
42:     filename_masses = os.path.join(_dir, '../output/masses.pkl')
43:     with open(filename_masses, 'wb') as fd:
44:         pickle.dump({'times': times, 'masses': masses}, fd)
...

```

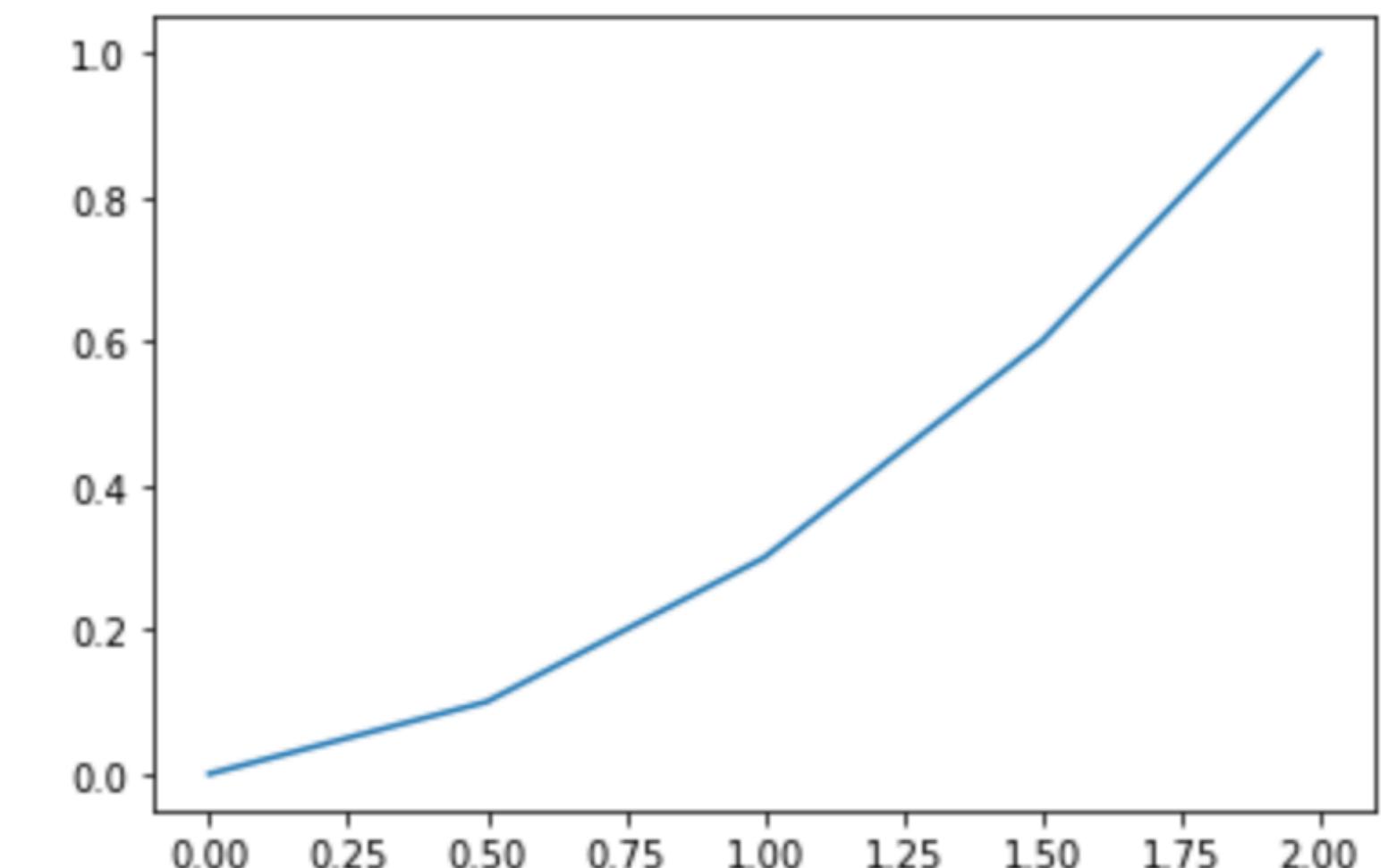
```
In [3]: tools.display_source('yamls/roots_v0.yml', number_lines=True)
run(['yamls/roots_v0.yml'], production_run=True)
```

```
file: yamls/roots_v0.yml
=====
1: model:
2:   name: roots
3:   language: python
4:   args: [./models/roots_v0.py, 0.0, 2.0, 0.5]
```

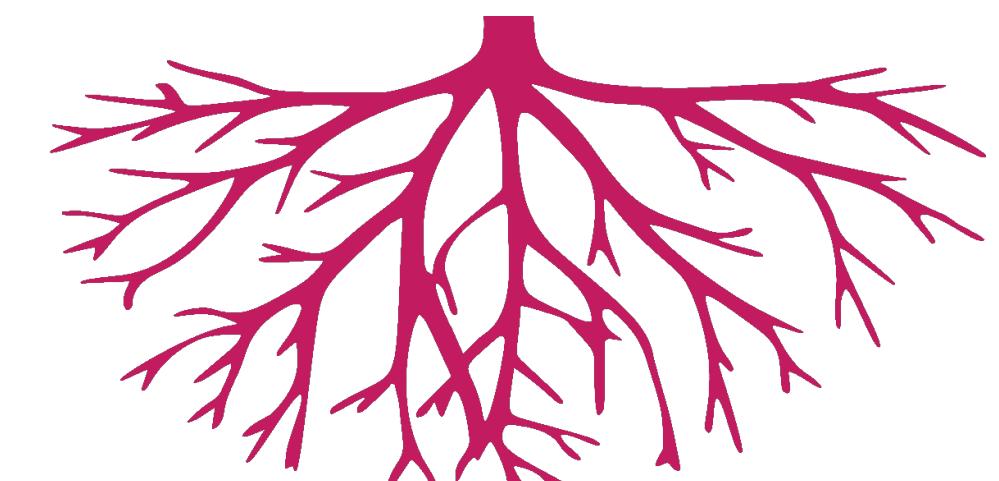
```
INFO:96257:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.
local in namespace yggdrasil with rank 0
/Users/langmnm/miniconda3/envs/conda36/bin/python /Users/langmnm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/roots_v0.py 0.0 2.0 0.5
INFO:96257:runner.waitModels[553]:YggRunner(runner): roots finished running.
INFO:96257:runner.waitModels[559]:YggRunner(runner): roots finished exiting.
INFO:96257:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:96257:runner.run[374]:YggRunner(runner):           init      0.000000
INFO:96257:runner.run[374]:YggRunner(runner):       load drivers    0.279398
INFO:96257:runner.run[374]:YggRunner(runner):     start drivers    0.039544
INFO:96257:runner.run[374]:YggRunner(runner):      run models     0.103028
INFO:96257:runner.run[374]:YggRunner(runner):        at exit      0.000614
INFO:96257:runner.run[376]:YggRunner(runner): =====
INFO:96257:runner.run[377]:YggRunner(runner):           Total      0.422584
```

