
PostCodesMaps

Release 1.0.0

Mateusz Gomulski

Apr 19, 2023

CONTENTS:

1	pcm_db.py	3
2	pcm_parser.py	5
3	pcm_utilities.py	7
4	Indices and tables	13
	Python Module Index	15
	Index	17

PostCodesMaps is an application written in Python that creates postcode maps of regions in Poland based on a set of address points from the Polish National Register of Boundaries Database (also known as the PRG database). As part of PostCodeMaps, a website has been created that allows visualisation of the generated postcodes maps.

PostCodesMaps creates SQL database containing all address points in Poland by parsing files in Geography Markup Language format into SQL tables. The main data source of GeocoderPL is The National Register of Boundaries Database (also known as PRG database) - state-maintained reference database of all address points in Poland (including administrative division of the country):

- <https://dane.gov.pl/pl/dataset/726,panstwowy-rejestr-granic-i-powierzchni-jednostek-podziaow-terytorialnych-kraju/resource/29538>
- <https://dane.gov.pl/pl/dataset/726,panstwowy-rejestr-granic-i-powierzchni-jednostek-podziaow-terytorialnych-kraju/resource/29515>

The resulting database was used to generate postcode maps of Poland (in .shp and .geojson formats), which were then overlaid on Google Maps maps for visualisation purposes.

PCM_DB.PY

Module that defines SQL database class in the PostCodesMaps project

class pcm_db.PRG(*kod_teryt, kod_pocztowy, szerokosc, dlugosc*)

Bases: Base

Class that defines columns of “PRG_TABLE”

__init__(*kod_teryt, kod_pocztowy, szerokosc, dlugosc*)

Method that creates objects from a class “PRG”

Parameters

- **kod_teryt** – TERYT code of the region in which the address point is located
- **kod_pocztowy** – Postcode where the address point is located
- **szerokosc** – Longitude of a given address point
- **dlugosc** – Latitude of a given address point

Returns

The method does not return any values

__repr__()

Method that represents an objects in a class “PRG” as a string

Return type

str

Returns

String that represents objects of the class “PRG”

PCM_PARSER.PY

XML Parser module of the PostCodesMaps project

class pcm_parser.**PRGDataParser**(*xml_path, tags_tuple, event_type*)

Bases: ABC

Class that parses adress points from PRG database to SQLAlchemy database

__init__(*xml_path, tags_tuple, event_type*)

Method that creates objects from a class “PRGDataParser”

Parameters

- **xml_path** (str) – Path of a given XML file
- **tags_tuple** (Tuple[str, ...]) – Tuple containig XML tags
- **event_type** (str) – Type of event in XML file

Returns

The method does not return any values

__weakref__

list of weak references to the object (if defined)

check_path()

Method that checks if path to file is valid

Return type

None

Returns

The method does not return any values

create_points_list(*xml_context*)

Method that creates list of data points

Parameters

xml_context (iterparse) – Root of XML data tree

Return type

None

Returns

The method does not return any values

create_teryt_dict()

Method that creates TERYT dictionary

Return type

None

Returns

The method does not return any values

parse_xml()

Method that parses xml file and saves data to SQL database

Return type

None

Returns

The method does not return any values

PCM_UTILITIES.PY

Module that collects variety utility functions for geospatial programming

`pcm_utilities.clear_xml_node(curr_node)`

Function that clears unnecessary XML nodes from RAM memory

Parameters

curr_node (Element) – Current XML node

Return type

None

Returns

Function does not return any values

`pcm_utilities.create_coords_transform(in_epsg, out_epsg, change_map_strateg=False)`

Function that creates object that transforms geographical coordinates

Parameters

- **in_epsg** (int) – Number of input EPSG coordinates system
- **out_epsg** (int) – Number of output EPSG coordinates system
- **change_map_strateg** (bool) – Flag indicating if map strategy should be changed

Return type

CoordinateTransformation

Returns

Coordinates transformation that transforms spatial references from input EPSG system to output EPSG system

`pcm_utilities.create_geom_dict(fin_geom_dict, teryt_arr, teryt_gmn_paths_dict, gmn_teryt_dict)`

Function that creates dictionary of all polygons in current province

Parameters

- **fin_geom_dict** (Dict[str, Dict[Any, Any]]) – Dictionary with basic parameters for a given region
- **teryt_arr** (ndarray) – Numpy array containing paths of shapefiles
- **teryt_gmn_paths_dict** (Dict[str, List[Any]]) – Dictionary with TERYT codes and border points paths of municipalities
- **gmn_teryt_dict** (Dict[str, str]) – Dictionary with TERYT codes and names of municipalities

