Documentation for *build_poi_database.py* functions

The *build_poi_database.py* file uses the Google Maps API to gather location data on points of interest and generate a MySQL table based on this information. This document provides details on the file's function descriptions.

Functions

`1. latPlusMeters(latitude, meters):`

Returns the approximate latitude resulting from traveling north or south across the surface of the earth for a certain number of meters. By convention, negative meters means south and vice versa.

Inputs

- latitude: The starting latitude.
- meters: The number of meters to travel north or south.

Output

- The new latitude value.

Example usage

```
latPlusMeters(34.0537136, 500)
```

`2. longPlusMeters(longitude, latitude, meters):`

Returns the approximate longitude resulting from traveling east or west across the surface of the earth for a certain number of meters. By convention, negative meters means west and vice versa.

Inputs

- longitude: The starting longitude.
- latitude: The current latitude. This is necessary as longitude lines get closer together at higher latitudes.
- meters: The number of meters to travel east or west.

Output

- The new longitude value.

Example usage

```
longPlusMeters(34.0537136, -118.24265330000003, 500)
```

`3. createSearchGrid(centerLat, centerLong, radius, gridSquareLength):`

Given a set of starting lat/long coordinates, a radius, and a grid square length, conceptually treats the radius as a concentric circle that fits perfectly inside of a square, breaks the square up into a grid where each grid square has the specified length (rounding up on the number of squares if the original square length is not evenly divisible), and returns the center coordinates of each grid square to be used for individual queries to Google maps. We break the original radius up in this way because it covers the whole original area (and more) with a higher granularity. It is not feasible to do a single large query to Google maps because each such query returns a maximum of 60 results.

Inputs

- centerLat: The latitude coordinate of the center of the original search radius.
- centerLong: The longitude coordinate of the center of the original search radius.
- radius: The radius of the original search, in meters.
- gridSquareLength: The length of each side of the grid squares in meters. This value should be at least 250 and generally a smaller value means higher granularity and more queries to run.

Output

- A list of grid square centers to use for queries to Google maps.

Example usage

```
createSearchGrid(34.0537136, -118.24265330000003, 1000, 250)
```

`4. searchArea(latitude, longitude, radius=DEFAULT_SEARCH_RADIUS_METERS, gridSquareLength=DEFAULT_GRID_SQUARE_LENGTH_METERS):`

Returns the final list of places that will be added to the database. The `addToDB` function adds the results of this function to the database. It loops through all the grid centers and places the results in one big list.

Inputs

- Latitude, Longitude: The center of the search area you want to search.
- Radius: How far out you would like to search from the center, in meters.
- gridSquareLength: The length of each square of the search grid, in meters. This controls the granularity of the search as larger grid squares will miss more locations past the 60 location per query limit.

Output

- A list of place objects.

Example usage

```
searchArea(34.0537136,-118.24265330000003, 500)
```

`5. getCertainTypesOfResults(lat, long, searchRadius):`

Returns a list of places of specific types. The types to be returned from the search are specified in the TYPES_OF_PLACES global variable. The `gmaps.places_nearby` function can only accept a single "type" parameter per query, so in order to search for more than one type of place, we have to loop through our TYPES_OF_PLACES list and get the places for each type individually and finally place all results in a single list to be returned.

Inputs

- Latitude, Longitude: The center of the search area you want to search. These are the grid centers passed in from the searchArea function.
- Radius: This is the gridSquareSearchRadius, which is the search radius for an individual grid square.

Output

- A list of place objects of specific types.

Example usage

```
getCertainTypesOfResults(34.0537136,-118.24265330000003, 12)
```

`6. getResults(lat, long, searchRadius, typeOfPlace):`

This function returns a list of places returned from the gmaps.places_nearby function. The `gmaps.place_nearby function` will return at most 60 results per query, and if there are more results, a next page token. This function will loop through the next page tokens and make sure to get all returned results. We retry an arbitrary number of times to the `gmaps.places_nearby` query with the next_page_token, because querying with this added parameter has unexpected behavior; however, after a few tries, the query goes through.

Inputs

- Latitude, Longitude: The center of the search area you want to search.
- Radius: How far out you would like to search from the center, in meters.
- typesOfPlace: the type of place to include in the `gmaps.places_nearby` query.

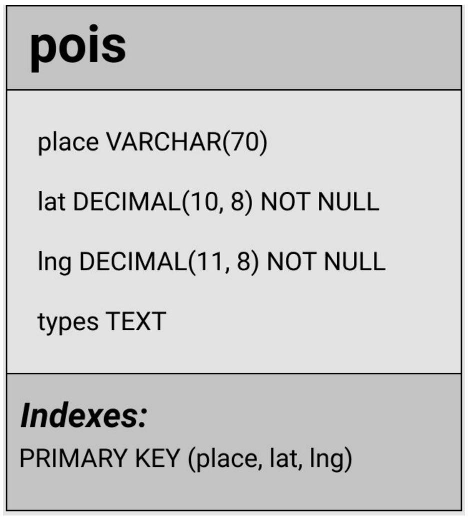
Output

- A list of place objects of a specific type.

Example usage
`getResults(34.0537136,-118.24265330000003, 500, ‘bar’)`

`7. addToDB(array):`

This function adds the information regarding the locations found by the `searchArea` function into a MySQL database. Each time the script is run, it will first check if the database ‘escality_location_db’ has already been created. If it has, it will overwrite the existing database; if not, it will create the database and use it. The database has a single table called ‘pois’ for the points of interest, and it ensures that duplicates (i.e. two points of interest with exactly the same name, latitude, and longitude coordinates) will not be added. The database schema is as follows:



pois

place VARCHAR(70)

lat DECIMAL(10, 8) NOT NULL

lng DECIMAL(11, 8) NOT NULL

types TEXT

Indexes:
PRIMARY KEY (place, lat, lng)

- The ‘place’ column contains the name of the point of interest.
    - E.g. “Los Angeles City Hall”
- The ‘lat’ column contains the latitude coordinate of the point of interest.
    - E.g. 34.05352670
- The ‘lng’ column contains the longitude coordinate of the point of interest.
    - E.g. -118.24293160
- The ‘types’ column contains the tags of the point of interest.
    - E.g.city_hall,premise,local_government_office,point_of_interest,establishment

Inputs
- A list of place objects. This is passed in from the result of the searchArea function.

Output
- None

Example usage
- `addToDB(searchArea(centerLat, centerLong, searchRadius))`

Note: Running this script will cause the pois table to be entirely regenerated. This ensures that the information in the database is up-to-date with the current information provided by the Google Maps API.