CLIMLAB 2.0: lessons learned and future roadmap for interactive, process-oriented climate modeling

Brian E. J. Rose
University at Albany (SUNY)

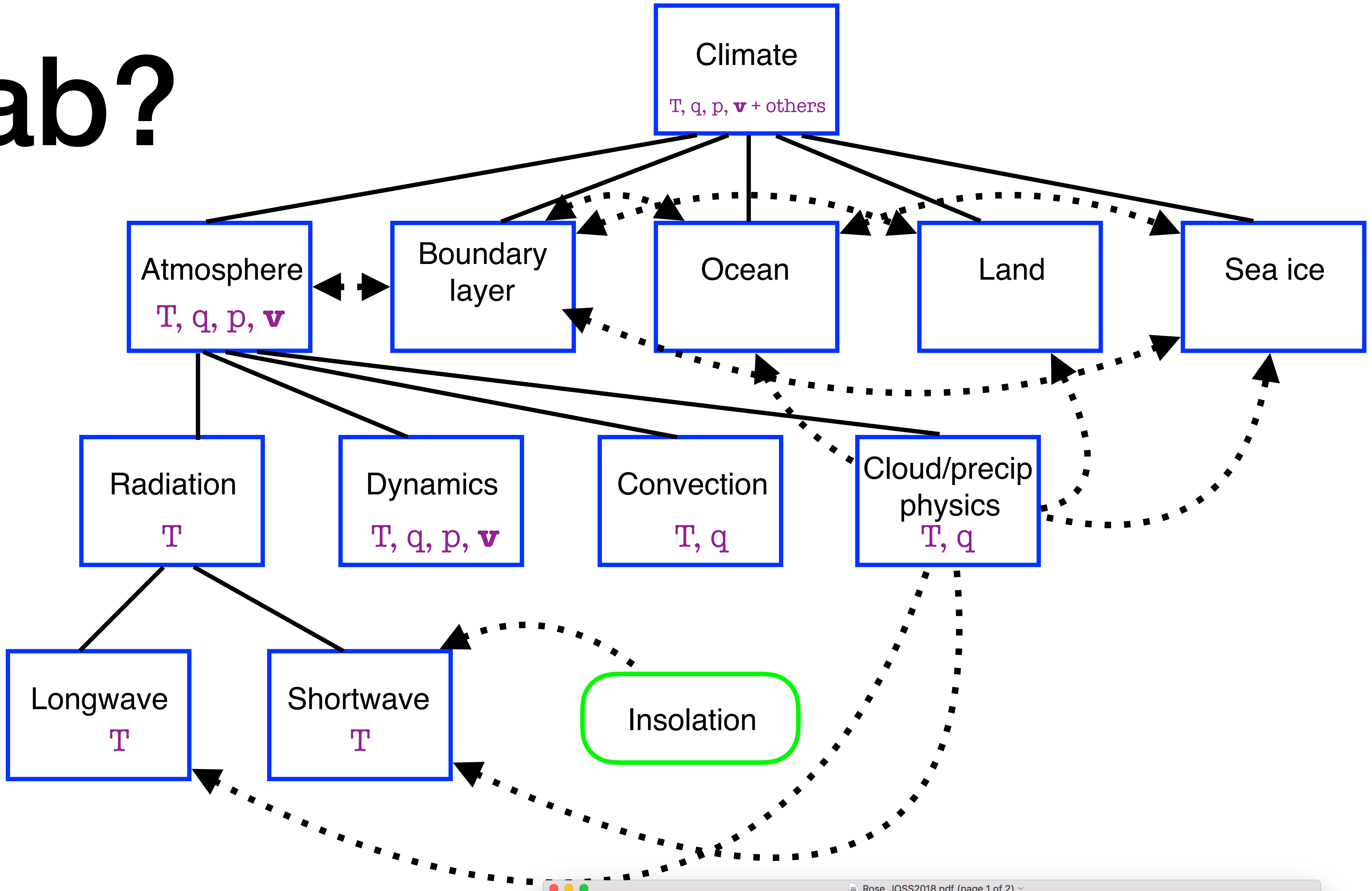
10th Symposium on Advances in Modeling and Analysis Using Python
AMS Annual Meeting 2020

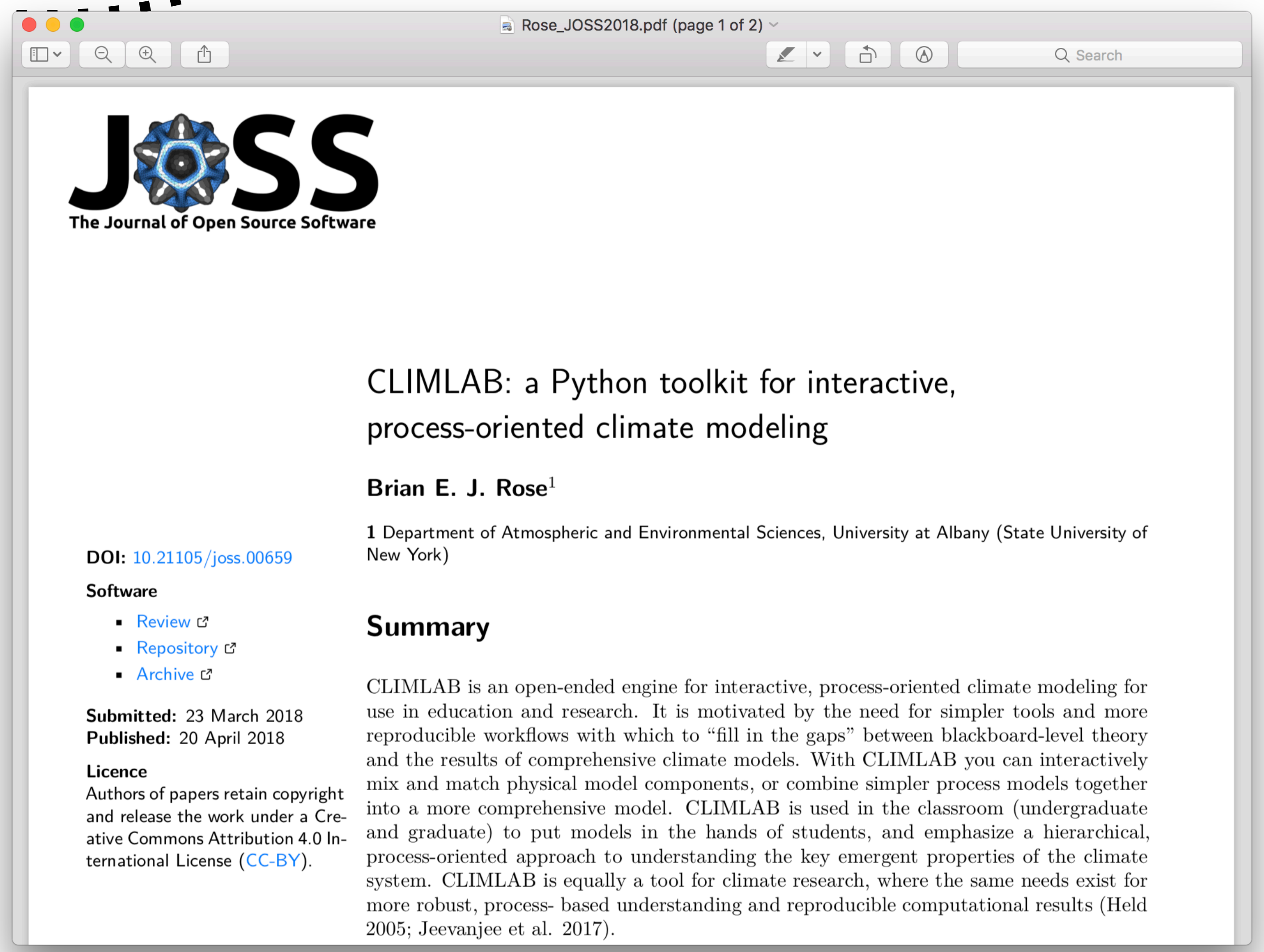
UNIVERSITY AT ALBANY
State University of New York

What is climlab?

