SuperMAG Python Client 1.0

SuperMAG Web Service API Python Client Documentation

Python 3 required. Package 'pandas' required (pip install pandas). If your worksite, like APL, requires SSL certs to access URLs, package 'certifi' must also be installed (pip install certifi)

◆ (status,stations)=SuperMAGGetInventory(userid,start,extent)

Python function that retrieves a list of available stations for a given event.

Parameters

userid your supermag user id
start start date of event, either in the format 'YYYY-MM-DDThhmm' or as a list [YYYY, MM, DD, hh, mm] (seconds are optional)
extent extent or length of the event in seconds (3600= 1 hour, 86400 = 1 day)

Returns

List of available stations. If there was an error, return is the error message.

Example Usage

```
start=[2019,11,15,10,40,00] # alt: start='2019-11-15T10:40'
(status,stations) = SuperMAGGetInventory(userid,start,3600)
print(stations)
for i in range(len(stations)-1):
   print(i,stations[i])
```

◆ (status,sm_data)=SuperMAGGetData(userid,start,extent,flags,station,FORMAT='list')

Python function that retrievies station magnetometer data for a given event and IAGA station code. By default it returns the data as a pandas dataframe. You can add the optional 'FORMAT' keyword to tell it to return the data as a python list instead of a pandas dataframe (the default).

Parameters

userid your supermag user id
yr start date of event, either in the format 'YYYY-MM-DDThhmm' or as a list [YYYY, MM, DD, hh, mm] (seconds are optional)
extent extent or length of the event in seconds (3600= 1 hour, 86400 = 1 day)
station IAGA code of the requested station
flags list in string or list form of which data items to return and processing flags to use (see below). The full list of data items is either 'all' or 'mlt,mag,geo,decl,sza'. Flags can alternately be in list format, e.g. ["mlt" "mag" "geo" "decl" "sza"]. Processing flags available are 'delta=start', 'baseline=none', 'baseline=yearly'. Flags are not case-sensitive

MLT	(optional) If supplied, The MLT/MCOLAT of the station will be returned in the two dimensional array of length extent/60 specified by MLT.
MAG	(optional) If supplied, The Magnetic coordinates of the station will be returned in the two dimensional array of length extent/60 specified by MAG.
GEO	(optional) If supplied, The Geographic coordinates of the station will be returned in the two dimensional array of length extent/60 specified by GEO.
DECL	(optional) If supplied, The Declination from IGRF Model will be returned in the array of length extent/60 specified by DECL.
SZA	(optional) If supplied, The solar zenith angle will be returned in the array of length extent/60 specified by SZA.
DELTA	(optional) If the keyword DELTA is supplied, The baseline NEZ vector start values will be subtracted from the NEZ vector components in the resulting n, e, and z lists.
BASELINE	(optional) If BASELINE is specified, It must be set to one of three values: "baseline='all'" (default): Subtract both the daily and yearly NEZ baselines; "baseline='yearly'": Subtract the yearly NEZ baseline, but do not subtract the daily NEZ baseline; "baseline='none'": Do not subtract either the yearly or the daily NEZ baseline

FORMAT='list' Optional, if given as "FORMAT='list'", routine will return a python list instead of a pandas dataframe

