ODP Python SDK

Release 0.3.9

Jan 26, 2022

Contents

1 Installation **3**

2 Contents **5**
2.1 Quickstart 5
2.2 API 6
2.3 Utilities 6

Index **11**

- *Installation*
- *Contents*

CHAPTER 1

Installation

To install this package:

```
$ pip install odp_sdk
```

To upgrade this package:

```
$ pip install -U odp_sdk
```

Note: Utility functions available in CastFunctions.py and DataStatsFunctions.py are not included in the pip install package and has to be downloaded separately

CHAPTER 2

Contents

2.1 Quickstart

2.1.1 Authenticate

In order to use the ODP SDK, you need to authenticate using your provided API-key. This is achieved by setting the *api_key*-argument when instantiating *ODPClient*:

```
from odp_sdk import ODPClient
client = ODPClient(api_key="<my-api-key>")
```

You can also set the *COGNITE_API_KEY* environment variable:

```
$ export COGNITE_API_KEY=<my-api-key>
```

2.1.2 Download Ocean Data

Downloading ocean data is very easy once you have instantiated the *ODPClient*. The data is then returned as a Pandas DataFrame

```
df = client.casts(longitude=[-25, 35], latitude=[50, 80], timespan=["2018-06-01",
↪"2018-06-30"])
```

It is also possible to specify what parameters to download:

```
df = client.casts(
    longitude = [-25, 35],
    latitude = [50, 80],
    timespan = ["2018-06-01", "2018-06-30"],
    parameters = ["date", "lon", "lat", "z", "Temperature", "Salinity"
)
```

In some instances, some filtering is necessary before downloading the data. This is achieved by first listing the available casts:

```
casts = client.get_available_casts(
    longitude = [-25, 35],
    latitude = [50, 80],
    timespan = ["2018-06-01", "2018-06-30"],
    metadata_parameters = ["extId", "date", "time", "lat", "lon", "country", "Platform
↪", "dataset_code"
)
```

Then apply any desirable filters before downloading the data:

```
casts_norway = casts[casts.country == "NORWAY"]
df = client.download_data_from_casts(casts_norway.extId.tolist(),
                                     parameters=["date", "lat", "lon", "z",
↪"Temperature", "Salinity")
```

You can also download the cast metadata:

```
df = client.get_metadata(casts_norway.extId.tolist())
```

2.2 API

2.2.1 ODPClient

2.3 Utilities

2.3.1 Advanced Helper Functions

Interpolate Casts to Z

UtilityFunctions.**interpolate_casts_to_z**(*variable*, *z_int*, *max_z_extrapolation=3*, *max_z_copy_single_value=1*, *kind='linear'*)

Interpolate profiles in dataframe to prescribed depth level.

Takes a complete dataframe from ODP and interpolates each cast by filtering out the values from each unique cast

Parameters

- **df** – Pandas DataFrame fromODP
- **variable** – Variable name to be interpolated as in the dataframe (Temperature, Oxygen, etc)
- **z_int** – List of the desired depth intervals to return, i.e [0,10,20]
- **max_z_extrapolation** – The maximum length to allow extrapolating. Nan values outside this distance.
- **max_z_copy_single_value** – If only one row is present in the cast, this is the maximum distance between the point and the interpolation level for copying the value
- **kind** – Type of interpolation as in interpolate_profile

Returns DataFrame of parameter values at prescribed depth levels.

Interpolate Casts to grid

UtilityFunctions.**interpolate_to_grid**(*values*, *int_points*, *interp_type='linear'*, *minimum_neighbors=3*, *gamma=0.25*, *kappa_star=5.052*, *search_radius=0.1*, *rbf_func='linear'*, *rbf_smooth=0.001*, *rescale=True*)

Interpolate unstructured ND data to a Nd grid

Powered by the metpy library

Parameters

- **points** – (N,D) array of points, typically latitude and longitude
- **values** – (N,1) array of corresponding values, i.e Temperature, Oxygen etc
- **int_points** – list of arrays for gridding i.e lat/long grid –> (np.linspace(-25,35,60*10+1),np.linspace(50,80,30*10+1))
- **interp_type** – What type of interpolation to use. Available options include: 1) "linear", "nearest", "cubic", or "rbf" from *scipy.interpolate*. 2) "natural_neighbor", "barnes", or "cressman" from *metpy.interpolate*. Default "linear".
- **minimum_neighbors** – Minimum number of neighbors needed to perform barnes or cressman interpolation for a point. Default is 3.
- **gamma** – Adjustable smoothing parameter for the barnes interpolation. Default 0.25.
- **kappa_star** – Response parameter for barnes interpolation, specified nondimensionally in terms of the Nyquist. Default 5.052
- **search_radius** – A search radius to use for the barnes and cressman interpolation schemes. If search_radius is not specified, it will default to the average spacing of observations.
- **rbf_func** – Specifies which function to use for Rbf interpolation. Options include: 'multiquadric', 'inverse', 'gaussian', 'linear', 'cubic', 'quintic', and 'thin_plate'. Defualt 'linear'. See *scipy.interpolate.Rbf* for more information.
- **rbf_smooth** – Smoothing value applied to rbf interpolation. Higher values result in more smoothing.
- **rescale** –

Returns Array representing the interpolated values for each input point

Return type values_interpolated

Interpolate profile

UtilityFunctions.**interpolate_profile**(*z_int*, *max_z_extrapolation=10*, *max_z_copy_single_value=1*, *kind='linear'*)

Interpolate profile zv (depth, parameter) to a user defined depth.

