pygcbrowse Documentation

Release 0.2.1 final

Jochen Schnelle

November 17, 2014

Contents

CHAPTER 1

User Manual (English)

This is the official, English documentation of pygcbrowse .

1.1 What is pygcbrowse?

pygcbrowse is a little program which allows you to custom-search by user-defined search terms through your geocaches logged as "Found it" at geocaching.com.

The program offers a HTML-based user-interface, which runs completely local in your browser. pygcbrowse does *not* require any online connection nor does it transfer *any* data over the network / internet.

1.2 Prerequisites

There are two prerequisites to use pygcbrowse, which are both fairly easy to fulfil: Python 3 needs to be installed on your computer and you need the GPX file with your finds locally saved.

1.2.1 Installing Python

pygcbrowse is entirely written in Python, thus it needs Python installed on your computer. pygcbrowse requires Python 3.2 or later, Python 2.7 is *not* supported.

On Linux and MacOS, Python is typically included, otherwise install it via the package manager.

On Windows, please visit the Python Website , download and install the latest version of Python 3. Furthermore, using Python on Windows is sometimes easier when you add the Python interpreter to the computer's standard search path. In case you don't know how to do that, just skip it or read this article (it's not as complicated as it seems to be initially...)

1.2.2 Getting the 'my finds' GPX file

To get the 'my finds' GPX file, five little steps are necessary:

1. Log in to geocaching.com
2. Go to the site www.geocaching.com/pocket/
3. Scroll down until you see button labelled *"Add to Queue"* and click onto that.
4. Once the pocket query finished running (which takes typically only a minute or so), you will get an e-mail telling you that a file is ready to download - do so.
5. In case the file is a zip file, open it. Inside you should find a GPX file with a file name consisting of numbers only. Unpack this file to your hard disk. Actually it is more handy if you place the file in the same directory where pygcbrowse is located - but this is not a must.

Unfortunately, you have to be a Premium-Member at geocaching.com to run Pocket Queries. For regular users it is not possible to do so.

1.2.3 Bringing pygcbrowse to your computer

pygcbrowse does not require a installation in a common sense. The only thing you have to do is to download from the project page the latest, stable version, which is typically provided as a zip file. Unpack this to any directory of our choice.

That's it. Once this is done, pygcbrowse is good to go.

1.3 Usage

pygcbrowse provides a HTML-based user interface which runs locally in your browser. Run the file:

```
$ python3 pygcbrowse.py
```

and your browser should open automatically. If not, open your web browser manually and enter the URL `http://localhost:8888/index.html`

Note that *no* data is transferred at any time over the internet or network! The computer can be completely offline when using pygcbrowse.

In the example above, pygcbrowse will look for a single GPX-file (holding all data on your finds) saved in the directory where the file **pygcbrowse.py** is located. In case there is no GPX-file or more than one, an error message will be displayed and the program aborts.

To specify a GPX-file to be loaded, simply add the `-i` option:

```
$ python3 pygcbrowse.py -i 8765432.gpx
```

which tells pygcbrowse to load the file **8765432.gpx** from the local directory. In case the file is located in another directory, just provide the full path along with the file name:

```
$ python3 pygcbrowse.py -i /path/to/file/8765432.gpx
```

Once the file is found and successfully loaded, a non-persistent in-memory database will be created. pygcbrowse does *not* write any files to your hard disc.

Now you will see the user-interface in your browser window. Here you can enter any search term in one or more fields. When entering more than one search term, the terms will be concatenated with "and", e.g. owner = "Supercacher" and cache name = "Secrete cache".

For text fields, the search terms are considered as parts, not a complete matches. So e.g. a search term for a cache owner like "bear" would find owner names like "BearAndWolf", "the bear" as well as "Bearing".

And finally the search is not case-sensitive, so searching for "great" will give the same results as searching for "Great".

To start the search, press the *"Search"* button and the result page is shown, displaying all results in a table.

