bernutils Documentation

Release 0.1

xp,da

July 08, 2015

CONTENTS

Contents:

CHAPTER

ONE

BERNUTILS INTRODUCTION

National Technical University of Athens *Dionysos Satellite Observatory Higher Geodesy Laboratory*

Documentation for the bernutils Python package

The *bernutils* package is a collection of Python modules to assist the automatic processing of GNSS data at the National Technical University of Athens, carried out via the Bernese v5.2 GNSS Software.

CHAPTER

TWO

MODULE : BCRD

This module contains the class **crdfile** which represents a Bernese v5.2 station coordinate file.

A class named **crdpoints** is also included, to make easier the handling of information recorded in the coordinate files.

Station Coordinate files (extension .CRD), contain coordinate components, names and flags for a list of stations. Corrdinates are always in a Cartesian reference frame, as [x,y,z] components, in meters.

Station Information file format format is very strict, and most of the functions/modules depend that this format is kept. For an example of .CRD files, see ftp://ftp.unibe.ch/aiub/BSWUSER52/STA/ and the collection of .CRD files placed there.

2.1 Documentation

class `bcrd.`**`crdpoint`**(*name=''*, *number=''*, *xcmp='.0'*, *ycmp='.0'*, *zcmp='.0'*, *flag=''*)

Class to represent a (GNSS/geodetic) Point as recorded in a Bernese format .CRD file.

`asString`(*aa=1*)

Compile a Bernese v5.2 .CRD file record line, using the instance's attributes. `aa` is the number of station (can be any positive integer), written at the begining of the returned line.

Returns A .CRD record line.

`flag_` = ''

flag

`name`(*use_marker_number=True*)

Return the full name of a station, i.e. `marker_name` + `marker_number`. If `use_marker_number` is set to `False` then the instance's `marker number` will not be included in the name returned.

`name_` = ''

Station name (4-digit); e.g. 'ANKR'

`number_` = ''

Station number; e.g. '20805M002'

`setFromCrdLine`(*line*)

Set (re-initialize) a crdpoint from a .CRD data line. Bernese v5.2 .CRD files have a strict format and this function expects such a format to be followed.

Parameters **line** – A line containing coordinate information, as extracted (read) from a .CRD file

Returns Nothing

`xcmp_` = 0.0

x-component

ycmp_ = 0.0
y-component

zcmp_ = 0.0
z-component

class bcrd.**crdfile**(*filename*)
A class to hold a Bernese v5.2 format .CRD file.

filename_ = ''
the name of the file

getFileHeader()
Return the header of a .CRD file as a list of lines, with no trailing newline chars.

Returns A list of lines included in the file header.

getListOfPoints(*stalst=None*, *disregard_number=False*)
Read points off from a .CRD file; return all points as list. If the optional argument `stalst` is given, (which is supposed to hold a list of station names), then only stations matched in the `stalst` list will be returned. By matched, i mean that the tuple `(marker_name, marker_number)` is the same for both stations. If `disregard_number` is set to `True` and `stalst` is other than `None`, then the comparisson of station names, will only be performed using the 4char station id (i.e. `self.name_`) and *NOT* the marker number.

Returns A list of points (i.e. `crdpoint` s)

2.2 Examples

CHAPTER

THREE

MODULE : BGPS

This module contains the class **gpsoutfile** which represents a Bernese v5.2 GPSEST output file.

3.1 Documentation

class bgps.**gpsoutfile**(*filename*)

A class to hold a Bernese v5.2 GPSEST output file

findFirstLine(*stream*, *line*, *eof_line='>>>'*, *max_lines=1000*)

Given a GPSEST output file, try to match the line passed as `line`, until `eof_line` is not matched and no more than `max_lines` are read. If the line is found, the stream input position, will be returned at the end of the matched line.

Parameters

- **stream** – The (calling) instance's input stream.
- **line** – The prototype line to match.
- **eof_line** – EOF record.
- **max_lines** – Max lines to be read befor quiting.

Returns On success, the matched line; input buffer is set at the end of the matched line.

Warning Note that the search will start from the current get position of the stream and *NOT* from the begining of the file.

getBaselineList()

This function will try to read all baselines processed in the run, and report their information. The baselines are read from the section: *2. OBSERVATION FILES, MAIN CHARACTERISTICS* Two tables are read, namely: [FILE OBSERVATION FILE HEADER OBSERVATION FILE SESS RECEIVER 1 RECEIVER 2] and [FILE TYP FREQ. STATION 1 STATION 2 SESS FIRST OBSERV.TIME #EPO DT #EF #CLK ARC #SAT W 12 #AMB L1 L2 L5 RM] and the concatenated list is returned, i.e.: [aa, baseline_name, type, frequency, station1, station2, first_observation, # of epochs]

getCrdSolInfo()

Given a GPSEST output file, this function will try to read information regarding the (solution) coordinate results. The information is collected from the table: 'NUM STATION NAME PARAMETER A PRIORI VALUE NEW VALUE NEW- A PRIORI RMS ERROR 3-D ELLIPSOID 2-D ELLIPSE' for every stations already mentioned in the instances station list. So make sure, the instances station list is already filled. The return list, contains a list for every station, in the following format: [name,3x(a-priori,estimated,new-old,rms),3x(new-old,rms)] for X, Y, Z HGT, LAT, LON TODO A same block of information maybe available in the section 'RESULTS PART 2'. Try reading that before reading the blok from 'RESULTS PART 1'.

getHeaderInfo()
Given a GPSEST output file, try to read the header information and return in in a list as: campaign_name, doy, session, year, date_run, username, # of stations

3.2 Examples

CHAPTER

FOUR

MODULE : BSTA

This module contains the class **stafile** which represents a Bernese v5.2 station information file.

