pygeobase Documentation

Release

TU Wien

Apr 04, 2017

Contents

This is the documentation of **pygeobase**.

The pygeobase package implements base class definitions for the I/O interface used in pytesmo, pynetCF, and other packages.

CHAPTER 1

Usage

The Abstract base classes in this package are used to provide a consistent interface for reading various kinds of data.

Image datasets

When we talk about a image dataset we are generally talking about a dataset that can be represented through one or several two dimensional arrays. Such a dataset might consist of multiple layers or bands. It can have implicit or explicit geolocation information attached. In the simplest case all the datapoints of a image are referenced to the the same time. But we can also envision a reference timestamp for a image with a layer of exact time stamps for each observation.

The *pygeobase.io_base.ImageBase* implements the reading and writing of a single file whereas *pygeobase.io_base.MultiTemporalImageBase* is responsible for building the filename for a reference timestamp and using the ImageBase class for the io. In this way any number of underlying file formats can be supported.

Figure 1 shows *pygeobase.io_base.ImageBase* which is the abstract base class for implementing a reader for a single image linked to one file on disk. The `read`, `write`, `flush` and `close` methods have to be implemented. For reading from a dataset it is generally enough to implement the `read` and `close` methods and use dummy methods for `write` and `flush`. The `read` method must return a *pygeobase.object_base.Image* instance.

Fig. 1.1: Figure 1

A implemented class for the ImageBase can then be used in *pygeobase.io_base.MultiTemporalImageBase*. This class models a dataset consisting of several files on disk. Each file is linked to a reference timestamp from which the filename can be built. *pygeobase.io_base.MultiTemporalImageBase* can be used directly when configured correctly. Configuration means setting the `fname_templ` and the `datetime_format` so that it fits to the dataset. If the single files are stored in subfolders by e.g. month or day then the keyword `subpath_templ` can be used to specify that. Please see *pygeobase.io_base.MultiTemporalImageBase* for detailed information about each keyword.

Example for implementing a new image dataset

Let's imagine we have a regular daily dataset stored on a global regular grid of 0.1 degrees. The folder structure and filenames of the dataset are e.g.

- /2015/01/dataset_2015-01-01.dat
- /2015/01/dataset_2015-01-02.dat
- ...
- /2015/02/dataset_2015-02-01.dat

For simplicities sake lets assume that the dat files are just pickled python dictionaries.

We could now write a new class based on *pygeobase.io_base.ImageBase* that reads one of these files:

```
from pygeobase.object_base import Image
from pygeobase.io_base import ImageBase
import pygeogrids.grids as grids
import pickle


class PickleImg(ImageBase):

    def __init__(self, filename, mode='r'):
        super(PickleImg, self).__init__(filename, mode=mode)
        self.grid = grids.genreg_grid(1, 1)

    def read(self, timestamp=None):

        data = pickle.load(self.filename)
        metadata = {'Type': 'pickle'}

        return Image(self.grid.arrlon,
                     self.grid.arrlat,
                     data,
                     metadata,
                     timestamp)

    def write(self, data):
        raise NotImplementedError()

    def flush(self):
        pass

    def close(self):
        pass
```

This new class `PickleImg` will read a pickled dictionary of data from the given filename. For the representation of the longitude and latitude of each datapoint the attributes of a `pygeogrids.grids.BasicGrid` object can be used but a regular numpy array would also do.

The next code snippet shows how this newly written class can be used in an implementation of *pygeobase.io_base.MultiTemporalImageBase*:

```
class PickleDs(MultiTemporalImageBase):

    def __init__(self, root_path):
        sub_path = ['%Y', '%m']
        fname_templ = "dataset_{datetime}.dat"
```

```
        datetime_format = "%Y-%m-%d"

        super(PickleDs, self).__init__(root_path, PickleImg,
                                       fname_templ=fname_templ,
                                       datetime_format=datetime_format,
                                       subpath_templ=sub_path)
```

The `sub_path` variable is a list of strings that build the path to the file from the python datetime object. The strftime syntax is used. `fname_templ` specifies the filename template in which `{datetime}` will be substituted by the string built by `datetime_format` according to the strftime syntax. There are more options to customize how the filename is build from a given python datetime. Please see the *pygeobase.io_base.MultiTemporalImageBase* documentation.

