The GMU Image Manage Documentation

Release 1.0.0

Mybrid Wonderful (The GMU), Gregg Yearwood (The GMU)

Nov 16, 2020

CONTENTS:

`thegmu_imagemanage` is used to manage a directory images using various bulk and indivdiual commands. The bulk commands are listed one per line on a command file. The individual file commands are listed per file inside a spreadsheet containing an image thumbnail and file details.

- Documentation: http://thegmu-imagemanage.readthedocs.org/
- Source Code: https://bitbucket.org/thegmu/thegmu-imagemanage
- Download: https://pypi.python.org/pypi/thegmu-imagemanage

Please feel free to ask questions via email: (mybrid@thegmu.com)

CHAPTER

ONE

THE GMU IMAGE MANAGE

https://www.thegmu.com/

Authors Mybrid Wonderful, Gregg Yearwood

Date 11/14/2020

Support mybrid@thegmu.com

Version 1.0.0

- Documentation: http://thegmu-imagemanage.readthedocs.org/
- Source Code: https://bitbucket.org/thegmu/thegmu-imagemanage

CHAPTER

TWO

INTRODUCTION

The GMU Image Manage is an Almost There Project where Almost There projects have the following distinctions:

1. Open source
2. Text interface
3. Text source
4. Do-it-yourself, or Maker design (DIY, Maker)

Almost There software has no web based or graphical interface because it is not considered a finished product. Also the goal is to get the public engaged into making slight changes to software such as naming things. Almost There projects represent a new market of enabling a much larger audience to modify software ever so slightly to realize big gains. See ALMOSTTHERESOFTWARE for further discussion.

I started this project because file manager software whether in Linux or in Windows only shows either a thumbnail or the file details but never both on a single line. This program does that.

Install this software and it will create a spreadsheet listing each image first as a thumbnail and then second a listing of image details, including width and height.

There are columns such as “delete” in the spreadsheet that one can use to manage files by first running the software to create the spreadsheet and afterwards again to process a spreadsheet that one has edited to delete, rename, and move images.

Bulk actions on all files in the directory are also possible such as removing all duplicate images with different names, removing files with zero size and removing all punctuation from a name so as to make the names more friendly to type within the spreadsheet.

CHAPTER

THREE

RUNNING

Quick reference:

```
cd ~/Pictures
source thegmu-imagemanage/almostthere/bin/activate
thegmu_im.py Pictures.commands.txt
```

3.1 Program

Program: **thegmu_im.py <command file>**

Each time you will need to do the following:

1. Open a command line shell.
2. Change directory to the image directory, example 'cd ~/Pictures'.
3. Activate your Almost There Python environment, 'source thegmu-imagemanage/almostthere/bin/activate'.
4. You will also need to create your first command file if you haven't already.

In addition to the text of the program name then one other piece of text may follow the program name and that is the command file name. If the command file is not found in the image directory the program stops.

3.2 Command File

Default: **thegmu_imagemanage.commands.txt**

You must create a command file before running the program. If you run this Almost There application without a command file then the default command file is assumed. Whether you use the default command file or a specific command file processing stops if the file doesn't exist.

Unlike traditional command line applications the Almost There software projects all use a file that lists one or more commands instead of passing commands on the command line. This do-it-yourself design requires combining many commands in a sequence as lines in a file where that sequence represents a new software application. Almost There developers can share command files like food recipes represent new cooking dishes.

Command files may contain only three kinds of lines:

1. Notes: lines that start with the pound '#' sign are notes and are ignored.

2. Blanks: blank lines with nothing on them that are used for clarity when reading.
3. Commands: lines that are not a note or blank are treated as a command and if the command is not found by the Almost software then an error stops processing command file at that line.

Example:

```
# BEGIN
catalog

remove_empty_files
remove_duplicate_files
convert WEBP JPEG
convert BMP3 JPEG
convert_format_simple PNG JPEG
remove_multiple_format_files
catalog

jpeg_optimize_size 6000

# END
```

You can see in the example that further instructions for a command are provided to the command by instructions listed on the same line separated by spaces. For example, the *convert* command requires two additional instructions listing what format to convert from and convert to. I tend to keep files in JPEG format because JPEG has a feature most formats do not support and that is compressing files to file size and not just shrinking a file by width and height. The 'jpeg_optimize_size 6000" command shrinks any JPEG image to approximately 6000KB, or 6MB.

CHAPTER

FOUR

COMMANDS

Commands start each line in the command file. Commands may or may not take further instructions separated by spaces.

4.1 Catalog

Command: **catalog <catalog file>**

Default: **thegmu_imagemanage.catalog.txt**

The catalog file is used internally for storing information about files. You may refresh this catalog file at any time but it is up to you in your command file to do so. Building the catalog file can take time for directories with thousands of images. Once you get to know commands then you will learn when to use the catalog command to build a fresh catalog to ensure subsequent commands have an up-to-date catalog.

The catalog contains the following fields:

1. **file_name**: i.e. 011mlxq4s6e51.jpg.
2. **format**: Imagemagick format of JPEG, PNG, GIF, etc.
3. **WxH**: width by height, i.e. 1920x1080.
4. **size**: file size in bytes, i.e. 2985169.
5. **date**: created orlast modified to a second, i.e. 2020-07-31-14:45:44.
6. **epoch**: seconds since 1970, i.e. 1596221144.
7. **md5sum**: unique binary identifier used for checking duplicates, i.e. 84aa9f5563d106e0627d3d0a2f4049fe.
8. **ext**: file extension, i.e.jpg.

