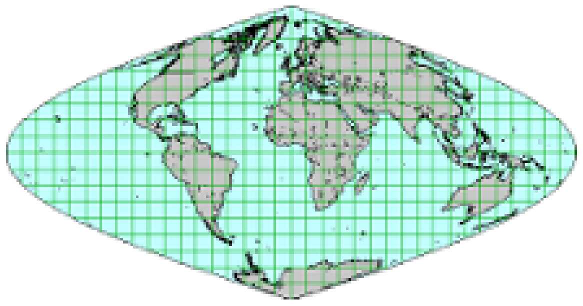


pyModis Documentation

Release 1.0.0

Luca Delucchi

January 16, 2015

CONTENTS

`pyModis` is a Free and Open Source Python based library to work with MODIS data. It offers bulk-download for user selected time ranges, mosaicking of MODIS tiles, and the reprojection from Sinusoidal to other projections, convert HDF format to other formats and the extraction of data quality information.

`pyModis` library was developed to replace old bash scripts developed by Markus Neteler to download MODIS data from NASA FTP server. It is very useful for GIS and Remote Sensing Platform of Fondazione Edmund Mach to update its large collection of MODIS data.

It has several features:

- it is very useful for downloading large numbers of MODIS HDF/XML files and for using this in a cron job for automated continuous updating; it support FTP and HTTP NASA repository
- it can parse the XML file to obtain information about the HDF files
- it can convert a HDF MODIS file to GEOTIF file using MODIS Reprojection Tool and also GDAL (from version 1.0)
- it can create a mosaic of several tiles using MODIS Reprojection Tool and also GDAL (from version 1.0)
- it can create the xml metadata file with the information of all tiles used in mosaic
- it can extract specific information from bit-encoded MODIS quality assesment layers of different product types
- Graphical User Interface for each script written in wxPython (from version 1.0)

We acknowledge the Fondazione Edmund Mach for promoting the development of free and open source software.

CHAPTER

ONE

ABOUT PYMODIS

1.1 Requirements

`pyModis` requires **Python GDAL** and **Numpy** packages.

If you want to use the Graphical User Interface you have to install also **wxPython** library.

You can use also software is MODIS Reprojection Tool to convert or mosaic MODIS HDF files.

1.2 How to install pyModis

1.2.1 Using pip

From version 0.6.3 it is possible to install `pyModis` using pip. You have to run the following command as administrator

```
pip install pyModis
```

If you need to update your `pyModis` version you have to run

```
pip install --upgrade pyModis
```

With `pip` it is also really simple to remove the library

```
pip uninstall pyModis
```

1.2.2 Compile from source

Compile `pyModis` is very simple. First you need to download `pyModis` source code from github repository.

You can use git to download the latest code (with the whole history and so it contain all the different stable versions, from the last to the first)

```
git clone https://github.com/lucadelu/pyModis.git
```

or download the latest stable version from the repository and decompress it.

Now enter the `pyModis` folder and launch as administrator of your computer

```
python setup.py install
```

If the installation doesn't return any errors you should be able to use `pyModis` library from a Python console. Then, launch a your favorite Python console (I really suggest `ipython`) and digit

```
import pymodis
```

If the console doesn't return any error like this

```
ImportError: No module named pymodis
```

the `pyModis` library has been installed properly and you can use it or one of the tools distributed with `pyModis`.

1.2.3 Install on Windows

The simple way to install `pyModis` on Windows is to install latest Python 2.7 from http://python.org/download/

Now you have to modify the "Path" environment variable using *powershell* running

```
[Environment]::SetEnvironmentVariable("Path",
"$env:Path;C:\Python27\;C:\Python27\Scripts\", "User")
```

Download and install the last version of Distribute for Windows from http://python-distribute.org/distribute_setup.py

At this point you have to move to standard command line (*cmd*) and install *pip* using *easy_install*

```
easy_install pip
```

Now install numpy library using *easy_install* because installation from pip is broken (this is required only for version >= 0.7.1)

```
easy_install numpy GDAL
```

If you want the Graphical User Interface you have to install also **wxPython**

```
easy_install WxPython WxPython-Common
```

Finally install `pyModis` using *pip*

```
pip install pyModis
```

If you want use the GUI you have to download and install wxPython

1.3 Troubleshooting

1.3.1 Problem installing dependencies with pip

Sometimes *pip* return error when it try to install Python GDAL, Numpy or wxPython. You can solve this problem installing Python GDAL or Numpy using the version of your operating system.

1.4 How to report a bug

If you find any problems in `pyModis` library you can report it using the issues tracker of github.

1.5 How to compile documentation

This documentation has been made with Sphinx, so you need to install it to compile the original files to obtain different output formats.

Please enter the `docs` folder of `pyModis` source and run

```
make <target>
```

with one of the following target to obtain the desired output:

- **html**: to make standalone HTML files
- **dirhtml**: to make HTML files named index.html in directories
- **singlehtml**: to make a single large HTML file
- **pickle**: to make pickle files
- **json**: to make JSON files
- **htmlhelp**: to make HTML files and a HTML help project
- **qthelp**: to make HTML files and a qthelp project
- **devhelp**: to make HTML files and a Devhelp project
- **epub**: to make an epub
- **latex**: to make LaTeX files, you can set PAPER=a4 or PAPER=letter
- **latexpdf**: to make LaTeX files and run them through pdflatex
- **text**: to make text files
- **man**: to make manual pages
- **texinfo**: to make Texinfo files
- **info**: to make Texinfo files and run them through makeinfo
- **gettext**: to make PO message catalogs
- **changes**: to make an overview of all changed/added/deprecated items
- **linkcheck**: to check all external links for integrity
- **doctest**: to run all doctests embedded in the documentation (if enabled)

1.5.1 PDF link in HTML

To insert a link to PDF file of pyModis documentation into HTML documentation (the link will be added on the sidebar) you have to compile first the PDF and after the HTML, so you need to launch:

```
make latexpdf
make html
```

If PDF file is missing no link will be added

1.6 Ohloh statistics

For more information about `pyModis` please visit the pyModis Ohloh page

CHAPTER

TWO

PYMODIS SCRIPTS

The `pyModis` **scripts** provide you with a complete toolkit to work with MODIS data, you can download, analyze and convert the data. They are developed to work from command line and inside scripts to automatically update your MODIS files dataset. From version 1.0 the scripts have also Graphical User Interface.

Currently the tools are:

- *modis_download.py*
- *modis_download_from_list.py*
- *modis_parse.py*
- *modis_multiparse.py*
- *modis_mosaic.py*
- *modis_convert.py*
- *modis_quality.py*

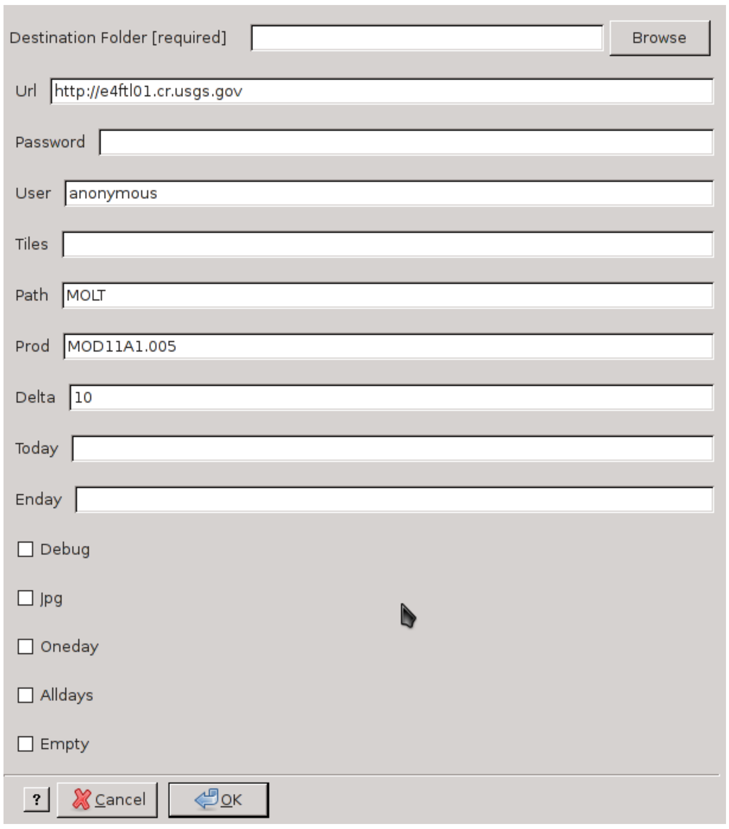
2.1 modis_download.py

modis_download.py downloads MODIS data from NASA FTP servers. It can download large amounts of data and it can be profitably used with cron jobs to receive data with a fixed delay of time.