Please see the modindex for more details.

CHAPTER 2

Contents

License

```
Copyright (c) 2016, Vienna University of Technology, Department of Geodesy
and Geoinformation.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of pygeogrids nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Developers

- Sebastian Hahn <sebastian.hahn@geo.tuwien.ac.at>
- Christoph Paulik <christoph.paulik@geo.tuwien.ac.at>
- Thomas Mistelbauer <thomas.mistelbauer@geo.tuwien.ac.at>
- Christoph Reimer <christoph.reimer@geo.tuwien.ac.at>

pygeobase

pygeobase package

Submodules

pygeobase.io_base module

class pygeobase.io_base.**GriddedBase**(*path*, *grid*, *ioclass*, *mode='r'*, *fn_format='{:04d}'*, *ioclass_kws=None*)

Bases: object

The GriddedBase class uses another IO class together with a grid object to read/write a dataset under the given path.

Parameters

- **path** (*string*) – Path to dataset.
- **grid** (*pygeogrids.BasicGrid of CellGrid instance*) – Grid on which the time series data is stored.
- **ioclass** (*class*) – IO class.
- **mode** (*str, optional*) – File mode and can be read 'r', write 'w' or append 'a'. Default: 'r'
- **fn_format** (*str, optional*) – The string format of the cell files. Default: '{:04d}'
- **ioclass_kws** (*dict, optional*) – Additional keyword arguments for the ioclass. Default: None

close()
Close file.

flush()
Flush data.

get_spatial_subset(*gpis=None*, *cells=None*, *ll_bbox=None*, *grid=None*)
Select spatial subset and return data set with new grid.

Parameters

- **gpis** (*numpy.ndarray*) – Grid point indices.
- **cells** (*numpy.ndarray*) – Cell number.
- **ll_bbox** (*tuple (latmin, latmax, lonmin, lonmax)*) – Lat/Lon bounding box
- **grid** (*pygeogrids.CellGrid*) – Grid object.

Returns **dataset** – New data set with for spatial subset.

Return type *GriddedBase* or child

iter_gp(***kwargs*)
Yield all values for all grid points.

Yields

- **data** (*pandas.DataFrame*) – Data set.
- **gp** (*int*) – Grid point.

read(**args*, ***kwargs*)
Takes either 1 or 2 arguments and calls the correct function which is either reading the gpi directly or finding the nearest gpi from given lat,lon coordinates and then reading it

write(**args*, ***kwargs*)
Takes either 1 or 2 arguments and calls the correct function which is either writing the gpi directly or finding the nearest gpi from given lat,lon coordinates and then writing it.

class pygeobase.io_base.**GriddedStaticBase**(*path*, *grid*, *ioclass*, *mode='r'*, *fn_format='{:04d}'*, *ioclass_kws=None*)
Bases: *pygeobase.io_base.GriddedBase*

The GriddedStaticBase class uses another IO class together with a grid object to read/write a dataset under the given path.

class pygeobase.io_base.**GriddedTsBase**(*path*, *grid*, *ioclass*, *mode='r'*, *fn_format='{:04d}'*, *ioclass_kws=None*)
Bases: *pygeobase.io_base.GriddedBase*

The GriddedTsBase class uses another IO class together with a grid object to read/write a time series dataset under the given path.

iter_ts(***kwargs*)
Yield time series for all grid points. :Yields: * **data** (*object*) – pygeobase.object_base.TS object

•**gp** (*int*) – Grid point.

read_ts(**args*, ***kwargs*)
Takes either 1 or 2 arguments and calls the correct function which is either reading the gpi directly or finding the nearest gpi from given lat,lon coordinates and then reading it

write_ts(**args*, ***kwargs*)
Takes either 1, 2 or 3 arguments (the last one always needs to be the data to be written) and calls the correct function which is either writing the gp directly or finding the nearest gp from given lon, lat coordinates and then reading it.

class pygeobase.io_base.**ImageBase**(*filename*, *mode='r'*, ***kwargs*)
Bases: object

ImageBase class serves as a template for i/o objects used for reading and writing image data.

Parameters

- **filename** (*str*) – Filename path.
- **mode** (*str, optional*) – Opening mode. Default: r