Above the table the current search term is displayed as well as the number of matches found. Below the table, the total number of caches found is shown. To make a new search, just click on *"New Search"* at the bottom of the page.

The result table is sorted by default by the log date in ascending order. However, the table can be sorted by any of the displayed columns. To sort e.g. by cache owner name in lexical order, just click on the header of the column. To alter between ascending and descending sorting, just click again on the header of the column.

Please note that the latter feature, custom sorting by column, requires JavaScript to be enabled in your browser, which is default for all browsers anyway.

When no further searching is required, just close the browser (or tab inside the browser), go back to the terminal window and stop the locally running server started by pygcbrowse by pressing the keys *"CTRL"* and *"C"* at the same time.

1.3.1 Setting your home coordinates

pygcbrowse offers a search option where the distance of found caches to your home coordinates can be specified. In case this search should be applied, the home coordinates need to be set correctly.

pygcbrowse reads the latitude and longitude of the home coordinates from the file **conf.ini**, which is located in the same directory as the **pygcbrowse.py** file. To set your home coordinates, open the file with an editor of your choice (e.g. notepad on Windows-based system) and set the right values for lat and lon at the corresponding keys, e.g.:

```
lat = 50.12345
lon = 7.98765
```

which corresponds to N 50.12345 E 007.98765

Please note that the coordinates needs to be set as decimal degree - as in the above example, without any prefixed N, S, E or W. Thus, any west and south coordinates need to be prefixed with a minus - sign, e.g.:

```
lat = -99.16721
lon = -85.01023
```

which corresponds to S 99.16721 W 85.01023

In case the is no need for a search by distance, there is no need to set the coordinates in the **conf.ini** file. Please note that both values default zero (0). In case an invalid value is given for lat or lon, pygcbrowse while output an message to the terminal and set the value to zero (0) instead.

And finally a small note on converting coordinates to decimal degree: The more common format for coordinates, at lease at geocaching, is indeed degree plus decimal minutes. However, converting this representation to decimal degree is easy. Just divide the decimal minutes by 60 and you get the decimal places.

An example: Coordinates like e.g. N 50°25.123 can be converted to decimal degree as follows: 50 + (25.123/60) = 50.418716667. As said above, south and west coordinates need to be prefixed additionally with a minus - to get the correct representation in decimal degree.

And, finally, it is worth to point out that the distance is calculated in Kilometers (and not Miles or so).

CHAPTER 2

Dokumentation (Deutsch)

Dies ist die offizielle, deutschsprachige Dokumentation zu pygcbrowse .

2.1 Was ist pygcbrowse?

pygcbrowse ist ein kleines Programm, mit dessen Hilfe man seine bei geocaching.com gefunden Caches frei durchsuchen kann.

Das Programm besitzt eine HMTL-basierte Oberfläche, welche lokal im Webbrowser läuft. pygcbrowse benötigte *keine* Internetverbindung und es werden auch *keine* Daten über das Netzwerk / Internet übertragen.

2.2 Vorbereitung und Voraussetzungen

Um pygcbrowse nutzen zu können, müssen zwei einfache Voraussetzungen erfüllt sein: Es muss Python 3 auf dem Rechner installiert sein und die GPX-Datei mit den gefundenen Caches muss lokal auf dem Rechner gespeichert sein.

2.2.1 Python installieren

pygcbrowse ist vollständig in Python programmiert und es wird zur Ausführung des Programms benötigt. pygcbrowse benötigt Python 3.x, Python 2.7 wird nicht unterstützt.

Unter Linux und MacOS ist Python üblicherweise vorinstalliert. Wenn nicht, kann dieses über die Paketverwaltung nachinstalliert werden.