CLIMLAB

- A Python-based **CLIMate LABoratory** for education and research!
- Toolkit for building **climate models** from **modular components**
- Designed for **easy, interactive investigation**
- **Open-source**, **documented** and hosted on **GitHub.**
- Development funded through NSF CAREER award





Rose_JOSS2018.pdf (page 1 of 2)

JOSS The Journal of Open Source Software

CLIMLAB: a Python toolkit for interactive, process-oriented climate modeling

Brian E. J. Rose[1]

1 Department of Atmospheric and Environmental Sciences, University at Albany (State University of New York)

DOI: 10.21105/joss.00659

Software

- Review
- Repository
- Archive

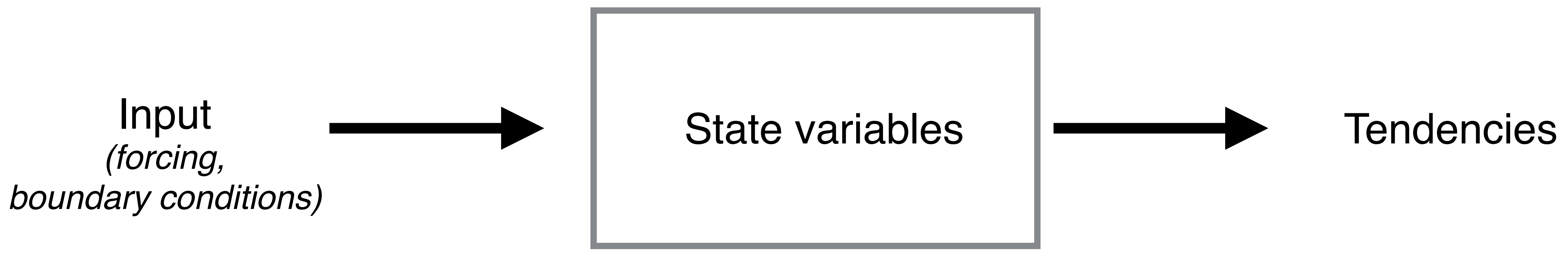
Submitted: 23 March 2018
Published: 20 April 2018

Summary

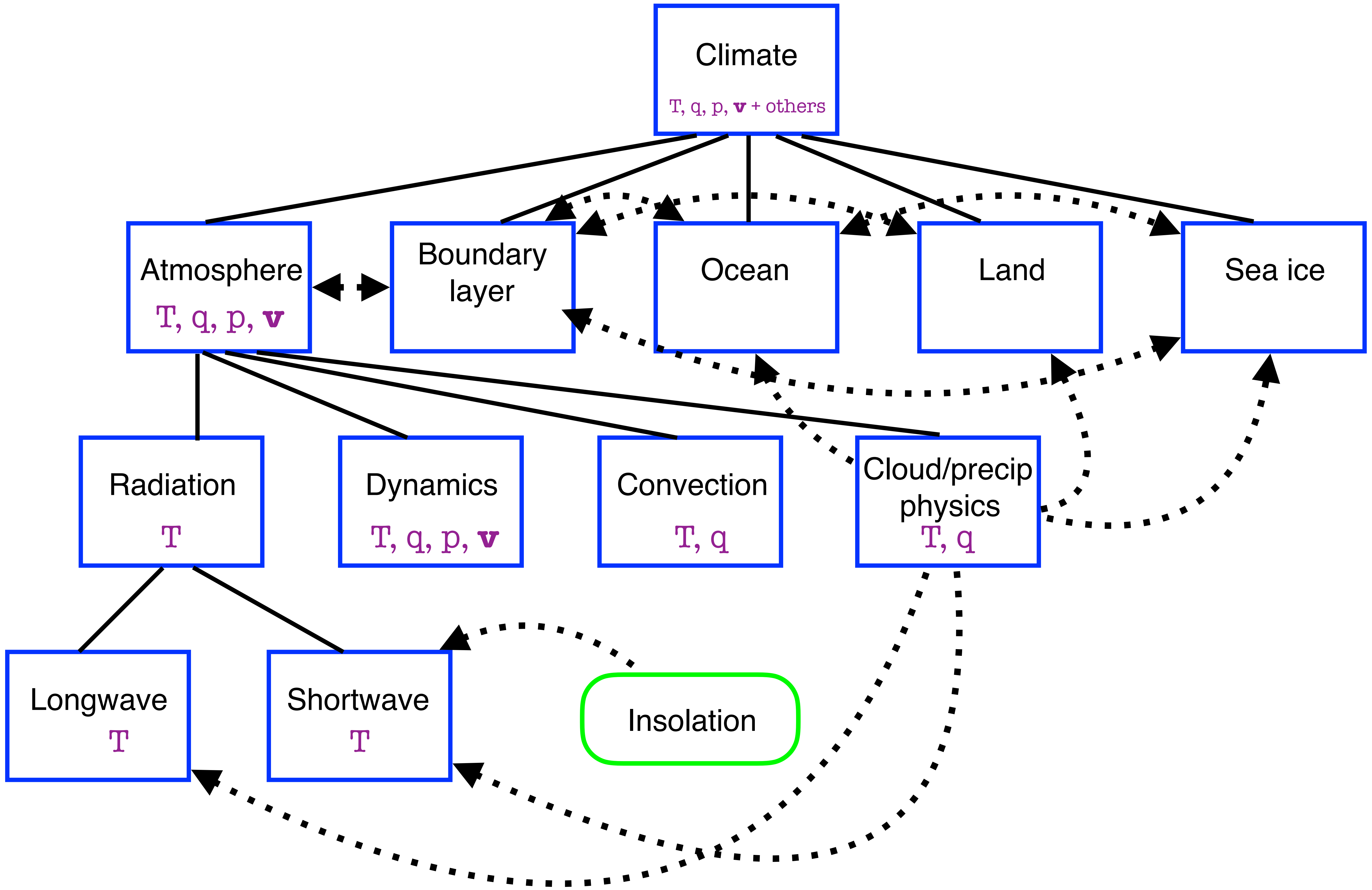
CLIMLAB is an open-ended engine for interactive, process-oriented climate modeling for use in education and research. It is motivated by the need for simpler tools and more reproducible workflows with which to "fill in the gaps" between blackboard-level theory and the results of comprehensive climate models. With CLIMLAB you can interactively mix and match physical model components, or combine simpler process models together into a more comprehensive model. CLIMLAB is used in the classroom (undergraduate and graduate) to put models in the hands of students, and emphasize a hierarchical, process-oriented approach to understanding the key emergent properties of the climate system. CLIMLAB is equally a tool for climate research, where the same needs exist for more robust, process- based understanding and reproducible computational results (Held 2005; Jeevanjee et al. 2017).

http://climlab.readthedocs.io

The `climlab.Process` object:



With climlab you build models as hierarchical trees of these objects



http://climlab.readthedocs.io

Current capabilities in CLIMLAB

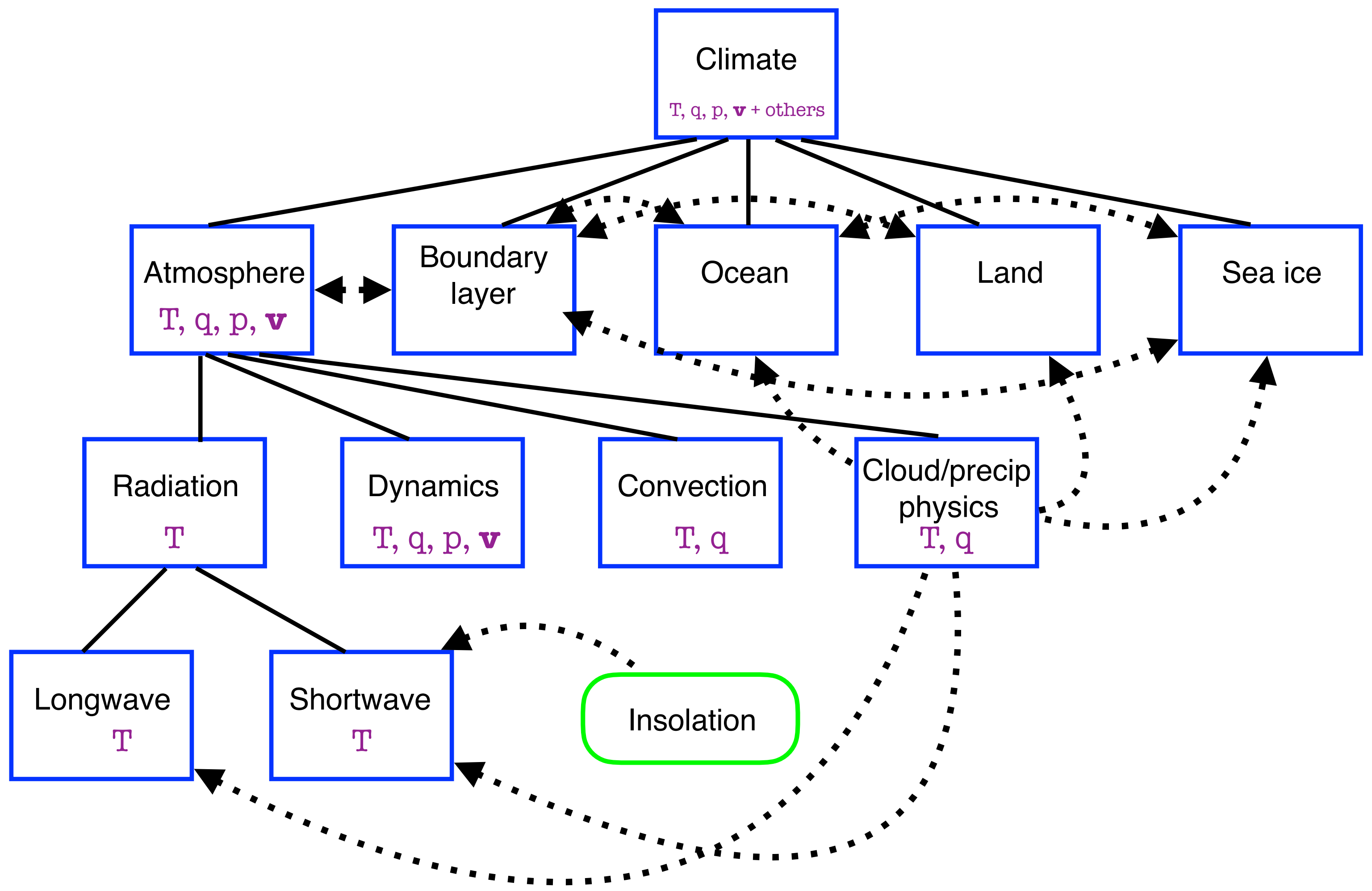
CLIMLAB

- Insolation and orbital calculations
- Several radiation schemes (including **RRTMG)**
- Simple convection schemes
- Bulk formula boundary layer schemes
- Flexible advection-diffusion solvers along arbitrary dimensions
- Slab ocean and diffusive deep ocean
- Simple land surface scheme (bucket hydrology)
- Arbitrary combinations of all-of-the-above on 1D, 2D or 3D grids.
- (partial) integration with xarray for data management and visualization

xarray

http://climlab.readthedocs.io

CLIMLAB



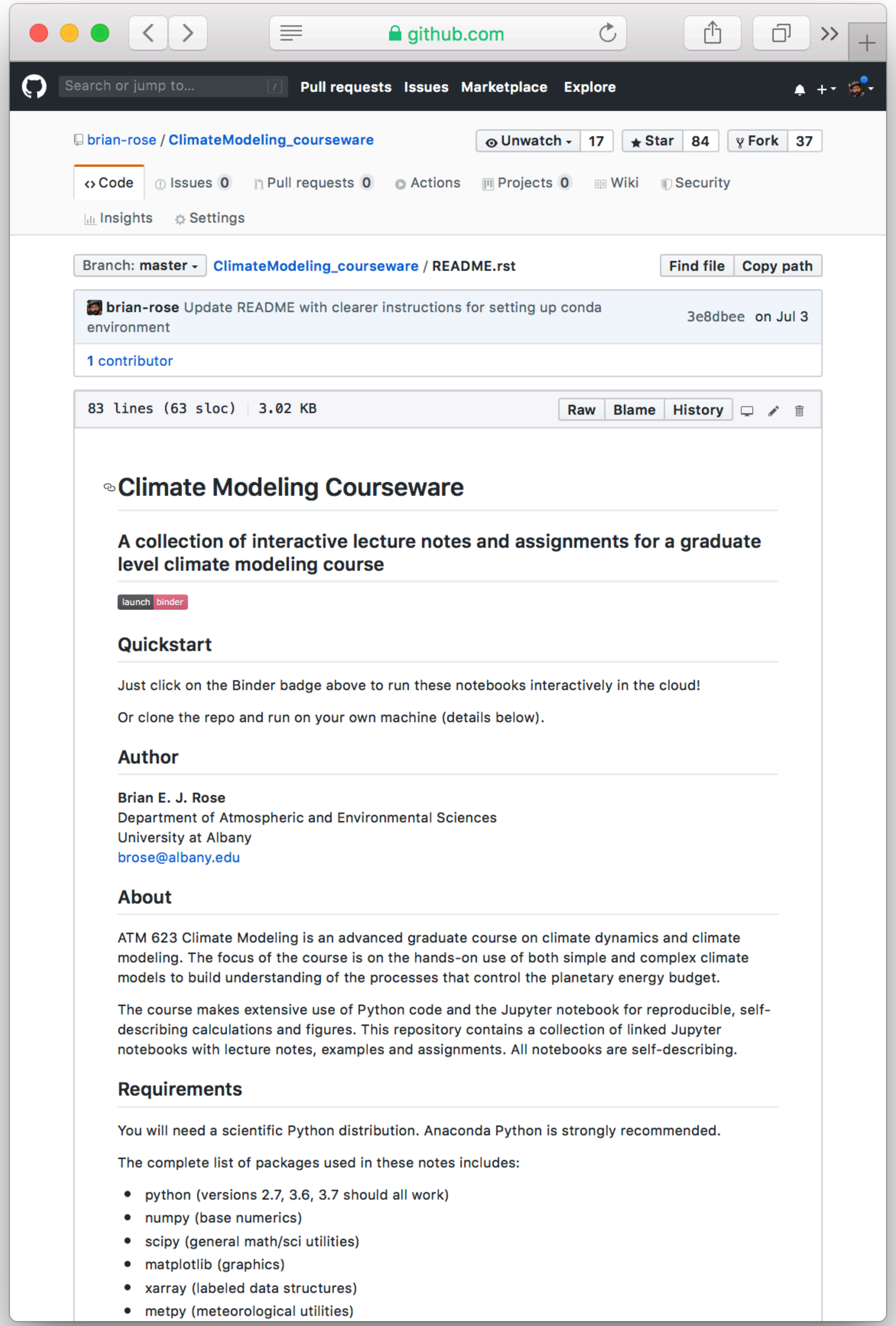
The goal of climlab is to enable more and better science with climate models across the hierarchy of complexity:

- ➤more transparent
- ➤more reproducible
- ➤more accessible to students

This begins in the classroom.
Active learning with climate models!

http://climlab.readthedocs.io

A full semester of Jupyter notebooks leveraging climlab


github.com

Search or jump to... Pull requests Issues Marketplace Explore

brian-rose / ClimateModeling_courseware

Unwatch 17 Star 84 Fork 37

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security Insights Settings

Branch: master ClimateModeling_courseware / README.rst Find file Copy path

brian-rose Update README with clearer instructions for setting up conda environment 3e8dbee on Jul 3

1 contributor

83 lines (63 sloc) 3.02 KB Raw Blame History

Climate Modeling Courseware

A collection of interactive lecture notes and assignments for a graduate level climate modeling course

launch binder

Quickstart

Just click on the Binder badge above to run these notebooks interactively in the cloud!

Or clone the repo and run on your own machine (details below).

Author

Brian E. J. Rose
Department of Atmospheric and Environmental Sciences
University at Albany
brose@albany.edu

About

ATM 623 Climate Modeling is an advanced graduate course on climate dynamics and climate modeling. The focus of the course is on the hands-on use of both simple and complex climate models to build understanding of the processes that control the planetary energy budget.

