Lezargus

Sparrow

Sep 11, 2023

CONTENTS:

1 Code Manual **1**

1.1 lezargus 1

2 Indices and tables **43**

Python Module Index **45**

Index **47**

CHAPTER

ONE

CODE MANUAL

The code manual is primarily for software maintainers and other IRTF staff. It details the software API documentation of Lezargus and its inner workings. This does not detail any of the design principles of the software but instead it is the function and class documentation of the software (generated by Sphinx).

Code coverage[1] is available in html. The code manual (and overall documentation) is generated via Sphinx and the code coverage is generated by a different tool. As such, it is manually linked rather than integrated.

1.1 lezargus

1.1.1 lezargus package

Subpackages

lezargus.container package

Submodules

lezargus.container.cube module

Spectral data cube container.

This module and class primarily deals with spectral data cubes containing both spatial and spectral information.

class `lezargus.container.cube.`**`LezargusCube`**(*wavelength: ndarray*[2], *data: ndarray*[3], *uncertainty: ndarray*[4] *= None*, *wavelength_unit: str | Unit*[5] *| None = None*, *data_unit: str | Unit*[6] *| None = None*, *mask: ndarray*[7] *| None = None*, *flags: ndarray*[8] *| None = None*, *header: Header*[9] *| None = None*)

Bases: *`LezargusContainerArithmetic`*

[1] https://psmd-iberutaru.github.io/Lezargus/build/html/coverage/index.html

Container to hold spectral cube data and perform operations on it.

`wavelength`

The wavelength of the spectra. The unit of wavelength is typically in microns; but, check the *wavelength_unit* value.

Type
ndarray

`data`

The flux of the spectra cube. The unit of the flux is typically in flam; but, check the *flux_unit* value.

Type
ndarray

`uncertainty`

The uncertainty in the flux of the spectra. The unit of the uncertainty is the same as the flux value; per *uncertainty_unit*.

Type
ndarray

`wavelength_unit`

The unit of the wavelength array.

Type
Astropy Unit

`flux_unit`

The unit of the flux array.

Type
Astropy Unit

`uncertainty_unit`

The unit of the uncertainty array. This unit is the same as the flux unit.

Type
Astropy Unit

`mask`

A mask of the flux data, used to remove problematic areas. Where True, the values of the flux is considered mask.

Type
ndarray

`flags`

Flags of the flux data. These flags store metadata about the flux.

Type
ndarray

`header`

The header information, or metadata in general, about the data.

Type

Header

`__init__`(*wavelength: ndarray[10], data: ndarray[11], uncertainty: ndarray[12] = None, wavelength_unit: str | Unit[13] | None = None, data_unit: str | Unit[14] | None = None, mask: ndarray[15] | None = None, flags: ndarray[16] | None = None, header: Header[17] | None = None*) → None

Instantiate the spectra class.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength of the spectra.
- **`data`** (*`ndarray`*) – The flux of the spectra.
- **`uncertainty`** (*`ndarray, default = None`*) – The uncertainty of the spectra. By default, it is None and the uncertainty value is 0.
- **`wavelength_unit`** (*`Astropy-Unit like, default = None`*) – The wavelength unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`data_unit`** (*`Astropy-Unit like, default = None`*) – The data unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`mask`** (*`ndarray, default = None`*) – A mask which should be applied to the spectra, if needed.
- **`flags`** (*`ndarray, default = None`*) – A set of flags which describe specific points of data in the spectra.
- **`header`** (*`Header, default = None`*) – A set of header data describing the data. Note that when saving, this header is written to disk with minimal processing. We highly suggest writing of the metadata to conform to the FITS Header specification as much as possible.

`classmethod read_fits_file`(*filename: str*) → Self

Read a Lezargus cube FITS file.

We load a Lezargus FITS file from disk. Note that this should only be used for 3-D cube files.

Parameters

`filename` (*`str`*) – The filename to load.

Returns

cube – The LezargusCube class instance.

Return type

Self-like

`write_fits_file`(*filename: str, overwrite: bool = False*) → Self

Write a Lezargus cube FITS file.

We write a Lezargus FITS file to disk.

Parameters

- **`filename`** (*`str`*) – The filename to write to.
- **`overwrite`** (*`bool, default = False`*) – If True, overwrite file conflicts.

Return type

None

lezargus.container.image module

Image data container.

This module and class primarily deals with images containing spatial information.

class `lezargus.container.image.`**`LezargusImage`**(*data: ndarray*[18], *uncertainty: ndarray*[19] *| None = None*, *wavelength: float | None = None*, *wavelength_unit: str | Unit*[20] *= None*, *data_unit: str | Unit*[21] *| None = None*, *mask: ndarray*[22] *| None = None*, *flags: ndarray*[23] *| None = None*, *header: Header*[24] *| None = None*)

Bases: *`LezargusContainerArithmetic`*

Container to hold image and perform operations on it.

`wavelength`

The wavelength of the image. The unit of wavelength is typically in microns; but, check the *wavelength_unit* value. If none has been provided, this value is an array of None.

Type

float

`data`

The flux of the spectra cube. The unit of the flux is typically in flam; but, check the *flux_unit* value.

[2] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[3] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[4] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[5] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[6] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[7] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[8] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[9] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[10] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[11] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[12] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[13] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[14] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[15] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[16] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[17] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header

Type
ndarray

`uncertainty`

The uncertainty in the flux of the spectra. The unit of the uncertainty is the same as the flux value; per *uncertainty_unit*.

Type
ndarray

`wavelength_unit`

The unit of the wavelength array.

Type
Astropy Unit

`flux_unit`

The unit of the flux array.

Type
Astropy Unit

`uncertainty_unit`

The unit of the uncertainty array. This unit is the same as the flux unit.

Type
Astropy Unit

`mask`

A mask of the flux data, used to remove problematic areas. Where True, the values of the flux is considered mask.

Type
ndarray

`flags`

Flags of the flux data. These flags store metadata about the flux.

Type
ndarray

`header`

The header information, or metadata in general, about the data.

Type
Header

`__init__`(*data: ndarray*[25], *uncertainty: ndarray*[26] *| None = None*, *wavelength: float | None = None*, *wavelength_unit: str | Unit*[27] *= None*, *data_unit: str | Unit*[28] *| None = None*, *mask: ndarray*[29] *| None = None*, *flags: ndarray*[30] *| None = None*, *header: Header*[31] *| None = None*) → None

Instantiate the spectra class.

Parameters

- **`data`** (*`ndarray`*) – The flux of the spectra.
- **`uncertainty`** (*`ndarray, default = None`*) – The uncertainty of the spectra. By default, it is None and the uncertainty value is 0.
- **`wavelength`** (*`ndarray, default = None`*) – The wavelength of the image. If this is not provided, it defaults to 0, otherwise, it is an array of a single value.
- **`wavelength_unit`** (*`Astropy-Unit like, default = None`*) – The wavelength unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`data_unit`** (*`Astropy-Unit like, default = None`*) – The data unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`mask`** (*`ndarray, default = None`*) – A mask which should be applied to the spectra, if needed.
- **`flags`** (*`ndarray, default = None`*) – A set of flags which describe specific points of data in the spectra.
- **`header`** (*`Header, default = None`*) – A set of header data describing the data. Note that when saving, this header is written to disk with minimal processing. We highly suggest writing of the metadata to conform to the FITS Header specification as much as possible.

`classmethod read_fits_file`(*filename: str*) → Self

Read a Lezargus image FITS file.

We load a Lezargus FITS file from disk. Note that this should only be used for 2-D image files.

Parameters

`filename` (*`str`*) – The filename to load.

Returns

cube – The LezargusImage class instance.

Return type

Self-like

`write_fits_file`(*filename: str*, *overwrite: bool = False*) → Self

Write a Lezargus image FITS file.

We write a Lezargus FITS file to disk.

Parameters

- **`filename`** (*`str`*) – The filename to write to.
- **`overwrite`** (*`bool, default = False`*) – If True, overwrite file conflicts.

Return type

None

lezargus.container.mosaic module

Mosaic data container.

This module and class primarily deals with a collection of data cubes pieced together into a single combined mosaic. Unlike the previous containers, this does not directly subclass Astropy NDData and instead acts as a collection of other reduced spectral cubes and performs operations on it.

class `lezargus.container.mosaic.`**`LezargusMosaic`**

Bases: `object`

TODO.

`__init__``()` → None

Init doc.

lezargus.container.parent module

Parent class for the containers to implement arithmetic and other functions.

