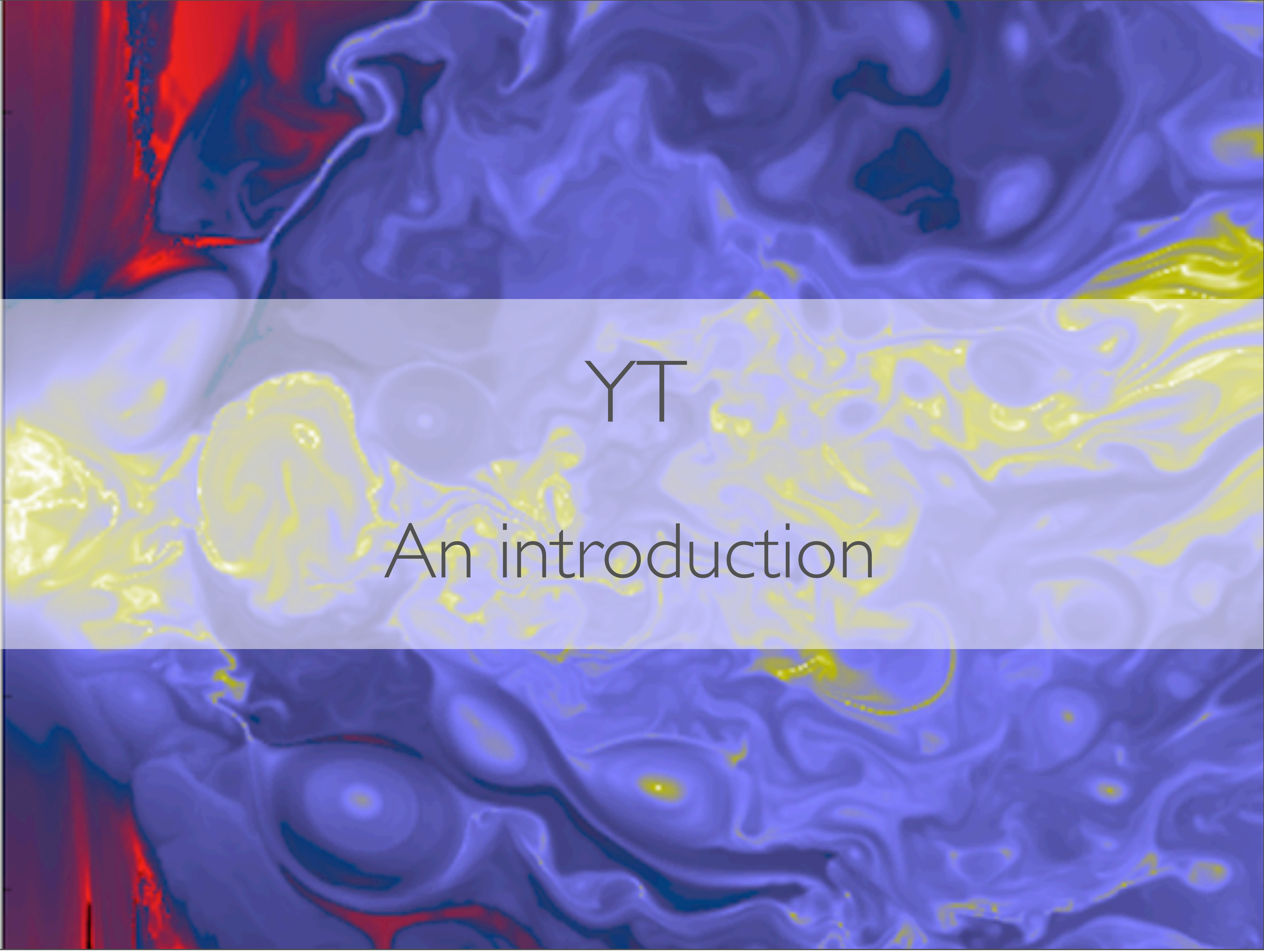


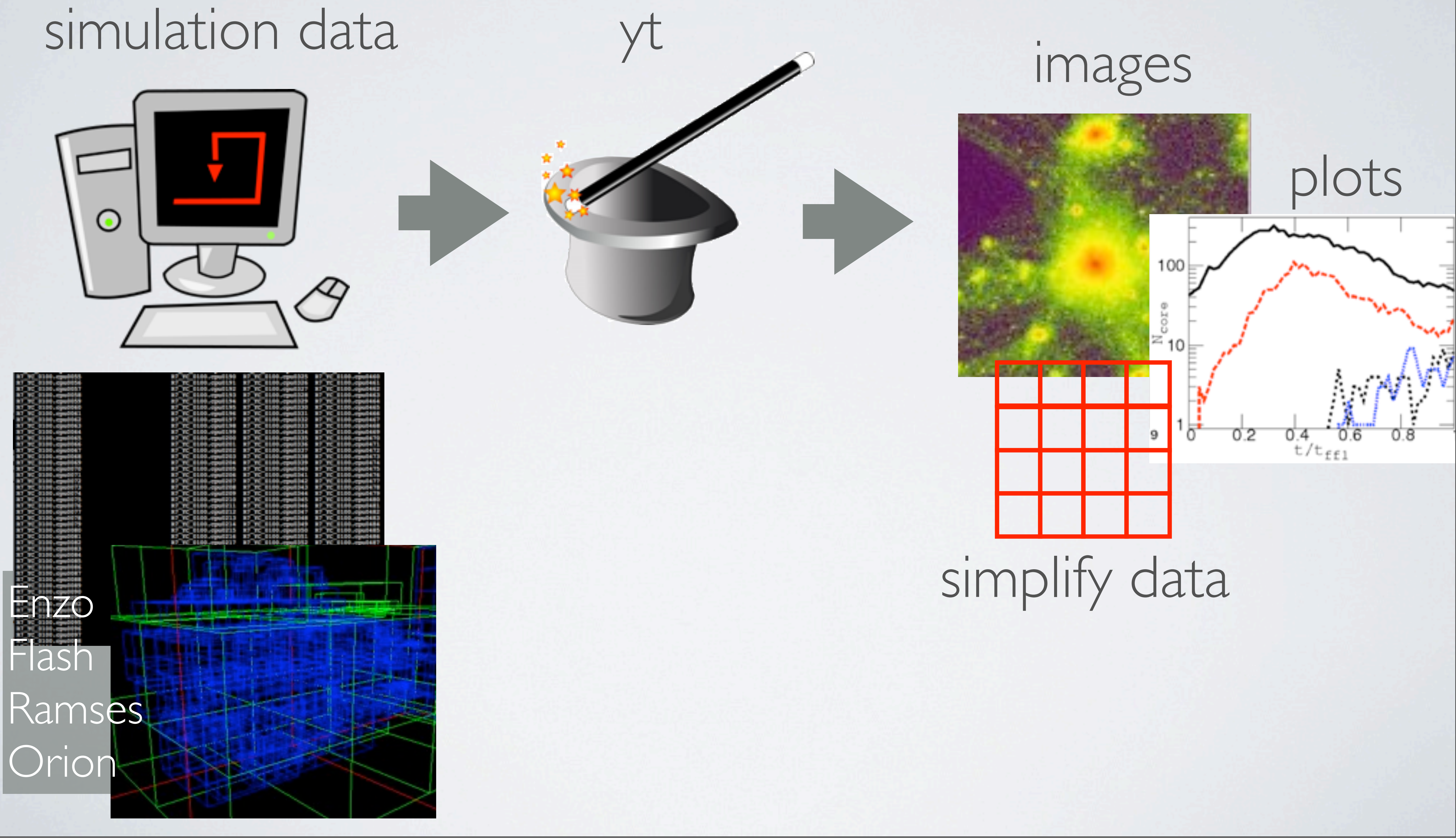
YT

An introduction

Contents

- What is **yt**?
- Installing **yt**
- Using **yt**
  - `> yt from the command line`
  - **yt** with the iPython notebook
  - scripting **yt**
- **yt**'s Cookbook

What is **yt**?



simulation data

yt

images

plots

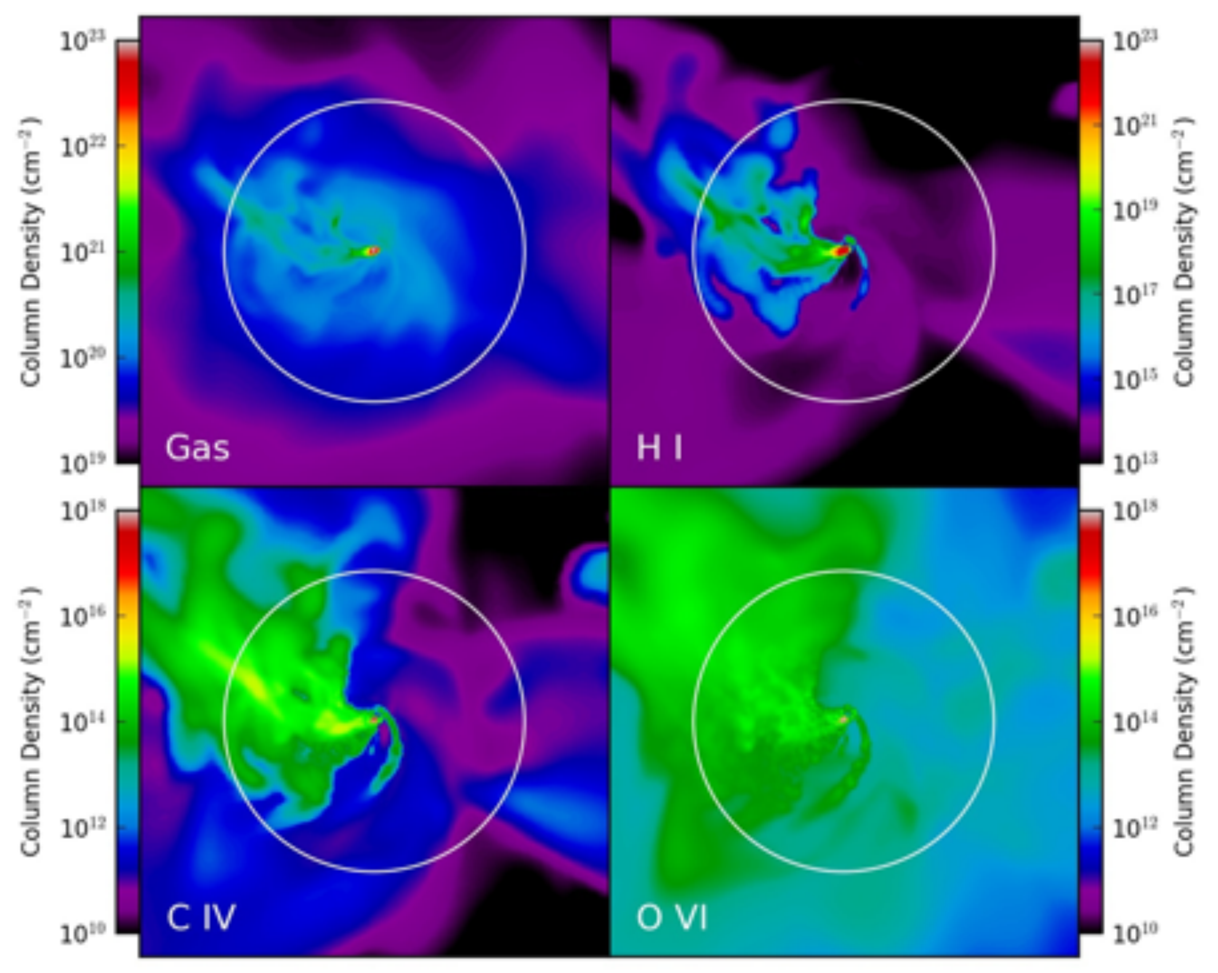
simplify data

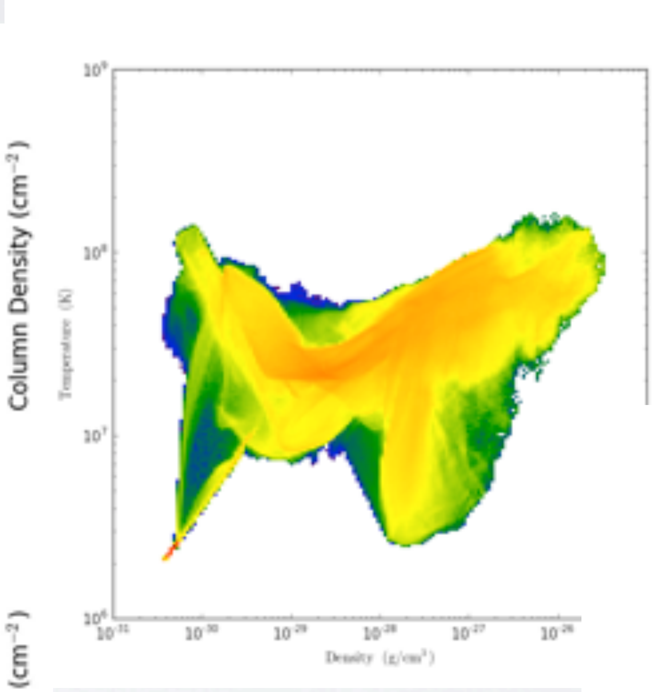
Enzo
Flash
Ramses
Orion

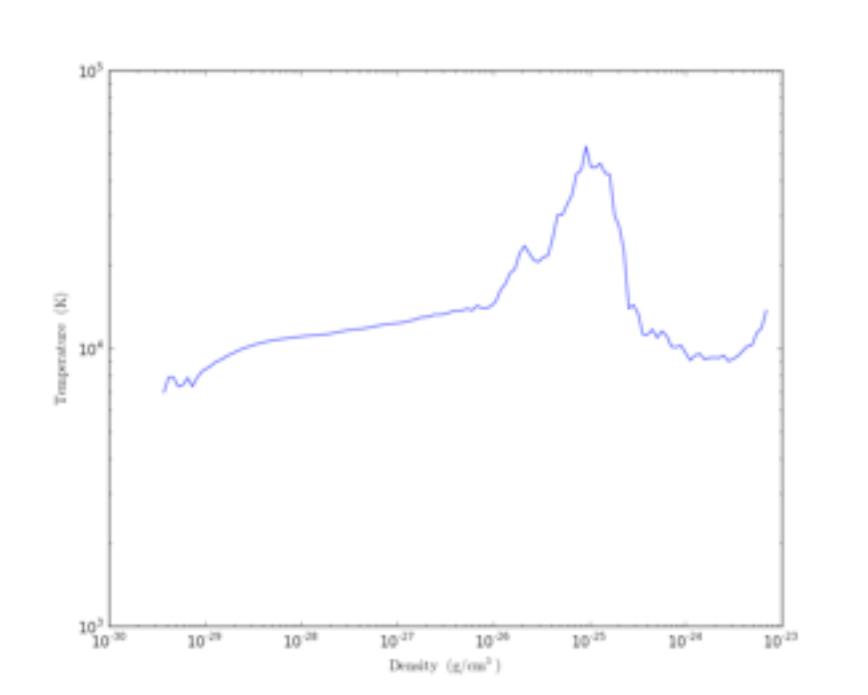
What is **yt**?

Analysis basics:

(Plots you always need to create)







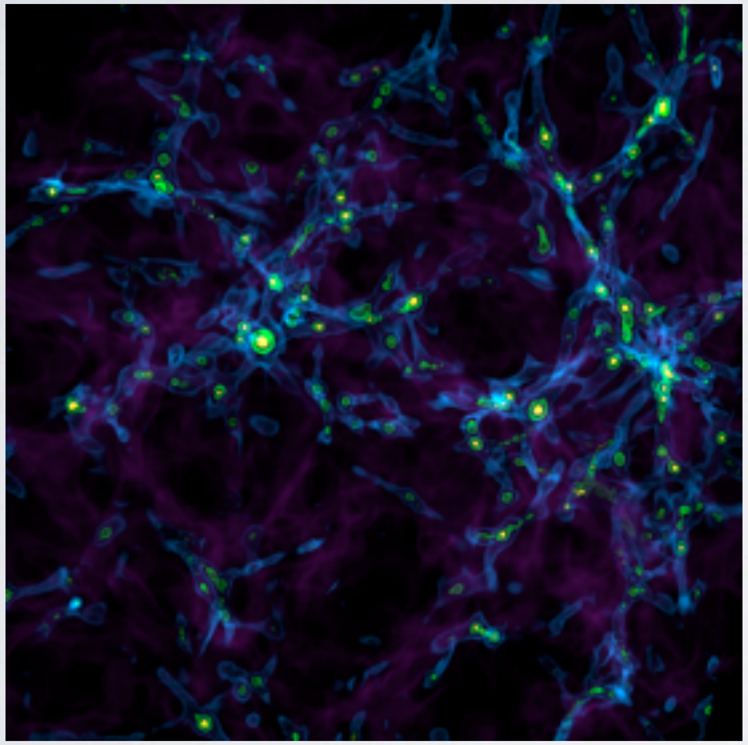
Very easy to make

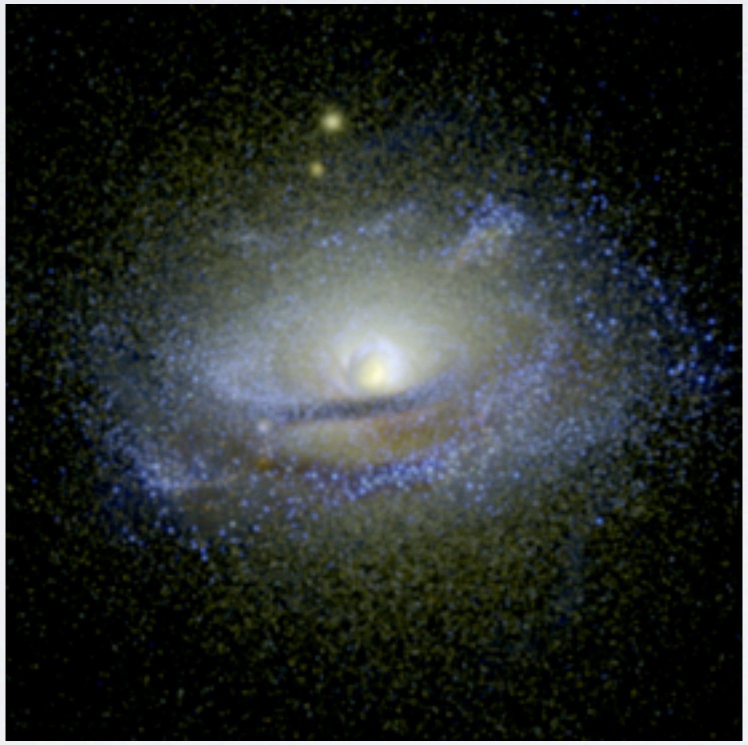
Slices, projections, 2D plots, 1D profiles....

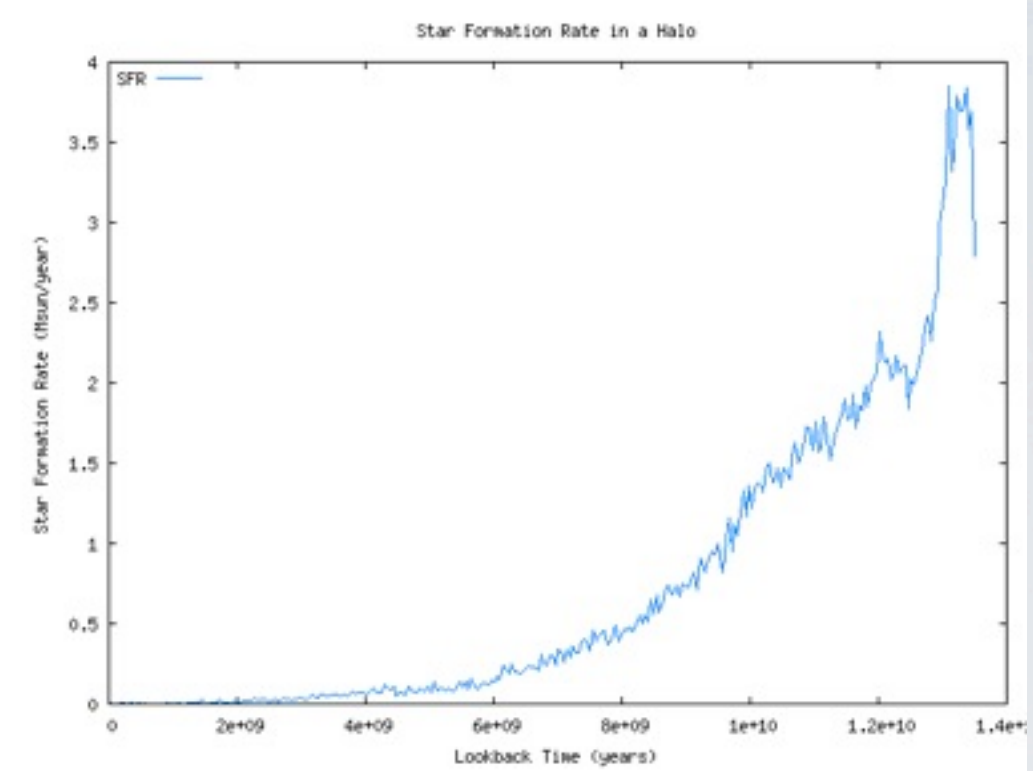
What is **yt**?

Advanced tools:

(Complicated analysis in easy-to-use routines)







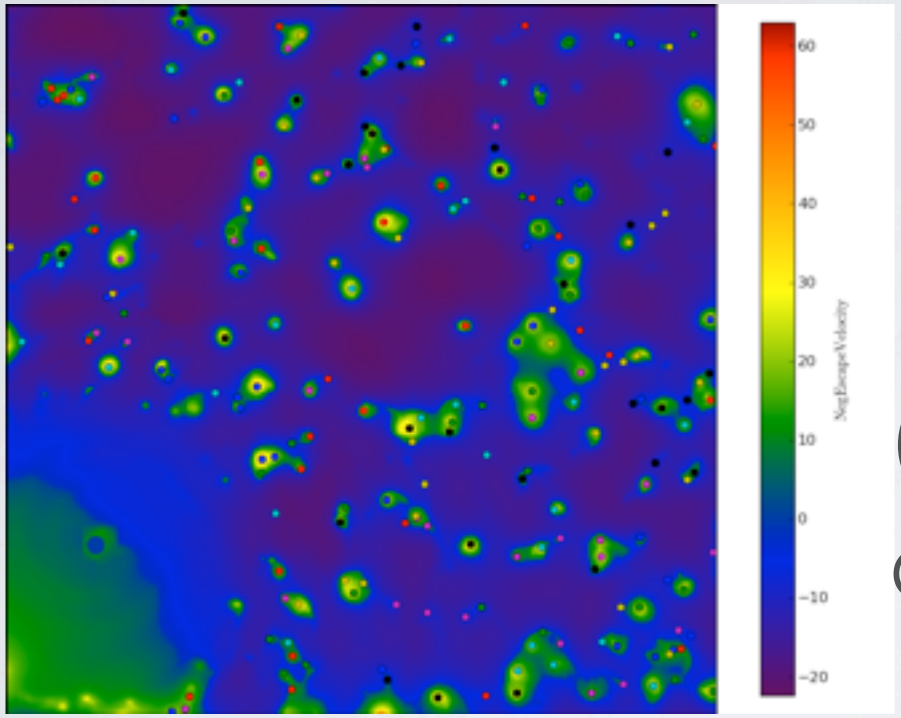
e.g. Dark matter halo finder and gas clump finder,

'Synthetic observations' with Sunrise (radiative transfer)

Calculate star formation rates in any region

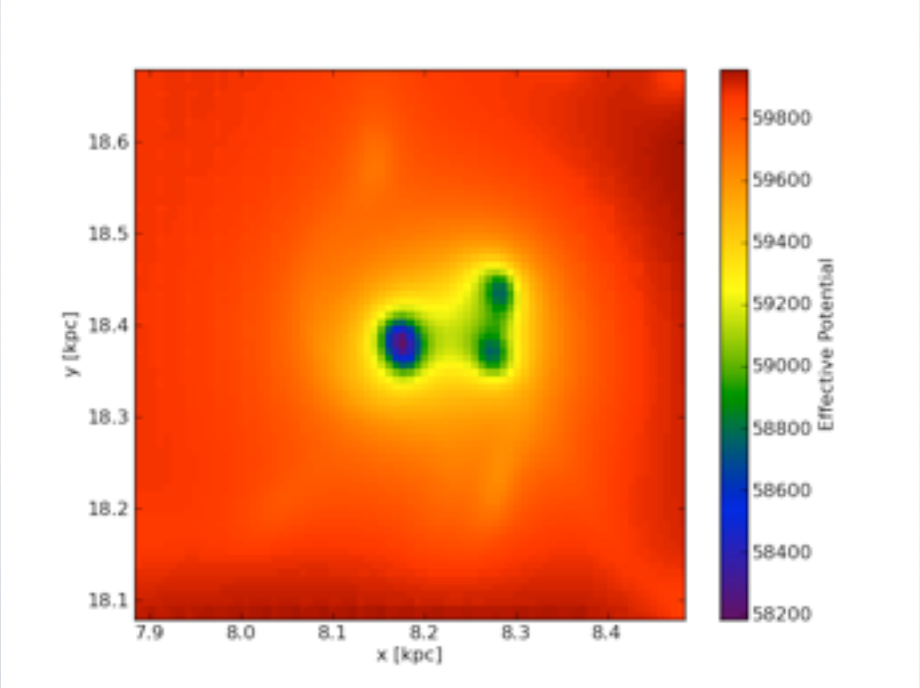
What is **yt**?