The course makes extensive use of Python code and the Jupyter notebook for reproducible, self-describing calculations and figures. This repository contains a collection of linked Jupyter notebooks with lecture notes, examples and assignments. All notebooks are self-describing.

Requirements

You will need a scientific Python distribution. Anaconda Python is strongly recommended.

The complete list of packages used in these notes includes:

- python (versions 2.7, 3.6, 3.7 should all work)
- numpy (base numerics)
- scipy (general math/sci utilities)
- matplotlib (graphics)
- xarray (labeled data structures)
- metpy (meteorological utilities)

Lecture notes

1. Planetary energy budget
2. Solving the zero-dimensional EBM
3. Climate Sensitivity and Feedback
4. Intro to CLIMLAB
5. The climate system and climate models
6. A Brief Review of Radiation
7. Elementary greenhouse models
8. Grey radiation modeling with climlab
9. Modeling non-scattering radiative transfer
10. Who needs spectral bands? We do. Some baby steps...
11. Radiative-Convective Equilibrium
12. Transient and equilibrium response to CO2 in the CESM
13. Toy models of transient warming
14. Clouds and cloud feedback
15. Insolation
16. Orbital variations, insolation, and the ice ages
17. Heat transport
18. The one-dimensional energy balance model
19. Seasonal cycle and heat capacity
20. A peak at numerical methods for diffusion models
21. Ice albedo feedback in the EBM
22. Snowball Earth and Large Ice Cap Instability in the EBM
23. The surface energy balance
24. Land-ocean contrasts under climate change
25. Water, water everywhere

COMING SOON: first release of
"The Climate Laboratory"
An open-source, interactive textbook
powered by **JupyterBook**

localhost

Installation — RISE 5.1.0 documentation | conda-forge feedstocks | community driven p... | Lectures/ | Lecture10 -- Radiative-Convective Equilibri...

ATM 623: Climate Modeling

Brian E. J. Rose, University at Albany

Lecture 10: Radiative-Convective Equilibrium

?

Here is an example of building a single-column RCM in climlab

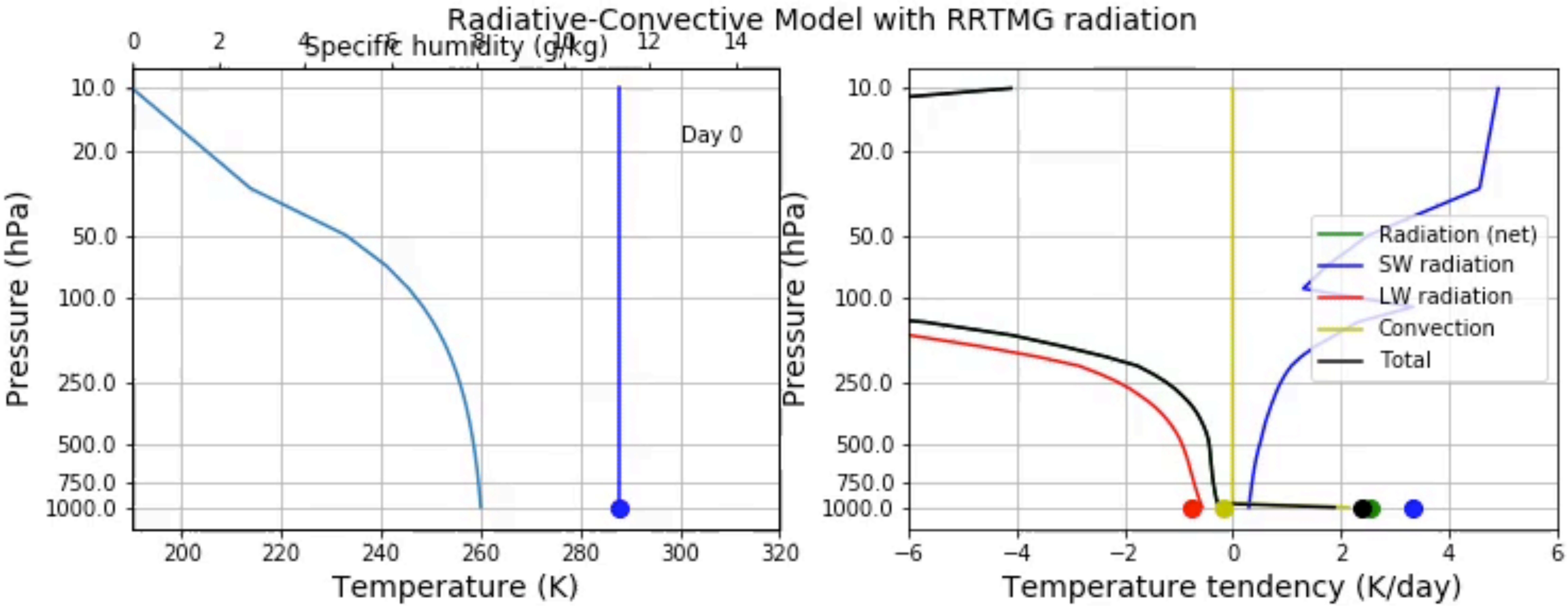
In [3]:

```
import climlab
#  Choose the surface albedo
alb = 0.2
#  State variables (Air and surface temperature)
state = climlab.column_state(num_lev=50)
#  Parent model process
rcm = climlab.TimeDependentProcess(state=state)
#  Fixed relative humidity
h2o = climlab.radiation.ManabeWaterVapor(state=state)
#  Couple water vapor to radiation
rad = climlab.radiation.RRTMG(state=state, specific_humidity=h2o.q, albedo=alb)
#  Convective adjustment
conv = climlab.convection.ConvectiveAdjustment(state=state, adj_lapse_rate=6)
#  Couple everything together
rcm.add_subprocess('Radiation', rad)
rcm.add_subprocess('WaterVapor', h2o)
rcm.add_subprocess('Convection', conv)
#rcm.compute_diagnostics()
```

```
Getting ozone data from /Users/br546577/anaconda3/envs/atm623/lib/python3.6/site-packages/climlab
/radiation/data/ozone/apeozone_cam3_5_54.nc
```

From isothermal initial condition to Radiative-Convective Equilibrium

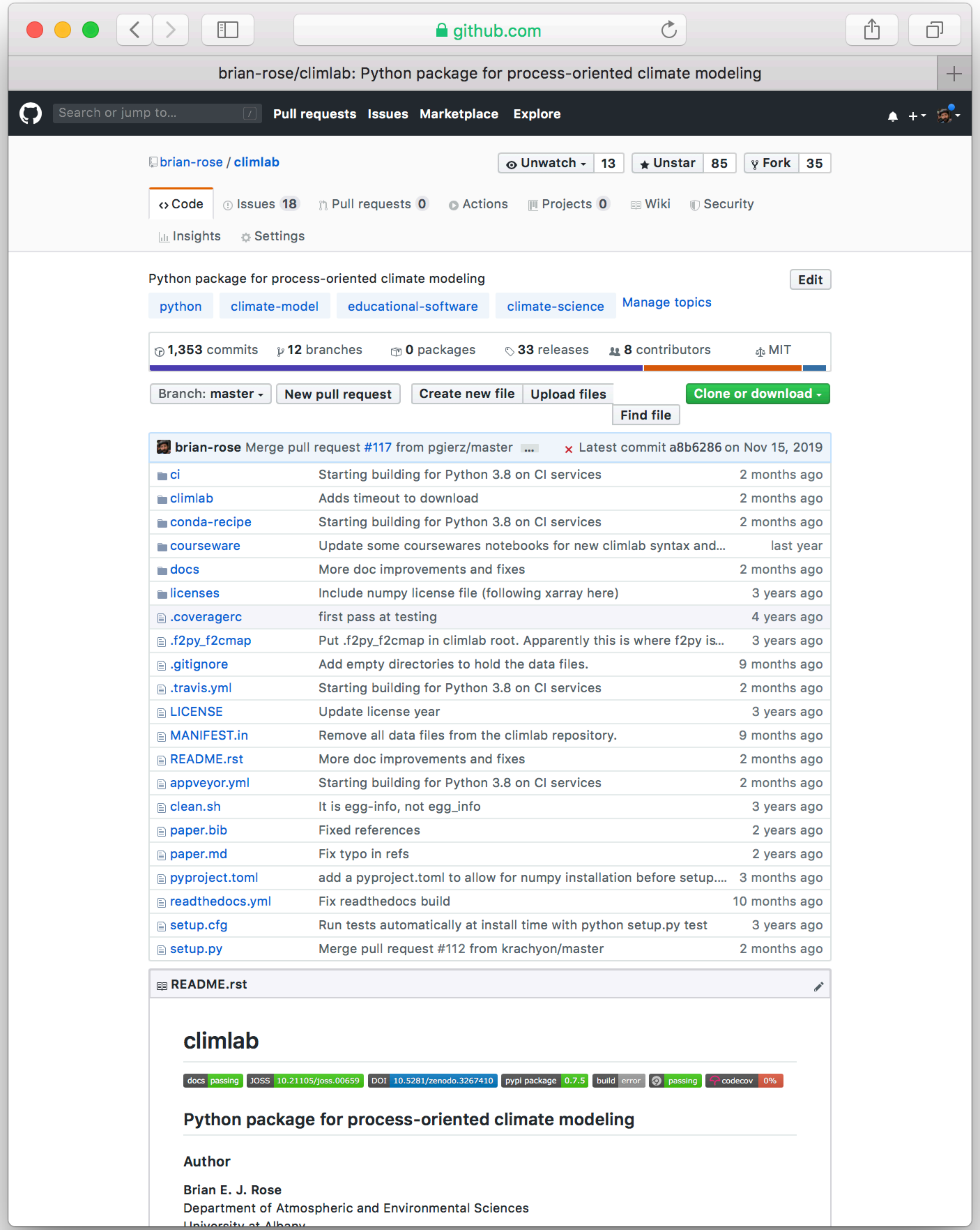
This animation is built interactively in the notebook using `matplotlib.animation`