2.1.1 Usage

```
modis_download.py [options] destination_folder
```

2.1.2 Options



Destination Folder [required]	
Url	http://e4ftl01.cr.usgs.gov
Password	
User	anonymous
Tiles	
Path	MOLT
Prod	MOD11A1.005
Delta	10
Today	
Enday	
☐ Debug	
☐ Jpg	
☐ Oneday	
☐ Alldays	
☐ Empty	

```
-h   --help          show the help message and exit
-u   --url           http/ftp server url [default=http://e4ftl01.cr.usgs.gov]
-P   --password      password to connect only if ftp server
-U   --username      username to connect only if ftp server
                     [default=anonymous]
-t   --tiles         string of tiles separated from comma
                     [default=none] for all tiles
-s   --source        directory on the http/ftp
                     [default=MOLT]
-p   --product       product name as on the http/ftp server
                     [default=MOD11A1.005]
-D   --delta         delta of day from the first day [default=10]
-f   --firstday      the day to start download, if you want change
                     data you have to use this format YYYY-MM-DD
                     ([default=none] is for today)
-e   --enddaythe     day to finish download, if you want change
```

```
                    data you have to use this format YYYY-MM-DD
                    ([default=none] use delta option)
-x                  this is useful for debugging the download
                    [default=False]
-j                  download also the jpeg files [default=False]
-O                  download only one day, it set delta=1 [default=False]
-A                  download all days, it useful for first download of a
                    product. It overwrite the 'firstday' and 'endday'
                    options [default=False]
-r                  remove files with size same to zero from
                    'destination_folder'  [default=False]
```

2.1.3 Examples

Download Terra LST data for a month for two tiles from HTTP server

```
modis_download.py -r -t h18v03,h18v04 -f 2008-01-01 -e 2008-01-31 lst_terra/
```

Download the last 15 days of Aqua LST data

```
modis_download.py -r -s MOLA -p MYD11A1.005 -t h18v03,h18v04 -D 15 lst_aqua/
```

Download all tiles of NDVI for one day (you have know the right day otherwise it download nothing)

```
modis_download.py -r -p MOD13Q1.005 -f 2010-12-31 -O
```

Download Snow product from FTP server

```
modis_download.py -u ftp://n4ftl01u.ecs.nasa.gov -p mail@pymodis.com
-s SAN/MOST -p MOD10A1.005
```

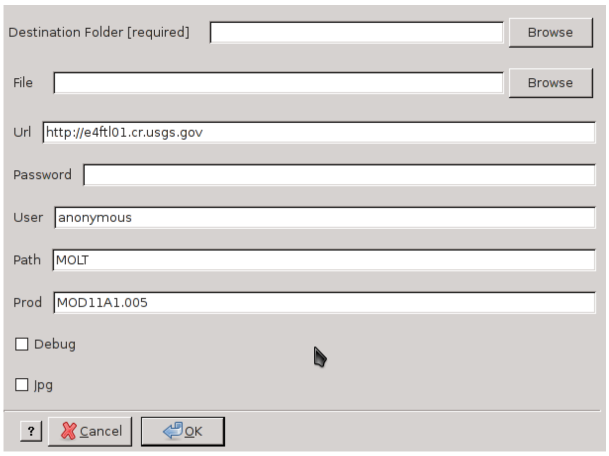
2.2 modis_download_from_list.py

modis_download_from_list.py downloads MODIS data from NASA servers, the names of files to download have to be contained into a text file.

2.2.1 Usage

```
modis_download_from_list.py [options] destination_folder
```

2.2.2 Options



Destination Folder [required]
File
Url http://e4ftl01.cr.usgs.gov
Password
User anonymous
Path MOLT
Prod MOD11A1.005
☐ Debug
☐ Jpg

```
-h  --help        show the help message and exit
-f  --file        Input file containing data to donwload
-u  --url         http/ftp server url [default=http://e4ftl01.cr.usgs.gov]
-P  --password    password to connect only if ftp server
-U  --username    username to connect only if ftp server
                  [default=anonymous]
-t  --tiles       string of tiles separated from comma
                  [default=none] for all tiles
-s  --source      directory on the http/ftp
                  [default=MOLT]
-p  --product     product name as on the http/ftp server
                  [default=MOD11A1.005]
-x                this is useful for debugging the download
                  [default=False]
-j                download also the jpeg files [default=False]
```

2.2.3 Examples

The following text should be in your *MODTiles.txt* file

```
MOD11A1.A2012278.h19v11.005.*.hdf*
MOD11A1.A2012278.h19v12.005.*.hdf*
MOD11A1.A2012278.h20v11.005.*.hdf*
MOD11A1.A2012278.h20v12.005.*.hdf*
MOD11A1.A2012278.h21v11.005.*.hdf*
```

Download Terra LST data from the above text file

```
modis_download_from_list.py -f /tmp/MODTiles.txt /tmp
```

The following text should be in your *MYDTiles.txt* file

```
MYD11A1.A2012278.h19v11.005.*.hdf*
MYD11A1.A2012278.h19v12.005.*.hdf*
MYD11A1.A2012278.h20v11.005.*.hdf*
MYD11A1.A2012278.h20v12.005.*.hdf*
MYD11A1.A2012278.h21v11.005.*.hdf*
```

Download Aqua LST data from the above text file

```
modis_download_from_list.py -s MOLA -p MYD11A1.005 -f /tmp/MYDTiles.txt /tmp
```

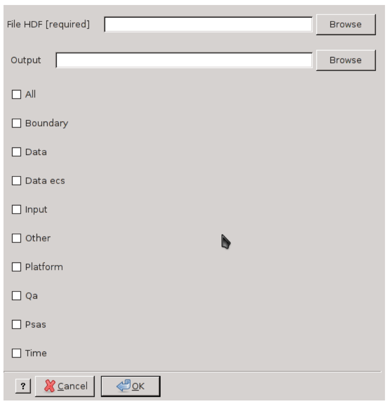
2.3 modis_parse.py

modis_parse.pys parses the XML metadata file for a MODIS tile and return the requested value. It can also write the metadata information in a text file.

2.3.1 Usage

```
modis_parse.py [options] hdf_file
```

2.3.2 Options



```
-h  --help     show the help
-a             print all possible values of metadata
-b             print the values related to the spatial max extent
-d             print the values related to the date files
-e             print the values related to the ECSDataGranule
-i             print the input layers
-o             print the other values
-p             print the values related to platform
-q             print the values related to quality
-s             print the values related to psas
-t             print the values related to times
-w  --write    write the chosen information into a file
```

2.3.3 Examples

Return all values of metadata

```
modis_parse.py -a FILE
```

Write all values to a file

```
modis_parse.py -a -w metadata_FILE.txt FILE
```

Print spatial extent and quality

```
modis_parse.py -b -q FILE
```

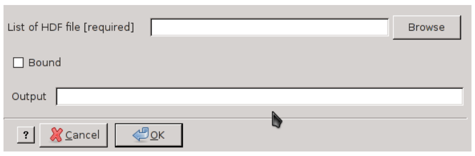
2.4 modis_multiparse.py

modis_multiparse.py parses several XML metadata files for MODIS tiles. It is very useful to create XML metadata file for a mosaic.

