GeocoderPL

Release 1.2.0

Mateusz Gomulski

Mar 07, 2023

CONTENTS:

GeocoderPL is an application written in Python, which can be used for geocoding address points in Poland along with the possibility to display basic information about a given address point and the building assigned to this address. GeocoderPL has a form of search engine with three map layers: OpenStreetMap, Google Maps and Stamens Map.

GeocoderPL creates SQLite database containing all address points and buildings in Poland by parsing files in Geography Markup Language format to SQL tables. The main data sources of GeocoderPL are following: The National Register of Boundaries Database (a.k.a. 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 Topographic Objects Database (a.k.a. BDOT10k database) - state maintained vector database which contains the spatial location of all topographic features in Poland:

- https://opendata.geoportal.gov.pl/bdot10k/Polska_GML.zip

The SQLite database created in this way can easily cooperate with GeocoderPL search engine - when the user type name of the city, street or postal code, query to SQLite database is sent.

Geographic coordinates of every address point from the PRG database are cross-validated by checking that they lie inside the polygon of their district. For every address point in PRG database the closest building in the BDOT10k database is found and if the distance between polygon of this building and address point is less than 10 meters then the building is assigned to adress point.

Geocoding using GeocoderPL search engine requires providing city name, street, building number or postal code of the address point for which we would like to find geographic coordinates. It is also possible to perform reverse geocoding - if you pass geographic coordinates to search engine then you will receive address point closest to these coordinates.

GeocoderPL can be also used for finding address point by providing name of public institution, church or shop - for part of builings such information is avaible in BDOT10k database, so they are also present in GeocoderPL search engine. GeocoderPL search engine does not utilize any external search engines - it relies only on data gather in SQLite database fed with Polish government data and on three map layers: OpenStreetMap, Google Maps and Stamen's Map (visualisation purpose only).

CHAPTER

ONE

DB_CLASSES.PY

Module that defines SQL database classes in the GeocoderPL project

class db_classes.**BDOT10K**(*kod_sektora*, *kat_budynku*, *nazwa_kart*, *stan_budynku*, *funkcja_budynku*, *liczba_kond*, *czy_zabytek*, *opis_budynku*, *powierzchnia*, *centr_lat*, *centr_long*, *bubd_geojson*)

Bases: `Base`

Class that defines columns of "BDOT10K_TABLE"

__init__(*kod_sektora*, *kat_budynku*, *nazwa_kart*, *stan_budynku*, *funkcja_budynku*, *liczba_kond*, *czy_zabytek*, *opis_budynku*, *powierzchnia*, *centr_lat*, *centr_long*, *bubd_geojson*)

Method that creates objects from a class "BDOT10K"

Parameters

- **kod_sektora** – Code of the sector in which the building is located
- **kat_budynku** – Type of building
- **nazwa_kart** – Cartographic name of the building
- **stan_budynku** – Condition of the building
- **funkcja_budynku** – Function of the building
- **liczba_kond** – Number of storeys in the building
- **czy_zabytek** – Flag indicating whether a building is a historic building
- **opis_budynku** – Brief description of the building
- **powierzchnia** – Building surface in square metres
- **centr_lat** – Latitude of the centroid of the building
- **centr_long** – Longitude of the centroid of the building
- **bubd_geojson** – Building outline in GEOJSON format

Returns

The method does not return any values

__repr__()

Method that represents an objects in a class "BDOT10K" as a string

Return type

`str`

Returns

String that represents objects of the class "BDOT10K"

class db_classes.**PRG**(*wojewodztwo*, *powiat*, *gmina*, *miejscowosc*, *miejscowosc2*, *ulica*, *numer*, *kod_pocztowy*, *status*, *szerokosc*, *dlugosc*, *zrodlo*, *czy_poprawny*, *odleglosc_od_gminy*, *bdot10_bubd_id*, *odleglosc_od_budynku*, *kod_sektora*, *dodatkowy_opis*)

Bases: Base

Class that defines columns of "PRG_TABLE"

