`mbib` Documentation

This is only a rough outline for now. I'm working on it.

`mbib` is a literature manager, with capabilities similar to `JabRef`, but intended to better cope with large databases. Key features and limitations are:

- Import from and export to BibTeX
- Import records from DOI and PubMed identifiers
- Push citations to Texmaker/TexStudio
- Push citations to OpenOffice/LibreOffice (collectively referred to as "`OOo`" below). Formatting of bibliographies in `OOo` piggybacks on `JabRef`, that is, you need to have `JabRef` installed in order to fully use `mbib` with `OOo`.
- Written in Python3
- Console-based GUI—basic yet responsive; based on the `urwid` and `urwidtrees` Python libraries.
- Data are stored in a SQLite database. Therefore, starting the program does *not* load the entire database into memory. The database has a fairy simple structure that lends itself to direct querying and manipulation with SQL.
- Developed and tested only on Linux, and will likely not work out of the box on other platforms. (Volunteers for porting it to other platforms are welcome.)

1 Preliminary notes

1.1 Motivation

I started writing `mbib` after my previous literature manager (`bibus`) broke down because of growing incompatibilities with LibreOffice, wxPython etc. I tried using `JabRef` for a while, and while it's really pretty good in many ways, it really slows down once you have several thousand references in your database. (The same goes for Zotero and Mendeley.)

1.2 Status

At present, `mbib` is alpha software. Incompatible changes might still happen to the code and the database structure. However, in the latter case, I will provide a script for migrating the database to the new format (since I will have to migrate my own data anyway).

The program was written with my own needs in mind; I work in biochemistry and use PubMed as my main online literature database and therefore have implemented import of references via PubMed identifiers. People in other fields might miss tighter integration with other databases. I'm open to adding support for those, but unless prodded, it won't be a priority. Similarly, some other bits of functionality are tailored to my own personal preferences and may seem a little idiosyncratic to others.

The code violates all manner of software engineering gospel. There are no unit tests, and I don't plan to add them; the doc strings are a bit spotty and not formatted for automatic conversion into API docs. That said, I will give an overview of the program structure below, which hopefully will help you find your way through the code.

2 Installation

In the following, I am going to describe how things work on *my* system. I am running Debian with a KDE desktop. I don't suppose there will be any major differences with other Linux distros or window managers, but I am not going to verify this by experiment. If you manage to get it to work on other systems and have some specific tricks to share, please let me know, and I will include them here.

2.1 Prerequisites

In order to run `mbib`, you first need to install these programs and libraries:

- Bash
- Python3
- SQLite
- The `urwid` and `urwidtrees` libraries for Python3
- If you intend to use `mbib` with `OOo`, you will also need `JabRef`
- If you want to copy items to the X clipboard, you will need `xclip`
- For viewing or emailing PDF files, `mbib` relies on `xdg-open` and `xdg-email`

On Debian, all of these prerequisites can be installed through the system's package manager. A copy of SQLite already comes as part of the standard library when you install Python3, but you may also want to install the `sqlite3` package, which provides the command line client that lets you run SQL statements on your database.

The `xdg-open` and `xdg-email` utilities are probably installed by default on any graphical Linux desktop; in Debian, they reside in the `xdg-utils` package.

2.2 Installing `mbib`

Just clone the repository and add the main directory (mbib) to your bash $PATH.

2.3 Configuration

The first program start will generate a configuration `mbib.ini` file in your home directory. The settings in this file are explained in comments.

3 Running `mbib`

The program runs inside a console window. Assuming you have put a start-up script into your shell's path, just open a shell window (e.g. konsole on KDE, or xterm on any X-based desktop) and run `mbib.sh`.

3.1 The user interface

- navigation (mouse, keyboard)
- viewing and editing references
- importing references
- moving things around (using selections)

- searching and filtering
- using the clipboard (requires xclip)
-

Some gotchas to point out

- Item selections are stored in the database. Maybe we should offer a configuration option to deselect everything, just as we do with the “recently added” folder. Yes, we do that now.
- Moving selected items from search results: the search folder is basically a simple folder in the database. So, if items in this folder are selected and moved, the copies in the permanent folder stay where they are.

  We may think about an option to erase all other copies of a reference. Yes, we have that now. Or all selected references – we don’t have that yet. If we did have it, I guess we should offer a confirmation dialog that specifies the number of selected references.
- And while we are on it: All the special top-level folders are special just in mbib, because specific operations are available and others excluded; in particular, they cannot be deleted, moved, or renamed. However, no such protective restrictions apply when working with the database in SQL mode.

  Generally speaking, it is a good idea to make a safety backup copy of the database file before performing major surgery in SQL.