The Astropy NDArrayData arithmetic class is not wavelength aware. This class overwrites and wraps around the NDArithmeticMixin class and allows it to be wavelength aware. We also avoid the need to do a lot of the recreating of the data object.

class `lezargus.container.parent.`**`LezargusContainerArithmetic`**`(`*wavelength: ndarray*[32]*, data: ndarray*[33]*, uncertainty: ndarray*[34]*, wavelength_unit: Unit*[35]*, data_unit: Unit*[36]*, mask: ndarray*[37]*, flags: ndarray*[38]*, header: dict*`)`

Bases: `object`

[18] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[19] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[20] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[21] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[22] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[23] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[24] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[25] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[26] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[27] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[28] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[29] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[30] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[31] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header

Lezargus wavelength-aware arithmetic.

This is the class which allows for the arithmetic behind the scenes to work with wavelength knowledge. All we do is overwrite the NDDataArray arithmetic functions to perform wavelength checks and pass it through without wavelength issues.

`wavelength`

The wavelength of the spectra. The unit of wavelength is typically in microns; but, check the *wavelength_unit* value.

Type

ndarray

`data`

The data or flux of the spectra cube. The unit of the flux is typically in flam; but, check the *data_unit* value.

Type

ndarray

`uncertainty`

The uncertainty in the data. The unit of the uncertainty is the same as the flux value; per *uncertainty_unit*.

Type

ndarray

`wavelength_unit`

The unit of the wavelength array.

Type

Astropy Unit

`data_unit`

The unit of the data array.

Type

Astropy Unit

`uncertainty_unit`

The unit of the uncertainty array. This unit is the same as the data unit.

Type

Astropy Unit

`mask`

A mask of the data, used to remove problematic areas. Where True, the values of the data is considered masked.

Type

ndarray

`flags`

Flags of the data. These flags store metadata about the data.

Type
ndarray

header

The header information, or metadata in general, about the data.

Type
Header

__add__(*operand: Self*) → Self

Perform an addition operation.

Parameters
operand (*Self-like*) – The container object to add to this.

Returns
result – A copy of this object with the resultant calculations done.

Return type
Self-like

__init__(*wavelength: ndarray*[39], *data: ndarray*[40], *uncertainty: ndarray*[41], *wavelength_unit: Unit*[42], *data_unit: Unit*[43], *mask: ndarray*[44], *flags: ndarray*[45], *header: dict*) → None

Construct a wavelength-aware NDDataArray for arithmetic.

Parameters

- **wavelength** (*ndarray*) – The wavelength of the spectra. The unit of wavelength is typically in microns; but, check the *wavelength_unit* value.
- **data** (*ndarray*) – The data of the spectra cube. The unit of the flux is typically in flam; but, check the *data_unit* value.
- **uncertainty** (*ndarray*) – The uncertainty in the data of the spectra. The unit of the uncertainty is the same as the data value; per *uncertainty_unit*.
- **wavelength_unit** (*Astropy Unit*) – The unit of the wavelength array.
- **data_unit** (*Astropy Unit*) – The unit of the data array.
- **mask** (*ndarray*) – A mask of the data, used to remove problematic areas. Where True, the values of the data is considered masked.
- **flags** (*ndarray*) – Flags of the data. These flags store metadata about the data.
- **header** (*Header, default = None*) – A set of header data describing the data. Note that when saving, this header is written to disk with minimal processing. We highly suggest writing of the metadata to conform to the FITS Header specification as much as possible.

Return type
None

`__justify_arithmetic_operation`(*operand: Self | float*) → bool

Justify operations between two objects is valid.

Operations done between different instances of the Lezargus data structure need to keep in mind the wavelength dependance of the data. We implement simple checks here to formalize if an operation between this object, and some other operand, can be performed.

Parameters

`operand` (*`Self-like or number`*) – The container object that we have an operation to apply with.

Returns

- **justification** (*bool*) – The state of the justification test. If it is True, then the operation can continue, otherwise, False.
- .. *note::* – This function will also raise exceptions upon discovery of incompatible objects. Therefore, the False return case is not really that impactful.

`__mul__`(*operand: Self*) → Self

Perform a multiplication operation.

Parameters

`operand` (*`Self-like`*) – The container object to add to this.

Returns

result – A copy of this object with the resultant calculations done.

Return type

Self-like

`__pow__`(*operand: Self*) → Self

Perform a true division operation.

Parameters

`operand` (*`Self-like`*) – The container object to add to this.

Returns

result – A copy of this object with the resultant calculations done.

Return type

Self-like

`__sub__`(*operand: Self*) → Self

Perform a subtraction operation.

Parameters

`operand` (*`Self-like`*) – The container object to add to this.

Returns

result – A copy of this object with the resultant calculations done.

Return type

Self-like

`__truediv__`(*operand: Self*) → Self

Perform a true division operation.

Parameters

`operand` (*`Self-like`*) – The container object to add to this.

Returns

result – A copy of this object with the resultant calculations done.

Return type

Self-like

`classmethod _read_fits_file`(*filename: str*) → Self

Read in a FITS file into an object.

This is a wrapper around the main FITS class for uniform handling. The respective containers should wrap around this for container-specific handling and should not overwrite this function.

Parameters

`filename` (*`str`*) – The file to read in.

Returns

container – The Lezargus container which was read into the file.

Return type

Self-like

`_write_fits_file`(*filename: str*, *overwrite: bool = False*) → None

Write a FITS object to disk..

This is a wrapper around the main FITS class for uniform handling. The respective containers should wrap around this for container-specific handling and should not overwrite this function.

Parameters

- **`filename`** (*`str`*) – The file to be written out.
- **`overwrite`** (*`bool, default = False`*) – If True, overwrite any file conflicts.

Return type

None

[32] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[33] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[34] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[35] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[36] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[37] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[38] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[39] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[40] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[41] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[42] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[43] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[44] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[45] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

lezargus.container.spectra module

Spectra data container.

This module and class primarily deals with spectral data.

class lezargus.container.spectra.**LezargusSpectra**(*wavelength: ndarray[46], data: ndarray[47], uncertainty: ndarray[48] | None = None, wavelength_unit: str | Unit[49] | None = None, data_unit: str | Unit[50] | None = None, mask: ndarray[51] | None = None, flags: ndarray[52] | None = None, header: Header[53] | None = None*)

Bases: *LezargusContainerArithmetic*

Container to hold spectral data and perform operations on it.

wavelength

The wavelength of the spectra. The unit of wavelength is typically in microns; but, check the *wavelength_unit* value.

Type

ndarray

data

The flux of the spectra. The unit of the flux is typically in flam; but, check the *flux_unit* value.

Type

ndarray

uncertainty

The uncertainty in the flux of the spectra. The unit of the uncertainty is the same as the flux value; per *uncertainty_unit*.

Type

ndarray

wavelength_unit

The unit of the wavelength array.

Type

Astropy Unit

flux_unit

The unit of the flux array.

Type

Astropy Unit

uncertainty_unit

The unit of the uncertainty array. This unit is the same as the flux unit.

Type
Astropy Unit

`mask`

A mask of the flux data, used to remove problematic areas. Where True, the values of the flux is considered mask.

Type
ndarray

`flags`

Flags of the flux data. These flags store metadata about the flux.

Type
ndarray

`header`

The header information, or metadata in general, about the data.

Type
Header

`__init__`(*wavelength: ndarray*[54], *data: ndarray*[55], *uncertainty: ndarray*[56] *| None = None*, *wavelength_unit: str | Unit*[57] *| None = None*, *data_unit: str | Unit*[58] *| None = None*, *mask: ndarray*[59] *| None = None*, *flags: ndarray*[60] *| None = None*, *header: Header*[61] *| None = None*) → None

Instantiate the spectra class.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength of the spectra.
- **`data`** (*`ndarray`*) – The flux of the spectra.
- **`uncertainty`** (*`ndarray, default = None`*) – The uncertainty of the spectra. By default, it is None and the uncertainty value is 0.
- **`wavelength_unit`** (*`Astropy-Unit like, default = None`*) – The wavelength unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`data_unit`** (*`Astropy-Unit like, default = None`*) – The data unit of the spectra. It must be interpretable by the Astropy Units package. If None, the the unit is dimensionless.
- **`mask`** (*`ndarray, default = None`*) – A mask which should be applied to the spectra, if needed.
- **`flags`** (*`ndarray, default = None`*) – A set of flags which describe specific points of data in the spectra.
- **`header`** (*`Header, default = None`*) – A set of header data describing the data. Note that when saving, this header is written to disk with minimal processing. We highly suggest writing of the metadata to conform to the FITS Header specification as much as possible.

classmethod read_fits_file(*filename: str*) → Self

Read a Lezargus spectra FITS file.

