March 29, 2021

Drew Larson

The goal of this manual and tutorial is to provide an overview of how to download and use scripts written for the curation of MICH herbarium image archives. All scripts are written in the python programming language and do a variety of tasks such as checking file names, moving images to their appropriate storage location, and copying images from their storage location to a new folder. The scripts are written to be able to be run on any operating system (Windows, Mac, or Linux) as well as with either python 2.7 or 3.x. A brief description of each script, an example of its input file where relevant, as well as a description of any user options is provided.

All scripts described in this tutorial are publicly available on Github:

<https://github.com/drewlarsonphylo/Herbarium-collections-management>

Table of Contents

**1. Downloading the scripts** (pp. 2-4)

**2. Checking that python is installed and callable on your computer** (pp. 5-6)

**3. Running the scripts** (pp. 7-11)

**4. Description of Herbarium scripts [as of March 29, 2021]** (pp. 12-37)

General | test\_script.py (p.13)

General | **verify\_in\_base\_directory\_existance\_list\_filenames\_**

**or\_barcodes\_option\_to\_move.py** (p.14)

General | **return\_list\_of\_all\_files\_in\_folders.py** (p.16)

Algae | **File\_MICH\_algae\_images\_after\_verification.py** (p.17)

Algae | **Verify\_existance\_list\_of\_algae\_filenames\_**

**or\_barcodes\_option\_to\_copy.py** (p.18)

Algae | **change\_file\_names\_based\_on\_list.py** (p.20)

Algae | **curate\_algae\_image\_archives.py** (p.22)

Algae | **list\_barcode\_nums\_in\_archive\_dir.py** (p.23)

Algae | **replace\_mich\_\_with\_mich-a-.py** (p.24)

Vascular\_plants | **rename\_image\_codes\_lowercase\_to\_uppercase.py** (p.25)

Vascular\_plants | **File\_vascular\_plant\_images\_**

**into\_archives\_checking\_if\_duplicates.py** (p.26)

Vascular\_plants | **Verify\_existance\_list\_of\_vascular\_**

**plant\_filenames\_option\_to\_copy.py** (p.28)

Vascular\_plants | **curate\_vascular\_image\_archives.py** (p.30)

General | **Unfile\_list\_of\_barcodes.py** (p. 31)

Fungi | **Verify\_existance\_list\_of\_fungi\_filenames\_option\_to\_copy.py (p. 32)**