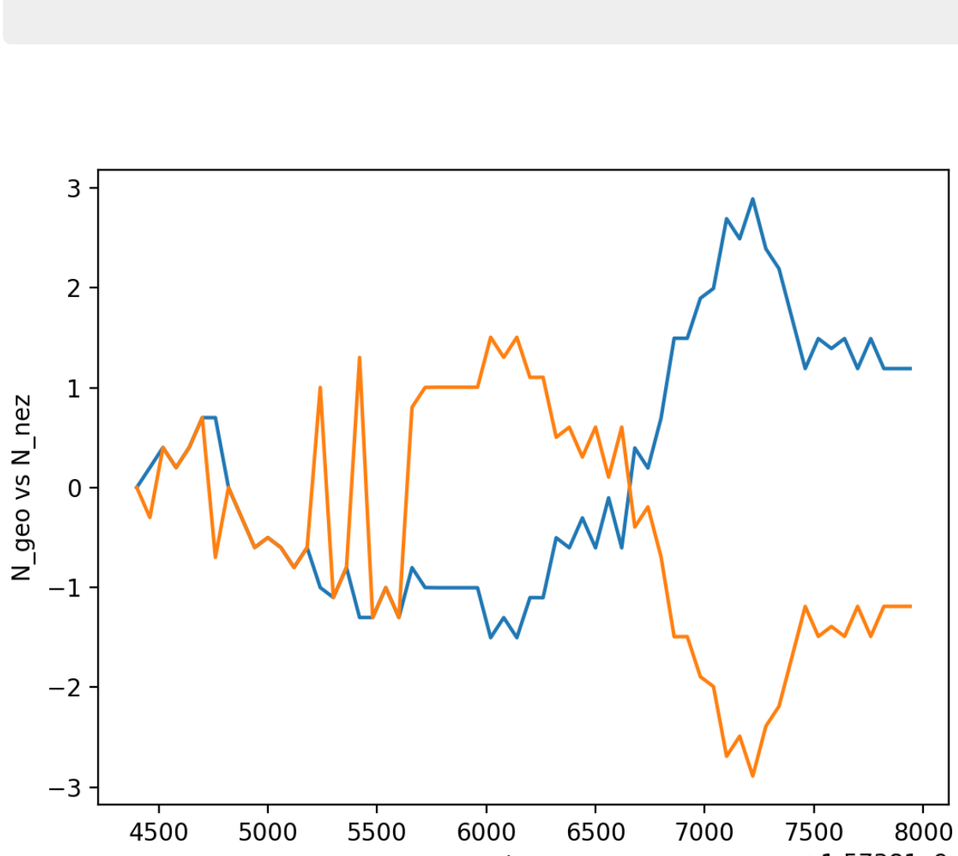
Returns

Structure with all return data. If there was an error, return is the error message. The format of the returns is as follows.

tval The time of the samples is returned as the structure element tval. The time array is an array of double precision numbers giving the time since 1970-01-01 0:00UTC (This is a standard representation of time on computer systems).
ext The binned duration for each sample is returned, typically '60' representing the 1-minute bins of standard SuperMAG data
iaga The 3-letter station code provided is returned in the structure, useful for identification when you have multiple sets of data.
N The N vector component is returned in the two structure element arrays of length extent/60 specified by N. The second dimension refers to the coordinate system, so 'N.nez' contains the component of the vector in the standard NEZ coordinates, 'N.geo' contains the geographic mapping of the N vector component.
E The E vector component is returned in the two structure element arrays of length extent/60 specified by E. The second dimension refers to the coordinate system, so 'E.nez' contains the component of the vector in the standard NEZ coordinates, 'E.geo' contains the geographic mapping of the E vector component.
Z The A vector component is returned in the two structure element arrays of length extent/60 specified by Z. The second dimension refers to the coordinate system, so 'Z.nez]' contains the component of the vector in the standard NEZ coordinates, 'Z.geo' contains the geographic mapping of the Z vector component.
mlt (optional) If supplied, The MLT/MCOLAT of the station will be returned in the two structure element arrays 'mlt' and 'mcolat' of length extent/60 specified by MLT.
mag (optional) If supplied, The Magnetic coordinates of the station will be returned in the two structure element arrays 'mlat' and 'mlon' of length extent/60 specified by MAG.
geo (optional) If supplied, The Geographic coordinates of the station will be returned in the two structure element arrays 'glon' and 'glat' of length extent/60 specified by GEO.
decl (optional) If supplied, The Declination from IGRF Model will be returned as a structure element array 'decl' of length extent/60 specified by DECL.
sza (optional) If supplied, The solar zenith angle will be returned as a structure element array 'sza' of length extent/60 specified by SZA.

Example Usage

```
tval=data.tval
mlt=data.mlt
### Python way
N_nez = [temp['nez'] for temp in data.N]
N_geo = [temp['geo'] for temp in data.N]
### or, supermag helper shorthand way
N_nez = sm_grabme(data,'N','nez')
N_geo = sm_grabme(data,'N','geo')
#
plt.plot(tval,N_nez)
plt.plot(tval,N_geo)
plt.ylabel('N_geo vs N_nez')
plt.xlabel('date')
plt.show()
```



◆ (status,sm_indices)=fetchSuperMAG(userid,start,extent,flags,FORMAT='list')

Python function that retrieves a set of magnetic indices for a given event. By default it returns the data as a pandas dataframe. You can add the optional 'FORMAT' keyword to tell it to return the data as a python list instead of a pandas dataframe (the default).

Parameters

userid your supermag user id
yr start date of event, either in the format 'YYYY-MM-DDThhmm' or as a list [YYYY, MM, DD, hh, mm] (seconds are optional)
extent extent or length of the event in seconds (3600= 1 hour, 86400 = 1 day)
flags list in string or list form of which data items to return and processing flags to use (see below). The full list of data items is either 'all' or any subset, e.g. 'sme, sunsme, darksme'. Flags can alternately be in list format, e.g. ["sme" "sunsme" "darksme"]. Several flags have alternative names which you are free to use (these are derived from the set of tags the SuperMAG web server uses natively.) Flags are not case-sensitive.

