
tree Documentation

Release 2.15.6

Brian Cherinka

Jul 26, 2019

Contents

1	Reference	3
2	Indices and tables	5
	Python Module Index	7
	Index	9

Welcome to tree's documentation!

This is the Sphinx documentation for the Python product tree

- What's new in tree?
- Installation of python package
- Introduction to the Tree
- Tree environment configuration
- Setting up the Tree in your local enviroment

1.1 tree Reference

1.1.1 Tree

class `tree.tree.Tree` (*args, **kwargs)

Bases: `object`

Initialize the sdss tree object

This class provides Python programmatic access to the SDSS tree environment structure

Parameters

- **key** (*str/list*) – A section or list of sections of the tree to add into the local environment
- **uproot_with** (*str*) – A new TREE_DIR path used to override an existing TREE_DIR environment variable
- **config** (*str*) – Name of manual config file to load. Default is sdsswork.
- **update** (*bool*) – If True, overwrites existing tree environment variables in your local environment. Default is False.
- **exclude** (*list*) – A list of environment variables to exclude from forced updates

Variables

- **treedir** (*str*) – The directory of the tree
- **environ** (*dict*) – The fully loaded SDSS config file held internally

add_limbs (*key=None*)

Add a new section from the tree into the existing os environment

Parameters **key** (*str*) – The section name to grab from the environment

add_paths_to_os (*key=None, update=None*)

Add the paths in tree environ into the os environ

This code goes through the tree environ and checks for existence in the os environ, then adds them

Parameters

- **key** (*str*) – The section name to check against / add
- **update** (*bool*) – If True, overwrites existing tree environment variables in your local environment. Default is False.

branch_out (*limb=None*)

Set the individual section branches

This adds the various sections of the config file into the tree environment for access later. Optically can specify a specific branch. This does not yet load them into the os environment.

Parameters limb (*str/list*) – The name of the section of the config to add into the environ or a list of strings

check_paths (*paths, update=None*)

Check if the path is in the os environ, and if not add it

Parameters:

paths (OrderedDict): An ordered dict containing all of the paths from the a given section, as key:val = name:path

update (bool): If True, overwrites existing tree environment variables in your local environment. Default is False.

get_paths (*key*)

Retrieve a set of environment paths from the config

Parameters key (*str*) – The section name to grab from the environment

Returns *self.envIRON[newkey]* (*OrderedDict*) – An ordered dict containing all of the paths from the specified section, as key:val = name:path

list_keys ()

List the available keys you can load

load_config (*config=None*)

loads a config file

Parameters config (*str*) – Optional name of manual config file to load

replant_tree (*config=None, exclude=None*)

Replant the tree with a different config setup

Parameters

- **config** (*str*) – The config name to reload
- **exclude** (*list*) – A list of environment variables to exclude from forced updates

set_roots (*uproot_with=None*)

Set the roots of the tree in the os environment

Parameters uproot_with (*str*) – A new TREE_DIR path used to override an existing TREE_DIR environment variable

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`

t

`tree.tree`, 3

A

`add_limbs()` (*tree.tree.Tree method*), 3
`add_paths_to_os()` (*tree.tree.Tree method*), 3

B

`branch_out()` (*tree.tree.Tree method*), 4

C

`check_paths()` (*tree.tree.Tree method*), 4

G

`get_paths()` (*tree.tree.Tree method*), 4

L

`list_keys()` (*tree.tree.Tree method*), 4
`load_config()` (*tree.tree.Tree method*), 4

R

`replant_tree()` (*tree.tree.Tree method*), 4

S

`set_roots()` (*tree.tree.Tree method*), 4

T

`Tree` (*class in tree.tree*), 3
`tree.tree` (*module*), 3