CH Zip Documentation

Release 0.1

Mathieu Clément

February 06, 2015

Contents

Contents:

CHAPTER 1

Preface

1.1 The Issue

The Swiss Postal Service provides a list of zip codes with the associated locality names. This is great, but to use it one would have to download the tab-delimited text file, convert it to a format easily accessible with Python. After that one would have to figure out the format of the data and what are all the fields for.

Most people having the need to look for zip codes and locality names would do it as simple as possible, ending with something hard to understand and not easily re-usable. They would probably not implement something such as automatic unpacking and updating and keep obsolete data in a database, which could lead to serious problems.

More importantly they would have lost a lot of time parsing a stupidly formatted file instead of spending more time on their work or with their friends, their family, their pets or taking care of them.

1.2 The Goal

The goal of this project is to avoid every Python programmer in Switzerland who need this data to figure out how to use it and implement a custom solution from scratch.

This library has been developed with its users in mind, and I am looking forward for your feedback and your usecases.

The most important thing I would like to archive is that you can install chzip and look for locality names associated with a zip code under a minute. Look at the *Usage* page and try this thing out, then tell me if I have succeeded.

CHAPTER 2

Installation

2.1 Manually

Download the latest version from PyPI and install using:

```
tar zxvf chzip-0.1.tar.gz
cd chzip-0.1/
sudo python setup.py install
```

2.2 Using *pip*

You can also install it with pip:

```
pip install chzip
```

Warning: Installation with pip seems broken. If you have trouble with this method, perform the installation manually.

2.3 Using *easy_install*

I do not recommend (for these reasons) to use *easy_install*.

CHAPTER 3

Data upgrade

After *installing chzip*, the ZIP codes and city names are already up-to-date.

This data is updated the 2nd of each month [1]. I recommend you to set up a cron job to never miss one update.

3.1 Update automatically using a cronjob

On UNIX, create a script with the following content:

```
#!/usr/bin/env python3
import chzip
chzip.upgrade_all()
```

Make it executable with `chmod +x my_script.py` and add this to your cronjobs with *crontab -e* to update the data every 2nd of the month at 03:04 AM:

```
# m h dom mon dow command
  4 3 2   *   *   /somewhere/my_script.py
```

If you don't want to create a separate file just for this you can also run it with `python -c 'import chzip; chzip.upgrade_all()'`

Note: The user running the `chzip.ChZip.upgrade_all()` method must have permissions to write into the installation folder. If you installed this library with your package manager or using *sudo*, you will need to be *root* to do that.

---

[1] Swiss Post updates are available on the 1st of the month but the data is valid from the 2nd. It makes almost no difference if you get the data one day earlier. By the way, ZIP codes don't change often, and if that's the case, the mail will be "redirected" to the appropriate location in the meantime. So, don't worry!

CHAPTER 4

Usage

4.1 Initialization

```
from chzip import ChZip, Locality
z = ChZip()
```

4.2 Download directory

The resources have been downloaded to a "resource directory" on your disk. It contains all the files needed for this library to work.

It can be set manually by calling the constructor with the absolute path as argument.

It defaults to the *res/* directory inside the installation path. If changed, you will need to call `chzip.download_and_unpack_all()` once, and `chzip.upgrade_all()` after that.

4.3 Get all localities matching a ZIP code

Example for ZIP code 1696, giving one result, i.e. one `chzip.common.Locality`, with a different short (18 chars) and long (27 chars) name:

```
> results = z.find(1696)
[<Locality[short_name='Vuisternens-Ogoz',
          long_name='Vuisternens-en-Ogoz',
          canton='FR']>]

> results[0].long_name
"Vuisternens-en-Ogoz"
```

Note: If possible, always use the long name.

4.4 Get all locality *names* matching a ZIP code

If all you want from the section "*Get all localities matching a ZIP code*" is the long name of localities, you can use this little snippet of code:

```
z.long_names( zip=1696 )
```

See also:

chzip.common.Locality.short_name and chzip.common.Locality.long_name to understand the difference between these two formats.

4.5 Find localities matching the exact name

```
> z.find( long_name='Vuisternens-en-Ogoz' )

[<Locality[short_name='Vuisternens-Ogoz',
long_name='Vuisternens-en-Ogoz',
canton='FR']>]
```

4.6 Find localities containing a search term

The following method is very handy when you need to match the user input with a locality of the database. It will returns a list of matching localities as previously.

```
> z.find( long_name_like='Vuisternens-en-Ogoz' )
[<Locality[short_name='Vuisternens-Ogoz',
           long_name='Vuisternens-en-Ogoz',
           canton='FR']>]
```

Warning: You are responsible to take care of the query string as this may return thousands of results with dumb queries, which could use a lot of memory. You may be served better with the *all* method as explained here: *Get all localities (no search criteria)*.

4.7 Get all localities (no search criteria)

chzip.ChZip.all() may be used to fetch all the data:

```
for locality in z.all():
    do_something_with( locality )
```

If you are looking for a list, then:

```
l = list( z.all() )
```

CHAPTER 5

Exploring the API

5.1 Summary

chzip	The *chzip* package provides a quick and easy Python interface to look for zip codes and cities in Sv
chzip.ChZip([resource_dir])	The main entrance for the *chzip* library.
chzip.common	Common classes and utilities functions
chzip.zipcodes	Implementation classes that do the heavy lifting of the module

5.2 The *chzip* package

The *chzip* package provides a quick and easy Python interface to look for zip codes and cities in Switzerland.

chzip.**download_and_unpack_all**(*download_dir='chzip/install/path/res'*)

Download and unpack all (available) resources.