SME	(optional) If supplied, the SME indice will be returned in the structure array '.SME' (See definition of SME indice)
SML	(optional) If supplied, the SML indice will be returned in the structure array '.SML' (See definition of SML indice)
SMU	(optional) If supplied, the SMU indice will be returned in the structure array '.SMU' (See definition of SMU indice)
NUM	(optional) If supplied, the number of stations used to compute the SME indices will be returned in the structure array '.SMEnum'
(optional) the following options return additional data items, but only if SME, SML and/or SMU is set (for SME, returns both .SMU and .SML entries; for SMU, only .SMU entries; for SML, only .SML entries)	
MLAT	(optional) If supplied, the magnetic latitude of the SME indice will be returned in the structure array '.SMLmlat' and '.SMUmlat'
MLT	(optional) If supplied, the magnetic local time of the SME indice will be returned in the structure array '.SMLmlt' and '.SMUmlt'
GLAT	(optional) If supplied, the geographic latitude of the SME indice will be returned in the structure array '.SMLglat' and '.SMUglat'
GLON	(optional) If supplied, the geographic longitude of the SME indice will be returned in the structure array '.SMLglon' and '.SMUglon'
STID	(optional) If supplied, the IAGA station codes of the stations used to compute the SME indices will be returned in the structure array '.SMLstid' and '.SMUstid'
baseall	(optional) If supplied, is the equivalent of the set of 'sme,sml,smu,mlat,mlt,glat,glon,stid,num'
SUNSME (alt: smes)	(optional) If supplied, the Sunlit SME indice will be returned in the structure array '.SMEs' (See definition of Sunlit SME indice)
SUNSML (alt: smls)	(optional) If supplied, the Sunlit SML indice will be returned in the structure array '.SMLs' (See definition of Sunlit SML indice)
SUNSMU (alt: smus)	(optional) If supplied, the Sunlit SMU indice will be returned in the structure array '.SMUs' (See definition of Sunlit SMU indice)
SUNNUM (alt: nums)	(optional) If supplied, the number of stations used to compute the Sunlit SME indices will be returned in the structure array '.sunnum'
(optional) the following options return additional data items, but only if SMEs, SMLs and/or SMUs is set (for SMEs, returns both .SMUs and .SMLs entries; for SMUs, only .SMUs entries; for SMLs, only .SMLs entries)	
SUNMLAT (alt: mlats)	(optional) If supplied, the magnetic latitude of the Sunlit SME indice will be returned in the structure array '.SMLsmlat' and '.SMUsmlat'
SUNMLT (alt: mlts)	(optional) If supplied, the magnetic local time of the Sunlit SME indice will be returned in the structure array '.SMLsmlt' and '.SMUsmlt'
SUNGLAT (alt: glats)	(optional) If supplied, the geographic latitude of the Sunlit SME indice will be returned in the structure array '.SMLsglat' and '.SMUsglat'
SUNGLON (alt: glons)	(optional) If supplied, the geographic longitude of the Sunlit SME indice will be returned in the structure array '.SMLsglon' and '.SMUsglon'
SUNSTID (alt: stids)	(optional) If supplied, the IAGA station codes of the stations used to compute the Sunlit SME indices will be returned in the structure array '.SMLstid' and '.SMUstid'
sunall	(optional) If supplied, is the equivalent of the set of 'smes,smls,smus,mlats,mlts,glats,glons,stids,nums'
DARKSME (alt: smed)	(optional) If supplied, the Dark SME indice will be returned in the structure array '.darksme' (See definition of Dark SME indice)
DARKSML (alt: smld)	(optional) If supplied, the Dark SML indice will be returned in the structure array '.darksml' (See definition of Dark SML indice)
DARKSMU (alt: smud)	(optional) If supplied, the Dark SMU indice will be returned in the structure array '.darksmu' (See definition of Dark SMU indice)
DARKNUM (alt: numd)	(optional) If supplied, the number of stations used to compute the Dark SME indices will be returned in the structure array '.darknum'
(optional) the following options return additional data items, but only if SMEd, SMLd and/or SMUd is set (for SMEd, returns both .SMUd and .SMLd entries; for SMUd, only .SMUd entries; for SMLd, only .SMLd entries)	
DARKMLAT (alt: mlatd)	(optional) If supplied, the magnetic latitude of the Dark SME indice will be returned in the structure array '.SMLdmlat' and '.SMUdmlat'
DARKMLT (alt: mltd)	(optional) If supplied, the magnetic local time of the Dark SME indice will be returned in the structure array '.SMLdmlt' and '.SMUdmlt'
DARKGLAT (alt: glatd)	(optional) If supplied, the geographic latitude of the Dark SME indice will be returned in the structure array '.SMLdglat' and '.SMUdglat'