We load a Lezargus FITS file from disk. Note that this should only be used for 1-D spectra files.

Parameters

filename (*str*) – The filename to load.

Returns

spectra – The LezargusSpectra class instance.

Return type

Self-like

write_fits_file(*filename: str*, *overwrite: bool = False*) → Self

Write a Lezargus spectra FITS file.

We write a Lezargus FITS file to disk.

Parameters

- **filename** (*str*) – The filename to write to.
- **overwrite** (*bool, default = False*) – If True, overwrite file conflicts.

Return type

None

Module contents

Containers for data.

This module contains the containers for spectral data. We have 4 main classes, broken into different files for ease. There is a parent class which we use to define connivent arithmetic.

[46] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[47] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[48] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[49] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[50] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[51] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[52] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[53] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[54] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[55] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[56] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[57] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[58] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[59] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[60] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[61] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header

lezargus.data package

Module contents

Supplemental data needed for Lezargus is stored here.

This directory comprises of all types of data which is required for Lezargus. No code should be placed in directory. To read and write the data from this directory, see lezargus/library/data.py. We have this information in a __init__.py file just so it is conveniently traced by the documentation build script.

The following conventions are used for the data files:

- **CSV files**

  [We use pipe-delimitated files. Pipes form natural column line for] quick reading. Comments are delimitated by # and serve to be a header for documentation.

- **FITS files**

  [FITS files listed are written by and loaded by the Lezargus] data containers. The FITS files within this directory should only be spectra, image, or cube data. Any other data is likely better in a different file format.

lezargus.library package

Submodules

lezargus.library.array module

Collection of array or image manipulation functions.

If there are any functions which are done on arrays (or anything that is just an array under the hood), we usually group it here. Moreover, functions which would otherwise operate on images are also placed here. As images are just arrays under the hood (and to avoid conflict with /lezargus/container/image.py), image manipulation functions are kept here too.

Note that all of these functions follow the axes convention of indexing being (x, y, lambda). If a cube is not of this shape, then it will likely return erroneous results, but, the functions themselves cannot detect this.

`lezargus.library.array.`**`bin_cube_array_spatially`**(*cube: ndarray*[62], *x_bin: int*, *y_bin: int*, *mode: str = 'add'*) → ndarray[63]

Bin a cube spatially into super pixels.

We only bin the cube in the spatial directions, the spectral direction is not touched.

Parameters

- **cube** (*ndarray*) – The data cube to binned.
- **x_bin** (*int*) – The number of pixels in the x-direction to bin over per super pixel.
- **y_bin** (*int*) – The number of pixels in the y-direction to bin over per super pixel.
- **mode** (*string, default = "add"*) – The mode to combine the data.

- *add* : Add the pixels together.
- *mean* : Use the mean of the pixels.

Returns
binned_image – The data cube after binning.

Return type
ndarray

`lezargus.library.array.`**`bin_image_array`**`(`*image: ndarray*[64], *x_bin: int*, *y_bin: int*, *mode: str = 'add'*`)` → ndarray[65]

Bin an image by using integer super pixels.

A lot of inspiration for this function is from here: https://scipython.com/blog/binning-a-2d-array-in-numpy/

Parameters

- **`image`** (*`ndarray`*) – The input image/array to binned.
- **`x_bin`** (*`int`*) – The number of pixels in the x-direction to bin over per super pixel.
- **`y_bin`** (*`int`*) – The number of pixels in the y-direction to bin over per super pixel.
- **`mode`** (*`string, default = "add"`*) – The mode to combine the data.
  - *add* : Add the pixels together.
  - *mean* : Use the mean of the pixels.

Returns
binned_image – The image/array after binning.

Return type
ndarray

`lezargus.library.array.`**`convolve_cube_by_image_array`**`(`*cube: ndarray*[66], *kernel: ndarray*[67]`)` → ndarray[68]

Convolve the image slices of a 3D cube with a 2D image.

We loop over and convolve image slices of the cube with the provided kernel; we do not try and do an entire 3D convolution of the cube due to memory limitations.

Parameters

- **`cube`** (*`ndarray`*) – The data cube from which we will convolve.
- **`kernel`** (*`ndarray`*) – The image kernel we are using to convolve.

Returns
convolved_cube – The cube, with the image slices convolved by the provided kernel.

[62] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[63] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[64] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[65] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

Return type
ndarray

`lezargus.library.array.`**`rotate_image_array`**(*input_array: ndarray*[69], *rotation: float*) → ndarray[70]

Rotate a 2D image array array.

This function is a connivent wrapper around scipy's function. The array is padded with NaNs so any data outside the original array after rotation is null.

Parameters

- **`input_array`** (*`ndarray`*) – The input array to be rotated.
- **`rotation`** (*`float`*) – The rotation angle, in radians.

Returns
rotated_array – The rotated array/image.

Return type
ndarray

`lezargus.library.array.`**`translate_image_array`**(*input_array: ndarray*[71], *x_shift: float*, *y_shift: float*) → ndarray[72]

Translate a 2D image array array.

This function is a convient wrapper around scipy's function. The array is padded with NaNs so any data outside the original array after translation is null.

Parameters

- **`input_array`** (*`ndarray`*) – The input array to be translated.
- **`x_shift`** (*`float`*) – The number of pixels that the array is shifted in the x-axis.
- **`y_shift`** (*`float`*) – The number of pixels that the array is shifted in the y-axis.

Returns
shifted_array – The shifted array/image.

Return type
ndarray

[66] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[67] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[68] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[69] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[70] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[71] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[72] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

lezargus.library.atmosphere module

Atmospheric functions and other operations.

This file keeps track of all of the functions and computations which deal with the atmosphere.

`lezargus.library.atmosphere.`**`absolute_atmospheric_refraction_function`**`(`*wavelength: ndarray*[73]*, zenith_angle: float, temperature: float, pressure: float, water_pressure: float*`)` → Callable[[ndarray[74]], ndarray[75]]

Compute the absolute atmospheric refraction function.

The absolute atmospheric refraction is not as useful as the relative atmospheric refraction function. To calculate how the atmosphere refracts one's object, use that function instead.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength over which the absolute atmospheric refraction is being computed over, in microns.
- **`zenith_angle`** (*`float`*) – The zenith angle of the sight line, in radians.
- **`temperature`** (*`float`*) – The temperature of the atmosphere, in Kelvin.
- **`pressure`** (*`float`*) – The pressure of the atmosphere, in Pascals.
- **`water_pressure`** (*`float`*) – The partial pressure of water in the atmosphere, Pascals.

Returns

abs_atm_refr_func – The absolute atmospheric refraction function, as an actual callable function.

Return type
Callable

`lezargus.library.atmosphere.`**`airmass`**(*zenith_angle: float | ndarray*[76]) → float | ndarray[77]

Calculate the airmass from the zenith angle.

This function calculates the airmass provided a zenith angle. For most cases the plane-parallel atmosphere method works, and it is what this function uses. However, we also use a more accurate formula for airmass at higher zenith angles (>80 degree), namely from DOI:10.1364/AO.28.004735. We use a weighted average between 75 < z < 80 degrees to allow for a smooth transition.

Parameters
`zenith_angle` (*`float or ndarray`*) – The zenith angle, in radians.

Returns
airmass_ – The airmass. The variable name is to avoid name conflicts.

Return type
float or ndarray

`lezargus.library.atmosphere.`**`gaussian_psf_kernel`**(*shape: tuple*, *x_stddev: float*, *y_stddev: float*, *rotation: float*) → ndarray[78]

Return a 2D Gaussian point spread function convolution kernel.

We normalize the point spread function via the amplitude of the Gaussian function as a whole for maximal precision: volume = 1. We require the input of the shape of the kernel to allow for *x_stddev* and *y_stddev* to be expressed in pixels to keep it general. By definition, the center of the Gaussian kernel is in the center of the array.

Parameters

- **`shape`** (*`tuple`*) – The shape of the 2D kernel, in pixels.
- **`x_stddev`** (*`float`*) – The standard deviation of the Gaussian in the x direction, in pixels.
- **`y_stddev`** (*`float`*) – The standard deviation of the Gaussian in the y direction, in pixels.
- **`rotation`** (*`float`*) – The rotation angle, increasing counterclockwise, in radians.

