This is the **PIM Cataloguer Tool** Instruction Guide.

unedited draft #1.

[created and named to honor Vanessa Reyes & Anastasia Weigle]

gb@bix.digital; benoit@fas.harvard.edu - ©June 28, 2022.

About the project:

The differences between pre- and post-coördinate systems affect quite materially our access to materials - be they personal images and videos or on a different scale of society in museums, archives, and libraries. Web search engines (Google, Bing, DuckDuckGo, and others) are part of the post-coordinate realm but these groups manipulate the relevancy ranking, favoring paid companies. Libraries, Museums, and Archives staff, on the other hand, want to provide the right materials to the person but can't - arguably because of a lack of knowledge of contemporary basic computing and because management actively crushes innovation - yet these groups do nothing but complain without taking action. [imho]

What if we could personalize descriptors - for a personal catalogue and webpage/json/ead/other output - or integrate a tool that allows volunteers or others in a hurry to create at least a record that (a) ascribes descriptors that make sense to the end-user, (b) can be mapped to LMA standard descriptive vocabularies, (c) can support post-coordinate searching of abstracts, (d) generate .xml and .json files for further professional development and incorporation into existing info systems … all while being portable to any Mac or Windows using *free* software tools and libraries, like python, MySQL, html, etc.? This version is not the best coding (any good programmer would grown if reading the source code) - but it does demonstrate a proof-of-concept, demos the inclusion of an EAD plug-in, anticipation of various outputs ("RWD" is the default in the drop down box in the GUI) and has a VRA (visual resources association hook built-in), all with OpenSource (read "free" materials) and documentation.

Give it a spin - not perfect by far - but undeniably this python script addresses long-held needs. And places real power over the creation and distribution of standards-based records in the hands of people who need, create, and want them.

In this first release version, some setup is required on the user's part - but it's not Herculean - but rather what's expected among a computer user today. Today people don't [usually] rely on a chauffeur to fill the car with gas and drive it; we fill it and drive it. So it is with contemporary computing - it's all there for free - so fill 'er up and take 'er for a drive![1]

In this document

The computing and use scenarios are described. Read the entire document to get a feeling for using it. Details matter so pay close attention to choices when saving data, outputting data, etc.

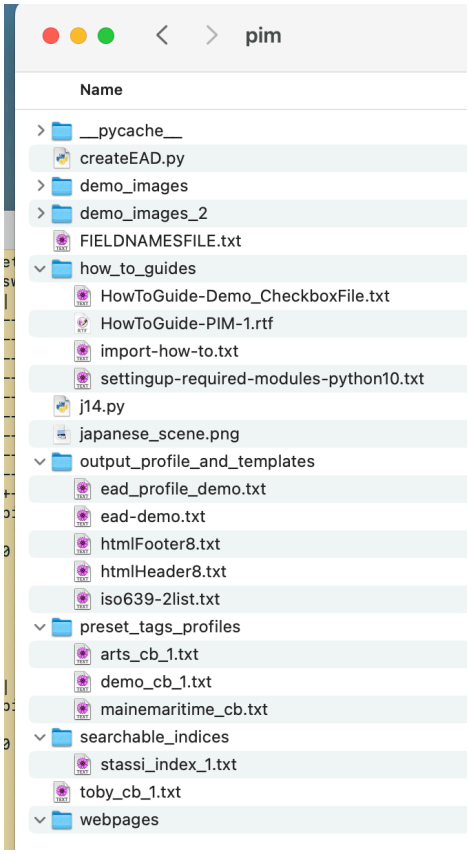
NB: The EAD output version has *not* been tested yet for "flat-file searches", only for RDBMS. Next version.

We start with the big picture - where the files go, required files, how to make your own tags for classifications, use the configurations window to expose the technical issues, but not require hidden tech skills to use, etc.

When all required computer technologies have been installed on your computer, just start the script by starting a Mac/Unix terminal window [and possibly Windows, too, depending on your OS], change the working directory to "pim" [in Unix, type cd Documents/pim if that's where you've stored the demo] and type `python j14.py` and press enter, e.g.:

```
gb@MainMac ~ % cd Documents/pim
gb@MainMac pim % python j14.py
```

1. The biggest jerk I ever met was the head of the Northeastern Univ. Library system. Astonished that I taught a course in XML he demanded to know why. "That's not librarianship - that's what the IT Dept is for!" Similarly the head at UCLA couldn't understand why librarians, archivist, museum specialists should know about computing at all - "Librarians should stick to the work of librarianship" was his response when asked about information systems and computing. Managers, lack of education, lack of professionalism, and who knows what else is why the practice fails the theory. Silence really does equal death.



This is the **pim** folder. Notice the folder and file names.

For end-users to read about the project: there's the **how_to_guides** that includes this guide, how to set up required modules (a technical point, clarified below), and the HowToGuide-Demo_CheckboxFile.txt. This file shows how to create your required checkbox (list of tags) file.

output_profile_and_templates folder holds the header/footer files used in creating the responsive web design (rwd) html output - the script has options for more to be added.

preset_tags_profiles is where you store your personal set of tags. There's a configuration window where you select the profile you want and other technical requires are auto-filled. Here there's an arts_cb_1.txt (for VRA kind of records); a demo_cb_1.txt (a kind of general personal family and friends tagging); mainemaritime_cb.txt for Stassi as a demo of domain-specific tags.

searchable_indices holds flat files if you choose to save your data in a searchable flat file (tab delimited); naturally the MySQL files are not in this folder.

webpages folder holds the output of searching/creating .html pages. **EAD** files are just stored in this pim folder for demo.

j14.py and **createEAD.py** are the programs. j14.py is the actual script; the createEAD demos adding your own plugins - here will generate seemingly well-formed EAD combining the data from the RDBMS and ead_profile_demo.txt as a template.

Setting up your computer for the script:

MySQL: In general if you have lots of images/media files, consider using MySQL8. Install it (detailed below). Before running the PIM tool, be sure to start the MySQL instance before running the PIM script. You'll have to create a user name, password, and database name … then give yourself permissions to use. If you don't have or want to use MySQL then records will be stored in a tab-delimited flat file, that's also searchable.

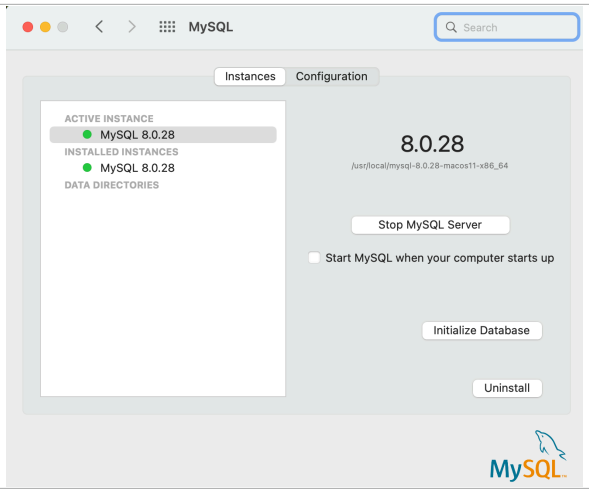
Python 3.8 or later: You need python version 3.8 or later. Likely you have version 2 already installed; download and update to 3.8 or later.

Code Libraries: There are a variety of imported libraries to be included. They appear below and loading them is detailed in the Appendix. Some are built into Python; some must be downloaded and installed on your system, using `pip` or homebrew … The project used `pip` to download. Installing python and these libraries depends on your operating system. Look up the download page for the various libraries if they're not installed and follow the installation instructions for Windows or Mac.

Terminal window: For Mac update/download MySQL 8. Comes with an system preference panel for starting MySQL that you turn on before using the PIM script. You may need to update your pip to pip3.

Folders: The project is expected to run in a single folder, e.g., Documents/pim/. The script lets you select any folder on your system for processing.

A **how-to-guide for setting up MySQL for Windows and Mac:** [see the **create_database.html** page in our pim folder; from my SJSU MySQL in Depth course.]



Turn this system panel on and you're ready to use MySQL. If you want to check out the SQL database and tables (on Mac/Unix) usually head to this in your terminal window: `/usr/local/mysql` and then start the mysql instance using `./bin/mysql –u root –p`
(where -u root means 'user name is root' and -p is a password challenge; use the password you created for MySQL when installing/updating.

Windows Users: You'll want the MySQL Shell. DL and Install. Then look for the "Services" panel on your Windows machine. There you'll find an option to turn on (and turn off) MySQL8.0 Once you've turned on the Service, you'll switch to the Shell in order to create users. [Appendix will have the how-to guide.]