Use as part of your own analysis programmes

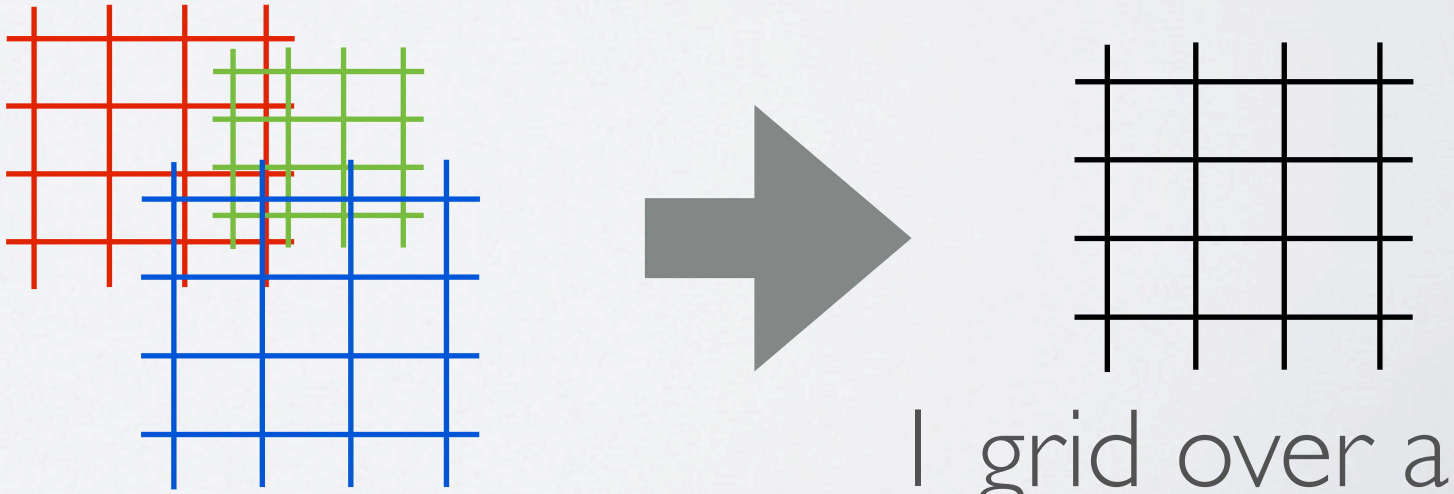


e.g. easily view new properties in images, plots etc

(escape velocity, density^2, mass x time, dinosaurs/cm^3 ...)



e.g. make data simple

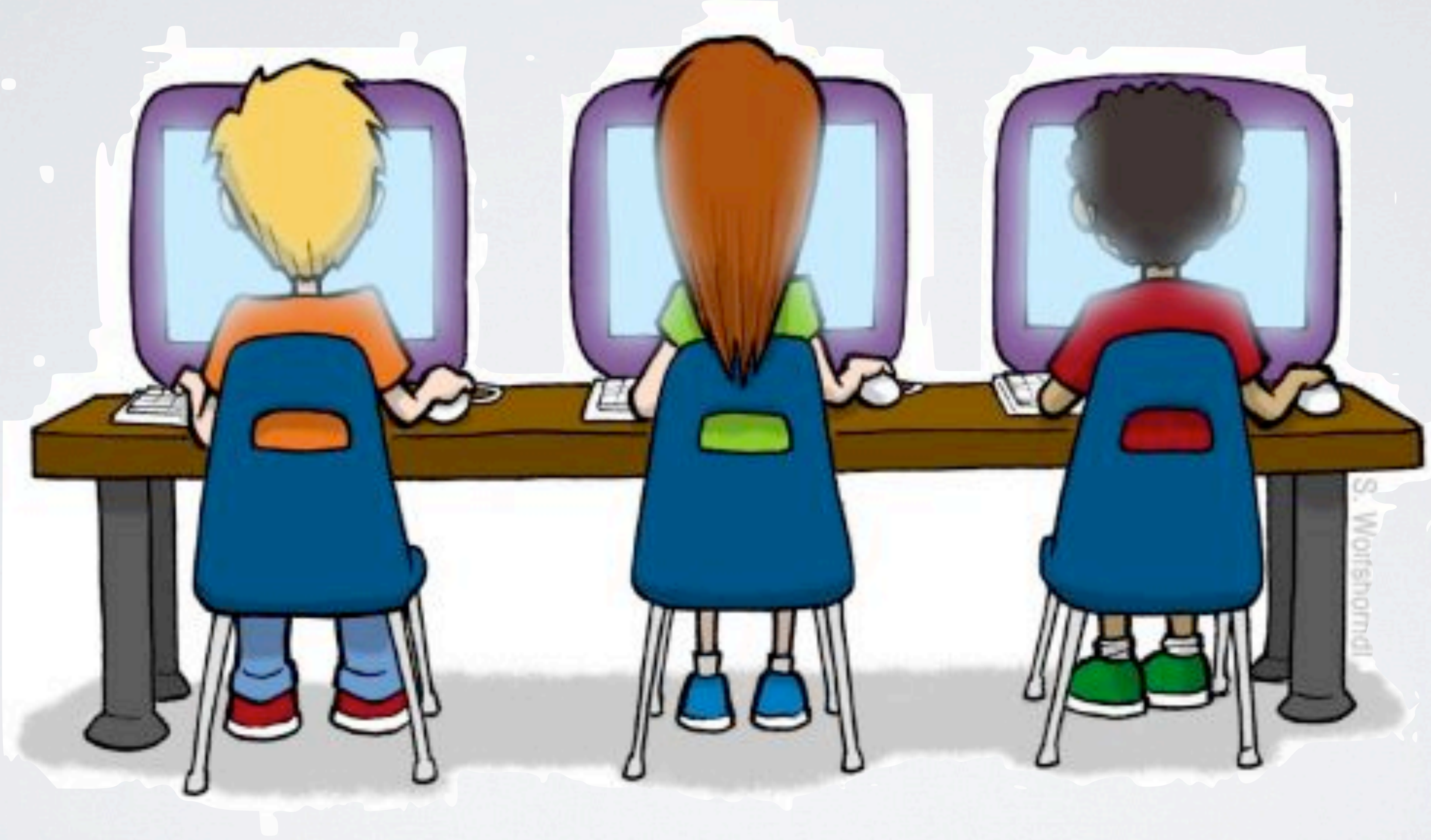


1000s of grids...

1 grid over any volume

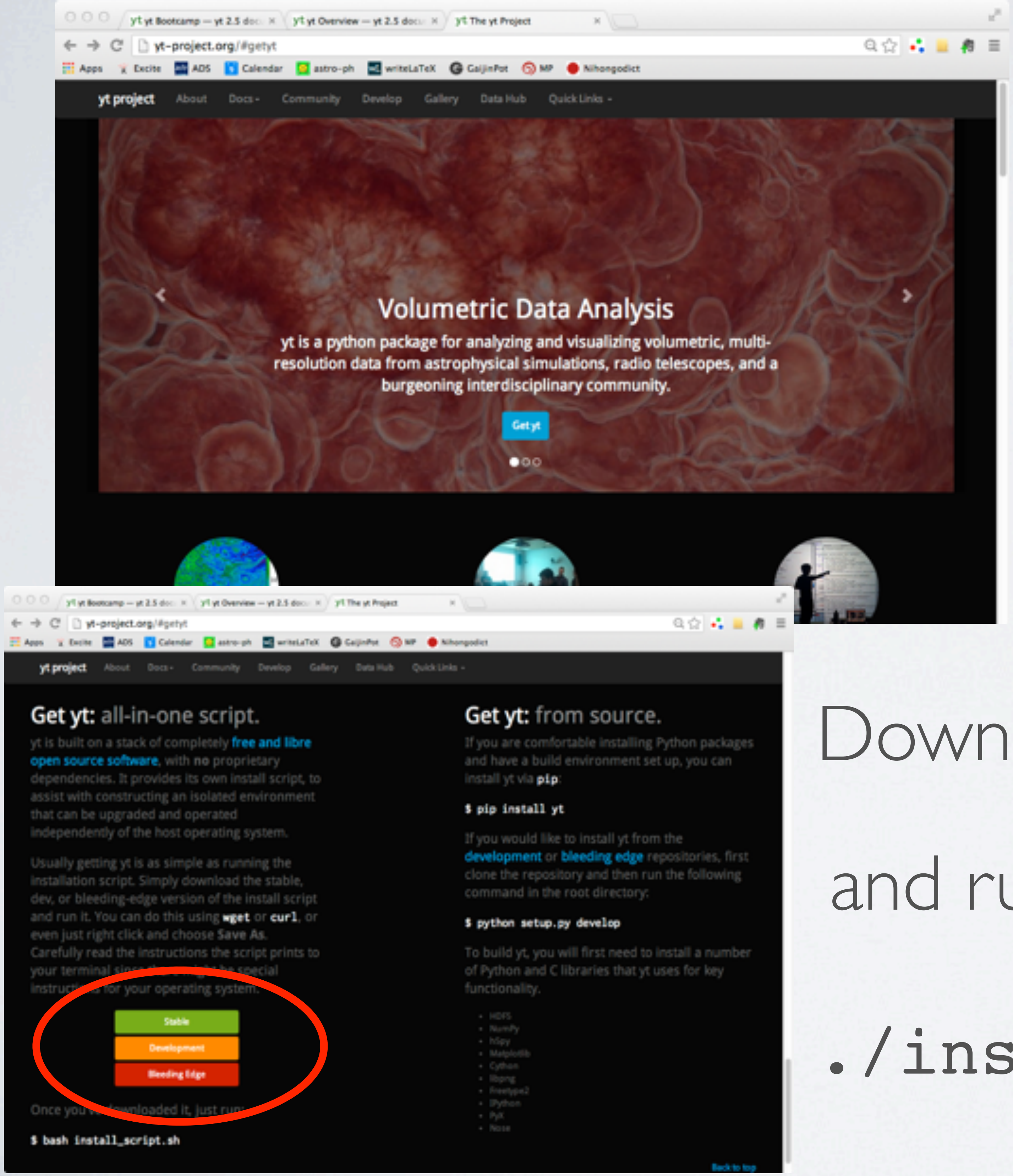
Installing **yt**

Let’s do this together....



Can everyone connect to the WWW?

Installing **yt**



Volumetric Data Analysis

yt is a python package for analyzing and visualizing volumetric, multi-resolution data from astrophysical simulations, radio telescopes, and a burgeoning interdisciplinary community.

Get yt: all-in-one script.

yt is built on a stack of completely free and libre open source software, with **no** proprietary dependencies. It provides its own install script, to assist with constructing an isolated environment that can be upgraded and operated independently of the host operating system.

Usually getting yt is as simple as running the installation script. Simply download the stable, dev, or bleeding-edge version of the install script and run it. You can do this using `wget` or `curl`, or even just right click and choose Save As. Carefully read the instructions the script prints to your terminal since there might be special instructions for your operating system.

Once you've downloaded it, just run:

```
$ bash install_script.sh
```

Get yt: from source.

If you are comfortable installing Python packages and have a build environment set up, you can install yt via `pip`:

```
$ pip install yt
```

If you would like to install yt from the development or bleeding edge repositories, first clone the repository and then run the following command in the root directory:

```
$ python setup.py develop
```

To build yt, you will first need to install a number of Python and C libraries that yt uses for key functionality.

- HDF5
- NumPy
- h5py
- Matplotlib
- Cython
- libpng
- Freetype2
- IPython
- PyX
- Nose

yt webpage:

http://yt-project.org

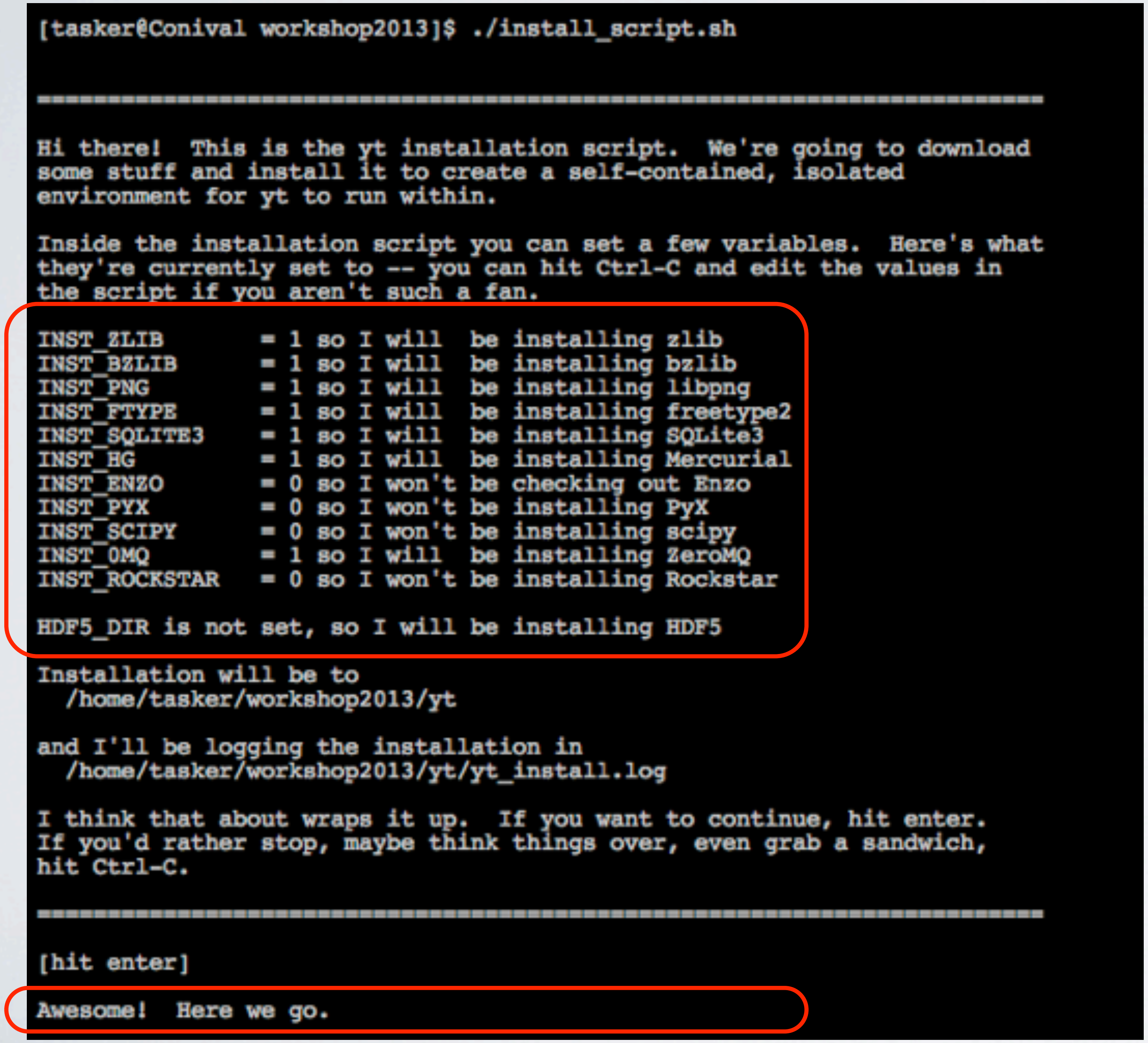
Download installation script

and run...

```
./install_script.sh
```

Installing **yt**

install_script.sh



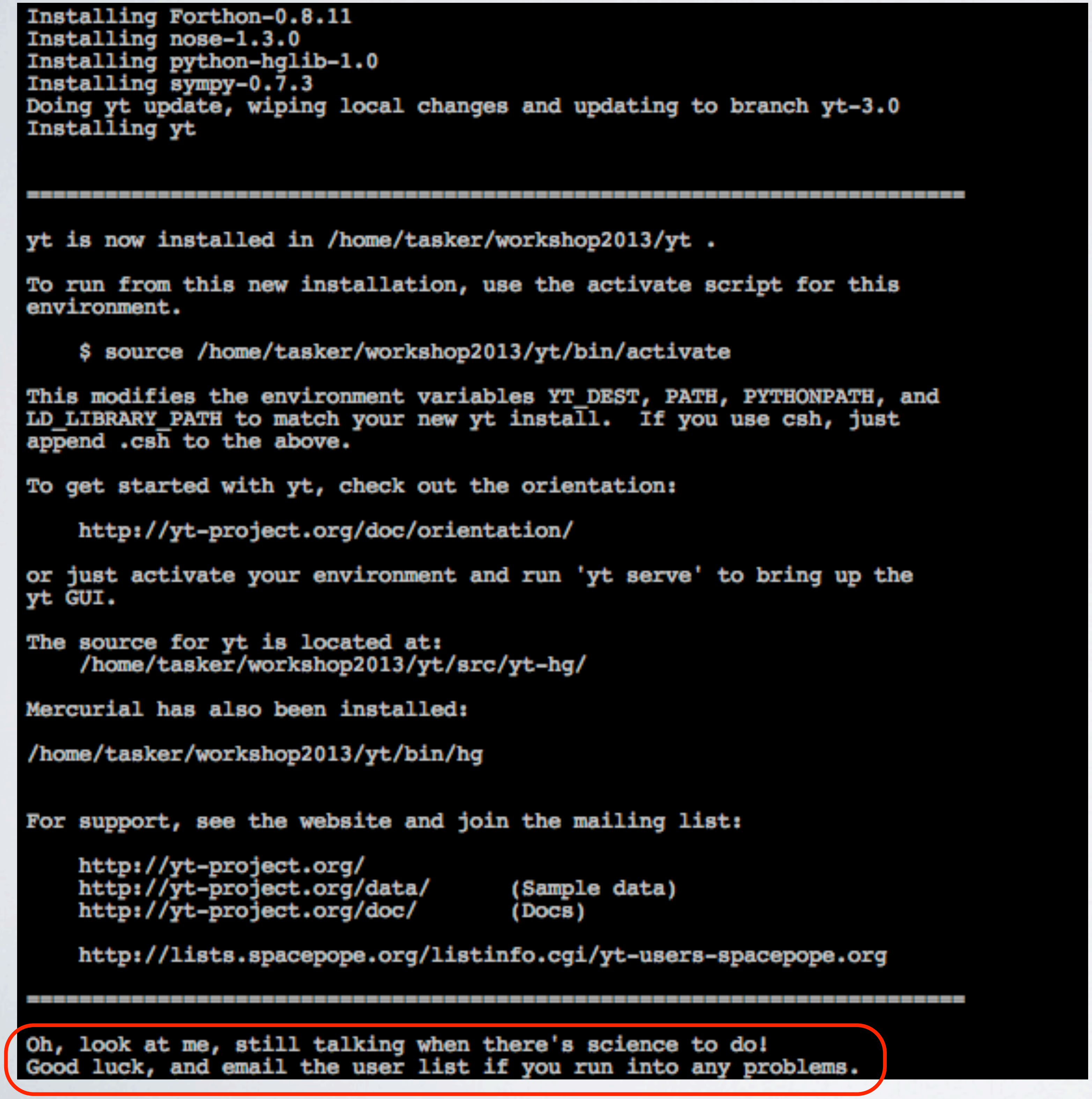
```
[tasker@Conival workshop2013]$ ./install_script.sh


========================================================================

Hi there!  This is the yt installation script.  We're going to download
some stuff and install it to create a self-contained, isolated
environment for yt to run within.

Inside the installation script you can set a few variables.  Here's what
they're currently set to -- you can hit Ctrl-C and edit the values in
the script if you aren't such a fan.

INST_ZLIB       = 1 so I will  be installing zlib
INST_BZLIB      = 1 so I will  be installing bzlib
INST_PNG        = 1 so I will  be installing libpng
INST_FTYPE      = 1 so I will  be installing freetype2
INST_SQLITE3    = 1 so I will  be installing SQLite3
INST_HG         = 1 so I will  be installing Mercurial
INST_ENZO       = 0 so I won't be checking out Enzo
INST_PYX        = 0 so I won't be installing PyX
INST_SCIPY      = 0 so I won't be installing scipy
INST_0MQ        = 1 so I will  be installing ZeroMQ
INST_ROCKSTAR   = 0 so I won't be installing Rockstar

HDF5_DIR is not set, so I will be installing HDF5

Installation will be to
  /home/tasker/workshop2013/yt

and I'll be logging the installation in
  /home/tasker/workshop2013/yt/yt_install.log

I think that about wraps it up.  If you want to continue, hit enter.
If you'd rather stop, maybe think things over, even grab a sandwich,
hit Ctrl-C.

========================================================================

[hit enter]

Awesome!  Here we go.
```

installs all necessary packages

very friendly!

Installing **yt**

install_script.sh

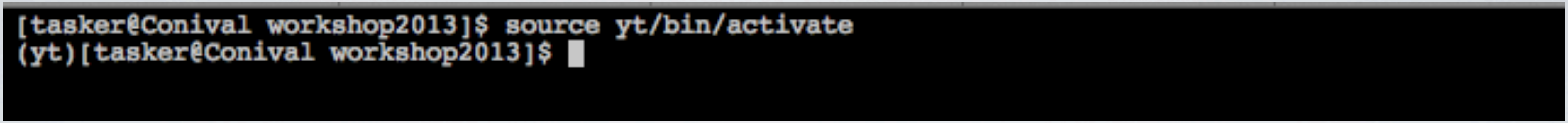


```
Installing Forthon-0.8.11
Installing nose-1.3.0
Installing python-hglib-1.0
Installing sympy-0.7.3
Doing yt update, wiping local changes and updating to branch yt-3.0
Installing yt


===============================================================================


yt is now installed in /home/tasker/workshop2013/yt .

To run from this new installation, use the activate script for this
environment.

    $ source /home/tasker/workshop2013/yt/bin/activate

This modifies the environment variables YT_DEST, PATH, PYTHONPATH, and
LD_LIBRARY_PATH to match your new yt install.  If you use csh, just
append .csh to the above.

To get started with yt, check out the orientation:

    http://yt-project.org/doc/orientation/

or just activate your environment and run 'yt serve' to bring up the
yt GUI.

The source for yt is located at:
    /home/tasker/workshop2013/yt/src/yt-hg/

Mercurial has also been installed:

/home/tasker/workshop2013/yt/bin/hg


For support, see the website and join the mailing list:

    http://yt-project.org/
    http://yt-project.org/data/      (Sample data)
    http://yt-project.org/doc/       (Docs)

    http://lists.spacepope.org/listinfo.cgi/yt-users-spacepope.org

===============================================================================

Oh, look at me, still talking when there's science to do!
Good luck, and email the user list if you run into any problems.
```

time

Finished!