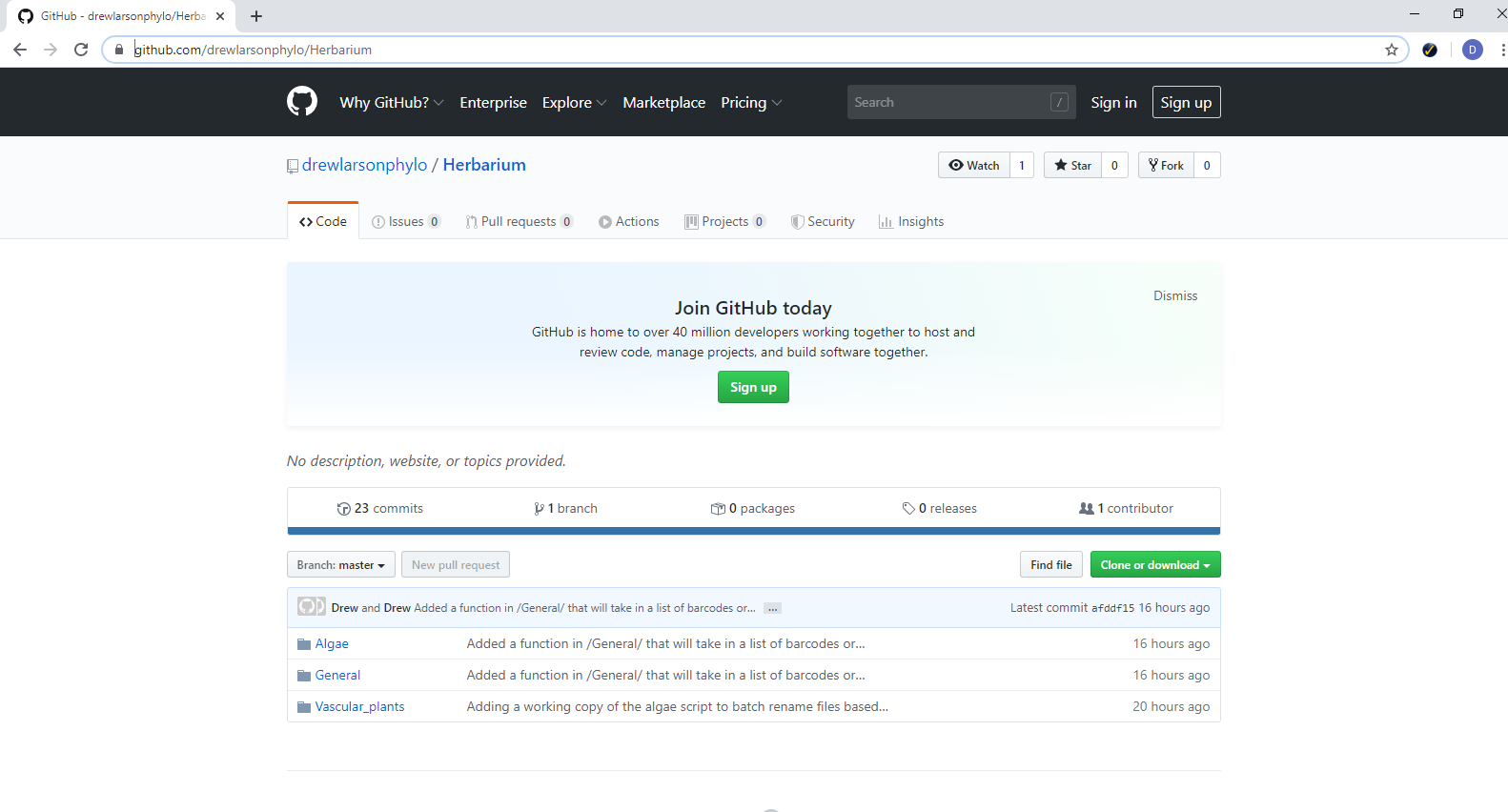
Fungi | **File\_fungi\_images\_into\_archives\_checking\_if\_duplicates.py (p.35)**

Fungi | **curate\_vascular\_image\_archives.py** (p.37)

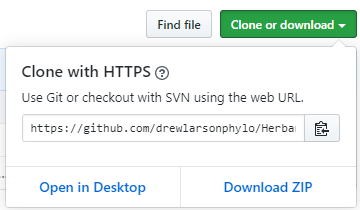
**1. Downloading the scripts**

There are a few different ways to download files from Github. This tutorial describes the most universal, that will work on any computer.

Open any web browser and navigate to <https://github.com/drewlarsonphylo/Herbarium>.

