tkrajina / gpxpy

gpx-py is a python GPX parser. GPX (GPS eXchange Format) is an XML based file format for GPS tracks.

gpx gpx-library python python3 gps

521 commits 15 branches 9 releases 23 contributors Apache-2.0

tkrajina Merge remote-tracking branch 'origin/master' Latest commit 2d965f3 on 24 Dec 2016

gpxpy	Merge remote-tracking branch 'origin/master'	2 months ago
test_files	More descriptive name for file	9 months ago
xsd	GPX1.0 and GPX1.1 schemas	3 years ago
.gitignore	Merge branch 'master' into gpx-1.1	3 years ago
.travis.yml	Run unit tests with coverage on Travis CI	11 months ago
LICENSE.txt	Licensed under Apache 2.0 License	5 years ago
NOTICE.txt	Added George Titsworth in NOTICE	3 years ago
README.md	README about GPX 1.0/1.1 changes	7 months ago
TODO.txt	waypoint tests	3 years ago
gpxinfo	add -s as a command line option via argparse	5 months ago
makefile	...	3 months ago
setup.py	PGP signature on pypi	3 months ago
test.py	parse time with single digits and tests added	3 months ago
validation_gpx10.gpx	...	3 months ago
validation_gpx11.gpx	...	3 months ago

README.md

build passing coverage 85%

gpxpy -- GPX file parser

This is a simple Python library for parsing and manipulating GPX files. GPX is an XML based format for GPS tracks.

You can see it in action on my online GPS track editor and organizer.

There is also a Golang port of gpxpy: gpxgo.

See also srtm.py if your track lacks elevation data.

Usage

```
import gpxpy
import gpxpy.gpx

# Parsing an existing file:
# -------------------------

gpx_file = open('test_files/cerknicko-jezero.gpx', 'r')
```

```
gpx = gpxpy.parse(gpx_file)

for track in gpx.tracks:
    for segment in track.segments:
        for point in segment.points:
            print 'Point at ({0},{1}) -> {2}'.format(point.latitude, point.longitude, point.elevation)

for waypoint in gpx.waypoints:
    print 'waypoint {0} -> ({1},{2})'.format(waypoint.name, waypoint.latitude, waypoint.longitude)

for route in gpx.routes:
    print 'Route:'
    for point in route.points:
        print 'Point at ({0},{1}) -> {2}'.format(point.latitude, point.longitude, point.elevation)

# There are many more utility methods and functions:
# You can manipulate/add/remove tracks, segments, points, waypoints and routes and
# get the GPX XML file from the resulting object:

print 'GPX:', gpx.to_xml()

# Creating a new file:
# --------------------

gpx = gpxpy.gpx.GPX()

# Create first track in our GPX:
gpx_track = gpxpy.gpx.GPXTrack()
gpx.tracks.append(gpx_track)

# Create first segment in our GPX track:
gpx_segment = gpxpy.gpx.GPXTrackSegment()
gpx_track.segments.append(gpx_segment)

# Create points:
gpx_segment.points.append(gpxpy.gpx.GPXTrackPoint(2.1234, 5.1234, elevation=1234))
gpx_segment.points.append(gpxpy.gpx.GPXTrackPoint(2.1235, 5.1235, elevation=1235))
gpx_segment.points.append(gpxpy.gpx.GPXTrackPoint(2.1236, 5.1236, elevation=1236))

# You can add routes and waypoints, too...

print 'Created GPX:', gpx.to_xml()
```

GPX Version:

gpx.py can parse and generate GPX 1.0 and 1.1 files. Note that the generated file will always be a valid XML document, but it may not be (strictly speaking) a valid GPX document. For example, if you set gpx.email to "my.email AT mail.com" the generated GPX tag won't confirm to the regex pattern. And the file won't be valid. Most applications will ignore such errors, but... Be aware of this!

WARNING: The only part of the GPX standard which is not completely implemented are GPX extensions. The API for GPX extensions will change in future versions!!!

Be aware that the gpxpy object model is not 100% equivalent with the underlying GPX XML file schema. That's because the library object model works with both GPX 1.0 and 1.1.

For example, the GPX 1.0 specified a `speed` attribute for every track point, but that was removed in GPX 1.1. If you parse GPX 1.0 and serialize back with `gpx.to_xml()` everything will work fine. But if you have a GPX 1.1 object, changes in the `speed` attribute will be lost after `gpx.to_xml()`. If you want to force using 1.0, you can `gpx.to_xml(version="1.0")`. Another possibility is to use `extensions` to save the speed in GPX 1.1.

XML parsing

If lxml is available, then it will be used for XML parsing. Otherwise minidom is used. Note that lxml is 2-3 times faster so, if you can choose -- use it :)

The GPX version is automatically determined when parsing by reading the version attribute in the gpx node. If this attribute is not present then the version is assumed to be 1.0. A specific version can be forced by setting the `version` parameter in the parse function. Possible values for the 'version' parameter are `1.0`, `1.1` and `None`.

Pull requests

OK, so you found a bug and fixed it. Before sending a pull request -- check that all tests are OK with Python 2.6+ and Python 3+.

Run all tests with:

```
$ python -m unittest test
$ python3 -m unittest test
```

Run only minidom parser tests with:

```
$ python -m unittest test.MinidomTests
$ python3 -m unittest test.MinidomTests
```

Run only lxml parser tests with:

```
$ python -m unittest test.LxmlTests
$ python3 -m unittest test.LxmlTests
```

Run a single test with:

```
$ python -m unittest test.LxmlTests.test_method
$ python3 -m unittest test.LxmlTests.test_method
```

GPXInfo

The repository contain a little command line utility to extract basic statistics from a file. Example usage:

```
$ gpxinfo voznjica.gpx
File: voznjica.gpx
  Length 2D: 63.6441229018
  Length 3D: 63.8391428454
  Moving time: 02:56:03
  Stopped time: 00:21:38
  Max speed: 14.187909492m/s = 51.0764741713km/h
  Total uphill: 1103.1626183m
  Total downhill: 1087.7812703m
  Started: 2013-06-01 06:46:53
  Ended: 2013-06-01 10:23:45
```

License

GPX.py is licensed under the Apache License, Version 2.0