Installing **yt**



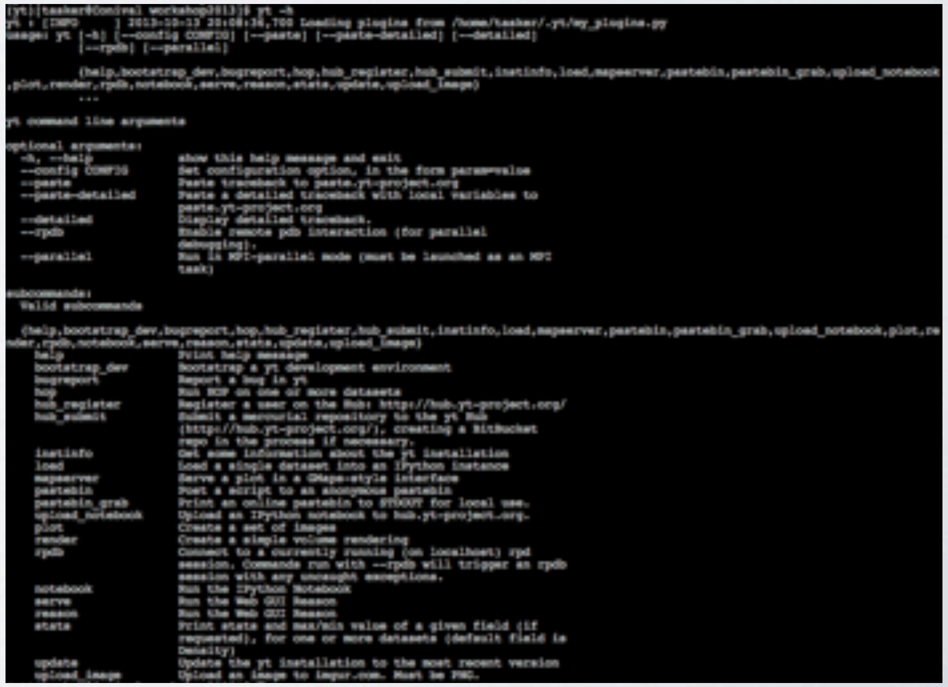
```
[tasker@Conival workshop2013]$ source yt/bin/activate
(yt)[tasker@Conival workshop2013]$ █
```

```
> source yt-x86_64/bin/activate
```

path to yt

```
> yt -h
```

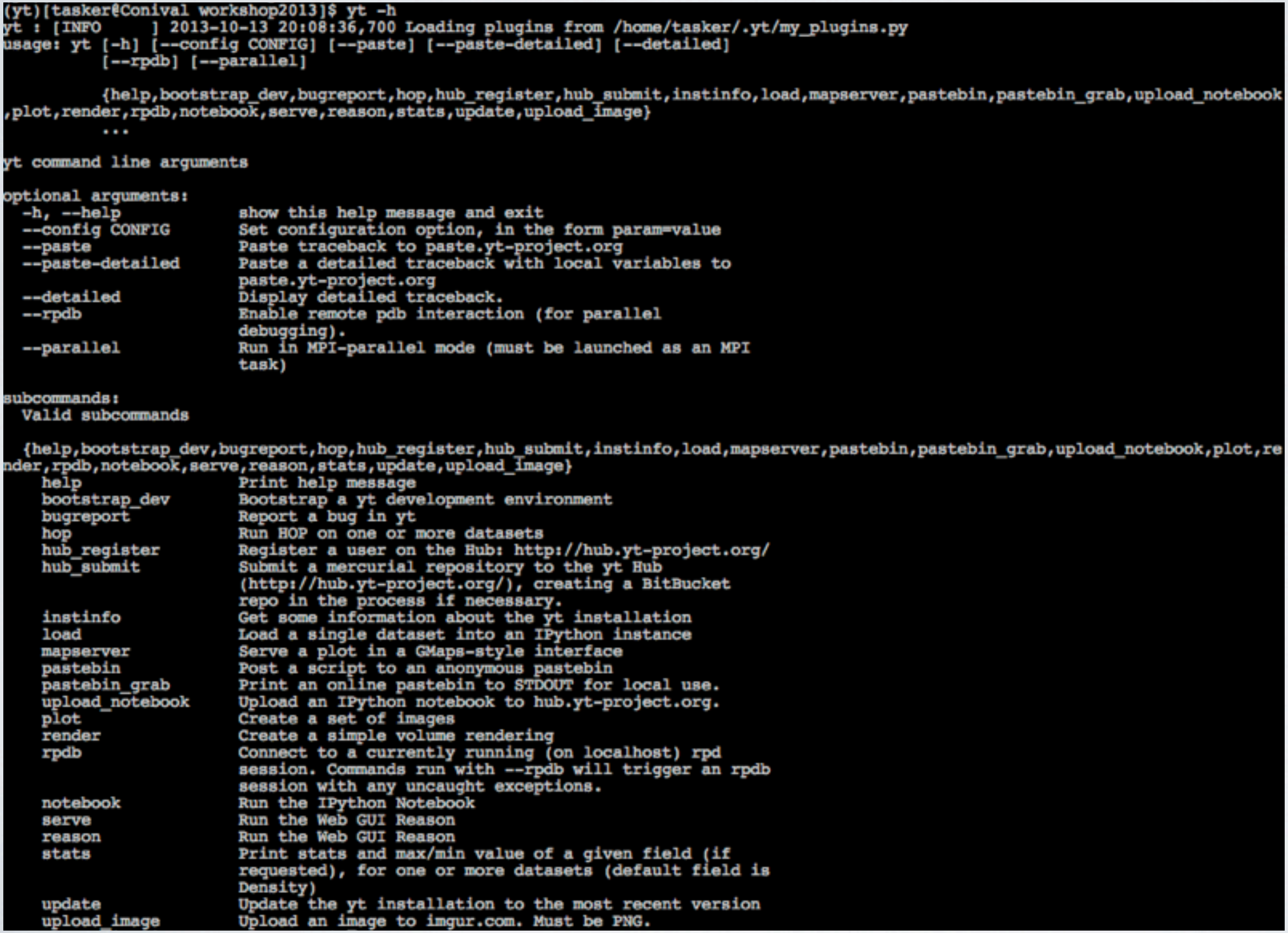
yt command line options



```
(yt)[tasker@Conival workshop2013]$ yt -h
yt : [INFO     ] 2013-10-13 20:08:36,700 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt [-h] [--config CONFIG] [--paste] [--paste-detailed] [--detailed]
          [--rpdb] [--parallel]

          {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook
,plot,render,rpdb,notebook,serve,reason,stats,update,upload_image}
          ...

yt command line arguments

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Set configuration option, in the form param=value
  --paste               Paste traceback to paste.yt-project.org
  --paste-detailed      Paste a detailed traceback with local variables to
                        paste.yt-project.org
  --detailed            Display detailed traceback.
  --rpdb                Enable remote pdb interaction (for parallel
                        debugging).
  --parallel            Run in MPI-parallel mode (must be launched as an MPI
                        task)

subcommands:
  Valid subcommands

  {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook,plot,re
nder,rpdb,notebook,serve,reason,stats,update,upload_image}
    help                Print help message
    bootstrap_dev       Bootstrap a yt development environment
    bugreport           Report a bug in yt
    hop                 Run HOP on one or more datasets
    hub_register        Register a user on the Hub: http://hub.yt-project.org/
    hub_submit          Submit a mercurial repository to the yt Hub
                        (https://hub.yt-project.org/), creating a BitBucket
                        repo in the process if necessary.
    instinfo            Get some information about the yt installation
    load                Load a single dataset into an IPython instance
    mapserver           Serve a plot in a GMaps-style interface
    pastebin            Post a script to an anonymous pastebin
    pastebin_grab       Print an online pastebin to STDOUT for local use.
    upload_notebook     Upload an IPython notebook to hub.yt-project.org.
    plot                Create a set of images
    render              Create a simple volume rendering
    rpdb                Connect to a currently running (on localhost) rpd
                        session. Commands run with --rpdb will trigger an rpdb
                        session with any uncaught exceptions.
    notebook            Run the IPython Notebook
    serve               Run the Web GUI Reason
    reason              Run the Web GUI Reason
    stats               Print stats and max/min value of a given field (if
                        requested), for one or more datasets (default field is
                        Density)
    update              Update the yt installation to the most recent version
    upload_image        Upload an image to imgur.com. Must be PNG.
```

Command line **yt**

Quickest way to use **yt**

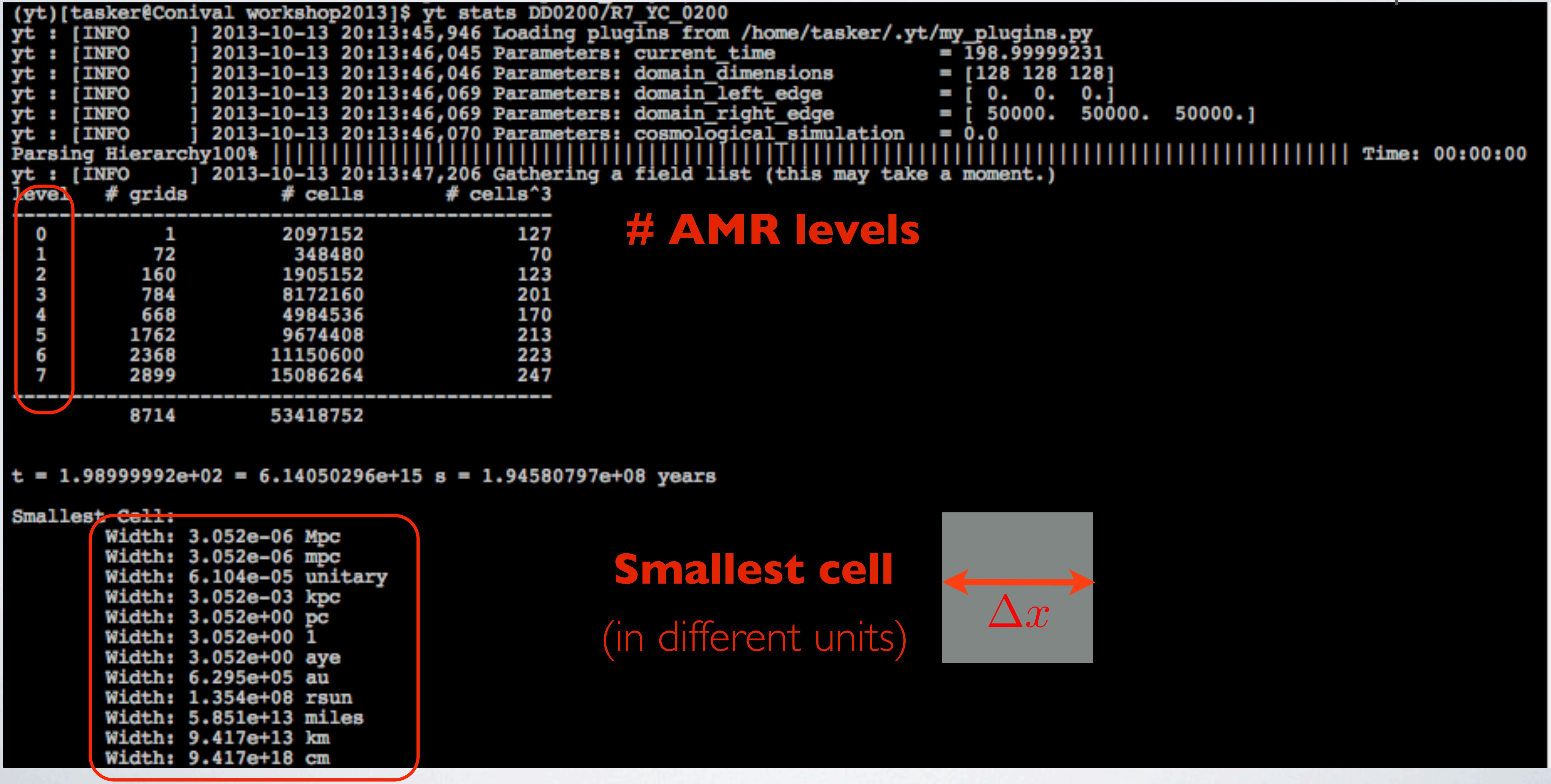


```
(yt)[tasker@Conival workshop2013]$ yt -h
yt : [INFO     ] 2013-10-13 20:08:36,700 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt [-h] [--config CONFIG] [--paste] [--paste-detailed] [--detailed]
          [--rpdb] [--parallel]

          {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook
,plot,render,rpdb,notebook,serve,reason,stats,update,upload_image}
          ...

yt command line arguments

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       Set configuration option, in the form param=value
  --paste               Paste traceback to paste.yt-project.org
  --paste-detailed      Paste a detailed traceback with local variables to
                        paste.yt-project.org
  --detailed            Display detailed traceback.
  --rpdb                Enable remote pdb interaction (for parallel
                        debugging).
  --parallel            Run in MPI-parallel mode (must be launched as an MPI
                        task)

subcommands:
  Valid subcommands

  {help,bootstrap_dev,bugreport,hop,hub_register,hub_submit,instinfo,load,mapserver,pastebin,pastebin_grab,upload_notebook,plot,re
nder,rpdb,notebook,serve,reason,stats,update,upload_image}
    help                Print help message
    bootstrap_dev       Bootstrap a yt development environment
    bugreport           Report a bug in yt
    hop                 Run HOP on one or more datasets
    hub_register        Register a user on the Hub: http://hub.yt-project.org/
    hub_submit          Submit a mercurial repository to the yt Hub
                        (http://hub.yt-project.org/), creating a BitBucket
                        repo in the process if necessary.
    instinfo            Get some information about the yt installation
    load                Load a single dataset into an IPython instance
    mapserver           Serve a plot in a GMaps-style interface
    pastebin            Post a script to an anonymous pastebin
    pastebin_grab       Print an online pastebin to STDOUT for local use.
    upload_notebook     Upload an IPython notebook to hub.yt-project.org.
    plot                Create a set of images
    render              Create a simple volume rendering
    rpdb                Connect to a currently running (on localhost) rpd
                        session. Commands run with --rpdb will trigger an rpdb
                        session with any uncaught exceptions.
    notebook            Run the IPython Notebook
    serve               Run the Web GUI Reason
    reason              Run the Web GUI Reason
    stats               Print stats and max/min value of a given field (if
                        requested), for one or more datasets (default field is
                        Density)
    update              Update the yt installation to the most recent version
    upload_image        Upload an image to imgur.com. Must be PNG.
```

Command line **yt**

```
(yt)> cd workshop

(yt)> yt stats M83/DD0200/R7_YC_0200
```

Enzo data output



```
(yt)[tasker@Conival workshop2013]$ yt stats DD0200/R7_YC_0200
yt : [INFO     ] 2013-10-13 20:13:45,946 Loading plugins from /home/tasker/.yt/my_plugins.py
yt : [INFO     ] 2013-10-13 20:13:46,045 Parameters: current_time              = 198.99999231
yt : [INFO     ] 2013-10-13 20:13:46,046 Parameters: domain_dimensions         = [128 128 128]
yt : [INFO     ] 2013-10-13 20:13:46,069 Parameters: domain_left_edge          = [ 0.   0.   0.]
yt : [INFO     ] 2013-10-13 20:13:46,069 Parameters: domain_right_edge         = [ 50000.  50000.  50000.]
yt : [INFO     ] 2013-10-13 20:13:46,070 Parameters: cosmological_simulation   = 0.0
Parsing Hierarchy100% ||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| Time: 00:00:00
yt : [INFO     ] 2013-10-13 20:13:47,206 Gathering a field list (this may take a moment.)
level    # grids         # cells        # cells^3
----------------------------------------------
  0          1          2097152           127
  1         72           348480            70
  2        160          1905152           123
  3        784          8172160           201
  4        668          4984536           170
  5       1762          9674408           213
  6       2368         11150600           223
  7       2899         15086264           247
----------------------------------------------
          8714         53418752


t = 1.98999992e+02 = 6.14050296e+15 s = 1.94580797e+08 years

Smallest Cell:
        Width: 3.052e-06 Mpc
        Width: 3.052e-06 mpc
        Width: 6.104e-05 unitary
        Width: 3.052e-03 kpc
        Width: 3.052e+00 pc
        Width: 3.052e+00 1
        Width: 3.052e+00 aye
        Width: 6.295e+05 au
        Width: 1.354e+08 rsun
        Width: 5.851e+13 miles
        Width: 9.417e+13 km
        Width: 9.417e+18 cm
```

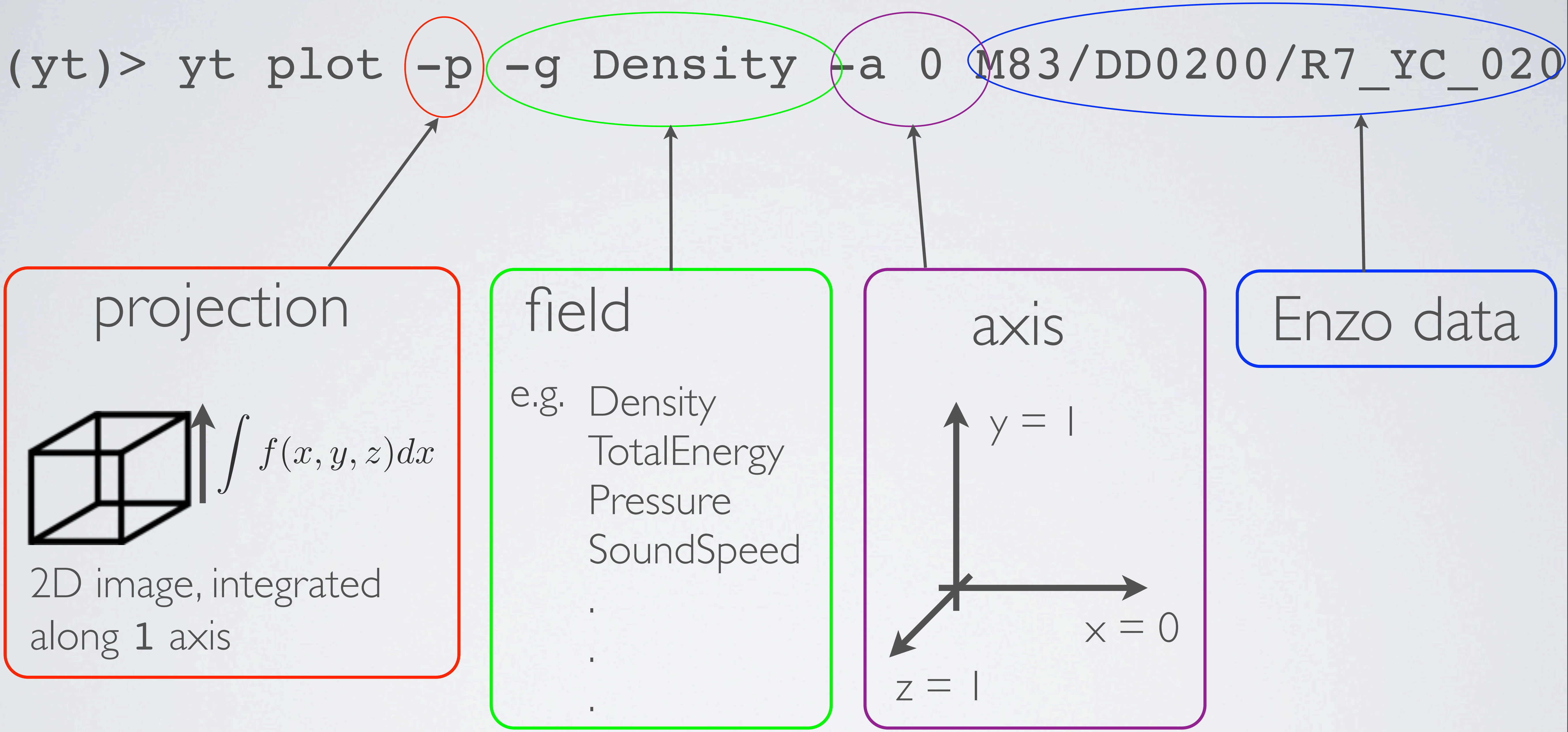
# AMR levels

Smallest cell

(in different units)

Command line **yt**

```
(yt)> yt plot -p -g Density -a 0 M83/DD0200/R7_YC_020
```



projection

$\int f(x,y,z)dx$

2D image, integrated along 1 axis

field

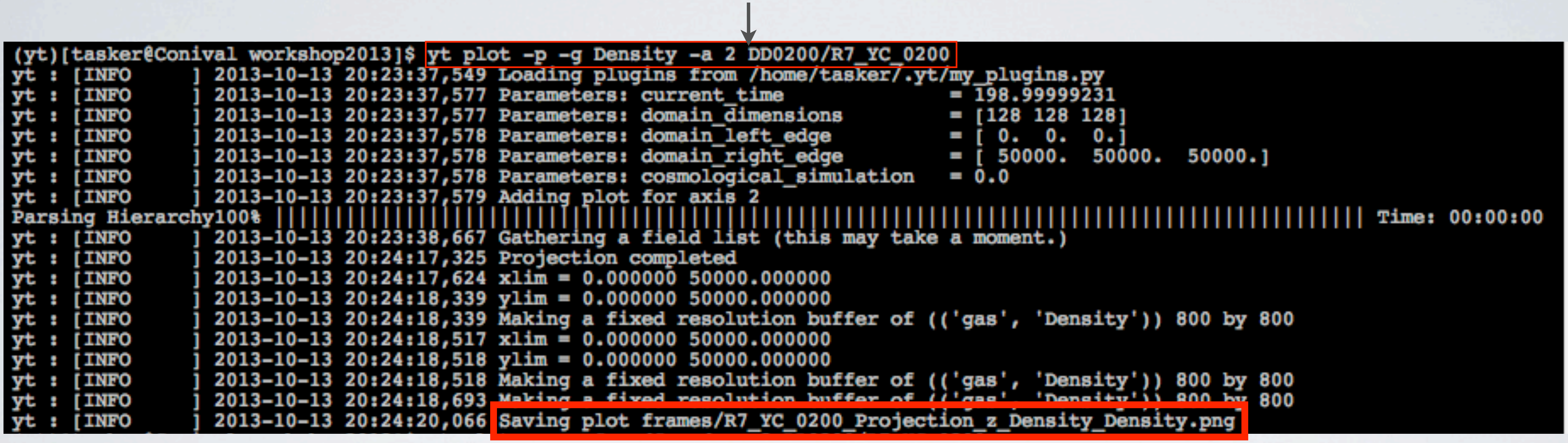
e.g. Density
TotalEnergy
Pressure
SoundSpeed
.
.
.

axis

Enzo data

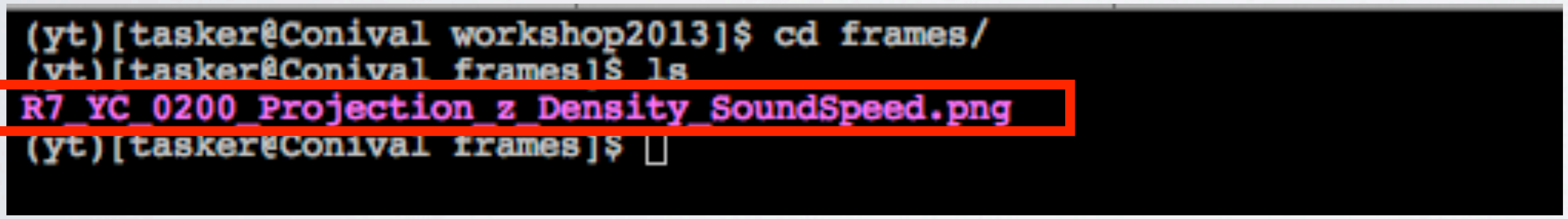
Command line **yt**

```
(yt)> yt plot -p -g Density -a 2 M83/DD0200/R7_YC_0200
```



```
(yt)[tasker@Conival workshop2013]$ yt plot -p -g Density -a 2 DD0200/R7_YC_0200
yt : [INFO     ] 2013-10-13 20:23:37,549 Loading plugins from /home/tasker/.yt/my_plugins.py
yt : [INFO     ] 2013-10-13 20:23:37,577 Parameters: current_time              = 198.99999231
yt : [INFO     ] 2013-10-13 20:23:37,577 Parameters: domain_dimensions         = [128 128 128]
yt : [INFO     ] 2013-10-13 20:23:37,578 Parameters: domain_left_edge          = [ 0.   0.   0.]
yt : [INFO     ] 2013-10-13 20:23:37,578 Parameters: domain_right_edge         = [ 50000.  50000.  50000.]
yt : [INFO     ] 2013-10-13 20:23:37,578 Parameters: cosmological_simulation   = 0.0
yt : [INFO     ] 2013-10-13 20:23:37,579 Adding plot for axis 2
Parsing Hierarchy100% |||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||| Time: 00:00:00
yt : [INFO     ] 2013-10-13 20:23:38,667 Gathering a field list (this may take a moment.)
yt : [INFO     ] 2013-10-13 20:24:17,325 Projection completed
yt : [INFO     ] 2013-10-13 20:24:17,624 xlim = 0.000000 50000.000000
yt : [INFO     ] 2013-10-13 20:24:18,339 ylim = 0.000000 50000.000000
yt : [INFO     ] 2013-10-13 20:24:18,339 Making a fixed resolution buffer of (('gas', 'Density')) 800 by 800
yt : [INFO     ] 2013-10-13 20:24:18,517 xlim = 0.000000 50000.000000
yt : [INFO     ] 2013-10-13 20:24:18,518 ylim = 0.000000 50000.000000
yt : [INFO     ] 2013-10-13 20:24:18,518 Making a fixed resolution buffer of (('gas', 'Density')) 800 by 800
yt : [INFO     ] 2013-10-13 20:24:18,693 Making a fixed resolution buffer of (('gas', 'Density')) 800 by 800
yt : [INFO     ] 2013-10-13 20:24:20,066 Saving plot frames/R7_YC_0200_Projection_z_Density_Density.png
```

```
(yt)>cd frames/
```



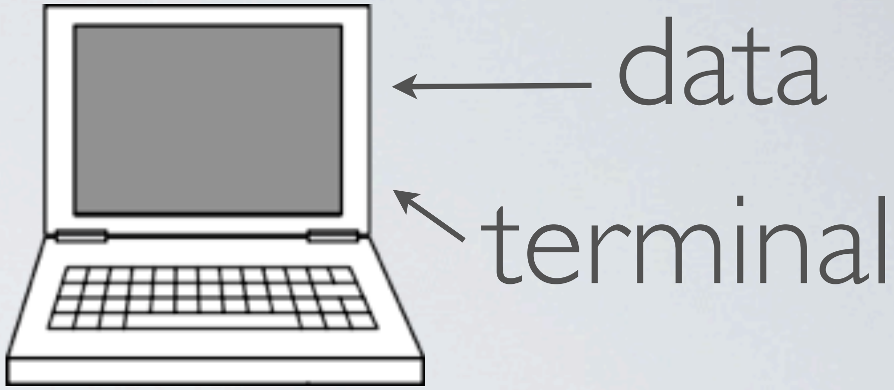
```
(yt)[tasker@Conival workshop2013]$ cd frames/
(yt)[tasker@Conival frames]$ ls
R7_YC_0200_Projection_z_Density_SoundSpeed.png
(yt)[tasker@Conival frames]$ 
```

image!