Turn this system panel on and you're ready to use MySQL. If you want to check out the SQL database and tables (on Mac/Unix) usually head to /usr/local/mysql and then start the mysql instance using ./bin/mysql -u root -p
(where -u root means 'user name is root' and -p is a password challenge; use the password you created for MySQL when installing/updating.

Computer Equipment:

Required:
Python 3.8 or later
Several python modules [in the Appendix]
Sufficient rights on your computer to save, read, and delete files.

Optional but strongly Preferred:
MySQL 8.0 or later

The Main Screen:

The main screen starts with an option to configure the script for adding/searching/outputting records, a folder icon for selecting a directory to be used (the source of your images/media files) for creating records and for searching. Start by selecting the configuration icon; when through select the folder you want to process. Once a folder is selected the back | forward and **Add Record** options become active.

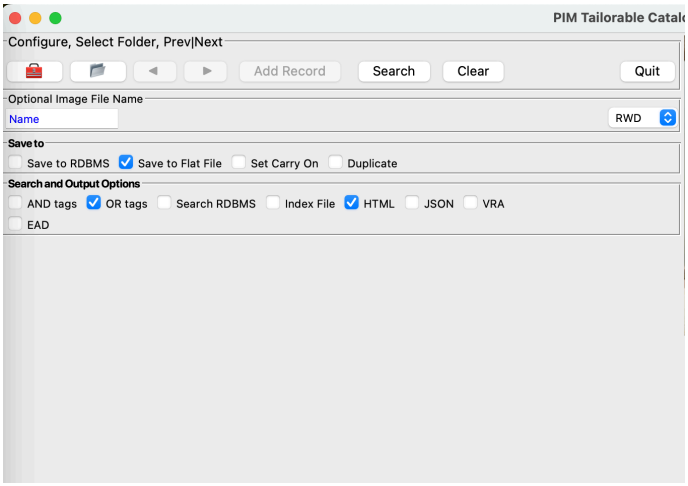
The **Search** button searches for records based on the tags you assign using the checkboxes that will appear after selecting the Checkbox Source File. All these are described in the Configuration section.

The **Clear** button resets all the tag checkboxes. The **Quit** button deletes various temporary .txt files the script creates and exits safely. The default is Save to Flat File (tho should use RDBMS for real experience); OR search and output as HTML.

Default checkboxes: The "Save to" box is set to Save to RDBMS. You can keep that selected. Optionally you can choose also, or alternatively, choose Save to Flat File if you aren't using MySQL. The "Set Carry On" option means "don't uncheck the cataloguing tags checkboxes" after creating a record.

Search and Output Options: By default, searches use "OR" (meaning find records where any condition is met) and use the RDBMS. You can switch to AND (meaning find only records where all conditions are met). If not using MySQL, check the Index File checkbox.

Output: You can output the data to HTML (a preset template for responsive web page) or to a .json file (demo outputs below). Data can be output, too, to EAD, or whatever other formats you need.



No tags appear until you select the configuration option and select the source (e.g., demo_cb_1.txt).

Notice the groupings: in the first row there are the controls for you: configure the session, load the folder of items to be processed; after their selection the < and > and Add Record options become enabled. After selecting the configuration you can search, without selecting a folder.
Line 2: the "name" is an optional text field for adding a name to a file (e.g., Steve's Room); and the drop down box RWD (the default output HTML template for a responsive web design page).
Line 3: "Save to:" are options for saving your tagging to a RDBMS, to a flat file; the "Set Carry On" means don't reset the checkboxes to they can be applied to the next record (useful when there's a series of similar items) and then "Duplicate" - tag a file to be deleted.
Search and Output Options: let you control how you'll search for item. The default is "OR", meaning any item that matches what you'll select in the checkboxes after loading your config preferences; You can search the RDBMS (if you opt for one) or the Flat Index File. The output options are to generate an HTML page, a .json file, VRA or EAD (xml) files. [It's easy to add more options.]

How it Works:

1. Start the script: navigate in your terminal window to the pim folder - probably located in Documents folder, e.g., `cd Documents/pim` and then start the program by invoking python and the name of the script, **j14.py**, e.g., `$ python j14.py`. Press return and the above screen appears. Take a moment to learn about each icon. The screen appears mostly blank until you select your configuration options (shown next). You may be able to just double-click on the icon of the .py file and the script will start - depends on how you have python setup.
2. Choose **Configuration** to set up the files/database login.
3. Choose the **folder** with the files you want to catalogue (from any drive or folder, not just the pim demo).
4. Checkboxes appear in groups after selecting a folder - the FIELDNAMESFILE.txt is created by parsing your required checkbox file - this file is used by the script to create the checkboxes on the screen.
5. **Creating Records:** Check/uncheck the cataloguing tags and press Add Record to save the file to the select storage options (Database or Flat file or Both). Enter optional notes (up to 255 characters) in the text area at the bottom of the screen.
    a. If either the flat file or database table are missing, the script creates automatically a flat file with the name from the config setup and/or database table of the fields in the checkbox list.
6. **Searching Records**:
    a. for searching you do *not* have to load a folder - just click the checkbox categories you want
    b. select the source for searching (database or flat file)
    c. by default the matching records can be viewed on the screen by pressing the forward/back buttons; optionally you can select an html webpage, json, and other formats, like EAD [as a plugin].

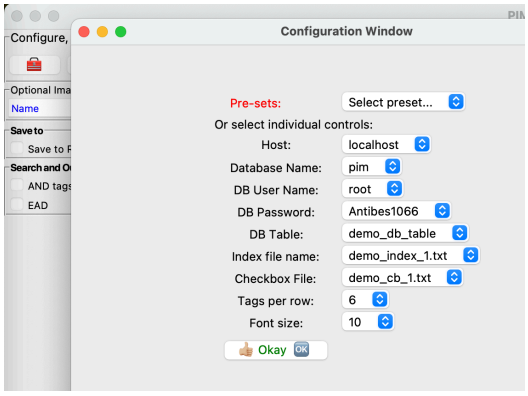
* *** *

Configuration: 🧰

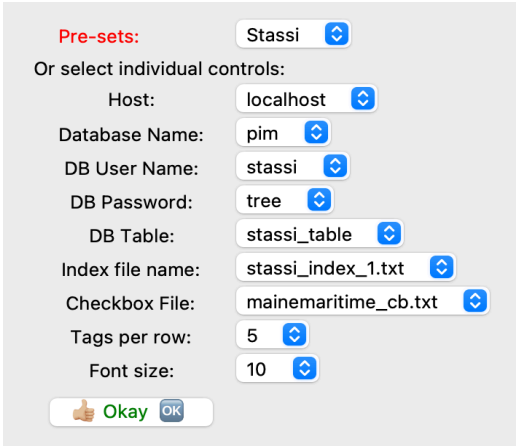
This section describes the workings of the configuration routine. You control the behavior of the program by selecting options in the configuration (config) window.

The config lets you choose (a) preset options for projects or users or (b) select options individually. For example, either leave the "Select preset..." unchanged and select individual options.

Selecting the Configuration Icon causes the Configuration Window to appear. There are a number of demo presets. Based on how large your tag file is (described below) you may want to change the number of tags per row or font size. In the demo of mainemaritime_cb.txt there are > 100 options so in the example so we can set "Tags per row:" to 5 rather than the default 6. But on a larger desktop screen, you can use a *lot* more tags before changing tags/row.



Here the pre-sets for Stassi are selected. In this version of the script, if you choose to use database option, the db must be created first and the user name/password created as demo'd below. The first time run, the script will create the database table. If you opt for the flat file, the first time run the script creates this file. The only *required* file is the Checkbox File. Creating one is described below.



Choosing a preset configuration automatically sets all the config choices.

Configuration Choices:

The script can use either a flat file for storing record data, or a MySQL database, or both. If you don't have MySQL you can select "Save to file."

Database: The script uses the free MySQL software that can be downloaded and installed on any computer. How you start MySQL depends on whether you have Windows OS or Mac/Unix. If you opt to use MySQL start it before running the PIM script.

In the appendix there are instructions for starting MySQL. You should create a database for your project, naming it something you can associate easily with the overall theme, like "Family" or "Architecture". The script will create the table for holding the data for that database. This way a single database can have multiple tables for many different collections.

Flat file Index: If you don't use MySQL or choose to save records to a flat file, the script will create the file based on the name in the config option "Flat file (.txt) index file".

Both DB and flat file records are searchable and can generate various outputs (described below).

Checkbox Source File: is the most important file since it contains all the groups of tags. It's easy to make - just a text file - but has a structure that must be precisely followed.