Returns
gaussian_kernel – The discrete kernel array.

Return type
ndarray

[73] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[74] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[75] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[76] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[77] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[78] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

`lezargus.library.atmosphere.`**`index_of_refraction_dry_air`**`(`*wavelength: ndarray*[79], *pressure: float*, *temperature: float*`)` → ndarray[80]

Calculate the refraction of air of pressured warm dry air.

The index of refraction depends on wavelength, pressure and temperature, we use the updated Edlén equations found in DOI: 10.1088/0026-1394/30/3/004.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength that we are calculating the index of refraction over. This must in microns.
- **`pressure`** (*`float`*) – The pressure of the atmosphere, in Pascals.
- **`temperature`** (*`float`*) – The temperature of the atmosphere, in Kelvin.

Returns

ior_dry_air – The dry air index of refraction.

Return type

ndarray

`lezargus.library.atmosphere.`**`index_of_refraction_ideal_air`**`(`*wavelength: ndarray*[81]`)` → ndarray[82]

Calculate the ideal refraction of air over wavelength.

The index of refraction of air depends slightly on wavelength, we use the updated Edlen equations found in DOI: 10.1088/0026-1394/30/3/004.

Parameters

`wavelength` (*`ndarray`*) – The wavelength that we are calculating the index of refraction over. This must in microns.

Returns

ior_ideal_air – The ideal air index of refraction.

Return type

ndarray

`lezargus.library.atmosphere.`**`index_of_refraction_moist_air`**`(`*wavelength: ndarray*[83], *temperature: float*, *pressure: float*, *water_pressure: float*`)` → ndarray[84]

Calculate the refraction of air of pressured warm moist air.

[79] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[80] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[81] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[82] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

The index of refraction depends on wavelength, pressure, temperature, and humidity, we use the updated Edlen equations found in DOI: 10.1088/0026-1394/30/3/004. We use the partial pressure of water in the atmosphere as opposed to actual humidity.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength that we are calculating the index of refraction over. This must in microns.
- **`temperature`** (*`float`*) – The temperature of the atmosphere, in Kelvin.
- **`pressure`** (*`float`*) – The pressure of the atmosphere, in Pascals.
- **`water_pressure`** (*`float`*) – The partial pressure of water in the atmosphere, Pascals.

Returns

ior_moist_air – The moist air index of refraction.

Return type

ndarray

[83] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[84] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

`lezargus.library.atmosphere.`**`relative_atmospheric_refraction_function`**`(`*wavelength: ndarray*[85]*, reference_wavelength: float, zenith_angle: float, temperature: float, pressure: float, water_pressure: float*`)` → Callable[[ndarray[86]], ndarray[87]]

Compute the relative atmospheric refraction function.

The relative refraction function is the same as the absolute refraction function, however, it is all relative to some specific wavelength.

Parameters

- **`wavelength`** (*`ndarray`*) – The wavelength over which the absolute atmospheric refraction is being computed over, in microns.
- **`reference_wavelength`** (*`float`*) – The reference wavelength which the relative refraction is computed against, in microns.
- **`zenith_angle`** (*`float`*) – The zenith angle of the sight line, in radians.
- **`temperature`** (*`float`*) – The temperature of the atmosphere, in Kelvin.
- **`pressure`** (*`float`*) – The pressure of the atmosphere, in Pascals.
- **`water_pressure`** (*`float`*) – The partial pressure of water in the atmosphere, Pascals.

Returns

rel_atm_refr_func – The absolute atmospheric refraction function, as an actual callable function.

Return type
Callable

lezargus.library.config module

Controls the inputting of configuration files.

This also serves to bring all of the configuration parameters into a more accessible space which other parts of Lezargus can use.

Note these configuration constant parameters are all accessed using capital letters regardless of the configuration file's labels. Moreover, there are constant parameters which are stored here which are not otherwise changeable by the configuration file.

`lezargus.library.config.`**`generate_configuration_file_copy`**(*filename: str*, *overwrite: bool = False*) → None

Generate a copy of the default configuration file to the given location.

Parameters

- **`filename`** (*`str`*) – The pathname or filename where the configuration file should be put to. If it does not have the proper yaml extension, it will be added.
- **`overwrite`** (*`bool, default = False`*) – If the file already exists, overwrite it. If False, it would raise an error instead.

Return type
None

`lezargus.library.config.`**`load_configuration_file`**(*filename: str*) → dict

Load the configuration file and output a dictionary of parameters.

Note configuration files should be flat, there should be no nested configuration parameters.

Parameters
`filename` (*`str`*) – The filename of the configuration file, with the extension. Will raise if the filename is not the correct extension, just as a quick check.

Returns
configuration_dict – The dictionary which contains all of the configuration parameters within it.

Return type
dictionary

[85] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[86] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[87] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

`lezargus.library.config.`**`load_then_apply_configuration`**(*filename: str*) → None

Load a configuration file, then applies it to the entire Lezargus system.

Loads a configuration file and overwrites any overlapping configurations. It writes the configuration to the configuration module for usage throughout the entire program.

Note configuration files should be flat, there should be no nested configuration parameters.

Parameters

`filename` (*`str`*) – The filename of the configuration file, with the extension. Will raise if the filename is not the correct extension, just as a quick check.

Return type

None

lezargus.library.conversion module

Functions to convert things into something else.

Any and all generic conversions (string, units, or otherwise) can be found in here. Extremely standard conversion functions are welcome in here, but, sometimes, a simple multiplication factor is more effective.

`lezargus.library.conversion.`**`convert_to_allowable_fits_header_data_types`**(*input_data: object*) → str | int | float | bool | Undefined

Convert any input into something FITS headers allow.

Per the FITS standard, the allowable data types which values entered in FITS headers is a subset of what Python can do. As such, this function converts any type of reasonable input into something the FITS headers would allow. Note, we mostly do basic checking and conversions. If the object is too exotic, it may cause issues down the line.

In general, only strings, integers, floating point, boolean, and no values are allowed. Astropy usually will handle further conversion from the basic Python types so we only convert up to there.

Parameters

`input_data` (*`object`*) – The input to convert into an allowable FITS header keyword.

Returns

header_output – The output after conversion. Note the None is not actually a None type itself, but Astropy's header None/Undefined type.

Return type

str, int, float, bool, or None

`lezargus.library.conversion.`**`parse_unit_to_astropy_unit`**`(`*unit_string: str*`)` → Unit[88]

Parse a unit string to an Astropy Unit class.

Although for most cases, it is easier to use the Unit instantiation class directly, Astropy does not properly understand some unit conventions so we need to parse them in manually. Because of this, we just build a unified interface for all unit strings in general.

Parameters

`unit_string` (*`str`*) – The unit string to parse into an Astropy unit. If it is None, then we return a dimensionless quantity unit.

Returns

unit_instance – The unit instance after parsing.

Return type

Unit

lezargus.library.data module

Data file functions.

This file deals with the loading in and saving of data files which are in the /data/ directory of Lezargus. Moreover, the contents of the data are accessed using attributes of this module.

`lezargus.library.data.`**`initialize_data_files`**`()` → None

Create all of the data files and instances of classes.

This function creates all of the data objects which represent all of the data and saves it to this module. This must be done in a function, and called by the initialization of the module, to avoid import errors and dependency issues.

Parameters

`None` –

Return type

None

[88] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit

lezargus.library.fits module

FITS file reading, writing, and other manipulations.

`lezargus.library.fits.`**`create_lezargus_fits_header`**(*header: Header*[89], *entries: dict | None = None*) → Header[90]

Create a Lezargus header.

This function creates an ordered Lezargus header from a header containing both Lezargus keywords and non-Lezargus keywords. We only include the relevant headers. WCS header information is also extracted and added as we consider it within our domain even though it does not follow the keyword naming convention (as WCS keywords must follow WCS convention).

Additional header entries may be provided as a last-minute overwrite. We also operate on a copy of the header to prevent conflicts.

Parameters

- **`header`** (*`Astropy Header`*) – The header which the entries will be added to.
- **`entries`** (*`dictionary, default = None`*) – The new entries to the header. By default, None means nothing is to be overwritten at the last minute.

Returns

lezargus_header – The header which Lezargus entries have been be added to. The order of the entries are specified.

Return type

Astropy Header

`lezargus.library.fits.`**`create_wcs_header_from_lezargus_header`**(*header: Header*[91]) → Header[92]

Create WCS header keywords from Lezargus header.