Impact, uptake, and usage of CLIMLAB

- Latest version 0.7.5 released July 2019
- 63 students used it in my classes (undergrad + grad)
- Used by small handful of instructors elsewhere
- 85 stars, 35 forks on GitHub
- 48 issues closed, 54 PRs closed
- Uncounted emails, tweets, and hallway conversations
- 5 citations of JOSS paper

Lots of community interest, but less actual use, not much community development



What I **would not do** if I were starting the CLIMLAB project today

CLIMLAB

- Write **numerical code** before designing a **user interface**
- Create clunky **custom data structure** for **grid-aware** operations
- Extensive use of **class inheritance**, implicit ****kwargs**
- Write **documentation** focusing on **implementation** rather than **use**
- Single, monolithic code base (resource-intensive compile-and-test)
- Support Python 2

*My impression: many of these issues are **barriers to new users and potential user-developers***

CLIMLAB 2.0

also, probably, a better logo

- **Goals?**
  - **More intuitive and extensible user interface**
  - **More comprehensible codebase**
  - **Easier, more interactive visualization**
  - **Clear framework and roadmap for development of new science capabilities**
- **How?**
  - CLIMLAB processes will **consume and produce xarray objects**
  - **More explicit inputs and outputs**, fewer hidden details. **Fewer classes, more functions!**
  - **Federation of packages**: climlab will focus on **UI** and **model coupling.** Numerics (e.g. RRTMG wrapper) spun off
  - **Separate viz package** powered by Bokeh will enable **live, browser-based animations** of in-memory simulations
- **When?**
  - …
  - Sooner if I get a few more helping hands

The evolving ecosystem

When I began this project…. (January 2014)

- Science libraries were mostly Python 2 only
- xarray (now a core dependency) didn't exist (v0.1 released May 2014)
- Jupyter notebook was called IPython notebook (v1.1)
- Jupyterhub didn't exist (v0.1 released March 2015)
- conda-forge didn't exist (2016)
- I was distributing *.py files to students by email

Being part of this extremely rapid, community-driven explosion of tools has been both a joy and a frustration.

Mostly a joy.

Resources

- CLIMLAB meta-paper: *Rose (2018): CLIMLAB: a Python toolkit for interactive, process-oriented climate modeling, JOSS (3), doi:10.21105/joss.00659*
- CLIMLAB source code: https://github.com/brian-rose/climlab
- Documentation: http://climlab.readthedocs.io
- Lecture notes in Jupyter notebook format:
  - Source: https://github.com/brian-rose/ClimateModeling_courseware
  - To view as static pages: http://nbviewer.ipython.org/github/brian-rose/ClimateModeling_courseware/blob/master/index.ipynb
  - To run in the cloud: https://mybinder.org/v2/gh/brian-rose/ClimateModeling_courseware/master

*`CLIMLAB` software and course notes are **open-source** and under **active development**.*

*Community contributions very welcome. **Fork away!***

localhost

Lectures/ | Lecture16 -- Orbital variations - Jupyter Notebook

ATM 623: Climate Modeling

Brian E. J. Rose, University at Albany

Lecture 16: Orbital variations, insolation, and the ice ages

3. Global, seasonal distribution of insolation (present-day orbital parameters)

Calculate an array of insolation over the year and all latitudes (for present-day orbital parameters). We'll use a dense grid in order to make a nice contour plot

In [7]:
```
lat = np.linspace( -90., 90., 500)
days = np.linspace(0, const.days_per_year, 365 )
Q = daily_insolation( lat, days )
```

And make a contour plot of Q as function of latitude and time of year.

In [8]:
```
fig, ax = plt.subplots(figsize=(10,8))
CS = ax.contour( days, lat, Q , levels = np.arange(0., 600., 50.) )
ax.clabel(CS, CS.levels, inline=True, fmt='%r', fontsize=10)
ax.set_xlabel('Days since January 1', fontsize=16 )
ax.set_ylabel('Latitude', fontsize=16 )
ax.set_title('Daily average insolation', fontsize=24 )
ax.contourf ( days, lat, Q, levels=[-1000., 0.], colors='k' )
```

Out[8]:
```
<matplotlib.contour.QuadContourSet at 0x1269a34e0>
```

3. Global, seasonal distribution of insolation (present-day orbital parameters)

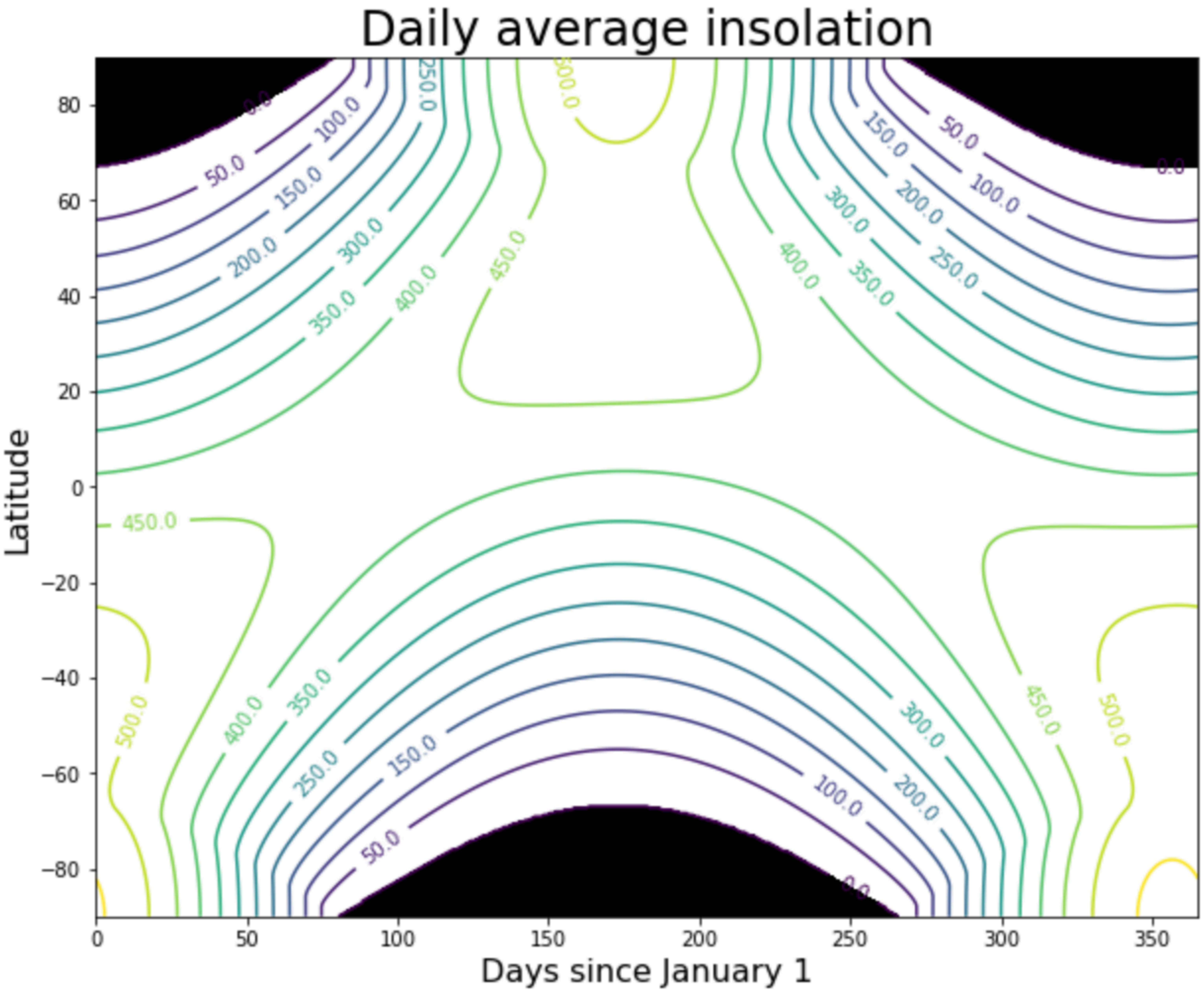
Calculate an array of insolation over the year and all latitudes (for present-day orbital parameters). We'll use a dense grid in order to make a nice contour plot