1. Decide what terms you want as descriptors; group them by related themes.
2. Each set of tag checkboxes has its own title (like 'Family and Friends', 'Ship Types', and so on.
3. There cannot be any duplicate tags. [Because these tags are used to create the database table and tables cannot have duplicate field names.]
4. Using any text editor (like BBEdit, Atom, NotePad) create a single file with this structure. NOTE the structure is absolutely vital and any deviation will cause the program not to work correctly.
   a. The START_GROUP and END_GROUP tags are required and they must be in all-caps.

b. The group of terms (the title for the checkbox) must end in a colon.

c. Don't use any blank lines in the file - neither the top or bottom or between groups. Avoid any special characters (like #!@*% as tag names because these, too, are not permitted by MySQL).

```
HOSTVAR = 0        # the server that hosts the DB instance (default is localhost; can be set to live IP #)
USERVAR = 1        # the database user (you) if you opt for MySQL
DBPASSWORDVAR = 2  # your database password
DBVAR = 3          # DB TO USE – demo uses “pim”
DBTABLEVAR = 4     # DB TABLE FIELD NAMES – created automatically from item in the checkbox file you make
FLATFILEVAR = 5    # SAVE FLAT FILE FOR INDEX
CBVAR = 6          # CHECKBOXES ON SCREEN – used for classifying and for searching
```

Here is an example of the syntax and a brief demo. See the demo file in the **preset_tags_profiles folder**. Fuller examples are in the appendix. The “demo_cb_1.txt” is a kind of personal/family/friends approach; arts_cb_1.txt is meant for visual arts libraries, mainemaritime_cb.txt is a demo of domain- or organization-specific needs.

Syntax:

```
START_GROUP
Name of Group of Checkboxes:
checkbox_1
checkbox_2
...
END_GROUP
START_GROUP
Second Group:
checkbox_1
checkbox_2
...
END_GROUP
```

Brief Example:

```
START_GROUP
Friends:
Albert
Jane
Sierra
Xerxes
END_GROUP
START_GROUP
Animals:
Aardvark
Canines
Elephants
Felines
Zebras
END_GROUP
```

Defaults - Presets:

The presets contain settings for all the config options. Here’s an example showing the position of each item:

```
my_defaults = [webserver, DB username, DB password, DB name, DB Table, File File Index.txt, Checkboxes.txt]
```

e.g.,

```
Toby_defaults = [host_name[0], db_name[2], db_username_name[1], db_password_name[1], db_table_name[2], index_filename_name[2],
checkbox_filename[2] ]
```

Selecting “demo” as a preset automatically updates all the following configurations.

After selecting the configuration, the Checkbox Source File is read. The contents are parsed to separate temp files (all starting with the letters “Tag” and a random number, and sequence numbers, starting at 0, e.g., Tag451_0.txt, Tag451_1.txt and so on. These are erased when pressing the Quit button; they can be erased, too, by hand after the session.

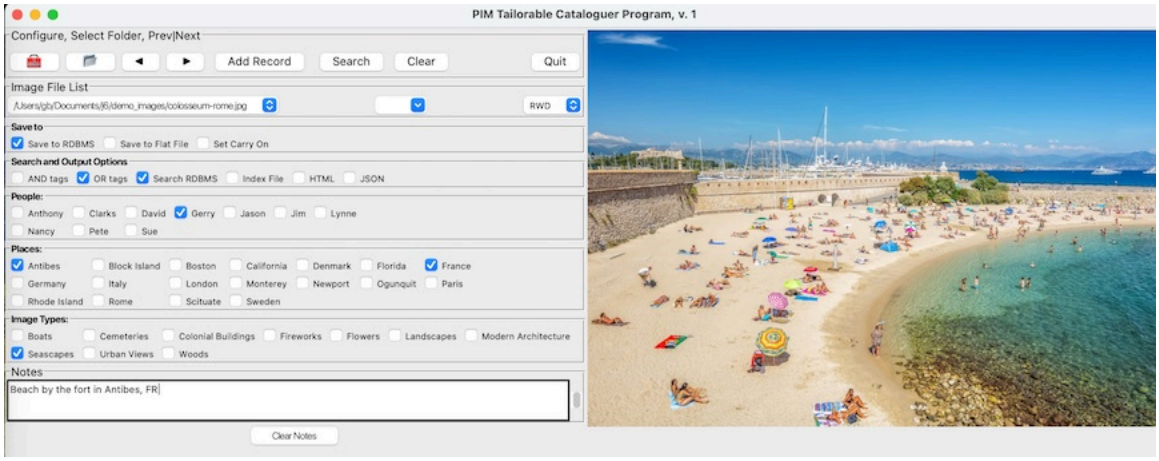
Folder to Process 📁:

After selecting the checkbox file to use, the screen is populated with your tags as checkboxes.

Now you're ready to process files. Select them by clicking the Folder icon to select a folder for processing. The script is set to load only these file extensions (digital images and media types) in a variable called "okFileTypes":

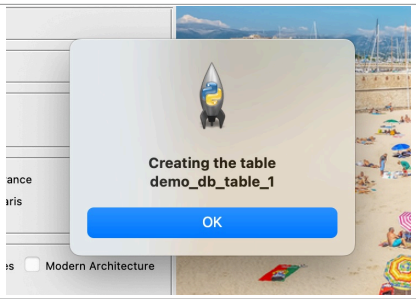
```
okFileTypes =
['.GIF','.gif','.heic','.JPEG','.jpeg','.JPG','.jpg','.mov','.mp3','.mp4','.PNG','.png','.TIF','.tif','.TIFF','
.tiff','.webm','.webp','.wmv']
```

Once a folder is selected, a variable is created holding the full file path name to the image; images and movies are all scaled correctly to appear in the display area; once they're ready, the navigation and Add Record buttons become active. Limit the number of items per folder otherwise the script may freeze because there's not enough RAM to process all the images and scale them [that's a lot of RAM]. **Note** video files may take a moment to load in the viewer and may not clear the screen entirely but they'll start to play - treat them like any other image for cataloguing and searching.

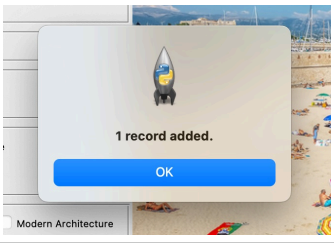


Having selected a folder, press the > arrow to begin to progress through all the images. Here we see the checkboxes have been created and displayed; image appears on the right. Notice the way the checkbox file has created several groups of tags (here, people, places, image types). Cycle thru the images by pressing the < and > buttons *or* select the image by selecting either the image file name or the image number from the drop down boxes. When ready to save a file, press Add Record.

A few checkboxes are selected; here the database is an option; the Save to Flat File will be selected, too. A new db table and a new flat file will be created automatically:



When a record to the database table is created successfully, there's a confirmation screen



Adding Records:

Once you've selected your presets (which determine the checkbox/tags) and selected a folder (containing media and image files), the script rescales all the images to fit the display area on your screen. That may take a few seconds - recommended not more than say 100 files to start off.

Now the Add Records and < > (forward, backwards) buttons and dropdown boxes are populated with data.

Press the > to advance from the placeholder image to your first file. Check the boxes you want to tag.

Saving: Note that the default is to save to a "flat file" - this is the most portable, non-tech way of saving data in a tab-delimited form. If you've installed MySQL and it's active, check the "Save to RDBMS" checkbox.

Once you've clicked the checkboxes that apply and added an optional name and notes, press the Add Record button. You'll have a confirmation that the data were saved to the flat file [if that option is selected] and to the rdbms [if that option is selected]. The "Set Carry On" means don't clear the checkboxes [this is useful when you have lots of files with the same themes and applicable tags]; the Clear button will reset all tags. Keep going 'til the folder is done or you're done.

[I don't recall if I tested switching to a new folder mid-stream; should work.]

Quitting the Script:

You can quit now after adding records. But you can also Search. The search option doesn't require selecting a folder - just the check boxes you want. Although you can close the script just by closing the script's window, it's best to quit by pressing the Quit button because that will cause certain memory-intensive variables to be cleared out and temporary files that the script creates will be erased.

Searching Records:

Search for records as you do for making records. Click on the button tags you want; press Search. Note that you can search RDBMS (default) or the Index File (must be selected). Can select both but that would likely create duplicate records.

The result of the search is automatically loaded as were the contents of selecting the original folder. You can cycle thru the images on the screen by pressing the < and > buttons.

"Duplicates" - this option will be used to remove duplicates - not active yet.

