base$_{b}ahDocumentation$

Release 0.1

Ruben Müller

Oct 25, 2021

CONTENTS

base_bah contains basic functions used in Python software from the Büro für Angewandte Hydrologie, Berlin. This package needs no other "heavy" packages, like numpy or pandas.

Contents:

CHAPTER

ONE

INSTALLING BASE_BAH

base_bah should work on Python 3.6 (or later) on Windows, Linux or OS X.

The following command will build and install base_bah:

```
python setup.py install
```

1.1 Installing binary wheels with pip

Binary wheel distributions of base_bah are hosted on *Pypi <https://bitbucket.org/BAH_Berlin/base_bah>_*.

```
pip install base_bah
```

CHAPTER

TWO

BASE_BAH REFERENCE DOCUMENTATION.

2.1 Read

2.1.1 Functions

Read ArcEGMO configuration files and output

pegel_dbf(arcegmo_ste, hyd_stat)	Read the pegel.dbf, creates a relation between fgw and gauge in a dictionary.
pegel_RegType(arcegmo_ste, hyd_stat, fgw, ...)	Read the pegel.dbf, returns the position of a FGW in the file
arcegmo_ste(path_to_root, filename[, ...])	Read the arc_egmo.ste into a dictionary.
config(control[, read_what, proj_conf])	Read other configuration files from arc_egmo.
configurationFile([folder, filex])	Read the configuration file for MOO.
path_met_hyd_data(inp)	Return the further path for MET and HYD input data.
complete_config(path_to_root, filename[, ...])	Read more or less the whole configuration.
arcegmo_qc(control, config[, results_file])	Read the simulation results in excel mode.
arcegmo_qt(arcegmo_ste, config[, model])	Read the observation time series in excel mode.
sim_obs(qc, qt, relation[, starttime, ...])	Return simulation and observation time series with the same starting and ending times.
cf_name(basefolder, filename[, spec, critical])	Find the correct file name or directory name in the folder ignoring the case.

base_bah.read.pegel_dbf

base_bah.read.**pegel_dbf**(*arcegmo_ste*, *hyd_stat*)

Read the pegel.dbf, creates a relation between fgw and gauge in a dictionary.

Parameters

arcegmo_ste [dict] the dictionary containing the arc_egmo.ste

hyd_stat [dict] the dictionary containing the HYD_STAT_DESCRIBE

Returns

.....

dict dictionary holding the relation

base_bah.read.pegel_RegType

base_bah.read.**pegel_RegType**(*arcegmo_ste*, *hyd_stat*, *fgw*, *regType*)
Read the pegel.dbf, returns the position of a FGW in the file

Parameters

arcegmo_ste [dict] the dictionary containing the arc_egmo.ste

hyd_stat [dict] the dictionary containing the HYD_STAT_DESCRIBE

fgw [str] the FGW value to search

regType [str] the regType under witch the fgw is located

Returns

int the location of the FGW

base_bah.read.arcegmo_ste

base_bah.read.**arcegmo_ste**(*path_to_root*, *filename*, *config_={}*, *threadnumber=None*, *pfadeInRoot=False*)
Read the arc_egmo.ste into a dictionary.

Parameters

dir [str] path to root path, assumes arc_egmo.ste in ./Arc_Egmo/

filename [str] name of arc_egmo.ste

config [dict] contains the configuration file from the MOO project

threadnumber [int] the current thread number –> if given, read from pfade.ste

pfadeInRoot [bool] if True, read pfade.ste from base folder if False, read PFade.ste from model folder

base_bah.read.config

base_bah.read.**config**(*control*, *read_what=None*, *proj_conf={}*)
Read other configuration files from arc_egmo.