In [7]:
```
lat = np.linspace( -90., 90., 500)
days = np.linspace(0, const.days_per_year, 365 )
Q = daily_insolation( lat, days )
```

And make a contour plot of Q as function of latitude and time of year.

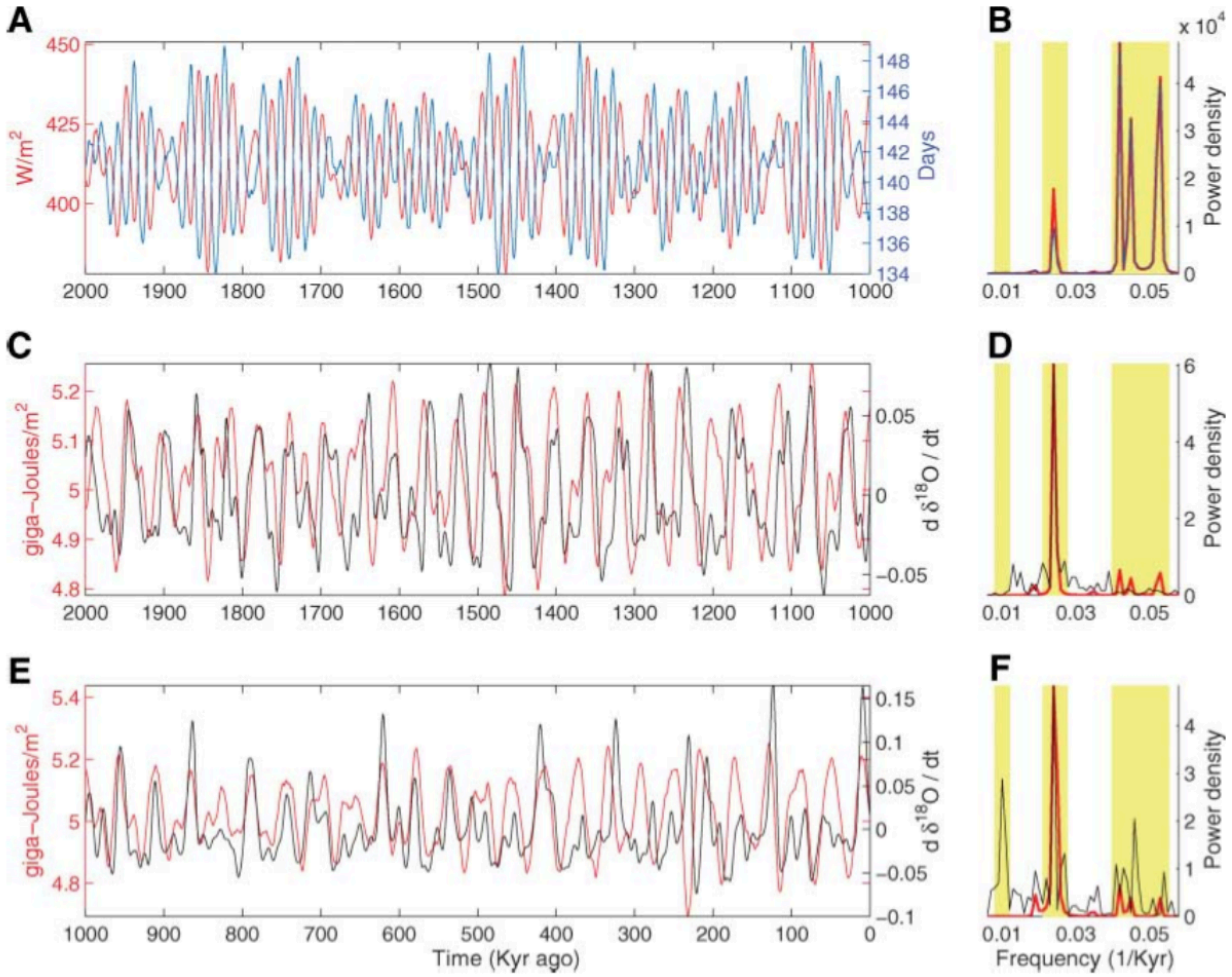
In [8]:
```
fig, ax = plt.subplots(figsize=(10,8))
CS = ax.contour( days, lat, Q , levels = np.arange(0., 600., 50.) )
ax.clabel(CS, CS.levels, inline=True, fmt='%r', fontsize=10)
ax.set_xlabel('Days since January 1', fontsize=16 )
ax.set_ylabel('Latitude', fontsize=16 )
ax.set_title('Daily average insolation', fontsize=24 )
ax.contourf ( days, lat, Q, levels=[-1000., 0.], colors='k' )
```

Out[8]: `<matplotlib.contour.QuadContourSet at 0x1269a34e0>`



Assignment: reproduce some published figures on high-latitude insolation (solsticial and seasonally integrated)

Fig. 2. Insolation forcing and Pleistocene glacial variability. **(A)** Number of days that insolation is above 275 W/m² (blue) and the average insolation intensity during this interval (red). Intensity and duration are anticorrelated. **(B)** Spectral estimate of the duration (blue) and intensity (red), showing that the majority of the variability is at the precession periods. Shaded bands from left to right indicate the 100-ky, 41-ky (obliquity), and 21-ky (precession) bands. **(C)** Summer energy (red) and the time rate of change of $\delta^{18}O$ (black) for the early Pleistocene and **(D)** the corresponding spectral estimates. Positive rates of change indicate decreasing ice volume. Variability in both records is predominantly at the 41-ky obliquity period. **(E** and **F)** Same as (C) and (D) but for the late Pleistocene. The time rate of change of $\delta^{18}O$ has variability at the 100-ky period not present in the forcing.



Huybers (2006) Science

Your tasks is to reproduce these orbital / insolation calculations using the `climlab` tools. Specifically:

1. **Reproduce the blue curve in Fig. 2A**: number of days per year that insolation is above 275 W m^{-2} at 65ºN, between 2000 and 1000 kyrs before present.
2. **Reproduce the red curve in Fig. 2C**: Integrated summer insolation: the total accumulated insolation for every day for which this insolation is above the threshold of 275 W m^{-2}.
3. **Reproduce the red curve in Fig. 2A**: Average summer insolation intensity. This is the integrated summer insolation above the intensity threshold divided by the number of seconds during which the insolation threshold is exceeded. (Preferably plot this on the same graph as the number of days to reproduce Fig. 2A completely).