```
In [4]: import matplotlib.pyplot as plt
filename_masses = 'output/masses.pkl'
with open(filename_masses, 'rb') as fd:
    masses = pickle.load(fd)
plt.plot(masses['times'], masses['masses'])
```

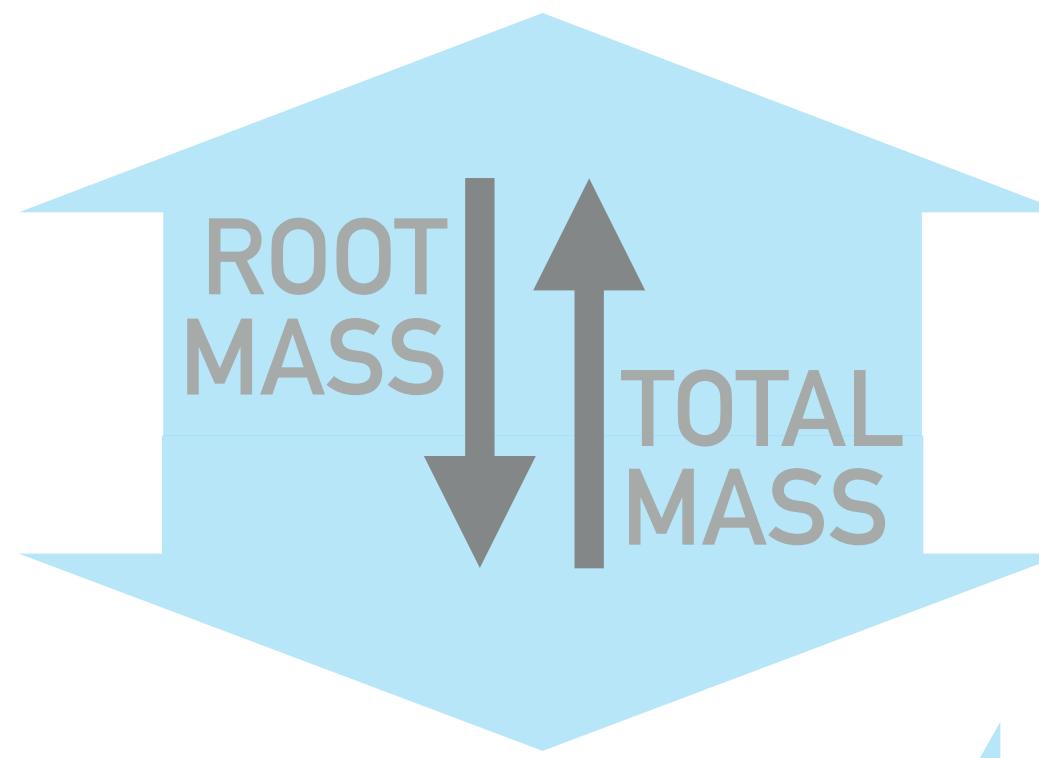
```
Out[4]: [
```



ROOT MODEL

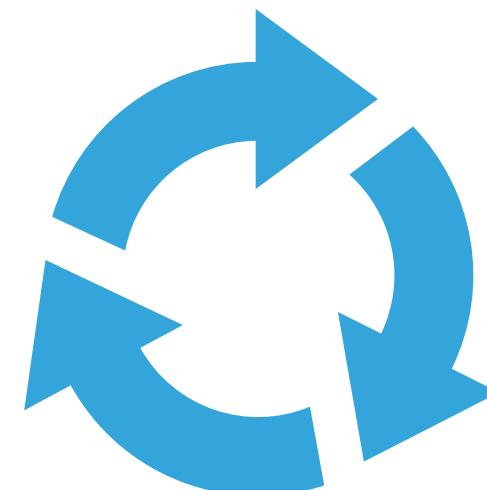


Shoot2root

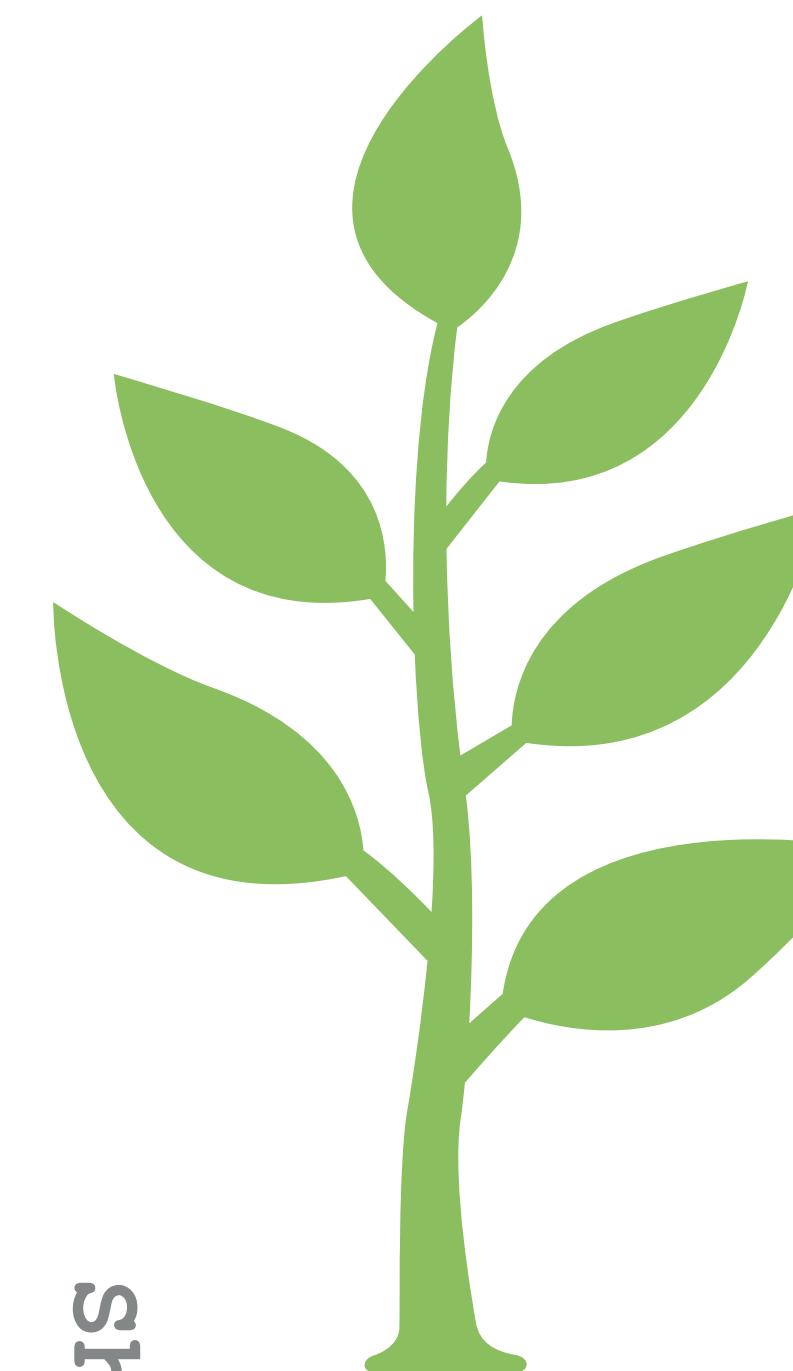


ROOT
MASS

TOTAL
MASS



TIMESYNC
“**MODEL**”



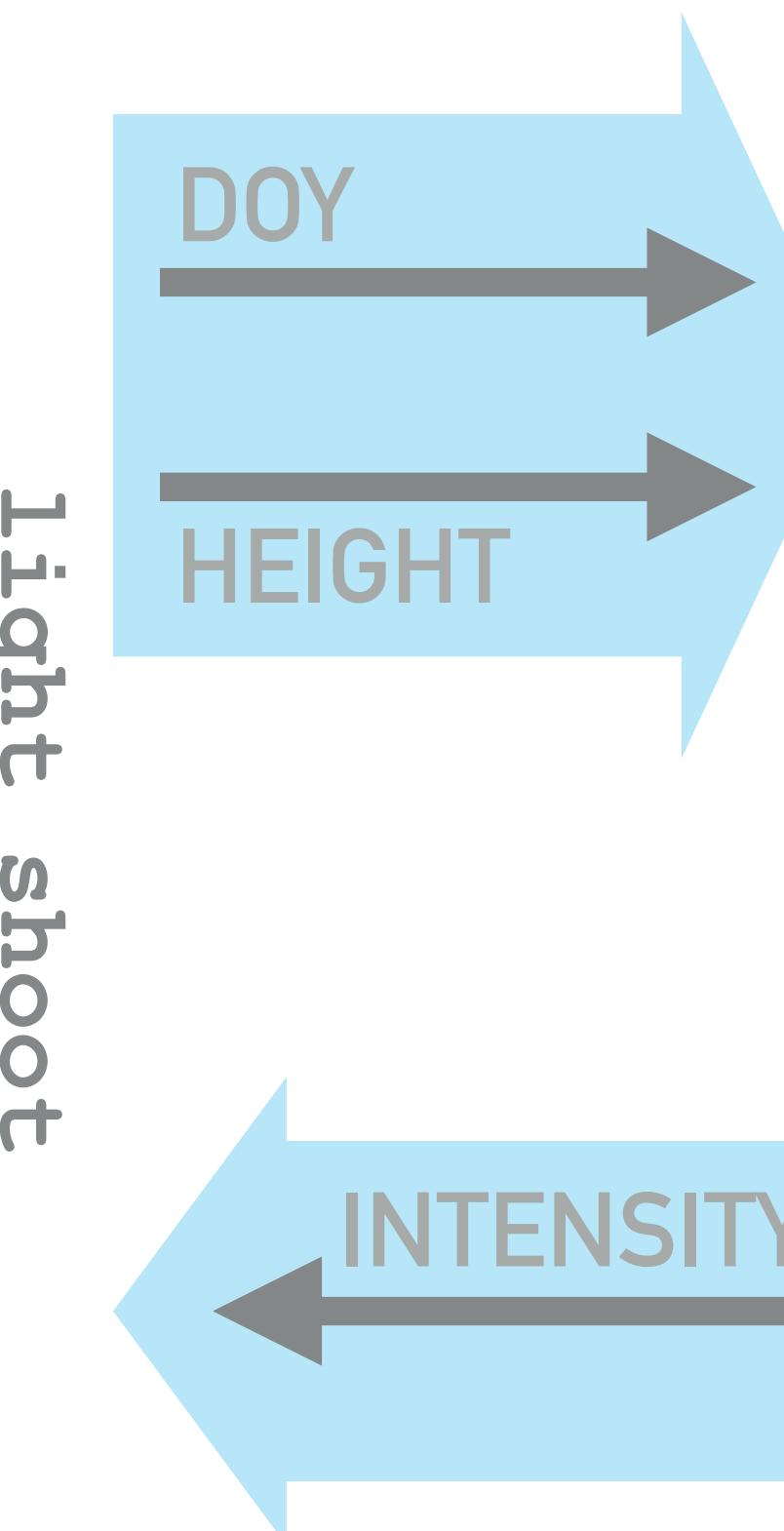
SHOOT
MODEL

Shoot2root

SHOOT MASS

TOTAL MASS

light_shoot



DOY

HEIGHT

INTENSITY

light:input

light:output

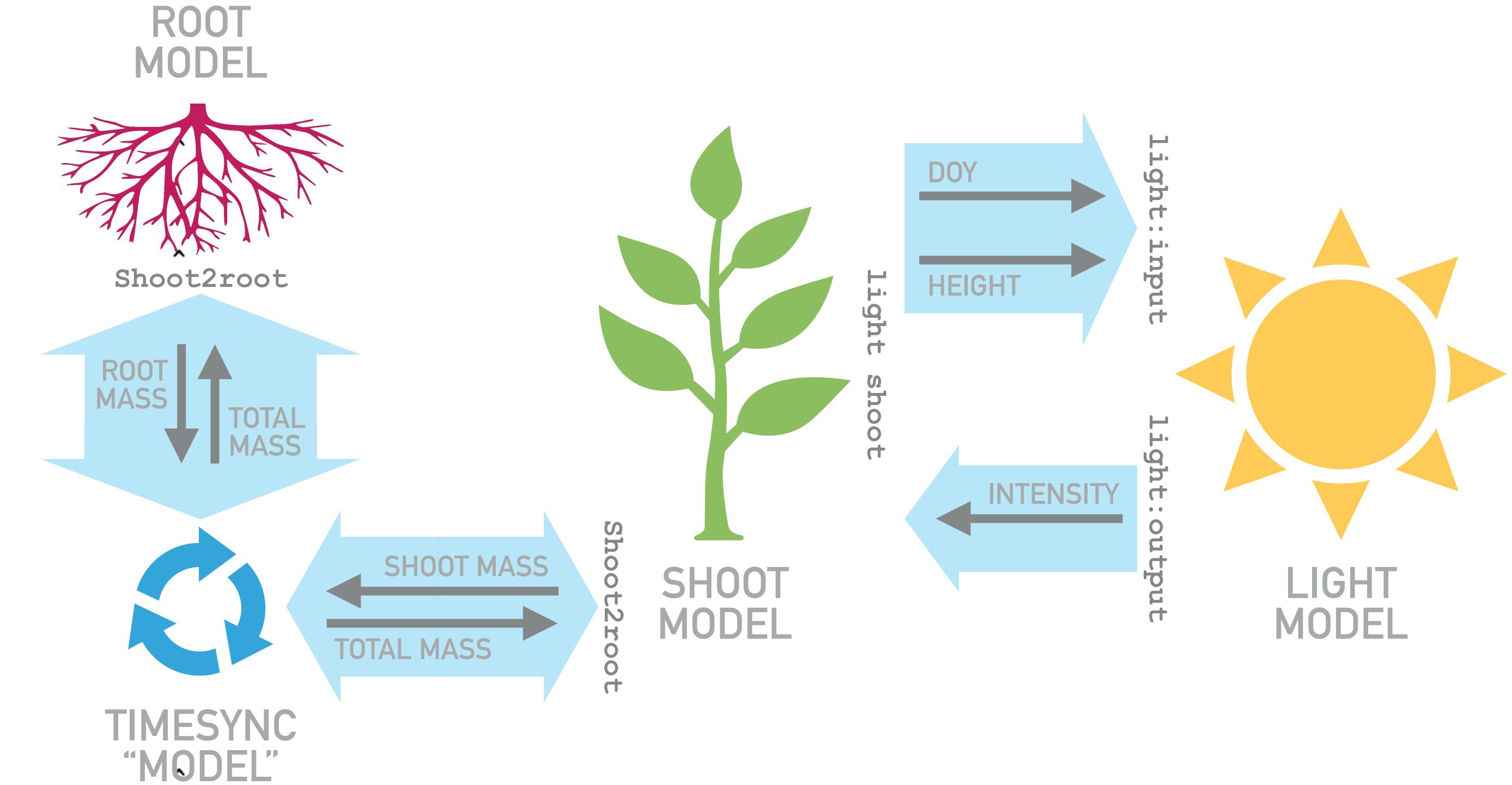


**LIGHT
MODEL**

```
In [5]: tools.display_source_diff('yamls/roots_v0.yml', 'yamls/roots_v1.yml', number_lines=True)
tools.display_source_diff('yamls/shoot_v2.yml', 'yamls/shoot_v3.yml', number_lines=True)
```

```
file1: yamls/roots_v0.yml
file2: yamls/roots_v1.yml
=====
1: model:
2:   name: roots
3:   language: python
-   args: [../models/roots_v0.py, 0.0, 2.0, 0.5]
?
4: + args: [../models/roots_v1.py, 0.0, 2.0, 0.5]
?
5: + timesync: shoot2root

file1: yamls/shoot_v2.yml
file2: yamls/shoot_v3.yml
=====
1: model:
2:   name: shoot
3:   language: python
-   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
?
4: + args: [../models/shoot_v3.py, 0.0, 48.0, 6.0]
?
5:   client_of: light
6: + timesync: shoot2root
```



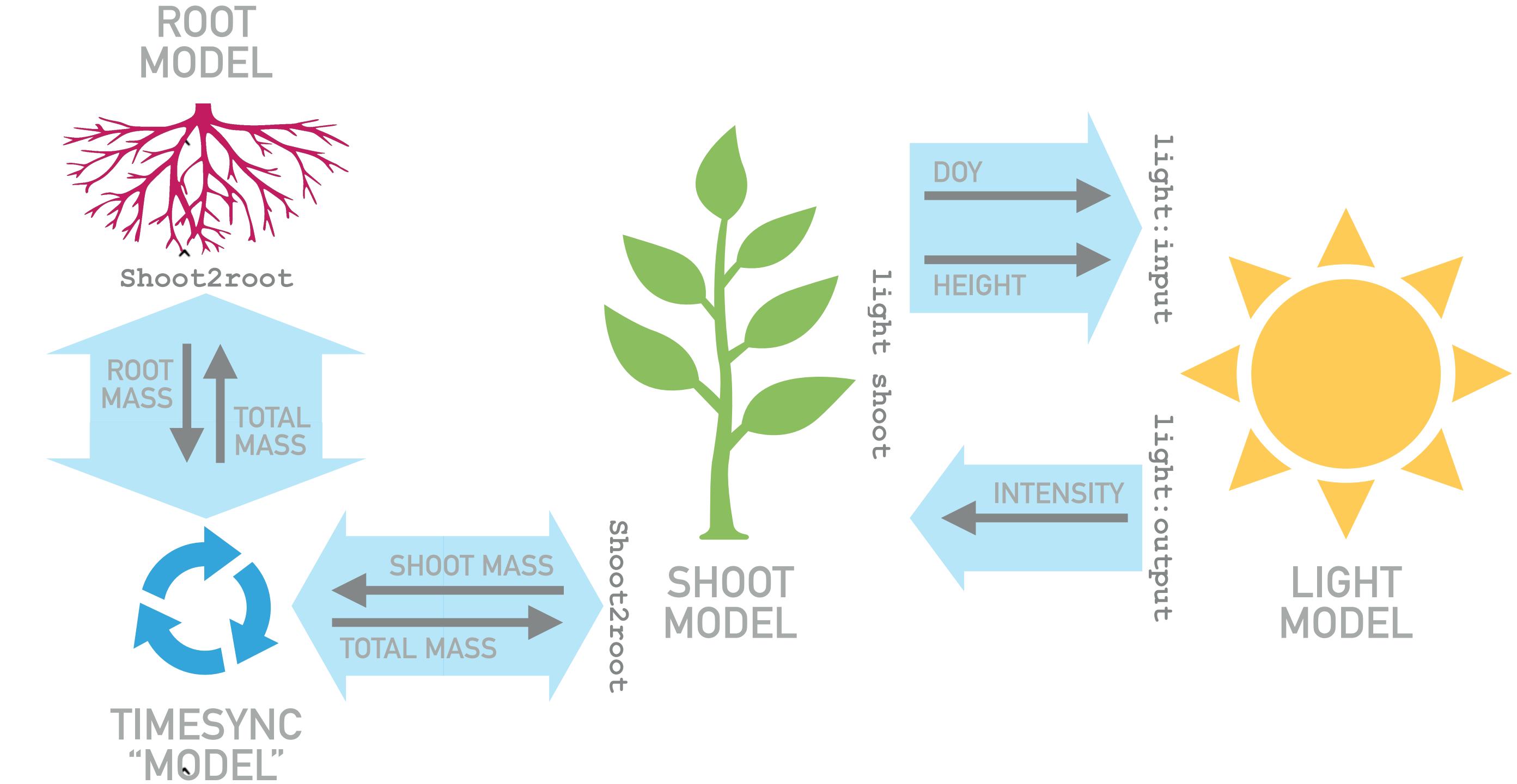
```
In [6]: tools.display_source('yamls/timesync.yml', number_lines=True)
```