But .... how do we view it?

Command line **yt**

If data is local, viewing the image is easy!

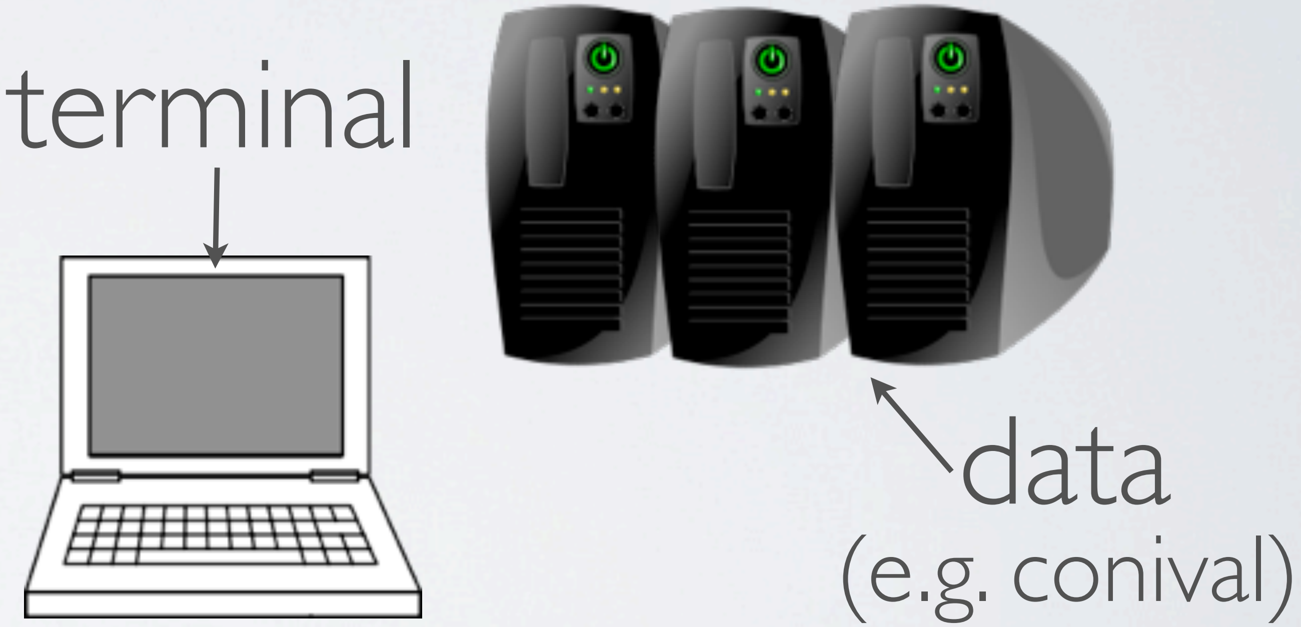

data

terminal

e.g. `(yt)>display R7_YC_0200_Projection_z_Density_Density.png`

If data is not local....

Can use scp ....


terminal

data

(e.g. conival)

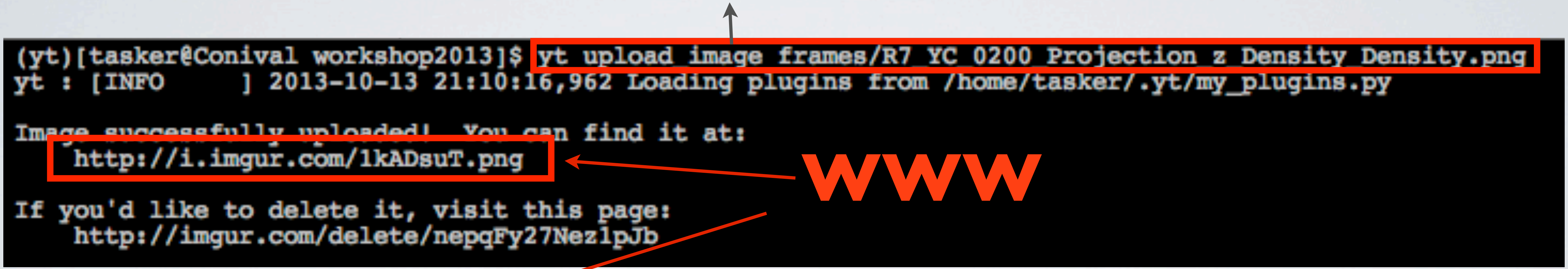
e.g.

```
(yt)>scp tasker@conival:workshop2013/frames/
R7_YC_0200_Projection_z_Density_Density.png
```

But this can be slow

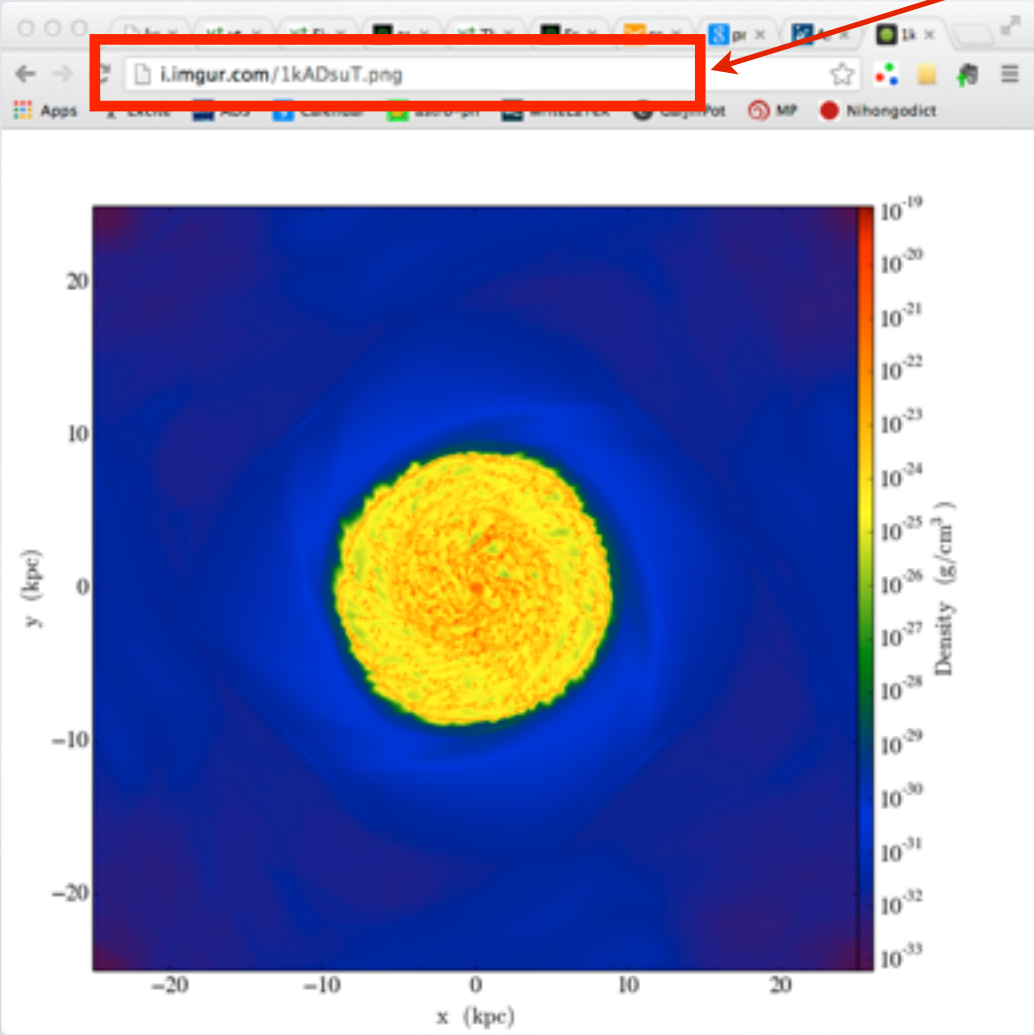
Command line **yt**

```
(yt)> yt upload_image frames/
R7_YC_0200_Projection_z_Density_Density.png
```



```
(yt)[tasker@Conival workshop2013]$ yt upload_image frames/R7_YC_0200_Projection_z_Density_Density.png
yt : [INFO     ] 2013-10-13 21:10:16,962 Loading plugins from /home/tasker/.yt/my_plugins.py

Image successfully uploaded!  You can find it at:
    http://i.imgur.com/1kADsuT.png

If you'd like to delete it, visit this page:
    http://imgur.com/delete/nepqFy27Nez1pJb
```

www



Upload to imgur.com

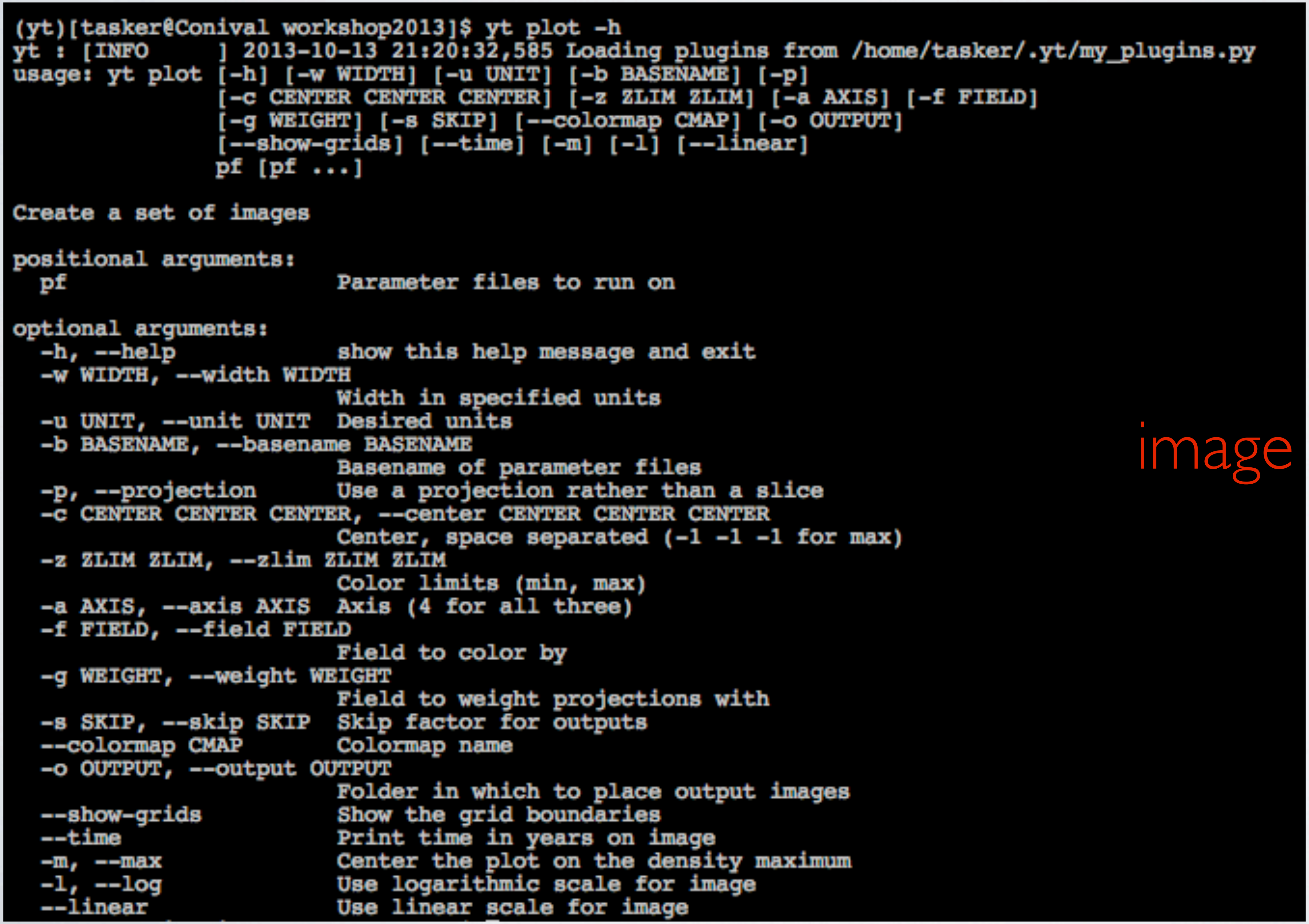
Easy to view,

easy to share @

Command line **yt**

Image changes

```
(yt)> yt plot -h
```



```
(yt)[tasker@Conival workshop2013]$ yt plot -h
yt : [INFO     ] 2013-10-13 21:20:32,585 Loading plugins from /home/tasker/.yt/my_plugins.py
usage: yt plot [-h] [-w WIDTH] [-u UNIT] [-b BASENAME] [-p]
               [-c CENTER CENTER CENTER] [-z ZLIM ZLIM] [-a AXIS] [-f FIELD]
               [-g WEIGHT] [-s SKIP] [--colormap CMAP] [-o OUTPUT]
               [--show-grids] [--time] [-m] [-l] [--linear]
               pf [pf ...]

Create a set of images

positional arguments:
  pf                    Parameter files to run on

optional arguments:
  -h, --help            show this help message and exit
  -w WIDTH, --width WIDTH
                        Width in specified units
  -u UNIT, --unit UNIT  Desired units
  -b BASENAME, --basename BASENAME
                        Basename of parameter files
  -p, --projection      Use a projection rather than a slice
  -c CENTER CENTER CENTER, --center CENTER CENTER CENTER
                        Center, space separated (-1 -1 -1 for max)
  -z ZLIM ZLIM, --zlim ZLIM ZLIM
                        Color limits (min, max)
  -a AXIS, --axis AXIS  Axis (4 for all three)
  -f FIELD, --field FIELD
                        Field to color by
  -g WEIGHT, --weight WEIGHT
                        Field to weight projections with
  -s SKIP, --skip SKIP  Skip factor for outputs
  --colormap CMAP       Colormap name
  -o OUTPUT, --output OUTPUT
                        Folder in which to place output images
  --show-grids          Show the grid boundaries
  --time                Print time in years on image
  -m, --max             Center the plot on the density maximum
  -l, --log             Use logarithmic scale for image
  --linear              Use linear scale for image
```

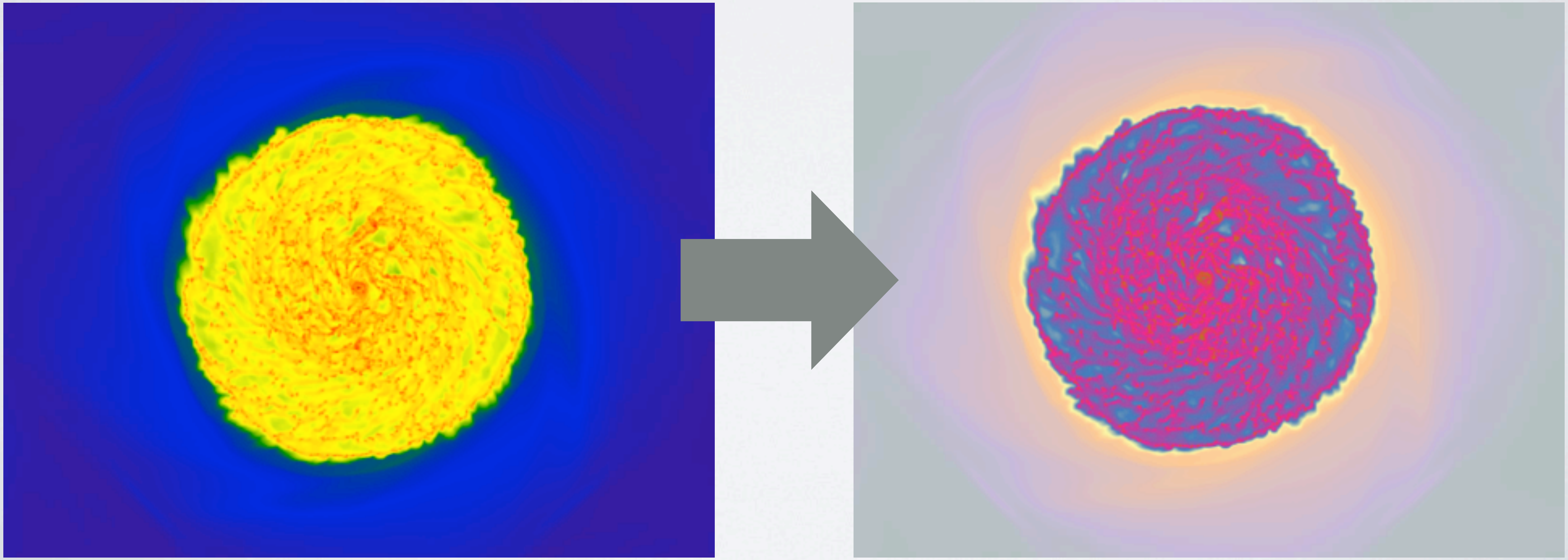
image options

Command line **yt**

Image changes

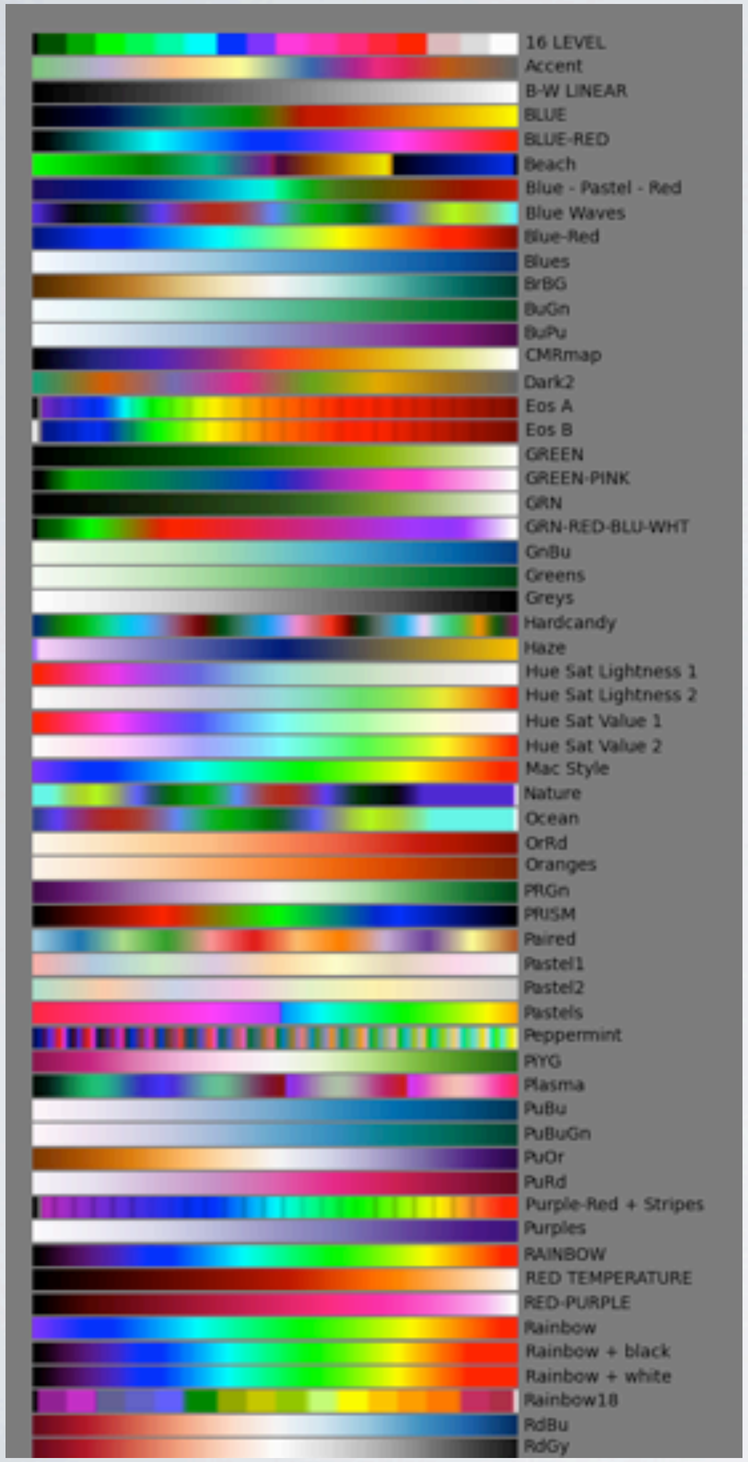
e.g.