DARKGLON (alt: glond)	(optional) If supplied, the geographic longitude of the Dark SME indice will be returned in the structure array '.SMLdglon' and '.SMUdglon'
DARKSTID (alt: stidd)	(optional) If supplied, the IAGA station codes of the stations used to compute the Dark SME indices will be returned in the structure array '.SMLdtid' and '.SMUdtid'
darkall	(optional) If supplied, is the equivalent of the set of 'smed,smld,smud,mlatd,mltd,glatd,glond,stidd,numd'
REGIONALSME (alt: smer)	(optional) If supplied, the Regional SME indice will be returned in the structure array '.SMEr' (See definition of Regional SME indice)
REGIONALSML (alt: smlr)	(optional) If supplied, the Regional SML indice will be returned in the structure array '.SMLr' (See definition of Regional SML indice)
REGIONALSMU (alt: smur)	(optional) If supplied, the Regional SMU indice will be returned in the structure array '.SMUr' (See definition of Regional SMU indice)
REGIONALNUM (alt: numr)	(optional) If supplied, the number of stations used to compute the Regional SME indices will be returned in the structure array '.SMErnum'
(optional) the following options return additional data items, but only if SMEr, SMLr and/or SMUr is set (for SMEr, returns both .SMUr and .SMLr entries; for SMUr, only .SMUr entries; for SMLr, only .SMLr entries)	
REGIONALMLAT (alt: mlatr)	(optional) If supplied, the magnetic latitude of the Regional SME indice will be returned in the structure array '.SMLsmlat' and '.SMUsmlat'
REGIONALMLT (alt: mltr)	(optional) If supplied, the magnetic local time of the Regional SME indice will be returned in the structure array '.SMLsmlt' and '.SMUsmlt'
REGIONALGLAT (alt: glatr)	(optional) If supplied, the geographic latitude of the Regional SME indice will be returned in the structure array '.SMLsglat' and '.SMUsglat'
REGIONALGLON (alt: glonr)	(optional) If supplied, the geographic longitude of the Regional SME indice will be returned in the structure array '.SMLsglon' and '.SMUsglon'
REGIONALSTID (alt: stidr)	(optional) If supplied, the IAGA station codes of the stations used to compute the Regional SME indices will be returned in the structure array '.SMLstid' and '.SMUstid'
regall	(optional) If supplied, is the equivalent of the set of 'sme,sml,smu,mlat,mlt,glat,glon,stid,num'
SMR	(optional) If supplied, the SMR indice will be returned in the structure array '.smr' (See definition of SMR indice)
LTSMR	(optional) If supplied, the SMR LT indices will be returned in the structure arrays '.smr00','.smr06','.smr12','.smr18' (See definition of SMR LT indice)
LTNUM	(optional) If supplied, the number of stations used to compute the SMR LTN indice will be returned in the structure arrays '.smrnum00','.smrnum06','.smrnum12','.smrnum18'
NSMR	(optional) If supplied, the number of stations used to compute the SMR indices will be returned in the structure array '.nsmr'
plusall	(optional) If supplied, is the equivalent of the set of 'sme,sml,smu,mlat,mlt,glat,glon,stid,num'
all	(optional) If supplied, is the equivalent to all the above, 'baseall,sunall,darkall,regall,plusall' (but not the swi and imf keys below)
BGSE	(optional) If supplied, the Solar Wind B field (GSE) parameter will be returned in the structure array '.bgse'
BGSM	(optional) If supplied, the Solar Wind B field (GSM) parameter will be returned in the structure array '.bgsm'
VGSE	(optional) If supplied, the Solar Wind V (GSE) parameter will be returned in the structure array '.vgse'
VGSM	(optional) If supplied, the Solar Wind V (GSM) parameter will be returned in the structure array '.vgsm'
imfall	(optional) If supplied, is the equivalent of the set of 'sme,sml,smu,mlat,mlt,glat,glon,stid,num'
PDYN	(optional) If supplied, the Solar Wind Dynamic Pressure parameter will be returned in the structure array '.pdyn'
EPSILON	(optional) If supplied, the Solar Wind ε Parameter parameter will be returned in the structure array '.epsilon'
NEWELL	(optional) If supplied, the Solar Wind Newell parameter will be returned in the structure array '.newell'
CLOCKGSE	(optional) If supplied, the IMF Clock Angle (GSE) parameter will be returned in the structure array '.clockgse'
CLOCKGSM	(optional) If supplied, the IMF Clock Angle (GSM) parameter will be returned in the structure array '.clockgsm'
DENSITY	(optional) If supplied, the Solar Wind Plasma Density parameter will be returned in the structure array '.density'
swiall	(optional) If supplied, is the equivalent of the set of 'sme,sml,smu,mlat,mlt,glat,glon,stid,num'