4.2 Catalog Excel

Command: **catalog_excel <catalog file>**

Output: **thegmu_imagemanage.catalog.2020-11-14.01.xlsx**

The spreadsheet catalog files created are intended for human editing using Excel, Libre Office or other spreadsheet editor. The output file name reflects the requested text catalog file name. The 'txt' file extension is replaced with 'DATE.01.xlsx'. The '01' designation is a count. Only one-thousand images are contained in each spreadsheet file. This means if there exists twenty-thousand images in a directory then there will be twenty different spreadsheet files created and '01', '02, '03', ... files will be created. This approach was chosen because testing of various spreadsheet programs revealed that the performance of Excel and LibreOffice Calc varied greatly after more than a thousand thumbnails were embedded in the file, albeit on one sheet or multiple sheets.

Spreadsheet programs have many ways to sort rows based upon column filters. Sorting images by the various columns is one of the primary uses of the application. This does require one to have a certain degree of familiarity with spreadsheets. This is in keeping with the Almost There philosophy of being a do-it-yourself person. The more you do things on your own the more power you have to express yourself as you are not limited by the choices software designers make. So take the time to learn Excel or LibreOffice Calc and how to filter and sort rows based upon the data in the file. The "auto-filter" feature is a good place to start.

Once you run this command then subsequent commands will stop processing if these files already exist. This is because the assumption is that these files are being updated by you. What I do is create a new document directory in my 'Documents' folder and move these files into that folder if I need to rerun this command.

These spreadsheet files contain columns at the beginning of the sheet not found in the text catalog, most notably a thumbnail of the image. However there are also columns to manage individual files.

Excel catalog columns:

1. **thumbnail**: a 100pixel high thumbnail image.
2. **original**: hyperlink to the original image.
3. **note**: Add your notes here.
4. **delete**: Enter 'delete' in this columnto delete the file.
5. **move**: Enter a "copy" or "move" along with a directory name to copy or move the image to another directory.
6. **rename**: Enter a new file name to rename the file.
7. **tags**: Enter a comma separated list of tags. Tags are used in bulk commands to help organize files.

4.3 Convert

Command: **convert <JPEG,PNG, etc.>**

Deletes the original file. Convert image files from one format to another. See "Convert List Formats" for a listing of all the possible formats available for conversion. The format is not determined by the file name. The format is determined by the ones and zeroes in the file. If the format to be converted from like say with PNG is detected then the original PNG file is deleted after a new file is created with the PNG format using the PNG file extension, 'png'. I standardize on JPEG format for all my images. I run this conversion process quite regularly and it is one of the more used features for me. Get a bunch of new images, convert them all to JPEG.

. .

4.4 Convert Format Simple

Command: **convert_format_simple <PNG,JPEG,etc.>**

See "Convert". This does the same thing as the convert command except the original file is not deleted. I use this in tandem with the remove_multiple_format_files command. The two commands will leave the smallest file size of the two. This is particularly noteworthy for cartoons and graphic files that only have a few colors and compress will with PNG. In that case the PNG file will be favored by the remove_multiple_format_files command.

4.5 Convert List Formats

Command: **convert_list_formats**

Prints a line of text showing all the formats allowed by the convert command:: 3FR 3G2 3GP AAI AI ART ARW AVI AVS BGR BGRA BGRO BIE BMP BMP2 BMP3 BRF CAL CALS CANVAS CAPTION CIN CIP CLIP CMYK CMYKA CR2 CRW CUR CUT DATA DCM DCR DCX DDS DFONT DJVU DNG DOT DPX DXT1 DXT5 EPDF EPI EPS EPS2 EPS3 EPSF EPSI EPT EPT2 EPT3 ERF EXR FAX FILE FITS FRACTAL FTP FTS G3 G4 GIF GIF87 GRADIENT GRAY GRAYA GROUP4 GV H HALD HDR HRZ HTM HTML HTTP HTTPS ICB ICO ICON IIQ INFO INLINE IPL ISOBRL ISOBRL6 JBG JBIG JNG JNX JPE JPEG JPG JPS JSON K25 KDC LABEL M2V M4V MAC MAGICK MAP MASK MAT MATTE MEF MIFF MKV MNG MONO MOV MP4 MPC MPEG MPG MRW MSL MSVG MTV MVG NEF NRW NULL ORF OTB OTF PAL PALM PAM PANGO PATTERN PBM PCD PCDS PCL PCT PCX PDB PDF PDFA PEF PES PFA PFB PFM PGM PGX PICON PICT PIX PJPEG PLASMA PNG PNG00 PNG24 PNG32 PNG48 PNG64 PNG8 PNM PPM PREVIEW PS PS2 PS3 PSB PSD PTIF PWP RAF RAS RAW RGB RGBA RGBO RGF RLA RLE RMF RW2 SCR SCT SFW SGI SHTML SIX SIXEL SR2 SRF STEGANO SUN SVG SVGZ TEXT TGA TIFF TIFF64 TILE TIM TTC TTF TXT UBRL UBRL6 UIL UYVY VDA VICAR VID VIFF VIPS VST WBMP WEBP WMF WMV WMZ WPG X X3F XBM XC XCF XPM XPS XV XWD YUV

4.6 Excel File Commands

Command: **excel_file_commands <catalog excel file>**

Only one Excel file can be processed per command. In order to process multiple Excel files then each file will need to be listed using a separate command.

This command will execute all the individual file commands entered into the spreadsheet such as to move, delete, or rename an image file.

See "Catalog Excel" for all the possible individual file commands.