__init__(*wojewodztwo*, *powiat*, *gmina*, *miejscowosc*, *miejscowosc2*, *ulica*, *numer*, *kod_pocztowy*, *status*, *szerokosc*, *dlugosc*, *zrodlo*, *czy_poprawny*, *odleglosc_od_gminy*, *bdot10_bubd_id*, *odleglosc_od_budynku*, *kod_sektora*, *dodatkowy_opis*)

Method that creates objects from a class "PRG"

Parameters

- **wojewodztwo** – Name of the province in which the address point is located
- **powiat** – Name of the county in which the address point is located
- **gmina** – Name of the municipality in which the address point is located
- **miejscowosc** – Name of the town in which the address point is located
- **miejscowosc2** – Additional name of the town where the address point is located
- **ulica** – Name of the street in which the address point is located
- **numer** – Number of the building in which the address point is located
- **kod_pocztowy** – Postcode where the address point is located
- **status** – Status of a given address point
- **szerokosc** – Longitude of a given address point
- **dlugosc** – Latitude of a given address point
- **zrodlo** – Source of data from which information on a given address point is derived
- **czy_poprawny** – Flag indicating whether an address point is correct
- **odleglosc_od_gminy** – The distance of a given point from the contour of its municipality (in meters)
- **bdot10_bubd_id** – D of the nearest building in the BDOT10k database
- **odleglosc_od_budynku** – Distance of a given address point from the nearest building
- **kod_sektora** – Code of the sector in which given point is located
- **dodatkowy_opis** – Additional description of the 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"

class db_classes.**RegJSON**(*json_name*, *json_teryt*, *json_shape*)

Bases: Base

Class that defines regional JSON files

__init__(*json_name*, *json_teryt*, *json_shape*)

Method that creates objects from a class "RegJSON"

Parameters

- **json_name** – Name of the reginal JSON file
- **json_teryt** – TERYT code of the reginal JSON file
- **json_shape** – JSON shape of region

Returns
The method does not return any values

__repr__()

Method that represents an objects in a class "RegJSON" as a string

Return type
str

Returns
String that represents objects of the class "RegJSON"

class db_classes.**TerytCodes**(*teryt_name*, *teryt_code*)

Bases: Base

Class that defines TERYT codes

__init__(*teryt_name*, *teryt_code*)

Method that creates objects from a class "TerytCodes"

Parameters

- **teryt_name** – TERYT name
- **teryt_code** – TERYT code

Returns
The method does not return any values

__repr__()

Method that represents an objects in a class "TerytCodes" as a string

Return type
str

Returns
String that represents objects of the class "TerytCodes"

class db_classes.**UniqPhrs**(*uniq_phrs*)

Bases: Base

Class that defines unique phrases

__init__(*uniq_phrs*)

Method that creates objects from a class "UniqPhrs"

Parameters
uniq_phrs – String containing unique street addresses in Poland

Returns
The method does not return any values

__repr__()

Method that represents an objects in a class "UniqPhrs" as a string

Return type
str

Returns
String that represents objects of the class "UniqPhrs"

CHAPTER

TWO

XML_PARSERS.PY

XML Parsers module of the GeocoderPL project

class xml_parsers.**BDOT10kDataParser**(*xml_path*, *tags_tuple*, *event_type*, *dicts_tags*, *tags_dict*)

Bases: *XmlParser*

BDOT10kDataParser class

__init__(*xml_path*, *tags_tuple*, *event_type*, *dicts_tags*, *tags_dict*)

Method that creates objects from a class "BDOT10kDataParser"

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
- **dicts_tags** (Dict[str, str]) – XML tags of BDOT10k dicts
- **tags_dict** (Dict[str, int]) – Tags dicts of BDOT10k buildings

Returns

The method does not return any values

check_path()

Method that checks if path to file is valid

Return type

None

Returns

The method does not return any values

parse_bdot10k_xml(*xml_contex*, *fin_row*)

Method that exctrats data from BDOT10k XML file

Parameters

- **xml_contex** (iterparse) – Root of XML data tree
- **fin_row** (List[Any]) – List containing information on a single building from the BDOT10k database