FORMAT='list' Optional, if given as "FORMAT='list'", routine will return a python list instead of a pandas dataframe

Returns

Structure with all return data. If there was an error, return is the error message. The format of the returns is as follows.

The following data structure arrays are returned, dependent on which optional flags you requested. Note that no data is returned unless flags are specified, there is no 'default dataset'.

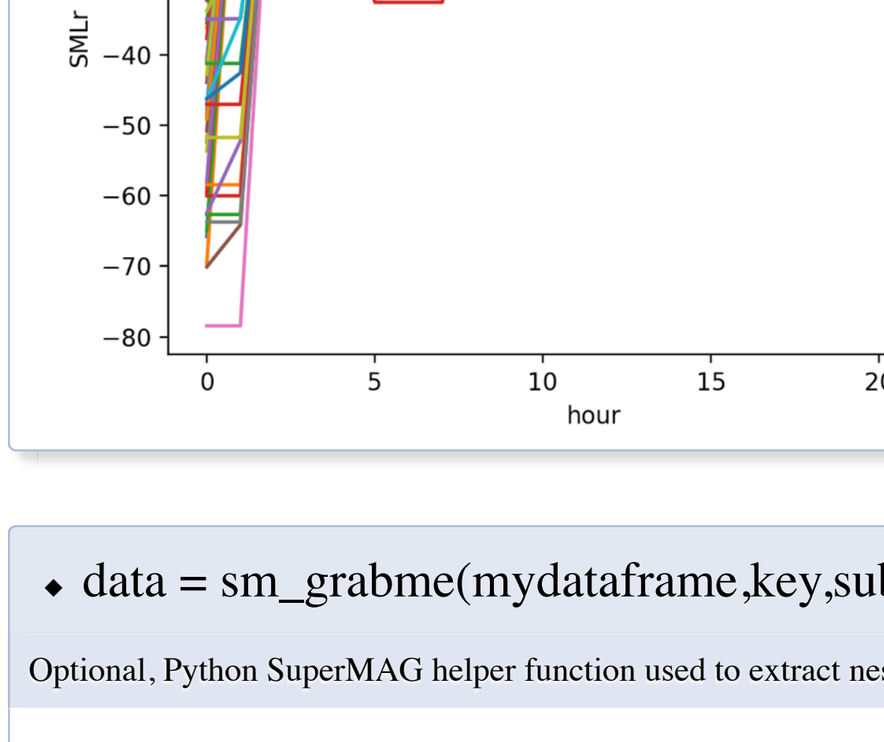
tval The time of the samples is returned as the structure element tval. The time array is a list of double precision numbers giving the time since 1970-01-01 0:00UTC (This is a standard representation of time on computer systems).
decl (optional) If supplied, The Declination from IGRF Model will be returned as a structure element array 'decl' of length extent/60 specified by DECL.
Base data: Structure elements as defined above and named: .SME, .SML, .SMLmlat, .SMLmlt, .SMLglat, .SMLglon, .SMLstid, .SMU, .SMUmlat, .SMUmlt, .SMUglat, .SMUglon, .SMUstid, .SMEnum
Sunlit data: Structure elements as defined above and named: .SMEs, .SMLs, .SMLsmlat, .SMLsmlt, .SMLsglat, .SMLsglon, .SMLsstid, .SMUs, .SMUsmlat, .SMUsmlt, .SMUsglat, .SMUsglon, .SMUsstid, .SMEsnum
Dark data: Structure elements as defined above and named: .SMEd, .SMLd, .SMLdmlat, .SMLdmlt, .SMLdglat, .SMLdglon, .SMLdstid, .SMUd, .SMUdmlat, .SMUdmlt, .SMUdglat, .SMUdglon, .SMUdstid, .SMEdnum
Regional data: Structure elements as defined above and named: .SMEr, .SMLr, .SMLrmlat, .SMLrmlt, .SMLrglat, .SMLrglon, .SMLrstid, .SMUr, .SMUrmlat, .SMUrmlt, .SMUrglat, .SMUrglon, .SMUrstid, .SMErnum
Plus data: Structure elements as defined above and named: .smr, .smrnum00, .smrnum06, .smrnum12, .smrnum18
IMF data: Structure elements as defined above and named: .bgse, .bgsm, .vgse, .vgsm
SWI data: Structure elements as defined above and named: .clockgse, .clockgsm, .density, .dynpres, .epsilon, .newell

Example Usage

```
(status,idxdata) = SuperMAGGetIndices(userid,start,3600,'swiall,density,dar\
kall,regall,smes')
  #print(status)
  #print(idxdata)
  allkeys = idxdata.keys() # an example of showing the data keys fetched

  # let us plot slices
  tval=idxdata.tval
  hours=list(range(24))
  y=idxdata.SMLr
  for i in range(len(tval)-1):
     plt.plot( hours, y[i] )
     plt.ylabel('SMLr')
     plt.xlabel('hour')
     plt.title('SMLr variation by hour, for successive days')

  plt.show()
```



◆ data = sm_grabme(mydataframe,key,subkey))

Optional, Python SuperMAG helper function used to extract nested structure items from the pandas dataframe.

Parameters

mydataframe A SuperMAG (or any) pandas dataframe with nested items.
key key to dataframe row item
subkey subkey of item in selected key data

Returns

Data subset item you requested from the dataframe.

Example Usage

```
(status,data) = SuperMAGGetData(userid,start,3600,'all')
tval =data.tval # get list of times

# data contains, in each row, item 'N' that has subkeys 'nez' and 'geo'
### Python way to access data
N_nez = [temp['nez'] for temp in data.N]
N_geo = [temp['geo'] for temp in data.N]
### Our helper function
N_nez = sm_grabme(data,'N','nez')
N_geo = sm_grabme(data,'N','geo')

# Either way, you can now use it and plot it
plt.plot(tval,N_nez)
plt.plot(tval,N_geo)
plt.ylabel('N_geo vs N_nez')
plt.xlabel('date')
plt.show()
```

◆ sm_microtest(testchoice,userid)

This is a simple unit test that runs each of the 3 fetches (stations, data, indices) for a small sample case. If this works, everything is set up accurately on your system.

Parameters

testchoice Indicate which test to run: 1 = get sample stations, 2 = get sample data, 3 = get sample indices, 4 = run all 3 tests.
userid your supermag user id

Returns

Displays and/or plots the sample data to verify functionality.