```
file: yamls/timesync.yml
=====
1: model:
2:   name: shoot2root
3:   language: timesync
4:   aggregation: sum
```

```
In [5]: tools.display_source_diff('yamls/roots_v0.yml', 'yamls/roots_v1.yml', number_lines=True)
tools.display_source_diff('yamls/shoot_v2.yml', 'yamls/shoot_v3.yml', number_lines=True)
```

```
file1: yamls/roots_v0.yml
file2: yamls/roots_v1.yml
=====
1:   model:
2:     name: roots
3:     language: python
-   args: [../models/roots_v0.py, 0.0, 2.0, 0.5]
?
4: +   args: [../models/roots_v1.py, 0.0, 2.0, 0.5]
?
5: +   timesync: shoot2root

file1: yamls/shoot_v2.yml
file2: yamls/shoot_v3.yml
=====
1:   model:
2:     name: shoot
3:     language: python
-   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
?
4: +   args: [../models/shoot_v3.py, 0.0, 48.0, 6.0]
?
5:   client_of: light
6: +   timesync: shoot2root
```



```
In [6]: tools.display_source('yamls/timesync.yml', number_lines=True)
```

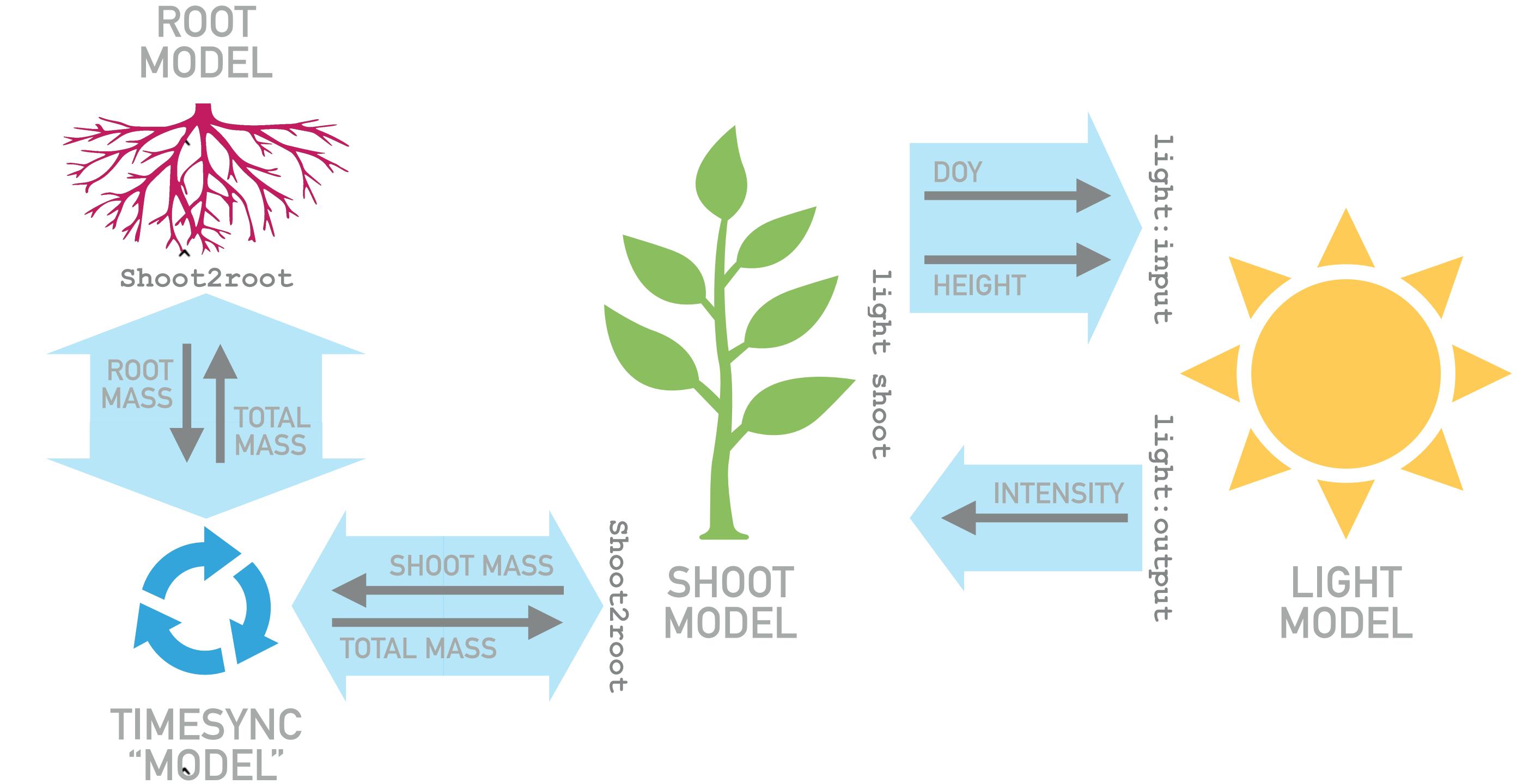
```
file: yamls/timesync.yml
=====
1: model:
2:   name: shoot2root
3:   language: timesync
4:   aggregation: sum
```

Match name of time sync
model "shoot2root"

```
In [5]: tools.display_source_diff('yaml/roots_v0.yml', 'yaml/roots_v1.yml', number_lines=True)
tools.display_source_diff('yaml/shoot_v2.yml', 'yaml/shoot_v3.yml', number_lines=True)
```

```
file1: yaml/roots_v0.yml
file2: yaml/roots_v1.yml
=====
1: model:
2:   name: roots
3:   language: python
-   args: [../models/roots_v0.py, 0.0, 2.0, 0.5]
?
4: + args: [../models/roots_v1.py, 0.0, 2.0, 0.5]
?
5: + timesync: shoot2root

file1: yaml/shoot_v2.yml
file2: yaml/shoot_v3.yml
=====
1: model:
2:   name: shoot
3:   language: python
-   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
?
4: + args: [../models/shoot_v3.py, 0.0, 48.0, 6.0]
?
5:   client_of: light
6: + timesync: shoot2root
```



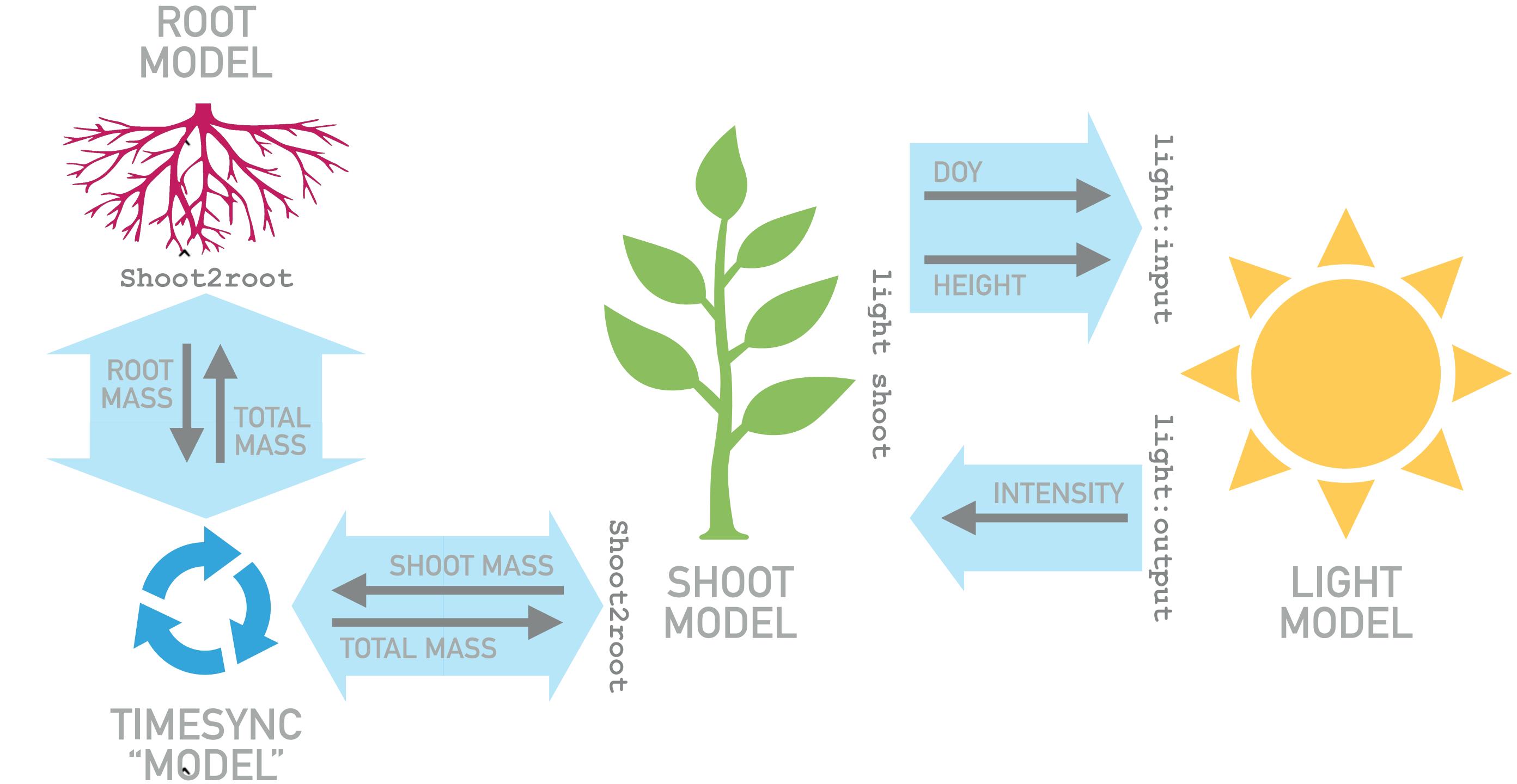
```
In [6]: tools.display_source('yaml/timesync.yml', number_lines=True)
```

```
file: yaml/timesync.yml
=====
1: model:
2:   name: shoot2root
3:   language: timesync
4:   aggregation: sum
```

```
In [5]: tools.display_source_diff('yamls/roots_v0.yml', 'yamls/roots_v1.yml', number_lines=True)
tools.display_source_diff('yamls/shoot_v2.yml', 'yamls/shoot_v3.yml', number_lines=True)
```

```
file1: yamls/roots_v0.yml
file2: yamls/roots_v1.yml
=====
1: model:
2:   name: roots
3:   language: python
-   args: [../models/roots_v0.py, 0.0, 2.0, 0.5]
?
4: + args: [../models/roots_v1.py, 0.0, 2.0, 0.5]
?
5: + timesync: shoot2root

file1: yamls/shoot_v2.yml
file2: yamls/shoot_v3.yml
=====
1: model:
2:   name: shoot
3:   language: python
-   args: [../models/shoot_v2.py, 0.0, 48.0, 6.0]
?
4: + args: [../models/shoot_v3.py, 0.0, 48.0, 6.0]
?
5:   client_of: light
6: + timesync: shoot2root
```