See the FITS standard for more information.

Parameters

`header` (*`Header`*) – The Lezargus header from which we will derive a WCS header from.

Returns

wcs_header – The WCS header.

Return type

Header

`lezargus.library.fits.`**`read_fits_header`**(*filename: str*, *extension: int | str = 0*) → Header[93]

Read a FITS file header.

[89] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[90] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[91] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[92] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header

This reads the header of fits files only. This should be used only if there is no data. Really, this is just a wrapper around Astropy, but it is made for consistency and to avoid the usage of the convince functions.

Parameters

- **`filename`** (*`str`*) – The filename that the fits image file is at.
- **`extension`** (*`int or str, default = 0`*) – The fits extension that is desired to be opened.

Returns

header – The header of the fits file.

Return type

Astropy Header

`lezargus.library.fits.`**`read_lezargus_fits_file`**(*filename: str*) → tuple[astropy.io.fits.header.Header[94], numpy.ndarray[95], numpy.ndarray[96], numpy.ndarray[97], astropy.units.core.Unit[98], astropy.units.core.Unit[99], numpy.ndarray[100], numpy.ndarray[101]]

Read in a Lezargus fits file.

This function reads in a Lezargus FITS file and parses it based on the convention of Lezargus. See TODO for the specification. However, we do not construct the actual classes here and instead leave that to the class reader and writers of the container themselves so we can reuse error reporting code there.

In general, it is advisable to use the reading and writing class functions of the container instance you want.

Parameters

`filename` (*`str`*) – The filename of the FITS file to read.

Returns

- **header** (*Header*) – The header of the Lezargus FITS file.
- **wavelength** (*ndarray*) – The wavelength information of the file.
- **data** (*ndarray*) – The data array of the Lezargus FITS file.
- **uncertainty** (*ndarray*) – The uncertainty in the data.
- **wavelength_unit** (*Unit*) – The unit of the wavelength array.
- **data_unit** (*Unit*) – The unit of the data.
- **mask** (*ndarray*) – The mask of the data.
- **flags** (*ndarray*) – The noted flags for each of the data points.

[93] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header

`lezargus.library.fits.`**`write_lezargus_fits_file`**(*filename: str*, *header: Header*[102], *wavelength: ndarray*[103], *data: ndarray*[104], *uncertainty: ndarray*[105], *wavelength_unit: Unit*[106], *data_unit: Unit*[107], *uncertainty_unit: Unit*[108], *mask: ndarray*[109], *flags: ndarray*[110], *overwrite: bool = False*) → None

Write to a Lezargus fits file.

This function reads in a Lezargus FITS file and parses it based on the convention of Lezargus. See TODO for the specification. However, we do not construct the actual classes here and instead leave that to the class reader and writers of the container themselves so we can reuse error reporting code there.

In general, it is advisable to use the reading and writing class functions of the container instance you want.

Parameters

- **`filename`** (*`str`*) – The filename of the FITS file to write to.
- **`header`** (*`Header`*) – The header of the Lezargus FITS file.
- **`wavelength`** (*`ndarray`*) – The wavelength information of the file.
- **`data`** (*`ndarray`*) – The data array of the Lezargus FITS file.
- **`uncertainty`** (*`ndarray`*) – The uncertainty in the data.
- **`wavelength_unit`** (*`Unit`*) – The unit of the wavelength array.
- **`data_unit`** (*`Unit`*) – The unit of the data.
- **`uncertainty_unit`** (*`Unit`*) – The unit of the uncertainty of the data.
- **`mask`** (*`ndarray`*) – The mask of the data.
- **`flags`** (*`ndarray`*) – The noted flags for each of the data points.
- **`overwrite`** (*`bool, default = False`*) – If True, overwrite the file upon conflicts.

Return type

None

[94] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[95] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[96] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[97] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[98] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[99] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[100] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[101] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

lezargus.library.flags module

Functions to deal with quality flag tracking, masks, and other information.

Flags and masks is the primary backbone of how the quality of data can be communicated. Here, we package all of the different functions regarding flags and masks.

`lezargus.library.flags.`**`combine_flags`**`(`**flags: ndarray*[111]`)` → ndarray[112]

Combine two or more flag arrays.

The flag values here follow the Lezargus convention, see [[TODO]].

Parameters

`*flags` (*`ndarray`*) – The set of flags to combine.

Returns

combined_flags – The combined mask.

Return type

ndarray

`lezargus.library.flags.`**`combine_masks`**`(`**masks: ndarray*[113]`)` → ndarray[114]

Combine two or more masks.

The masks follow the Numpy convention; a True value means that the data is considered masked.

Parameters

`*masks` (*`ndarray`*) – The set of masks to combine.

Returns

combined_mask – The combined mask.

Return type

ndarray

`lezargus.library.flags.`**`reduce_flags`**`(`*flag_array: ndarray*[115]`)` → ndarray[116]

Reduce the flag value to the minimum it can be.

Flags, based on the Lezargus convention (see [[TODO]]), rely on the prime factors to determine the total flags present. As multiplication is how flags propagate, the value can get big quickly. We reduce the values within a flag array to the lowest it can be.

[102] https://docs.astropy.org/en/stable/io/fits/api/headers.html#astropy.io.fits.Header
[103] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[104] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[105] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[106] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[107] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[108] https://docs.astropy.org/en/stable/api/astropy.units.Unit.html#astropy.units.Unit
[109] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[110] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[111] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[112] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[113] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[114] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

Parameters
: `flag_array` (*ndarray*) – The flag array to be reduced into its lowest form.

Returns
: **lowest_flag_array** – The flags, reduced to the lowest value.

Return type
: ndarray

lezargus.library.hint module

A collection of types for linting.

These are redefinitions and wrapping variables for type hints. Its purpose is for just uniform hinting types.

This should only be used for types which are otherwise not native and would require an import, including the typing module. The whole point of this is to be a central collection of types for the purpose of type hinting.

This module should never be used for anything other than hinting. Use proper imports to access these classes. Otherwise, you will likely get circular imports and other nasty things.

lezargus.library.logging module

Error, warning, and logging functionality pertinent to Lezargus.

Use the functions here when logging or issuing errors or other information.

exception `lezargus.library.logging.`**`AccuracyError`**

Bases: *LezargusError*

An error for inaccurate results.

This error is used when some elements of the simulation or data reduction would yield very undesirable results. This class should only be used for cases where the results would be quite wrong for most cases; say, much greater +/- 10%. This is just a rule of thumb. For accuracy issues much tamer and closer to the rule of thumb, use AccuracyWarning instead.

exception `lezargus.library.logging.`**`AccuracyWarning`**

Bases: *LezargusWarning*

A warning for inaccurate results.

This warning is used when some elements of the simulation or data reduction would yield less than desireable results. This class should only be used for cases where the results would be slightly wrong for most cases; say, on the order of +/- 10%. This is just a rule of thumb. For accuracy issues much larger, use AccuracyError instead.

[115] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[116] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

exception `lezargus.library.logging.`**`AlgorithmWarning`**

Bases: *`LezargusWarning`*

A warning for issues with algorithms or methods.

This warning should be used when something went wrong with an algorithm. Examples include when continuing would lead to inaccurate results, or when alternative methods must be used, or when predicted execution time would be slow. This warning should be used in conjunction with other warnings to give a full picture of the issue.

exception `lezargus.library.logging.`**`ArithmeticalError`**

Bases: *`LezargusError`*

An error to be used when undefined arithmetic operations are attempted.

This error is to be used when any arithmetic functions are being attempted which do not have a definition. The most common use case for this error is doing operations on incompatible Lezargus containers. Note, it is named ArithmeticalError to avoid a name conflict with the built-in ArithmeticError.

class `lezargus.library.logging.`**`ColoredLogFormatter`**(*message_format: str*, *date_format: str*, *color_hex_dict: dict[slice(<class 'int'>, <class 'str'>, None)] | None = None*)

Bases: `Formatter`

Use this formatter to have colors.

`message_format`

The message format, passed directly to the logger formatter after the color keys are added.

Type

str

`date_format`

The date format, passed directly to the logger formatter.

Type

str

`color_formatting`

The formatting for the color.

Type

dict

`__convert_color_hex_to_ansi_escape`() → str

Convert a hex code to a ANSI escape code.

Parameters

`color_hex` (*`str`*) – The HEX code of the color, including the # symbol.

Returns

color_ansi_escape – The ANSI escape code for the color.