Example Usage

```
from supermag_api import *
userid='YOUR_SUPERMAG_USER_ID'
sm_microtest(4,userid)
```

◆ Optional, pandas dataframes versus Comma Separated Values (csv) files

Examples: Here is a simple example of saving to a '.csv' file, then reading the pandas data back from it.

Example Usage

```
# first fetch the data
(status,idxdata) = SuperMAGGetIndices(userid,start,3600,'all,swiall,imfall')

# to save the data, there are many formats.  Here is how to save as csv
idxdata.to_csv('mydata.csv')

# to read it back in later
import pandas as pd
import re
# you can read it into any variable name, we just used 'mydata2b' as an example
mydata2b=pd.read_csv('mydata.csv',index_col=0)

# now you can do all the above items again, with one exception: each line of the CVS file got split into a dict (key-value pairs)
# but items like 'vgse' are part of the pandas structure
# the 'd.get()' approach will _not_ work once read from csv

# goal is a list of stations
stationlist2=sm_csvitem_to_list(mydata2b.SMLrstid)
# grabs a list of stations for row 0
slist = stationlist2[0]
# grabs the first station for row 0
s1 = stationlist2[0][0]

# goal is a dict of coords or other values
vgse=sm_csvitem_to_dict(mydata2b.vgse)
# grab just the 'X' value for the 1st row of data
x = vgse[0]['X']
#grab all the 'X' values as a new list
vgse_x = [mydat['X'] for mydat in vgse]
# grab all 3 as their own list
vgse_xyz = [(mydat['X'],mydat['Y'],mydat['Z']) for mydat in vgse]
```

◆ Optional, Extended Examples

Examples: Here are some sample manipulations of the SuperMAG data and indices items. Code is provided, with python comments after the # indicating sample data that is retrieved by the code.