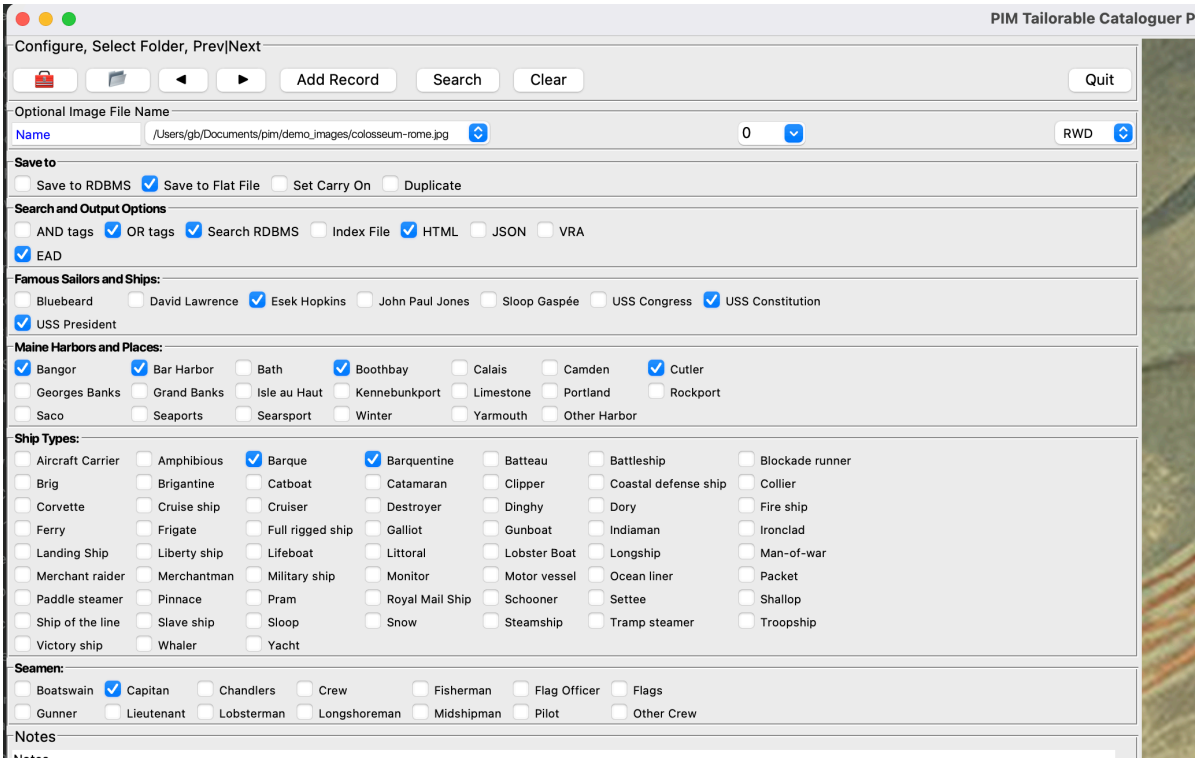
Notice the Optional Image File Name area: there's a text input box if you want to name a file; the drop-down box shows all the files in the selected folder. You can scroll down and select one by name; next to that box is the numeric equivalent (so if you stop working at image #99 you can skip to #100) …. On the far right there's a dropdown box with "RWD" meaning "responsive web design" - the current default (and only active) option for generating web pages from our searches.

Optionally, select an **output**. There's a dropdown box with preset options for web page templates (e.g., RWD is the default for a responsive design, demo'd below). The **HTML**, **JSON,** and **EAD** options output separate files.

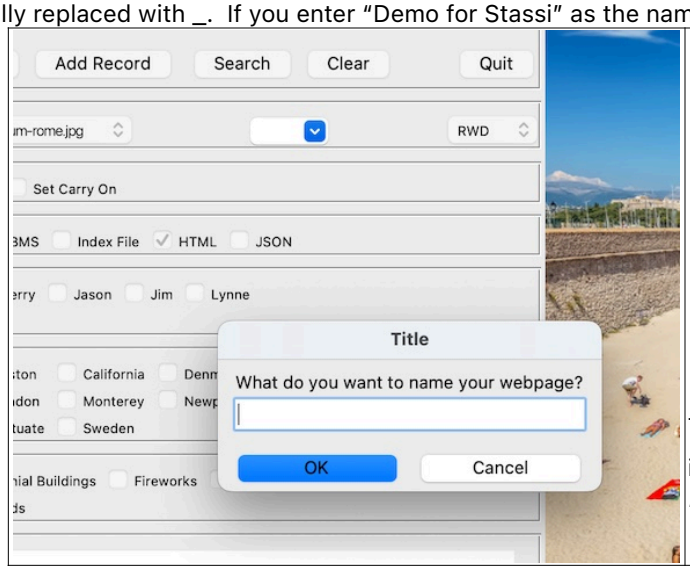
The HTML option uses a header and footer text files that you can change anytime. Useful to changing the CSS [in the header file] and digital rights/contact [in the footer file].

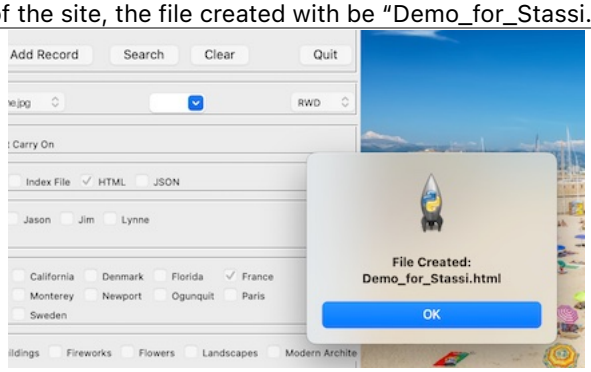
OR and **AND** options: By default searches OR the tags - meaning if any of the selected tags are in the index or database they'll be selected. The AND option means only those records were all selected tags appear. In this example, the HTML option is selected and consequently the user is prompted to name the webpage.

In the following screen capture - notice there are a number of check boxes selected for search (using "OR"). Notice, too, that the results of the search (a) can be viewed in the program using the < > arrows, (b) can be output to HTML (the default using the responsive web-design template RWD), and other options, here EAD.



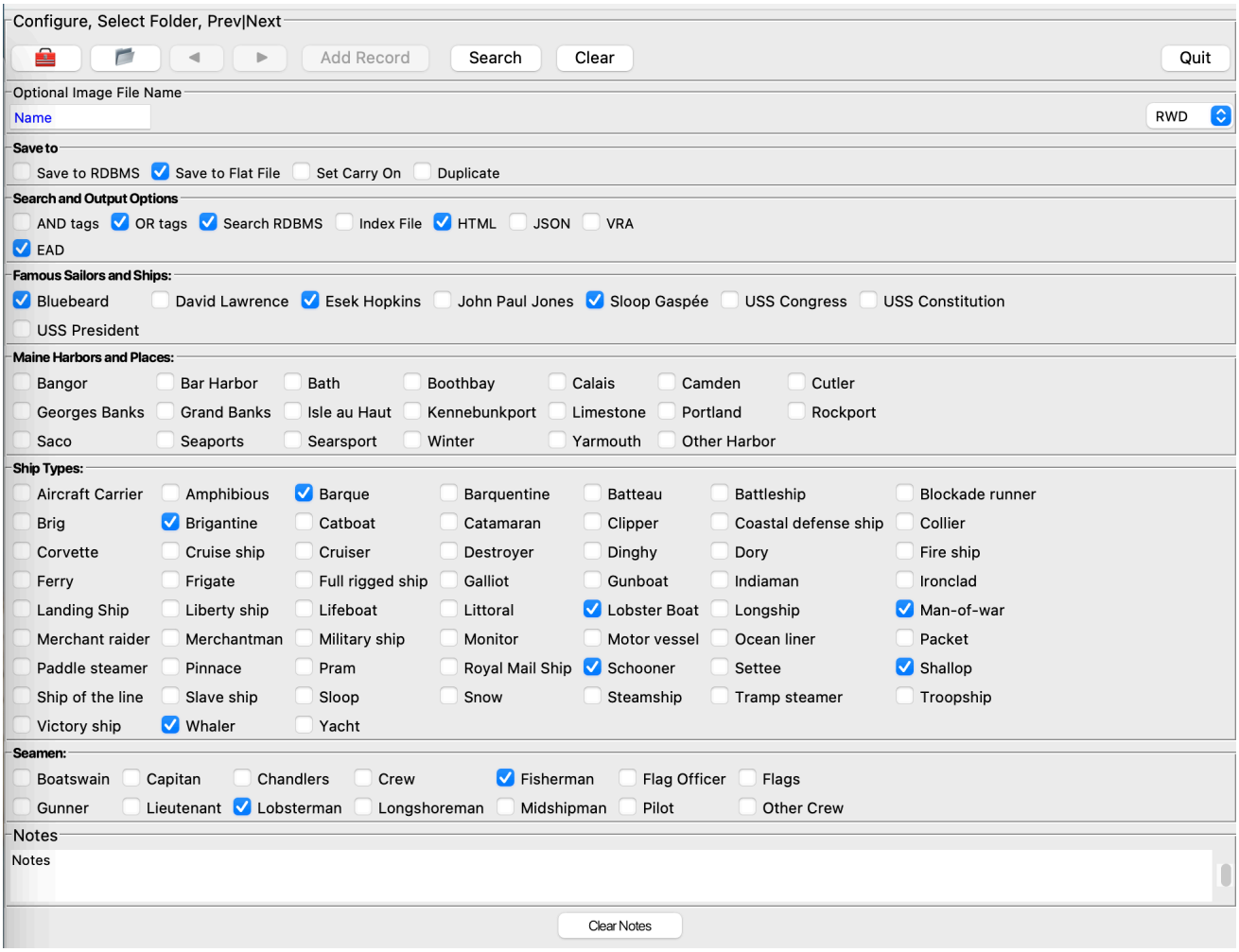
If you select HTML, JSON, EAD, [and other tools you can plug-in], you'll be prompted for a file name. Spaces are automatically replaced with _. If you enter "Demo for Stassi" as the name of the site, the file created with be "Demo_for_Stassi.html"