4.7 Flatten Comma Names

Command : **flatten_comma_names**

The **catalog** command ignores all files with commas in the name. This is required because the catalog file separates fields using commas. The program will display all files skipped when the catalog command is run. If you see files ignored because they contain commas and you are comfortable replacing the comma with the underscore *"_"* for ALL files then run this commmand. You can always individually rename files and run the catalog command.

4.8 Flatten File Names

Command : **flatten_file_names**

Punctuation:

```
{}[](),:;<>!'`"@#$%^&*|
```

Flatten file names removes the listed punctutation from file names. In addition all spaces are replaced with underscores. I love this feature because image files that come from the wild do so with strange names, where the wild can be web site downloads and emails. Typing file names with punctuation is a pain so I use this command to strip the puncuation. The only caveat is if the resulting, stripped name already exists as a file. In that case this command will insert underscores at the first occurrence of puncuation until a new name can be found that doesn't already exist.

4.9 JPEG Optimize Size

Command: **jpeg_optimize_size <bytes>**

There is a common Linux program called, *jpegoptim*. If you have this program installed on a Linux system then run this corresponding command to shrink all JPEG image files in a directory to approximately the size requested. The size given is in Kilobytes. This means a number like **6000** means 6MB, or 6000KB. Files of size smaller than the requested size are ignored.

4.10 List Empty Files

Command: **list_empty_files**

If one has thousands of files in a directory then something as simple as listing all empty image files can be quite handy. Empty files are often indication of failed downloads of image files. That is a common occurrence for me with my ISP and the size of some image files.

4.11 Remove Duplicate Files

Command: **remove_duplicate_files**

Remove all duplicate files leaving the original intact. The original is the file with earliest date. A duplicate file is determined by comparing file contents and not file names. If two files have exactly the same ones and zeroes that make up the file then the file is considered a duplicate and the file name is never taken into consideration.

4.12 Remove Empty Files

Command: **remove_empty_files**

Remove all empty image files in a directory.

4.13 Remove Multiple Format Files

Command: **remove_multiple_format_files**

Given two file names only differ by the file extension, like say 'jpg' versus 'png', then remove the larger one. I don't know about you but sometimes I save a file as a different format, typically from say PNG to JPEG, and then I leave both files in the directory. Which format is removed? The answer is the file with the largest size. Whichever file is larger in size in bytes is the one removed. This is because I generally save PNG to JPEG to save space. However, sometimes PNG is smaller, especially for cartoons and other simple images. Whereas removing duplicate files only looks at the content of a file then removing multiple format files only looks at the file name. This means if two original files exist with the same name except for the file extension then one will be deleted. Make sure your original files have original names before running this command.

CHAPTER

FIVE

INSTALLATION

Things listed in the prerequisites require instructions found on the prerequisites web site for your operating system.

There exists a version of this application on the PyPi web site. However installing the package defeats the design of any Almost There software. Instead insall from source as instructed below. Build the package and then install from this build. Then you can start modifying the software files.

5.1 Installation Prerequisites

1. Linux: any current version should due but this has only been tested on Ubuntu 20.
2. GIT: a command line version runnable as "git".
3. Imagemagick: Programs and libraries where libraries are used by Python for image processing. Some Imagemagick commands are used instead of libraries.
4. Python 3.6+: Dependencies require 3.6 or later.
5. Python virtualenv: Almost There software should not be installed to the OS but the image directory where all the files are owned by the user account and can be completely removed.
6. Make: Builds the application.
7. jpegpotim: This is optional and is used to compress JPEG files to a byte size.

5.2 Make

See MAKE.

MAKE.rst contains a comprehensive list of *Makefile* commands. Commands for runnging tests, creating web documents, and running code analysis using pylint are included.

The GMU Image Manage application is built using the The GMU PyPi Template project and the *Makefile* is from this project.

5.3 Installation Instructions

1. Open a command line shell.
2. Change directory into iamge directory that you wish to manage, example 'cd ~/Pictures'.
3. Download the application with git, 'git clone https://bitbucket.org/thegmu/thegmu-imagemanage'.
4. Change directory to the source directory, 'cd thegmu-imagemanage'.
5. Create the Python environment for the Almost There application, 'python3 -m venv almostthere'.
6. Activate your new Python envrionment, 'source almostthere/bin/activate'.
7. Activate the build environment, 'source bin/activate-almostthere'.
8. Build and install the application using make, 'make install'.
9. Test installation with an empty command file that does nothing, 'thegmu_im.py test/data/commands/no.commands.txt'.

Output:

```
thegmu_im.py test/data/commands/no.commands.txt
[11/14/2020 15:57:29] gim.prog.thegmu_im.py.119 % 'test/data/commands/no.commands.txt
↪' command file name requested.
```

Shell commands only:

```
cd ~/Pictures
git clone https://bitbucket.org/thegmu/thegmu-imagemanage
cd thegmu-imagemanage
python3 -m venv almostthere
source almostthere/bin/activate
source bin/activate-almostthere
make install
thegmu_im.py test/data/commands/no.commands.txt
```

5.4 The End

CHAPTER

SIX

MAKEFILE

Tools:

1. **autopep8**: pep8 code beautifier
2. **pylint**: coding standards
3. **pytest**: test source
4. **readthedocs.org**: public documentation using sphinx
5. **sphinx**: html documentation
6. **tox**: test the source as installed package
7. **twine**: deploy the package to pypi.org, test.pypi.org
8. **Makefile**: run the tools

Configuration files:

1. **.gitignore**: ignore pylint, pytest, tox and build files as well .settings, .project, and .pydevproject directories from Eclipse.
2. **.pylintrc**: The GMU specific PEP8 suppression.

