StackClub

Release 0.1.0

Dec 08, 2018

Contents

The LSST science collaborations' Stack Club is learning the LSST software "stack" by writing tutorial Jupyter notebooks about it. These notebooks are organized by topic area, and can be browsed at the links below. There is also the `stackclub` package of useful python tools that you can import and use - click through below to learn more about them.

CHAPTER 1

Tutorial Notebooks

1.1 Getting Started

Wondering how you can get started learning about the LSST software stack, by writing tutorial notebooks and contributing them to the Stack Club's growing library? Need help getting going on the LSST Science Platform (LSP) JupyterLab? See the index table below for links to various resources, including: notes on the LSP, notebooks to walk you through the Stack Club workflow, and some help on how to explore the Stack code. Click on the "rendered" links to see the notebooks with their outputs.

Notebook	Short description	Links	Owner
Notes on Getting Started	Some brief notes on the LSST Science Platform JupyterLab set-up.	markdown	Phil Marshall
Hello World	Read about the Stack Club git/GitHub workflow, and make your first contribution to a notebook.	ipynb, rendered	Phil Marshall
Templates	A folder containing a template notebook, and a template folder README file, to help you get your project started.	link	Phil Marshall
Finding Docs	Locate the documentation for Stack code objects, including using the `stackclub` library `where_is` utility function.	ipynb, rendered	Phil Marshall
Import Tricks	Learn how to use some `stackclub` library utilities for importing notebooks and remote modules.	ipynb, rendered	Phil Marshall

1.2 Basics

This set of tutorial notebooks will help you explore the basic properties of the LSST software Stack data structures, classes and functions. The table contains links to the notebook code, and also to auto-rendered views of the notebooks with their outputs.

Notebook	Short description	Links	Owner
Calexp Guided Tour	Shows how to read an exposure object from a data repository, and how to access and display various parts.	ipynb, rendered	David Shupe
Data Inventory	Explore the available datasets in the LSST Science Platform shared folders.	ipynb, rendered	Phil Marshall

1.3 Visualization

See the table below for a set of tutorial notebooks (some provided by the Project) demonstrating visualization technologies available in the LSST Science Platform notebook aspect.

Notebook	Short description	Links	Owner
Firefly Visualization Demo	Introduction to the Firefly interactive plotter and image viewer.	ipynb, video	Simon Krughoff
Interactive Visualization with Bokeh, HoloViews, and Datashader	Examples of interactive visualization with the Boken, HoloViews, and Datashader plotting packages available in PyViz suite of data analysis python modules; brushing and linking with large datasets	ipynb, rendered	Keith Bechtol
“With Globular” LSST 2018 Tutorial	General purpose tutorial including interactive Firefly visualization.	ipynb	Jim Bosch

1.4 Image Processing

Here, we explore the image processing routines in the LSST science pipelines. See the index table below for links to the notebook code, and an auto-rendered view of the notebook with outputs.

Notebook	Short description	Links	Owner
Re-run HSC	End-to-end processing of the `ci_hsc` test dataset using the DM Stack.	ipynb, rendered, bash script	Justin Myles
BrighterFatterCorrection.ipynb	Analysis of Beam Simulator Images and Brighter-fatter Correction.	ipynb, rendered	Andrew Bradshaw

1.5 Source Detection

While source detection in the LSST science pipelines is carried out (first) during the image processing step, there are subsequent detection phases - and, moreover, we are interested in how sources are detected (and how their measured properties depends on that process). See the index table below for links to tutorial notebooks exploring this.

Notebook	Short description	Links	Owner
LowSurfaceBrightness.ipynb	Run source detection, deblending, and measurement tasks; subtract bright sources from an image; convolve image and detect low-surface brightness sources.	ipynb, rendered	Alex Drlica-Wagner

1.6 Deblending

This folder contains a set of tutorial notebooks exploring the deblending of LSST objects. See the index table below for links to the notebook code, and an auto-rendered view of the notebook with outputs.

Notebook	Short description	Links	Owner
SCARLET Tutorial	Introduction to the SCARLET deblender, how to configure and run it.	ipynb, rendered	Fred Moolekamp
Deblending in DRP	Where and how the deblending happens, in the DRP pipeline.	ipynb, rendered	Fred Moolekamp

1.7 Validation

This set of tutorial notebooks explores the validation packages accompanying the LSST software Stack, and also contains some stand-alone notebooks useful for examining various aspects of data quality.

Notebook	Short description	Links	Owner
image_quality_demo.ipynb	Examples of image shape measurements in the Stack including PSF size and ellipticity, shape measurements with and without PSF corrections; visualizing image quality statistics aggregated with pandas; examining PSF model ellipticity residuals	ipynb, rendered	Keith Bechtol

CHAPTER 2

The `stackclub` Package

The Stack Club tutorial Jupyter notebooks make use of a number of homegrown functions and classes, which are kept in the `stackclub` package for easy import. You can browse these modules below.