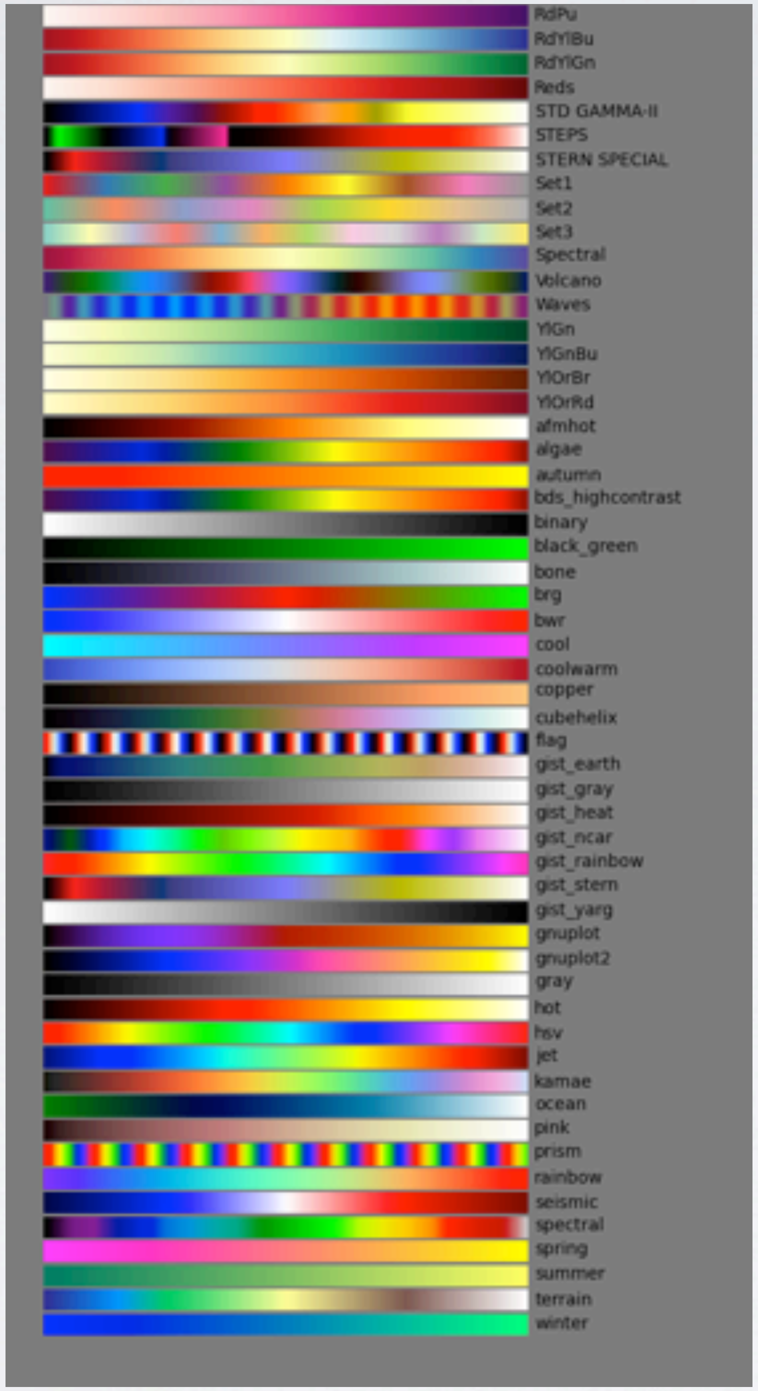
```
(yt)> yt plot --colormap Accent -p -g Density -a 2
M83/DD0200/R7_YC_0200
```



Command line **yt**

Image changes





color maps

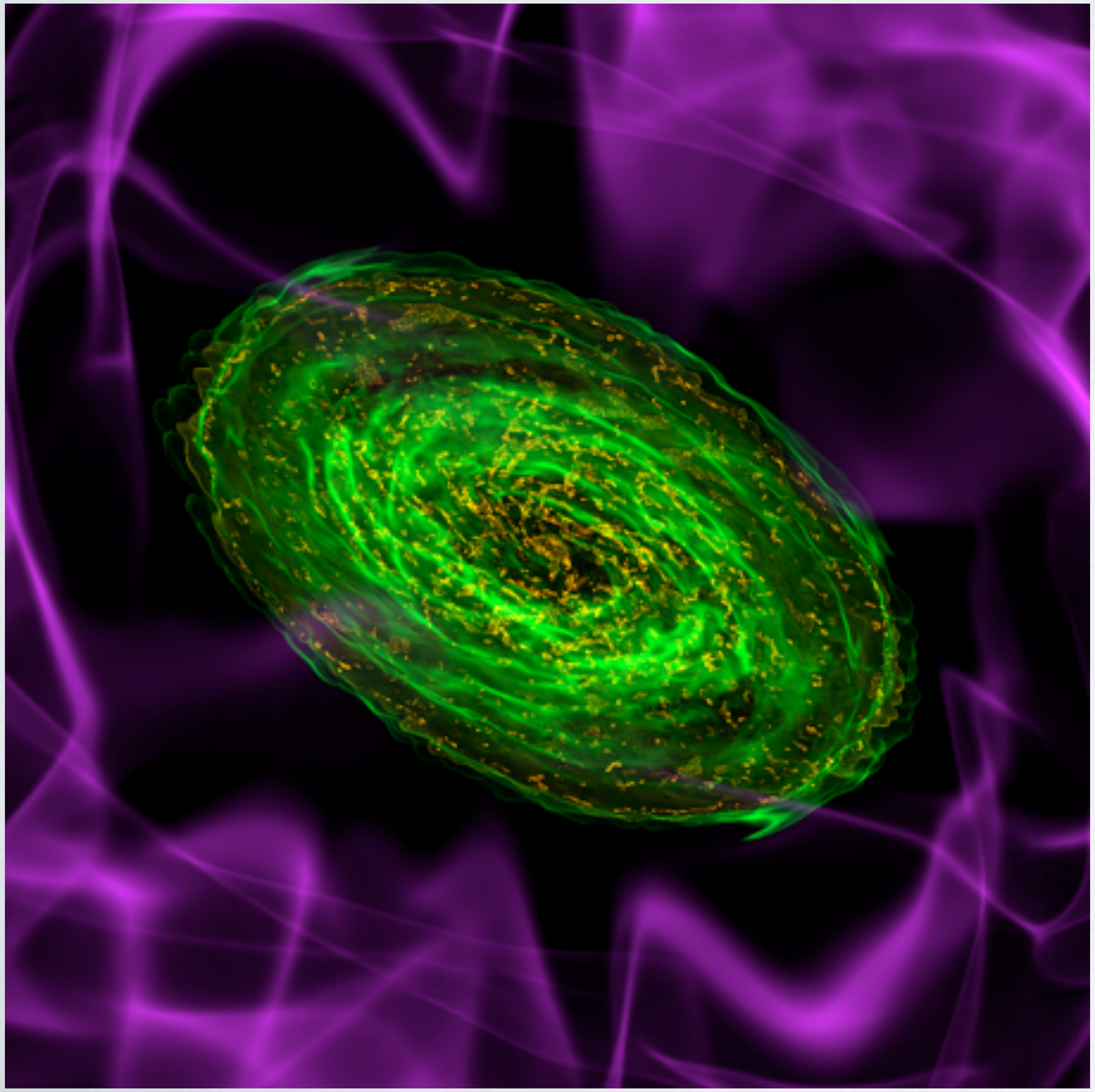
`http://yt-project.org/docs/dev/visualizing/colormaps/index.html`

Command line **yt**

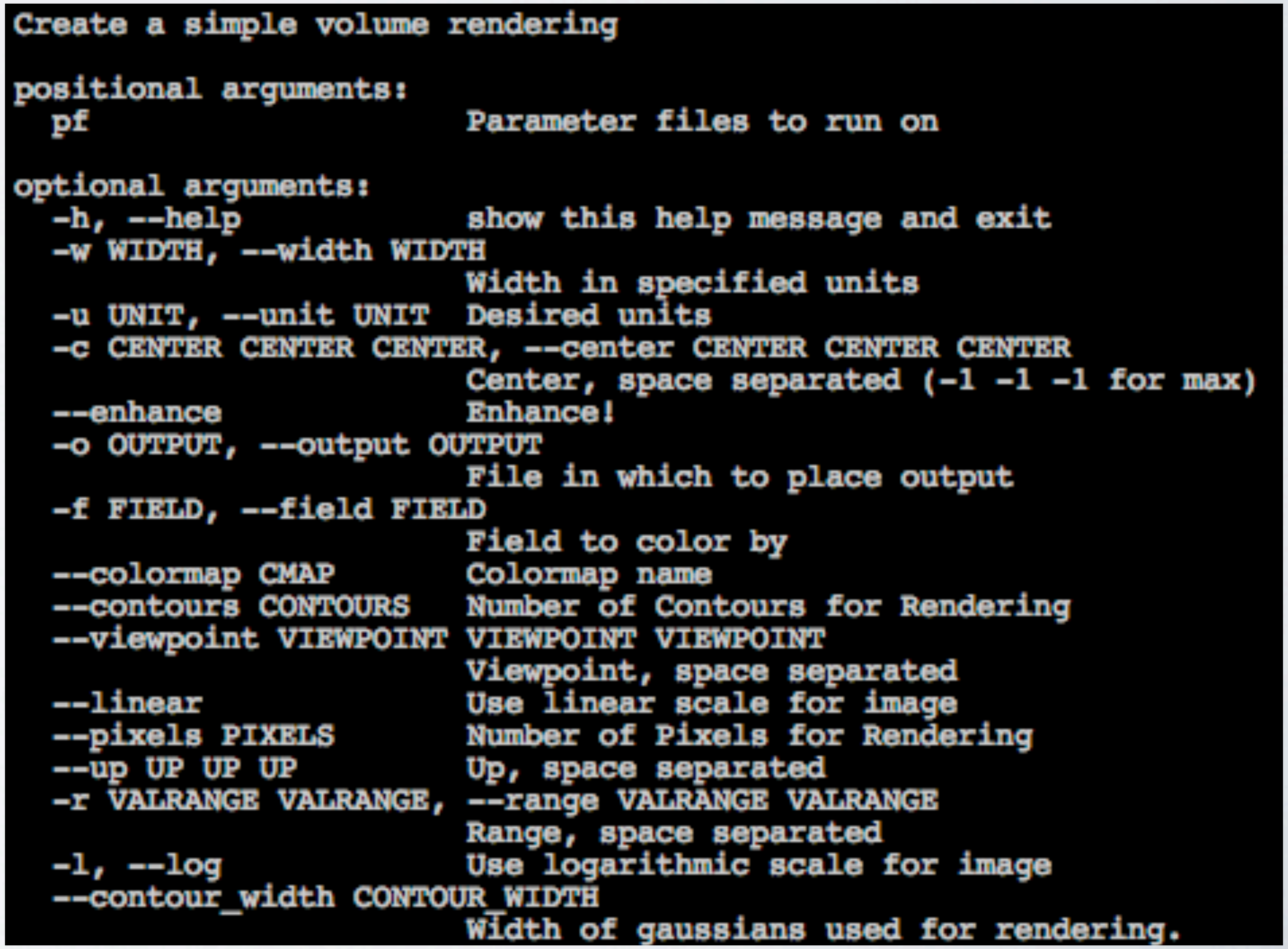
Volume rendering

```
(yt)> yt render --enhance --pixels=1024 M83/DD0200/R7_YC_0200
```

R7_YC_0200_Density_rendering.png



```
(yt)> yt render -h
```



```
Create a simple volume rendering

positional arguments:
  pf                    Parameter files to run on

optional arguments:
  -h, --help            show this help message and exit
  -w WIDTH, --width WIDTH
                        Width in specified units
  -u UNIT, --unit UNIT  Desired units
  -c CENTER CENTER CENTER, --center CENTER CENTER CENTER
                        Center, space separated (-1 -1 -1 for max)
  --enhance             Enhance!
  -o OUTPUT, --output OUTPUT
                        File in which to place output
  -f FIELD, --field FIELD
                        Field to color by
  --colormap CMAP       Colormap name
  --contours CONTOURS   Number of Contours for Rendering
  --viewpoint VIEWPOINT VIEWPOINT VIEWPOINT
                        Viewpoint, space separated
  --linear              Use linear scale for image
  --pixels PIXELS       Number of Pixels for Rendering
  --up UP UP UP         Up, space separated
  -r VALRANGE VALRANGE, --range VALRANGE VALRANGE
                        Range, space separated
  -l, --log             Use logarithmic scale for image
  --contour_width CONTOUR_WIDTH
                        Width of gaussians used for rendering.
```

iPython notebook

Command line is quick

but hard to save

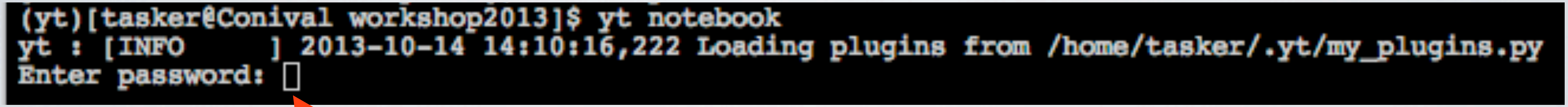
and share

Let’s try iPython notebook:

yt in your web browser

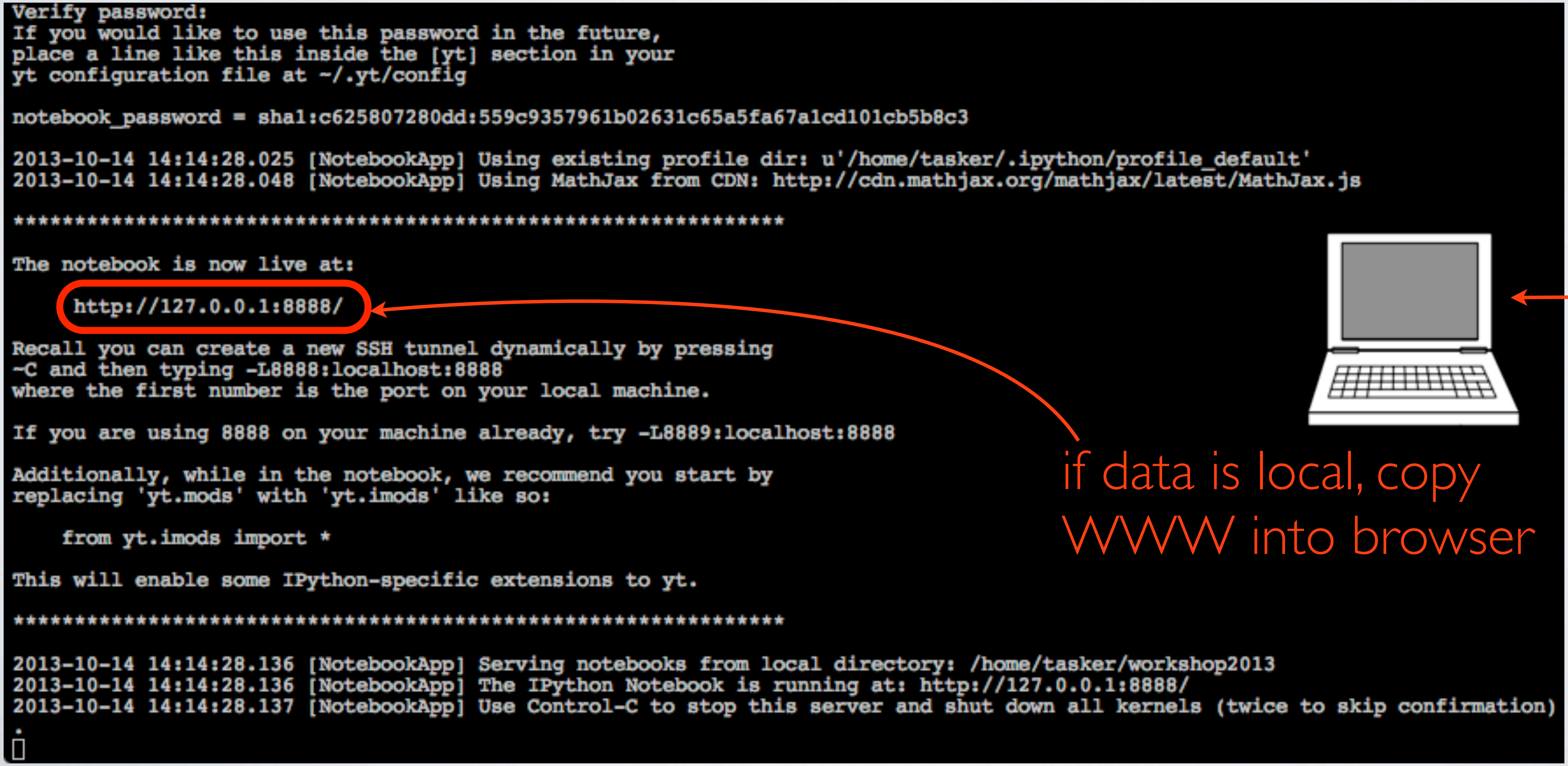
iPython notebook

```
(yt)> yt notebook
```



```
(yt)[tasker@Conival workshop2013]$ yt notebook
yt : [INFO     ] 2013-10-14 14:10:16,222 Loading plugins from /home/tasker/.yt/my_plugins.py
Enter password: 
```

any password OK

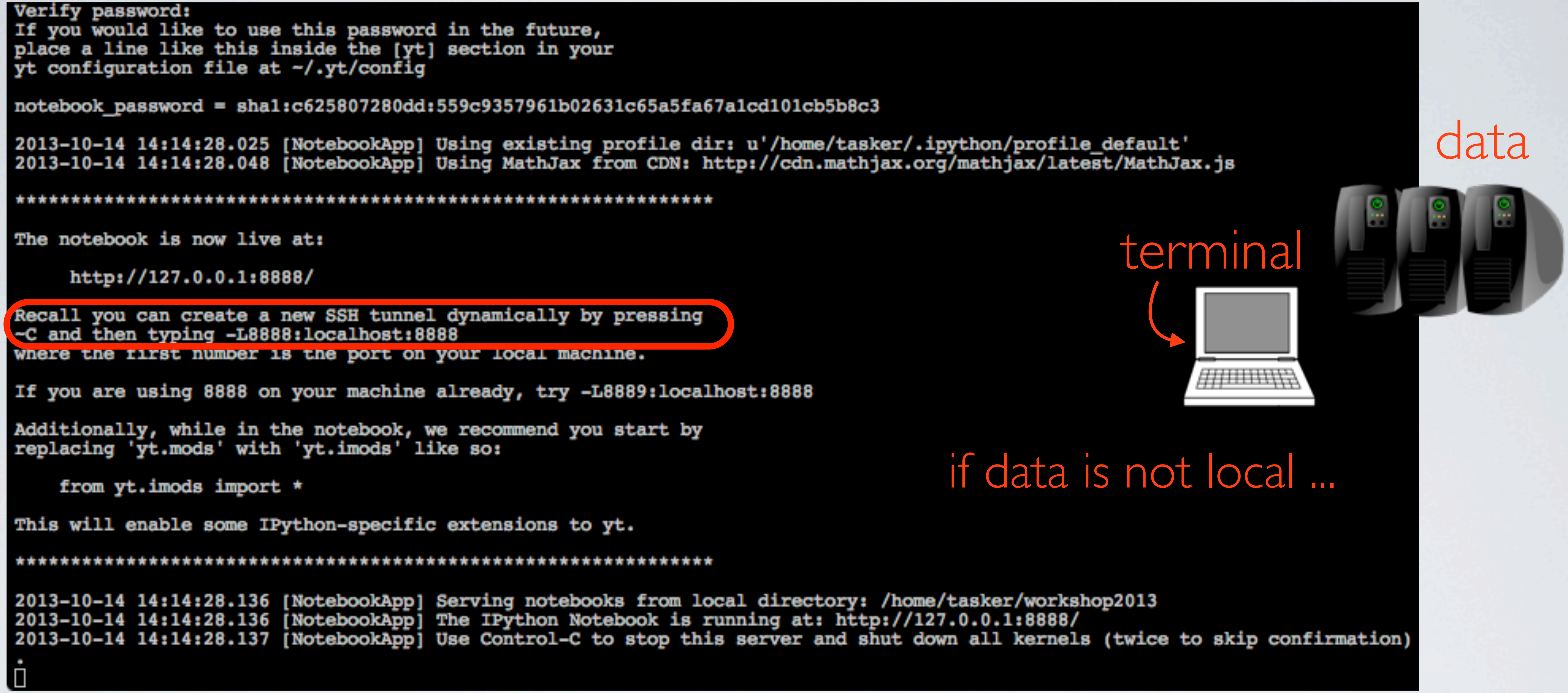


```
Verify password:
If you would like to use this password in the future,
place a line like this inside the [yt] section in your
yt configuration file at ~/.yt/config

notebook_password = sha1:c625807280dd:559c9357961b02631c65a5fa67a1cd101cb5b8c3

2013-10-14 14:14:28.025 [NotebookApp] Using existing profile dir: u'/home/tasker/.ipython/profile_default'
2013-10-14 14:14:28.048 [NotebookApp] Using MathJax from CDN: http://cdn.mathjax.org/mathjax/latest/MathJax.js

*****************************************************************

The notebook is now live at:

     http://127.0.0.1:8888/

Recall you can create a new SSH tunnel dynamically by pressing
~C and then typing -L8888:localhost:8888
where the first number is the port on your local machine.

If you are using 8888 on your machine already, try -L8889:localhost:8888

Additionally, while in the notebook, we recommend you start by
replacing 'yt.mods' with 'yt.imods' like so:

    from yt.imods import *

This will enable some IPython-specific extensions to yt.

*****************************************************************

2013-10-14 14:14:28.136 [NotebookApp] Serving notebooks from local directory: /home/tasker/workshop2013
2013-10-14 14:14:28.136 [NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
2013-10-14 14:14:28.137 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
.
```

data

if data is local, copy WWW into browser

iPython notebook



```
Verify password:
If you would like to use this password in the future,
place a line like this inside the [yt] section in your
yt configuration file at ~/.yt/config

notebook_password = sha1:c625807280dd:559c9357961b02631c65a5fa67a1cd101cb5b8c3

2013-10-14 14:14:28.025 [NotebookApp] Using existing profile dir: u'/home/tasker/.ipython/profile_default'
2013-10-14 14:14:28.048 [NotebookApp] Using MathJax from CDN: http://cdn.mathjax.org/mathjax/latest/MathJax.js

****************************************************************

The notebook is now live at:

     http://127.0.0.1:8888/

Recall you can create a new SSH tunnel dynamically by pressing
~C and then typing -L8888:localhost:8888
where the first number is the port on your local machine.

If you are using 8888 on your machine already, try -L8889:localhost:8888

Additionally, while in the notebook, we recommend you start by
replacing 'yt.mods' with 'yt.imods' like so:

    from yt.imods import *

This will enable some IPython-specific extensions to yt.

****************************************************************

2013-10-14 14:14:28.136 [NotebookApp] Serving notebooks from local directory: /home/tasker/workshop2013
2013-10-14 14:14:28.136 [NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888/
2013-10-14 14:14:28.137 [NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation)
.
```

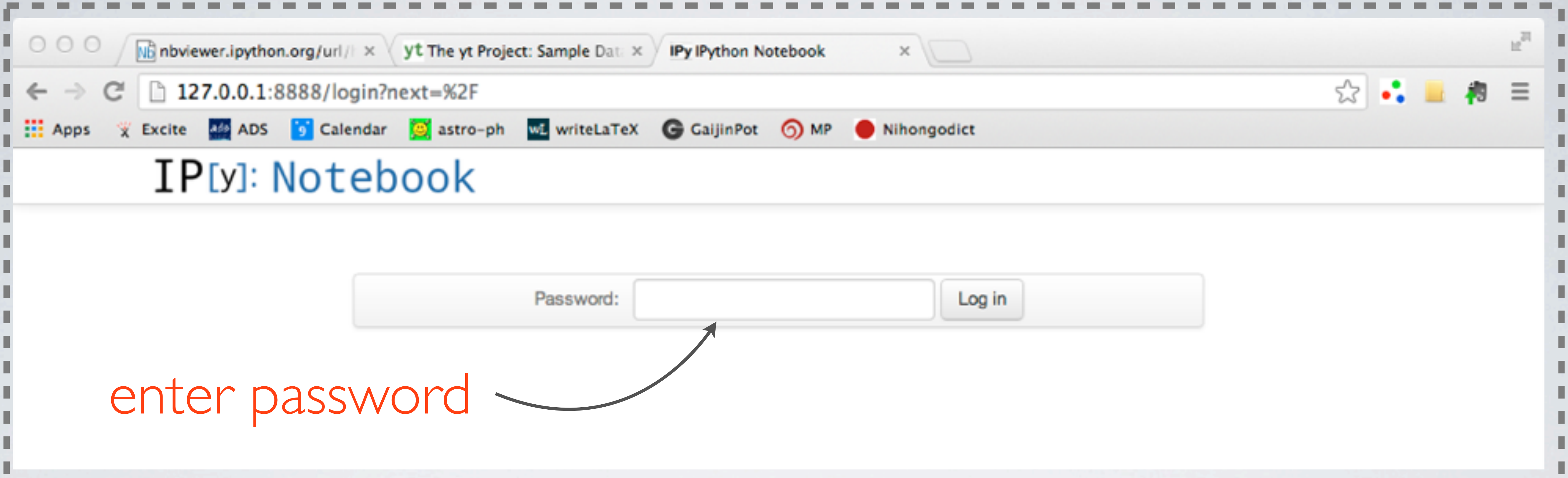
if data is not local ...

```
(yt)> ~C

ssh > -L8888:localhost:8888
```