6.1 Makefile Commands

make <command>

_default: Same as help.

backup-docs: Create a temp directory, 'docs.tmp.XXX', using mktemp and copy the docs directory to it.

clean: Removes Python compiled files, pytest files, and tox test files. See clean-pyc and clean-tox.

clean-dist: Removes Python packaging files.

clean-docs: Removes sphinx documentation build files. Configuration files are not removed.

clean-pyc: Removes Python compiled files and pytest files.

clean-tox: Removes tox test files.

destroy-docs Removes all sphinx config and manually edited document files as well as all generated files. See clean-docs. See backup-docs.

dist: Creates source and binary Python packages suitable for PyPi.

docs: Build the the HTML documentation files in docs/_build.

help: Displays this file.

init:

1. Install Python tools used by this Makefile.
2. Run project-init, see project-init.

pep8: Run `autopep8` and update all the project and test files in place with white space changes.

project-init:

1. setup.py: NAME, AUTHOR, AUTHOR_EMAIL, URL, SCRIPTS all updated.
2. test/sample_test.py: import of project name updated.
3. tox.ini: envlist updated

publish:

1. Publish the package to production 'pypi.org'.
2. User name and password prompt are given.

publish-test: Publish the package to test 'test-pypi.org'. User name and password prompt are given.

pylint: Run `pylint` and output results. No other action is taken. See `pep8` option to fix white space problems.

requirements: Python 'pip' packages for the tools.

test: Run the tests from source using pytest.

test-dist: Run the tests from virtual envinorments using tox. Builds the package and then run the test as packages in temporary Python virtualenv environments.

upgrade: Upgrade Python 'pip' packages for the tools.

> The reasonable person adapts themself to the world; the unreasonable one persists in trying to adapt the world to themself. Therefore all progress depends on the unreasonable person. –George Bernard Shaw

The End

CHAPTER

SEVEN

SOURCE

7.1 thegmu_imagemanage module

7.1.1 thegmu_im_catalog.py

Library that creates a flat file database of all image files in a directory.

#. Create a catalog for grouping by size, dimensions, format, creation date, etc. for manual management.

class thegmu_imagemanage.thegmu_im_catalog.**TheGMUImageManageCatalog**

Bases: object

Use /usr/bin/stat and ImageMagick indentity to create the catalog data.

CATALOG_FILE_DEFAULT = 'thegmu_imagemanage.catalog.txt'

CATALOG_RECORD = {'WxH': None, 'date': None, 'epoch': None, 'ext': None, 'file_name

catalog(*cmd_name*, *working_directory=None*, *catalog_file=None*)

Given a line of text containing the command to catalog then create a catalog file. Repeated calls update the catalog of file changes in the directory of request. The catalog file is never updated once created so the file is recreated each time with 100% of all data.

Parameters

- **cmd_name** – 'catalog'.
- **working_directory** – directory with images to catalog.
- **catalog_file** – file to list all the image information.

catalog_excel(*cmd_name*, *working_directory=None*, *catalog_file=None*)

Invokes catalog and then creates an excel file.

Parameters

- **cmd_name** – display and logging only.
- **working_directory** – any directory subject to override by the catalog.
- **catalog_file** – the text catalog file for the images.

catalog_init(*cmd_name*, *working_directory*, *catalog_file*)

Initialize and validate paramaters passed to catalog as well as load an previous catalog file found.

Parameters **cmd_name** – display purposes only for logging.

Working_directory image directory.

Catalog_file the catalog file for the iamges.

catalog_load()
Load an exist catalog file.

catalog_set_directory_files()
List the working_directory for files and create the image record. Not all files will be be images and these non-image files will be pruned later

catalog_set_file_stats()
Update the file record by adding os.stat data per file.

catalog_set_filetype()
Add the image file mime type data per file,

catalog_update()
Remove previous catalog files that no longer exist or or have more recent dates and then populate the catalog with image date from the previous catalog for those files that have the same date.

7.1.2 thegmu_im_catalog_excel.py

Library that loads and saves the catalog to an Excel spreadsheet format. New columns with image links to a thumbnail and original image are added as well as new column for individual file commands.

1. file:///thumbnail/image.jpg column added.
2. file:///image.jpg column added.
3. command column added.

class thegmu_imagemanage.thegmu_im_catalog_excel.**TheGMUImageManageCatalogExcel**
Bases: object

Load and save the catalog to an Excel format. Includes two new image link columns and a command column.

CATALOG_CELL_SIZES = {'WxH': 1.3, 'date': 2, 'delete': 0.7, 'epoch': 1.2, 'ext': 0

CATALOG_EXCEL_CELL_HEIGHT = 80

CATALOG_EXCEL_CELL_WIDTH = 20

CATALOG_EXCEL_COLS = ('thumbnail', 'original', 'note', 'delete', 'move', 'rename', 'ta

CATALOG_EXCEL_EXT = 'xlsx'

CATALOG_EXCEL_MAX_ROWS = 1000

CATALOG_EXCEL_SHEET_KEY = 'Catalog'

CATALOG_EXCEL_THUMBNAIL_DIR = 'thumbnails'

CATALOG_EXCEL_THUMBNAIL_HEIGHT = 100

CATALOG_EXCEL_THUMBNAIL_WIDTH = 300

CATALOG_EXCEL_USER_COLS = ('note', 'delete', 'move', 'rename', 'tags')

7.1.3 thegmu_im_convert.py

Library that converts images between formats such as PNG and JPEG using the Imagemagick convert library.

class thegmu_imagemanage.thegmu_im_convert.**TheGMUImageManageConvert**
Bases: object

Use ImageMagick 'convert' to change formats from PNG to JPEG, etc.

convert(*cmd_name*, *format_from*, *format_to*, *working_directory=None*, *catalog_file=None*)
Convert all images of one format in the working_directory using one parameter conversion routine that relies on Imagemagick default quality and memory usage policies set in the Imagematick policy /etc/imagemagick/policy.xml

Deletes the original file.