CLIMLAB

CLIMLAB in the classroom

The problem:

Deploying “serious” scientific software (e.g. the RRTMG radiation code) so it’s accessible to students who’ve never compiled a Fortran code before

Two solutions:

CONDA-FORGE

Pre-built binaries, “one-click” installation on Mac, Linux and Windows!

jupyter

Running climlab remotely through a JupyterHub server

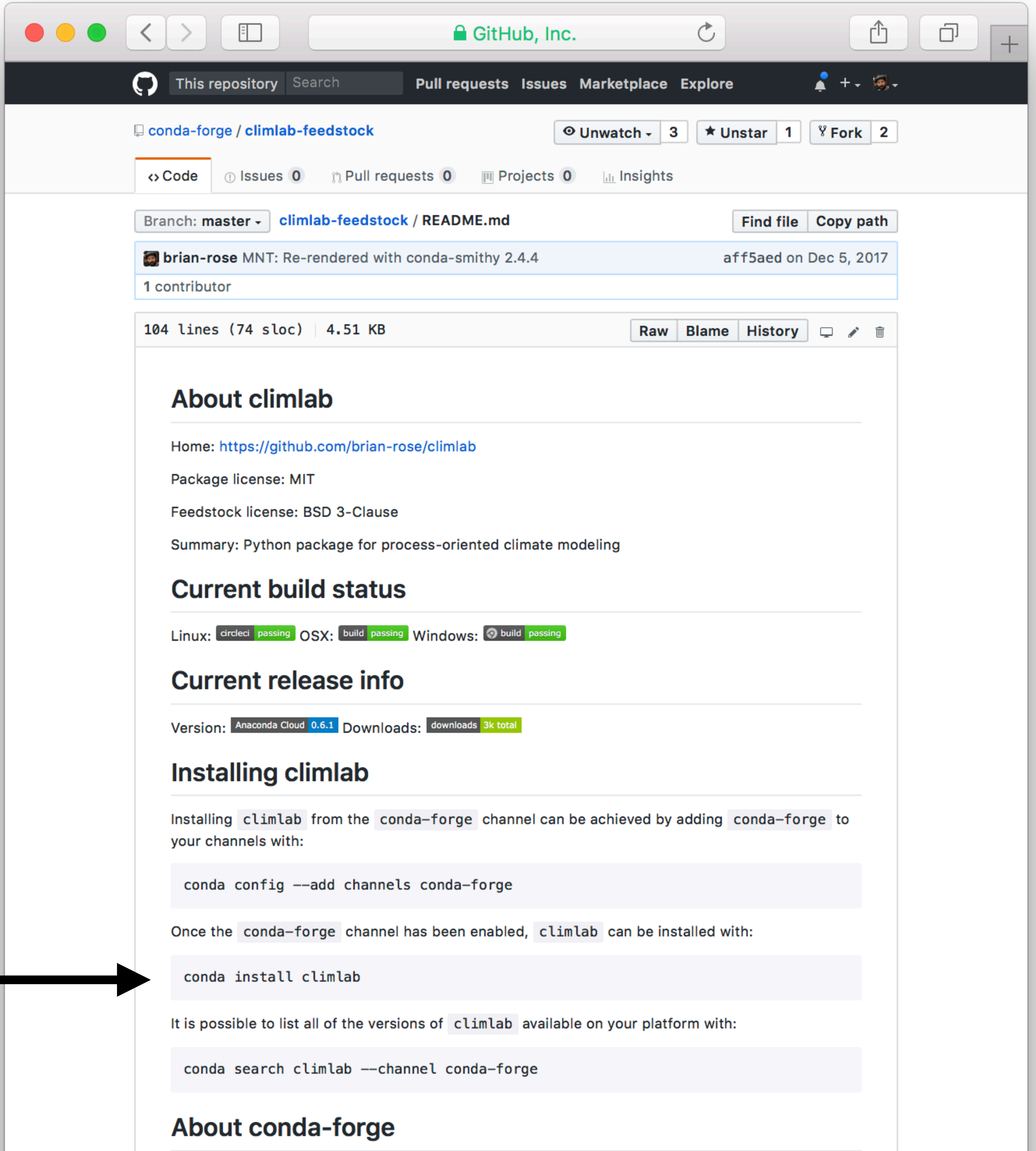
CONDA-FORGE

A community led collection of build recipes, infrastructure and distributions for the conda package manager.

Harnesses several CI services to build working binaries for Mac, Linux and Windows, and make

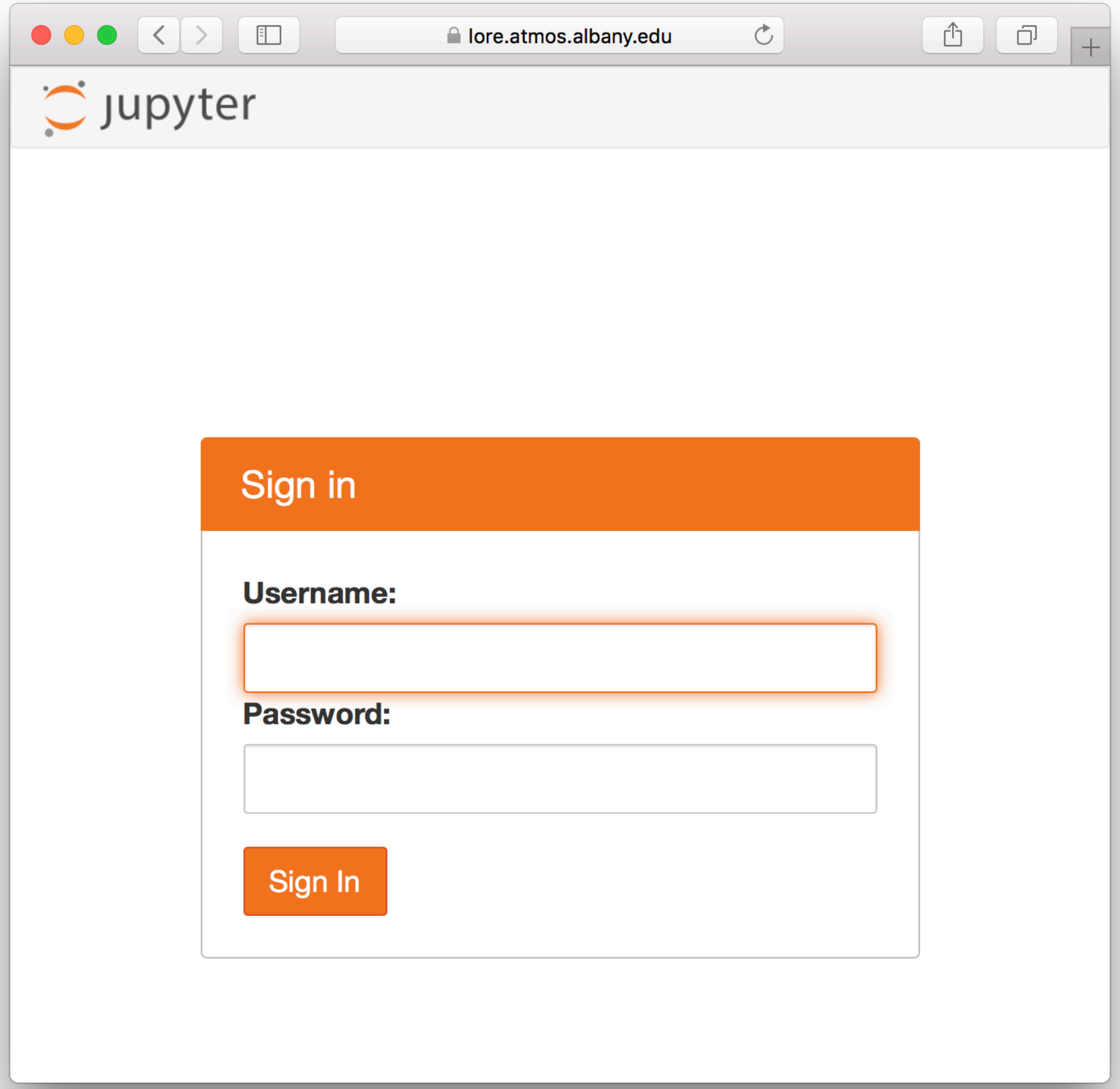
I (the climlab developer) don't need to touch Windows at all!