```
In [6]: tools.display_source('yamls/timesync.yml', number_lines=True)
```

```
file: yamls/timesync.yml
=====
1: model:
2:   name: shoot2root
3:   language: timesync
4:   aggregation: sum
```

“aggregation”
determines how values
should be combined

```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42:             flag = height_out.send(
43:                 [units.add_units(t, 'hrs'),
44:                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45:             if not flag:
46:                 raise Exception("Error sending height to output")
47:
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

Code to call yggdrasil inside "if blocks" so that model runs exactly the same without yggdrasil

```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42:             flag = height_out.send(
43:                 [units.add_units(t, 'hrs'),
44:                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45:             if not flag:
46:                 raise Exception("Error sending height to output")
47:
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

Import yggdrasil functions and connect to the channel that will be listed in the YAML.

Code to call yggdrasil inside “if blocks” so that model runs exactly the same without yggdrasil

```
In [15]: tools.display_source_diff('models/shoot_v0.py', 'models/shoot_v1.py', number_lines=True)
```

```
file1: models/shoot_v0.py
file2: models/shoot_v1.py
=====
1: import os
2: import trimesh
3: import argparse
4:
5: _dir = os.path.dirname(os.path.realpath(__file__))
6:
7: # Parse command-line arguments
8: parser = argparse.ArgumentParser("Simulate a shoot's growth over time.")
9: parser.add_argument('tmin', help='Starting time (in hours)', type=float)
10: parser.add_argument('tmax', help='Ending time (in hours)', type=float)
11: parser.add_argument('tstep', help='Time step (in hours)', type=float)
12: parser.add_argument('--meshfile', help='Path to file where mesh is stored.',
13:                     default='../meshes/plants-2.obj')
14: args = parser.parse_args()
15: tmin = args.tmin
16: tmax = args.tmax
17: tstep = args.tstep
18: mesh = trimesh.load_mesh(args.meshfile)
19:
20: # Set initial conditions
21: mass = 2000.0
22: t = tmin
23: i = 0
24:
25: + # Check if model is running as a part of an yggdrasil integration
26: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
27: +
28: + # If the model is running as part of an yggdrasil integration, import
29: + # the relevant yggdrasil routines and use the interface routine to
30: + # complete the connection defined in the YAML
31: + if with_yggdrasil:
32: +     from yggdrasil import units
33: +     from yggdrasil.languages.Python.YggInterface import YggOutput
34: +     height_out = YggOutput('height')
35: +
36:     # Continue simulation until time limit is reached
37:     while t <= tmax:
38:
39: +         # If running as part an yggdrasil integration, send the time and
40: +         # maximum height of the mesh to the height channel with units
41: +         if with_yggdrasil:
42: +             flag = height_out.send(
43: +                 [units.add_units(t, 'hrs'),
44: +                  units.add_units(max(mesh.vertices[:, 2]), 'm')])
45: +             if not flag:
46: +                 raise Exception("Error sending height to output")
47: +
48:         # Compute the scale factor
49:         # (pretend this is a biologically complex calculation)
50:         scale = mass / 4.5e4
51:
52:         # Grow the shoot
53:         # (pretend this is a biologically complex calculation)
54:         mesh.vertices[:, 2] += mesh.vertices[:, 2] * scale
55:         mass += mass * scale
56:
57:         # Save mesh for this timestep
58:         filename_mesh = os.path.join(_dir, f'../output/mesh_{i:03d}.obj')
59:         with open(filename_mesh, 'w') as fd:
60:             mesh.export(fd, 'obj')
61:
62:         # Advance time step
63:         t += tstep
64:         i += 1
```

Send height to output channel

Code to call yggdrasil inside "if blocks" so that model runs exactly the same without yggdrasil

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/roots_v0.py
file2: models/roots_v1.py
=====
23: + # Check if model is running as a part of an yggdrasil integration
24: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
25: +
26: + # If the model is running as part of an yggdrasil integration, import
27: + # the relevant yggdrasil routines and use the interface routine to
28: + # complete the connection defined in the YAML
29: + if with_yggdrasil:
30: +     from yggdrasil import units
31: +     from yggdrasil.languages.Python.YggInterface import YggTimesync
32: +     shoot2root = YggTimesync('shoot2root')
33: +
34:     # Continue simulation until time limit is reached
35:     while t <= tmax:
36:
37:         - # Compute the scale factor
38:         + # If running as part an yggdrasil integration, send the time and
39:         + # mass to the timesync channel and then updated the mass based on
40:         + # the returned state
41:         + if with_yggdrasil:
42:             root_state = {'mass': units.add_units(mass, 'kg')}
43:             flag, total_state = shoot2root.call(units.add_units(t, 'days'),
44:                                                 root_state)
45:             if not flag:
46:                 raise Exception("Error performing time-step synchronization "
47:                                 "with shoot model.")
48:         +     # Compute the scale factor using total mass, stripping units
49:         +     # of the result to allow use with original code
50:         -     # (pretend this is a biologically complex calculation)
51:         +     # (pretend this is a biologically complex calculation)
52:         ? +++
53:         +     scale = units.get_data(
54:             units.convert_to(
55:                 units.add_units(0.05, 'days-1') * total_state['mass'],
56:                 'kg/day'))
57:         +     else:
58:             # Compute the scale factor
59:             # (pretend this is a biologically complex calculation)
60:             -     scale = 0.2
61:             +     scale = 0.2
62:         ? +++
```

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/roots_v0.py
file2: models/roots_v1.py
=====
23: + # Check if model is running as a part of an yggdrasil integration
24: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
25: +
26: + # If the model is running as part of an yggdrasil integration, import
27: + # the relevant yggdrasil routines and use the interface routine to
28: + # complete the connection defined in the YAML
29: + if with_yggdrasil:
30: +     from yggdrasil import units
31: +     from yggdrasil.languages.Python.YggInterface import YggTimesync
32: +     shoot2root = YggTimesync('shoot2root')
33: +
34:     # Continue simulation until time limit is reached
35:     while t <= tmax:
36:
37:         - # Compute the scale factor
38:         + # If running as part an yggdrasil integration, send the time and
39:         + # mass to the timesync channel and then updated the mass based on
40:         + # the returned state
41:         + if with_yggdrasil:
42:             root_state = {'mass': units.add_units(mass, 'kg')}
43:             flag, total_state = shoot2root.call(units.add_units(t, 'days'),
44:                                                 root_state)
45:             if not flag:
46:                 raise Exception("Error performing time-step synchronization "
47:                                 "with shoot model.")
48:         +     # Compute the scale factor using total mass, stripping units
49:         +     # of the result to allow use with original code
50:         -     # (pretend this is a biologically complex calculation)
51:         +     # (pretend this is a biologically complex calculation)
52:         ? +++
53:         scale = units.get_data(
54:             units.convert_to(
55:                 units.add_units(0.05, 'days-1') * total_state['mass'],
56:                 'kg/day'))
57:         else:
58:             # Compute the scale factor
59:             # (pretend this is a biologically complex calculation)
60:             - scale = 0.2
61:             scale = 0.2
62:         ? +++
```

Import yggdrasil functions and connect to the time sync channel listed in the YAML.

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/roots_v0.py
file2: models/roots_v1.py
=====
23: + # Check if model is running as a part of an yggdrasil integration
24: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
25: +
26: + # If the model is running as part of an yggdrasil integration, import
27: + # the relevant yggdrasil routines and use the interface routine to
28: + # complete the connection defined in the YAML
29: + if with_yggdrasil:
30: +     from yggdrasil import units
31: +     from yggdrasil.languages.Python.YggInterface import YggTimesync
32: +     shoot2root = YggTimesync('shoot2root')
33: +
34:     # Continue simulation until time limit is reached
35:     while t <= tmax:
36:
37:         - # Compute the scale factor
38:         + # If running as part an yggdrasil integration, send the time and
39:         + # mass to the timesync channel and then updated the mass based on
40:         + # the returned state
41:         + if with_yggdrasil:
42:             root_state = {'mass': units.add_units(mass, 'kg')}
43:             flag, total_state = shoot2root.call(units.add_units(t, 'days'),
44:                                                 root_state)
45:             if not flag:
46:                 raise Exception("Error performing time-step synchronization "
47:                                 "with shoot model.")
48:         + # Compute the scale factor using total mass, stripping units
49:         + # of the result to allow use with original code
50:         - # (pretend this is a biologically complex calculation)
51:         + # (pretend this is a biologically complex calculation)
52:         ? +++
53:         scale = units.get_data(
54:             units.convert_to(
55:                 units.add_units(0.05, 'days-1') * total_state['mass'],
56:                 'kg/day'))
57:         else:
58:             # Compute the scale factor
59:             - # (pretend this is a biologically complex calculation)
60:             scale = 0.2
61:             scale = 0.2
62:         ? +++
```

Send root state to the time sync model
and receive the total state back

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/roots_v0.py
file2: models/roots_v1.py
=====
23: + # Check if model is running as a part of an yggdrasil integration
24: + with_yggdrasil = os.environ.get('YGG_SUBPROCESS', False)
25: +
26: + # If the model is running as part of an yggdrasil integration, import
27: + # the relevant yggdrasil routines and use the interface routine to
28: + # complete the connection defined in the YAML
29: + if with_yggdrasil:
30: +     from yggdrasil import units
31: +     from yggdrasil.languages.Python.YggInterface import YggTimesync
32: +     shoot2root = YggTimesync('shoot2root')
33: +
34:     # Continue simulation until time limit is reached
35:     while t <= tmax:
36:
37:         - # Compute the scale factor
38:         - # If running as part an yggdrasil integration, send the time and
39:         - # mass to the timesync channel and then updated the mass based on
40:         - # the returned state
41:         + if with_yggdrasil:
42:             root_state = {'mass': units.add_units(mass, 'kg')}
43:             flag, total_state = shoot2root.call(units.add_units(t, 'days'),
44:                                                 root_state)
45:             if not flag:
46:                 raise Exception("Error performing time-step synchronization "
47:                                 "with shoot model.")
48:         +
49:             # Compute the scale factor using total mass, stripping units
50:             # of the result to allow use with original code
51:             - # (pretend this is a biologically complex calculation)
52:             + # (pretend this is a biologically complex calculation)
53:             ? +++
54:             scale = units.get_data(
55:                 units.convert_to(
56:                     units.add_units(0.05, 'days-1') * total_state['mass'],
57:                     'kg/day'))
58:             else:
59:                 # Compute the scale factor
60:                 # (pretend this is a biologically complex calculation)
61:                 - scale = 0.2
62:                 scale = 0.2
63:             ? +++
```

Compute the scale using the total state
when run with yggdrasil

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
      tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v3.py
=====
...
31: if with_yggdrasil:
32:     from yggdrasil import units
33: -     from yggdrasil.languages.Python.YggInterface import YggRpcClient
33: +     from yggdrasil.languages.Python.YggInterface import YggRpcClient, YggTimesync
34:     ?
+++++
34:         light_rpc = YggRpcClient('light_shoot')
35: +     shoot2root = YggTimesync('shoot2root')
36:
37: # Continue simulation until time limit is reached
38: while t <= tmax:
39:
40:     # If running as part an yggdrasil integration, send the time and
41: -     # maximum height of the mesh to the height channel with units
41: +     # maximum height of the mesh to the height channel with units and
42:     ?
        +++++
42: +     # send the current mass and tiem to the timesync channel
43: if with_yggdrasil:
44: +     shoot_state = {'mass': units.add_units(mass, 'g')}
45: +     flag, total_state = shoot2root.call(units.add_units(t, 'hrs'),
46:                                         shoot_state)
47: +     if not flag:
48: +         raise Exception("Error performing time-step synchronization "
49:                         "with root model.")
50: +
...
60:     scale = units.get_data(
61: -         units.add_units(mass, 'g') * intensity /
62: ?             ^^^  ^ -----
63:
64:             total_state['mass'] * intensity /
65: ?             ^^^^ +++++ ^^^^
66:
67: -             units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
68: +             units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
69: ? ++++
```