Parameters

- **cmd_name** – display only logging name.
- **format_from** – any valid Imagemagick format.
- **format_to** – any valid Imagemagick format.
- **working_directory** – directory with images to convert.
- **catalog_file** – the text file to store image information.

Data catalog self.data['catalog']['current']

Return count count of files converted.

convert_format_simple(*cmd_name*, *format_from*, *format_to*, *working_directory=None*, *catalog_file=None*)
Convert all images in the working_directory using one parameter conversion routine that relies on Imagemagick default quality and memory usage policies set in the Imagematick policy /etc/imagemagick/policy.xml

Leaves the original file in place.

Parameters

- **cmd_name** – display only logging name.
- **format_from** – any valid Imagemagick format.
- **format_to** – any valid Imagemagick format.
- **working_directory** – directory with images to convert.
- **catalog_file** – the text file to store image information.

Data catalog self.data['catalog']['current']

Return count count of files converted.

convert_list_formats(*_cmd_name=None*)
Print to console the possible Imagemagick formats available for conversion.

Parameters **cmd_name** – display only.

7.1.4 thegmu_im_errors.py

Library of simple errors with only distinct names as a feature.

exception thegmu_imagemanage.thegmu_im_errors.**TheGMUImageManageBadCommandError**
Bases: *thegmu_imagemanage.thegmu_im_errors.TheGMUImageManageError*

Command Exception.

exception thegmu_imagemanage.thegmu_im_errors.**TheGMUImageManageCatalogError**
Bases: *thegmu_imagemanage.thegmu_im_errors.TheGMUImageManageError*

Catalog Exception

exception thegmu_imagemanage.thegmu_im_errors.**TheGMUImageManageConvertError**
Bases: *thegmu_imagemanage.thegmu_im_errors.TheGMUImageManageError*

Imagemagick format convert error.

exception thegmu_imagemanage.thegmu_im_errors.**TheGMUImageManageError**
Bases: Exception

Base exception.

7.1.5 thegmu_im_excel_file.py

Library that loads existing Excel catalog files and manages image files based upon individual file commands in the spreadsheet.

class thegmu_imagemanage.thegmu_im_excel_file.**TheGMUImageManageExcelFile**
Bases: object

Load a previously created Excel catalog file and execute individual image commands. Does not change the Excel file, only read it.

excel_file_commands(*cmd_name*, *excel_file*, *working_directory=None*, *catalog_file=None*)
Run individual file commands from an Excel spreadsheet the user as updated.

Parameters

- **cmd_name** – command name for display only.
- **excel_file** – user requested file taken from the command file.
- **working_directory** – the image directory.
- **catalog_file** – the catalog file of images.

7.1.6 thegmu_im_os.py

Library that like the python "os" and "os.path" libraries provides various destructive and informative file operations.

class thegmu_imagemanage.thegmu_im_os.**TheGMUImageManageOS**
Bases: object

Operating system services for updating and deleting files. ImageMagick may be used to convert from one type to another.

FLATTEN_SPEC = ' {}[]()\,:;<>!\'"@#$%^&*|`'

JPEG_OPTIMIZE_ARGS = ' -q -s -S%i '

JPEG_OPTIMIZE_CMD = 'jpegoptim'

JPEG_OPTIMIZE_MIN_SIZE = 10000

flatten_comma_names(*cmd_name*, *working_directory=None*, *catalog_file=None*)
See flaten_file_names, calls flatten_file_names passing a list of a single comma as the spec. File names with commas are ignored for processing. Use this command to get rid of the comma file names.

Parameters

- **cmd_name** – command name for display only.
- **working_directory** – the immage directory.
- **catalog_file** – the image catalog file.

Return count the number of files renamed.

flatten_file_names(*cmd_name*, *working_directory=None*, *catalog_file=None*, *spec=None*)
Flatten file names means to replace file name puncutation with and spaces with underscore '_'. In the event that the name already exists then a new version is created with '_N_'.

Parameters

- **cmd_name** – for logging and display only.
- **working_directory** – any directory subject to override by the catalog library.
- **catalog_file** – previously created catalog file.
- **spec** – a string of characters to flatten.

Return count count of file names flattened.

jpeg_optimize_size(*cmd_name*, *kilobytes*, *working_directory=None*, *catalog_file=None*)
Run the program "jpegoptim" to shrink a jpeg to a target size.

Parameters

- **cmd_name** – for display purposes only.
- **working_directory** – any directory subject to default and override by the catalog library.

Return count count of optimized files.

list_empty_files(*cmd_name*, *working_directory=None*)
List empty files in the working directory where working_directory is defined by TheGMUImageManage-Catalog library.

Parameters

- **command_name** – 'remove_empty_files'
- **working_directory** – any directory subject to default and override by the catalog library.

Return count count of empty files.

remove_duplicate_files(*cmd_name*, *working_directory=None*)
Remove duplicate files in a working_directory as determined by the catalog library.

Parameters

- **cmd_name** – reporting name only.
- **working_directory** – a directory that can be None or possiby overridden by emnvironment variable.

