A fond look at MesaScript, and introducing Mesa Reader for Python

Bill Wolf, UCSB
MESA Summer School 2014

MESA *Script*

For automating inlist generation and organization.

What is MesaScript?

- A domain-specific language (DSL)
- Combines Ruby programming language and inlist commands (you get all of Ruby for free in an inlist)
- Bottom line: you can use variables and control statements to easily generate individual or batches of inlists

Example (sample.rb)

```
require 'mesa_script'

masses = [1,2,3]
masses.each do |mass|
  Inlist.make_inlist('inlist_' + mass.to_s) do
    log_title = "LOGS_" + mass.to_s
    log_directory log_title

    mass_change 1e-7
    load_saved_model true
    saved_model_name "test_#{mass.to_s}.mod"
  end
end
```

Loads Inlist class

Loop over three masses

Define Variable

Assign Variables/ Values to inlist controls

```
>>> ruby sample.rb
```

```
inlist_1

inlist_2

inlist_3
```

```
&star_job
   load_saved_model = .true.
   saved_model_name = 'test_1.mod'

/ ! end of star_job namelist

&controls
   log_directory = 'LOGS_1'

   mass_change = 1d-07


/ ! end of controls namelist

&pgstar

/ ! end of pgstar namelist
```

inlist_1

- Commands automatically sorted to namelists
- Also automatically ordered by how they appear in defaults files
- Unknown commands will throw error
- Argument types checked, and if needed, converted on the fly

Why Use MesaScript?

- Decent error checking
- DRY (don’t repeat yourself) on batch jobs
- Better record-keeping than using `run_star_extras.f`
- Neatly formats and organizes inlist
- You can use variables and control statements!

From today's first exercise…

First define the variables of interest…

Loop through each case, making each appropriate inlist…

Small amount of personal witchcraft points main inlist to new one, then run.

```
require 'mesa_script'

stop_xs = [0.69, 0.40, 0.01]
names = {
  0.69 => "h_ignition",
  0.40 => "h_burn",
  0.01 => "h_depletion"
}

load_names = {
  0.40 => "h_ignition",
  0.01 => "h_burn"
}

stop_xs.each do |stop_x|
  Inlist.make_inlist("inlist_to_#{names[stop_x]}") do
      # save a model at the end of the run
        save_model when_terminate true
        save_model_filename "#{names[stop_x]}.model"

        load_saved_model(stop_x < 0.6)
        saved_model_name "#{load_names[stop_x]}.model"

        pgstar_flag true

        log_directory "LOGS_#{names[stop_x]}"

        initial_mass 20 # in Msun units

        xa_central_lower_limit_species[1] = 'h1'
        xa_central_lower_limit[1] = stop_x
    end
  system("mesa point inlist_to_#{names[stop_x]}")
  system("./rn")
end
```

Advanced Features

- Can use different namelists (can remove any of three default namelists or add new ones).

- Can convert existing inlists to mesascript with one line (using bundled script):

```
>>> inlist2mesascript inlist_project project.rb
```

Beware!!!

- In `Inlist.make_inlist` block:
  - Using assignment operator (=) will ALWAYS make/re-assign a variable
  - `initial_mass = 1.0` will create a variable named `initial_mass` and set it to 1.0
  - `initial_mass 1.0` and `initial_mass(1.0)` will cause `initial_mass = 1.0` to show up in resulting inlist (more likely what you want)

Array Assignments

- `xa_central_lower_limit(1) = 1e-2` will not do what you want it to do

- Instead, do `xa_central_lower_limit[1] = 1e-2`, `xa_central_lower_limit(1, 1e-2)`, or `xa_central_lower_limit 1, 1e-2`

Known Limitation(s)

- Just discovered today: can’t deal with rank 2+ arrays
- `Text_Summary1_name(1,3) = ‘log_cntr_Rho'`
- Others? Let me know.

How do I get it?

www.github.com/wmwolf/MesaScript

- README gives installation instructions
- Also more detailed documentation there
- Contact me if you need help or have suggestions!

PyMesaReader… or mesa_reader.py… or… whatever. No logo this time.

Public Launch TODAY!

What is it?

- Series of classes for dealing with MESA output files (profiles / history / whole log directory)

- Sort of a restricted mesa.py (if you've used that)