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
      tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v3.py
=====
...
31: if with_yggdrasil:
32:     from yggdrasil import units
33: -     from yggdrasil.languages.Python.YggInterface import YggRpcClient
33: +     from yggdrasil.languages.Python.YggInterface import YggRpcClient, YggTimesync
34:     ?
35: ++++++
36:
37:         light_rpc = YggRpcClient('light_shoot')
38: +     shoot2root = YggTimesync('shoot2root')
39:
40:     # Continue simulation until time limit is reached
41:     while t <= tmax:
42:     #
43:         # If running as part an yggdrasil integration, send the time and
44:         # maximum height of the mesh to the height channel with units
45:         # maximum height of the mesh to the height channel with units and
46:         ?
47:         ++++++
48:         # send the current mass and tiem to the timesync channel
49:         if with_yggdrasil:
50:             shoot_state = {'mass': units.add_units(mass, 'g')}
51:             flag, total_state = shoot2root.call(units.add_units(t, 'hrs'),
52:                                                 shoot_state)
53:             if not flag:
54:                 raise Exception("Error performing time-step synchronization "
55:                                 "with root model.")
56:
57:
58:             scale = units.get_data(
59:                 units.add_units(mass, 'g') * intensity /
60:                 ^^^^ ^ -----
61:             total_state['mass'] * intensity /
62:                 ^^^^ +++++ ^^^^
63:                 +-----+
64:                 units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
65:                 units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
66:                 ? +++++
```

Import the time sync interface &
connect to the “shoot2root”
channel

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v3.py
=====
...
31:     if with_yggdrasil:
32:         from yggdrasil import units
33:         - from yggdrasil.languages.Python.YggInterface import YggRpcClient
34:         + from yggdrasil.languages.Python.YggInterface import YggRpcClient, YggTimesync
35:         ?
36: =====+
37:         light_rpc = YggRpcClient('light_shoot')
38:         + shoot2root = YggTimesync('shoot2root')
39: 
40:         # Continue simulation until time limit is reached
41:         while t <= tmax:
42:             # If running as part an yggdrasil integration, send the time and
43:             # maximum height of the mesh to the height channel with units
44:             # maximum height of the mesh to the height channel with units and
45:             ?                               +++
46:             # send the current mass and tiem to the timesync channel
47:             if with_yggdrasil:
48:                 shoot_state = {'mass': units.add_units(mass, 'g')}
49:                 flag, total_state = shoot2root.call(units.add_units(t, 'hrs'),
50:                                                       shoot_state)
51:                 if not flag:
52:                     raise Exception("Error performing time-step synchronization "
53:                                     "with root model.")
54: 
55:             ...
56:             scale = units.get_data(
57:                 - units.add_units(mass, 'g') * intensity /
58:                 ?                               ^-----^
59: 
60:                 + total_state['mass'] * intensity /
61:                 ?                               ^+++
62: 
63:                 - units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
64:                 + units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
65: 
66:             ? ++++
```

Send the shoot state to the time sync
model & receive back the total state

```
In [7]: tools.display_source_diff('models/roots_v0.py', 'models/roots_v1.py', number_lines=True)
tools.display_source_diff('models/shoot_v2.py', 'models/shoot_v3.py', number_lines=True)
```

```
file1: models/shoot_v2.py
file2: models/shoot_v3.py
=====
...
31:     if with_yggdrasil:
32:         from yggdrasil import units
33:         - from yggdrasil.languages.Python.YggInterface import YggRpcClient
34:         + from yggdrasil.languages.Python.YggInterface import YggRpcClient, YggTimesync
35:         ?
36: =====+
37:         light_rpc = YggRpcClient('light_shoot')
38:         + shoot2root = YggTimesync('shoot2root')
39: 
40:         # Continue simulation until time limit is reached
41:         while t <= tmax:
42:             # If running as part an yggdrasil integration, send the time and
43:             # maximum height of the mesh to the height channel with units
44:             # maximum height of the mesh to the height channel with units and
45:             ?                               +++
46:             # send the current mass and tiem to the timesync channel
47:             if with_yggdrasil:
48:                 shoot_state = {'mass': units.add_units(mass, 'g')}
49:                 flag, total_state = shoot2root.call(units.add_units(t, 'hrs'),
50:                                                       shoot_state)
51: 
52:             if not flag:
53:                 raise Exception("Error performing time-step synchronization "
54:                                 "with root model.")
55: 
56:             ...
57:             scale = units.get_data(
58:                 - units.add_units(mass, 'g') * intensity /
59:                 ?                               ^-----^
60: 
61:                 + total_state['mass'] * intensity /
62:                 ?                               ++++++ ^^^^
63: 
64:                 - units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
65:                 + units.add_units(4.0e10, 'g*erg/(cm**2*s)'))
66:                 ? ++++
```

Change the scale calculation to use
the total state

```
In [8]: run(['yamls/shoot_v3.yml', 'yamls/roots_v1.yml', 'yamls/light_v1_python.yml', 'yamls/timesync.yml'], production_run=True)
```

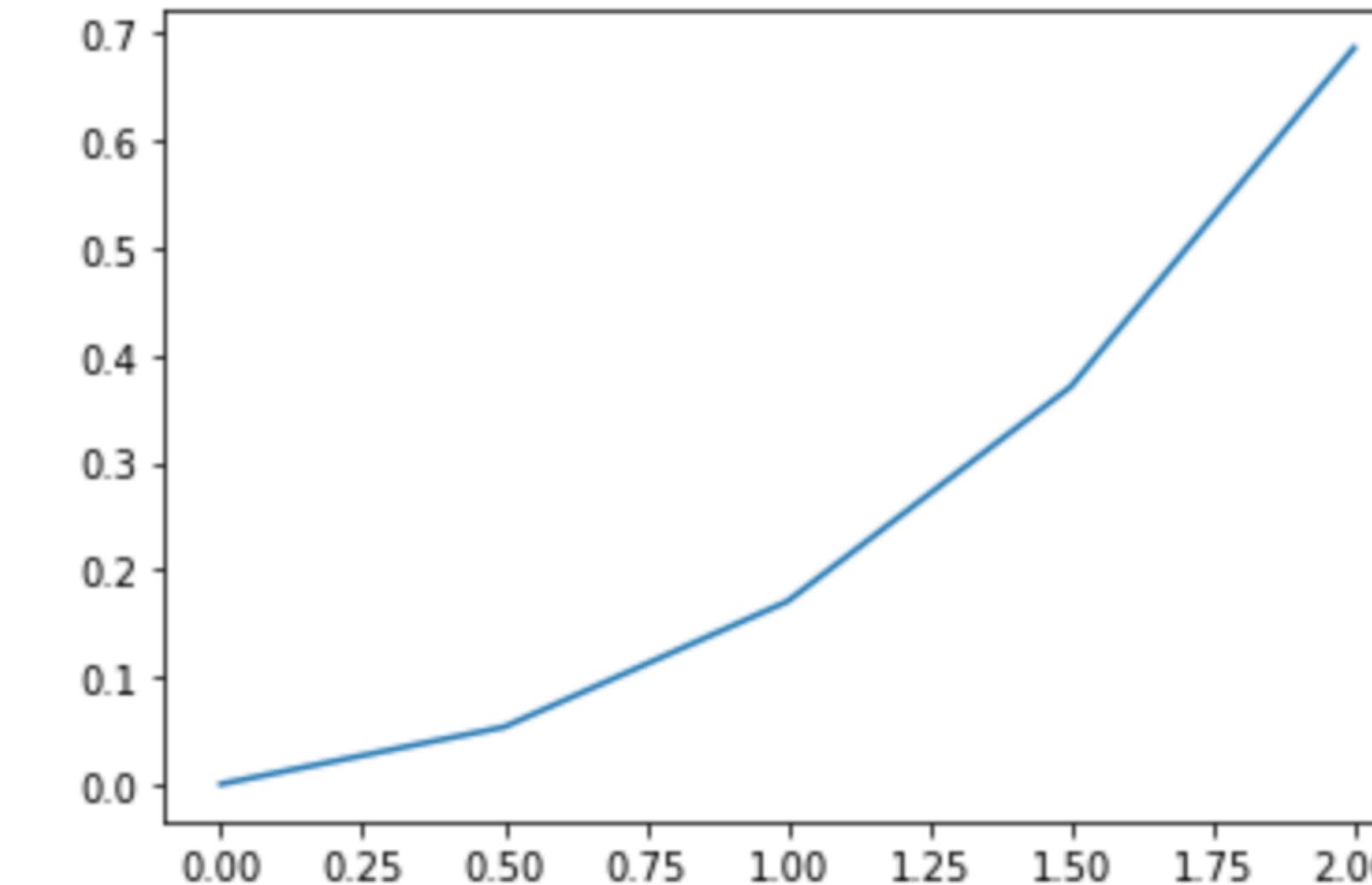
```
INFO:96257:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/ygg  
_light_v0.py  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/sho  
ot_v3.py 0.0 48.0 6.0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/roo  
ts_v1.py 0.0 2.0 0.5  
End of input from temp_doy.  
INFO:96257:runner.waitModels[553]:YggRunner(runner): shoot finished running.  
No more messages from model process.  
INFO:96257:DSLModelDriver.after_loop[131]:TimeSyncModelDriver(shoot2root): returncode = 0  
INFO:96257:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.  
INFO:96257:runner.waitModels[553]:YggRunner(runner): light finished running.  
INFO:96257:runner.waitModels[559]:YggRunner(runner): light finished exiting.  
INFO:96257:runner.waitModels[553]:YggRunner(runner): roots finished running.  
INFO:96257:runner.waitModels[559]:YggRunner(runner): roots finished exiting.  
INFO:96257:runner.waitModels[553]:YggRunner(runner): shoot2root finished running.  
INFO:96257:runner.waitModels[559]:YggRunner(runner): shoot2root finished exiting.  
INFO:96257:runner.waitModels[573]:YggRunner(runner): All models completed  
INFO:96257:runner.run[374]:YggRunner(runner): init 0.000001  
INFO:96257:runner.run[374]:YggRunner(runner): load drivers 0.051068  
INFO:96257:runner.run[374]:YggRunner(runner): start drivers 0.303378  
INFO:96257:runner.run[374]:YggRunner(runner): run models 19.234870  
INFO:96257:runner.run[374]:YggRunner(runner): at exit 0.117888  
INFO:96257:runner.run[376]:YggRunner(runner): =====  
INFO:96257:runner.run[377]:YggRunner(runner): Total 19.707205
```

```
In [8]: run(['yamls/shoot_v3.yml', 'yamls/roots_v1.yml', 'yamls/light_v1_python.yml', 'yamls/timesync.yml'], production_run=True)
```