Station Information files (extension .STA), contain information for GPS/GNSS stations regarding naming, equipment, etc, all of which are related to certain time intervals (epochs). In most cases when using the functions/modules of a **stafile** instance to collect information, an epoch must be supplied (in addition of-course to a station name).

Station Information file format format is very strict, and most of the functions/modules depend that this format is kept. For an example of .STA files, see ftp://ftp.unibe.ch/aiub/BSWUSER52/STA/ and the collection of .STA files placed there.

4.1 Documentation

class `bsta.`**`stafile`**(*filename*)

A station information class, to represent Bernese v5.2 format .STA files.

`findStationType01`(*station*, *epoch=None*)

This function will search for the station information (concerning a specific station) included in the *'TYPE 01'* block. Given a station name (e.g. 'ANKR' or 'ANKR 20805M002'), it will try to match the information line provided in the *'TYPE 01'* block, with a flag of *'001'* and **NOT** *'003'*. The station matching is **NOT** performed with the column *'STATION NAME'*, but with *'OLD STATION NAME'* In some cases, it might be necessary to also suply the epoch for which the info is needed (some times the stations are renamed and different names are adopted previous before and after a certain epoch).

Parameters

- **station** – The name of the station to match.
- **epoch** – A `datetime` object, epoch for which the info is wanted.

Returns The *'TYPE 01'* block record for this station (for this epoch).

TODO [If a renaming line (flag 003) is encountered it is skipped.] What should i do with this line ??

`findStationType02`(*station*, *epoch=None*)

This function will search for station info recorded in *'TYPE 002'* and return it. The station name provided, will be resolved using the *'TYPE 001'* block using the function `findStationType01`. Hence, the name used to match info in the *'TYPE 002'* block will be the column *'STATION NAME'* E.g., given station name 'ANKR' and using the CODE.STA file, we will get the name 'ANKR 20805M002' and we will search the block 'TYPE 002' for this name. Note that in case of renaming, a specific date may be needed (see `findStationType01`). If no date is provided, all entried for the station will be matched and returned; else, only the one withing the specified interval will be returned.

Parameters

- **station** – The name of the station to match.
- **epoch** – A `datetime` object, epoch for which the info is wanted.

Returns A list of *'TYPE 02'* block records for this station (for this epoch).

findTypeStart (*stream*, *type*, *max_lines=1000*)
Given a (sta) input file stream, go to the line where a specific type starts. E.g. if the type specified is *'1'*, then this function will search for the line: *'TYPE 001: RENAMING OF STATIONS'*. The line to search for, is compiled using the `type` and the `__type_names` dictionary. If the line is not found after `max_lines` are read, then an exception is thrown. In case of sucess, the (header) line is returned, and the stream buffer is placed at the end of the header line. Note that the `type` parameter must be a positive integer in the range `[1, len(___type_names)]`.

Parameters

- **stream** – The input stream for this istance.
- **type** – The number of type to search for.
- **max_lines** – Max lines to read before quiting.

Returns On sucess, the matched line; the input stream is left at the end of the matched line.

getStationAntenna (*station*, *epoch=None*)
Find and return the antenna type, as recorded in the *'TYPE 002'* block

getStationName (*station*, *epoch=None*)
Find and return the station name, as recorded in the *'TYPE 001'* block

getStationReceiver (*station*, *epoch=None*)
Find and return the receiver type, as recorded in the *'TYPE 002'* block

4.2 Examples

Example usage of the class **stafile**, using CODE's .STA file (available at <ftp://ftp.unibe.ch/aiub/BSWUSER52/STA/CODE.STA>)

```
## Test Program
x = StaFile('CODE.STA')
## a renaming takes place, so this will fail if no epoch is given
ln1 = x.findStationType01('S071',datetime.datetime(2008,01,01,01,00,00))
print ln1
## a renaming takes place, so this will fail if no epoch is given
ln1 = x.findStationType01('S071',datetime.datetime(2005,01,01,01,00,00))
print ln1
## following two should be the same
ln1 = x.findStationType01('OSN1 23904S001',datetime.datetime(2005,01,01,01,00,00))
print ln1
ln1 = x.findStationType01('OSN1 23904S001')
print ln1
## a renaming takes place, so this will fail if no epoch is given
ln2 = x.findStationType02('S071',datetime.datetime(2005,01,01,01,00,00))
print ln2
## return all entries, for all epochs
ln2 = x.findStationType02('ANKR')
print ln2
## return entry for a specific interval
ln2 = x.findStationType02('ANKR',datetime.datetime(2015,07,01,01,00,00))
print ln2
```

```
print x.getStationName('S071',datetime.datetime(2008,01,01,01,00,00))
print x.getStationName('S071',datetime.datetime(2005,01,01,01,00,00))
print x.getStationName('ANKR')
print x.getStationAntenna('S071',datetime.datetime(2008,01,01,01,00,00))
print x.getStationAntenna('S071',datetime.datetime(2005,01,01,01,00,00))
print x.getStationAntenna('ANKR',datetime.datetime(2005,01,01,01,00,00))
print x.getStationReceiver('S071',datetime.datetime(2008,01,01,01,00,00))
print x.getStationReceiver('S071',datetime.datetime(2005,01,01,01,00,00))
print x.getStationReceiver('ANKR',datetime.datetime(2005,01,01,01,00,00))
```

CHAPTER

FIVE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*