Return type
str

`__init__`(*message_format: str*, *date_format: str*, *color_hex_dict: dict[slice(<class 'int'>, <class 'str'>, None)] | None = None*) → None

Initialize the color formatter.

Parameters

- **`message_format`** (*`str`*) – The message format, passed directly to the logger formatter after the color keys are added.
- **`date_format`** (*`str`*) – The date format, passed directly to the logger formatter.
- **`color_hex_dict`** (*`dict, default = None`*) – The dictionary containing the color pairings between logging levels and its actual color. It should be a {level_number:hex_color} dictionary.

Return type
None

`format`(*record: LogRecord*) → str

Format a log record.

The name of this function cannot be helped as it is required for the Python logging module.

Parameters
`record` (*`LogRecord`*) – The record to format.

Returns
formatted_record – The formatted string.

Return type
str

exception `lezargus.library.logging.`**`CommandLineError`**

Bases: *`LezargusError`*

An error used for an error with the command-line.

This error is used when the entered command-line command or its options are not correct.

exception `lezargus.library.logging.`**`ConfigurationError`**

Bases: *`LezargusError`*

An error used for an error with the configuration file.

This error is to be used when the configuration file is wrong. There is a specific expectation for how configuration files and configuration parameters are structures are defined.

exception `lezargus.library.logging.`**`ConfigurationWarning`**

Bases: *`LezargusWarning`*

A warning for inappropriate configurations.

This warning is to be used when the configuration file is wrong. There is a specific expectation for how configuration files and configuration parameters are structures are defined.

exception `lezargus.library.logging.`**`DataLossWarning`**

Bases: *`LezargusWarning`*

A warning to caution on data loss.

This warning is used when something is being done which might result in a loss of important data, for example, because a file is not saved or only part of a data file is read.

exception `lezargus.library.logging.`**`DevelopmentError`**

Bases: *`LezargusBaseError`*

An error used for a development error.

This is an error where the development of Lezargus is not correct and something is not coded based on the expectations of the software itself. This is not the fault of the user.

exception `lezargus.library.logging.`**`DevelopmentWarning`**

Bases: *`LezargusWarning`*

A warning used for a development issue.

This is a warning where the development of Lezargus is not correct and something is not coded based on the expectations of the software itself. This is not the fault of the user.

exception `lezargus.library.logging.`**`DirectoryError`**

Bases: *`LezargusError`*

An error used for directory issues.

If there are issues with directories, use this error.

exception `lezargus.library.logging.`**`ElevatedError`**

Bases: *`LezargusError`*

An error used when elevating warnings or errors to critical level.

Only to be used when elevating via the configuration property.

exception `lezargus.library.logging.`**`ExpectedCaughtError`**

Bases: *`LezargusBaseError`*

An error used when raising an error to be caught later is needed.

This error should only be used when an error is needed to be raised which will be caught later. The user should not see this error at all as any time it is used, it should be caught. This name also conveniently provides an obvious and unique error name.

exception `lezargus.library.logging.`**`FileError`**

Bases: *`LezargusError`*

An error used for file issues.

If there are issues with files, use this error. This error should not be used in cases where the problem is because of an incorrect format of the file (other than corruption).

exception `lezargus.library.logging.`**`FileWarning`**

Bases: *`LezargusWarning`*

A warning used for file and permission issues which are not fatal.

If there are issues with files, use this warning. However, unlike the error version FileError, this should be used when the file issue is a case considered and is recoverable. This warning should not be used in cases where the problem is because of an incorrect format of the file (other than corruption).

exception `lezargus.library.logging.`**`InputError`**

Bases: *`LezargusError`*

An error used for issues with input parameters or data.

This is the error to be used when the inputs to a function are not valid and do not match the expectations of that function.

exception `lezargus.library.logging.`**`InputWarning`**

Bases: *`LezargusWarning`*

A warning for a weird input.

This warning is used when the input of a function or a field is not expected, but may be able to be handled.

exception `lezargus.library.logging.`**`LezargusBaseError`**

Bases: `BaseException`

The base inheriting class which for all Lezargus errors.

This is for exceptions that should never be caught and should bring everything to a halt.

exception `lezargus.library.logging.`**`LezargusError`**

Bases: `Exception`

The main inheriting class which all Lezargus errors use as their base.

This is done for ease of error handling and is something that can and should be managed.

exception `lezargus.library.logging.`**`LezargusWarning`**

Bases: `UserWarning`

The main inheriting class which all Lezargus warnings use as their base.

The base warning class which all of the other Lezargus warnings are derived from.

exception `lezargus.library.logging.`**`LogicFlowError`**

Bases: *`LezargusBaseError`*

An error used for an error in the flow of program logic.

This is an error to ensure that the logic does not flow to a point to a place where it is not supposed to. This is helpful in making sure changes to the code do not screw up the logical flow of the program.

exception `lezargus.library.logging.`**`MemoryFullWarning`**

Bases: *`LezargusWarning`*

A warning for when there is not enough volatile memory.

This warning is used when the program detects that the machine does not have enough memory to proceed with a given process and so it tries an alternative method to do a similar calculation. We use the name MemoryFullWarning to avoid a name collision with MemoryWarning and to be a little more specific about what the issue is.

exception `lezargus.library.logging.`**`NotSupportedError`**

Bases: *`LezargusBaseError`*

An error used for something which is beyond the scope of work.

This is an error to be used when what is trying to be done does not seem reasonable. Usually warnings are the better thing for this but this error is used when the assumptions for reasonability guided development and what the user is trying to do is not currently supported by the software.

exception `lezargus.library.logging.`**`OutOfOrderError`**

Bases: *`LezargusError`*

An error used when things are done out-of-order.

This error is used when something is happening out of the expected required order. This order being in place for specific publicly communicated reasons.

exception `lezargus.library.logging.`**`ReadOnlyError`**

Bases: *`LezargusError`*

An error used for problems with read-only files.

If the file is read-only and it needs to be read, use FileError. This error is to be used only when variables or files are assumed to be read only, this error should be used to enforce that notion.

exception `lezargus.library.logging.`**`UndiscoveredError`**

Bases: *`LezargusBaseError`*

An error used for an unknown error.

This is an error used in cases where the source of the error has not been determined and so a more helpful error message or mitigation strategy cannot be devised.

`lezargus.library.logging.`**`add_file_logging_handler`**(*filename: str*, *log_level: int = 10*) → None

Add a stream handler to the logging infrastructure.

Parameters

- **`filename`** (*`str`*) – The filename path where the log file will be saved to.
- **`log_level`** (*`int`*) – The logging level for this handler.

Return type

None

`lezargus.library.logging.`**`add_stream_logging_handler`**(*stream: object*, *log_level: int = 10*, *use_color: bool = True*) → None

Add a stream handler to the logging infrastructure.

Parameters

- **`stream`** (*`Any`*) – The stream where the logs will write to.
- **`log_level`** (*`int`*) – The logging level for this handler.
- **`use_color`** (*`bool`*) – If True, use colored log messaged based on the configuration file parameters.

Return type

None

`lezargus.library.logging.`**`critical`**(*critical_type:* LezargusError, *message: str*) → None

Log a critical error and raise.

Use this for issues which are more serious than warnings and should raise/throw an exception. The main difference between critical and error for logging is that critical will also raise the exception as error will not and the program will attempt to continue.

This is a wrapper around the critical function to standardize it for Lezargus.

Parameters

- **`critical_type`** (`LezargusError`) – The class of the critical exception error which will be used and raised.
- **`message`** (*`str`*) – The critical error message.

Return type

None

`lezargus.library.logging.`**`debug`**(*message: str*) → None

Log a debug message.

This is a wrapper around the debug function to standardize it for Lezargus.

Parameters

`message` (*`str`*) – The debugging message.

Return type

None

`lezargus.library.logging.`**`error`**(*error_type:* LezargusError, *message: str*, *elevate: bool | None = None*) → None

Log an error message, do not raise.

Use this for issues which are more serious than warnings but do not result in a raised exception.

This is a wrapper around the error function to standardize it for Lezargus.

Parameters

- **`error_type`** (`LezargusError`) – The class of the error which will be used.
- **`message`** (*`str`*) – The error message.
- **`elevate`** (*`bool, default = None`*) – If True, always elevate the error to a critical issue. By default, use the configuration value.

Return type
None

`lezargus.library.logging.`**`info`**`(`*message: str*`)` → None

Log an informational message.

This is a wrapper around the info function to standardize it for Lezargus.

Parameters
`message` (*`str`*) – The informational message.