Parameters

control [str] string with path and file name or read_arcegmo_ste dictionary

read_what: str HYDROLOGIE_DATEN to read hyd_data in ./Arc_EGMO or HYD_STAT_DESCRIBE to read hyd_data.sdf in /GIS/describe HYD_DAT_DESCRIBE to read hyd_data.sdf in /Zeit.dat/describe KALIBRIERUNG to read KALIBRIERUNGS.ste in *

Returns

dict contains the configuration file

base_bah.read.configurationFile

base_bah.read.**configurationFile**(*folder='.'*, *filex='Konfigurationsdatei.txt'*)
Read the configuration file for MOO.

base_bah.read.path_met_hyd_data

base_bah.read.**path_met_hyd_data**(*inp*)
Return the further path for MET and HYD input data.

base_bah.read.complete_config

base_bah.read.**complete_config**(*path_to_root*, *filename*, *threadnumber=None*, *config_={}*, *pfadeInRoot=False*)
Read more or less the whole configuration.

base_bah.read.arcegmo_qc

base_bah.read.**arcegmo_qc**(*control*, *config*, *results_file='fgw_mit.qc'*)
Read the simulation results in excel mode.

Parameters

control: dict the dictionary containing the arc_egmo.ste

results_file [string] override for fgw_mit.qc

timestep [str] override for time step in control

Returns

dict the dictionary contains the time series of all FGW in the results file and the date times

base_bah.read.arcegmo_qt

base_bah.read.**arcegmo_qt**(*arcegmo_ste*, *config*, *model=None*)
Read the observation time series in excel mode.

Parameters

arcegmo_ste [dict] the dictionary containing the arc_egmo.ste

arcegmo_ste [dict] the dictionary containing the hyd_data pointing to the location of the observation files

relation [dict] dictionary with the relation FGW-gauge

timestep [Stunde] override for timestep in control

model [str] the name of the model (multi-model mode)

Returns

dict the dictionary contains the time series of all FGW in the results file and the date times

base_bah.read.sim_obs

base_bah.read.**sim_obs**(*qc*, *qt*, *relation*, *starttime=''*, *endtime=''*, *calc_what=4280*, *extended=False*)
Return simulation and observation time series with the same starting and ending times.

If available the returned time series range from starttime to endtime. otherwise the returned time series covers the time period between starttime to endtime where both have values.

Parameters

qc [dict] dictionary with the simulated time series

qt [dict] dictionary with the observed time series

relation [dict] relates the FGWID to the gauge names

starttime [datetime] datetime object with the wanted starting time

endtime [datetime] datetime object with the wanted ending time

extended [bool, optional] if TRUE, additional variables are returned.

Returns

list in case of extended == False: np arrays with (1) simulation values, (2) observation values and (3) simulation times, (4) observation times in case of extended == True: additional the indizes of the beginning and ending time steps for simulation and observation are returned

base_bah.read.cf_name

base_bah.read.**cf_name**(*basefolder*, *filename*, *spec='generic'*, *critical=False*) → str
Find the correct file name or directory name in the folder ignoring the case.

Parameters

basefolder [str] the path and foldername in which the arc_egmo folder resides (the model)

ae_filename [str] the name of the arc_egmo.ste

spec [str, optional] generic... find directory or file; dir... find directory only; file... find file only

2.1.2 Classes

Parse the ArcEGMO configuration tree

Ae_config2(basefolder, ae_filename[, guide, ...])	Class that provides functionality to read the ArcEGMO confilguration file hierarchy.

base_bah.read.Ae_config2

class base_bah.read.**Ae_config2**(*basefolder*, *ae_filename*, *guide=None*, *barebone=False*)
Class that provides functionality to read the ArcEGMO confilguration file hierarchy.