Windows-Nutzer können von der Python Homepage die neuste Version von Python 3 herunterladen und installieren. Die Nutzung von Python unter Windows ist oft einfacher, wenn der Python-Interpreter dem Standardsuchpfad des Systems hinzugefügt wird. Wer jetzt keine Ahnung hat, was dies bedeutet kann einfach zum nächsten Abschnitt springen - oder den entsprechenden Artikel in der Python-Dokumentation lesen (es ist einfacher, als es zuerst den Anschein hat!)

2.2.2 Die ‘my finds’ GPX-Datei erzeugen

Um die ‘my finds’ GPX-Datei zu erzeugen sind fünf kleine Schritte notwendig:

1. bei geocaching.com einloggen
2. die Seite www.geocaching.com/pocket/ auswählen
3. Nach unten scrollen, bis die Schaltfläche *“Zur Warteschlange hinzufügen”* erscheint. Dann diese anklicken.

4. Sobald der Pocket Query fertig erstellt ist (dauert typischerweise weniger als eine Minute), wird die entsprechende Meldung automatisch per E-Mail verschickt, inklusive dem Download-Link. Diese Datei muss nun heruntergeladen werden.
5. Falls die Datei eine zip-Datei ist, muss diese entpackt werden. In dem zip-Archiv befindet sich eine GPX-Datei, deren Dateiname nur aus Zahlen besteht. Dies ist nun aus dem Archiv zu extrahieren. Praktischer Weise sollte die GPX-Datei im selben Verzeichnis wie pygcbrowse gespeichert werden. Dies ist aber keine Pflicht.

Leider können nur Premium-Mitglieder Pocket Queries erstellen, die kostenlose Mitgliedschaft bei geocaching.com erlaubt dies nicht.

2.2.3 pygcbrowse auf dem Rechner installieren

pygcbrowse bedarf keiner Installation im eigentlichen Sinne. Es wird lediglich die neuste, stabile Version von pygcbrowse benötigt, welche von der Projektseite als zip-Datei herunterladbar ist. Dieses Archiv ist vollständig, also inklusive der darin enthaltenen Verzeichnisse, in eine beliebiges Verzeichnis (wie z.B. eigene das Home-Verzeichnis) zu entpacken.

Das ist alles. pygcbrowse ist nun einsatzbereit.

2.3 Benutzung

pygcbrowse bietet eine HTML-basierte Oberfläche, welche vollständig lokal im Browser läuft. Diese kann über den folgenden Befehl gestartet werden:

```
$ python3 pygcbrowse.py
```

Der Browser sollte sich dann automatisch öffnen. Falls nicht, so ist der Webbrowser manuell zu starten und die Adresse `http://localhost:8888/index.html` aufzurufen.

pygcbrowse überträgt *keinerlei* Daten ins Internet! Das Programm läuft komplett lokal.

Bei Aufruf der obigen Befehls sucht pygcbrowse nach einer einzelnen GPX-Datei (welche die Daten zu den gefundenen Caches enthält) im Verzeichnis, in dem pygcbrowse installiert ist. Liegen dort mehrere GPX-Dateien oder wird keine gefunden, wird eine Fehlermeldung ausgegeben und das Programm beendet.

Mit der Option `-i` kann explizit die zu nutzende GPX-Datei angegeben werden:

```
$ python3 pygcbrowse.py -i 8765432.gpx
```

pygcbrowser lädt so die Datei **8765432.gpx** aus dem lokalen Verzeichnis. Sollte die Datei in einem anderen Verzeichnis liegen, dann muss dem Dateinamen noch der volle Pfad vorangestellt werden:

```
$ python3 pygcbrowse.py -i /pfad/zur/datei/8765432.gpx
```

Sobald die Datei erfolgreich geladen wurde wurde eine nicht-persistente Datenbank im RAM des Computers angelegt. pygcbrowse schreibt *keinerlei* Daten auf lokale oder andere Laufwerke.