Return type

List[List[Any]]

Returns

List contsining data extracted from BDOT10k database

parse_xml()

Method that parses XML file and saves data to SQL database

Return type
None

Returns
The method does not return any values

class xml_parsers.**BDOT10kDictsParser**(*xml_path*, *tags_tuple*, *event_type*)

Bases: *XmlParser*

BDOT10kDictsParser class

__init__(*xml_path*, *tags_tuple*, *event_type*)

Method that creates objects from a class "BDOT10kDictsParser"

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

check_path()

Method that checks if path is valid

Return type
None

Returns
The method does not return any values

get_bdot10k_dicts()

Method that returns final BDOT10k dicts

Return type
Dict[Any, Any]

Returns
Method that returns final BDOT10k dicts

parse_xml()

Method that parses XML file to dictionairy object

Return type
None

Returns
The method does not return any values

class xml_parsers.**PRGDataParser**(*xml_path*, *tags_tuple*, *event_type*, *perms_dict*)

Bases: *XmlParser*

PRGDataParser class

__init__(*xml_path*, *tags_tuple*, *event_type*, *perms_dict*)

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
- **perms_dict** (Dict[int, List[int]]) – Dictionary containing superpermutation indices

Returns
The method does not return any values

check_path()

Method that checks if path to file is valid

Return type
None

Returns
The method does not return any values

check_prg_pts_add_db(*points_arr*, *woj_name*, *teryt_arr*, *json_arr*, *wrld_pl_trans*, *sekt_addr_phrs*)

Function that converts spatial reference of PRG points from 2180 to 4326, checks if given PRG point belongs to shapefile of its district and finds closest building shape for given PRG point

Parameters

- **points_arr** (ndarray) – Numpy array containing all address points in a given province
- **woj_name** (str) – Name of the province
- **teryt_arr** (ndarray) – Numpy array containing TERYT names and TERYT codes
- **json_arr** (ndarray) – Numpy array containing JSON shapefiles
- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 2180 to EPSG 4326
- **sekt_addr_phrs** (ndarray) – Numpy array containing address phrases

Return type
None

Returns
The method does not return any values

create_points_list(*xml_contex*)

Creating list of data points

Parameters
xml_contex (iterparse) – Root of XML data tree

Return type
List[List[str]]

Returns
List containing lists of address points

parse_xml()

Method that parses XML file and saves data to SQL database

Return type
None

Returns
The method does not return any values

class xml_parsers.**XmlParser**(*xml_path*, *tags_tuple*, *event_type*)

Bases: ABC

Parent XML parsers class

__init__(*xml_path*, *tags_tuple*, *event_type*)

Abstract method that creates objects from a class "XmlParser"

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)

abstract check_path()

Abstract method that checks if path is valid

Return type

None

Returns

The method does not return any values

property get_xml_path: str

Abstract method that returs path of a given XML file

Returns

Path of a given XML file

abstract parse_xml()

Abstract method that parses XML file

Return type

None

Returns

The method does not return any values

xml_parsers.**read_bdot10k_dicts**()

Function that reads to RAM BDOT10k dictionaries

Return type

Dict[str, Dict[str, ndarray]]

Returns

Dictionary containing BDOT10k dictionaries

CHAPTER

THREE

SUPER_PERMUTATIONS.PY

Class that calculates indices providing superpermutations for lists of strings with length of maximum 5 elements

class super_permutations.**SuperPerms**(*max_v*)

Bases: `object`

Class that defines superpermutation indices

__init__(*max_v*)

Method that creates objects from a class "SuperPerms"

Parameters
max_v (`int`) – Number of words in superpermutation

Returns
The method does not return any values

__weakref__

list of weak references to the object (if defined)

get_super_perm(*p_vals=None*, *c_perms=None*)

Function that finds final superpermutation (shortests list of indices)

Parameters

- **p_vals** (`Optional[str]`) – Previous permutation values
- **c_perms** (`Optional[List[str]]`) – List of current permutation