Parameters

basefolder [str] the path and foldername in which the arc_egmo folder resides (the model folder)

ae_filename [str] the name of the arc_egmo.ste

guide [dict, optional] describes the hierarchy and configuration files. If argument is not provided, the file ae_file_guide.json will be read.

barebone [bool] read only the minimum information for the dummy mode

__init__(*basefolder*, *ae_filename*, *guide=None*, *barebone=False*)

Methods

__init__(basefolder, ae_filename[, guide, ...])	
check_for_pfade()	Read the Pfade.ste if the Argument for PROJEKT is datei.
first_entry(key1, key2)	
get_ppath(read_what)	Return the top level directory for a ArcEGMO keyword, given in the guide.
gwp_dbf(level[, critical, file_overwrite])	Read the pegel.dbf, create a relation between fgw and gauge in a dictionary.
pegel_dbf(level[, critical, file_overwrite])	Read the pegel.dbf, create a relation between fgw and gauge in a dictionary.
read([read_what, level, critical])	Read the configuration for a keyword.
read_all(level[, critical])	Read all the configuration files recusively until a certain level.
read_core(cfile[, guide])	Core function to read to configuration file with the structure.
read_file(cfile, key, guidefile)	Read a certain ASCII configuration file with the structure.

2.2 Write

save_res_beo(obj, name)	saves the results or simulation dictionaries
load_res_beo(name)	loads the results or simulation dictionaries
configure_configfile(arcegmo_ste[, keyword, ...])	Configures arc_EGMO configuration files with new parameter values, parameters are given for specific sections in the file
y_TX_file(threadnumber, obj_fun)	writes the y_TX.txt file containing the objective function values and backup filename
write_logfile(threadnumber, obj_fun, path, ...)	writes the log file for each thread

continues on next page

Table 4 – continued from previous page

overwrite_ae(arcegmo, config_, model)	ov3rride an arc_egmo.ste file to set another BERECHNUNGS_VARIANTE
arc_egmo_conf(compare, path_and_file[, ...])	writes a dictionary back into a configuration file, uses a template of the file

2.2.1 base_bah.write.save_res_beo

base_bah.write.**save_res_beo**(*obj*, *name*)
saves the results or simulation dictionaries

Parameters

obj [dict] the dictionary containing the results or observations

name [string] path and filename of the binary save file

2.2.2 base_bah.write.load_res_beo

base_bah.write.**load_res_beo**(*name*)
loads the results or simulation dictionaries

Parameters

name [string] path and filename of the binary save file

2.2.3 base_bah.write.configure_configfile

base_bah.write.**configure_configfile**(*arcegmo_ste*, *keyword='MODULSTEUERUNG'*, *compare={'Q_ELS': {'RUECKGANGSFAKTOR': 1.111}}*, *tmp_ending='tmp'*, *ste_ending='ste'*)
Configures arc_EGMO configuration files with new parameter values, parameters are given for specific sections in the file

Parameters

arcegmo_ste [dict] the dictionary containing the arc_egmo.ste

keyword [string] the keyword in arcegmo_ste that gets the name of the configuration file. To modify arc_egmo.ste (any name) start the keyword with '__' and add the name arc_egmo.ste template without ending

compare [dict] dictionary with the sections keywords in the first level and the parameter keywords in the second level a typical example is

```
>>> compare={'Q_ELS' : {'RUECKGANGSFAKTOR': (1.111),
>>>                     'RUECKGANGSExponent': (999)},
>>>          'Q_KalMil': {'RUECKGANGSFAKTOR':(2.222),
>>>                       'ModellTyp': (9999)},
>>>          'ABFLUSSKOMPONENTEN': {'RG': (1), 'RH':(2,3), 'RN':(4)},
>>>          '+2.Schicht': {'RG': (5,6), 'RH':(6.1,7), 'RN':(8,9)}}
```

tmp_ending [string, optional] the file extension of the template file

ste_ending [string, optional] the file extension of the configuration file to write the new values into

2.2.4 base_bah.write.y_TX_file

base_bah.write.**y_TX_file**(*threadnumber*, *obj_fun*)
writes the y_TX.txt file containing the objective function values and backup filename

2.2.5 base_bah.write.write_logfile

base_bah.write.**write_logfile**(*threadnumber*, *obj_fun*, *path*, *parameters*, *name_mod=False*)
writes the log file for each thread