Return count count of files deleted.

remove_empty_files(*cmd_name*, *working_directory=None*)
Remove empty files in a working_directory as determined by the catalog library.

Parameters

- **cmd_name** – reporting name only.
- **working_directory** – a directory that can be None or possiby overridden by emnvironment variable.

Return count count of files deleted.

remove_multiple_format_files(*cmd_name*, *working_directory=None*)
Remove files with multiple formats such as jpg and png using a working_directory as determined by the catalog library. When multiple formats are detected then the larger file is deleted. Given files 'a.jpg' and 'a.png' then delete the larger file.

Parameters

- **cmd_name** – reporting name only.
- **working_directory** – a directory that can be None or possiby overridden by emnvironment variable.

Return count count of files deleted.

set_directory_files(*cmd_name*, *working_directory=None*, *catalog_file=None*)
List directory files in the working_directory where working_directory is defined by the catalog library. Set list to self.data['catalog]['work_files']

Parameters

- **cmd_name** – for display only.
- **working_directory** – any directory subject to default if None or overridden by environment variable.

Return count count of files set in self.data['catalog]['work_files'].

set_duplicate_files(*cmd_name*, *working_directory=None*)
List duplicate files in the working directory where working_directory is defined by the catalog library.

Duplicate files must first have equal file sizes. Only after equal file size is determined is the md5sum calculated.

The original is the one with the earliest time stamp.

Parameters

- **cmd_name** – for display purposes only.
- **working_directory** – any directory subject to default and override by the catalog library.

Return count count of duplicate files.

7.1.7 thegmu_im_util.py

Library of utilities such has creating the md5sum checksum for a file.

class thegmu_imagemanage.thegmu_im_util.**TheGMUImageManageUtil**
Bases: object

Utilities such as creating md5sum checksums.

CONVERSION_EXT = {'JPE': 'jpg', 'JPEG': 'jpg', 'unknown': 'none'}

IDENTIFY_FORMATS_CMD = "identify -list format | grep -E -e'^\\s+[A-Z0-9]+*?\\s+[A-Z0

IMAGEMAGICK_FORMATS = []

static get_file_ext(*work_file*)
Cananical file extension determination for this application.

Parameters **work_file** – the file name

Return extension the file extension.

classmethod get_image_format_ext(*image_format*)
Use the Imagemagick 'identify –list format' program to list image file formats.

Parameters **image_format** – JPEG, PNG, or other Imagemagick format.

Return extension the file extension associated with the format.

classmethod get_image_formats()
Use the Imagemagick 'identify –list format' program to list image file formats.

Return formats a list of Imagemagick formats.

static get_list_csv_text(*csv_list*)
Converts a Python list into a thegmu_imagemange CSV text line.

Parameters **csv_list** – a Python iterable to be cast as a list.

Return csv_text formated csv text from csv_list

static get_md5sum(*md5sum_file*)
Python md5sum checksum for file similarity comparison.

Keep In Mind

Files are slurped into main memory, although images shouldn't be more than a few megabytes.

Parameters **md5sum_file** – any file that fits in memory.

Return md5sum string in hexidecimal format.

static get_mtime_datetime(*mtime*)
Given the modified, mtime from a stat call return a canonical date used by this application.

Parameters **mtime** – from a stat calle, st_mtime

Return datetime string YYYY-MM-DD-HH:MM:SS

classmethod replace_file_ext(*work_file*, *new_ext*)
Replace the existing file extension with a new one.

Parameters **work_file** – the file name to have extension replaced.

Return new_work_file the file extension replaced

classmethod replace_file_ext_by_format(*work_file*, *image_format*)
Replace the existing file extension with a new one using a passed in image format.

Parameters work_file – the file name to have extension replaced.

Return new_work_file the file extension replaced

7.2 log module

7.2.1 thegmu_log.py

Standard Python Logging extension.

1. Interleave dependent package messages using a unique three letter acronym name.
2. Configure logging to taste.
3. Default configuration resolves all method names to four characters. this prevents wavy indent where 'warning' being 6 characters and 'debug' is five and 'info' is 4.
4. If you prefer the standard Python logging names then just create a context as such and pass that in.
5. GLOBAL class variables are used because the logging module is global state.

class thegmu_imagemanage.thegmu_log.**TheGMULog**(*tla=None*, *context=None*)
Bases: object

TheGMULog builds on Python logging which is a quasi singleton design pattern with context switching. YAML configuration is provided.

CLITLA = 'cli'

CURRENT_CONTEXT = {}

DEFAULT_CONTEXT_YAML = "\ncontext: null\ndefault_context: null\ndefault_level: PROG

LOG_FRAME_DEPTH = 2

STDOUT_STREAMHANDLER = <StreamHandler <stdout> (NOTSET)>

TLA_CONTEXT = {}

context_check(*context*)
Validate the state of the context passed in.

context_switch(*context*)
Set up the logging.

Parameters context – is a Python dictionary initially copied from DEFAULT_CONTEXT_YAML.

static get_default_context_copy()
copy.deepcopy the DEFAULT_CONTEXT_YAML.

get_level_name()
Return the current level name as string, i.e "DEBUG".

get_level_name_for_method_name(*method_name*)
Map a method name to its level string.

Parameters **method_name** – the name to map. It is an exception to pass an unregistered method_name.

get_level_name_method_name(*level_name*)
Map level name string to a class method.

Parameters **level_name** – The level name to map. It is an exception to pass an unconfigured level.

classmethod get_log_frame()
Stack trace log frame of current function.

is_current_context(*context*)
Timestamp is checked in case the context for the tla has been updated.

log_by_level_name(*msg*)
logging level output using a string instead of constant.