Nun erscheint im Browser die Benutzeroberfläche, standardmäßig auf Englisch. Um auf die deutsche Seite zu wechseln, ist einfach auf *"Zur deutschen Seite wechseln"* zu klicken. Hier können ein oder mehrere Suchbegriff eingegeben werden. Wird mehr als ein Suchbegriff vorgegeben, werden die Begriff mit "und" verknüpft, also z.B owner = "Supercacher" and cache name = "Secrete cache".

Die Suchbegriffe werden als Teil des gesamten Begriffs gesehen, nicht als der komplette Begriff. So findet z.B. der Suchbegriff "bear" bei der Suche nach dem Cacheowner sowohl "BearAndWolf", "The Bear" als auch "Bearing".

Die Suche unterscheidet nicht zwischen Groß- und Kleinschreibung. D.h. das die Suche nach "Supercacher" die gleichen Suchtreffer gibt wie die Suche nach "supercacher".

Um die Suche zu starten, ist auf die Schaltfläche *"Suchen"* weiter unten auf der Seite zu klicken. Daraufhin erscheinen die Suchergebnisse als Tabelle.

Der aktuelle Suchbegriff wird oberhalb der Tabelle angezeigt, ebenso die Anzahl der Suchtreffer. Unterhalb der Tabelle wird angezeigt, wie viele Caches insgesamt als gefunden geloggt wurden. Um eine neue Suche zu starten, ist einfach auf *"New Search"* unterhalb der Tabelle zu klicken.

Die Ergebnistabelle ist nach dem Logdatum aufsteigend sortiert. Aber sie kann auch nach jeder beliebig anderen Spalte sortiert werden. Dazu ist einfach auf die jeweilige Spaltenüberschrift zu klicken. Um zwischen absteigender und aufsteigender Sortierung zu wechseln muss einfach nur nochmals auf die Spaltenüberschrift geklickt werden.

Damit das Ändern der Sortierung der Tabelle funktioniert muss im Browser JavaScript aktiviert sein, was aber standardmäßig bei allen Browsern der Fall ist.

Wenn keine weitere Such durchgeführt werden soll kann der Browser (oder der Tab im Browser) einfach geschlossen werden. Anschließend wechselt man noch ins Terminalfenster, in dem pygcbrowse gestartet wurde und beendet den lokal laufenden Webserver mit dem gleichzeitigen Druck auf die Tasten *"STRG"* and *"C"*.

2.3.1 Heimatkoordinaten festlegen

pygcbrowse bietet die Möglichkeit nach Caches zu suchen, welche eine bestimmte Distanz von den Heimatkoordinaten haben. Wenn solch eine Suche durchgeführt werden soll, müssen die Heimatkoordinaten zuerst korrekt gesetzt werden.

pygcbrowse liest die Breitengrad (lat) und Längengrad (lon) der Heimatkoordinaten aus der Datei **conf.ini**, welche sich im gleichen Verzeichnis wie die Datei **pygcbrowse.py** befindet. Um die Heimatkoordinaten festzulegen öffnet man die Datei **conf.ini** mit einem Texteditor (z.B. notepad auf Windows-basierten Systemen) und trägt als Wert für lat und lon die entsprechenden Zahlen ein, wie z.B.:

```
lat = 50.12345
lon = 7.98765
```

was N 50.12345 E 007.98765 entspricht.

Die Koordinaten müssen wie oben gezeigt in Dezimalgrad angegeben werden, ohne führendes N, S, E oder W. Süd und West Koordinaten muss also entsprechend ein Minus - vorangestellt werden z.B.:

```
lat = -99.16721
lon = -85.01023
```

für S 99.16721 W 85.01023

Wenn nicht nach der Entfernung gesucht werden soll muss in der Datei nichts eingetragen werden. Die Vorgabewert für lat und lon sind Null (0). Falls ein ungültiger Wert für die Koordinaten hinterlegt wird gibt pygcbrowse eine Warnmeldung im Terminal aus und setzt stattdessen Null (0) als Wert fest.