2.2.6 base_bah.write.overwrite_ae

base_bah.write.**overwrite_ae**(*arcegmo*, *config_*, *model*)
ov3rride an arc_egmo.ste file to set another BERECHNUNGS_VARIANTE

2.2.7 base_bah.write.arc_egmo_conf

base_bah.write.**arc_egmo_conf**(*compare*, *path_and_file*, *tmp_ending='tmp'*, *ste_ending='ste'*)
writes a dictionary back into a configuration file, uses a template of the file

Parameters

compare [dict] the dictionary with the keywords and values to change

path_and_file [str] path and filename to read and write to, without endings

tmp_endin g: str the file extension of the template file

ste_ending [str] the file extension of the configuration file to write the new values into

2.3 Format

Functions to format strings.

arcEGMOheader(selection_idx[, date])	compile an time header for ArcEGMO and format string
len2(variable)	assures that month or day are returned as %mm and are string
make_shape_two(item[, date, length, filler])	returns a string with a given length.
arcegmoDateFromDatetime(dtobj[, offset])	generates an arcEGMO time entry for hourly data.
update_progress(progress)	prints a nice status bar and updates it, if called again.
add_quotes_to_datecolumn(path, file)	Add back the quotes around the date columns like "01.01.2018 00:15" to ArcEGMO files.
strip_path(string)	strip leading and tailing slash and backslash from path-segment
get_ending_number(string, where)	returns a number from the beginning or ending of a string

2.3.1 base_bah.format.arcEGMOheader

base_bah.format.**arcEGMOheader**(*selection_idx*, *date='Termin'*)
compile an time header for ArcEGMO and format string

Parameters

selection_idx [array] array with the idx of the catchment box

date [str] the column name for the date

2.3.2 base_bah.format.len2

base_bah.format.**len2**(*variable*)
assures that month or day are returned as %mm and are string

2.3.3 base_bah.format.make_shape_two

base_bah.format.**make_shape_two**(*item*, *date=None*, *length=2*, *filler='0'*)
returns a string with a given length. padding with 0. – if item is a datetime-object, then date must be given. – if item is a int or str, then date is None

Parameters

item [datetime-object or str / int] the str to be padded

date [string] unit (month, year, day, minute, second)

length [int] length of output string

2.3.4 base_bah.format.arcegmoDateFromDatetime

base_bah.format.**arcegmoDateFromDatetime**(*dtobj*, *offset=None*)
generates an arcEGMO time entry for hourly data.

Parameters

dtobj [datetime.datetime] the time stemp

offset [int, optional] an additional offset for the timestep

2.3.5 base_bah.format.update_progress

base_bah.format.**update_progress**(*progress*)
prints a nice status bar and updates it, if called again.

2.3.6 base_bah.format.add_quotes_to_datecolumn

base_bah.format.**add_quotes_to_datecolumn**(*path*, *file*)

Add back the quotes around the date columns like "01.01.2018 00:15" to ArcEGMO files. The function creates another output file with an added "2" to the file-ending in the same folder as **path**.

2.3.7 base_bah.format.strip_path

base_bah.format.**strip_path**(*string*)

strip leading and tailing slash and backslash from pathsegment

Parameters

string [string] path segment

2.3.8 base_bah.format.get_ending_number

base_bah.format.**get_ending_number**(*string*, *where*)

returns a number from the beginning or ending of a string

Parameters

string: str the string to look at

where: str use 'last' to search the tail and 'first' for the beginning

Returns

int

2.4 FTP

Class to handle downloads from a FTP-Server

MyFTP(FZ_serverD)	Class to download from FTP servers.

2.4.1 base_bah.ftp.MyFTP

class base_bah.ftp.**MyFTP**(*FZ_serverD*)

Class to download from FTP servers.

