

---

# **PostCodesMaps**

***Release 1.0.0***

**Mateusz Gomulski**

**Apr 17, 2023**



**CONTENTS:**

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



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 and buildings 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 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.rmv_sml_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.PRG method*), 3  
`__init__()` (*pcm\_parser.PRGDataParser method*), 5  
`__repr__()` (*pcm\_db.PRG method*), 3  
`__weakref__` (*pcm\_parser.PRGDataParser attribute*), 5

## C

`check_path()` (*pcm\_parser.PRGDataParser 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.PRGDataParser method*), 5  
`create_postal_codes_shps()` (*in module pcm\_utilities*), 8  
`create_teryt_dict()` (*pcm\_parser.PRGDataParser 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.PRGDataParser method*), 6  
*pcm\_db*  
    module, 3  
*pcm\_parser*  
    module, 5

*pcm\_utilities*  
    module, 7  
`prepare_merging()` (*in module pcm\_utilities*), 10  
*PRG* (*class in pcm\_db*), 3  
*PRGDataParser* (*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