Then go to:

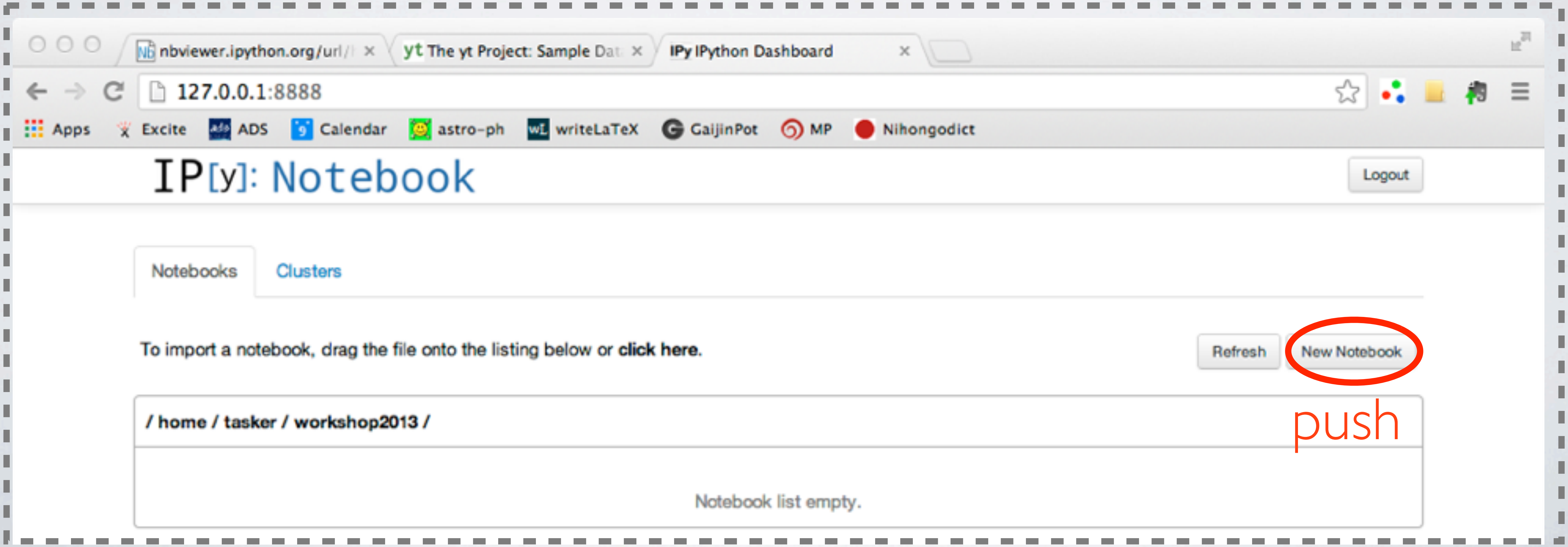
`http://127.0.0.1:8888/` in web browser

iPython notebook



Password:

enter password



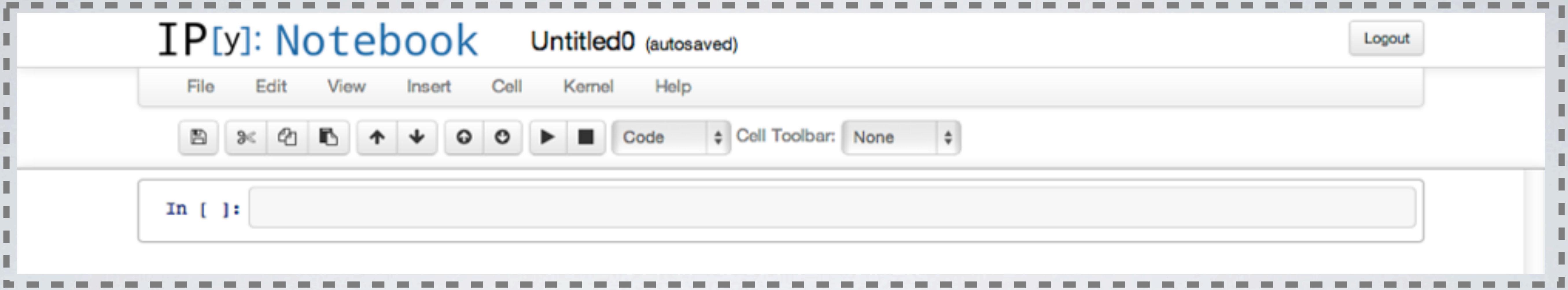
To import a notebook, drag the file onto the listing below or **click here**.

/ home / tasker / workshop2013 /

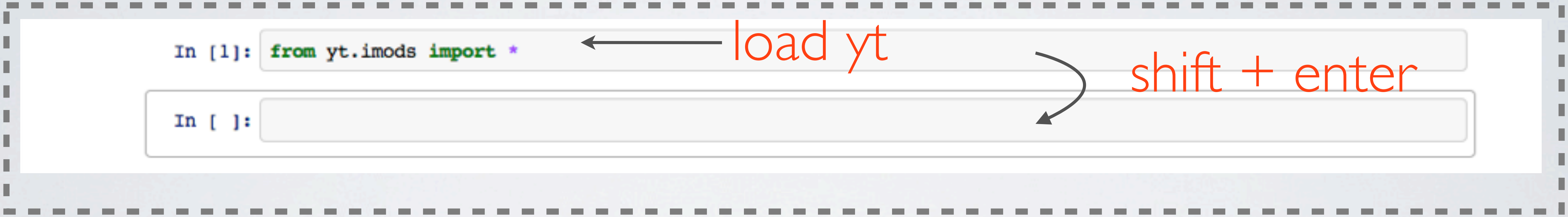
Notebook list empty.

push

iPython notebook



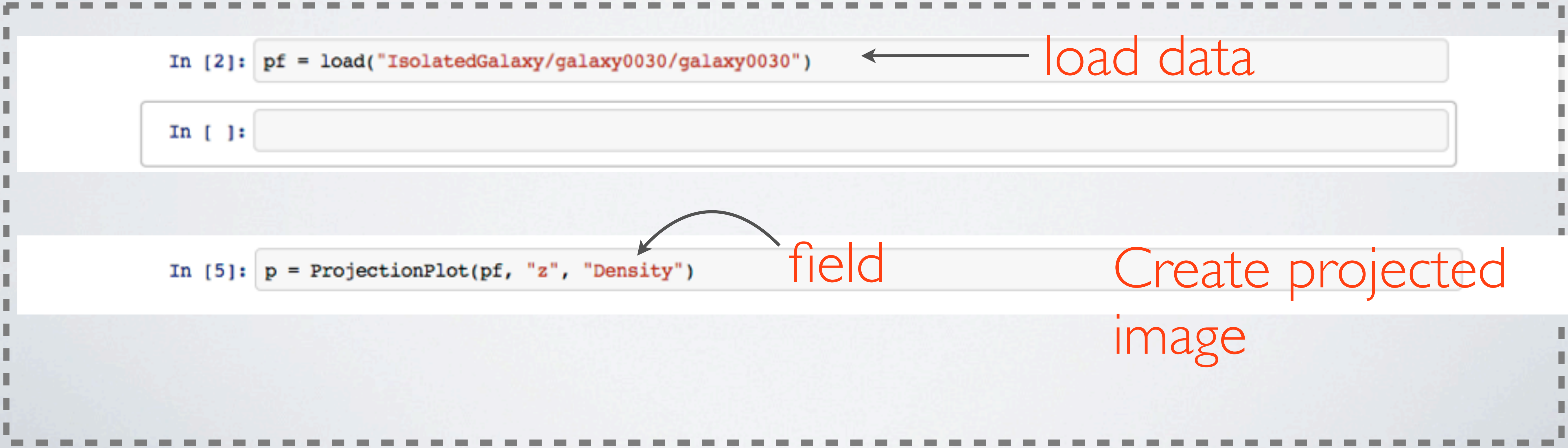
```
In [ ]:
```



```
In [1]: from yt.imods import *

In [ ]:
```

load yt

shift + enter



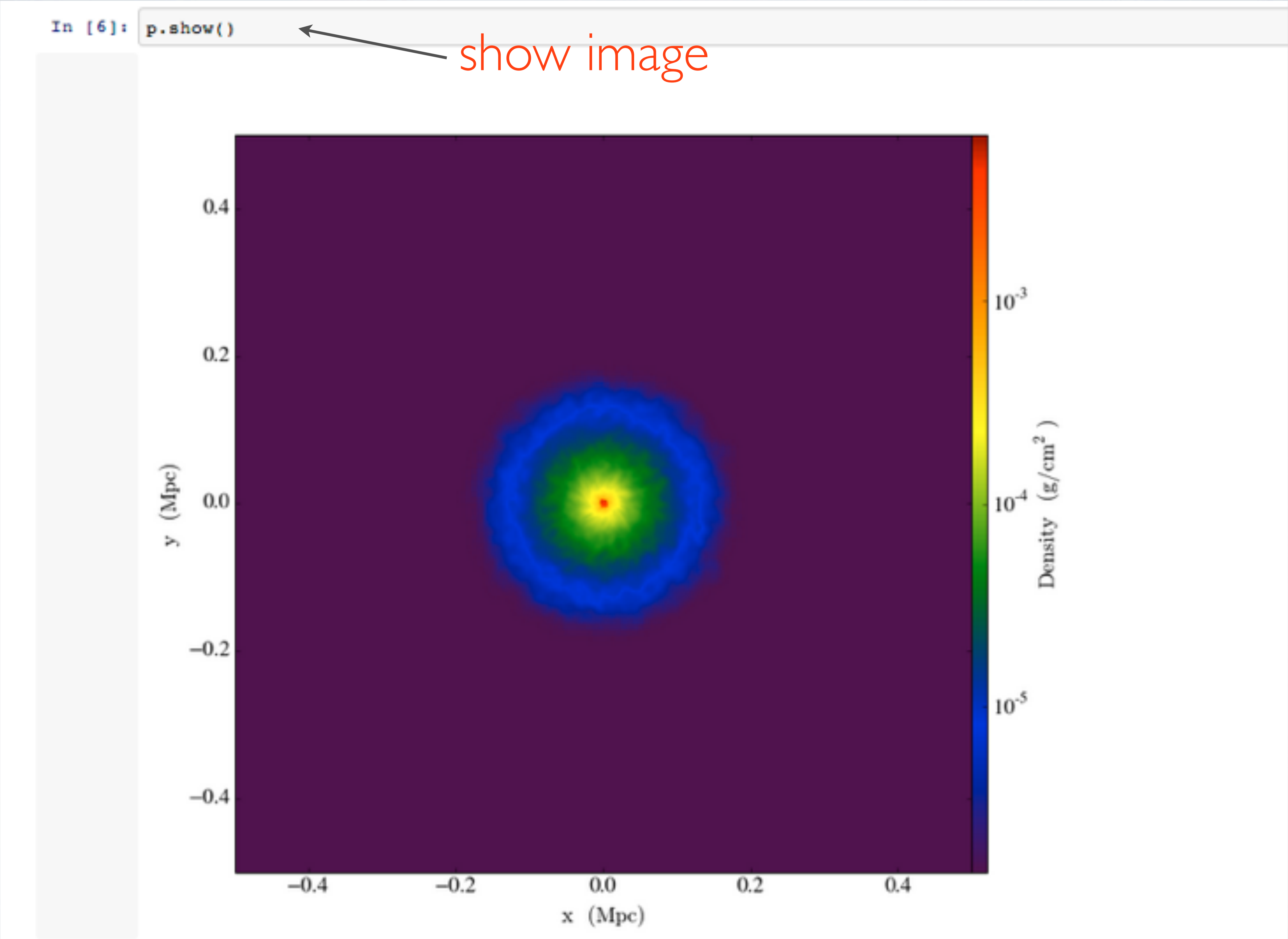
```
In [2]: pf = load("IsolatedGalaxy/galaxy0030/galaxy0030")

In [ ]:

In [5]: p = ProjectionPlot(pf, "z", "Density")
```

load data

field

Create projected image

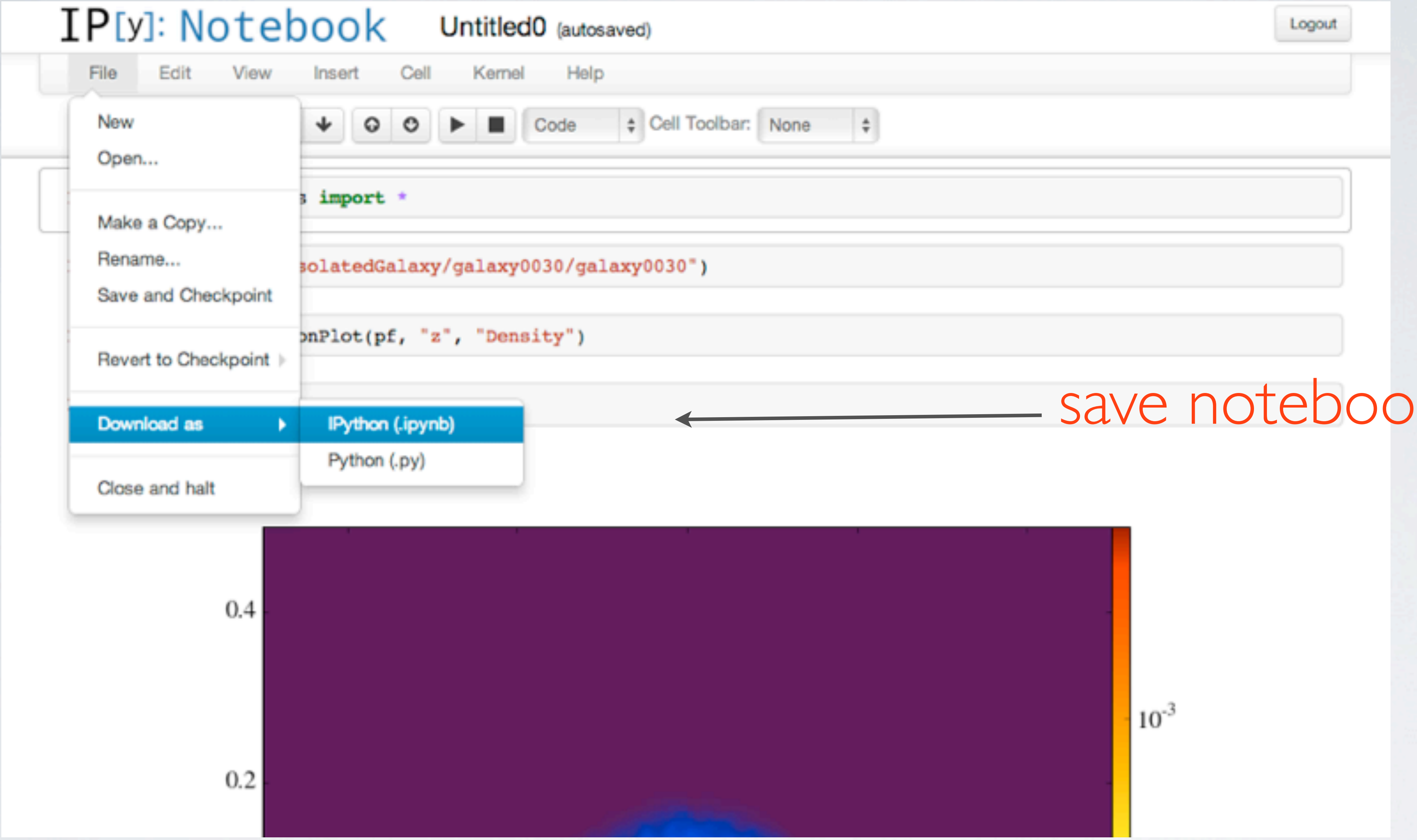
iPython notebook



```
In [6]: p.show()
```

show image

iPython notebook



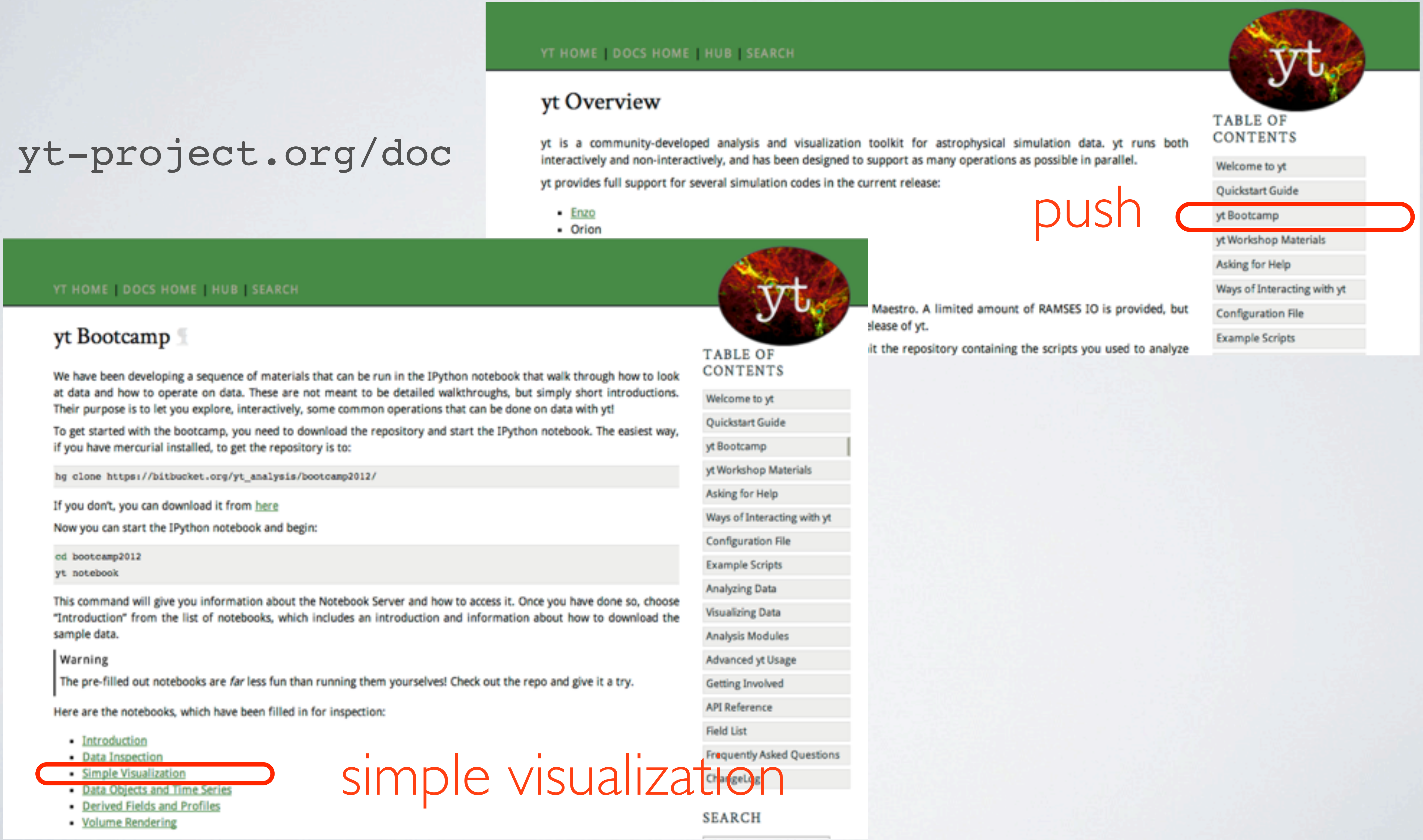
```
s import *
solatedGalaxy/galaxy0030/galaxy0030")
onPlot(pf, "z", "Density")
```

save notebook for later!

iPython notebook

Let’s try a bigger example

`yt-project.org/doc`



YT HOME | DOCS HOME | HUB | SEARCH

yt Overview

yt is a community-developed analysis and visualization toolkit for astrophysical simulation data. yt runs both interactively and non-interactively, and has been designed to support as many operations as possible in parallel.

yt provides full support for several simulation codes in the current release:

- Enzo
- Orion

Maestro. A limited amount of RAMSES IO is provided, but

elease of yt.

it the repository containing the scripts you used to analyze

TABLE OF CONTENTS

- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts

push

YT HOME | DOCS HOME | HUB | SEARCH

yt Bootcamp

We have been developing a sequence of materials that can be run in the IPython notebook that walk through how to look at data and how to operate on data. These are not meant to be detailed walkthroughs, but simply short introductions. Their purpose is to let you explore, interactively, some common operations that can be done on data with yt!

To get started with the bootcamp, you need to download the repository and start the IPython notebook. The easiest way, if you have mercurial installed, to get the repository is to:

```
hg clone https://bitbucket.org/yt_analysis/bootcamp2012/
```

If you don't, you can download it from here

Now you can start the IPython notebook and begin:

```
cd bootcamp2012
yt notebook
```

This command will give you information about the Notebook Server and how to access it. Once you have done so, choose “Introduction” from the list of notebooks, which includes an introduction and information about how to download the sample data.

> Warning
>
> The pre-filled out notebooks are *far* less fun than running them yourselves! Check out the repo and give it a try.