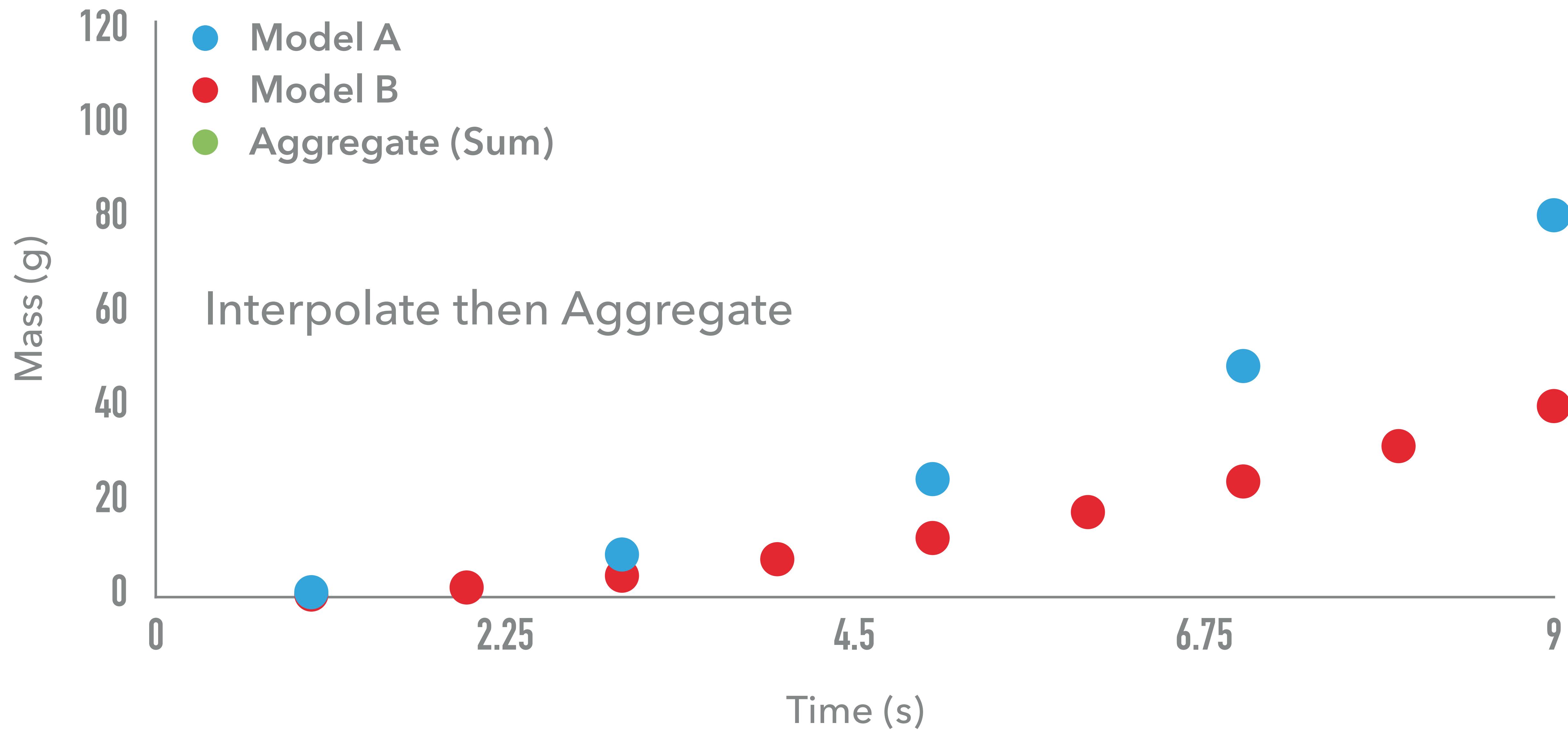
```
INFO:96257:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system Meagans-MacBook-Air.  
local in namespace yggdrasil with rank 0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/ygg  
_light_v0.py  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/sho  
ot_v3.py 0.0 48.0 6.0  
/Users/langmm/miniconda3/envs/conda36/bin/python /Users/langmm/yggdrasil/yggdrasil/demos/Cis2021-hackathon/models/roo  
ts_v1.py 0.0 2.0 0.5  
End of input from temp_doy.  
INFO:96257:runner.waitModels[553]:YggRunner(runner): sho  
No more messages from model process.  
INFO:96257:DSLModelDriver.after_loop[131]:TimeSyncModelD  
INFO:96257:runner.waitModels[559]:YggRunner(runner): sho  
INFO:96257:runner.waitModels[553]:YggRunner(runner): lig  
INFO:96257:runner.waitModels[559]:YggRunner(runner): lig  
INFO:96257:runner.waitModels[553]:YggRunner(runner): roo  
INFO:96257:runner.waitModels[559]:YggRunner(runner): roo  
INFO:96257:runner.waitModels[553]:YggRunner(runner): sho  
INFO:96257:runner.waitModels[559]:YggRunner(runner): sho  
INFO:96257:runner.waitModels[573]:YggRunner(runner): All  
INFO:96257:runner.run[374]:YggRunner(runner):  
INFO:96257:runner.run[374]:YggRunner(runner):          lo  
INFO:96257:runner.run[374]:YggRunner(runner):          sta  
INFO:96257:runner.run[374]:YggRunner(runner):  
INFO:96257:runner.run[374]:YggRunner(runner):  
INFO:96257:runner.run[376]:YggRunner(runner): ======  
INFO:96257:runner.run[377]:YggRunner(runner):
```

```
In [9]: import matplotlib.pyplot as plt  
filename_masses = 'output/masses.pkl'  
with open(filename_masses, 'rb') as fd:  
    masses = pickle.load(fd)  
plt.plot(masses['times'], masses['masses'])
```

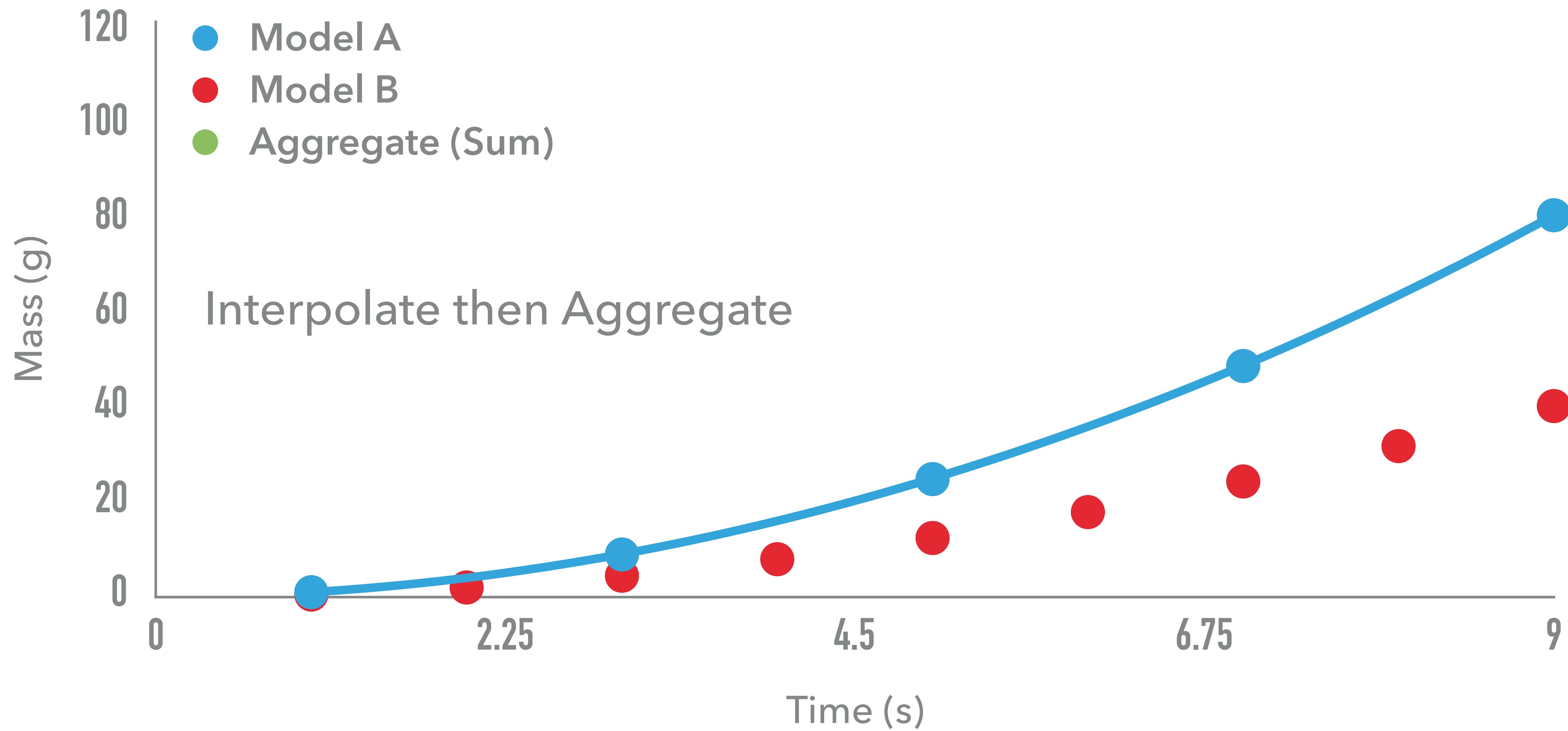
```
Out[9]: <matplotlib.lines.Line2D at 0x147d40048>
```



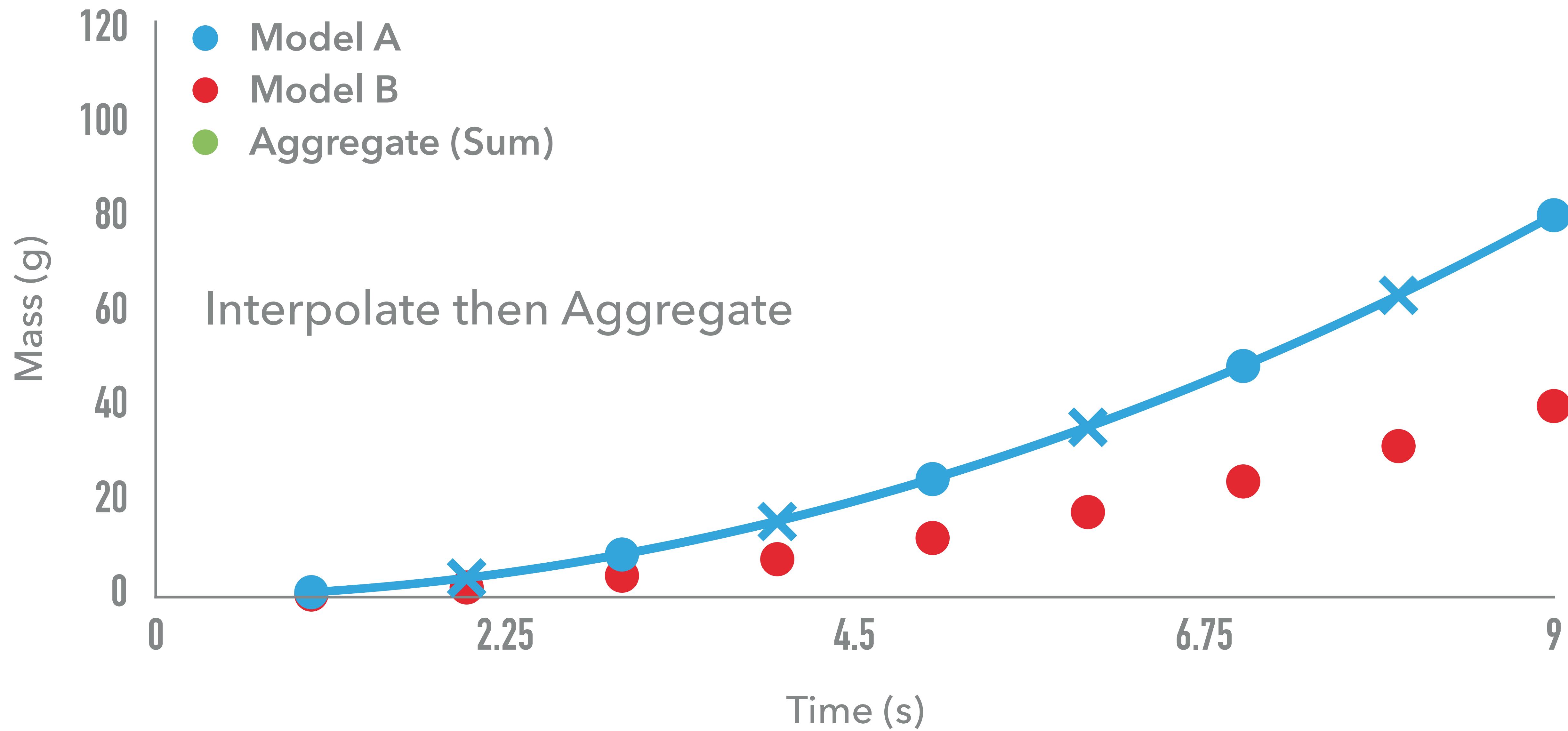
AGGREGATION OF TIME STEPS



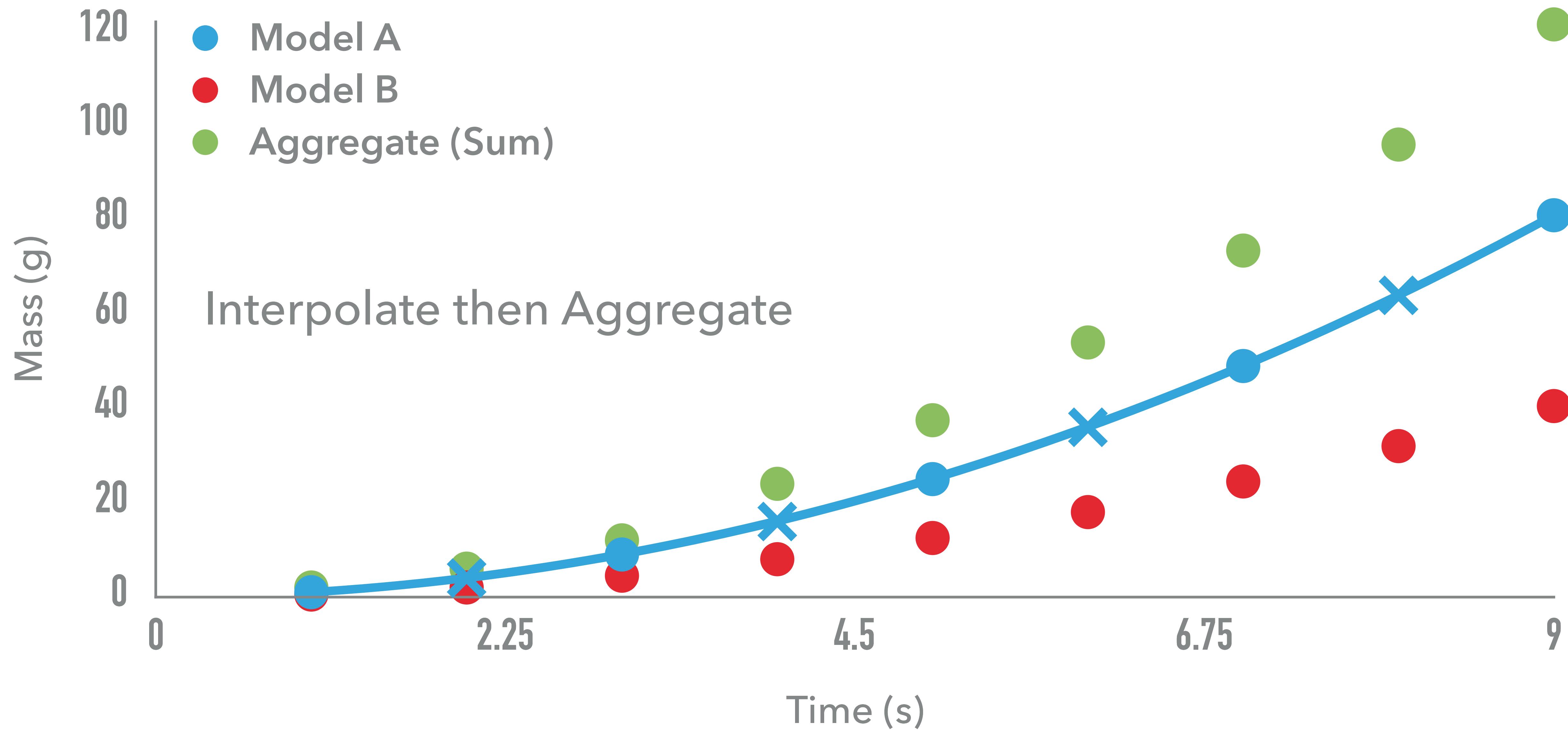
AGGREGATION OF TIME STEPS



AGGREGATION OF TIME STEPS



AGGREGATION OF TIME STEPS



NEW NOTEBOOK!
(BREAK TIME)