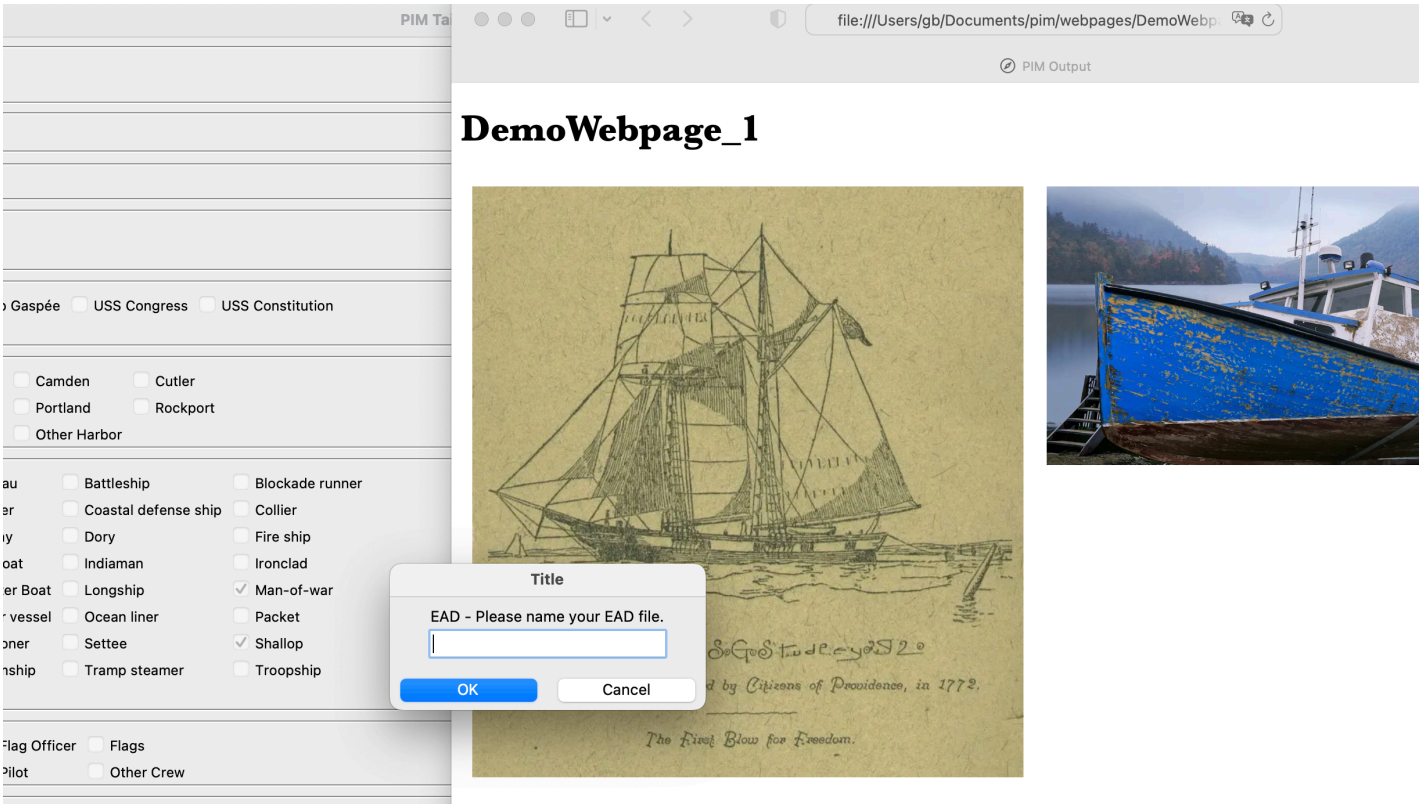


The search is conducted and if there are hits, the web page is automatically created. Right now using the default "RWD"

In this example, I've loaded the Stassi profile and selected tags - notice the Search RDBMS, HTML, and EAD. The result is the DemoWebpage_1.html and the eadDemo1.ead pages. The webpage auto starts a browser (Firefox - if you're Firefox doesn't allow local files you won't see the images - that's a setting on your machine, not a function of the program). Check out these files in the pim folder for details.



In this example, I selected both create HTML and EAD. The HTML page is created first and will load in a browser; then the script will prompt for the name of your EAD file, shown below.



QUIT: the program

You can exit the program just by closing the window. But the best approach is to use the "**Quit**" button option because temporary files that are created for your session will then be deleted and certain variables reset to clear out memory.

Potential Error Messages

- If MySQL 8 isn't turned on and you've selected to save to the DB you'll receive an error.
- If you haven't created your own user and password (as demo'd above) - an error!
- If you're using the wrong version of Python and MySQL 8 there's an authorization error conflict. Let me know.
- Not all files are checked for their existence - they will be - so if you're missing a required file things will go awry.

Appendix:

Setting up your computer for the PIM python script
Creating the database
Sample Checkbox List [mainemaritime_cb.txt]
Sample EAD profile [ead_profile_demo.txt]
Presets in the Script - how to edit

Setting up Python 3.10.x, MySQL8.x, and required Python libraries

Everyone's configuration is different. This document outlines a clean install of all the required libraries. At the end of the directions, there's the output of a setup as a way of showing what might happen during your installation - a lot of ver-biage! Read this entire document before proceeding. All installations in this example use Mac/Unix terminal window.

This demo uses Python 3.10.x and MySQL8.x.

There are several tools used to install the python libraries. One is to visit the homepage of the library (as noted in the documentation) and issuing the "pip" command ... Note that one of the most common hiccups is that the download program (pip) isn't the latest version. In which case you'll update it. Pip will show you how, e.g., `pip install --upgrade pip`

Python 3.10.x

First you'll need the latest Python version. You may find suggestions to use Anaconda or conda - do not use these distributions! Go to https://www.python.org/downloads/macos/ and download and install the latest version. In some cases the original symbolic link on Mac would point any originally-installed version of python, meaning to use python you may have to use `python3` as the command instead of just `python`. [You can update the link to python on your own later if you want.]

Test the installation, if you want, by opening a terminal window (with % as the demo prompt) and typing
`% python --version`

```
gb@MainMac ~ % python --version
Python 3.10.0
```

Remember, if there's more than one version of python on the machine, the latest 3.10.x will likely be python3 --version [but you never know (grin)].

MySQL 8 and MySQL Connect Module

MySQL 8 is a significant change from the previous popular version, 5.6. Follow the instructions on the MySQL page (https://dev.mysql.com/doc/refman/8.0/en/macos-installation-pkg.html).

Be sure to install the MySQL Preference Pane, too, from https://dev.mysql.com/doc/refman/8.0/en/macos-installation-prefpane.html This took makes it very easy to turn on/off MySQL. The MySQL requires a usual download file that then must be unarchived and installed. Just follow the instructions and you'll be fine. ARM or x86? If you're using an M1 chip use ARM version, else the x86 version from the download page.

Using `pip` for the rest of the installations.

MySQL Connection Library

Unlike the rest of the libraries to be describe, the MySQL Connection Library requires a special download from Oracle. You don't have sign-up for it, tho, the webpage makes it look that way.

From this site (https://pypi.org/project/mysql-connector-python/) you'll see the reference to pip. In your terminal window, enter this command: `pip install mysql-connector-python`. Note that you may have to use sudo (super user) mode first - so the command would be `sudo pip install mysql-connector-python`. [Don't type the final period in the examples.] The terminal may then ask for your password and install the software.