close()
Close file.

flush()
Flush data.

read(***kwargs*)
Read data of an image file.

Returns **image** – pygeobase.object_base.Image object

Return type object

read_masked_data(***kwargs*)
Read data of an image file and mask the data according to specifications.

Returns **image** – pygeobase.object_base.Image object

Return type object

resample_data(*image*, *index*, *distance*, *weights*, ***kwargs*)
Takes an image and resample (interpolate) the image data to arbitrary defined locations given by index and distance.

The default implementation just takes the weighted mean of all defined distances.

Parameters

- **image** (*:py:class`pygeobase.object_base.Image` or numpy.recarray*) – Image or numpy.recarray like object with shape = (x,)
- **index** (*np.array*) – Index into image data defining a look-up table for data elements used in the interpolation process for each defined target location. For each point in image the neighbors in the targed grid are in the index array. This array is of shape (x, max_neighbors)
- **distance** (*np.array*) – Array representing the distances of the image data to the arbitrary defined locations. The distances of points not to use are set to np.inf This array is of shape (x, max_neighbors)
- **weights** (*np.array*) – Array representing the weights of the image data that should be used during resampling. The weights of points not to use are set to np.nan This array is of shape (x, max_neighbors)

Returns **target** – dictionary with a numpy.ndarray for each field in the input image. We can not return a image here since we do not know the target latitudes and longitudes.

Return type dict

write(*image*, ***kwargs*)
Write data to an image file.

Parameters **image** (*object*) – pygeobase.object_base.Image object

class pygeobase.io_base.**MultiTemporalImageBase**(*path*, *ioclass*, *mode='r'*, *fname_templ=''*, *datetime_format=''*, *subpath_templ=None*, *ioclass_kws=None*, *exact_templ=True*, *dtime_placeholder='datetime'*)

Bases: object

The MultiTemporalImageBase class make use of an ImageBase object to read/write a sequence of multi temporal images under a given path.

Parameters

- **path** (*string*) – Path to dataset.
- **ioclass** (*class*) – IO class.
- **mode** (*str, optional*) – File mode and can be read 'r', write 'w' or append 'a'. Default: 'r'

- **fname_templ** (*str*) – Filename template of the data to read. Default placeholder for parsing datetime information into the fname_templ is "{datetime}". e.g. "ASCAT_{datetime}_image.nc" will be translated into the filename ASCAT_20070101_image.nc for the date 2007-01-01.
- **datetime_format** (*str*) – String specifying the format of the datetime object to be parsed into the fname_template. e.g. "%Y/%m" will result in 2007/01 for datetime 2007-01-01 12:15:00
- **subpath_templ** (*list, optional*) – If given it is used to generate a sub-paths from the given timestamp. Each item in the list represents one folder level. This can be used if the files for May 2007 are e.g. in folders 2007/05/ then the files can be accessed via the list ['%Y', '%m'].
- **ioclass_kws** (*dict*) – Additional keyword arguments for the ioclass.
- **exact_templ** (*boolean, optional*) – If True then the fname_templ matches the filename exactly. If False then the fname_templ will be used in glob to find the file.
- **dtime_placeholder** (*str*) – String used in fname_templ as placeholder for datetime. Default value is "datetime".

close()
Close file.

daily_images(*day*, ***kwargs*)
Yield all images for a day.

Parameters **day** (*datetime.date*) –

Returns **img** – pygeobase.object_base.Image object

Return type object

flush()
Flush data.

get_tstamp_from_filename(*filename*)
Return the timestamp contained in a given file name in accordance to the defined fname_templ.

Parameters **filename** (*string*) – File name.