Return type
`List[int]`

Returns
Final list of indices

CHAPTER

FOUR

GEO_UTILITIES.PY

Module that collects variety utility functions for GeocoderPL project

geo_utilities.**calc_pnt_dist**(*c_paths*, *x_val*, *y_val*, *wrld_pl_trans*)

Function that calculates distances of point to given polygon

Parameters

- **c_paths** (List[Path]) – List containing matplotlib paths of regions
- **x_val** (float) – Longitude of a given address point
- **y_val** (float) – Latitude of a given address point
- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 4326 to EPSG 2180

Return type
float

Returns
Distance from a givent address point do closest polygon

geo_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
The method does not return any values

geo_utilities.**convert_coords**(*all_coords*, *in_system*, *out_system*)

Function that converts multiple coordinates between given systems

Parameters

- **all_coords** (Union[ndarray, List[List[float]]]) – Numpy array / list contining coordinates that should be transformed from one EPSG coordinates system to the other
- **in_system** (str) – Input coordinates system (EPSG string number)
- **out_system** (str) – Output coordinates system (EPSG string number)

Return type
Transformer

Returns
Transformation object that converts coordinates from one EPSG system (in_system) to the other (out_system)

geo_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

geo_utilities.**create_logger**(*name*)

Function that creates logging file

Parameters
name (str) – Name of logger

Return type
Logger

Returns
Logger object

geo_utilities.**csv_to_dict**(*c_path*)

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

Parameters
c_path (str) – Path of the CSV file that should be read to dictionary

Return type
Dict[str, str]

Returns
Dictionary read from CSV file

geo_utilities.**fill_regs_tables**()

Function that fills tables with parameters of regions shapes

Return type
None

Returns
The method does not return any values

geo_utilities.**gen_fin_bubds_ids**(*c_coords*, *c_len*, *top_geojson*, *top_ids*, *bdot10k_dist*, *bdot10k_ids*, *crds_inds*, *pow_bubd_arr*, *dod_opis_list*, *addr_phrs_list*, *addr_phrs_len*, *c_addr_phrs_uniq*, *wrld_pl_trans*)

Function that finds closest buidling shape for given PRG point

Parameters

- **c_coords** (ndarray) – Numpy array containing all address points in given sector

- **c_len** (int) – Numper of current address points
- **top_geojson** (ndarray) – Numpy array containing top "n" BDOT10k buildinigs located closest to given address point
- **top_ids** (ndarray) – Numpy array containing IDs of top "n" BDOT10k buildinigs located closest to given address point
- **bdot10k_dist** (ndarray) – Numpy arrray cointaining distance of a given address point to closest building from BDOT10k database
- **bdot10k_ids** (ndarray) – Numpy array containing IDs of buildings from BDOT10k database
- **crds_inds** (ndarray) – Numpy array containng BDOT10k buildings indices for given sector
- **pow_bubd_arr** (ndarray) – Numpy array containing information about all BDOT10k buildings in current sector
- **dod_opis_list** (ndarray) – Numpy array containing additional descriptions of an address point
- **addr_phrs_list** (List[str]) – List containing address points phrases
- **addr_phrs_len** (int) – Length of address points phrases list
- **c_addr_phrs_uniq** (str) – Current unique addresses string
- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 4326 to EPSG 2180

Return type
Tuple[str, str]

Returns

- c_adr_phr (str) - current addresses phrase
- c_addr_phrs_uniq (str) - current unique addresses string

geo_utilities.**get_bdot10k_id**(*curr_coords*, *coords_inds*, *bdot10k_ids*, *bdot10k_dist*, *dod_opis_list*, *addr_phrs_list*, *addr_phrs_len*, *wrld_pl_trans*, *addr_phrs_uniq*, *sekts_arr*, *sekts_ids*, *pow_bubd_all*, *sekt_addr_phrs*)