2.4.1 Usage

```
modis_multiparse.py [options] hdf_files_list
```

2.4.2 Options


List of HDF file [required]

Bound

Output

```
-h  --help     show the help
-b             print the values related to the spatial max extent
-w  --write    write the MODIS XML metadata file for MODIS mosaic
```

2.4.3 Examples

Print values of spatial bounding box

```
modis_multiparse.py -b FILE1 FILE2 ...
```

Write xml file to use with hdf file create by *modis_convert.py*

```
modis_multiparse.py -w FILE_mosaic.xml FILE1 FILE2 ...
```

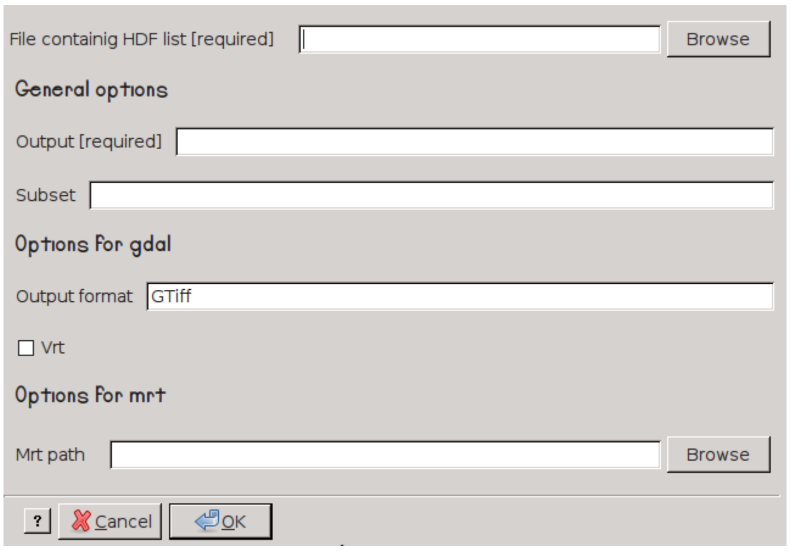
2.5 modis_mosaic.py

modis_mosaic.py creates a mosaic of several MODIS tiles in HDF format, using MRT mrtmosaic software or GDAL library.

2.5.1 Usage

```
modis_mosaic.py [options] hdflist_file
```

2.5.2 Options



File containig HDF list [required]

General options

Output [required]

Subset

Options for gdal

Output format GTiff

☐ Vrt

Options for mrt

Mrt path

```
General options:
  -o OUTPUT_FILE, --output=OUTPUT_FILE
                        (Required) the name or prefix (for VRT) of output
                        mosaic
  -s SUBSET, --subset=SUBSET
                        a subset of product layers. The string should be
                        similar to: 1 0 [default: all layers]

Options for GDAL:
  -f OUTPUT_FORMAT, --output-format=OUTPUT_FORMAT
                        output format supported: GTiff, HDF4Image
                        [default=GTiff]
  -v, --vrt             Create a GDAL VRT file. No other GDAL options have to
                        been set

Options for MRT:
  -m MRT_PATH, --mrt=MRT_PATH
                        (Required) the path to MRT software
```

2.5.3 Examples

MODIS Reprojection Tools

Convert all the layers of several tiles:

```
modis_mosaic.py -m "/usr/local/bin/" -o FILE_mosaik MOSAIK_FILES_LIST
```

Convert LAYERS of several LST MODIS tiles:

```
modis_mosaic.py -s "1 0 1 0" -m "/usr/local/bin/" -o FILE_mosaik MOSAIK_FILES_LIST
```

GDAL

Convert the first LAYERS of several tiles with resolution 1km in GeoTIFF format:

```
modis_mosaic.py -o FILE_mosaik.tif -s "1"  MOSAIK_FILES_LIST
```

Create a mosaic with all the layers of several tiles in HDF4Image format:

```
modis_mosaic.py -o FILE_mosaik.hdf -f HDF4Image MOSAIK_FILES_LIS
```

Create VRT file for all subset. It create a VRT file for each subset with the choosen prefix (`-o` flag) and the name of layer as suffix:

```
modis_mosaic.py -o mosaik_vrt -v MOSAIK_FILES_LIS
```

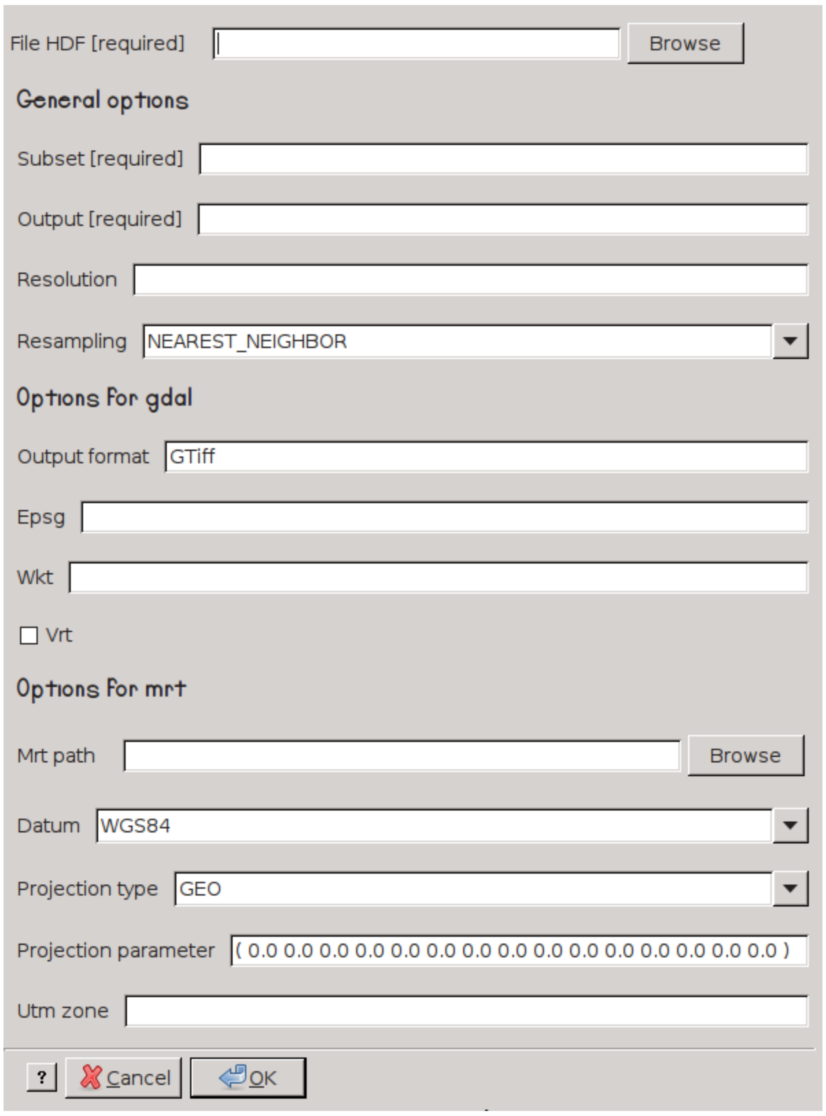
2.6 modis_convert.py

modis_convert.py converts MODIS data to TIF formats and different projection reference system. It is an interface to MRT mrtmosaic software or GDAL library.