Example for an dictionary for initialization:

```
>>> FZ_serverD = {
>>>                'servername': "ftp-outgoing2.dwd.de"
>>>               , 'username': 'xxx'
>>>               , 'password': 'xxx'
>>>               , 'directory_ftp': ['gds/radar_products/fz/']
>>>               , 'directory_local': ['/media/sf_Q_DRIVE/RADVOR__/']
>>>              }
```

__init__(*FZ_serverD*)

Methods

__init__(FZ_serverD)	

check_connection()	check if we are still connected, if not reconnect
disconnect()	disconnect from the FTP server
get_binary(filename, local_file[, ...])	get binary file from FTP server
get_by_line(filename, local_file[, mode, ...])	get text file from FTP server
get_foldercontents()	return a list of files in the current directory of the FTP server
get_nlst()	return str with files in the current directory of the FTP server
login()	handles the login
set_ftp_directory(directory_ftp)	change the directory on the FTP server
set_local_directory(directory_local)	'set the local directory to save data into
set_overwrite_mode(mode)	to overwrite local files set mode True

2.5 Download

Functions for downloading from the interwebs.

folder_ACC([local_folder, server, ...])	Download all files found in the given FTP-folder into a folder with a foldername derived from the ACC file in DWD forecast folder
folder([local_folder, server, ...])	Download all files found in the given FTP-folder
link(urlLink, filename[, timeout])	Download a file given the link.
https(urlLink[, filename, timeout, headers])	downloads a link to a file from a https site..
get_station_filename2(fo_list, station, mode)	Create the file name for a DWD weather station with 2 parameter.
get_url_content(url[, log_dict, parameter, ...])	get the folder contents --> get html file

2.5.1 base_bah.download.folder_ACC

base_bah.download.**folder_ACC**(*local_folder='C:/test/'*, *server='ftp.dwd.de'*, *directory_ftp='/pub/CDC/grids_germany/hourly/radolan/historical/bin/'*, *user='anon'*, *pswrd='none'*, *date_loc=13*, *check_file=True*)

Download all files found in the given FTP-folder into a folder with a foldername derived from the ACC file in DWD forecast folder

Parameters

local_folder [string] folder in which the downloaded files are placed

server [string] Name of server, without ftp:// pard

directory_ftp [string] Path to folder with content

date_loc [int] location of the starting date of forecast in the ACC.temp used to create folder

2.5.2 base_bah.download.folder

base_bah.download.**folder**(*local_folder='C:/test/'*, *server='ftp-cdc.dwd.de'*, *directory_ftp='/pub/CDC/grids_germany/hourly/radolan/historical/bin/'*, *user='anon'*, *pswrd='none'*, *check_file=True*)

Download all files found in the given FTP-folder

Parameters

local_folder [string] folder in which the downloaded files are placed

server [string] Name of server, without ftp:// pard

directory_ftp [string] Path to folder with content

2.5.3 base_bah.download.link

base_bah.download.**link**(*urlLink*, *filename*, *timeout=300*)

Download a file given the link.

Parameters

urlLink [str] the link

filename [str] path and filename for the save file

2.5.4 base_bah.download.https

base_bah.download.**https**(*urlLink*, *filename=False*, *timeout=300*, *headers={}*)

downloads a link to a file from a https site..

Parameters

urlLink [str] the link

filename [str] path and filename for the save file

2.5.5 base_bah.download.get_station_filename2

base_bah.download.**get_station_filename2**(*fo_list*, *station*, *mode*, *recent=True*, *parameter=''*, *parameter2=None*, *datef='%Y%m%d'*)

Create the file name for a DWD weather station with 2 parameter.