Parameters **msg** – Typically a one line log message.

classmethod remove_all_loggers()
Update the global state in the Python package 'logging'

remove_existing_logger_handlers()
Sets up the logging Stream handler for the current TLA

set_current_context(*current_context*)
Update the class global state with the passed on state.

Parameters **current_context** – The new context information to copy verbatim.

set_level_from_string(*level_name*)
Python logging uses integers, set the integer value mapped to this string.

Parameters **level_name** – A previously configured level_name mapped to a level integer. It is an exception to pass an unregistered level_name.

exception thegmu_imagemanage.thegmu_log.**TheGMULogException**
Bases: Exception

TheGMULog class exception.

7.3 script module

7.3.1 script.py

Command line script utilities.

Most notably *runcmd()* that runs Linux bash command strings and outputs to console as needed.

exception thegmu_imagemanage.script.**ScriptException**
Bases: Exception

Exception specific to this file, notably the 'initlock' function.

thegmu_imagemanage.script.**begin**()
Print *timestamp()* with BEGIN.

thegmu_imagemanage.script.**command_exists**(*command_name*)
Given the name of an operating system program assumed to be in the current path then return True or False if the command exists.

Parameters **commmand_name** – the command name

Return True if the command exists.

thegmu_imagemanage.script.**end**()
Print *timestamp()* with END.

thegmu_imagemanage.script.**env_string_replace**(*env_string*, *env_vars=None*, *empty_sub=False*)
Replace all enviornment variables in a string with the environment variable value.

1. If "env_vars" list is given then ignore os.environ and use this list.
2. ${HOME}: all substitutions variables require curly braces as in ${HOME}.
3. If "env_vars" dict is given the both key and values are taken from the dictionary.
4. If an environment variable does not exist then no substitution is made.
5. If empty_sub is True then if an environment variable does not exist it will be replaced with an empty string.

Parameters

- **env_string** – the string for substituion with env variables.
- **env_vars** – substitution for os.envioron list of environment variables.
- **empty_sub** – If True then variables not found are removed, otherwise the they are left as in the string.

thegmu_imagemanage.script.**fatal**(*msg*, *exit_ok=True*)
Prints timestamp() with 'FATAL' to stderr prepended to a msg. Exit with -1 unless exit_ok is False.

Parameters **exit_ok** – If True then call sys.exit(-1) after printing message.

thegmu_imagemanage.script.**get_hostname**(*host_name_only=False*)
Retrun `socket.gethostbyaddr()` string of the FQDN, or fully qualified domain name.

Parameters **host_name_only** – short name only Example:

```
localhost.localdomain -> localhost
```

thegmu_imagemanage.script.**get_log_frame**()
stack trace log frame of current function.

thegmu_imagemanage.script.**getnow**(*now_type=None*, *target_datetime=None*)
getnow() get a log file timestamp.

Parameters

- **now_type** – format option: #: None: 20190401:103226.82 #: 'script': 2019-04-01-10:32:26 #: 'script_date': 2019-04-01 #: 'SQL': 2019-04-01-10:32:26
- **target_datetime** – datetime.datetime() object. When None then defaults to datetime.now().

thegmu_imagemanage.script.**initlock**(*lockpath*)
Use a lock file to ensure only a single instance of a script is running at any one time.

1. The process id, PID, is the first and only line of the lock file.
2. Subsequent calls to *initlock()* check if the process corresponding to the PID is running and if not then acquires the lock, else the program exits.

Note: initlock fails if Linux account permission is denied by file permissions.

Parameters **lockpath** – The file name to hold the PID, for example:

```
/tmp/myscript.sh.lock
```

thegmu_imagemanage.script.**msg_error_code**(*msg*, *error_code*)
Create a new msg with 'ERRORCODE error_code:' for keyword log parsing.

Parameters

- **msg** – one line log message.
- **error_code** – any string but typically digits only.

thegmu_imagemanage.script.**print_dashes**(*msg*, *for_return=False*)
Call *print_header()* with char '-'

thegmu_imagemanage.script.**print_hashes**(*msg*, *for_return=False*)
Call *print_header()* with char '#'

thegmu_imagemanage.script.**print_header**(*char*, *msg*, *for_return=False*, *width=80*)
Print 3 lines per message using character line separators of the specified char, for example:

```
++++++++++++++++++++++++++++++++++++++++
print_header passing char as '+'.
++++++++++++++++++++++++++++++++++++++++
```

Parameters

- **char** – The character to repeat.
- **msg** – A one line log message.
- **for_return** – If True return as string.
- **width** – How many char to repeat, default is 80.

thegmu_imagemanage.script.**print_sql_comment**(*msg*, *for_return=False*)
Prefix '–' to every line passed in msg.

Parameters

- **msg** – A multi-line log message.
- **for_return** – If True then return the string.

thegmu_imagemanage.script.**runcmd**(*cmd*, *console=False*, *exception_continue=False*, *encoding=None*, *return_code=False*)
Characterized subprocess.check_call/check_output design patterns.

Note: stderr is always redirected to stdout.

Parameters

- **cmd** – A Linux command.
- **console** – If True then cmd is printed first than output is sent to stdout.

- **exception_continue** – If True exceptions become warnings and execution continues.
- **encoding** – binary bytes are returned unless encoding is passed as 'utf-8', 'ascii' or other encoding.
- **return_code** – Only return the integer return code value when console is True or an exception occurs.

thegmu_imagemanage.script.**success**(*msg='success'*, *eol=True*, *success_name=None*)

Print function name with 'success' or an optional msg to stdout.