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#) [⟳](#)

<input type="checkbox"/>	0	▼	 /	Name 	Last Modified	File size
<input type="checkbox"/>	 images				33 minutes ago	
<input type="checkbox"/>	 input				33 minutes ago	
<input type="checkbox"/>	 meshes				33 minutes ago	
<input type="checkbox"/>	 models				33 minutes ago	
<input type="checkbox"/>	 yaml				33 minutes ago	
<input type="checkbox"/>	 00-intro.ipynb				33 minutes ago	457 kB
<input type="checkbox"/>	 01-connections.ipynb				33 minutes ago	470 kB
<input type="checkbox"/>	 02-timesync.ipynb				33 minutes ago	298 kB
<input type="checkbox"/>	 03-misc.ipynb				33 minutes ago	3.56 kB

[Files](#) [Running](#) [Clusters](#) [Nbextensions](#)

Select items to perform actions on them.

[Upload](#) [New ▾](#)

<input type="checkbox"/> 0	/	Name	Last Modified	File size
<input type="checkbox"/>	images		33 minutes ago	
<input type="checkbox"/>	input		33 minutes ago	
<input type="checkbox"/>	meshes		33 minutes ago	
<input type="checkbox"/>	models		33 minutes ago	
<input type="checkbox"/>	yaml s		33 minutes ago	
<input type="checkbox"/>	00-intro.ipynb		33 minutes ago	457 kB
<input type="checkbox"/>	01-connections.ipynb		33 minutes ago	470 kB
<input type="checkbox"/>	02-timesync.ipynb		33 minutes ago	298 kB
<input type="checkbox"/>	03-misc.ipynb		33 minutes ago	3.56 kB

IMPORTING MODELS AS PYTHON FUNCTIONS

```
In [1]: from yggdrasil import tools
tools.display_source('models/light_v0.f90', number_lines=True)
tools.display_source('yamls/light_v0_fortran.yml', number_lines=True)

file: models/light_v0.f90
=====
1: !-----
2: !> @brief Compute the intensity of light.
3: !
4: !> @param[in] doy: Day of year.
5: !> @param[in] height: Distance from ground in cm.
6: !
7: !> @return intensity: Intensity of light in ergs cm^-2 s^-1.
8: !-----
9: function light(doy, height) result(intensity)
10:    real(kind=8) :: doy
11:    real(kind=8) :: height
12:    real(kind=8) :: intensity
13:    real, parameter :: Pi = 3.1415927
14:
15:    ! Define parameters that are static across a run
16:    real, parameter :: amplitude = 80.0
17:    real, parameter :: doy_offset = 0.0
18:
19:    ! Calculate intensity
20:    intensity = amplitude * height * (1.0 + SIN(2.0 * Pi * (doy - doy_offset) / 365))
21: end function light

file: yamls/light_v0_fortran.yml
=====
1: model:
2:   name: light
3:   language: fortran
4:   args: ../models/light_v0.f90
5:   function: light
6:   inputs:
7:     - name: input
8:       vars: [doy, height]
9:       datatype:
10:         type: array
11:         items:
12:           - type: float
13:             units: day
14:           - type: float
15:             units: cm
16:   output:
17:     - name: output
18:       datatype:
19:         type: float
20:         units: ergs/(cm**2*s)
```

```
In [2]: from yggdrasil import import_as_function
```

```
light = import_as_function('yamls/light_v0_fortran.yml')
```

```
INFO:97475:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system None in namespace yg  
gdrasil with rank 0
```

```
/Users/langmm/yggdrasil/yggdrasil/demos/CiS2021-hackathon/models/ygg_light_v0_f90_gfortranx_gfortranx.out
```

```
INFO:97475:runner.run[374]:YggRunner(runner):           init      0.000001
```

```
INFO:97475:runner.run[374]:YggRunner(runner):       load drivers    0.828631
```

```
INFO:97475:runner.run[374]:YggRunner(runner):      start drivers   0.061332
```

```
INFO:97475:runner.run[376]:YggRunner(runner): =====
```

```
INFO:97475:runner.run[377]:YggRunner(runner):           Total      0.889964
```

```
In [3]: light.model_info()
```

```
Models: light
```

```
Inputs:
```

```
    light:input_to_light:input (vars=['doy', 'height'])
```

```
Outputs:
```

```
    light:output (vars=['intensity'])
```

Model can be called from Python regardless of the language

```
In [4]: import numpy as np  
print(light(100.0, 100.0))  
print(light(1.0, 2.9))  
print(light(2.0, 3.0))  
  
{'intensity': unyt_quantity(15909.42066435, 'erg/(cm**2*s)')}  
{'intensity': unyt_quantity(235.99349874, 'erg/(cm**2*s)')}  
{'intensity': unyt_quantity(248.26118702, 'erg/(cm**2*s)'})
```

Model units are assumed if not provided

```
In [4]: import numpy as np  
print(light(100.0, 100.0))  
print(light(1.0, 2.9))  
print(light(2.0, 3.0))  
  
{'intensity': unyt_quantity(15909.42066435, 'erg/(cm**2*s)')}  
{'intensity': unyt_quantity(235.99349874, 'erg/(cm**2*s)')}  
{'intensity': unyt_quantity(248.26118702, 'erg/(cm**2*s)'})
```

Model units are assumed if not provided

```
In [5]: from yggdrasil import units  
print(light(units.add_units(24.0, 'hrs'), units.add_units(2.9, 'cm')))  
print(light(units.add_units(1.0, 'days'), units.add_units(0.029, 'm')))   
  
{'intensity': unyt_quantity(235.99349874, 'erg/(cm**2*s)')}  
{'intensity': unyt_quantity(235.99349874, 'erg/(cm**2*s)'})
```

If units are provided, yggdrasil can handle transformations to/from the model's units

COMMAND LINE INTERFACE (CLI)

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

The screenshot shows a Jupyter Notebook interface with the following elements:

- Header:** "jupyter" logo, "Visit repo", "Copy Binder link", and "Quit" buttons.
- Navigation:** "Files", "Running", "Clusters", and "Nbextensions" tabs. The "Running" tab is selected.
- File Browser:** A table listing files and folders in the current directory. The table includes columns for selection, name, last modified, and file size.
- Table Data:**

	Name	Last Modified	File size
<input type="checkbox"/>	0		
<input type="checkbox"/>	meshes	6 hours ago	
<input type="checkbox"/>	models	6 hours ago	
<input type="checkbox"/>	output	6 hours ago	
<input type="checkbox"/>	yamls	6 hours ago	
<input type="checkbox"/>	yggdrasil	6 hours ago	
<input type="checkbox"/>	plant.ipynb	6 hours ago	17.4 kB
<input type="checkbox"/>	environment.yml	6 hours ago	168 B
<input type="checkbox"/>	LICENSE	6 hours ago	1.52 kB
<input type="checkbox"/>	postBuild	6 hours ago	202 B
<input type="checkbox"/>	README.md	6 hours ago	486 B
<input type="checkbox"/>	utils.py	6 hours ago	4.3 kB

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

The screenshot shows the Jupyter Notebook interface. At the top, there is a navigation bar with tabs for "Files", "Running", "Clusters", and "Nbextensions". On the far right of the bar are "Quit" and "Logout" buttons. Below the navigation bar, there is a message "Select items to perform actions on them." To the left, there is a file browser showing a directory structure with files like "00-intro.ipynb", "01-connections.ipynb", and "02-timesync.ipynb". To the right of the file browser, there is a context menu with options for "Upload", "New", and "Copy". A dropdown menu is open under "New", showing options for "Notebook: Python 3", "Text File", "Folder", and "Terminal".

Name	Type	Last Modified	Size
0	Folder	5 days ago	
images	Folder	5 days ago	
input	Folder	5 days ago	
meshes	Folder	5 days ago	
models	Folder	5 days ago	
output	Folder	5 days ago	
yamls	Folder	5 days ago	
00-intro.ipynb	Notebook	3 hours ago	457 kB
01-connections.ipynb	Notebook	2 hours ago	470 kB
02-timesync.ipynb	Notebook	5 days ago	298 kB
03-misc.ipynb	Notebook	5 days ago	3.56 kB
environment.yml	File	9 days ago	169 B
LICENSE	File	9 days ago	1.52 kB
postBuild	File	2 hours ago	290 B

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

The screenshot shows the Jupyter Notebook interface. At the top, there is a navigation bar with tabs: 'Files' (selected), 'Running', 'Clusters', and 'Nbextensions'. On the right side of the header, there are 'Quit' and 'Logout' buttons. Below the header, a sidebar displays a file tree with the following structure:

- 0
- /
- images
- input
- meshes
- models
- output
- yamls
- 00-intro.ipynb
- 01-connections.ipynb
- 02-timesync.ipynb
- 03-misc.ipynb
- environment.yml
- LICENSE
- postBuild

On the right, there is a 'New' button with a dropdown menu. The menu includes options: 'Notebook:' (with 'Python 3' selected), 'Other:' (with 'Text File' and 'Folder' listed), and 'Terminal' (which is circled in red). The terminal entry is located at the bottom of the 'Other' section.

