nyc-geoclient Documentation

Release 0.0.2

John Krauss

May 03, 2017

Contents

Python bindings for the REST NYC Geoclient API.

Release v0.0.2.

The NYC Geoclient API provides accurate geocoding for New York City. It supports lookups based off of address, BBL, BIN, blockface, intersection, and place name. It also provides informative error messages in cases where the geocoding fails.

These Python bindings make it easier to make requests to the API.

CHAPTER 1

Installation

See the Readme on GitHub for information on installation.

CHAPTER 2

Contents

Geoclient

NYC Geoclient should be used through the public methods of `Geoclient`.

class `nyc_geoclient.api.`**`Geoclient`**(*app_id*, *app_key*)

This object's methods provide access to the NYC Geoclient REST API.

You must have registered an app with the NYC Developer Portal (http://developer.cityofnewyork.us/api/geoclient-api-beta), and make sure that you check off access to the Geoclient API for the application. Take note of the Application's ID and key. You will not be able to use the ID and key until DoITT approves you – this could take several days, and you will receive an email when this happens. There isn't any indication of your status on the dashboard, but all requests will return a 403 until you are approved.

All methods return a dict, whether or not the geocoding succeeded. If it failed, the dict will have a *message* key with information on why it failed.

Parameters

- **`app_id`** – Your NYC Geoclient application ID.
- **`app_key`** – Your NYC Geoclient application key.

`address`(*houseNumber*, *street*, *borough*)

Given a valid address, provides blockface-level, property-level, and political information.

Parameters

- **`houseNumber`** – The house number to look up.
- **`street`** – The name of the street to look up.
- **`borough`** – The borough to look within. Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).

Returns A dict with blockface-level, property-level, and political information.

`address_zip`(*houseNumber*, *street*, *zip*)

Like the above address function, except it uses "zip code" instead of borough

Parameters

- **houseNumber** – The house number to look up.
- **street** – The name of the street to look up
- **zip** – The zip code of the address to look up.

Returns A dict with blockface-level, property-level, and political information.

bbl(*borough*, *block*, *lot*)
Given a valid borough, block, and lot provides property-level information.

Parameters

- **borough** – The borough to look within. Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).
- **block** – The tax block to look up.
- **lot** – The tax lot to look up.

Returns A dict with property-level information.

bin(*bin*)
Given a valid building identification number (BIN) provides property-level information.

Parameters **bin** – The BIN to look up.

Returns A dict with property-level information.

blockface(*onStreet*, *crossStreetOne*, *crossStreetTwo*, *borough*, *boroughCrossStreetOne=None*, *boroughCrossStreetTwo=None*, *compassDirection=None*)
Given a valid borough, "on street" and cross streets provides blockface-level information.

Parameters

- **onStreet** – "On street" (street name of target blockface).
- **crossStreetOne** – First cross street of blockface.
- **crossStreetTwo** – Second cross street of blockface.
- **borough** – The borough to look within. Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).
- **boroughCrossStreetOne** – (optional) Borough of first cross street. Defaults to value of borough parameter if not supplied.
- **boroughCrossStreetTwo** – (optional) Borough of second cross street. Defaults to value of borough parameter if not supplied.
- **compassDirection** – (optional) Used to request information about only one side of the street. Valid values are: N, S, E or W.

Returns A dict with blockface-level information.

intersection(*crossStreetOne*, *crossStreetTwo*, *borough*, *boroughCrossStreetTwo=None*, *compassDirection=None*)
Given a valid borough and cross streets returns information for the point defined by the two streets.

Parameters

- **crossStreetOne** – First cross street
- **crossStreetTwo** – Second cross street

- **borough** – Borough of first cross street or of all cross streets if no other borough parameter is supplied. Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).
- **boroughCrossStreetTwo** – (optional) Borough of second cross street. If not supplied, assumed to be same as borough parameter. Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).
- **compassDirection** – (optional) Optional. for most requests. Required for streets that intersect more than once. Valid values are: N, S, E or W.

Returns A dict with intersection-level information.

place (*name*, *borough*)

Same as 'Address' above using well-known NYC place name for input.

Parameters

- **name** – Place name of well-known NYC location.
- **borough** – Must be 'Bronx', 'Brooklyn', 'Manhattan', 'Queens', or 'Staten Island' (case-insensitive).

Returns A dict with place-level information.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

n

Index