Function that returns id and distance of polygon closest to PRG point

Parameters

- **curr_coords** (ndarray) – Numpy array containing all address points in region
- **coords_inds** (ndarray) – Numpy array containng BDOT10k buildings indices
- **bdot10k_ids** (ndarray) – Numpy array containing IDs of buildings from BDOT10k database
- **bdot10k_dist** (ndarray) – Numpy arrray cointaining distance of a given address point to closest building from BDOT10k database
- **dod_opis_list** (ndarray) – Numpy array containing additional descriptions of an address point
- **addr_phrs_list** (List[str]) – List containing address points phrases
- **addr_phrs_len** (int) – Length of address points phrases list

- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 4326 to EPSG 2180
- **addr_phrs_uniq** (str) – Unique addresses string
- **sekts_arr** (ndarray) – Numpy array contaning sectors of address points
- **sekts_ids** (ndarray) – Numpy array containing indices of sectors
- **pow_bubd_all** (ndarray) – Numpy array containing information about all BDOT10k buildings in current region
- **sekt_addr_phrs** (ndarray) – Numpy array containing sectors of address points

Return type
str

Returns
Unique addresses string

geo_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

geo_utilities.**get_osm_coords**(*address*, *outside_pts*, *c_paths*, *popraw_list*, *c_ind*, *coord1*, *coord2*, *dists_list*, *zrodlo_list*, *wrld_pl_trans*)

Function that returns OSM coordinates of address point or distance from the district shapefile

Parameters

- **address** (str) – Address string
- **outside_pts** (ndarray) – Numpy array of address points identified as beeing outsiode of given region border
- **c_paths** (List[Path]) – List containing matplotlib paths of regions
- **popraw_list** (List[int]) – List containing flags indicating if a given address point is valid
- **c_ind** (int) – Current index of a given address point
- **coord1** (float) – Longitude of a given address point
- **coord2** (float) – Latitude of a given address point
- **dists_list** (List[float]) – List cointaining distance of a given address point to its municipility border
- **zrodlo_list** (List[str]) – List containing names of the source of a given address point
- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 4326 to EPSG 2180

Return type
None

Returns
The method does not return any values

geo_utilities.**get_region_shapes**()

Function that creates shapes for each regions

Return type
Dict[str, Geometry]

Returns
Ordered dictionary containing shapes of regions

geo_utilities.**get_sector_codes**(*poly_centr_y*, *poly_centr_x*)

Function that returns sector code for given coordinates

Parameters

- **poly_centr_y** (ndarray) – Numpy array containing latitudes of address points
- **poly_centr_x** (ndarray) – Numpy array containing longitudes of address points

Return type
Tuple[ndarray, ndarray]

Returns

- c_sekt_szer (np.ndarray) - rows indices of sectors for given coordinates
- c_sekt_dl (np.ndarray) - columns indices of sectors for given coordinates

geo_utilities.**get_sectors_params**()

Funtion that calculates basic parameters of sectors

Return type
Tuple[float, float, int, int]

Returns

- sekt_szer (float) - width of sectors
- sekt_dl (float) - height of sectors
- plnd_min_szer (int) - min latitude of Poland
- plnd_min_dl (int) - min longitude of Poland

geo_utilities.**get_super_permut_dict**(*max_len*)

Function that creates indices providing superpermutations for lists of strings with length of maximum 5 elements

Parameters
max_len (int) – Maximum length of superpermutation

Return type
Dict[int, List[int]]

Returns
Dictionary containing superpermutation indices

geo_utilities.**points_in_shape**(*c_paths*, *curr_coords*)

Checking if point lies inside shape of district

Parameters

- **c_paths** (List[Path]) – List containing matplotlib paths of regions
- **curr_coords** (ndarray) – Numpy array containing all address points in region

Return type
ndarray

Returns
Numpy array of flags indicating if given address point is inside given region shape

geo_utilities.**points_inside_polygon**(*grouped_regions*, *woj_name*, *trans_crds*, *points_arr*, *popraw_list*, *dists_list*, *zrodlo_list*, *bdot10k_ids*, *bdot10k_dist*, *sekt_kod_list*, *dod_opis_list*, *addr_phrs_list*, *addr_phrs_len*, *teryt_arr*, *json_arr*, *wrld_pl_trans*, *sekt_addr_phrs*)