2.1 Finding Documentation

There are a number of good places to find information about the classes and functions in the LSST software Stack: the built-in Jupyter notebook `help()` function already gets us a long way, but if you want to locate and read the source code, the `stackclub.where_is` function can help.

`where_is.`**`where_is`**`(`*object*, *in_the=None*`)`

Print a markdown hyperlink to the source code of *object*.

Parameters

- **object** (*python object*) – The class or function you are looking for.
- **in_the** (*string, optional*) – The kind of place you want to look in: *['source', 'repo', 'technotes']*

Examples

```
>>> from stackclub import where_is
>>> from lsst.daf.persistence import Butler
>>> where_is(Butler.get, in_the='source')
>>> where_is(Butler, in_the='repo')
>>> where_is(Butler, in_the='technotes')
```

Notes

See also the FindingDocs tutorial notebook for a working demo.

2.2 Importing Notebooks as Modules

Once this module has been imported, further `import` statements will treat Jupyter notebooks as importable modules. It's unlikely that you will need to call any of the functions or classes in *`nbimport`* yourself - this section is just for reference.

This module was adapted from the Jupyter notebook documentation (copyright (c) Jupyter Development Team, and distributed under the terms of the Modified BSD License) for use in the `stackclub` package.

class nbimport.**NotebookFinder**

Module finder that locates Jupyter Notebooks.

Notes

Once an instance of this class is appended to `sys.meta_path`, the `import` statement will work on notebook names.

Examples

To gain the ability to import notebooks, we just import the *`nbimport`* module. The DataInventory notebook might contain a useful function - here's how we'd import it:

```
>>> import stackclub
>>> import DataInventory
```

We can also import remote notebooks, using *`wimport`*:

```
>>> import stackclub
>>> dm_butler_skymap_notebook = "https://github.com/LSSTDESC/DC2-analysis/raw/
↪master/tutorials/dm_butler_skymap.ipynb"
>>> skymapper = stackclub.wimport(dm_butler_skymap_notebook, vb=True)
```

The DataInventory notebook provides a live demo of this example.

find_module(*fullname*, *path=None*)

Find the notebook module and return a suitable loader.

Parameters

- **fullname** (*string*) – Name of the notebook to be found (without ipynb extension)
- **path** (*string*) – Path of folder containing notebook (optional).

Returns **loaders[path]** – Suitable loader object for dealing with Notebook import statements.

Return type *NotebookLoader*

class nbimport.**NotebookLoader**(*path=None*)

Module Loader for Jupyter Notebooks

load_module(*fullname*)

Import a notebook as a module

Parameters **fullname** (*string*) – Name of notebook (without the .ipynb extension)

Returns **mod** – Notebook in module form, after it has been imported (executed).

Return type module

Notes

All code cells in the notebook are executed, silently (by redirecting the standard output).

`nbimport.`**`find_notebook`**(*fullname*, *path=None*)

Find a notebook, given its fully qualified name and an optional path.

Parameters

- **fullname** (*string*) – Name of the notebook to be found (without ipynb extension)
- **path** (*string, optional*) – Path of folder containing notebook.

Returns **nb_path** – File name of notebook, if found (else None)

Return type string

Notes

The input notebook name "foo.bar" is turned into "foo/bar.ipynb". Tries turning "Foo_Bar" into "Foo Bar" if Foo_Bar does not exist.

`nbimport.`**`stdoutIO`**(*stdout=None*)

Catch the stdout of the imported notebook cells.

Notes

Adapted from stackoverflow.com/questions/3906232 Note that this approach does not capture any rich notebook output, e.g. from `IPython.display`.

2.3 Importing Modules from the Web

This is pretty experimental!

`wimport.`**`wimport`**(*url*, *vb=False*)

Download a module and import it.

Parameters

- **url** (*string*) – Web address of the target module
- **vb** (*boolean, optional*) – Verbose in operation [def=False]

Returns **globals()[modulename]** – The module, as imported.

Return type module

Notes

`wimport` maintains a secret local cache of downloaded modules, hidden from the user so that they are not tempted to edit the module locally. (If they need to do that, they should clone the relevant repo.)

Examples

Suppose the `stackclub` library did _not_ include the *where_is* module: we could still download it and import it, using *wimport*.

```
>>> where_is_url = "https://github.com/LSSTScienceCollaborations/StackClub/raw/
↪issue/79/library/stackclub/where_is.py"
>>> from stackclub import wimport
>>> so = wimport(where_is_url, vb=True)
>>> so.where_is(Butler.get, in_the='source')
```

Python Module Index

Index