Use `upgrade_all()` if you don't want to re-download everything but only update what has changed. This method begins by erasing previous versions in the download directory.

Parameters **download_dir** (*str*) – Download directory (with write permissions)

Note: You are very unlikely to ever call this method, except if you included chzip directly in your project, without installing it.

---

chzip.**upgrade_all**(*download_dir='chzip/install/path/res'*, *force=False*)

Upgrade all resources.

Unlike *download_all* resources files that have not been changed will not be erased and downloaded again. Also, it takes care of keeping old (working) files if the upgrade fails (network error, bad download, corrupted file, error when unpacking, etc.)

It is recommended to run this method regularly to get up-to-date data.

Parameters

- **download_dir** (*str*) – Download directory (with write permission) containing previous versions of the resource files.
- **force** (*bool*) – Force upgrade, even if files are up-to-date.

5.3 The ChZip class

class chzip.**ChZip**(*resource_dir='chzip/install/path/res'*)
The main entrance for the *chzip* library.

all()
Returns all the localities from the database, in no particular order.

find(*zip=None*, *long_name=None*, *short_name=None*, *canton=None*, *onrp=None*)
Refer to *Usage*.

long_names(**args*, ***kwargs*)
Is the same as `find()` but returns a list of long names instead.

5.4 The Common and Utility classes

Common classes and utilities functions

class chzip.common.**Downloader**
The parent (abstract) class of Downloaders. Can NOT be instantiated.

Subclasses can redefine `DEFAULT_RES_DIRNAME`.

DEFAULT_RES_DIRNAME = 'res'
The name of the directory that will contain the downloaded files.

download_and_unpack(*download_dir*)
Download and unpack resources files related to this downloader. This method should probably not be called on a UI thread or similar because this can be a long-running task.

upgrade_and_unpack(*download_dir*)
Similar to `download_and_unpack()` but does not do anything if files are up-to-date and does not let you with a broken installation in case of a download or unpacking failure.

class chzip.common.**Locality**(*zip=0*, *short_name=None*, *long_name=None*, *canton=None*, *_zip_type_number=None*, *_onrp=None*)
A locality is the name of a town, village or any "string" that goes after the ZIP code in the address.

canton = None
Canton. Official 2-letter uppercased abbreviation as used for vehicles number plates.

Note: FL is used for Liechtenstein addresses, and so is DE for 8238 Büsingen and IT for 66911 Campione. These are the only exceptions.

---

long_name = None
Long name (27 chars). Official designation. Must be preferred to *short_name*.

short_name = None
Short name (18 chars)

zip = 0
ZIP code

zip_type = None
A ZIP code might be only available for PO boxes, physical adresses, both of these, an entreprise, or dedicated for mail sorting.

See also:

ZipType()

class chzip.common.**ZipType**
The ZipType tells if A ZIP code might be only available for PO boxes, physical adresses, both of these, an entreprise, or dedicated for mail sorting.

ENTERPRISES = 40
Enterprises

HOMES_AND_PO_BOXES = 10
Homes and PO boxes

HOMES_ONLY = 20
Homes only

INTERNAL = 80
For mail sorting by the post offices. Only used by the Post!

PO_BOXES_ONLY = 30
PO boxes only

static okay_for_stuff_delivery(*zip_type*)
Returns true if the specified ZIP type is appropriate to deliver merchandise. Returns False for a "PO boxes only" or "internal" ZIP code.

5.5 Where the magic happens

Implementation classes that do the heavy lifting of the module

class chzip.zipcodes.**Downloader**
The Downloader of Swiss Post MAT[CH] ZIP resources.

download_and_unpack(*download_dir*)
Downloads the ZIP files from MAT[CH], extract the text file, populate the internal SQLite3 database, and delete all the intermediary files.

Parameters **download_dir** (*str*) – Resource directory that will contain both the temporary files (which will be deleted) and the database.

is_up_to_date(*abs_resource_file*)

Returns true if the specified file is up-to-date.

upgrade_and_unpack(*download_dir*, *force=False*)
Does the same as download_and_unpack() but returns immediately if files are up-to-date and do not delete previous database. This can be handy in cases of failure, so is the recommended way to update the database.

Parameters

- **download_dir** (*str*) – Resource directory that will contain both the temporary files (which will be deleted) and the database.
- **force** (*bool*) – Set to true to force update even if files are up-to-date.

class chzip.zipcodes.**ResourceInstaller**(*zip_path*, *extract_dir*, *extract_to_filename*)
Unzip the file downloaded from Swiss Post and install it at the appropriate location.

Usage: call unzip() and install() for a complete installation.

delete()
Delete temporary files

extracted_txt_path()

Returns always returns the location of the text file, no matter if `install()` was called or not.

install()
Move file to its definitive location

real_path()

Returns the path of the extracted text file, before moving it.

So this is the real path of the file just after unzipping.

If you want to work on the file before installing, call this method else call `wanted_path()`. `extracted_txt_path()` will figure this out for you.

unzip()
Unzip the first file from ZIP to the specified directory

Returns the filename extracted

wanted_path()

Returns the desired path of the extracted text file.

CHAPTER 6

Indices and tables

- *genindex*
- *modindex*
- *search*

C