Here are the notebooks, which have been filled in for inspection:

- Introduction
- Data Inspection
- Simple Visualization
- Data Objects and Time Series
- Derived Fields and Profiles
- Volume Rendering

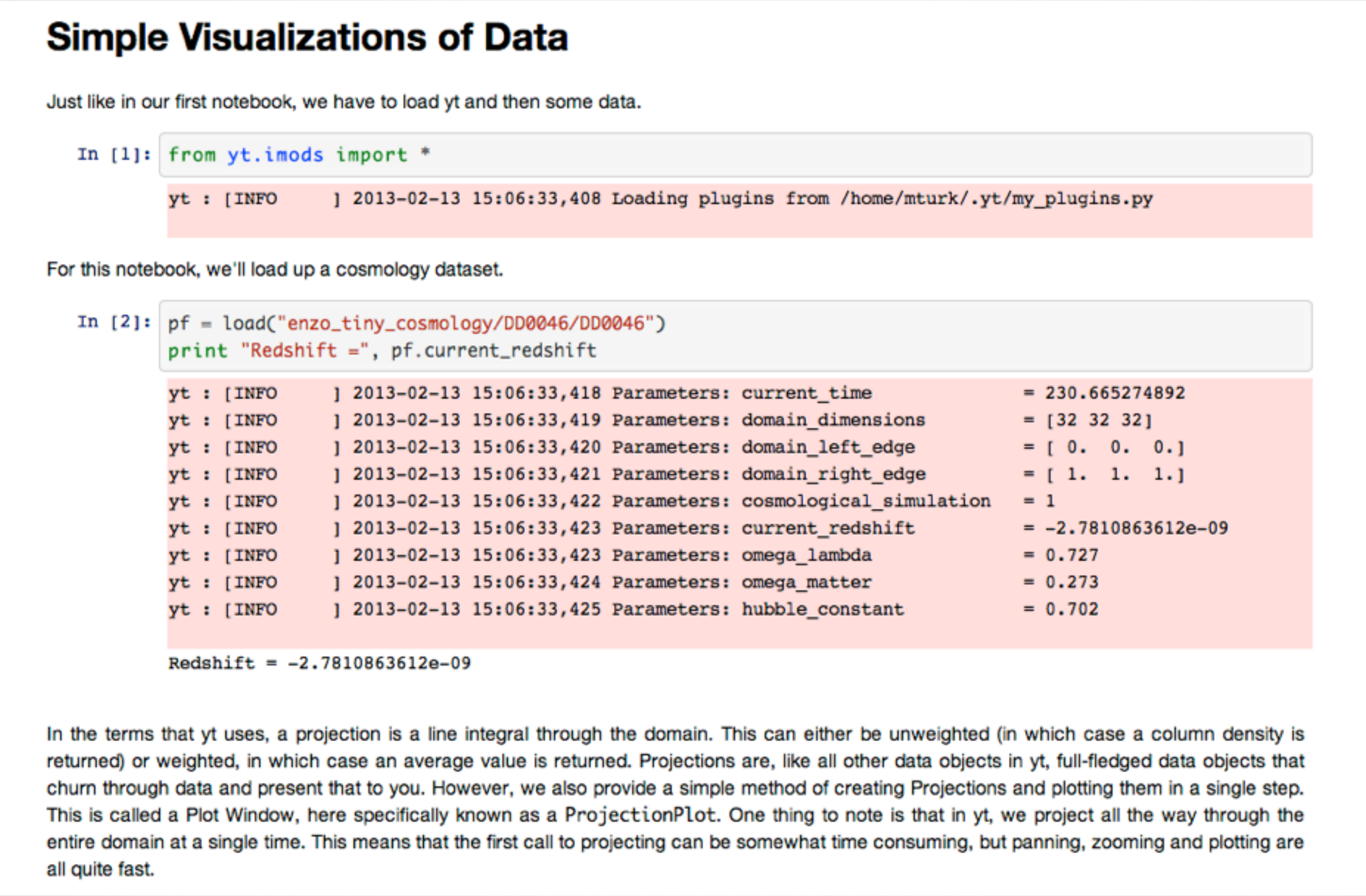
TABLE OF CONTENTS

- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts
- Analyzing Data
- Visualizing Data
- Analysis Modules
- Advanced yt Usage
- Getting Involved
- API Reference
- Field List
- Frequently Asked Questions
- Changelog

SEARCH

simple visualization

iPython notebook



Simple Visualizations of Data

Just like in our first notebook, we have to load yt and then some data.

```
In [1]: from yt.imods import *
```

```
yt : [INFO     ] 2013-02-13 15:06:33,408 Loading plugins from /home/mturk/.yt/my_plugins.py
```

For this notebook, we'll load up a cosmology dataset.

```
In [2]: pf = load("enzo_tiny_cosmology/DD0046/DD0046")
        print "Redshift =", pf.current_redshift
```

```
yt : [INFO     ] 2013-02-13 15:06:33,418 Parameters: current_time              = 230.665274892
yt : [INFO     ] 2013-02-13 15:06:33,419 Parameters: domain_dimensions         = [32 32 32]
yt : [INFO     ] 2013-02-13 15:06:33,420 Parameters: domain_left_edge          = [ 0.  0.  0.]
yt : [INFO     ] 2013-02-13 15:06:33,421 Parameters: domain_right_edge         = [ 1.  1.  1.]
yt : [INFO     ] 2013-02-13 15:06:33,422 Parameters: cosmological_simulation   = 1
yt : [INFO     ] 2013-02-13 15:06:33,423 Parameters: current_redshift          = -2.7810863612e-09
yt : [INFO     ] 2013-02-13 15:06:33,423 Parameters: omega_lambda              = 0.727
yt : [INFO     ] 2013-02-13 15:06:33,424 Parameters: omega_matter              = 0.273
yt : [INFO     ] 2013-02-13 15:06:33,425 Parameters: hubble_constant           = 0.702
```

```
Redshift = -2.7810863612e-09
```

In the terms that yt uses, a projection is a line integral through the domain. This can either be unweighted (in which case a column density is returned) or weighted, in which case an average value is returned. Projections are, like all other data objects in yt, full-fledged data objects that churn through data and present that to you. However, we also provide a simple method of creating Projections and plotting them in a single step. This is called a Plot Window, here specifically known as a `ProjectionPlot`. One thing to note is that in yt, we project all the way through the entire domain at a single time. This means that the first call to projecting can be somewhat time consuming, but panning, zooming and plotting are all quite fast.

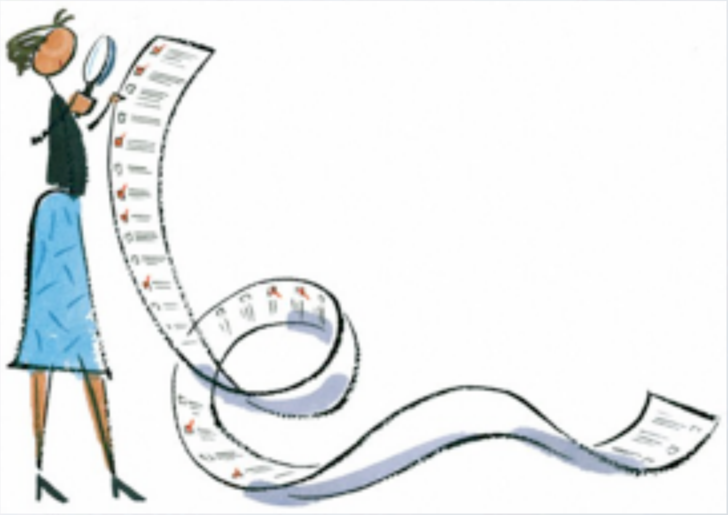
do

Scripts

Command line & iPython notebook are great for quick analysis...

But what if you want repeat the same commands 100s of times?

or write a longer programme?

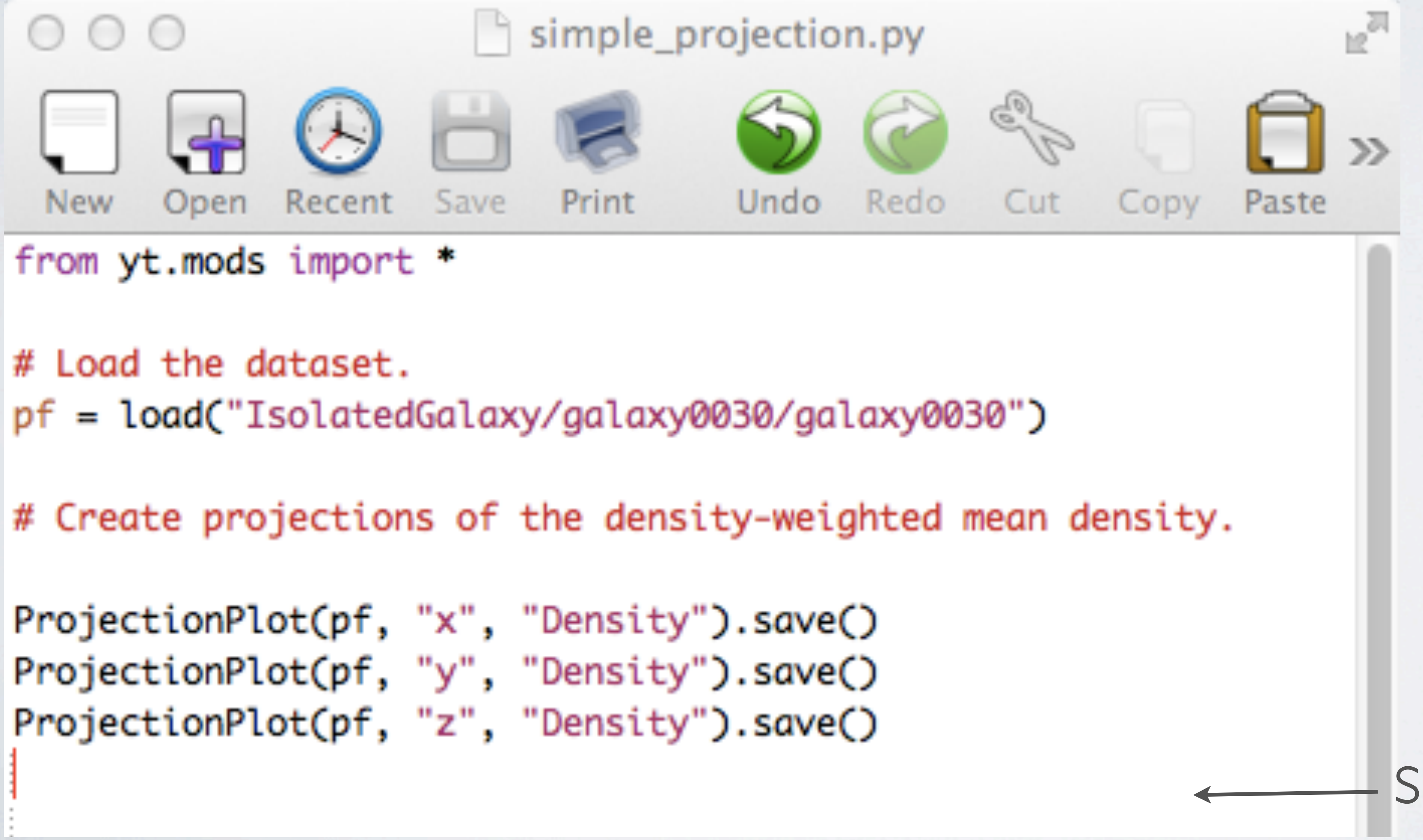


Scripts

Let’s write a script:

```
(yt) > emacs -nw simple_projection.py
```

any text editor, e.g. emacs, vim ....



```
from yt.mods import *

# Load the dataset.
pf = load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create projections of the density-weighted mean density.

ProjectionPlot(pf, "x", "Density").save()
ProjectionPlot(pf, "y", "Density").save()
ProjectionPlot(pf, "z", "Density").save()
```

save this

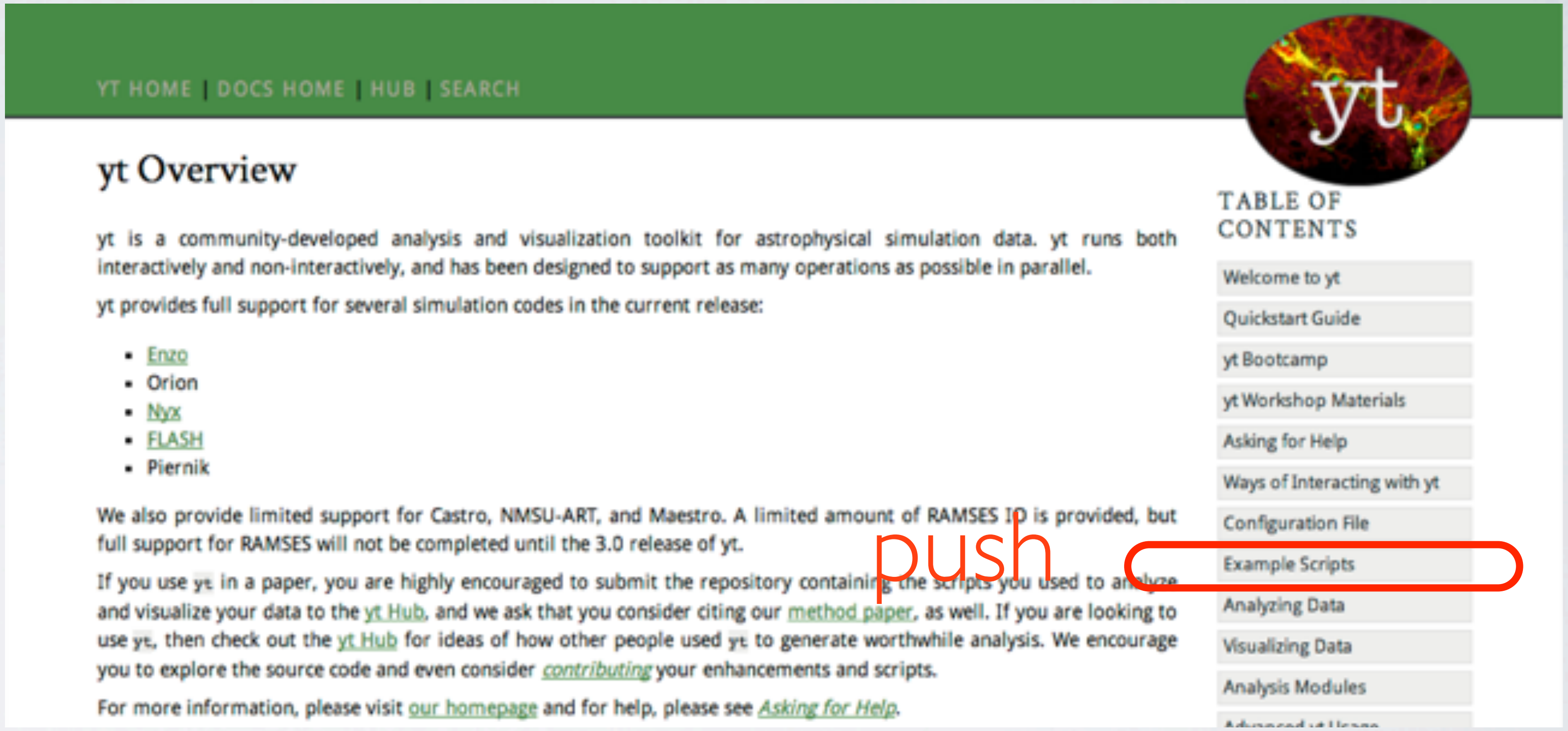
Scripts

```
(yt)> iyt

(yt)> execfile(“simple_projection.py”)

(yt)> exit()
```

Scripts

What about other plots?

`yt-project.org/doc`



YT HOME | DOCS HOME | HUB | SEARCH

yt Overview

yt is a community-developed analysis and visualization toolkit for astrophysical simulation data. yt runs both interactively and non-interactively, and has been designed to support as many operations as possible in parallel.

yt provides full support for several simulation codes in the current release:

- Enzo
- Orion
- Nyx
- FLASH
- Piernik

We also provide limited support for Castro, NMSU-ART, and Maestro. A limited amount of RAMSES IO is provided, but full support for RAMSES will not be completed until the 3.0 release of yt.

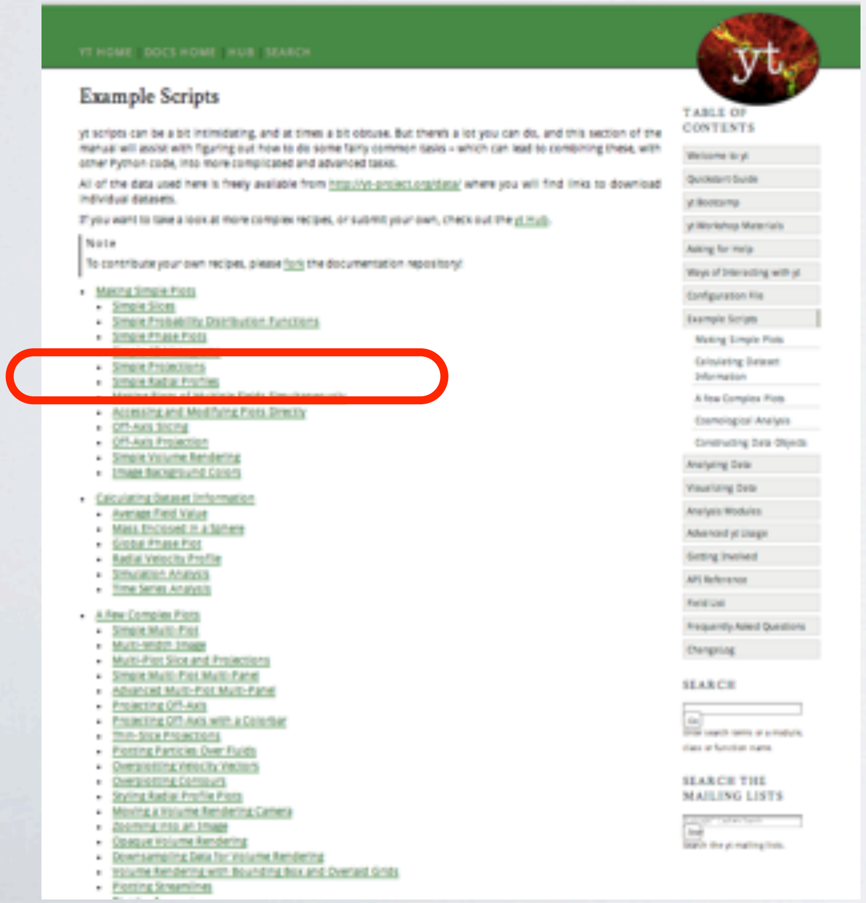
If you use yt in a paper, you are highly encouraged to submit the repository containing the scripts you used to analyze and visualize your data to the yt Hub, and we ask that you consider citing our method paper, as well. If you are looking to use yt, then check out the yt Hub for ideas of how other people used yt to generate worthwhile analysis. We encourage you to explore the source code and even consider *contributing* your enhancements and scripts.

For more information, please visit our homepage and for help, please see *Asking for Help*.

TABLE OF CONTENTS

- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts
- Analyzing Data
- Visualizing Data
- Analysis Modules
- Advanced yt Usage

'Example Scripts'



YT HOME | DOCS HOME | HUB | SEARCH

Example Scripts

yt scripts can be a bit intimidating, and at times a bit obtuse. But there's a lot you can do, and this section of the manual will assist with figuring out how to do some fairly common tasks – which can lead to combining these, with other Python code, into more complicated and advanced tasks.

All of the data used here is freely available from http://yt-project.org/data/ where you will find links to download individual datasets.

If you want to take a look at more complex recipes, or submit your own, check out the yt Hub.

> Note
>
> To contribute your own recipes, please fork the documentation repository!

- Making Simple Plots
  - Simple Slices
  - Simple Probability Distribution Functions
  - Simple Phase Plots
  - [illegible]
  - Simple Projections
  - Simple Radial Profiles
  - Making Plots of Multiple Fields Simultaneously
  - Accessing and Modifying Plots Directly
  - Off-Axis Slicing
  - Off-Axis Projection
  - Simple Volume Rendering
  - Image Background Colors
- Calculating Dataset Information
  - Average Field Value
  - Mass Enclosed in a Sphere
  - Global Phase Plot
  - Radial Velocity Profile
  - Simulation Analysis
  - Time Series Analysis
- A Few Complex Plots
  - Simple Multi-Plot
  - Multi-Width Image
  - Multi-Plot Slice and Projections
  - Simple Multi-Plot Multi-Panel
  - Advanced Multi-Plot Multi-Panel
  - Projecting Off-Axis
  - Projecting Off-Axis with a Colorbar
  - Thin-Slice Projections
  - Plotting Particles Over Fluids
  - Overplotting Velocity Vectors
  - Overplotting Contours
  - Styling Radial Profile Plots
  - Moving a Volume Rendering Camera
  - Zooming into an Image
  - Opaque Volume Rendering
  - Downsampling Data for Volume Rendering
  - Volume Rendering with Bounding Box and Overlaid Grids
  - Plotting Streamlines

TABLE OF CONTENTS

- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts
  - Making Simple Plots
  - Calculating Dataset Information
  - A Few Complex Plots
  - Cosmological Analysis
  - Constructing Data Objects
- Analyzing Data
- Visualizing Data
- Analysis Modules
- Advanced yt Usage
- Getting Involved
- API Reference
- Field List
- Frequently Asked Questions
- Changelog

SEARCH

Go

Enter search terms or a module, class or function name.

SEARCH THE MAILING LISTS

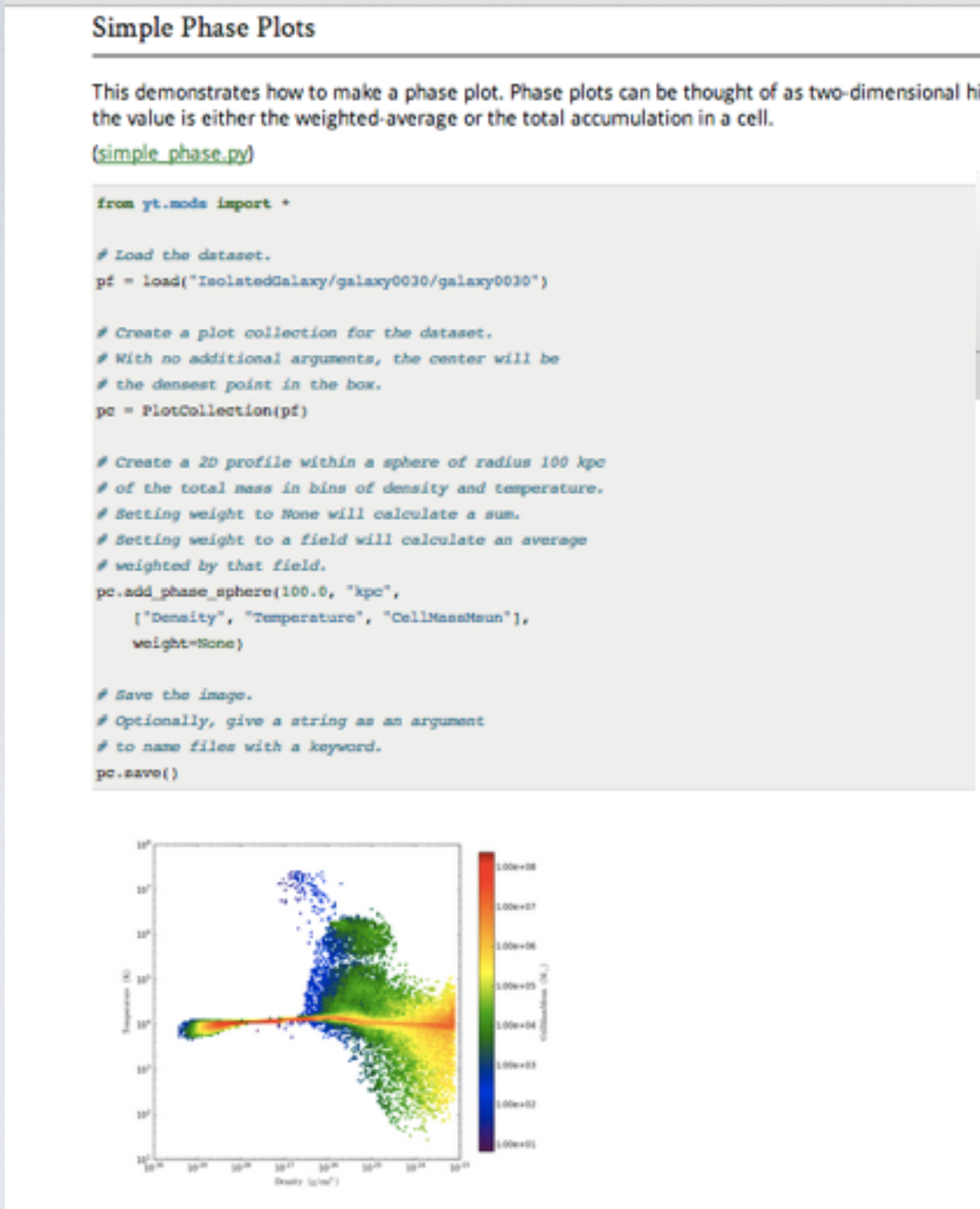
Search

Search the yt mailing lists.

Many examples!