Return type

None

Returns

Function does not return any values

`pcm_utilities.create_geom_refs_dict()`

Function that creates dictionary with TERYT codes and border points paths of municipalities

Return type

Dict[str, List[Any]]

Returns

Dictionary with TERYT codes and border points paths of municipalities

`pcm_utilities.create_logger(name)`

Function that creates logging file

Parameters

name (str) – Name of logger

Return type

Logger

Returns

Logger object

`pcm_utilities.create_pc_shps(grp_prg_pts, a_width, a_height, fin_pc_arr, shp_fold, ur_coords, teryt_code, path_num)`

Function that creates shapefiles of postcodes zones

Parameters

- **grp_prg_pts** (Dict[str, ndarray]) – Dictionary containing geographical coordinates grouped by postal code
- **a_width** (int) – Width of path bounding box
- **a_height** (int) – Height of path bounding box
- **fin_pc_arr** (ndarray) – TERYT code of the region
- **shp_fold** (str) – Path where the Shapefile will be saved
- **ur_coords** (ndarray) – Coordinates of the upper right vertex of the path bounding box
- **teryt_code** (str) – TERYT code of the region
- **path_num** (int) – Path number

Return type

None

Returns

Function does not return any values

`pcm_utilities.create_postal_codes_shps()`

Function that creates shapefiles of postal codes zones

Return type

None

Returns

Function does not return any values

`pcm_utilities.csv_to_dict(c_path, pl_names_dict)`

Function that imports CSV file and creates dictionary from first two columns of that file

Parameters

- **c_path** (str) – Path of the CSV file that should be read to dictionary
- **pl_names_dict** (Dict[str, str]) – Dictionary containing Polish names of regions

Return type

Dict[str, str]

Returns

Dictionary read from CSV file and corrected using “pl_names_dict” dictionary

`pcm_utilities.get_corr_reg_name(curr_name)`

Function that corrects wrong regions names

Parameters

curr_name (str) – Current region name

Return type

str

Returns

Corrected region name

`pcm_utilities.merge_all_shps_save(mrgd_plg_path_shp, mrgd_plg_path_geoj, pc_dict, all_pl_pc_dict, fin_schema, fin_srs, coords_prec, add_pts_nums)`

Function that merges all polygons of post codes areas and saves them to files .shp and .geojson

Parameters

- **mrgd_plg_path_shp** (str) – SHP path under which merged polygon files are to be saved
- **mrgd_plg_path_geoj** (str) – GEOJSON path under which merged polygon files are to be saved
- **pc_dict** (Dict[str, List[Any]]) – Dictionary of current polygons
- **all_pl_pc_dict** (Dict[str, Dict[Any, Any]]) – Dictionary of all polygons
- **fin_schema** (Dict[str, Any]) – Dictionary containing final parameters of SHP files
- **fin_srs** (str) – String containing the name of the final coordinate system
- **coords_prec** (str) – String containing final precision of coordinates
- **add_pts_nums** (Dict[str, Dict[Any, Any]]) – Dictionary containing the number of address points and the area of a given municipality

Return type

None

Returns

Function does not return any values

`pcm_utilities.mult_pc_zones_shps(teryt_code, paths_list, c_post_cods, prg_cols, shp_fold)`

Function that creates multiple shapefiles of postcodes zones for single municipality

Parameters

- **teryt_code** (str) – TERYT code of the region
- **paths_list** (List[Any]) – List of points representing the boundaries of the region

- **c_post_cods** (List[str]) – List of expected postcodes for giving region
- **prg_cols** (List[Any]) – List with postcode, latitude and longitude of a given address point
- **shp_fold** (str) – Path where the Shapefile will be saved

Return type

None

Returns

Function does not return any values

`pcm_utilities.prepare_merging(fin_geom_dict, wod_pow_shape, pc_dict, curr_pc)`

Function that prepares postcodes polygons for merging

Parameters

- **fin_geom_dict** (Dict[str, Dict[Any, Any]]) – Dictionary with basic parameters for a given region
- **wod_pow_shape** (Polygon) – Polygons representing surface water shapes
- **pc_dict** (Dict[str, List[Any]]) – Dictionary of current polygons
- **curr_pc** (List[int]) – List for counting address points for a given postcode

Return type

None

Returns

Function does not return any values

`pcm_utilities.read_wod_pow_shapes()`

Function that reads shapes of surface waters in Poland

Return type

Polygon

Returns

Shape of surface waters in Poland

`pcm_utilities.rmvlsl_pc(fin_pc_arr, prg_pts_ids)`

Function that removes isolated postal codes from the area of other postal codes

Parameters

- **fin_pc_arr** (ndarray) – TERYT code of the region
- **prg_pts_ids** (ndarray) – List of points representing the boundaries of the region