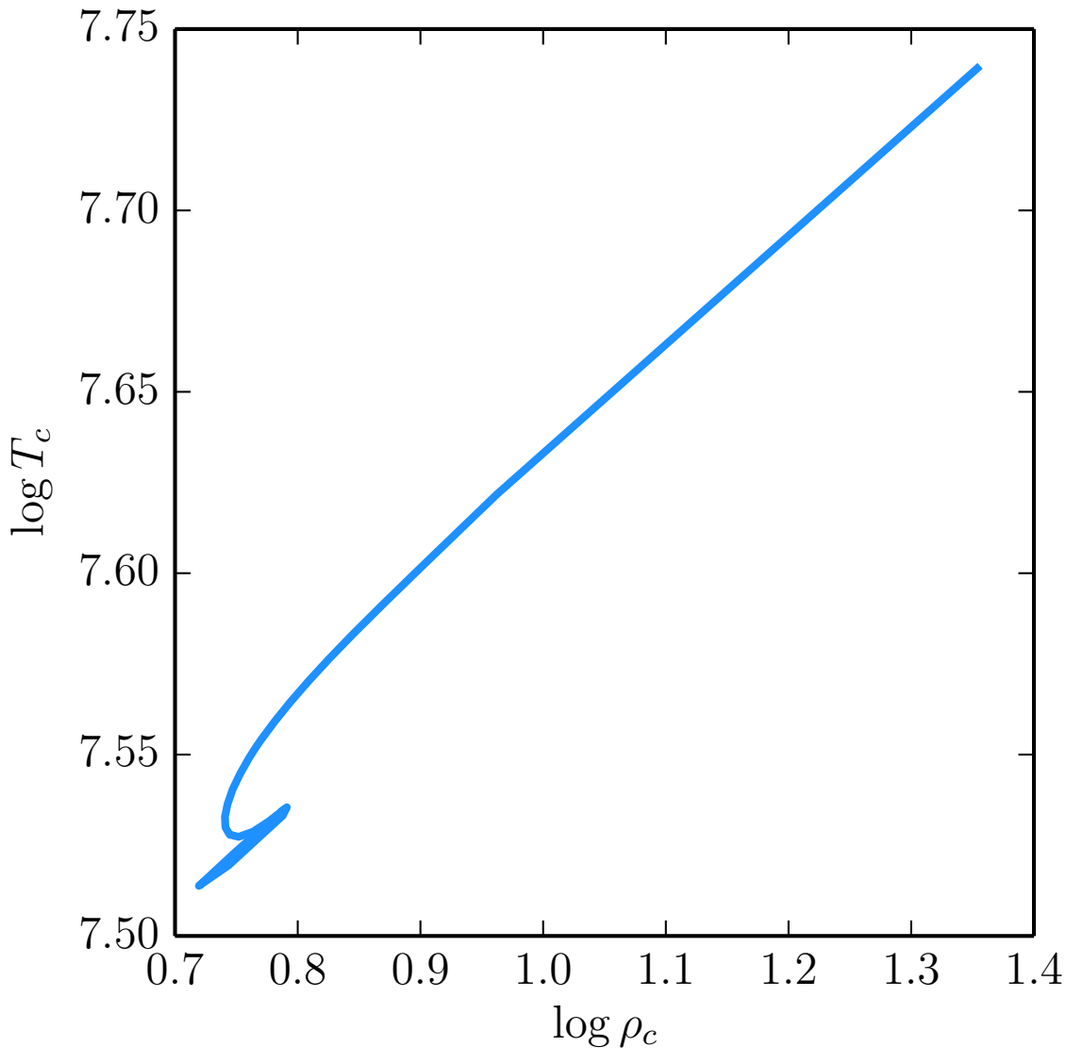
- MesaLogDir: Deals with entire LOGS directory

- MesaProfileIndex: Deals with linking profile numbers to model numbers through profiles.index

- MesaData: Interface with data in profile or history file

ipython qtconsole --pylab

```
In [1]: from mesa_reader import *

In [2]: l = MesaLogDir()

In [3]: h = l.history

In [4]: plot(h.log_center_Rho, h.log_center_T)
Out[4]: [<matplotlib.lines.Line2D at 0x1119a9c50>]

In [5]: xlabel(r'$\log \rho_c$'); ylabel(r'$\log T_c$')
Out[5]: <matplotlib.text.Text at 0x10b0af250>

In [6]: savefig('tc_rhoc.pdf')
```



MesaData Tidbits

- Get at numpy arrays of interest with `h.data('star_age')` or `h.star_age` (the latter reduces to the former)

- Get at header data with `h.header('version_number')` or `h.version_number` (again, latter -> former)

- If source has a model_number entry, restarts/backups are automatically removed.

- Get available categories with `h.header_names` and `h.bulk_names`

Masks

```
>>> good_indices = h.log_Teff > 3.5 & \
h.log_Teff < 4.0 & h.log_L > 1.0 & h.log_L < 2.0
```

```
>>>plot(h.log_Teff[good_indices],
       h.log_L[good_indices])
```

Should only plot hr points in a given box

MesaLogDir Tidbits

- Main feature is to get profile data from `l.profile_data(model_number, profile_number)`

- With no argument, gives last profile saved, otherwise uses provided model number (if given) or profile number

- Nifty feature: `l.select_models()`:
  takes in a function and some history names, returns model numbers (that have profiles) where that function returns true

Select_models example

```
l = MesaLogDir('./LOGS')

# function to determine if given Teff, L are in HR box
def in_box(log_Teff, log_L):
    return (log_Teff > 3.0 and log_Teff < 4.0 and log_L > 1.0 and log_L < 2.0)

# models (with profiles) where
box_model_nums = l.select_models(in_box, "log_Teff", "log_L")

# first model (in time) in the box
p_first = l.profile_data(box_model_nums[0])
```

Main thing to remember is that arguments of function and supplied keys must be same length and correspond (and be in the history file).

Learn More!

- Get it here: www.github.com/wmwolf/py_mesa_reader

- Documentation here: wmwolf.github.io/py_mesa_reader

- Installation instructions nonexistent… put in your local folder (i.e. work directory) or put somewhere on your PYTHONPATH.

- But reports and suggestions welcome!

Special Thanks

- Josiah Schwab, tester and patcher

- Friendly people who have said nice things about MesaScript