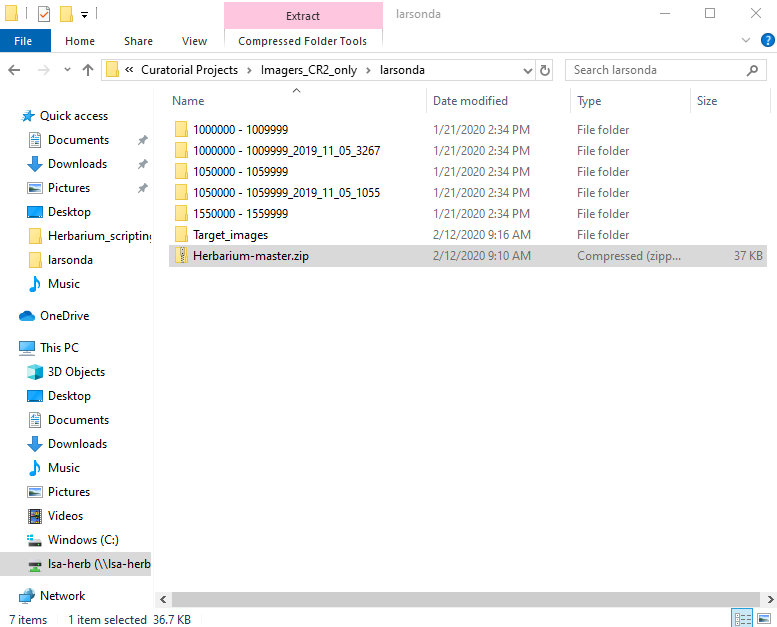


Click on the Green “Clone or download” button and select “Download ZIP”

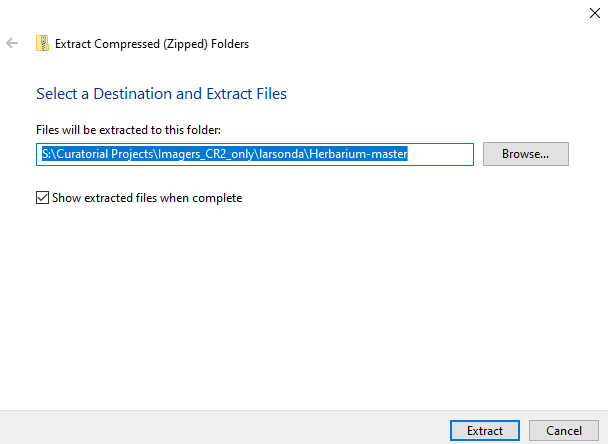


Then, select the location where you would like to save the zipped folder. It doesn't really matter where you choose, but for simplicity, it’s best to put it in the location where you plan to store the scripts.

In this example using a Windows computer, the zipped folder is saved to **S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\**



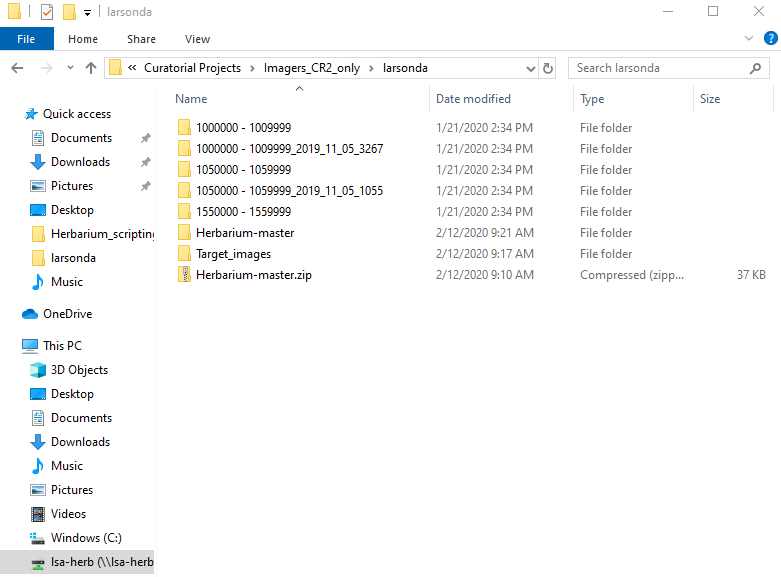
The next step is to unzip the folder. There are a variety of programs that can do this. Right click on the zipped folder and select “Extract All...”. Then select the location to which you would like to extract the folder and click “Extract”.



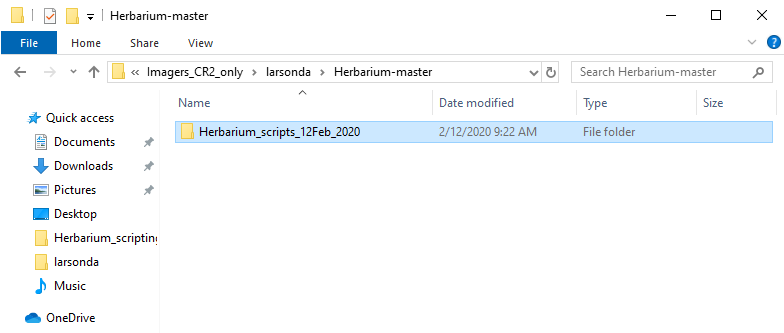
Now there is a non-zipped copy of the folder “Herbarium-master” and inside of this folder, there will be a second folder with the same name that contains the scripts. Rename the *inner* “Herbarium-master” something that makes it clear what date you downloaded it, as well as any other information you would like to include about it. Move the renamed folder to the final location where you plan to store the scripts. In this example, this is the same place where the zip folder was saved. In this example, the folder is renamed “Herbarium\_scripts\_12Feb\_2020”.