Return type

None

Returns

Function does not return any values

`pcm_utilities.rmvprg_outlyrs(grp_prg_pts)`

Function that removes from list of address points potentially incorrect zip codes

Parameters

grp_prg_pts (Dict[str, ndarray]) – Dictionary containing geographical coordinates grouped by postal code

Return type

Dict[str, ndarray]

Returns

Corrected dictionary of geographical coordinates grouped by postcode

`pcm_utilities.rmvsml_ovrlp_polygs(fin_geom_dict)`

Function that removes too small and overlapping polygons from dictionary “fin_geom_dict”

Parameters

fin_geom_dict (Dict[str, Dict[Any, Any]]) – Dictionary with basic parameters for a given region

Return type

None

Returns

Function does not return any values

`pcm_utilities.save_merged_shps(shp_fold, wod_pow_shape, teryt_gmn_paths_dict, fin_schema, curr_pc)`

Function that merges postcodes shapefiles by provinces and save them to the hard disk

Parameters

- **shp_fold** (str) – The path to the folder where the shapefiles will be saved
- **wod_pow_shape** (Polygon) – Polygons representing surface water shapes
- **teryt_gmn_paths_dict** (Dict[str, List[Any]]) – Dictionary with TERYT codes and boundary paths of municipalities
- **fin_schema** (Dict[str, Any]) – Dictionary containing final parameters of SHP files
- **curr_pc** (List[int]) – List for counting address points for a given postcode

Return type

None

Returns

Function does not return any values

`pcm_utilities.sngl_pc_zone_shp(teryt_code, paths_list, c_post_cods, shp_fold)`

Function that creates single shapefile of postal codes zones for single municipality

Parameters

- **teryt_code** (str) – TERYT code of the region
- **paths_list** (List[Any]) – List of points representing the boundaries of the region
- **c_post_cods** (List[str]) – List of expected postcodes for giving region
- **shp_fold** (str) – Path where the Shapefile will be saved

Return type

None

Returns

Function does not return any values

`pcm_utilities.time_decorator(func)`

Decorator that logs information about time of function execution

Parameters

func – Function call that should be wrapped

Return type

Callable

Returns

Time wrapper function call

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`pcm_db`, [3](#)

`pcm_parser`, [5](#)

`pcm_utilities`, [7](#)

Symbols

`__init__()` (*pcm_db.PRGR method*), 3
`__init__()` (*pcm_parser.PRGRDataParser method*), 5
`__repr__()` (*pcm_db.PRGR method*), 3
`__weakref__` (*pcm_parser.PRGRDataParser attribute*), 5

C

`check_path()` (*pcm_parser.PRGRDataParser method*), 5
`clear_xml_node()` (*in module pcm_utilities*), 7
`create_coords_transform()` (*in module pcm_utilities*), 7
`create_geom_dict()` (*in module pcm_utilities*), 7
`create_geom_refs_dict()` (*in module pcm_utilities*), 8
`create_logger()` (*in module pcm_utilities*), 8
`create_pc_shps()` (*in module pcm_utilities*), 8
`create_points_list()` (*pcm_parser.PRGRDataParser method*), 5
`create_postal_codes_shps()` (*in module pcm_utilities*), 8
`create_teryt_dict()` (*pcm_parser.PRGRDataParser method*), 5
`csv_to_dict()` (*in module pcm_utilities*), 8

G

`get_corr_reg_name()` (*in module pcm_utilities*), 9

M

`merge_all_shps_save()` (*in module pcm_utilities*), 9
module
 pcm_db, 3
 pcm_parser, 5
 pcm_utilities, 7
`mult_pc_zones_shps()` (*in module pcm_utilities*), 9

P

`parse_xml()` (*pcm_parser.PRGRDataParser method*), 6
pcm_db
 module, 3
pcm_parser
 module, 5

pcm_utilities
 module, 7
`prepare_merging()` (*in module pcm_utilities*), 10
PRGR (*class in pcm_db*), 3
PRGRDataParser (*class in pcm_parser*), 5

R

`read_wod_pow_shapes()` (*in module pcm_utilities*), 10
`rmv_isl_pc()` (*in module pcm_utilities*), 10
`rmv_prg_outlyrs()` (*in module pcm_utilities*), 10
`rmv_sml_ovrlp_polygs()` (*in module pcm_utilities*), 11

S

`save_merged_shps()` (*in module pcm_utilities*), 11
`sngl_pc_zone_shp()` (*in module pcm_utilities*), 11

T

`time_decorator()` (*in module pcm_utilities*), 11