For the rest of these installations, you'll use `pip`. [See notes for the output of my installation just fyi.]
In short: confirm python and pip (using % as the Unix prompt).

```
% python -V
% pip list
```

[you may need to upgrade, so use

```
% pip install --upgrade pip
```

Now add the various libraries:

```
pip install tk
pip install tkVideo
pip install pymediainfo
pip install Pillow
pip install mysql-connector-python
```

Setting up your computer for the PIM python script

MySQL: In general if you have lots of images/media files, consider using MySQL8. Install it (detailed below). Before running the PIM tool, be sure to start the MySQL instance before running the PIM script.

Python 3.8 or later: You need python version 3.8 or later. Likely you have version 2 already installed; download and update to 3.8 or later.

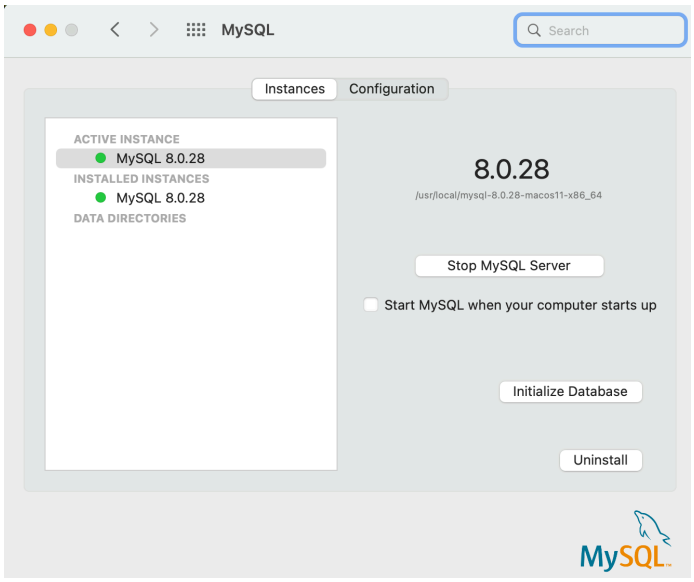
Code Libraries: There are a variety of imported libraries to be included. They appear below. Some are built into Python; some must be downloaded and installed on your system, using pip or homebrew … The project used pip to download. Installing python and these libraries depends on your operating system. Not bothering here with Windows.

Terminal window: For Mac update/download MySQL 8. Comes with an system preference panel for starting MySQL that you turn on before using the PIM script. You may need to update your pip to pip3.

Built-in Python Libraries	GUI libraries: tkinter	Libraries for SQL	Libraries for Images and Videos; output as EAD and automatically start web browser to show .html page

import sys import os import os.path from os import system, name import pathlib import random import math from datetime import datetime from functools import partial import configparser	from tkinter import * from tkvideo import tkvideo import tkinter as tk from tkinter import filedialog from tkinter import font as tkFont from tkinter import messagebox from tkinter import scrolledtext from tkinter import simpledialog from tkinter import ttk from tkinter.messagebox import askokcancel, showinfo, WARNING import tkinter.messagebox as tkmb import tkinter.messagebox as tkmb import webbrowser from pymediainfo import MediaInfo from createEAD import createEAD # THE EAD EXTENSION	import mysql.connector from mysql.connector import Error from mysql.connector import error-code	from PIL import Image, ImageTk from pymediainfo import MediaInfo import webbrowser from createEAD import createEAD
	Some of these can probably be dropped since we're importing tkinter import *; but keep any library using the "as" keyword.	***Download the extra libraries from*** https://pypi.org/project/mysql-connector-python/ https://pypi.org/project/Pillow/ https://pypi.org/project/pymediainfo/	

Folders: The project is expected to run in a single folder, e.g., `Documents/pim/` but you can store it anywhere you want. The script lets you select any folder on your system for processing.



Preferred Option:
Turn this system panel on and you're ready to use MySQL.

If you want to check out the SQL database and tables (on Mac/Unix; and I recommend this) the usual commands are in a Unix terminal window: `/usr/local/mysql` and then start the mysql instance using `./bin/mysql –u root –p` (where -u root means 'user name is root' and -p is a password challenge; use the password you created for MySQL when installing/updating.
When installing MySQL you were probably asked for a password *just for MySQL* - not the same as for your login and operating system. And by default you'll be user "root" - tho you'll want to create a new user, described below, and give that user all the rights over MySQL.

Turn this system panel on and you're ready to use MySQL. If you want to check out the SQL database and tables (on Mac/Unix) usually head to `/usr/local/mysql` and then start the mysql instance using `./bin/mysql –u root –p`

(where -u root means 'user name is root' and -p is a password challenge; use the password you created for MySQL when installing/updating).

Creating the database:

Once you've started MySQL and have a terminal window ready to go (after creating a MySQL session using the commands above, you'll want to create the database with support for Unicode. Replace the "mydatabase" with your own database name. Here we'll create pim as a database that will hold tables.

Syntax:

```
CREATE DATABASE mydatabase CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

Example:

```
CREATE DATABASE pim CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

To use that database:

Once you've created the database, issue the command `USE pim;` (note SQL commands end with a colon). MySQL will respond with a note "Database changed."

The first time you start the program and select your pre-set profile, the table for your data will be automatically created.

NOTE: In the next version the database creation activity will be automatic.

Creating the users:

Using the MySQL Shell or the Terminal Windows, start a MySQL session (see below). For this demo, I'm using the Unix terminal window first to navigate to the mysql installation, start a session as root, enter the root password when prompted by MySQL and then entering these commands to create a new user with full rights over 'pim'. The demo new user is "toby". localhost refers to your own computer; it's a reserved word so just use it.

```
cd /usr/local/mysql
./bin/mysql -u root -p
```

(enter your root password)

Now at the sql prompt `mysql>` type

```
CREATE USER 'toby'@'localhost' IDENTIFIED BY 'toby';
GRANT ALL ON pim.* TO 'toby'@'localhost';
FLUSH PRIVILEGES;
```

What you'll see using a Unix terminal with mysql running:

```
mysql> CREATE USER 'toby'@'localhost' IDENTIFIED BY 'toby';
Query OK, 0 rows affected (0.04 sec)
mysql> GRANT ALL ON pim.* TO 'toby'@'localhost';
Query OK, 0 rows affected (0.01 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

NOTE: if you get a permissions denied ... make sure your options are correct.

```
1044 (42000): Access denied for user 'toby'@'localhost' to database 'toby'
```

Example: Here user Stassi's account is created for MySQL, with a password of "tree". [So be sure to update the preferences demo'd below.]

```
mysql> CREATE USER 'stassi'@'localhost' IDENTIFIED BY 'tree';
Query OK, 0 rows affected (0.02 sec)
mysql> GRANT ALL ON pim.* TO 'stassi'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

In the mysql window, check the privileges using the user name and machine, e.g.,

```
SHOW GRANTS FOR 'stassi'@'localhost';
```

The result should look like this:

```
mysql> SHOW GRANTS FOR 'stassi'@'localhost';
+-----------------------------------------------------------+
| Grants for stassi@localhost                               |
+-----------------------------------------------------------+
| GRANT USAGE ON *.* TO `stassi`@`localhost`                |
| GRANT ALL PRIVILEGES ON `pim`.* TO `stassi`@`localhost`   |
+-----------------------------------------------------------+
2 rows in set (0.01 sec)
```

Checkout the documentation at https://dev.mysql.com/doc/refman/8.0/en/show-grants.html

The whole setup looks like

```
Mysql> CREATE DATABASE pim CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
Query OK, 1 row affected (0.23 sec)
mysql> CREATE USER 'stassi'@'localhost' IDENTIFIED BY 'tree';
Query OK, 0 rows affected (0.55 sec)
mysql> GRANT ALL ON pim.* TO 'stassi'@'localhost';
Query OK, 0 rows affected (0.08 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql>
```

Creating the Tags for Checkboxes:

Notice the structure of this single text file.

1. The very first line is START_GROUP,
2. Next comes the title of the set of tags [ending with a required colon :],
3. then the list whatever tags you want - but no duplicates allowed, and then
4. close the group with END_GROUP [*The phrases START_GROUP and END_GROUP must be in all capital letters.*]

In this example, there are different kinds of tags - Sailors and Ships - limited to a few; Harbors and Places are limited, too, but have an extra Other Harbor tag, just in case. The types of ships is very long - so there's a trade off to balance the tags you want and the screen size. The final set demos narrower and broader terms (Midshipman versus Crew). There cannot be duplicate tag names; do not have blank lines between sets.

```
START_GROUP
Famous Sailors and Ships:
Bluebeard
David Lawrence
Esek Hopkins
John Paul Jones
Sloop Gaspée
USS Congress
USS Constitution
```

```
USS President
END_GROUP
START_GROUP
Maine Harbors and Places:
```

… and so on. See the HowToGuide-Demo_CheckboxFile.txt file for the complete example.