You can now delete the .zip folder and the empty “Herbarium-master” folder.

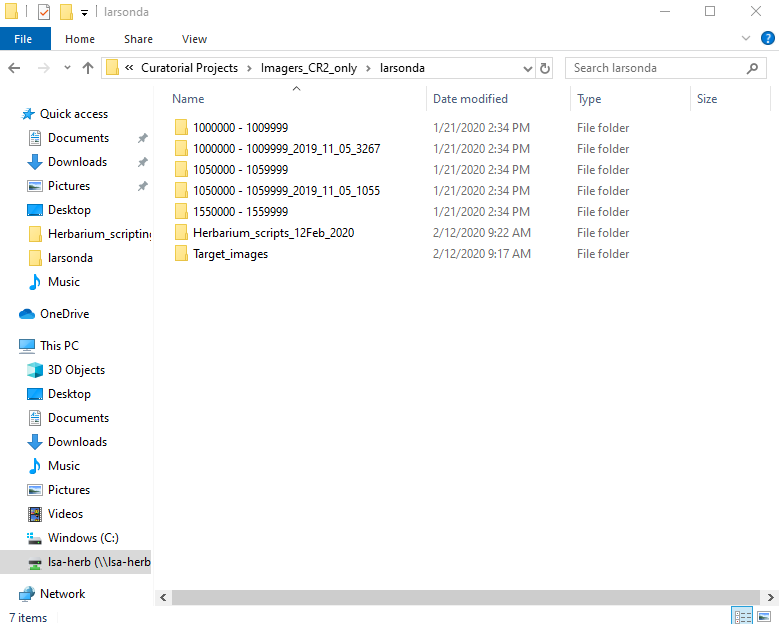
*After saving the zip folder*



*Renaming the inner “Herbarium-master”*



*Final location of the renamed folder*

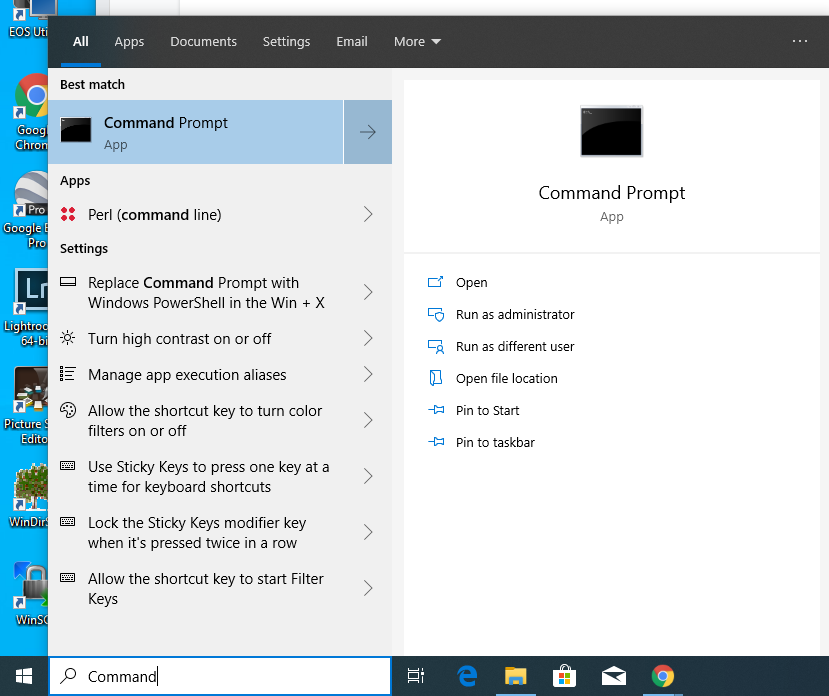


**2. Checking that python is installed and callable on your computer**

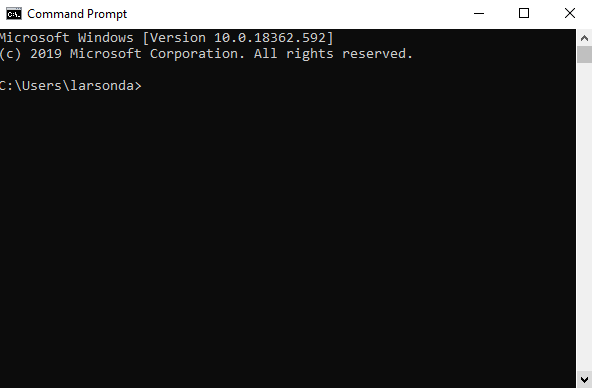
Before you can run any of the scripts, you need to be sure you have python 2.7 or python 3 installed on your computer and know how to call on python to do things.

If you are using a Mac or Linux computer, you should already have python installed. Open your terminal and simply type: “python”. If you don’t get an error message, this means that python is already installed and in your path. Basically, this means you don’t have to do anything else to install python. Type “exit()” to quit the interactive python interface.

If you are using a Windows computer, you should also already have python 3 installed, type “Command Prompt” in the search bar and open the application.

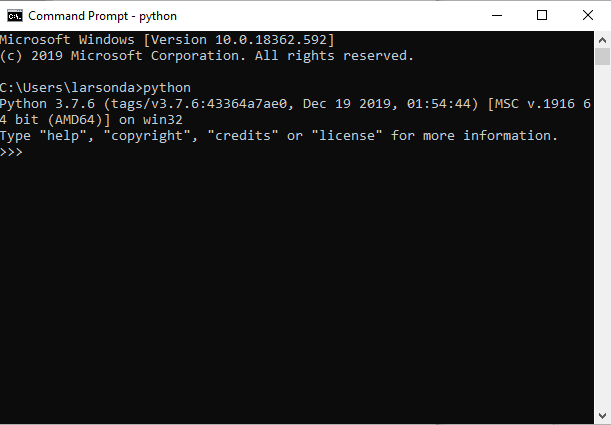


A window should open that looks like this:



Type “python” into the Command Prompt and hit enter. If you don’t get an error, that means that python is already installed on your computer. In this example, the command “python” is telling the computer to run Python version 3.7.6.

Type “exit()” and hit enter to quit the interactive python interface.



If python is not already installed on your Windows computer, you can try installing it with the Software Center, or contact IT about installing python to your computer’s path.

There are additional ways that python can be called to run without adding it to your computer's path, but those won’t be covered here. Contact IT if you need help installing python to your path.

**3. Running the scripts**

Once you know how to call python, and have downloaded, extracted, renamed, and decided where to store the script folder, you are ready to begin running the scripts.

All the scripts are designed so that they can be run from **any location on your computer** and will act on files in your **current working directory**. This means that you should never have to move the scripts, in some cases doing so will actually cause them not to work.

Your current working directory is essentially your computer’s way of keeping track of where you (the user) are working relative to other files in the computer’s memory. To run a python script, simply call on python and then tell python which script you would like it to run.