Parameters

fo_list [dict or list] the output of readlines from the https overview file for a meteorological parameter or the same as stored undter the parameter as a key in the dict

station [str or int] the station id number

recent [bool] if true return the file name for recent measurements, for historical otherwise

parameter [str] the meteorological parameter to read

parameter2 [str] the meteorological parameter to read

Returns

filename as str

2.5.6 base_bah.download.get_url_content

base_bah.download.**get_url_content**(*url*, *log_dict=None*, *parameter=None*, *recent=True*, *overwrite=False*)
get the folder contents –> get html file

Parameters

url [str] the directory on the https server with the directory overview file

log_dict [dict] holds the read directory overview files. parameter is keyword, or if not provided the url

parameter [str] the meteorological parameter and also the keyword for the dictionary log_dict

parameter [str] the meteorological parameter to read

Returns

dict with parameter as key and recent and historical sub-dicts

2.6 Output

Functions to handle command line outputs

update_progress(progress)	prints a nice status bar and updates it, if called again.

2.6.1 base_bah.output.update_progress

base_bah.output.**update_progress**(*progress*)
prints a nice status bar and updates it, if called again.

2.7 PKL

Functions to create and read (gzip compressed) pickle files

save_obj_compressed(name, obj[, ending])	Saves a struct or list into a prickle binary file.
load_obj_compressed(name[, ending])	Loads a prickle binary file.
save_obj(name, obj[, ending])	Save a struct or list into a prickle binary file.
load_obj(name[, ending])	Load a prickle binary file.

2.7.1 base_bah.pkl.save_obj_compressed

base_bah.pkl.**save_obj_compressed**(*name*, *obj*, *ending='pklz'*)
Saves a struct or list into a prickle binary file.

Parameters

name [string] path and filename for prickle file without file ending, functions add .pkl

obj [struct, list,...] object to save in prickle file

2.7.2 base_bah.pkl.load_obj_compressed

base_bah.pkl.**load_obj_compressed**(*name*, *ending='pklz'*)
Loads a prickle binary file.

Parameters

name [string] path and filename for prickle file

2.7.3 base_bah.pkl.save_obj

base_bah.pkl.**save_obj**(*name*, *obj*, *ending='pkl'*)
Save a struct or list into a prickle binary file.

Parameters

name [string] path and filename for prickle file without file ending, functions add .pkl

obj [struct, list,...] object to save in prickle file

2.7.4 base_bah.pkl.load_obj

base_bah.pkl.**load_obj**(*name*, *ending='pkl'*)
Load a prickle binary file.

Parameters

name [string] path and filename for prickle file

2.8 Utilities

2.8.1 Functions

Random stuff

modify_and_import(module_name, package, ...)	Load and modify the source of a module.

base_bah.util.modify_and_import

base_bah.util.**modify_and_import**(*module_name*, *package*, *modification_func*)
Load and modify the source of a module.

2.9 Classification

Classification of dates

month_conv(month)	Return the month as int, for names or abbreviations.
ae_timestr_from_sec(sec)	Look up the time scale.
pandas_periode_from_sec(sec)	Return the pandas period string for value in seconds
pandas_periode_to_sec(inp)	Return the time in seconds for a pandas period string

2.9.1 base_bah.classification.month_conv

base_bah.classification.**month_conv**(*month*)

Return the month as int, for names or abbreviations.

Parameters

month [str] the name of the month or an abbreviation thereof

2.9.2 base_bah.classification.ae_timestr_from_sec

base_bah.classification.**ae_timestr_from_sec**(*sec*)

Look up the time scale.

Parameters

sec [float] seconds

Returns

–> "Minute", "Stunde", "Tag"

2.9.3 base_bah.classification.pandas_periode_from_sec

base_bah.classification.**pandas_periode_from_sec**(*sec*)

Return the pandas period string for value in seconds

2.9.4 base_bah.classification.pandas_periode_to_sec

base_bah.classification.**pandas_periode_to_sec**(*inp*)