EAD tag output

If you want to add EAD output, the script follows a guide from the Minnesota Archives as a kind of minimum version. There's a demo output file in the output_profile_and_templates/ead_profile_demo.txt folder.

You, the user, tailor the EAD by providing a single .txt file, whose contents are integrated into the output, along with the records data.

The data must be in this order, each on a single line. **Note** that multiple languages are supported - they're listed following ISO639-2 code (see the iso639-2list.txt for all the languages and their full names) - separated by a single space. Don't use a comma or a dash - only a space, e.g., eng fre spa ger (for English French Spanish German).

```
MAINAGENCYCODE = 0
TITLE_PROPER_ORG_OR_PERSON = 1
SUBTITLE = 2
AUTHOR = 3
SPONSOR = 4
PUBLISHER = 5
ADDRESSLINE = 6
NAME_OF_COLLECTION = 7
ENCODED_BY = 8
PUBLISHER  = 9
LANGUAGES = 10
PREFER_CITE = 11
PHYSICAL_LOCATION_YOUR_URL = 12
YOUR_CONTACT_URL = 13
```

Here's an example of what your file might look like:

```
MSA100
Maine Maritime Museum Collection
Lobstermen and Flags
Stassi H Γυναίκα του Δάσους
State of Maine
MMM Collection
5 Smith Street, Portland, ME
Seagoing Adventures Collection
Lars Ingersol
Published by Maine Maritime Consortium
eng fre spa
2022 MMM Consortium
https://www.mmc.org/collections/ourstuff.html
lars@mmc.org
```

Defaults and Required Text files used in the script - that you can edit.

In the script there are a number of required fields - lists of files to use, user names, etc., they're listed below. At the moment you can edit them by using a text editor to edit them directly in the .py file. I suggest copying the original first in case you want to roll back to it. **Note: in this version** of the script anyone can create create their own checkbox/tag file. use the same name as the default for your pre-set. Will automate integrating new pre-sets in the next release.

The "`user_defaults`" list is a set of preset defaults for projects or users - just select the one you want and the specifics [host name, database name, database user name, database password, database table name, index file and the all-

important checkbox file name are selected. Or leave the user_default on “Select preset…” and you can select individual options from the boxes. For instance, say you have a project at your museum all about ships but now want to create a separate table all about architecture - everything else is the same, tho. Add the new checkbox option to the checkbox_filename list and select it, keeping everything your usual present.

Finally, notice the named preset options use the options (the first part of the below boxes) and references them by their index number, e.g., in the host_name options there’s “localhost” and “127.0.0.1”. The named defaults (Nancy, Toby, Stassi, Vanessa) refer in their list to host_name[0] to show their default is “localhost”. If you wanted to use a different server, say “mymuseum.com”, then add “mymuseum.com” to host_name list (e.g., `host_name = [“localhost”, “127.0.0.1”, “mymuseum.com”]` and update your preferences
(e.g., `Stassi_defaults =[host_name[2], …]`). Adding a preset - just follow the model.
In the source .py file, search for `START OF PRESET DEFAULTS` to find this area in the code and then edit.
Remember: only the checkbox tag file is absolutely required and needs to be stored in the same folder as the script.
Editable Defaults from the script:

user_defaults = ["Select preset...","demo", "Toby","Nancy", "Stassi", "Vanessa"]

```
# individual options.
host_name = ["localhost", "127.0.0.1"]
db_name = ["pim", "nancy", "toby", "stassi", "vanessa"]
db_username_name = ["root", "bix", "nancy", "stassi", "vanessa"]
db_password_name = ["Antibes1066", "Fishlips", "nancy_pw", "tree", "vanessa_pw"]
db_table_name = ["demo_db_table", "nancy_table", "toby_table", "stassi_table","vanessa_table"]
index_filename_name = ["demo_index_1.txt", "nancy_index_1.txt", "toby_index_1.txt","stassi_index_1.txt","vanes-
sa_index_1.txt"]
checkbox_filename = ["demo_cb_1.txt", "nancy_cb_1.txt", " toby_cb_1.txt","stassi_cb_1.txt","vanessa_cb_1.txt"]
```

#default options: we start counting at 0, so “host_name[0]” would mean “localhost”
in order: host, db, dbuser, dbpassword, dbtable, index, checkbox
in index of the above "individual options" - vital.

```
demo_defaults = [ host_name[0], db_name[0], db_username_name[0], db_password_name[0], db_table_name[0], index_-
filename_name[0], checkbox_filename[0] ]
Nancy_defaults = [host_name[0], db_name[1], db_username_name[2], db_password_name[2], db_table_name[1], index_-
filename_name[1], checkbox_filename[1] ]
Toby_defaults = [host_name[0], db_name[2], db_username_name[0], db_password_name[0], db_table_name[2], index_-
filename_name[2], checkbox_filename[2] ]
Stassi_defaults = [host_name[0], db_name[0], db_username_name[3], db_password_name[3], db_table_name[3], in-
dex_filename_name[3], checkbox_filename[3] ]
Vanessa_defaults = [host_name[0], db_name[0], db_username_name[3], db_password_name[4], db_table_name[4],
index_filename_name[4], checkbox_filename[4] ]
```

Example:

Say I want to change a checkbox_filename from “stassi_cb_1.txt” to “mainemaritime_cb.txt”, either replace the former with latter, or add the new entry to the end of list and change the checkbox_filename[3]

adding an optional checkbox file

`checkbox_filename = ["demo_cb_1.txt", "nancy_cb_1.txt", " toby_cb_1.txt","stassi_cb_1.txt","vanes-sa_cb_1.txt", "`**`mainemaritime_cb.txt`**`"]`

and then you can select that new file for tags …

OR change the checkbox_filename[] default for Stassi from 3 to 5. [Start counting from 0, not 1.]

`Stassi_defaults = [host_name[0], db_name[0], db_username_name[3], db_password_name[3], db_table_-name[3], index_filename_name[3], `**`checkbox_filename[5]`**

Finally, you can load new presets by updating the check() function. [Yes, this should be updated to use python's new match command but I'll do that in the next version.

```
def check(*args):
        # user choice of pre-sets:
        #print(f"variable changed to '{variable.get()}'")
        # THIS SHOULD BE UPDATED FOR python 10.x WITH MATCH statement ...
        preset_name = variable.get()
        if preset_name == user_defaults[1]: # = "demo":
                host_var.set(demo_defaults[HOSTNAME])
                db_var.set(demo_defaults[DBNAME])
                db_username_var.set(demo_defaults[DBUSERNAME])
                db_password_var.set(demo_defaults[DBPASSWORDNAME])
                db_table_name_var.set(demo_defaults[DBTABLENAME])
                index_filename_name_var.set(demo_defaults[INDEXFILENAME])
                checkbox_filename_var.set(demo_defaults[CHECKBOXFILENAME])
                # set up the rest here...
        elif preset_name == user_defaults[TOBY]:
                host_var.set(Toby_defaults[HOSTNAME])
                db_var.set(Toby_defaults[DBNAME])
                db_username_var.set(Toby_defaults[DBUSERNAME])
                db_password_var.set(Toby_defaults[DBPASSWORDNAME])
                db_table_name_var.set(Toby_defaults[DBTABLENAME])
                index_filename_name_var.set(Toby_defaults[INDEXFILENAME])
                checkbox_filename_var.set(Toby_defaults[CHECKBOXFILENAME])
        elif preset_name == user_defaults[NANCY]:
                host_var.set(Nancy_defaults[HOSTNAME])
                db_var.set(Nancy_defaults[DBNAME])
                db_username_var.set(Nancy_defaults[DBUSERNAME])
                db_password_var.set(Nancy_defaults[DBPASSWORDNAME])
                db_table_name_var.set(Nancy_defaults[DBTABLENAME])
                index_filename_name_var.set(Nancy_defaults[INDEXFILENAME])
                checkbox_filename_var.set(Nancy_defaults[CHECKBOXFILENAME])
        elif preset_name == user_defaults[STASSI]:
                host_var.set(Stassi_defaults[HOSTNAME])
                db_var.set(Stassi_defaults[DBNAME])
                db_username_var.set(Stassi_defaults[DBUSERNAME])
                db_password_var.set(Stassi_defaults[DBPASSWORDNAME])
                db_table_name_var.set(Stassi_defaults[DBTABLENAME])
                index_filename_name_var.set(Stassi_defaults[INDEXFILENAME])
                checkbox_filename_var.set(Stassi_defaults[CHECKBOXFILENAME])
        elif preset_name == user_defaults[VANESSA]:
                host_var.set(Vanessa_defaults[HOSTNAME])
                db_var.set(Vanessa_defaults[DBNAME])
                db_username_var.set(Vanessa_defaults[DBUSERNAME])
                db_password_var.set(Vanessa_defaults[DBPASSWORDNAME])
                db_table_name_var.set(Vanessa_defaults[DBTABLENAME])
                index_filename_name_var.set(Vanessa_defaults[INDEXFILENAME])
                checkbox_filename_var.set(Vanessa_defaults[CHECKBOXFILENAME])
        else: # none
                host_var.get()
                db_var.get()
                db_username_var.get()
                db_password_var.get()
                db_table_name_var.get()
                index_filename_name_var.get()
                checkbox_filename_var.get()
```