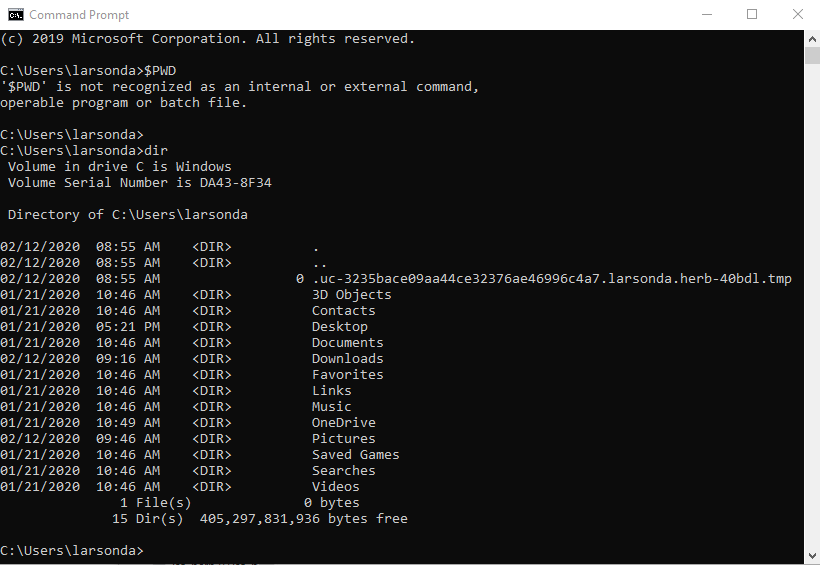
This tutorial will show how to run a script using the script’s **absolute pathname** when python is in your computers path variable (see previous section). It’s also possible to call on scripts using **relative pathnames** in the Command Prompt or terminal, but there isn’t any need to do so for these scripts.

The first step in running these scripts is to navigate through your computer’s directory structure until your current working directory is the directory where you want to script to take action.

For example, if we want to use scripts on a Windows computer to curate several folders containing dng images from MICH vascular plants, the first step is to open the Command Prompt and navigate to the location those folders are stored.

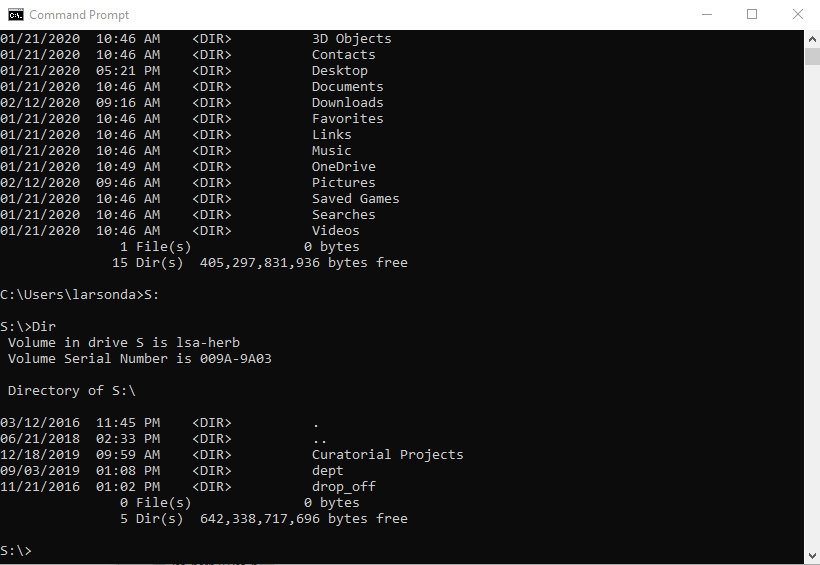
In this case: S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Vascular\_plant\_example\_archives

Open the Command Prompt. Next to the blinking cursor will be your current working directory. The command “**Dir**” will tell the Windows Command Prompt to print the screen several facts about your current working directory, including which drive you are in and all the other files and directories contained in your current directory.



The C: part of the current working directory is the current **drive**. If you want to work with files in a different drive (for example the S: drive) you can change drives by simply typing the name of the drive you would like to change to followed by enter.

*Typing “S:” changes the current drive to the S: drive. You can use “Dir” to double-check*



Once you are in the right drive, you need to navigate to the right directory on that drive. The command “**cd**” stands for “change directory” and tells the command prompt to change the current working directory to whatever follows. It will give you an error if you try to change into a directory that doesn’t exist.