Return type
None

`lezargus.library.logging.`**`terminal`**`()` → None

Terminal error function which is used to stop everything.

Parameters
`None` –

Return type
None

`lezargus.library.logging.`**`update_global_minimum_logging_level`**`(`*log_level: int = 10*`)` → None

Update the logging level of this module.

This function updates the minimum logging level which is required for a log record to be recorded. Handling each single logger handler is really unnecessary.

Parameters
`log_level` (*`int, default = logging.DEBUG`*) – The log level which will be set as the minimum level.

Return type
None

`lezargus.library.logging.`**`warning`**`(`*warning_type:* LezargusWarning, *message: str*, *elevate: bool | None = None*`)` → None

Log a warning message.

This is a wrapper around the warning function to standardize it for Lezargus.

Parameters

- **`warning_type`** (`LezargusWarning`) – The class of the warning which will be used.
- **`message`** (*`str`*) – The warning message.
- **`elevate`** (*`bool, default = None`*) – If True, always elevate the warning to a critical issue. By default, use the configuration value.

Return type
None

lezargus.library.path module

Functions to deal with different common pathname manipulations.

As Lezargus is going to be cross platform, this is a nice abstraction.

`lezargus.library.path.`**`get_directory`**(*pathname: str*) → str

Get the directory from the pathname without the file or the extension.

Parameters
: **`pathname`** (*`str`*) – The pathname which the directory will be extracted.

Returns
: **directory** – The directory which belongs to the pathname.

Return type
: str

`lezargus.library.path.`**`get_file_extension`**(*pathname: str*) → str

Get the file extension only from the pathname.

Parameters
: **`pathname`** (*`str`*) – The pathname which the file extension will be extracted.

Returns
: **extension** – The file extension only.

Return type
: str

`lezargus.library.path.`**`get_filename_with_extension`**(*pathname: str*) → str

Get the filename from the pathname with the file extension.

Parameters
: **`pathname`** (*`str`*) – The pathname which the filename will be extracted.

Returns
: **filename** – The filename with the file extension.

Return type
: str

`lezargus.library.path.`**`get_filename_without_extension`**(*pathname: str*) → str

Get the filename from the pathname without the file extension.

Parameters
: **`pathname`** (*`str`*) – The pathname which the filename will be extracted.

Returns
: **filename** – The filename without the file extension.

Return type
: str

`lezargus.library.path.`**`get_most_recent_filename_in_directory`**(*directory: str, extension: str | list = None, recursive: bool = False, recency_function: Callable[[str], float] = None*) → str

Get the most recent filename from a directory.

Because of issues with different operating systems having differing issues with storing the creation time of a file, this function sorts based off of modification time.

Parameters

- **`directory`** (*`str`*) – The directory by which the most recent file will be derived from.
- **`extension`** (*`str or list, default = None`*) – The extension by which to filter for. It is often the case that some files are created but the most recent file of some type is desired. Only files which match the included extensions will be considered.
- **`recursive`** (*`bool, default = False`*) – If True, the directory is searched recursively for the most recent file based on the recency function.
- **`recency_function`** (*`callable, default = None`*) – A function which, when provided, provides a sorting index for a given filename. This is used when the default sorting method (modification time) is not desired and a custom function can be provided here. The larger the value returned by this function, the more "recent" a given file will be considered to be.

Returns

recent_filename – The filename of the most recent file, by modification time, in the directory.

Return type

str

`lezargus.library.path.`**`merge_pathname`**(*directory: str | list = None, filename: str | None = None, extension: str | None = None*) → str

Join the directories, filenames, and file extensions into one pathname.

Parameters

- **`directory`** (*`str or list, default = None`*) – The directory(s) which is going to be used. If it is a list, then the paths within it are combined.
- **`filename`** (*`str, default = None`*) – The filename that is going to be used for path construction.
- **`extension`** (*`str, default = None`*) – The filename extension that is going to be used.

Returns
: **pathname** – The combined pathname.

Return type
: str

`lezargus.library.path.`**`split_pathname`**`(`*pathname: str*`)` → tuple[str, str, str]

Return a pathname split into its components.

This is a wrapper function around the more elementary functions *get_directory*, *get_filename_without_extension*, and *get_file_extension*.

Parameters
: **`pathname`** (*`str`*) – The combined pathname which to be split.

Returns

- **directory** (*str*) – The directory which was split from the pathname.
- **filename** (*str*) – The filename which was split from the pathname.
- **extension** (*str*) – The filename extension which was split from the pathname.

lezargus.library.wrapper module

Function wrappers.

We borrow a lot of functions from different packages; however, for a lot of them, we build wrappers around them to better integrate them into our package provided its own idiosyncrasies.

`lezargus.library.wrapper.`**`blackbody_function`**`(`*temperature: float*`)` → Callable[[ndarray[117]], ndarray[118]]

Return a callable blackbody function for a given temperature.

This function is a wrapper around the Astropy blackbody model. This wrapper exists to remove the unit baggage of the original Astropy blackbody model so that we can stick to the convention of Lezargus.

Parameters
: **`temperature`** (*`float`*) – The blackbody temperature, in Kelvin.

Returns
: **blackbody** – The blackbody function, the wavelength callable is in microns. The return units are in FLAM/sr.

Return type
: Callable

`lezargus.library.wrapper.`**`cubic_interpolate_1d_function`**`(`*x: ndarray[119], y: ndarray[120]*`)` → Callable[[ndarray[121]], ndarray[122]]

[117] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[118] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

Return a wrapper around Scipy's Cubic interpolation.

Parameters

- **x** (*ndarray*) – The x data to interpolate over.
- **y** (*ndarray*) – The y data to interpolate over.

Returns

interpolate_function – The interpolation function of the data.

Return type

Callable

Module contents

Common routines which are important functions of Lezargus.

Submodules

lezargus.__main__ module

Just a small hook for the main execution.

This section parses arguments which is then passed to execution to do exactly as expected by the commands.

lezargus.__version__ module

The version of the software. Use *hatch* to upgrade this version number.

Please do not manually edit this value.

Module contents

Lezargus: The software package related to IRTF SPECTRE.

[119] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[120] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[121] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray
[122] https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html#numpy.ndarray

CHAPTER TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

l

lezargus, 41
lezargus.__main__, 41
lezargus.__version__, 41
lezargus.container, 14
lezargus.container.cube, 1
lezargus.container.image, 4
lezargus.container.mosaic, 7
lezargus.container.parent, 7
lezargus.container.spectra, 12
lezargus.data, 15
lezargus.library, 41
lezargus.library.array, 15
lezargus.library.atmosphere, 18
lezargus.library.config, 23
lezargus.library.conversion, 24
lezargus.library.data, 25
lezargus.library.fits, 26
lezargus.library.flags, 29
lezargus.library.hint, 30
lezargus.library.logging, 30
lezargus.library.path, 38
lezargus.library.wrapper, 40

INDEX

Non-alphabetical

`__add__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 9
`__convert_color_hex_to_ansi_escape()` (*lezargus.library.logging.ColoredLogFormatter method*), 31
`__init__()` (*lezargus.container.cube.LezargusCube method*), 3
`__init__()` (*lezargus.container.image.LezargusImage method*), 5
`__init__()` (*lezargus.container.mosaic.LezargusMosaic method*), 7
`__init__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 9
`__init__()` (*lezargus.container.spectra.LezargusSpectra method*), 13
`__init__()` (*lezargus.library.logging.ColoredLogFormatter method*), 32
`__justify_arithmetic_operation()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 9
`__mul__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 10
`__pow__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 10
`__sub__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 10
`__truediv__()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 10
`_read_fits_file()` (*lezargus.container.parent.LezargusContainerArithmetic class method*), 11
`_write_fits_file()` (*lezargus.container.parent.LezargusContainerArithmetic method*), 11

A

`absolute_atmospheric_refraction_function()` (*in module lezargus.library.atmosphere*), 18
`AccuracyError`, 30
`AccuracyWarning`, 30
`add_file_logging_handler()` (*in module lezargus.library.logging*), 35
`add_stream_logging_handler()` (*in module lezargus.library.logging*), 35
`airmass()` (*in module lezargus.library.atmosphere*), 19
`AlgorithmWarning`, 30
`ArithmeticalError`, 31

B

`bin_cube_array_spatially()` (*in module lezargus.library.array*), 15
`bin_image_array()` (*in module lezargus.library.array*), 16
`blackbody_function()` (*in module lezargus.library.wrapper*), 40