Return the time in seconds for a pandas period string

Other classifications

code_lookup

2.10 Date

Functions for dates

roundTime([dt, roundTo])	Round a datetime object to any time laps in seconds
add_timedelta_sweeptimes(sweep_times, lag)	Add a lagtime to a list of dates.
months_between(date1, date2)	Return the number of months between date1 and date2
last_day_of_month(date)	Return the number of days in the month
from_to(tstart, tend, tdelta[, xformat])	Return a list of timesteps from <tstart> to <tend> of length <tdelta>
json_serial(obj)	JSON serializer for objects not serializable by default json code extended from here: https://stackoverflow.com/questions/11875770/how-to-overcome-datetime-datetime-not-json-serializable-in-python
DateTimeEncoder(*[, skipkeys, ensure_ascii, ...])	Class for JSON encodings of datetimes.
time_mhd(inp)	Return the timestep as string for the given int in seconds
doy(Y[, M, D])	Given year, month, day return day of year Astronomical Algorithms, Jean Meeus, 2d ed, 1998, chap 7
ymd(Y, N)	Given year = Y and day of year = N, return year, month, day Astronomical Algorithms, Jean Meeus, 2d ed, 1998, chap 7
woy(time_curr[, year])	Given a datetime or a list with doy and year, lookup the week of the year
full_steps_timediff(d1, d2)	Calculate the timesteps in weeks, months and days between 2 dates.
calc_period(ss, se)	Return two list with months (seasons) such that ss is the beginning of the first season and se the end of the first season.
is_leap_year(year)	If year is a leap year return True else return False

2.10.1 base_bah.date.roundTime

base_bah.date.**roundTime**(*dt=None*, *roundTo=60*)

Round a datetime object to any time laps in seconds

Parameters

dt [datetime.datetime, default now.] the date to round

roundTo [int] Closest number of seconds to round to, default 1 minute.

Author: Thierry Husson 2012 - Use it as you want but don't blame me.

2.10.2 base_bah.date.add_timedelta_sweeptimes

base_bah.date.**add_timedelta_sweeptimes**(*sweep_times*, *lag*)
Add a lagtime to a list of dates.

Parameters

sweep_times: list array of dates (sweeptimes)

lag [int of datetime.timedelta] the lag in minutes or any timedelta

2.10.3 base_bah.date.months_between

base_bah.date.**months_between**(*date1*, *date2*)
Return the number of months between date1 and date2

Parameters

date1 [datetime-object] date at the beginning of period

date2 [datetime-object] date at the end of period

2.10.4 base_bah.date.last_day_of_month

base_bah.date.**last_day_of_month**(*date*)
Return the number of days in the month

Parameters

date1 [datetime-object] includes the month to check

2.10.5 base_bah.date.from_to

base_bah.date.**from_to**(*tstart*, *tend*, *tdelta*, *xformat='%Y-%m-%d %H:%M:%S'*)
Return a list of timesteps from <tstart> to <tend> of length <tdelta>

Parameters

tstart [datetime isostring (%Y%m%d %H:%M:%S), e.g. 2000-01-01 15:34:12] or datetime object

tend [datetime isostring (%Y%m%d %H:%M:%S), e.g. 2000-01-01 15:34:12] or datetime object

tdelta [integer representing time interval in SECONDS]

Returns

output [list of datetime.datetime objects]

2.10.6 base_bah.date.json_serial

base_bah.date.**json_serial**(*obj*)

JSON serializer for objects not serializable by default json code extended from here: https://stackoverflow.com/questions/11875770/how-to-overcome-datetime-datetime-not-json-serializable-in-python

2.10.7 base_bah.date.DateTimeEncoder

class base_bah.date.**DateTimeEncoder**(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Class for JSON encodings of datetimes.

__init__(**, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, sort_keys=False, indent=None, separators=None, default=None*)

Constructor for JSONEncoder, with sensible defaults.

If skipkeys is false, then it is a TypeError to attempt encoding of keys that are not str, int, float or None. If skipkeys is True, such items are simply skipped.

If ensure_ascii is true, the output is guaranteed to be str objects with all incoming non-ASCII characters escaped. If ensure_ascii is false, the output can contain non-ASCII characters.