Since we want to navigate to

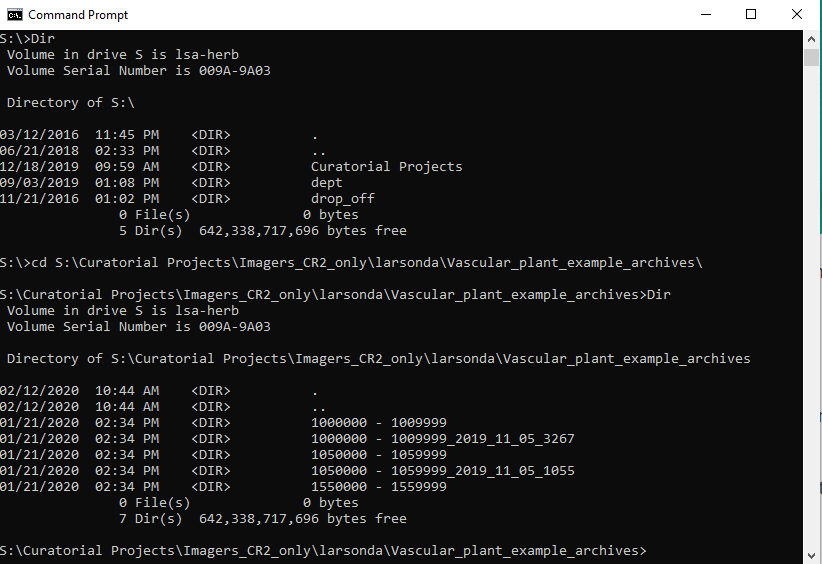
S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Vascular\_plant\_example\_archives\

and we are in the right drive (S:) we can get to this location by typing the command

“cd S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Vascular\_plant\_example\_archives\”

Notice that there is a space between “cd” and the location of the directory.

*You can drag and drop a file from your desired directory into the Command Prompt as a shortcut. Doing so will automatically load the absolute filepath for the file you dropped, which can be modified as needed to get to the correct directory. Modifications might include erasing the filename after the last “slash”, but keeping in the quotes “ “ if the folder names have spaces in them.*



You can check that you are in your desired directory with the “Dir” command. You should see all the files and folders that you want to work with listed on the screen. There will be a lot if you are in a directory with lots of files.

Once you have set your current working directory correctly, you are ready to run the scripts. To call any of the scripts, just type the command to call python followed by a space and the absolute filepath for the script you would like to run. Since in the example, we saved the script folder as S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Herbarium\_scripts\_12Feb\_2020\

We can call a test script to check that everything is working by typing the following command and hitting enter:

‘python “S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Herbarium\_scripts\_12Feb\_2020\General\test\_script.py” ’

Notice the space after “python”. The quotes are necessary around the pathname to the script because there is a space in the path (between the words Curatorial and Projects). Again, dragging and dropping can be used as a shortcut to avoid typing everything out. Alternatively you can use Ctrl+v to paste text into the Command Prompt.

If you can successfully run **test\_script.py** then you should be able to run the other scripts the same way. Just specify a different script for python to run by using that script file path. For example:

**python “S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Herbarium\_scripts\_12Feb\_2020\Algae\curate\_algae\_image\_archives.py”**

or

**python “S:\Curatorial Projects\Imagers\_CR2\_only\larsonda\Herbarium\_scripts\_12Feb\_2020\Vascular\_plants\rename\_image\_codes\_lowercase\_to\_uppercase.py”**

**4. Description of Herbarium scripts [as of Feb 7, 2020]**

There are four folders that contain Herbarium scripts: General, Vascular\_plants, Fungi, and Algae.

**General** contains scripts meant to be useful for manipulating image files or archives in ways that don’t require anything taxon specific. For example, these scripts don’t look for “MICH-V-” or verify the legitimacy of file names, since these vary depending on the taxa.

**Vascular\_plants** contains scripts that are meant to be used for MICH vascular plant images. Several of the scripts in this folder require python functions they inherit from vascular\_plant\_utils.py. Practically, what this means is that these scripts need to be in the same folder as vascular\_plant\_utils.py, so it’s best to leave them all in Vascular\_plants.

