python-geosupport

Ian Shiland, Jeremy Neiman

Jun 15, 2023

CONTENTS

Python bindings for NYC Geosupport Desktop Edition.

CHAPTER

ONE

CONTENTS

1.1 Installation

1.1.1 python-geosupport

You can install *python-geosupport* using pip:

```
$ pip install python-geosupport
```

Or clone this repository, *cd* into it and:

```
$ python setup.py install
```

1.1.2 Geosupport Desktop Edition

python-geosupport requires a local installation of Geosupport Desktop Edition (19b):

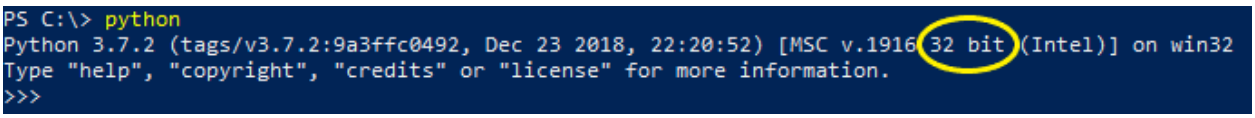
- Geosupport Desktop Edition for Windows (32-bit)
- Geosupport Desktop Edition for Windows (64-bit)
- Geosupport Desktop Edition for Linux

Geosupport Desktop is not available for Mac but it can be used through a containerized environment like Docker. See the *Related Projects* section for more information.

Windows

Important: Ensure you select the correct Geosupport installation that corresponds to the Python interpreter you are using. Ex., Python 32-bit will only work with Geosupport 32-bit.

You can determine if you are running a 32-bit or 64-bit python interpreter by opening the command prompt or powershell and typing *python*. You should see something like this:



Environmental Variables

By default, a clean installation of Geosupport Desktop Edition will set the appropriate environmental variables. If you are experiencing issues setting up your environment, you will want to ensure the following environmental variables are set.

- **GEOFILES**: Path to the *fls* directory of Geosupport. Example: `C:\Program Files (x86)\Geosupport Desktop Edition\fls\`
- **PATH**: Add the *bin* directory of Geosupport to the *path* variable. Example: `C:\Program Files (x86)\Geosupport Desktop Edition\bin`

Alternatively see the *Configuration* section for other (easier) options on specifying a path to the Geosupport directory.

Linux

Extract the .zip to a folder of your choice and set the `GEOFILES` and `LD_LIBRARY_PATH` environmental variables of the fls and lib directories:

```
export GEOFILES=/var/geosupport/version-19b/fls/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/var/geosupport/version-19b/lib/
```

1.2 Configuration

Important: These features are available on Windows only. Linux doesn't support library path modifications during runtime. These configurations override any Geosupport environmental variables.

1.2.1 Geosupport Path

If you have multiple versions of geosupport and want to switch between them or you simply want to specify the path to Geosupport, you can pass the installation path to `Geosupport`:

```
from geosupport import Geosupport

g = Geosupport(geosupport_path="C:\\Program Files\\Geosupport 19B")
```

1.2.2 Configuration File

You can also create a configuration file that persists any Geosupport path settings.

Create a *.python-geosupport.cfg* in your home directory that specifies the names and installation paths of your Geosupport versions.

The *.python-geosupport.cfg* file looks like:

```
[versions]
18b=C:\Program Files\Geosupport Desktop Edition
18c=C:\Program Files\Geosupport 18C
18c_32=C:\Program Files (x86)\Geosupport 18C
```

Then you can select the version by name:

```
g = Geosupport(geosupport_version="18c")
```

1.3 Usage

1.3.1 Basic Usage

```
# Import the library and create a `Geosupport` object.
from geosupport import Geosupport
g = Geosupport()

# Call the address processing function by name
result = g.address(house_number=125, street_name='Worth St', borough_code='Mn')
```

`result` is a dictionary with the output from Geosupport. For example:

```
{
    '2010 Census Block': '1012',
    '2010 Census Tract': '31',
    'Assembly District': '65',
    'Atomic Polygon': '112',
    'B10SC - First Borough and Street Code': '14549001010',
    'BOE Preferred B7SC': '14549001',
    'BOE Preferred Street Name': 'WORTH STREET',
    'BOROUGH BLOCK LOT (BBL)': {
        'BOROUGH BLOCK LOT (BBL)': '1001680032',
        'Borough Code': '1',
        'Tax Block': '00168',
        'Tax Lot': '0032'
    },
    'Blockface ID': '0212261942',
    ...
}
```

1.3.2 Calling Geosupport Functions

python-geosupport is flexible with how you call functions. You can use either Geosupport function codes or human readable alternate names, and access them either through python object attribute notation or dictionary item notation:

```
# Different ways of calling function 3S which processes street stretches
g.street_stretch(...)
g['street_stretch'](...)
g['3S'](...)
g.call({'function': '3S', ...})
g.call(function='3S', ...)
```

You can pass arguments as a dictionary, keyword arguments.

```
# Use a dictionary with short names
g.street_stretch({'borough_code': 'MN', 'on': '1 Av', 'from': '1 st', 'to': '2 st'})
# Use keyword arguments with short names
g.street_stretch(
    borough_code='MN', street_name_1='1 Av',
```

(continues on next page)

(continued from previous page)

```
    street_name_2='1 st', street_name_3='9 st'
)
# Use dictionary with full names
g.street_stretch({
    'Borough Code-1': 'MN',
    'Street Name-1': '1 Av',
    'Street Name-2': '1 st',
    'Street Name-3': '9 st'
})
```

1.3.3 Mode

A number of Geosupport functions support several modes: Exetended, Long, and TPAD Long. You can set the flags individually as you would with using Geosupport directly, but python-geosupport makes it easier with the `mode` argument. `mode` can be one of `regular` (default), `extended`, `long` and `long+tpad`.

```
# Call BL (Block and Lot) function in long mode
g.BL(mode='long', ...)
g.BL(mode='long+tpad', ...) # With TPAD

# Call 3 (Street Segment) function in extended mode
g.street_segment(mode='extended', ...)
```

1.4 Error Handling

python-geosupport will raise a `GeosupportError` when Geosupport returns an error code. Sometimes there is more information returned, in which case the exception will have a result dictionary.

```
from geosupport import GeosupportError

try:
    g.get_street_code(borough='MN', street='Wort Street')
except GeosupportError as e:
    print(e) # 'WORT STREET' NOT RECOGNIZED. THERE ARE 010 SIMILAR NAMES.
```

1.5 Interactive Help

Full function help can be viewed by calling `g.help()`.

```
# View an overview of all the functions available:
g.help()

# View help for an individual function including a description, inputs, outputs and␣
↪valid modes.
g.address.help()
g.help('address')

# View a list of all possible inputs to Geosupport
g.help('input')
```

1.6 Development

1.6.1 Running tests

```
$ python setup.py test
```

Installing the dev dependencies (nosetests and invoke) will give you more control over running tests. Install in develop mode with dev dependencies with:

```
$ pip install -e .[dev]
```

And then run tests with:

```
$ invoke test unit
$ invoke test functional
$ invoke test all
```

1.7 Related Projects

- Geosupport UPG Official user guide to the Geosupport System.
- api-geosupport Create a RESTful web API using *python-geosupport* and *Flask*.
- geosupport-suggest Retrieve address suggestions from Geosupport using *python-geosupport* and a single input address.
- node-geosupport Node.js bindings for Geosupport.
- geosupport-docker Dockerfiles for installing, configuring and using Geosupport from a Docker container.
- docker-geosupport Dockerized Geosupport.
- python-geoclient Geocode with Python using the RESTful NYC Geoclient API.