It "just works"

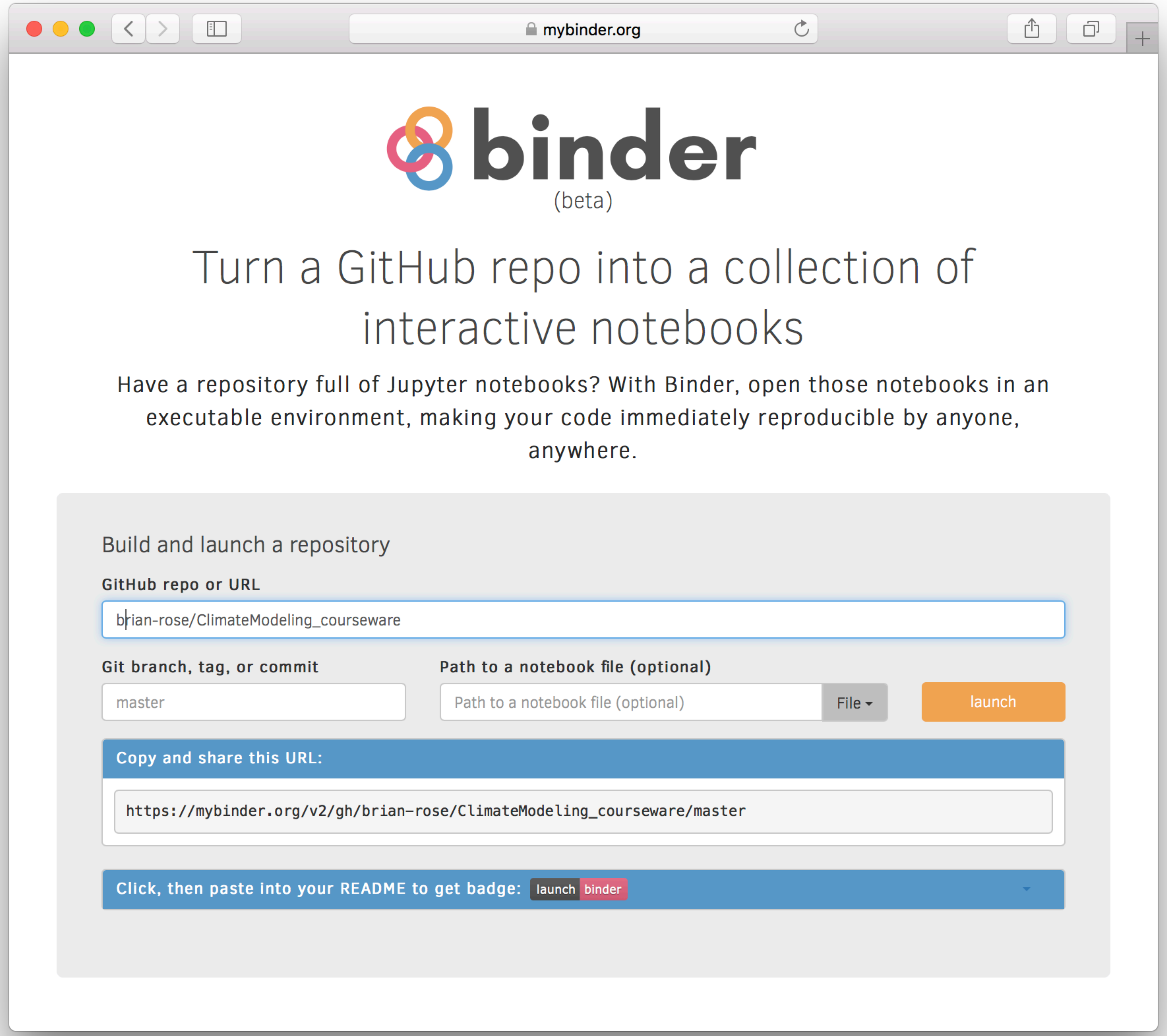


Running climlab remotely with JupyterHub

A dedicated **JupyterHub server** with a **complete** and **consistent** Python environment (UAlbany credentials required)

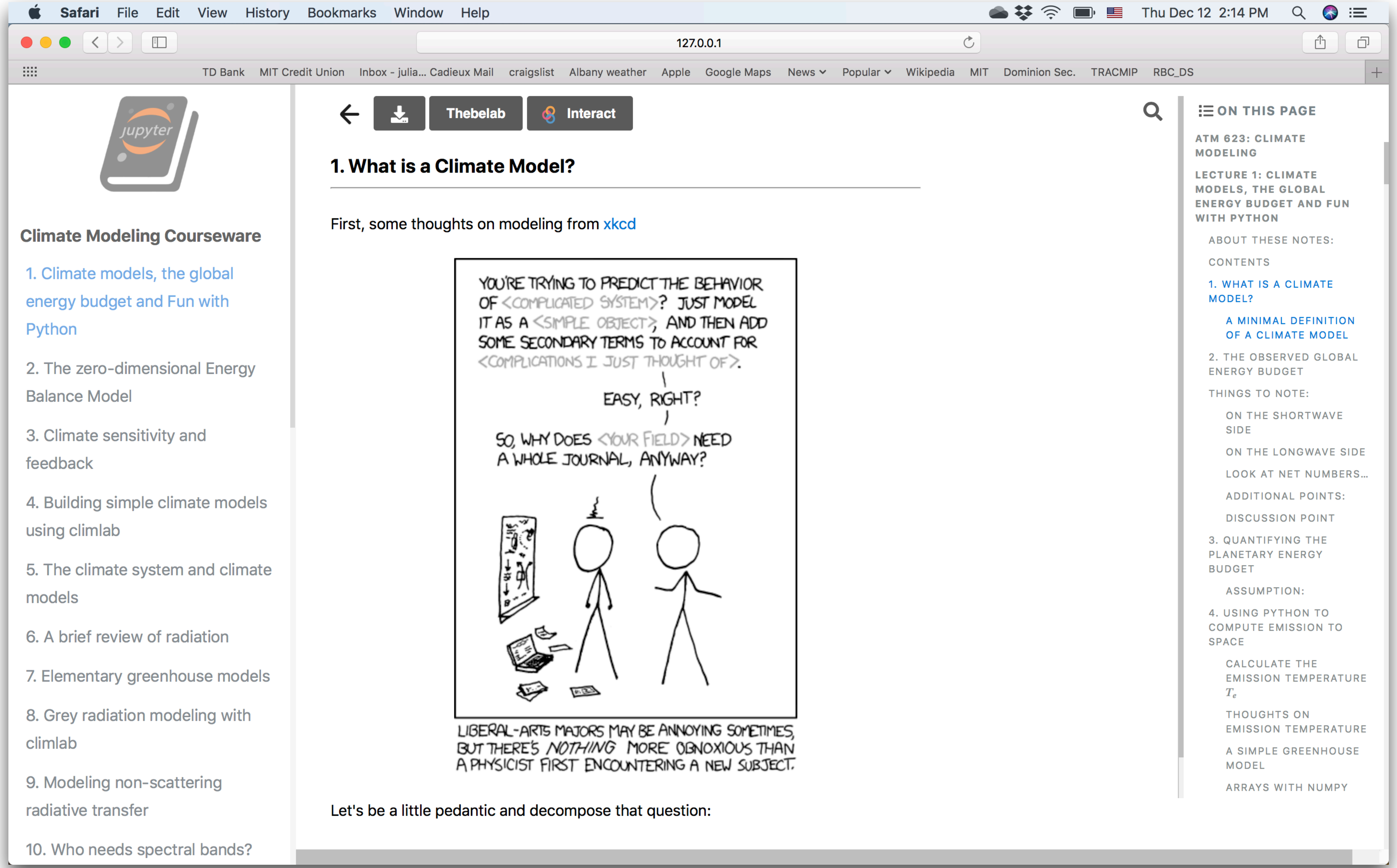


Cloud-based Jupyter and climlab with binder!



The next big thing: Packaging my loose collection of notebooks with JupyterBook

https://jupyterbook.org/



(coming soon)