Parameters

- **zv** – 2-D array of depth and a parameter (temperature, oxygen, . . .)
- **z_int** – 1-D array of depth levles to interpolate to

- **max_z_extrapolation** – Maximum distance to extrapolate outside profile. Use 0 for no extrapolation.
- **max_z_copy_single_value** – Maximum distance for copying the value of a single value profile.
- **kind** – Specifies the kind of interpolation as a string ('linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 'previous', 'next', where 'zero', 'slinear')

Returns Returns array of interpolated values

Example:

```
zv=array(
   [[ 0.        , 21.64599991],
   [ 9.93530941, 21.54500008],
   [19.87013626, 20.96299934],
   [20.40699959, 29.80448341],
   [19.36800003, 49.67173004],
   [18.8010006 , 74.50308228],
   [18.27400017, 99.3314209 ]]
)

z_int = [0,0,25,50,75,100,125]

v_int = interpolate_profile(ZV,z_int)

print(v_int)
# >>> array([21.64599991, 20.67589412, 19.36050431, 18.79045314, 18.25980907,␣
↪nan])
```

Plot Casts

UtilityFunctions.**plot_casts**(*df, longitude, latitude, cmap='viridis', vrange=[None, None]*)

Plot casts :param variable: str of oceanographic vairable, i.e. 'Temperature' :param df: Pandas DataFrame from ODP with lat, lon, and variable columns :param longitude: List of min and max longitude, i.e [-10,35] :param latitude: List of min and max latitude, i.e [50,80] :param cmap: colormap specification :param vrange: Ranges for variables to be showsn, i.e. [0,20]

Returns Map with variable measurments plotted as points

Plot Grid

UtilityFunctions.**plot_grid**(*latitude, int_lon, int_lat, g, cmap='viridis', vrange=[None, None], crs_latlon=<sphinx.ext.autodoc.importer._MockObject object>, variable_name=''*)

Plot Grid :param int_lon: (M,N) array of longitude grid :param int_lat: (M,N) array of latitude grid :param g: (M,N) grid to be shown :param cmap: colormap :param vrange: Ranges for grid to be shown i.e [0,35] :param crs_latlon: :param variable_name:

Returns Map with interpolated values

Get Units

UtilityFunctions.**get_units**()

Get dict describing the units of the different columns

Returns Dict of units

Plot percentage of nulls for each variable in variable list

UtilityFunctions.**plot_nulls**(*var_list=None*)

Plot percentage of nulls for each variable in variable list.

Takes a dataframe from ODP and a list of variables and plots the percentage of missing values

Parameters

- **df** – Pandas dataframe from ODP
- **var_list** – list of variables (column names) that user is interested in default list is all the columns

Returns Plot of percentage of values missing at each measuremtn (lat, lon, depth)

Plot metadata-statistics

UtilityFunctions.**plot_meta_stats**(*variable*)

Get bar graph of percentage of data belonging to a specific variable subset in the metadata

Parameters

- **df** – Pandas DataFrame with *extId*-column
- **variable** – Variable in subset of metadata

Returns Bar graph with percentage of data belonging to variable subset (i.e. data belonging to different modes of data collection ('dataset'))

Plot distribution of values

UtilityFunctions.**plot_distributions**(*var_list*)

Plot the distributions of the values for a list of variables

Parameters

- **df** – Pandas DataFrame from ODP containing oceanographic variables and values
- **var_list** – list of variables (column names) that should be plotted

Returns Plots of distributions of values for each variable in variable list

Plot casts belonging to specific dataset

UtilityFunctions.**plot_datasets**(*variable*, *latitude*, *longitude*)

Plots on a map casts belonging to specific dataset (mode of data collection, i.e. ctd, xbt)

Parameters

- **df** – Pandas DataFrame
- **variable** – Variable of choice
- **latitude** – Bounding box latitude
- **longitude** – Bounding box longitude

Returns Map with color coded casts based on dataset_code

Internal Helper Functions

UtilityFunctions.**geo_map**()
Helper function for mapping :param ax: Matplotlib axis

UtilityFunctions.**missing_values**(*var_list*)
Get dataframe of nulls for each variable in variable list.

Takes a dataframe from ODP and a list of variables and return dataframe of missing values

Parameters

- **df** – Pandas DataFrame from ODP
- **var_list** – list of variables (column names) that user is interested in default list is all the columns

Returns Dataframe percentage of values missing at each measuremtn (lat, lon, depth)

2.3.2 Geographic Utilities

Convert Latitude and Longitude to Geo-Index

Convert Latitude and Longitude to grid-coordinates

Convert Geo-Index to grid-coordinates

Convert Geo-Index to Latitude and Longitude

Get all grid-coordinates within a rectangle

Get all Geo-Indices within a rectangle

Index

G

geo_map() (*Examples.UtilityFunctions method*), 10
get_units() (*Examples.UtilityFunctions method*), 8

I

interpolate_casts_to_z() (*Examples.UtilityFunctions method*), 6
interpolate_profile() (*Examples.UtilityFunctions method*), 7
interpolate_to_grid() (*Examples.UtilityFunctions method*), 7

M

missing_values() (*Examples.UtilityFunctions method*), 10

P

plot_casts() (*Examples.UtilityFunctions method*), 8
plot_datasets() (*Examples.UtilityFunctions method*), 9
plot_distributions() (*Examples.UtilityFunctions method*), 9
plot_grid() (*Examples.UtilityFunctions method*), 8
plot_meta_stats() (*Examples.UtilityFunctions method*), 9
plot_nulls() (*Examples.UtilityFunctions method*), 9