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python



jupyter

[Visit repo](#)

[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$
```

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python



jupyter

[Visit repo](#)

[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$ yggrun -h
```

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

[Visit repo](#)[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$ yggrun -h
usage: Run an integration. [-h] [--loglevel LOGLEVEL] [--rmq-loglevel RMQ_LOGLEVEL] [--client-loglevel CLIENT_LOGLEVEL]
                           [--validate-components] [--validate-messages {False,True,First}] [--namespace NAMESPACE]
                           [--host HOST] [--vhost VHOST] [--user USER] [--password PASSWORD] [--cluster CLUSTER]
                           [--default-comm DEFAULT_COMM] [--production-run] [--debug]
                           yamlfile [yamlfile ...]

positional arguments:
  yamlfile            One or more yaml specification files.

optional arguments:
  -h, --help          show this help message and exit
  --loglevel LOGLEVEL    Logging level for yggdrasil operations.
  --rmq-loglevel RMQ_LOGLEVEL, --rmqloglevel RMQ_LOGLEVEL
                        Logging level for RabbitMQ operations.
  --client-loglevel CLIENT_LOGLEVEL, --clientloglevel CLIENT_LOGLEVEL
                        Logging level for yggdrasil operations on model processes.
  --validate-components, --validatecomponents
                        Validate components on creation using their JSON schema (Decreases performance).
  --validate-messages {False,True,First}, --validatemessages {False,True,First}
                        Which messages should be validated during communication. 'True': all messages (decreases
                        performance), 'False': no messages, or 'First': only the first message a comm sends/receives.
  --namespace NAMESPACE
                        RabbitMQ namespace.
  --host HOST
                        RabbitMQ host address.
  --vhost VHOST
                        RabbitMQ virtual host address.
  --user USER
                        RabbitMQ username.
  --password PASSWORD
                        RabbitMQ password.
  --cluster CLUSTER
                        Cluster that should be used.
  --default-comm DEFAULT_COMM, --defaultcomm DEFAULT_COMM
                        Comm type that should be used by default.
```

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

[Visit repo](#)[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$ yggrun yaml/shoot_v3.yml yaml/roots_v1.yml yaml/light_v1_python.yml yaml/timesync.yml --production-run
```

COMMAND LINE INTERFACE (CLI)

Run yggdrasil integration from the command line w/o opening Python

[Visit repo](#)[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79yl26:~$ yggrun yaml/shoot_v3.yml yaml/roots_v1.yml yaml/light_v1_python.yml yaml/timesync.yml --production-run
INFO:1423:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system jupyter-cropsinsilico-2dcis
2021-2dhackathon-2d9y79yl26 in namespace yggdrasil with rank 0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/ygg_light_v0.py
/srv/conda/envs/notebook/bin/python /home/jovyan/models/shoot_v3.py 0.0 48.0 6.0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/roots_v1.py 0.0 2.0 0.5
End of input from temp_doy.
timesync server: End of input.
INFO:1423:runner.waitModels[553]:YggRunner(runner): shoot finished running.
No more messages from model process.
INFO:1423:DSLModelDriver.after_loop[131]:TimeSyncModelDriver(shoot2root): returncode = 0
INFO:1423:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:1423:runner.waitModels[553]:YggRunner(runner): light finished running.
INFO:1423:runner.waitModels[559]:YggRunner(runner): light finished exiting.
INFO:1423:runner.waitModels[553]:YggRunner(runner): roots finished running.
INFO:1423:runner.waitModels[559]:YggRunner(runner): roots finished exiting.
INFO:1423:runner.waitModels[553]:YggRunner(runner): shoot2root finished running.
INFO:1423:runner.waitModels[559]:YggRunner(runner): shoot2root finished exiting.
INFO:1423:runner.waitModels[573]:YggRunner(runner): All models completed
INFO:1423:runner.run[374]:YggRunner(runner):           init      0.000001
INFO:1423:runner.run[374]:YggRunner(runner):       load drivers    0.164377
INFO:1423:runner.run[374]:YggRunner(runner):     start drivers    0.372987
INFO:1423:runner.run[374]:YggRunner(runner):       run models    23.552745
INFO:1423:runner.run[374]:YggRunner(runner):         at exit     0.054842
INFO:1423:runner.run[376]:YggRunner(runner): =====
INFO:1423:runner.run[377]:YggRunner(runner):           Total     24.144952
```

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79yl26:~$ █
```

“PRODUCTION RUN” FLAG

Speeds up execution by skipping type checking and input validation
(only used once integration has been debugged)

```
In [10]: run(['yamls/light_v1_python.yml', 'yamls/shoot_v2_split.yml'], production_run=True)
mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.show()
```

[Visit repo](#)[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$ yggrun yamls/
yamls/roots_v1.yml yamls/light_v
1_python.yml yamls/timesync.yml --production-run
INFO:1423:runner.startDrivers[499]:YggRunner(runner): Starting I/O drivers and models on system jupyter-cropsinsilico-2dcis
2021-2dhackathon-2d9y79y126 in namespace yggdrasil with rank 0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/ygg_light_v0.py
/srv/conda/envs/notebook/bin/python /home/jovyan/models/shoot_v3.py 0.0 48.0 6.0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/roots_v1.py 0.0 2.0 0.5
End of input from temp_doy.
timesync server: End of input.
INFO:1423:runner.waitModels[553]:YggRunner(runner): shoot finished running.
No more messages from model process.
INFO:1423:DSLModelDriver.after_loop[131]:TimeSyncModelDriver(shoot2root): returncode = 0
INFO:1423:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:1423:runner.waitModels[553]:YggRunner(runner): light finished running.
```

“PRODUCTION RUN” FLAG

Speeds up execution by skipping type checking and input validation
(only used once integration has been debugged)

```
In [10]: run(['yamls/light_v1_python.yml', 'yamls/shoot_v2_split.yml'], production_run=True)
mesh = trimesh.load_mesh('output/mesh_008.obj')
mesh.show()
```

[Visit repo](#)[Copy Binder link](#)

```
jovyan@jupyter-cropsinsilico-2dcis2021-2dhackathon-2d9y79y126:~$ yggrun yamls/
yamls/roots_v1.yml yamls/light_v
1_python.yml yamls/timesync.yml --production-run
INFO:1423:runner.startDrivers[1]: YggRunner(runner): Starting I/O drivers and models on system jupyter-cropsinsilico-2dcis
2021-2dhackathon-2d9y79y126 in namespace yggdrasil with rank 0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/ygg_light_v0.py
/srv/conda/envs/notebook/bin/python /home/jovyan/models/shoot_v3.py 0.0 48.0 6.0
/srv/conda/envs/notebook/bin/python /home/jovyan/models/roots_v1.py 0.0 2.0 0.5
End of input from temp_doy.
timesync server: End of input.
INFO:1423:runner.waitModels[553]:YggRunner(runner): shoot finished running.
No more messages from model process.
INFO:1423:DSLModelDriver.after_loop[131]:TimeSyncModelDriver(shoot2root): returncode = 0
INFO:1423:runner.waitModels[559]:YggRunner(runner): shoot finished exiting.
INFO:1423:runner.waitModels[553]:YggRunner(runner): light finished running.
```

ADDITIONAL EXAMPLES



Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. We will be using Github Issues to track problems encountered during the hackathon.
- Try launching a mybinder instance by clicking on this icon (or the link below). It may take a few moments to initialize. If you encounter an error, open an issue and try with another browser. If you still cannot launch the binder, follow the instruction [here](#) for downloading and installing the materials locally.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [yggdrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)

40+ EXAMPLES IN SUPPORTED LANGUAGES



Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. We will be using Github Issues to track problems encountered during the hackathon.
- Try launching a mybinder instance by clicking on this icon (or the link below). It may take a few moments to initialize. If you encounter an error, open an issue and try with another browser. If you still cannot launch the binder, follow the instruction [here](#) for downloading and installing the materials locally.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [vaadrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)

40+ EXAMPLES IN SUPPORTED LANGUAGES

README.md

 launch binder

Getting Started

All of the materials for the 2021 Crops in Silico Hackathon can be found in [this repository](#).

Requirements

- Browser (tested on Google Chrome, Safari, Firefox)
- Github Account

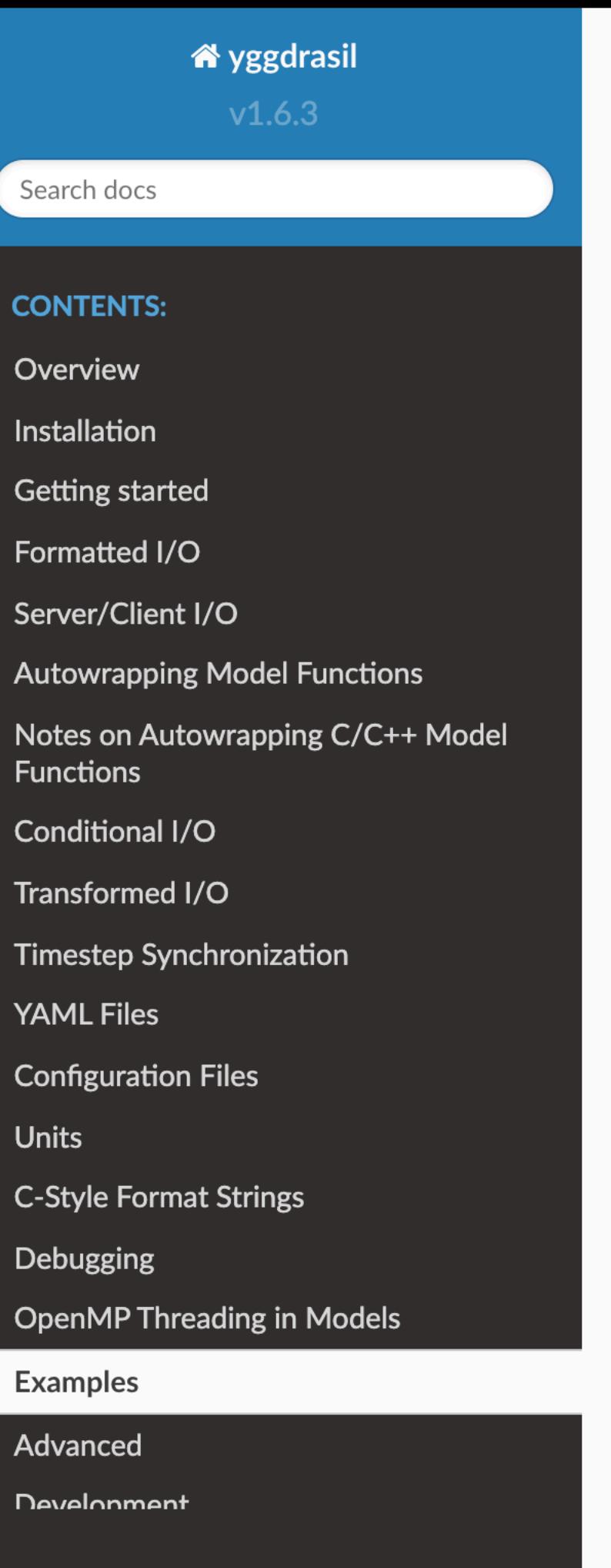
Preparing for the hackathon

- Check that you can sign-in to Github, creating an account as necessary. This will help you troubleshoot problems encountered during the hackathon.
- Try launching a mybinder instance by clicking on this  [launch binder](#) icon. It may take a few moments to initialize. If you encounter an error, open an issue and try visiting the [yggdrasil GitHub repository](#). If you still have trouble launching the binder, follow the instruction [here](#) for downloading and installing the software.

<https://mybinder.org/v2/gh/cropsinsilico/CiS2021-hackathon/HEAD>

Useful links

- [Hackathon Repository](#)
- [yggdrasil Repository](#)
- [yggdrasil Documentation](#)
- [Additional Examples](#)
- [Debugging Tips & Documented Errors](#)



The screenshot shows the yggdrasil documentation website. At the top, it says "yggdrasil v1.6.3". Below that is a search bar labeled "Search docs". The main content area is titled "CONTENTS:" and lists various sections: Overview, Installation, Getting started, Formatted I/O, Server/Client I/O, Autowrapping Model Functions, Notes on Autowrapping C/C++ Model Functions, Conditional I/O, Transformed I/O, Timestep Synchronization, YAML Files, Configuration Files, Units, C-Style Format Strings, Debugging, OpenMP Threading in Models, Examples, Advanced, and Development.

40+ EXAMPLES IN SUPPORTED LANGUAGES

[View page source](#)

Examples

Name	Description
SaM	One model that receives input from two files and sends output to one file in the temporary directory. The YAML uses the deprecated driver-based method to represent connections.
ascii_io	A single model that receives data from three input files and sends received messages to three output files. The first file is an unstructured ASCII text file, the second file is a table delimited ASCII table that is read row by row, and the third file is a table identical to the second that is read all at once.
backwards	A single model using the deprecated "Cis" versions of the interface.
conditional_io	Three models, A, B1, & B2, that conditionally pass messages. Model A receives input from a tab delimited table and sends output to both models B1 & B2. The outputs to models B1 & B2 only succeed if the data satisfies conditions described in the YAML. Output from both models B1 & B2 are sent to a tab delimited table.
fakeplant	Four models that can be run in isolation or as an integration. Each model approximates a simplified model of a process governing plant growth. When run in isolation each model receives input from a file and sends output to a file. In the larger integration, the canopy model receives input from three files (initial structure, time steps, and some growth parameters). For each time step received, the canopy model also receives a growth rate from the growth model, computes the new structure, and send the structure to the light model. The light model receives the ambient light level from a file, calculates the intensity for each element of the structure, and sends the output to the photosynthesis model. The photosynthesis model receives temperature and CO ₂ from files, calculates the photosynthesis rate, and sends the result to the growth model. The growth model calculates the growth rate and sends the output to the canopy model.
formatted_io1	Two models, A & B, that send/receive ASCII data (strings). Model A receives input from a file and then sends its output to model B. Model B receives input from model A and sends its output to a file.
formatted_io2	Two models, A & B, that send/receive rows of table data. Model A receives input from a file and then sends its output to model B. Model B receives input from model A and sends its output to a file.

QUESTIONS?

Github: <https://github.com/cropsinsilico/yggdrasil>

Docs: <https://cropsinsilico.github.io/yggdrasil/>

Paper: <https://doi.org/10.1093/insilicoplants/diz001>

Project Website: <http://cropsinsilico.org/>