Example Usage

```
###################
# DATA fetches
# BARE CALL, dataframe returned
(status,mydata1a) = SuperMAGGetData(userid,start,3600,'','HBK')
mydata1a        # is 1440 rows x 6 columns dataframe
mydata1a.keys() # Index(['tval', 'ext', 'iaga', 'N', 'E', 'Z'], dtype='object')

# CALL with ALLINDICES, dataframe returned
(status,mydata1a) = SuperMAGGetData(userid,start,3600,'all','HBK')
mydata1a        # is 1440 rows x 12 columns dataframe
mydata1a.keys() # Index(['tval', 'ext', 'iaga', 'glon', 'glat', 'mlt', 'mcolat', 'decl', 'sza', 'N', 'E', 'Z'], dtype='object')

# BARE CALL, list returned
(status,mydata1b) = SuperMAGGetData(userid,start,3600,'','HBK',FORMAT='list')
len(mydata1b)  # is 1440 rows of dicts (key-value pairs)
mydata1b[0:1]  # {'tval': 1572726240.0, 'ext': 60.0, 'iaga': 'DOB', 'N': {'nez': -3.942651, 'geo': -5.964826}, 'E': {'nez': 4.492887, 'geo': 0.389075}, 'Z': {'nez': 7.608168, 'geo': 7.608168}}

# CALL with ALLINDICES, list returned (status,mydata1b) =
SuperMAGGetData(userid,start,3600,'all','HBK',FORMAT='list')
mydata1b
"""is 1440 rows of dicts (key-value pairs) mydata1b[0:1]
{'tval': 1572726240.0, 'ext': 60.0, 'iaga': 'DOB', 'glon': 9.11,
'glat': 62.07, 'mlt': 21.694675, 'mcolat': 30.361519, 'decl':
3.067929, 'sza': 124.698227, 'N': {'nez': -3.942651, 'geo':
-5.964826}, 'E': {'nez': 4.492887, 'geo': 0.389075}, 'Z': {'nez':
7.608168, 'geo': 7.608168}}"""

###################
# INDICES fetches
(status,idxdata) = SuperMAGGetIndices(userid,start,3600)
idxdata  # empty!

(status,idxdata) =
SuperMAGGetIndices(userid,start,3600,'all,swiall,imfall')
idxdata #1440 rows x 77 columns dataframe
idxdata.keys()
""" Index(['tval',
'SME', 'SML', 'SMLmlat', 'SMLmlt', 'SMLglat', 'SMLglon',
'SMLstid', 'SMU', 'SMUmlat', 'SMUmlt', 'SMUglat', 'SMUglon',
'SMUstid', 'SMEnum', 'SMEs', 'SMLs', 'SMLsmlat', 'SMLsmlt',
'SMLsglat', 'SMLsglon', 'SMLsstid', 'SMUs', 'SMUsmlat', 'SMUsmlt',
'SMUsglat', 'SMUsglon', 'SMUsstid', 'SMEsnum', 'SMEd', 'SMLd',
'SMLdmlat', 'SMLdmlt', 'SMLdglat', 'SMLdglon', 'SMLdstid', 'SMUd',
'SMUdmlat', 'SMUdmlt', 'SMUdglat', 'SMUdglon', 'SMUdstid',
'SMEdnum', 'SMEr', 'SMLr', 'SMLrmlat', 'SMLrmlt', 'SMLrglat',
'SMLrglon', 'SMLrstid', 'SMUr', 'SMUrmlat', 'SMUrmlt', 'SMUrglat',
'SMUrglon', 'SMUrstid', 'SMErnum', 'smr', 'smr00', 'smr06',
'smr12', 'smr18', 'smrnum', 'smrnum00', 'smrnum06', 'smrnum12',
'smrnum18', 'bgse', 'bgsm', 'vgse', 'vgsm', 'clockgse',
'clockgsm', 'density', 'dynpres', 'epsilon', 'newell'], dtype='object')"""
# just INDICESALL = 67 columns, above 'tval' through 'smrnum18'
# just IMFALL = 5 columns, Index(['tval', 'bgse', 'bgsm', 'vgse', 'vgsm'], dtype='object')
# just SWIALL = 7 columns, Index(['tval', 'clockgse', 'clockgsm', 'density', 'dynpres', 'epsilon', 'newell'], dtype='object')
# Dataframes are awesome!  To manipulate, just pull out what you need
import pandas as pd
# call once at the top of your code if you are using dataframes
tval = idxdata.tval
density = idxdata.density
vgse = idxdata.vgse
# or all as 1 line of code tval, density,
vgse = idxdata.tval, idxdata.density, idxdata.vgse
# note that vgse is itself a dictionary of values for X/Y/Z,
# so you can get subitems from it like this
vgse_x = [d.get('X') for d in idxdata.vgse]

################### # Using Lists instead of DataFrames # # We
also offer a list format, for users who prefer to work in python
lists (status,mydata2c) =
SuperMAGGetIndices(userid,start,3600,'all,swiall,imfall',FORMAT='list')
len(mydata2c) # is 1440 rows of dicts (key-value pairs)
mydata2c[0:1] #
"""{'tval': 1572726240.0, 'SME': 58.887299, 'SML':
-27.709055, 'SMLmlat': 73.529922, 'SMLmlt': 23.321493, 'SMLglat':
76.510002, 'SMLglon': 25.01, 'SMLstid': 'HOP', 'SMU': 31.178246,
'SMUmlat': 74.702339, 'SMUmlt': 2.090216, 'SMUglat': 79.480003,
'SMUglon': 76.980003, 'SMUstid': 'VIZ', 'SMEnum': 118, 'SMEs':
34.451469, 'SMLs': -16.599854, 'SMLsmlat': 62.368008, 'SMLsmlt':
9.399416, 'SMLsglat': 62.299999, 'SMLsglon': 209.800003,
'SMLsstid': 'T39', 'SMUs': 17.851616, 'SMUsmlat': 73.989975,
'SMUsmlt': 18.228394, 'SMUsglat': 67.93, 'SMUsglon': 306.429993,
'SMUsstid': 'ATU', 'SMEsnum': 54, 'SMEd': 58.887299, 'SMLd':
-27.709055, 'SMLdmlat': 73.529922, 'SMLdmlt': 23.321493,
'SMLdglat': 76.510002, 'SMLdglon': 25.01, 'SMLdstid': 'HOP',
'SMUd': 31.178246, 'SMUdmlat': 74.702339, 'SMUdmlt': 2.090216,
'SMUdglat': 79.480003, 'SMUdglon': 76.980003, 'SMUdstid': 'VIZ',
'SMEdnum': 64, 'SMEr': [29.685059, 29.857538, 31.387127,
41.707573, 10.320444, 10.885443, 9.604616, 13.479583, 15.471248,