Zum Abschluss noch eine Anmerkung zum Umrechnen von Koordinaten in Dezimal Grad: das, zumindest beim Geocaching gängigere, Format ist Grad plus Dezimal Minuten. Diese Darstellung in Dezimal Grad umzurechnen ist jedoch einfach. Dazu teil man einfach die Dezimal Minuten durch 60 und erhält so die Nachkommastellen.

Ein Beispiel: Koordinaten wie z.B. N 50°25.123 können wie folgt in Dezimal Grad umgerechnet werden: 50 + (25.123/60) = 50.418716667 . Wie oben bereits erwähnt muss Süd- und West-Koordinaten zusätzlich ein Minuszeichen - vorangestellt werden, um die korrekte Darstellung zu erreichen.

Zu guter Letzt sei noch erwähnt, dass pygcbrowser die Entfernung in Kilometer berechnet (und nicht in Meilen).

CHAPTER 3

Tech Talk

This section of the documentation gives a more in-depth view, on how pygcbrowse works in the background.

3.1 How pygcbrowse works

As said before, pygcbrowse is written fully in Python. All scripts and modules use only modules which are shipped with a standard installation of Python 3.x (or above). Namely, those modules are argparse, configparse, xml.etree.ElementTree and sqlite.

But how does pygcbrowse work "under the hood"? Well, it's fairly simple. First, the input file, a XML-type GPX file, is parsed by ElementTree. A selection of attributes per found cache are stored in a dictionary, each dictionary is put in a list, which, once the parsing is done, holds information on all found caches.

After that, this list is used as a source to fill an in-memory SQLite database, having one row for each found cache.

When the database is populated with data, a series of SQL "SELECT" commands are run against the database, grabbing all necessary data to generated the statistics. This data is stored in another result dictionary. Once that is done, this result dictionary is passed to a formatter, generating the requested statistics.

3.2 How the HTML-based user-interface works technically

This is a brief description on how the HTML-based user-interface for pygcbrowse is served and how (and why) the page appears in your local browser. It's slightly technical, but (hopefully) not to much.

First, and probably most important, of all: no extra software of whatever type is installed for this. pygcbrowse.py makes use of Python modules only which come along with the standard installation of Python - which you need anyway to run pygcbrowse.py and pygcbrowse.py.

Python comes with a simple, yet fully functional web server, which is used for pygcbrowse.py, too. The server runs locally only, no data is transferred to wherever. It runs on the IP-address *localhost* (=127.0.0.1), which is your local machine. It is not accessible from the outside (as long as you did not make some very funny or strange settings on your internet router...). The server serves, after being started from within the file pygcbrowse.py automatically, a single HTML file called "index.html", which shows the dialogue for generating the statistics. The statistics are generated "on the fly" in the background and the corresponding HTML page is generated dynamically and, once it's ready, displayed in your browser. All local, no network access needed.

There's only one potential corner stone you should at least be aware of: by design, the built-in web server from Python exposes all files within its directory and all subdirectories via the server. They are not necessarily obviously visible, but accessible (if you know how). Anyway, this is not problem at all, as the program pygcbrowse.py is located within the pygcbrowse directory, which does contain only files used from pygcbrowse.

Thus, there are two things you should probably avoid - as long as you don't exactly know what you are doing:

1. Don't copy any other files beside the GPX-file with your finds into the pygcbrowse directory.

2. Don't move the file pygcbrowse.py to a higher / top level directory. Just keep it within its pygcbrowse directory. That's where it is supposed to be.

And, of course, don't access the Python code of pygcbrowse.py and change the address parameter of the Python web server to another, public address. Well, probably you won't do anyway... but it's better to mention here :-)

Last point: In case you ask yourself how pygcbrowse.py open your browser magically and displays the corresponding page for generating the statistics: there is a Python module included within Python which provides the functionality. Thus, no "magic", dirty hacks or evil code.