2.6.1 Usage

```
modis_convert.py [options] hdf_file
```

2.6.2 Options



File HDF [required]

General options

Subset [required]

Output [required]

Resolution

Resampling: NEAREST_NEIGHBOR

Options for gdal

Output format: GTiff

Epsg

Wkt

☐ Vrt

Options for mrt

Mrt path

Datum: WGS84

Projection type: GEO

Projection parameter: (0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)

Utm zone

```
-h, --help            show this help message and exit

Required options:
  -s SUBSET, --subset=SUBSET
                      (Required) a subset of product's layers. The string
                      should be similar to: ( 1 0 )
  -o OUTPUT_FILE, --output=OUTPUT_FILE
```

```
                      (Required) the prefix of output file
  -g RESOLUTION, --grain=RESOLUTION
                      the spatial resolution of output file
  -r RESAMPLING_TYPE, --resampl=RESAMPLING_TYPE
                      the method of resampling. -- mrt methods:
                      'NEAREST_NEIGHBOR', 'BICUBIC', 'CUBIC_CONVOLUTION',
                      'NONE' -- gdal methods: 'AVERAGE', 'BILINEAR',
                      'CUBIC', 'CUBIC_SPLINE', 'LANCZOS', 'MODE',
                      'NEAREST_NEIGHBOR' [default=NEAREST_NEIGHBOR]

Options for GDAL:
  -f OUTPUT_FORMAT, --output-format=OUTPUT_FORMAT
                      output format supported by GDAL [default=GTiff]
  -e EPSG, --epsg=EPSG
                      EPSG code for the output
  -w WKT, --wkt_file=WKT
                      file or string containing projection definition in WKT
                      format
  -v, --vrt           Read from a GDAL VRT file.
  --formats           print supported GDAL formats

Options for MRT:
  -m MRT_PATH, --mrt=MRT_PATH
                      the path to MRT software
  -d DATUM, --datum=DATUM
                      the code of datum. Available: 'NODATUM', 'NAD27',
                      'NAD83', 'WGS66', 'WGS72', 'WGS84' [default=WGS84]
  -t PROJECTION_SYSTEM, --proj_type=PROJECTION_SYSTEM
                      the output projection system. Available: 'AEA', 'GEO',
                      'HAM', 'IGH', 'ISIN', 'LA', 'LCC', 'MOL', 'PS', 'SIN',
                      'TM', 'UTM', 'MERCAT' [default=GEO]
  -p PROJECTION_PARAMETERS, --proj_parameters=PROJECTION_PARAMETERS
                      a list of projection parameters, for more info check
                      the 'Appendix C' of MODIS reprojection tool user's
                      manual https://lpdaac.usgs.gov/content/download/4831/2
                      2895/file/mrt41_usermanual_032811.pdf [default=( 0.0
                      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
                      0.0 )]
  -u UTM_ZONE, --utm=UTM_ZONE
                      the UTM zone if projection system is UTM
            the UTM zone if projection system is UTM
```

2.6.3 Examples

Warning: The resolution value in `modis_convert.py` has to be set with the right value depending on the projection used. 1 kilometer in metrical projection has to be set as 1000 meter, instead in latitude and longitude something like 0.01* depending on the placement in the Earth.

MODIS Reprojection Tools

Warning: You can find the supported projections in the 'Appendix C' of MODIS reprojection tool user's manual and the datums at section `Datum Conversion` of the same manual

Convert layers from MODIS data with the original resolution in latitude and longitude reference system

```
modis_convert.py -s "( 1 0 1 0 )" -o OUTPUT_FILE -m "/usr/local/bin/" FILE
```

Convert layers from MODIS data with output resolution in 500 meters with UTM projection in the 32 zone

```
modis_convert.py -s "( 1 0 1 0 )" -o OUTPUT_FILE -m "/usr/local/bin/" -g 500 -p UTM -u 3
```

GDAL

Convert the first layer in latitude and longitude with the original resolution

```
modis_convert.py -s "( 1 )" -o OUTPUT_FILE -e 4326 FILE
```

Convert the first three layers from MODIS data with output resolution in 500 meters with UTM projection in the 32 zone

```
modis_convert.py -s "( 1 1 1 )" -o OUTPUT_FILE -g 500 -e 32632 FILE
```

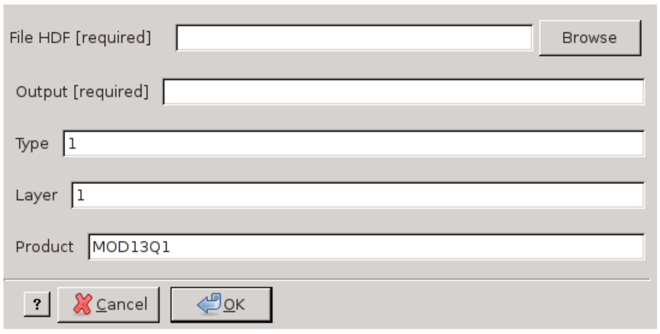
2.7 modis_quality.py

modis_quality.py checks the quality of MODIS data using the QA layer

2.7.1 Usage

```
modis_quality.py [options] input_file destination_file
```

2.7.2 Options


File HDF [required]

Output [required]

Type: 1

Layer: 1

Product: MOD13Q1

```
-h, --help            show this help message and exit
-o OUTPUT_FILE, --output=OUTPUT_FILE
                      (Required) the prefix of output file
-t TYPE, --type=TYPE  quality type either as number or name (e.g. 1 or
                      VIQuality for MOD13 products) [default=1]
-l LAYER, --qualitylayer=LAYER
                      quality layer of the dataset, dependent on the used
                      MODIS product. (e.g. 1 or QC_Day for the Daytime QC
                      Layer of MOD11) [default=1]
-p PRODUCT, --producttype=PRODUCT
                      quality layer of the dataset, dependent on the used
                      MODIS product. (e.g. 1 or QC_Day for the Daytime QC
                      Layer of MOD11) [default=MOD13Q1]
```

2.7.3 Examples

Extract VI Usefulness value from MOD13 product

```
modis_quality.py -t 2 infile.hdf outfile.tif
```

Extract shadow mask from MOD13 product

```
modis_quality.py -t 9 input_file.hdf destination_file.tif
```

Extract Emissitivity error flag of Nighttime LSTE quality control from MOD11C1 product

```
modis_quality.py -t 4 -l 2 infile.hdf outfile.tif
```

Extract MODLAND QA value from MOD13Q1 mosaic

```
modis_quality.py -t 1 -p MOD13Q1 input_file.hdf destination_file.tif
```

CHAPTER THREE

EXAMPLES

Some example about `pyModis` library and script

3.1 Scripts

- *Example of a full process with GDAL library*
- *Example reproject data with MRT*

3.1.1 Example of a full process with GDAL library

In this short example you can understand how to concatenate the scripts to obtain a GeoTIFF file for each band of the chosen product using as backend GDAL library.

Warning: This example is based on a Linux based system. Please if you use other OS change the paths where data will be saved

Downloading data

For first you need to obtain data, so you need to use *modis_download.py*

```
modis_download.py -f 2012-12-05 -O -t h28v05,h29v05,h28v04 /tmp
```

Warning: In this example we are working on Italian extension, so please change the name of tiles according with your region.
In this example we download data for only one day (2012-12-05) using the option "-O".

Inside `/tmp/` directory you will find a file called *listfileMOD11A1.005.txt* containing the names of files downloaded. The name of file it is related to the product that you download.

Warning: Every time that you download new files of same product it will be overwrite, so if you need it, you should rename the file

Mosaic data

At this point you need to create the mosaic of the tiles downloaded. *modis_mosaic.py* is the script to use. We create a *VRT* file (`flag -v`) to improve the speed of analysis, without lose any data only for the first layer

```
modis_mosaic.py -s "1" -o /tmp/mosaik -v /tmp/listfileMOD11A1.005.txt
```

The command will create a file called `mosaik_LST_Day_1km.vrt` in /tmp/ directory

Convert data

The last part of the procedure is to convert the mosaic using *modis_convert.py*. Using *VRT* format it create dataset of only one later, so you are forced to use `-s "( 1 )"`. The following command create a GeoTIFF file called final_mosaik_LST_Day_1km.vrt.tif

```
modis_convert.py -v -s "( 1 )" -o /tmp/final -e 4326 /tmp/mosaik_LST_Day_1km.vrt
```

3.1.2 Example reproject data with MRT

In this short example you can understand how to concatenate the scripts to obtain a GeoTIFF file for each band of the chosen product using as backend MODIS Reprojection Tools (MRT).