Function that checks if given points are inside polygon of their districts and finds closest building shape for given PRG point

Parameters

- **grouped_regions** (Dict[Hashable, ndarray]) – Regions dictionary grouped by district and municipality name
- **woj_name** (str) – Name of the province
- **trans_crds** (ndarray) – Numpy array containing transformed coordinates of address points
- **points_arr** (ndarray) – Numpy array containing coordinates of address points
- **popraw_list** (List[int]) – List containing flags indicating if a given address point is valid
- **dists_list** (List[float]) – List cointaining distance of a given address point to its municipility border
- **zrodlo_list** (List[str]) – List containing names of the source of a given address point
- **bdot10k_ids** (ndarray) – Numpy array containing IDs of buildings from BDOT10k database
- **bdot10k_dist** (ndarray) – Numpy arrray cointaining distance of a given address point to closest building from BDOT10k database
- **sekt_kod_list** (ndarray) – Numpy array containing sector codes of address points
- **dod_opis_list** (ndarray) – Numpy array containing additional descriptions of an address point
- **addr_phrs_list** (List[str]) – List containing address points phrases
- **addr_phrs_len** (int) – Length of address points phrases list
- **teryt_arr** (ndarray) – Numpy array containing TERYT codes od address points
- **json_arr** (ndarray) – Numpy array containing GeoJSON shapes
- **wrld_pl_trans** (CoordinateTransformation) – Coordinates transformation that transforms spatial references from EPSG 4326 to EPSG 2180
- **sekt_addr_phrs** (ndarray) – Numpy array containing sectors of address points

Return type
None

Returns
The method does not return any values

geo_utilities.**reduce_coordinates_precision**(*geojson_poly*, *precision*)

Function that reduce decimal precision of coordinates in GeoJSON file:

- 0 decimal places is a precision of about 111 km
- 1 decimal place is a precsion of about 11 km

- 2 decimal places is a precison of about 1.1 km
- 3 decimal places is a precison of about 111 m
- 4 decimal places is a precison of about 11 m
- 5 decimal places is a precison of about 1.1 m
- 6 decimal places is a precison of about 11 cm
- 7 decimal places is a precison of about 1.1 cm

Parameters

- **geojson_poly** (str) – GeoJSON string representing polynomial shape
- **precision** (int) – The precision to which the accuracy of the coordinates should be reduced

Return type
str

Returns
Final GeoJSON string with reduce precision

geo_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

CHAPTER

FIVE

GEO_GUI.PY

Module that creates GUI window for GeocoderPL project

class geo_gui.**MyGeoGUI**

Bases: QWidget

Class that creates GUI window

__init__()

Method that creates objects from a class "MyGeoGUI"

Returns
The method does not return any values

change_sekts_order(*c_sekt*)

Method that changes order of sectors in adds_list

Parameters
c_sekt (ndarray) – Numpy array containing sectors numbers

Return type
None

Returns
The method does not return any values

on_text_changed()

Method that implements event on text change in QLineEdit

Return type
None

Returns
The method does not return any values

on_text_selected()

Methond that implements event on text select in QCompleter

Return type
None

Returns
The method does not return any values

geo_gui.**get_addr_spiral_ids**(*addr_arr_shp*, *spiral_ids_arr*)

Function that returs indices of numpy array in spiral mode up to 'add_arr' shape starting from central point

Parameters

- **addr_arr_shp** (int) – Shape of address array
- **spiral_ids_arr** (ndarray) – Numpy array with indices arranged in spirals

Return type
None

Returns
The method does not return any values

geo_gui.**get_prg_ids**(*prg_num*, *c_addrs*, *curr_text*, *ids_row*)

Function that generates indices of points in PRG table matching current text

Parameters

- **prg_num** (int) – PRG number
- **c_addrs** (str) – Current address
- **curr_text** (str) – Current text
- **ids_row** (List[str]) – List of indices

Return type
bool

Returns
Flag indicating that current text was found in PRG table

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

B

C

D

F

G

M