Extended Archival Descriptors

If you wish to add an EAD record (or anything else like VRA) - there's a template here that can be used for each item created following

http://www2.mnhs.org/library/findaids/CMToolkit/BestPractices/FindingAids_EADTaggingGuidelines.pdf although I just discovered https://eadiva.com/elements/ as a potential template for correct EAD tags.

For this demo there's a set of required EAD tags. Some of the data must be consistent and so require you to create another text file that'll be integrated with this template. Other data will come from your cataloguing/tagging with the PIM script. For instance, [FILENAME], [NOTES], "XXX", and similar data below show the placement of the data in the template.

EAD records that are created by the PIM script can be improved by editing the EAD itself and by additions to the next version, based on your input.

Below is the TEMPLATE; after that appears an output from the PIM script.

```
<ead>
    <eadheader audience="internal" countryencoding="iso3166-1"
    dateencoding="iso8601" langencoding="iso639-2"
    repositoryencoding="iso15511">

    <eadid countrycode="us" mainagencycode="XXXX">
           ###FILE.xml
    </eadid>
    <filedesc>
           <titlestmt>
                  <titleproper>[ORG/PERSON NAME]
                  </titleproper>
                  <subtitle>An inventory of [TITLE] at [ORG]</subtitle>
                  <author>Finding aid prepared by [YOU]</author>
                  <sponsor>[ORG HERE]</sponsor>
           </titlestmt>
           <publicationstmt>
                  <publisher encodinganalog="Publisher">[ORG or YOU]</publlisher>
                  <address>
                         <addressline>[YOURPLACE]</addressline>
                  </address>
           </publicationstmt>

           <seriesstmt>
                  <p>[NAME OF YOUR COLLECTION</p>
           </seriesstmt>
    </filedesc>

    <profiledesc>
           <creation>Finding aid encoded by [NAME]
                  <date>[GEN DATE]</date>
           </creation>
           <langusage>Finding aid written in
                  <language langcode="eng">English</language>, and
                  <language langcode="fre">French</language>
           </langusage>
    </profiledesc>

    <revisiondesc>
           <change>
                  <date></date>
                  <item>[REASON FOR CHANGE]</item>
           </change>
    </revisiondesc>

    </eadheader>
    <archdesc>

    <did>
           <unitid></unitid>
           <repository></repository>
           <origination>
                  <corpnane> </corpname>
           </origination>
           <unittitle> </unittitle>
           <unitdate> </unitdate>
           <langmaterial>
                  <language>
                  </language>
           </langmaterial>
```

```
            <abstract>[NOTES]</abstract>
            <physdesc>Webpage</physdesc>
            <physloc> </physloc>
    </did>

    <descgrp type="admininfo">
            <prefercite></prefercite>
            <acqinfo></acqinfo>
    </descgrp>
    <dsc>
    <c01></c01>
    <did>
            <physloc>[FILENAME]</physloc>
            <container></container>
            <unittitle> </unittitle>
            <unitdate> </unitdate>
            <physdesc> </physdesc>
    </did>

    </dsc>
    </archdesc>

    <controlaccess>
            <subject>
                    <part>[SUBJECT HERE]</part>
            </subject>
    </controlaccess>

    <control>
            <recordid>[RECNO]</recordid>
            <filedesc>
                    <titlestmt>
                            <titleproper>[. some thing ? ]</titleproper>
                    </titlestmt>
                    <editionstmt>
                            <edition>1st</edition>
                            <publicationstmt>
                                    <publisher>YOU</publisher>
                            <publicationstmt>
            </filedesc>

            <notestmt>
            <controlnote>
                    <p>Contact information:
                            <ref show="new" actuate="onrequest" href="http://hdl.loc.gov/loc.mss/mss.contact">
http://hdl.loc.gov/loc.mss/mss.contact
                            </ref>
                    </p>
            </controlnote>
            <controlnote>
                    <p>Catalog Record:
                            <ref href="" actuate="onrequest"
                            linktitle="MARC record for collection"></ref>
                    </p>
            </controlnote>
            </notestmt>

</ead>
```

Appendix of terminal outputs, just fyi, to see what you should expect.

This is the transcript of installing fresh all the libraries. Commands you should note and execute are in **bold**. The first few commands confirm python and its version and what pip has already loaded.

```
gb@MainMac ~ % python -V
Python 3.10.0
gb@MainMac ~ % python
Python 3.10.0 (v3.10.0:b494f5935c, Oct  4 2021, 14:59:20) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("roar")
```

```
roar
>>> exit();
gb@MainMac ~ % pip list
Package    Version
---------- ---------
certifi    2021.10.8
pip        22.0.4
PyMySQL    1.0.2
setuptools 58.1.0
WARNING: You are using pip version 22.0.4; however, version 22.1.2 is available.
You should consider upgrading via the '/usr/local/bin/python3.10 -m pip install --upgrade pip' command.
gb@MainMac ~ % pip install --upgrade pip
Requirement already satisfied: pip in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (22.0.4)
Collecting pip
  Using cached pip-22.1.2-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.4
    Uninstalling pip-22.0.4:
      Successfully uninstalled pip-22.0.4
Successfully installed pip-22.1.2
gb@MainMac ~ % pip list
Package    Version
---------- ---------
certifi    2021.10.8
pip        22.1.2
PyMySQL    1.0.2
setuptools 58.1.0
gb@MainMac ~ % pip --version
pip 22.1.2 from /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/pip (python 3.10)
gb@MainMac ~ % pip install tk
Collecting tk
  Using cached tk-0.1.0-py3-none-any.whl (3.9 kB)
Installing collected packages: tk
Successfully installed tk-0.1.0
gb@MainMac ~ % pip install tkVideo
Collecting tkVideo
  Using cached tkVideo-0.1-py3-none-any.whl (4.7 kB)
Collecting imageio
  Using cached imageio-2.19.3-py3-none-any.whl (3.4 MB)
Collecting pillow
  Using cached Pillow-9.1.1-cp310-cp310-macosx_10_10_x86_64.whl (3.1 MB)
Collecting imageio-ffmpeg
  Using cached imageio_ffmpeg-0.4.7-py3-none-macosx_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (22.5 MB)
Collecting numpy
  Using cached numpy-1.23.0-cp310-cp310-macosx_10_9_x86_64.whl (18.1 MB)
Installing collected packages: pillow, numpy, imageio-ffmpeg, imageio, tkVideo
Successfully installed imageio-2.19.3 imageio-ffmpeg-0.4.7 numpy-1.23.0 pillow-9.1.1 tkVideo-0.1
gb@MainMac ~ % pip install Pillow
Requirement already satisfied: Pillow in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (9.1.1)
gb@MainMac ~ % pip install pymediainfo
Collecting pymediainfo
  Using cached pymediainfo-5.1.0-py3-none-macosx_10_9_x86_64.whl (6.4 MB)
Installing collected packages: pymediainfo
Successfully installed pymediainfo-5.1.0
gb@MainMac ~ % pip install mysql-connector-python
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.29-cp310-cp310-macosx_11_0_x86_64.whl (5.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.0/5.0 MB 4.5 MB/s eta 0:00:00
Collecting protobuf>=3.0.0
  Downloading protobuf-4.21.2-cp37-abi3-macosx_10_9_universal2.whl (483 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 483.1/483.1 kB 4.2 MB/s eta 0:00:00
Installing collected packages: protobuf, mysql-connector-python
Successfully installed mysql-connector-python-8.0.29 protobuf-4.21.2
gb@MainMac ~ % pip install pillow
Requirement already satisfied: pillow in /Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages (9.1.1)
gb@MainMac ~ %
```

_____ end of the transcript _____

End of the first draft of the PIM how-to guide. Comments welcome.