Warning: This example is based on a Linux based system. Please if you use other OS change the paths where data will be saved

Downloading data

For first you need to obtain data, so you need to use *modis_download.py*

```
modis_download.py -f 2012-12-05 -O -t h28v05,h29v05,h28v04 /tmp
```

Warning: In this example we are working on Japan extension, so please change the name of tiles according with your region.
In this example we download data for only one day (2012-12-05) using the option "-O".

Inside `/tmp/` directory you will find a file called *listfileMOD11A1.005.txt* containing the names of files downloaded. The name of file it is related to the product that you download.

Warning: Every time that you download new files of same product it will be overwrite, so if you need it, you should rename the file

Mosaic data

modis_mosaic.py is the script to use.

```
modis_mosaic.py -m /path/to/mrt/ -o /tmp/outputfile /tmp/listfileMOD11A1.005.txt
```

Warning: `/path/to/mrt/` is the directory where Modis Reprojection Tools is stored

The output of this command are *outputfile.hdf* and *outputfile.hdf.xml* inside the directory `/tmp`. It's reading the input files contained in *listfileMOD11A1.005.txt*

Convert data

The last part of the procedure is to convert the mosaic, from HDF format and sinusoidal projection, to GeoTIFF with several projection. You have to use *modis_convert.py*

```
modis_convert.py -s '( 1 1 1 1 1 1 1 1 1 1 1 1 )' -m /path/to/mrt/
                 -o /tmp/finalfile.tif -g 250 /tmp/outputfile.hdf
```

Extract quality information

If necessary, you can extract specific quality type from the chosen quality layer. In this particular case, we extract the Mandatory QA flag of the daytime temperature. You have to use *modis_quality.py*

```
modis_quality.py -p MOD11A1 -l 1 -t 1 /tmp/outputfile.hdf
/tmp/mod11a1_daytime_qaflag.tif
```

3.2 Library

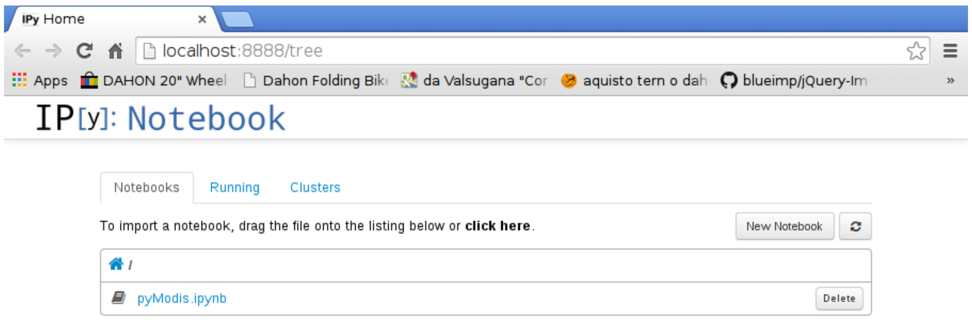
To test `pyModis` library you can find an Ipython notebook example in the documentation source code. If you already downloaded `pyModis` source code you have just to move inside the directory `pyModis/docs/source/examples` otherwise you can download the needed file from source code and move to the directory where you downloaded the file.

At this point you can start Ipython notebook running a notebook server from the command line using the following command

```
ipython notebook
```

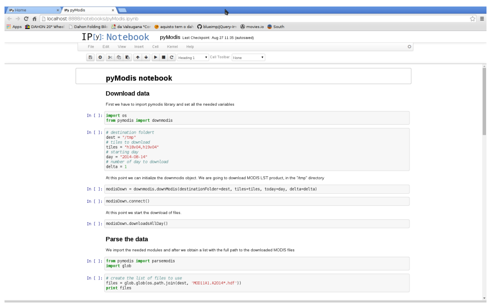
This will print some information about the notebook server in your console, and open a web browser to the URL of the web application.

The landing page of the IPython notebook web application, the dashboard, shows the notebooks currently available in the notebook directory (in our case only **pyModis.ipynb**).



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

Clicking on **pyModis.ipynb** link you will start the notebook


IPy pyModis
localhost:8888/notebooks/pyModis.ipynb
IP[y]: Notebook
pyModis Last Checkpoint: Aug 27 11:35 (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Heading 1
Cell Toolbar:
None

pyModis notebook

Download data

First we have to import pymodis library and set all the needed variables

```
In [ ]: import os
        from pymodis import downmodis
```

```
In [ ]: # destination foldert
        dest = "/tmp"
        # tiles to download
        tiles = "h18v04,h19v04"
        # starting day
        day = "2014-08-14"
        # number of day to download
        delta = 1
```

At this point we can initialize the downmodis object. We are going to download MODIS LST product, in the "/tmp" directory

```
In [ ]: modisDown = downmodis.downModis(destinationFolder=dest, tiles=tiles, today=day, delta=delta)
```

```
In [ ]: modisDown.connect()
```

At this point we start the download of files.

```
In [ ]: modisDown.downloadsAllDay()
```

Parse the data

We import the needed modules and after we obtain a list with the full path to the downloaded MODIS files

```
In [ ]: from pymodis import parsemodis
        import glob
```

```
In [ ]: # create the list of files to use
        files = glob.glob(os.path.join(dest, 'MOD11A1.A2014*.hdf'))
        print files
```

Warning: You have to install `pyModis` before run the Ipython notebook example.

CHAPTER

FOUR

PYMODIS LIBRARY

`pyModis` library it is a Python library to work with MODIS data.

It can easily used in your application to download, analyze, convert and check the quality of MODIS data.

`pyModis` can be used in other free and open source software, it is already present in GRASS GIS used by r.in.modis addon.

It is compose by this modules:

- *downmodis module*
- *parsemodis module*
- *convertmodis module*
- *convertmodis_gdal module*
- *qualitymodis module*
- *optparse_required module*

4.1 `downmodis` module

Module to download MODIS HDF files from NASA repository. It supports both FTP and HTTP repositories

Classes:

- `modisHtmlParser`
- `downModis`

Functions:

- `urljoin()`
- `getNewerVersion()`
- `str2date()`

class `pymodis.downmodis.`**`downModis`**(*destinationFolder*, *password=None*, *user='anonymous'*, *url='http://e4ftl01.cr.usgs.gov'*, *tiles=None*, *path='MOLT'*, *product='MOD11A1.005'*, *today=None*, *enddate=None*, *delta=10*, *jpg=False*, *debug=False*, *timeout=30*)

A class to download MODIS data from NASA FTP or HTTP repositories

Parameters

- **destinationFolder** (*str*) – where the files will be stored
- **password** (*str*) – the password, it should be your email address to connect to a FTP server. Do not use this variable if the server is an HTTP server
- **user** (*str*) – the user name, by default 'anonymous', used to connect to an FTP server. Do not use this variable if the server is an HTTP server
- **url** (*str*) – the base url from where to download the MODIS data, it can be FTP or HTTP but it has to start with 'ftp://' or 'http://'
- **path** (*str*) – the directory where the data that you want to download are stored on the FTP server. For HTTP requests, this is the part of the url between the 'url' parameter and the 'product' parameter.
- **product** (*str*) – the code of the product to download, the code should be idential to the one of the url
- **tiles** (*str*) – a set of tiles to be downloaded, None == all tiles. This can be passed as a string of tileIDs separated by commas, or as a list of individual tileIDs
- **today** (*str*) – the day to start downloading; in order to pass a date different from today use the format YYYY-MM-DD
- **enddate** (*str*) – the day to end downloading; in order to pass a date use the format YYYY-MM-DD. This day must be before the 'today' parameter. Downloading happens in reverse order (currently)
- **delta** (*int*) – timelag i.e. the number of days starting from today backwards. Will be overwritten if 'enddate' is specifed during instantiation