Let's try 'Simple Phase Plots'

Scripts

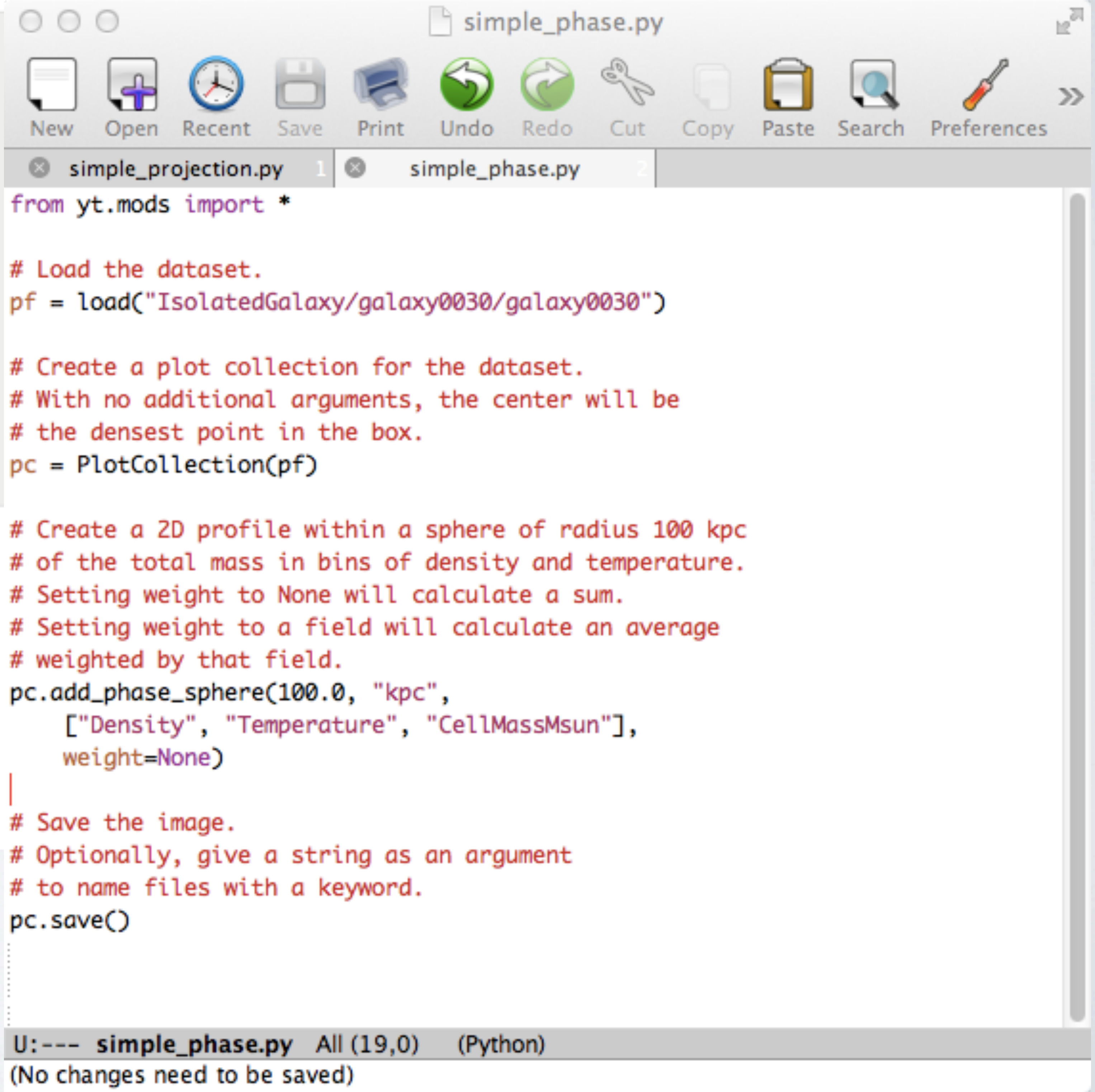


Simple Phase Plots

This demonstrates how to make a phase plot. Phase plots can be thought of as two-dimensional histograms, where the value is either the weighted-average or the total accumulation in a cell.

(simple_phase.py)

```
from yt.mods import *

# Load the dataset.
pf = load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create a plot collection for the dataset.
# With no additional arguments, the center will be
# the densest point in the box.
pc = PlotCollection(pf)

# Create a 2D profile within a sphere of radius 100 kpc
# of the total mass in bins of density and temperature.
# Setting weight to None will calculate a sum.
# Setting weight to a field will calculate an average
# weighted by that field.
pc.add_phase_sphere(100.0, "kpc",
    ["Density", "Temperature", "CellMassMsun"],
    weight=None)

# Save the image.
# Optionally, give a string as an argument
# to name files with a keyword.
pc.save()
```



```
from yt.mods import *

# Load the dataset.
pf = load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create a plot collection for the dataset.
# With no additional arguments, the center will be
# the densest point in the box.
pc = PlotCollection(pf)

# Create a 2D profile within a sphere of radius 100 kpc
# of the total mass in bins of density and temperature.
# Setting weight to None will calculate a sum.
# Setting weight to a field will calculate an average
# weighted by that field.
pc.add_phase_sphere(100.0, "kpc",
    ["Density", "Temperature", "CellMassMsun"],
    weight=None)

# Save the image.
# Optionally, give a string as an argument
# to name files with a keyword.
pc.save()
```

```
U:---  simple_phase.py   All (19,0)    (Python)
(No changes need to be saved)
```

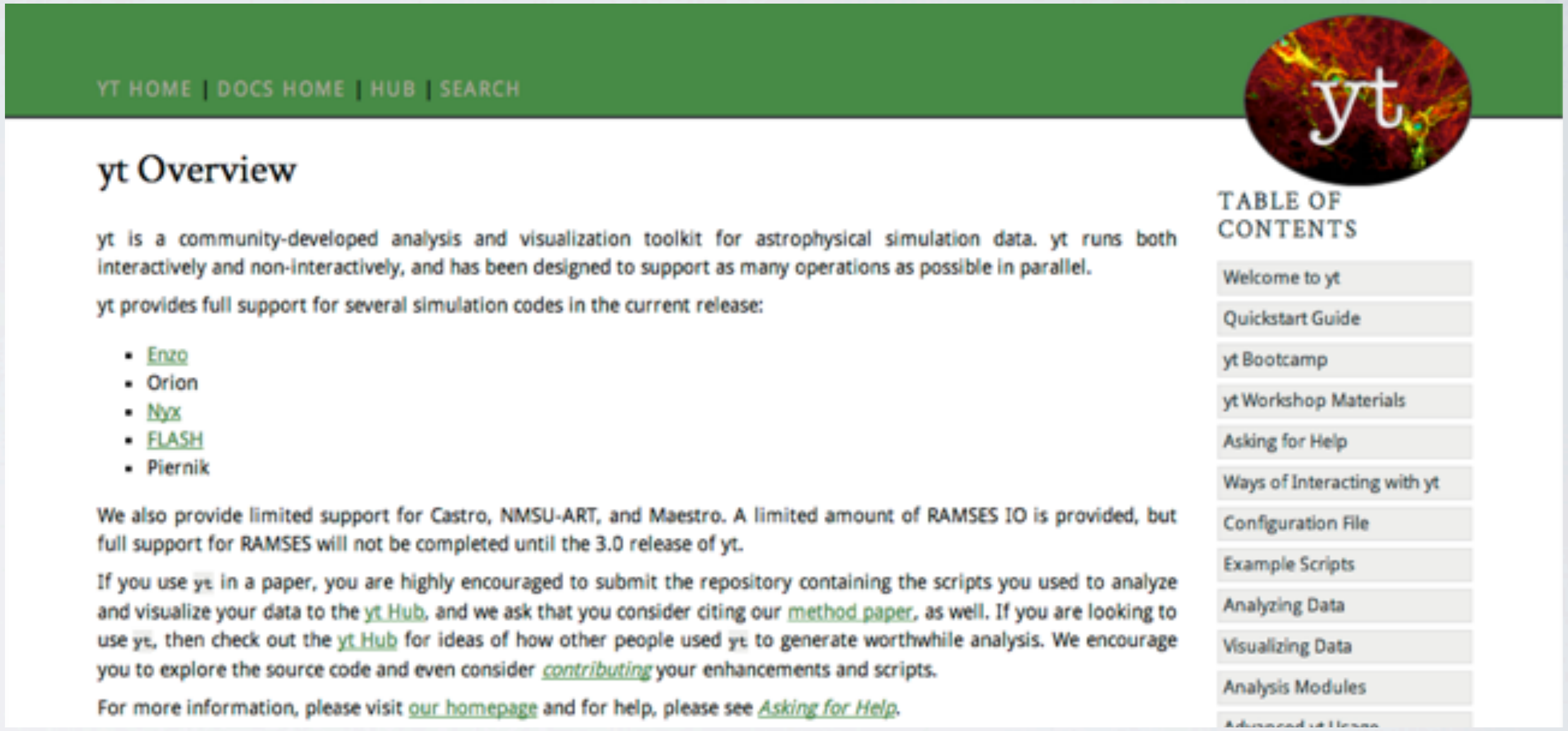
Scripts

```
(yt)> iyt

(yt)> execfile(“simple_phase.py”)

(yt)> exit()
```

The docs

Still not enough information?

`yt-project.org/doc`



yt Overview

yt is a community-developed analysis and visualization toolkit for astrophysical simulation data. yt runs both interactively and non-interactively, and has been designed to support as many operations as possible in parallel.

yt provides full support for several simulation codes in the current release:

- Enzo
- Orion
- Nyx
- FLASH
- Piernik

We also provide limited support for Castro, NMSU-ART, and Maestro. A limited amount of RAMSES IO is provided, but full support for RAMSES will not be completed until the 3.0 release of yt.

If you use `yt` in a paper, you are highly encouraged to submit the repository containing the scripts you used to analyze and visualize your data to the yt Hub, and we ask that you consider citing our method paper, as well. If you are looking to use `yt`, then check out the yt Hub for ideas of how other people used `yt` to generate worthwhile analysis. We encourage you to explore the source code and even consider *contributing* your enhancements and scripts.

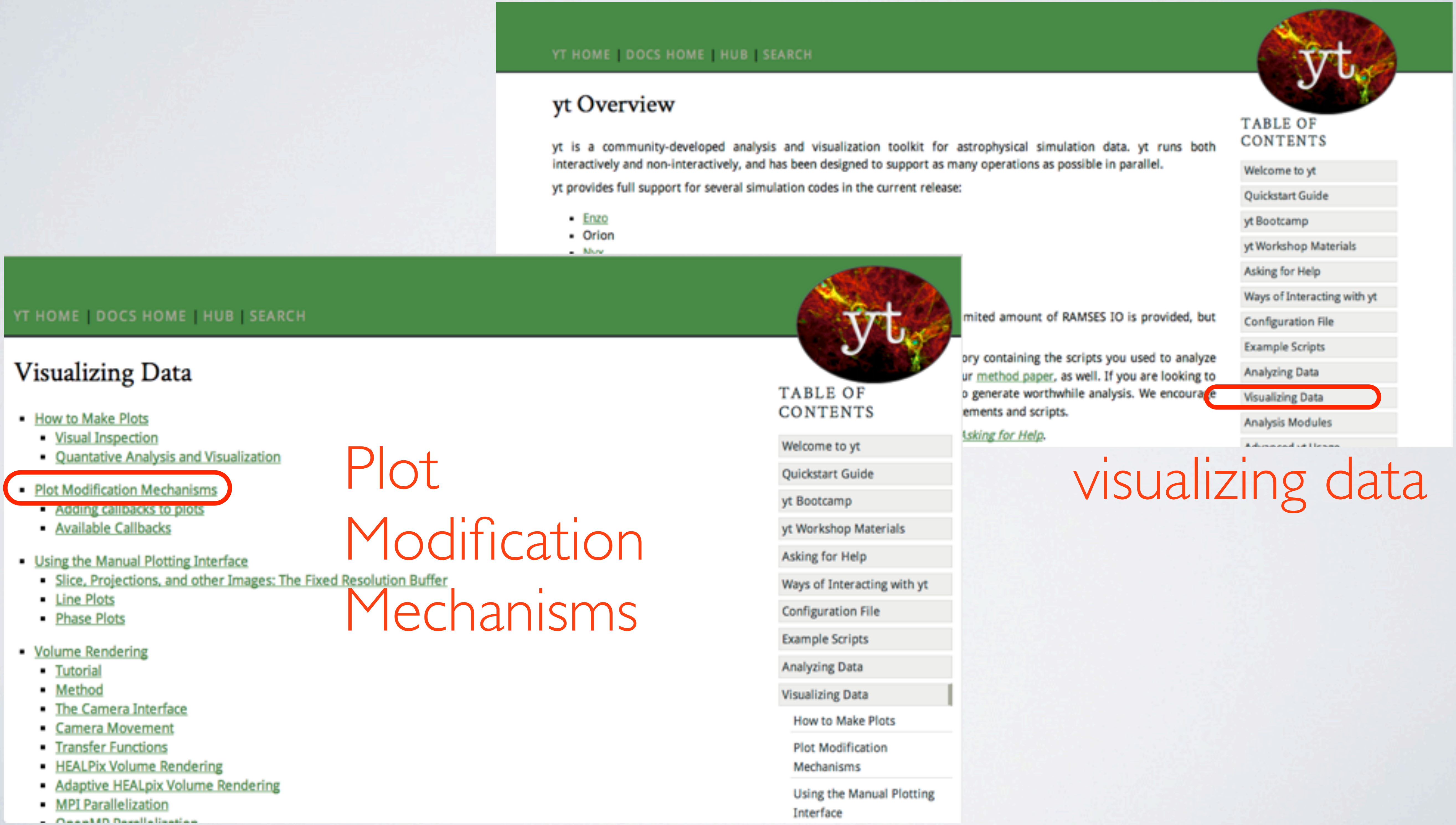
For more information, please visit our homepage and for help, please see *Asking for Help*.

TABLE OF CONTENTS

The documentation (docs) contain much more!

The docs

e.g. let's modify an image



YT HOME | DOCS HOME | HUB | SEARCH

yt Overview

yt is a community-developed analysis and visualization toolkit for astrophysical simulation data. yt runs both interactively and non-interactively, and has been designed to support as many operations as possible in parallel.

yt provides full support for several simulation codes in the current release:

- Enzo
- Orion
- Nyx

mited amount of RAMSES IO is provided, but

ory containing the scripts you used to analyze ur method paper, as well. If you are looking to o generate worthwhile analysis. We encourage ements and scripts.

Asking for Help.

TABLE OF CONTENTS

- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts
- Analyzing Data
- Visualizing Data
- Analysis Modules
- Advanced yt Usage

YT HOME | DOCS HOME | HUB | SEARCH

Visualizing Data

- How to Make Plots
  - Visual Inspection
  - Quantative Analysis and Visualization
- Plot Modification Mechanisms
  - Adding callbacks to plots
  - Available Callbacks
- Using the Manual Plotting Interface
  - Slice, Projections, and other Images: The Fixed Resolution Buffer
  - Line Plots
  - Phase Plots
- Volume Rendering
  - Tutorial
  - Method
  - The Camera Interface
  - Camera Movement
  - Transfer Functions
  - HEALPix Volume Rendering
  - Adaptive HEALpix Volume Rendering
  - MPI Parallelization
  - OpenMP Parallelization

TABLE OF CONTENTS

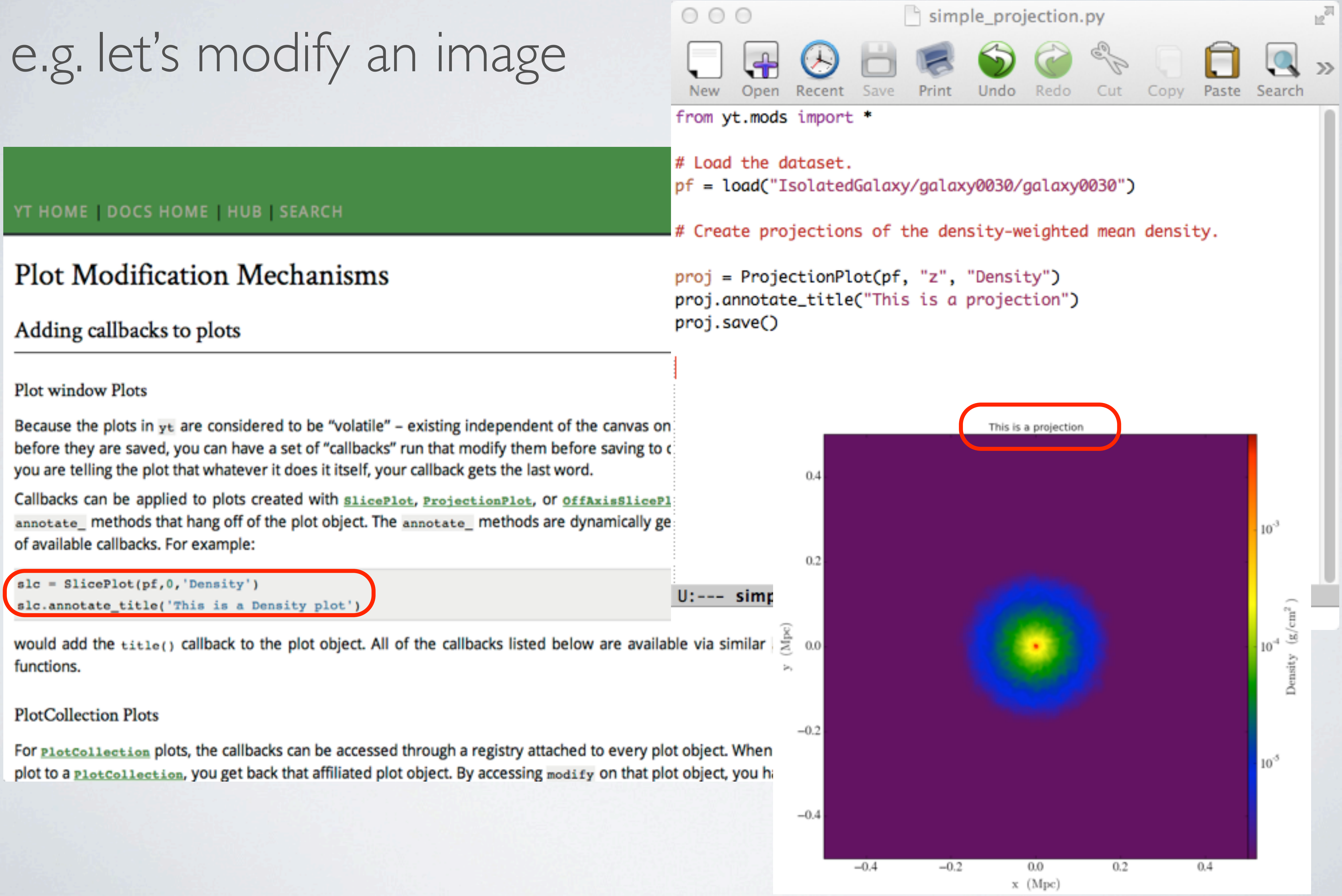
- Welcome to yt
- Quickstart Guide
- yt Bootcamp
- yt Workshop Materials
- Asking for Help
- Ways of Interacting with yt
- Configuration File
- Example Scripts
- Analyzing Data
- Visualizing Data
  - How to Make Plots
  - Plot Modification Mechanisms
  - Using the Manual Plotting Interface

Plot Modification Mechanisms

visualizing data

The docs

e.g. let’s modify an image



```
from yt.mods import *

# Load the dataset.
pf = load("IsolatedGalaxy/galaxy0030/galaxy0030")

# Create projections of the density-weighted mean density.

proj = ProjectionPlot(pf, "z", "Density")
proj.annotate_title("This is a projection")
proj.save()
```

Plot Modification Mechanisms

Adding callbacks to plots

Plot window Plots

Because the plots in `yt` are considered to be “volatile” – existing independent of the canvas on
before they are saved, you can have a set of “callbacks” run that modify them before saving to c
you are telling the plot that whatever it does it itself, your callback gets the last word.

Callbacks can be applied to plots created with `SlicePlot`, `ProjectionPlot`, or `OffAxisSlicePl`
`annotate_` methods that hang off of the plot object. The `annotate_` methods are dynamically ge
of available callbacks. For example:

```
slc = SlicePlot(pf,0,'Density')
slc.annotate_title('This is a Density plot')
```

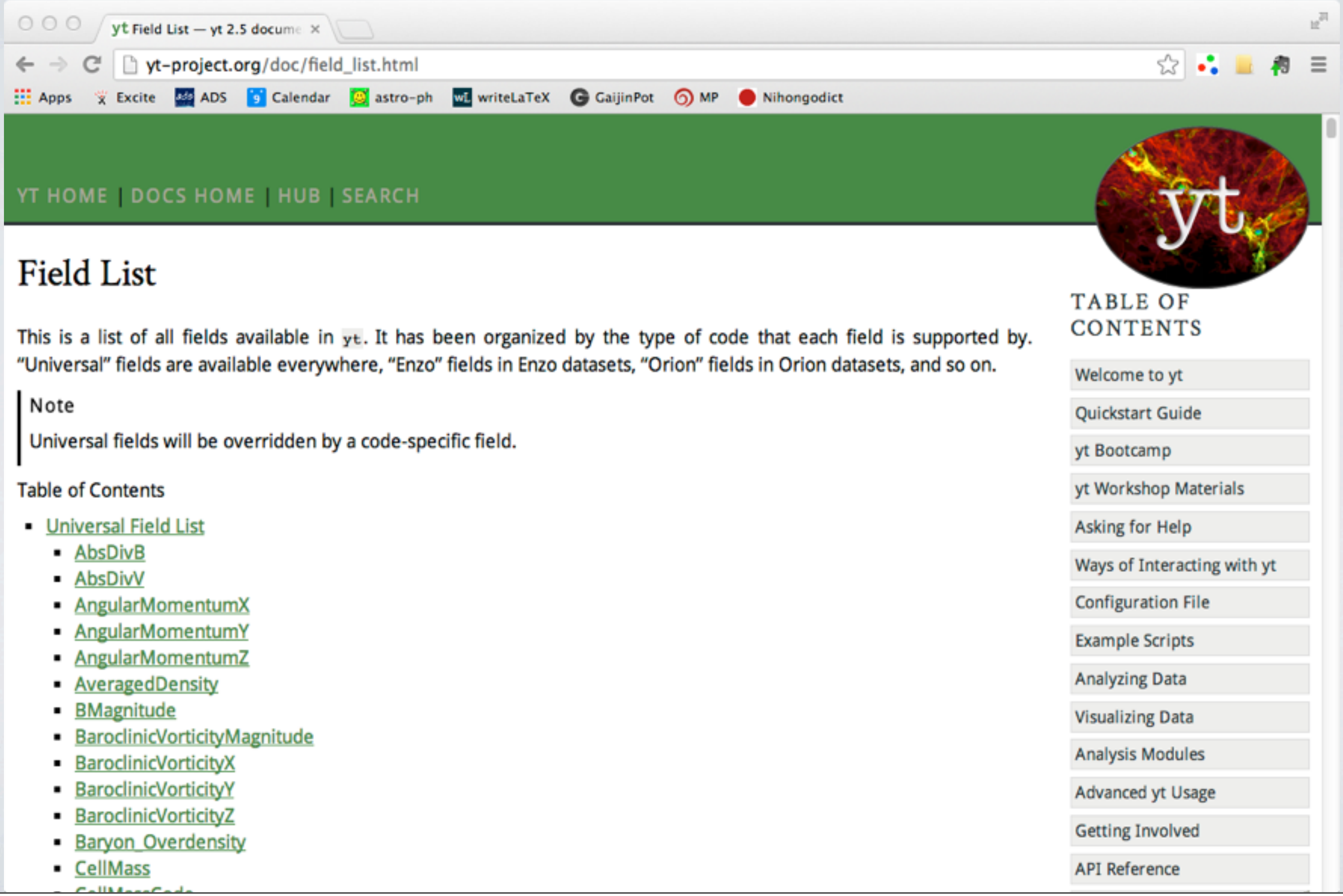
would add the `title()` callback to the plot object. All of the callbacks listed below are available via similar
functions.

PlotCollection Plots

For `PlotCollection` plots, the callbacks can be accessed through a registry attached to every plot object. When
plot to a `PlotCollection`, you get back that affiliated plot object. By accessing `modify` on that plot object, you ha

The docs

Also very very useful ...


yt Field List — yt 2.5 docume ×
yt-project.org/doc/field_list.html
Apps
Excite
ADS
Calendar
astro-ph
writeLaTeX
GaijinPot
MP
Nihongodict
YT HOME | DOCS HOME | HUB | SEARCH
yt

Field List

This is a list of all fields available in `yt`. It has been organized by the type of code that each field is supported by. "Universal" fields are available everywhere, "Enzo" fields in Enzo datasets, "Orion" fields in Orion datasets, and so on.

> Note
>
> Universal fields will be overridden by a code-specific field.

Table of Contents

TABLE OF CONTENTS

Summary

- You can run **yt** by.....

the command line: very quick!

with the iPython notebook: easy to save and share

like an online lab book

scripts: great for repeating jobs

best for more complicated programmes

- Practice running examples from the docs

create: A slice

A radial profile