Returns **tstamp** – Time stamp according to fname_templ as datetime object.

Return type datetime.dateime

iter_images(*start_date*, *end_date*, ***kwargs*)
Yield all images for a given date range.

Parameters

- **start_date** (*datetime.date or datetime.datetime*) – start date
- **end_date** (*datetime.date or datetime.datetime*) – end date

Returns **image** – pygeobase.object_base.Image object

Return type object

read(*timestamp*, ***kwargs*)
Return an image for a specific timestamp.

Parameters **timestamp** (*datetime.datetime*) – Time stamp.

Returns **image** – pygeobase.object_base.Image object

Return type object

resample_image(**args*, ***kwargs*)

tstamps_for_daterange(*start_date*, *end_date*)
Return all valid timestamps in a given date range. This method must be implemented if iteration over images should be possible.

Parameters

- **start_date** (*datetime.date or datetime.datetime*) – start date
- **end_date** (*datetime.date or datetime.datetime*) – end date

Returns **dates** – list of datetimes

Return type list

write(*timestamp*, *data*, ***kwargs*)
Write image data for a given timestamp.

Parameters

- **timestamp** (*datetime.datetime*) – exact timestamp of the image
- **data** (*object*) – pygeobase.object_base.Image object

class pygeobase.io_base.**StaticBase**(*filename*, *mode='r'*, ***kwargs*)
Bases: object

The StaticBase class serves as a template for i/o objects used in GriddedStaticBase.

Parameters

- **filename** (*str*) – File name.
- **mode** (*str, optional*) – Opening mode. Default: r

close()
Close file.

flush()
Flush data.

read(*gpi*)
Read data for given grid point.

Parameters **gpi** (*int*) – Grid point index.

Returns **data** – Data set.

Return type numpy.ndarray

write(*data*)
Write data.

Parameters **data** (*numpy.ndarray*) – Data records.

class pygeobase.io_base.**TsBase**(*filename*, *mode='r'*, ***kwargs*)
Bases: object

The TsBase class serves as a template for i/o objects used in GriddedTsBase.

Parameters

- **filename** (*str*) – File name.
- **mode** (*str, optional*) – Opening mode. Default: r

close()
Close file.

flush()
Flush data.

read_ts(*gpi*, ***kwargs*)
Read time series data for given grid point.

Parameters **gpi** (*int*) – Grid point index.

Returns **data** – pygeobase.object_base.TS object.

Return type object

write_ts(*gpi*, *data*, ***kwargs*)
Write data.

Parameters

- **gpi** (*int*) – Grid point index.
- **data** (*object*) – pygeobase.object_base.TS object.

pygeobase.object_base module

class pygeobase.object_base.**Image**(*lon*, *lat*, *data*, *metadata*, *timestamp*, *timekey=None*)
Bases: object

The Image class represents the base object of an image.

Parameters

- **lon** (*numpy.array*) – array of longitudes
- **lat** (*numpy.array*) – array of latitudes
- **data** (*dict*) – dictionary of numpy arrays that holds the image data for each variable of the dataset
- **metadata** (*dict*) – dictionary that holds metadata
- **timestamp** (*datetime.datetime*) – exact timestamp of the image
- **timekey** (*str, optional*) – Key of the time variable, if available, stored in data dictionary.

dtype
Fake numpy recarray dtype field based on the dictionary keys and the dtype of the numpy array.

class pygeobase.object_base.**TS**(*gpi*, *lon*, *lat*, *data*, *metadata*)
Bases: object

The TS class represents the base object of a time series.

Parameters

- **lon** (*float*) – Longitude of the time series
- **lat** (*float*) – Latitude of the time series
- **data** (*pandas.DataFrame*) – Pandas DataFrame that holds data for each variable of the time series
- **metadata** (*dict*) – dictionary that holds metadata

plot(**args*, ***kwargs*)
wrapper for pandas.DataFrame.plot which adds title to plot and drops NaN values for plotting

Returns **ax** – matplotlib axes of the plot

Return type axes

Module contents

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index