- **jpeg** (*bool*) – set to True if you want to download the JPG overview file in addition to the HDF
- **debug** (*bool*) – set to True if you want to obtain debug information
- **timeout** (*int*) – Timeout value for HTTP server (seconds)

`checkDataExist` (*listNewFile*, *move=False*)
Check if a file already exists in the local download directory

Parameters

- **listNewFile** (*list*) – list of all files, returned by getFilesList function
- **move** (*bool*) – it is useful to know if a function is called from download or move function

Returns list of files to download

`checkFile` (*filHdf*)
Check by using GDAL to be sure that the download went ok

Parameters **filHdf** (*str*) – name of the HDF file to check

Returns 0 if file is correct, 1 for error

`closeFTP` ()
Close ftp connection and close the file list document

`closeFilelist` ()
Function to close the file list of where the files are downloaded

`connect` (*ncon=20*)
Connect to the server and fill the dirData variable

Parameters **ncon** (*int*) – maximum number of attempts to connect to the HTTP server before failing

`dayDownload` (*day*, *listFilesDown*)
Downloads tiles for the selected day

Parameters

- **day** (*str*) – the day in format YYYY.MM.DD
- **listFilesDown** (*list*) – list of the files to download, returned by checkDataExist function

`debugDays` ()
This function is useful to debug the number of days

`debugLog` ()
Function to create the debug file

Returns a Logger object to use to write debug info

`debugMaps` ()
Prints the files to download to the debug stream

`downloadFile` (*filDown*, *filHdf*, *day*)
Download a single file

Parameters

- **filDown** (*str*) – name of the file to download
- **filHdf** (*str*) – name of the file to write to
- **day** (*str*) – the day in format YYYY.MM.DD

`downloadsAllDay`(*clean=False*, *allDays=False*)
Download all requested days

Parameters

- **clean** (*bool*) – if True remove the empty files, they could have some problems in the previous download
- **allDays** (*bool*) – download all passable days

`getAllDays`()
Return a list of all days

`getFilesList`(*day=None*)
Returns a list of files to download. HDF and XML files are downloaded by default. JPG files will be downloaded if self.jpeg == True.

Parameters **day** (*str*) – the date of data in format YYYY.MM.DD

Returns a list of files to download for the day

`getListDays`()
Return a list of all selected days

`removeEmptyFiles`()
Remove files in the download directory that have filesize equal to 0

`setDirectoryIn`(*day*)
Enter into the file directory of a specified day

Parameters **day** (*str*) – a string representing a day in format YYYY.MM.DD

`setDirectoryOver`()
Move up within the file directory

`pymodis.downmodis.`**`getNewerVersion`**(*oldFile*, *newFile*)
Check two files to determine which is newer

Parameters

- **oldFile** (*str*) – one of the two similar files
- **newFile** (*str*) – one of the two similar files

Returns the name of newer file

class `pymodis.downmodis.`**`modisHtmlParser`**(*fh*)
Bases: `HTMLParser.HTMLParser`

A class to parse HTML

Parameters **fh** – content of http request

`get_all`()
Return everything

`get_dates`()
Return a list of directories with date

get_tiles(*prod*, *tiles*, *jpeg=False*)
Return a list of files to download

Parameters

- **prod** (*str*) – the code of MODIS product that we are going to analyze
- **tiles** (*list*) – the list of tiles to consider
- **jpeg** (*bool*) – True to also check for jpeg data

handle_starttag(*tag*, *attrs*)

pymodis.downmodis.**str2date**(*datestring*)
Convert to datetime.date object from a string

:param str datestring string with format (YYYY-MM-DD)

Returns a datetime.date object representing datestring

pymodis.downmodis.**urljoin**(**args*)
Joins given arguments into a url. Trailing but not leading slashes are stripped for each argument. http://stackoverflow.com/a/11326230

Returns a string

4.2 `parsemodis` module

Simple class to parse MODIS metadata file, it can also write the XML metadata file for a mosaic.

Classes:

- `parseModis`
- `parseModisMulti`

class `pymodis.parsemodis.`**`parseModis`**(*filename*)

Class to parse MODIS xml files, it can also create the parameter configuration file for resampling MODIS DATA with the MRT software or convertmodis Module

Parameters **filename** (*str*) – the name of MODIS hdf file

`confResample`(*spectral*, *res=None*, *output=None*, *datum='WGS84'*, *resample='NEAREST_NEIGHBOR'*, *projtype='GEO'*, *utm=None*, *projpar='(0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0)'*, *bound=None*)

Create the parameter file to use with resample MRT software to create tif (geotiff) file

Parameters

- **spectral** (*str*) – the spectral subset to be used, see the product table to understand the layer that you want use. For example:
  - NDVI (1 1 1 0 0 0 0 0 0 0 0 0) copy only layer NDVI, EVI and QA VI the other layers are not used
  - LST (1 1 0 0 1 1 0 0 0 0 0 0) copy only layer daily and nightly temperature and QA
- **res** (*int*) – the resolution for the output file, it must be set in the map unit of output projection system. The software will use the original resolution of input file if res not set
- **output** (*str*) – the output name, if not set if not set the prefix name of input hdf file will be used
- **utm** – the UTM zone if projection system is UTM
- **resample** (*str*) – the type of resampling, the valid values are:
  - NN (nearest neighbor)
  - BI (bilinear)
  - CC (cubic convolution)
- **projtype** (*str*) – the output projection system, valid values are:
  - AEA (Albers Equal Area)
  - ER (Equirectangular)
  - GEO (Geographic Latitude/Longitude)
  - HAM (Hammer)
  - ISIN (Integerized Sinusoidal)
  - IGH (Interrupted Goode Homolosine)

  - LA (Lambert Azimuthal)
  - LCC (LambertConformal Conic)
  - MERCAT (Mercator)
  - MOL (Mollweide)
  - PS (Polar Stereographic)
  - SIN (Sinusoidal)
  - UTM (Universal TransverseMercator)
- **datum** (*str*) – the datum to use, the valid values are:
  - NAD27
  - NAD83
  - WGS66
  - WGS76
  - WGS84
  - NODATUM
- **projpar** (*str*) – a list of projection parameters, for more info check the Appendix C of MODIS reprojection tool user manual https://lpdaac.usgs.gov/content/download/4831/22895/file/mrt41_usermanual_032811.pdf
- **bound** (*dict*) – dictionary with the following keys:
  - max_lat
  - max_lon
  - min_lat
  - min_lon

`confResample_swath`(*sds*, *geoloc*, *res*, *output=None*, *sphere='8'*, *resample='NN'*, *projtype='GEO'*, *utm=None*, *projpar='0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0'*, *bound=None*)

Create the parameter file to use with resample MRT software to create tif (geotiff) file

Parameters

- **sds** (*str*) – Name of band/s (Science Data Set) to resample
- **geoloc** (*str*) – Name geolocation file (example MOD3, MYD3)
- **res** (*int*) – the resolution for the output file, it must be set in the map unit of output projection system. The software will use the original resolution of input file if res not set
- **output** (*str*) – the output name, if not set the prefix name of input hdf file will be used
- **sphere** (*int*) – Output sphere number. Valid options are:
  - 0=Clarke 1866

  - 1=Clarke 1880
  - 2=Bessel
  - 3=International 1967
  - 4=International 1909
  - 5=WGS 72
  - 6=Everest
  - 7=WGS 66
  - 8=GRS1980/WGS 84
  - 9=Airy
  - 10=Modified Everest
  - 11=Modified Airy
  - 12=Walbeck
  - 13=Southeast Asia
  - 14=Australian National
  - 15=Krassovsky
  - 16=Hough
  - 17=Mercury1960
  - 18=Modified Mercury1968
  - 19=Sphere 19 (Radius 6370997)
  - 20=MODIS Sphere (Radius 6371007.181)
- **resample** (*str*) – the type of resampling, the valid values are:
  - NN (nearest neighbor)
  - BI (bilinear)
  - CC (cubic convolution)
- **projtype** (*str*) – the output projection system, valid values are:
  - AEA (Albers Equal Area)
  - ER (Equirectangular)
  - GEO (Geographic Latitude/Longitude)
  - HAM (Hammer)
  - ISIN (Integerized Sinusoidal)
  - IGH (Interrupted Goode Homolosine)
  - LA (Lambert Azimuthal)
  - LCC (LambertConformal Conic)
  - MERCAT (Mercator)