C

`color_formatting` (*lezargus.library.logging.ColoredLogFormatter attribute*), 31

ColoredLogFormatter (*class in lezargus.library.logging*), 31
combine_flags() (*in module lezargus.library.flags*), 29
combine_masks() (*in module lezargus.library.flags*), 29
CommandLineError, 32
ConfigurationError, 32
ConfigurationWarning, 32
convert_to_allowable_fits_header_data_types() (*in module lezargus.library.conversion*), 24
convolve_cube_by_image_array() (*in module lezargus.library.array*), 16
create_lezargus_fits_header() (*in module lezargus.library.fits*), 26
create_wcs_header_from_lezargus_header() (*in module lezargus.library.fits*), 26
critical() (*in module lezargus.library.logging*), 36
cubic_interpolate_1d_function() (*in module lezargus.library.wrapper*), 40

D

data (*lezargus.container.cube.LezargusCube attribute*), 2
data (*lezargus.container.image.LezargusImage attribute*), 4
data (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
data (*lezargus.container.spectra.LezargusSpectra attribute*), 12
data_unit (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
DataLossWarning, 33
date_format (*lezargus.library.logging.ColoredLogFormatter attribute*), 31
debug() (*in module lezargus.library.logging*), 36
DevelopmentError, 33
DevelopmentWarning, 33
DirectoryError, 33

E

ElevatedError, 33
error() (*in module lezargus.library.logging*), 36
ExpectedCaughtError, 33

F

FileError, 33
FileWarning, 33
flags (*lezargus.container.cube.LezargusCube attribute*), 2
flags (*lezargus.container.image.LezargusImage attribute*), 5
flags (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
flags (*lezargus.container.spectra.LezargusSpectra attribute*), 13
flux_unit (*lezargus.container.cube.LezargusCube attribute*), 2
flux_unit (*lezargus.container.image.LezargusImage attribute*), 5
flux_unit (*lezargus.container.spectra.LezargusSpectra attribute*), 12
format() (*lezargus.library.logging.ColoredLogFormatter method*), 32

G

gaussian_psf_kernel() (*in module lezargus.library.atmosphere*), 19
generate_configuration_file_copy() (*in module lezargus.library.config*), 23
get_directory() (*in module lezargus.library.path*), 38

`get_file_extension()` (*in module lezargus.library.path*), 38
`get_filename_with_extension()` (*in module lezargus.library.path*), 38
`get_filename_without_extension()` (*in module lezargus.library.path*), 38
`get_most_recent_filename_in_directory()` (*in module lezargus.library.path*), 38

H

`header` (*lezargus.container.cube.LezargusCube attribute*), 2
`header` (*lezargus.container.image.LezargusImage attribute*), 5
`header` (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 9
`header` (*lezargus.container.spectra.LezargusSpectra attribute*), 13

I

`index_of_refraction_dry_air()` (*in module lezargus.library.atmosphere*), 19
`index_of_refraction_ideal_air()` (*in module lezargus.library.atmosphere*), 20
`index_of_refraction_moist_air()` (*in module lezargus.library.atmosphere*), 20
`info()` (*in module lezargus.library.logging*), 37
`initialize_data_files()` (*in module lezargus.library.data*), 25
`InputError`, 34
`InputWarning`, 34

L

`lezargus`
  `module`, 41
`lezargus.__main__`
  `module`, 41
`lezargus.__version__`
  `module`, 41
`LezargusBaseError`, 34
`lezargus.container`
  `module`, 14
`LezargusContainerArithmetic` (*class in lezargus.container.parent*), 7
`lezargus.container.cube`
  `module`, 1
`lezargus.container.image`
  `module`, 4
`lezargus.container.mosaic`
  `module`, 7
`lezargus.container.parent`
  `module`, 7
`lezargus.container.spectra`
  `module`, 12
`LezargusCube` (*class in lezargus.container.cube*), 1
`lezargus.data`
  `module`, 15
`LezargusError`, 34
`LezargusImage` (*class in lezargus.container.image*), 4
`lezargus.library`
  `module`, 41

lezargus.library.array
    module, 15
lezargus.library.atmosphere
    module, 18
lezargus.library.config
    module, 23
lezargus.library.conversion
    module, 24
lezargus.library.data
    module, 25
lezargus.library.fits
    module, 26
lezargus.library.flags
    module, 29
lezargus.library.hint
    module, 30
lezargus.library.logging
    module, 30
lezargus.library.path
    module, 38
lezargus.library.wrapper
    module, 40
LezargusMosaic (*class in lezargus.container.mosaic*), 7
LezargusSpectra (*class in lezargus.container.spectra*), 12
LezargusWarning, 34
load_configuration_file() (*in module lezargus.library.config*), 23
load_then_apply_configuration() (*in module lezargus.library.config*), 23
LogicFlowError, 34

M

mask (*lezargus.container.cube.LezargusCube attribute*), 2
mask (*lezargus.container.image.LezargusImage attribute*), 5
mask (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
mask (*lezargus.container.spectra.LezargusSpectra attribute*), 13
MemoryFullWarning, 34
merge_pathname() (*in module lezargus.library.path*), 39
message_format (*lezargus.library.logging.ColoredLogFormatter attribute*), 31
module
    lezargus, 41
    lezargus.__main__, 41
    lezargus.__version__, 41
    lezargus.container, 14
    lezargus.container.cube, 1
    lezargus.container.image, 4
    lezargus.container.mosaic, 7
    lezargus.container.parent, 7
    lezargus.container.spectra, 12
    lezargus.data, 15

lezargus.library, 41
lezargus.library.array, 15
lezargus.library.atmosphere, 18
lezargus.library.config, 23
lezargus.library.conversion, 24
lezargus.library.data, 25
lezargus.library.fits, 26
lezargus.library.flags, 29
lezargus.library.hint, 30
lezargus.library.logging, 30
lezargus.library.path, 38
lezargus.library.wrapper, 40

N

NotSupportedError, 35

O

OutOfOrderError, 35

P

parse_unit_to_astropy_unit() (*in module lezargus.library.conversion*), 25

R

read_fits_file() (*lezargus.container.cube.LezargusCube class method*), 3
read_fits_file() (*lezargus.container.image.LezargusImage class method*), 6
read_fits_file() (*lezargus.container.spectra.LezargusSpectra class method*), 14
read_fits_header() (*in module lezargus.library.fits*), 26
read_lezargus_fits_file() (*in module lezargus.library.fits*), 27
ReadOnlyError, 35
reduce_flags() (*in module lezargus.library.flags*), 29
relative_atmospheric_refraction_function() (*in module lezargus.library.atmosphere*), 21
rotate_image_array() (*in module lezargus.library.array*), 17

S

split_pathname() (*in module lezargus.library.path*), 40

T

terminal() (*in module lezargus.library.logging*), 37
translate_image_array() (*in module lezargus.library.array*), 17

U

uncertainty (*lezargus.container.cube.LezargusCube attribute*), 2
uncertainty (*lezargus.container.image.LezargusImage attribute*), 5
uncertainty (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
uncertainty (*lezargus.container.spectra.LezargusSpectra attribute*), 12
uncertainty_unit (*lezargus.container.cube.LezargusCube attribute*), 2
uncertainty_unit (*lezargus.container.image.LezargusImage attribute*), 5
uncertainty_unit (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8

uncertainty_unit (*lezargus.container.spectra.LezargusSpectra attribute*), 12
UndiscoveredError, 35
update_global_minimum_logging_level() (*in module lezargus.library.logging*), 37

W

warning() (*in module lezargus.library.logging*), 37
wavelength (*lezargus.container.cube.LezargusCube attribute*), 2
wavelength (*lezargus.container.image.LezargusImage attribute*), 4
wavelength (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
wavelength (*lezargus.container.spectra.LezargusSpectra attribute*), 12
wavelength_unit (*lezargus.container.cube.LezargusCube attribute*), 2
wavelength_unit (*lezargus.container.image.LezargusImage attribute*), 5
wavelength_unit (*lezargus.container.parent.LezargusContainerArithmetic attribute*), 8
wavelength_unit (*lezargus.container.spectra.LezargusSpectra attribute*), 12
write_fits_file() (*lezargus.container.cube.LezargusCube method*), 3
write_fits_file() (*lezargus.container.image.LezargusImage method*), 6
write_fits_file() (*lezargus.container.spectra.LezargusSpectra method*), 14
write_lezargus_fits_file() (*in module lezargus.library.fits*), 28