If check_circular is true, then lists, dicts, and custom encoded objects will be checked for circular references during encoding to prevent an infinite recursion (which would cause an OverflowError). Otherwise, no such check takes place.

If allow_nan is true, then NaN, Infinity, and -Infinity will be encoded as such. This behavior is not JSON specification compliant, but is consistent with most JavaScript based encoders and decoders. Otherwise, it will be a ValueError to encode such floats.

If sort_keys is true, then the output of dictionaries will be sorted by key; this is useful for regression tests to ensure that JSON serializations can be compared on a day-to-day basis.

If indent is a non-negative integer, then JSON array elements and object members will be pretty-printed with that indent level. An indent level of 0 will only insert newlines. None is the most compact representation.

If specified, separators should be an (item_separator, key_separator) tuple. The default is (', ', ': ') if *indent* is `None` and (',', ': ') otherwise. To get the most compact JSON representation, you should specify (',', ':') to eliminate whitespace.

If specified, default is a function that gets called for objects that can't otherwise be serialized. It should return a JSON encodable version of the object or raise a `TypeError`.

Methods

__init__(*[, skipkeys, ensure_ascii, ...])	Constructor for JSONEncoder, with sensible defaults.
default(o)	Implement this method in a subclass such that it returns a serializable object for `o`, or calls the base implementation (to raise a `TypeError`).
encode(o)	Return a JSON string representation of a Python data structure.
iterencode(o[, _one_shot])	Encode the given object and yield each string representation as available.

Attributes

item_separator
key_separator

2.10.8 base_bah.date.time_mhd

base_bah.date.**time_mhd**(*inp*)

Return the timestep as string for the given int in seconds

Parameters

inp [dict, float] dict must have the key 'ZEITSCHRITTWEITE' (ArcEGMO configuration)

2.10.9 base_bah.date.doy

base_bah.date.**doy**(*Y*, *M=None*, *D=None*)

Given year, month, day return day of year Astronomical Algorithms, Jean Meeus, 2d ed, 1998, chap 7

2.10.10 base_bah.date.ymd

base_bah.date.**ymd**(*Y*, *N*)

Given year = Y and day of year = N, return year, month, day Astronomical Algorithms, Jean Meeus, 2d ed, 1998, chap 7

2.10.11 base_bah.date.woy

base_bah.date.**woy**(*time_curr*, *year=None*)

Given a datetime or a list with doy and year, lookup the week of the year

2.10.12 base_bah.date.full_steps_timediff

base_bah.date.**full_steps_timediff**(*d1*, *d2*)

Calculate the timesteps in weeks, months and days between 2 dates.

Parameters

d1 [datetime or iterable] the starting date as datetime or as [doy, year]

d2 [datetime or iterable] the ending date as datetime or as [doy, year]

Returns

dict [dict] the number of timesteps for month (key 12), week (key 7) and days (key 1)

2.10.13 base_bah.date.calc_period

base_bah.date.**calc_period**(*ss*, *se*)

Return two list with months (seasons) such that ss is the beginning of the first season and se the end of the first season. the second season includes all other months. ss and se are int {1,2,...,12}

Parameters

ss [int] the start of season1

se [int] the end of season2

2.10.14 base_bah.date.is_leap_year

base_bah.date.**is_leap_year**(*year*)

If year is a leap year return True else return False

CHAPTER

THREE

LICENSE

3.1 Software

base_bah is licensed under the GNU General Public License, Version 3.0 or later.

Copyright (C) 2018-2020 Ruben Müller, Büro für Angewandte Hydrologie, Berlin

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston MA 02110-1301 USA.

3.2 Documentation

This documentation is licensed under the GNU Free Documentation License, Version 1.3 or later.

Copyright (C) 2018-2020 Ruben Müller, Büro für Angewandte Hydrologie, Berlin

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

INDEX

Symbols

A

C

D

F

G

H

I

J

L

M

O

P

R

S

T

U

W

Y