  - MOL (Mollweide)
  - PS (Polar Stereographic),
  - SIN ()Sinusoidal)
  - UTM (Universal TransverseMercator)
- **utm** – the UTM zone if projection system is UTM
- **projpar** (*str*) – a list of projection parameters, for more info check the Appendix C of MODIS reprojection tool user manual https://lpdaac.usgs.gov/content/download/4831/22895/file/mrt41_usermanual_032811.pdf
- **bound** (*dict*) – dictionary with the following keys:
  - max_lat
  - max_lon
  - min_lat
  - min_lon

`getGranule``()`
Set the GranuleURMetaData element

`getRoot``()`
Set the root element

`retBoundary``()`
Return the maximum extend (Bounding Box) of the MODIS file as dictionary

`retBrowseProduct``()`
Return the BrowseProduct element

`retCollectionMetaData``()`
Return the CollectionMetaData element as dictionary

`retDTD``()`
Return the DTDVersion element

`retDataCenter``()`
Return the DataCenterId element

`retDataFiles``()`
Return the DataFiles element as dictionary

`retDataGranule``()`
Return the ECSDataGranule elements as dictionary

`retDbID``()`
Return the DbID element

`retGranuleUR``()`
Return the GranuleUR element

`retInputGranule``()`
Return the input files (InputGranule) used to process the considered file

`retInsertTime``()`
Return the InsertTime element

`retLastUpdate``()`
Return the LastUpdate element

`retMeasure``()`
Return statistics of QA as dictionary

`retPGEVersion``()`
Return the PGEVersion element

`retPSA``()`
Return the PSA values as dictionary, the PSAName is the key and and PSAValue is the value

`retPlatform``()`
Return the platform values as dictionary.

`retRangeTime``()`
Return the RangeDateTime elements as dictionary

class `pymodis.parsemodis.`**`parseModisMulti`**`(`*hdflist*`)`
A class to obtain some variables for the xml file of several MODIS tiles. It can also create the xml file

Parameters **hdflist** (*list*) – python list containing the hdf files

`valBound``()`
Function return the Bounding Box of mosaic

`valBrowseProduct``(`*obj*`)`
Function to add BrowseGranuleId

Parameters **obj** – element to add BrowseGranuleId

`valCollectionMetaData``(`*obj*`)`
Function to add CollectionMetaData

Parameters **obj** – element to add CollectionMetaData

`valDTD``(`*obj*`)`
Function to add DTDVersion

Parameters **obj** – element to add DTDVersion

`valDataCenter``(`*obj*`)`
Function to add DataCenter

Parameters **obj** – element to add DataCenter

`valDataFiles``(`*obj*`)`
Function to add DataFileContainer

Parameters **obj** – element to add DataFileContainer

`valDataGranule``(`*obj*`)`
Function to add DataFileContainer

Parameters **obj** – element to add DataFileContainer

`valDbID``(`*obj*`)`
Function to add DbID

Parameters **obj** – element to add DbID

valGranuleUR(*obj*)
Function to add GranuleUR

Parameters **obj** – element to add GranuleUR

valInputPointer(*obj*)
Function to add InputPointer

Parameters **obj** – element to add InputPointer

valInsTime(*obj*)
Function to add the minimum of InsertTime

Parameters **obj** – element to add InsertTime

valInsertTime(*obj*)
Function to add InsertTime elements

Parameters **obj** – element to add InsertTime elements

valLastUpdate(*obj*)
Function to add LastUpdate elements

Parameters **obj** – element to add LastUpdate elements

valMeasuredParameter(*obj*)
Function to add ParameterName

Parameters **obj** – element to add ParameterName

valPGEVersion(*obj*)
Function to add PGEVersion

Parameters **obj** – element to add PGEVersion

valPSA(*obj*)
Function to add PSA

Parameters **obj** – element to add PSA

valPlatform(*obj*)
Function to add Platform elements

Parameters **obj** – element to add Platform elements

valRangeTime(*obj*)
Function to add RangeDateTime

Parameters **obj** – element to add RangeDateTime

writexml(*outputname*, *pretty=True*)
Write a xml file for a mosaic

Parameters

- **outputname** (*str*) – the name of output xml file
- **pretty** (*bool*) – write prettyfy output, by default true

4.3 `convertmodis` module

Convert MODIS HDF file to GeoTiff file or create a HDF mosaic file for several tiles using Modis Reprojection Tools.

Classes:

- `convertModis`
- `createMosaic`
- `processModis`

Functions:

- `checkMRTpath()`

`pymodis.convertmodis.`**`checkMRTpath`**`(`*mrtpath*`)`

Function to check if MRT path it correct

Parameters **mrtpath** (*str*) – the path to MRT directory

Returns The path to 'bin' and 'data' directory inside MRT path

class `pymodis.convertmodis.`**`convertModis`**`(`*hdfname*, *confile*, *mrtpath*`)`

A class to convert modis data from hdf to tif using resample (from MRT tools)

Parameters

- **hdfname** (*str*) – the full path to the hdf file
- **confile** (*str*) – the full path to the paramater file
- **mrtpath** (*str*) – the full path to mrt directory which contains the bin and data directories

`executable``()`

Return the executable of resample MRT software

`run``()`

Exec the convertion process

class `pymodis.convertmodis.`**`createMosaic`**`(`*listfile*, *outprefix*, *mrtpath*, *subset=False*`)`

A class to convert several MODIS tiles into a mosaic

Parameters

- **listfile** (*str*) – the path to file with the list of HDF MODIS file
- **outprefix** (*str*) – the prefix for output files
- **mrtpath** (*str*) – the full path to mrt directory which contains the bin and data directories
- **subset** (*str*) – a string composed by 1 and 0 according with the layer to mosaic. The string should something like '1 0 1 0 0 0 0'

`executable``()`

Return the executable of mrtmosaic MRT software

`run``()`

Exect the mosaic process

write_mosaic_xml()

Write the XML metadata file for MODIS mosaic

class pymodis.convertmodis.**processModis**(*hdfname*, *confile*, *mrtpath*)

A class to process raw modis data from hdf to tif using swath2grid (from MRT Swath tools)

Parameters

- **hdfname** (*str*) – the full path to the hdf file
- **confile** (*str*) – the full path to the paramater file
- **mrtpath** (*str*) – the full path to mrt directory which contains the bin and data directories

executable()

Return the executable of resample MRT software

run()

Exec the convertion process

4.4 `convertmodis_gdal` module

Convert MODIS HDF file using GDAL Python bindings. It can create GeoTiff file (or other GDAL supported formats) or HDF mosaic file for several tiles.