15.471248, 15.714731, 5.434914, 12.13654, 11.156847, 9.62884,
14.752592, 14.752592, 24.204388, 21.41181, 21.41181, 27.121195,
46.345322, 51.403328, 51.403328], 'SMLr': [-27.709055, 1.320708,
-0.208882, -10.529325, -10.529325, -10.529325, -9.248499,
-13.123466, -16.599854, -16.599854, -16.599854, -5.449972,
-5.449972, -4.470279, -2.942272, -6.352773, -6.352773, -6.352773,
-3.560194, -3.560194, -7.514064, -22.651047, -27.709055,
-27.709055], 'SMLrmlat': [73.529922, 51.264774, 47.791527,
66.696564, 66.696564, 66.696564, 41.771515, 70.602707, 62.368008,
62.368008, 62.368008, 67.471809, 67.471809, 60.639145, 68.500282,
72.20977, 72.20977, 72.20977, 75.762718, 75.762718, 77.33667,
71.889503, 73.529922, 73.529922], 'SMLrmlt': [23.321493, 2.119074,
3.578985, 4.929673, 4.929673, 4.929673, 5.414416, 8.57761,
9.399416, 9.399416, 9.399416, 11.35623, 11.35623, 12.266475,
13.977451, 16.720993, 16.720993, 16.720993, 19.65963, 19.65963,
21.307804, 22.863134, 23.321493, 23.321493], 'SMLrglat':
[76.510002, 55.029999, 52.169998, 71.580002, 71.580002, 71.580002,
47.799999, 71.300003, 62.299999, 62.299999, 62.299999, 72.300003,
72.300003, 76.769997, 74.5, 76.510002, 76.510002], 'SMLrglon':
[25.01, 82.900002, 104.449997, 129.0, 129.0, 129.0, 132.414001,
203.25, 209.800003, 209.800003, 209.800003, 238.770004,
238.770004, 247.026001, 265.920013, 291.480011, 291.480011,
291.480011, 321.700012, 321.700012, 341.369995, 19.200001, 25.01,
25.01], 'SMLrstid': ['HOP', 'NVS', 'IRT', 'TIK', 'TIK', 'TIK',
'BRN', 'BRW', 'T39', 'T39', 'T39', 'FSP', 'FSP', 'C06', 'FCC',
'IQA', 'IQA', 'IQA', 'SUM', 'SUM', 'DMH', 'BJN', 'HOP', 'HOP'],
'SMUr': [1.976003, 31.178246, 31.178246, 31.178246, -0.208882,
0.356117, 0.356117, 0.356117, -1.128606, -1.128606, -0.885122,
-0.015059, 6.686568, 6.686568, 6.686568, 8.399819, 8.399819,
17.851616, 17.851616, 17.851616, 19.60713, 23.694275, 23.694275,
23.694275], 'SMUrmlat': [52.904049, 74.702339, 74.702339,
74.702339, 47.791527, 54.29908, 54.29908, 66.244217,
66.244217, 57.76614, 54.597057, 55.715378, 55.715378, 55.715378,
57.829525, 57.829525, 73.989975, 73.989975, 73.989975, 70.473801,
68.194489, 68.194489, 68.194489], 'SMUrmlt': [0.510692, 2.090216,
2.090216, 2.090216, 3.578985, 6.394085, 6.394085, 6.394085,
9.99274, 9.99274, 11.729218, 12.269058, 13.969843, 13.969843,
13.969843, 16.160952, 16.160952, 18.228394, 18.228394, 18.228394,
21.200783, 22.967857, 22.967857, 22.967857], 'SMUrglat':
[56.432999, 79.480003, 79.480003, 79.480003, 52.169998, 59.970001,
59.970001, 59.970001, 64.047997, 64.047997, 51.882999, 47.664001,
45.870998, 45.870998, 45.870998, 48.650002, 48.650002, 67.93,
67.93, 67.93, 70.900002, 71.089996, 71.089996, 71.089996],
'SMUrglon': [58.567001, 76.980003, 76.980003, 76.980003,
104.449997, 150.860001, 150.860001, 150.860001, 220.889999,
220.889999, 239.973999, 245.791, 264.916992, 264.916992,
264.916992, 287.549988, 287.549988, 306.429993, 306.429993,
306.429993, 351.299988, 25.790001, 25.790001, 25.790001],
'SMUrstid': ['ARS', 'VIZ', 'VIZ', 'VIZ', 'IRT', 'MGD', 'MGD',
'MGD', 'DAW', 'DAW', 'C13', 'C10', 'C08', 'C08', 'C08', 'T50',
'T50', 'ATU', 'ATU', 'ATU', 'JAN', 'NOR', 'NOR', 'NOR'],
'SMErnum': [5, 3, 3, 4, 5, 6, 6, 4, 8, 9, 12, 13, 20, 17, 17, 11,
12, 14, 12, 14, 22, 51, 51, 35], 'smr': 0.252399, 'smr00':
-0.531382, 'smr06': 0.885406, 'smr12': 1.051192, 'smr18':
-0.395618, 'smrnum': 72, 'smrnum00': 26, 'smrnum06': 23,
'smrnum12': 6, 'smrnum18': 17, 'bgse': {'X': 1.07, 'Y': -3.75,
'Z': -0.74}, 'bgsm': {'X': 1.07, 'Y': -3.82, 'Z': -0.06}, 'vgse':
{'X': -351.100006, 'Y': -5.5, 'Z': -4.0}, 'vgsm': {'X':
351.100006, 'Y': 6.128625, 'Z': -2.947879}, 'clockgse':
258.340698, 'clockgsm': 268.664337, 'density': 5.03, 'dynpres':
1.25, 'epsilon': 29.468521, 'newell': 2504.155029} """

# sample accessing
print(mydata2c[0]['tval'],mydata2c[0]['density'])
# single element
result=[ (myeach['tval'],myeach['density']) for myeach in mydata2c]
# pull out pairs e.g. 'tval, density')
# two-line method for extracting any variable set from this
pairsets= [ (myeach['tval'],myeach['density'],myeach['vgse']) for myeach in mydata2c]
# same, pull out pairs, only assign e.g.
x=tval
y=densit
vgse = [ [z[i] for z in pairsets] for i in (0,1,2)]
# since 'vgse' is itself an dict of 3 values X/Y/Z, you can pull out nested items like this
pairsets= [ (myeach['tval'],myeach['density'],myeach['vgse']['X']) for myeach in mydata2c]
# the above methods are extensible to any number of variables,
# just update the (0,1,2) to reflect now many you have
```