Parameters

- **msg** – optional log message.
- **eol** – if True then append os.linesep.
- **success_name** – Substitute the function name with this string.

thegmu_imagemanage.script.**timestamp**(*msg='TIMESTAMP'*, *for_return=False*)

print a log message using a prefix of timestamp, file name, function name and line number prefix.

Example:

```
[2019-04-01-11:02:07] case.py.run.605 % hello
```

Parameters

- **msg** – A one line log message.
- **for_return** – If True return the msg.

thegmu_imagemanage.script.**warn**(*msg*, *eol=True*, *warn_prefix=None*)

prints timestamp('WARNING) to stderr along with the msg. Example: [2019-04-01-11:07:25] system_test.py.test00_python_stuff.50 % WARNING hello

Parameters

- **msg** – A one line log message.
- **eol** – If True append os.linesep to the msg.
- **warn_prefix** – Replace timestamp('WARNING') with this string.

CHAPTER

EIGHT

ALMOST THERE SOFTWARE

https://www.thegmu.com/

Authors Mybrid Wonderful, Gregg Yearwood

Date 12/015/2020

Support mybrid@thegmu.com

Version 1.0.0

8.1 Almost There Introduction

The objective of the Almost There initiative is to create jobs for people being displaced by automation. Almost There workers are like plumbers or auto mechanics who are trades people who have trade training and can be self taught.

Almost There Projects are intended to open the development of software to a larger audience than just software developers. The idea of being almost there is that any user can tweak a software program just a little.

The tasks of software development that require skills outside the trade worker are the graphical user interface (GUI) and data management. Therefore Almost There software has a texting interface and no GUI. Furthermore data manipulation is at the simplest level designed to be no more complicated than manipulating ingredients for cooking.

8.1.1 Almost There Inspiration One

One inspiration for Almost There software came from when I read an article where people who grew up texting could type faster on a phone than people using a keyboard. I thought to myself why not take advantage of that skill and give users a texting interface to running software?

There is no GUI in Almost There software because texting the computer itself is the interface.

8.1.2 Almost There Inspiration Two

Software companies want to make the most money possible. This means they design software to people with the least computing skills possible. This means there is a gap between what's possible and what's available. That gap comes from GUI's that must be the simplest possible to attract to the most buyers and this simplicity eliminating complex features due to feature complexity.

Almost There projects fill the gap between simple user interfaces and and missing complex features.

8.1.3 Almost There Software Mechanic

We have all the experience of buying clothes like say jeans and the the sizes available in those jeans are not quite right. We just learn to live with the misfit. However, sometimes we'll keep looking for a new jean manufacturer or perhaps get them tailor made.

Almost There workers are customizers. Think of the workers as artisans. You know how we have Art Fairs every summer outdoors, the Art and Wine Festivals?

Similarly you could have Almost There Fairs. We set up booths and people take there phones in and have them software customized, Why not? Almost There software then is not about getting rich, but just earning a day-to-day living providing personal service much the same way a plumber or an auto mechanic does.

There is not such personal software market today. The Almost There project mission is to create that market and hopefully millions of jobs in the process.

8.2 Almost There Details

Almost There projects have the following details:

1. Open Source code
2. 100% Modifiable
3. Text interface
4. Text source
5. Do-it-yourself, or Maker design (DIY, Maker)

Almost There software has no web based or graphical interface because it is not considered a finished product. Also the goal is to get the public engaged into making slight changes to software such as naming things.

If any Almost There project is worthy of having a graphical interface due to the value of that project then the Almost There project should be sold or licensed to the company putting the UI on it.

8.2.1 Almost There Community

Computing is hard. If you are considering working on an Almost There software project just realize how hard it will be. First off most of the online documentation for software is written by developers for developers. You'll be wanting to create a blog with all your experience for not only yourself for others.

You will struggle going alone. You'll want to search out help beyond just what you can find online. If you are lucky you'll have access to a developer like me.

Installation and configuration will be the big hurdles. Modifying the software is designed to be a mechanics work, trade work. However, installing Linux, installing Python, installing Imagemagick, and all other kinds of software is hard. When you run into roadblocks don't give up. Ask for help.

Community can be the job. You should seriously consider not only becoming an Almost There mechanic for making money, but a content creator as well.

Learn how to make money on Youtube, Twitch, and blogs using advertising revenue.

The Almost There market will require far more community than the current software development community because of skill levels. Don't be afraid or ashamed of that skill level difference, use it to your advantage and treat the community as a job as much as the job itself.

8.2.2 Almost There Future

Almost There software is just one piece of future governing called Irreni World Scale. Irreni relies on self-governing. Thomas Jefferson said, "The government that governs least governs best because the people govern themselves." Jefferson never expanded on what that meant.

Irreni has a world plan for self-governing and a big part of that is understanding that governing yourself means doing for yourself by becoming a Maker. Long before Capitalism sold us finished consumer products for everything we made our own clothes. Every home had a sewing machine.

Almost There software enables you to do for yourself. Sure, in the beginning Almost There software capability will be limited because most software today is consumer software. But as the Almost There market grows then so will the formal software development change direction to produce materials and parts and not just finished goods. This will enable people with a mechanics skill in software to produce every larger arrays of finished software goods themselves.

Eventually Almost There software will allow 3D printing and Almost There hardware to expand. Once we start down a social path of becoming our own Makers for food, clothing, shelter and software then we will rely less on governing to do so. Relying less on governing is us governing ourselves.

8.3 The End

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

t

INDEX

W