Classes:

- `file_info`
- `createMosaicGDAL`
- `convertModisGDAL`

Functions:

- `getResampling()`
- `raster_copy()`
- `raster_copy_with_nodata()`

class `pymodis.convertmodis_gdal.`**`convertModisGDAL`**(*hdfname*, *prefix*, *subset*, *res*, *outformat='GTiff'*, *epsg=None*, *wkt=None*, *resampl='NEAREST_NEIGHBOR'*, *vrt=False*)

A class to convert modis data from hdf to GDAL formats using GDAL

Parameters

- **hdfname** (*str*) – name of input data
- **prefix** (*str*) – prefix for output data
- **subset** (*str*) – the subset to consider
- **res** (*int*) – output resolution
- **outformat** (*str*) – output format, it is possible to use all the supported GDAL format
- **epsg** (*int*) – the EPSG code for the preojection of output file
- **wkt** (*str*) – the WKT string for the preojection of output file
- **resampl** (*str*) – the resampling method to use
- **vrt** (*bool*) – True to read GDAL VRT file created with createMosaicGDAL

`run`()

Reproject all the subset of choosen layer

`run_vrt_separated`()

Reproject VRT created by createMosaicGDAL, function write_vrt with sepatated=True

class `pymodis.convertmodis_gdal.`**`createMosaicGDAL`**(*hdfnames*, *subset*, *outformat='HDF4Image'*)

A class to mosaic modis data from hdf to GDAL formats using GDAL

Parameters

- **hdfnames** (*list*) – a list containing the name of tile to mosaic

- **subset** (*str*) – the subset of layer to consider
- **outformat** (*str*) – the output format to use, this parameter is not used for the VRT output, supported values are HDF4Image, GTiff, HFA, and maybe something else not tested.

run (*output*)
Create the mosaik

Parameters **output** (*str*) – the name of output file

write_mosaic_xml (*prefix*)
Write the XML metadata file for MODIS mosaic

Parameters **prefix** (*str*) – the prefix for the XML file containing metadata

write_vrt (*output*, *separate=True*)
Write VRT file

Parameters

- **output** (*str*) – the prefix of output file
- **separate** (*bool*) – True to write a VRT file for each band, False to write an unique file

class pymodis.convertmodis_gdal.**file_info**
A class holding information about a GDAL file.

Class copied from gdal_merge.py

Parameters **filename** (*str*) – Name of file to read.

Returns 1 on success or 0 if the file can't be opened.

copy_into (*t_fh*, *s_band=1*, *t_band=1*, *nodata_arg=None*)
Copy this files image into target file.

This method will compute the overlap area of the file_info objects file, and the target gdal.Dataset object, and copy the image data for the common window area. It is assumed that the files are in a compatible projection. no checking or warping is done. However, if the destination file is a different resolution, or different image pixel type, the appropriate resampling and conversions will be done (using normal GDAL promotion/demotion rules).

Parameters

- **t_fh** – gdal.Dataset object for the file into which some or all of this file may be copied.
- **s_band** –
- **t_band** –
- **nodata_arg** –

Returns 1 on success (or if nothing needs to be copied), and zero one failure.

init_from_name (*filename*)
Initialize file_info from filename

pymodis.convertmodis_gdal.**getResampling** (*res*)
Return the GDAL resampling method

Parameters **res** (*str*) – the string of resampling method

`pymodis.convertmodis_gdal.`**`raster_copy`**(*s_fh*, *s_xoff*, *s_yoff*, *s_xsize*, *s_ysize*, *s_band_n*, *t_fh*, *t_xoff*, *t_yoff*, *t_xsize*, *t_ysize*, *t_band_n*, *nodata=None*)

Copy a band of raster into the output file.

Function copied from gdal_merge.py

`pymodis.convertmodis_gdal.`**`raster_copy_with_nodata`**(*s_fh*, *s_xoff*, *s_yoff*, *s_xsize*, *s_ysize*, *s_band_n*, *t_fh*, *t_xoff*, *t_yoff*, *t_xsize*, *t_ysize*, *t_band_n*, *nodata*)

Copy a band of raster into the output file with nodata values.

Function copied from gdal_merge.py

4.5 `qualitymodis` module

A class for the extraction and transformation of MODIS quality layers to specific information

Classes:

- `QualityModis`

class `pymodis.qualitymodis.`**`QualityModis`**(*infile*, *outfile*, *qType=None*, *qLayer=None*, *pType=None*)

A Class for the extraction and transformation of MODIS quality layers to specific information

Parameters

- **infile** (*str*) – the full path to the hdf file
- **outfile** (*str*) – the full path to the parameter file

`exportData`()

writes calculated QA values to physical .tif file

`loadData`()

loads the input file to the object

`loadQAArray`()

loads the QA layer to the object

`qualityConvert`(*modisQaValue*)

converts encoded Bit-Field values to designated QA information

`run`()

Function defines the entire process

`setProductGroup`()

read productGroup from Metadata of hdf file

`setProductType`()

read productType from Metadata of hdf file

`setQAGroup`()

set QA dataset group type

`setQALayer`()

function sets the input path of the designated QA layer

4.6 `optparse_required` module

Module to extend optparse, it add required options and new types to use into gui module.

Classes:

- `OptionWithDefault`
- `OptionParser`

class `pymodis.optparse_required.`**`OptionParser`**(***kwargs*)

Bases: `optparse.OptionParser`

Extend optparse.OptionParser

`check_values`(*values*, *args*)

Check if value is required for an option

class `pymodis.optparse_required.`**`OptionWithDefault`**(**opts*, ***attrs*)

Bases: `optparse.Option`

Extend optparse.Option add required to the attributes and some new types for the GUI

`ATTRS` **= ['action', 'type', 'dest', 'default', 'nargs', 'const', 'choices', 'callback', 'callback_args', 'callback_**

`TYPES` **= ('string', 'int', 'long', 'float', 'complex', 'choice', 'file', 'output', 'directory')**

4.7 `optparse_gui` module

A drop-in replacement for optparse ("import optparse_gui as optparse") Provides an identical interface to optparse(.OptionParser), But displays an automatically generated wx dialog in order to enter the options/args, instead of parsing command line arguments

Classes:

- `OptparseDialog`
- `UserCancelledError`
- `Option`
- `OptionParser`

Functions:

- `checkLabel()`

class `pymodis.optparse_gui.`**`Option`**(**opts*, ***attrs*)

Bases: `optparse.Option`

Extended optparse.Option class

`ACTIONS` **= ('store', 'store_const', 'store_true', 'store_false', 'append', 'append_const', 'count', 'callback',**

`ATTRS` **= ['action', 'type', 'dest', 'default', 'nargs', 'const', 'choices', 'callback', 'callback_args', 'callback_**

`SUPER`

alias of `Option`

`TYPED_ACTIONS` **= ('store', 'append', 'callback', 'group_name')**

`TYPES` **= ('string', 'int', 'long', 'float', 'complex', 'choice', 'file', 'output', 'directory', 'group_name')**

class `pymodis.optparse_gui.`**`OptionParser`**(**args*, ***kwargs*)

Bases: `optparse.OptionParser`

Extended optparse.OptionParser to create the GUI for the module

`SUPER`

alias of `OptionParser`

`error`(*msg*)

Return an error message with wx.MessageDialog

Parameters **msg** (*str*) – is the error string to pass to message dialog

`parse_args`(*args=None*, *values=None*)

This is the heart of it all overrides optparse.OptionParser.parse_args

Parameters

- **arg** – is irrelevant and thus ignored, it's here only for interface compatibility
- **values** – is irrelevant and thus ignored, it's here only for interface compatibility

class `pymodis.optparse_gui.`**`OptparseDialog`**(*optParser*, *title*, *parent=None*, *ID=0*, *pos=wx.Point(-1, -1)*, *size=wx.Size(-1, -1)*, *style=536877120*)

Bases: `wx._windows.Dialog`

The dialog presented to the user with dynamically generated controls, to fill in the required options.

Parameters

- **optParser** – the optparse object
- **title** (*str*) – the title to add in the GUI
- **parent** – the parent GUI
- **ID** (*int*) – the ID of GUI
- **pos** – the position of GUI
- **size** – the dimension of GUI
- **style** – the style of GUI

Based on the wx.Dialog sample from wx Docs & Demos

`getOptionsAndArgs``()`
Parse the options and args

Returns a dictionary of option names and values, a sequence of args

`onBrowse``(`*event*`)`
Choose file

`onText``(`*event*`)`
File changed

exception `pymodis.optparse_gui.`**`UserCancelledError`**
Bases: `exceptions.Exception`

??

`pymodis.optparse_gui.`**`checkLabel`**`(`*option*`)`
Create the label for an option, it add the required string if needed

Parameters **option** – and Option object

PYTHON MODULE INDEX