**Fungi** contains scripts that are meant to be used for MICH Fungi (including Lichen) images. These scripts are based on their corresponding scripts for vascular plants, but have been altered to work with Fungi images, which do not have leading zeros and can have anywhere between 1 and 7 digits in their barcode numbers. Several of the scripts in this folder require python functions they inherit from fungi\_utils.py. Practically, what this means is that these scripts need to be in the same folder as vascular\_plant\_utils.py, so it’s best to leave them all in Fungi.

**Algae** contains scripts that are meant to be used for MICH algae images. Several of the scripts in this folder require python functions they inherit from Algae\_utils.py. Again, this means these scripts need to be in the same folder as Algae\_utils.py, so it’s best not to move anything from the folder. Several of the scripts in Algae have close counterparts in the Vascular\_plants folder, however, because there are differences between the two in how image files are named, separate scripts have been developed for each.

The following are descriptions of the purpose of each script, an example of what any required input file should look like, and instruction on how to use and modify any user options that are included in the script.

General | **test\_script.py**

This script tests that python is installed correctly on the computer and is meant to provide users an opportunity to practice calling a python script. It prints the following to the screen:

"Script is running successfully"

The version of python used to run the script

The current working directory

The path used to call the script

User options:

None

Input File:

None

Output File:

None

General | **verify\_in\_base\_directory\_existance\_list\_filenames\_or\_barcodes\_option\_to\_move.py**

The script takes a target list as an input file and looks through the current working directory. It reports any files that match any of the targets. For each target, it reports the number of hits and the file paths for each hit.

User options:

**inputTargetPath="TargetList.txt"**

The input file with a list of targets for which you want to search.

**output\_filename="Output.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten.

**header=True**

Change to True or False depending on if you want a header line specifying the columns labels of the output file. If True, the output file will have column labels, if False, the output file will be produced without column labels.

**include\_full\_filepath\_names=True**

True/False if you would like full file path names for any hits included in the output. If False, only the folder the image is in will be included. If True, the entire path will be output.

**move\_image\_hits=False**

If True, all images that hit to one of the targets will be **moved** (not copied) to the folder specified below in "move\_image\_directory" .

**move\_image\_directory="Targeted\_images"**

The name of the folder that images that hit a target will be moved to if move\_image\_hits is set to True.

Input file:

This script takes in a list of targets, one per line, as an input file. Unless you specify which directory contains your input file, python will look for the input file in your current working directory.

Example input file:

1234560

1234561

1234562

1234563

1234566

1234567

MICH-V-1234564

7654321A

7654321B

654321\_T

654311\_E

MICH-A-1122334.dng

MICH-A-1122333\_T.dng

MICH-V-1001234C\_E\_G\_T\_2.tiff

Output File:

The output file for this script is a comma separated values (csv) file that can be opened in excel or another spreadsheet program. The first column is the target, the second is whether or not there was any matches, the third specifies how many matches there were for that target, and the fourth and later columns are the files names that matched which the target (either the folder+filename or the entire absolute path for the file depending whether you specified **include\_full\_filepath\_names=True**. The output file will be created in your current working directory, unless you specify another directory in the output\_file= user option.

General | **return\_list\_of\_all\_files\_in\_folders.py**

This script iterates through every directory in the current working directory and outputs the names of all files in those directories to an output file, with an option of also outputting any 6 or 7 digit barcode associated with those files. It looks through a bunch of folders and returns a single list of all the files in those folders.

User options:

**include\_barcode=True**

If True, in addition to the file name, the output file will also contain the 6 or 7 digit barcode in a second column of a csv

**output\_file="All\_files\_in\_folders.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten.

Input File:

None

Output File:

A csv with either one or two columns depending on the options specified. The first column is a list of all the files found in the directories searched, the optional second column are the 6 or 7 digit barcodes associated with each of the files.

Algae | **File\_MICH\_algae\_images\_after\_verification.py**

This script takes a folder of MICH Algae images, checks that the filename is valid for an algae image, and files it in the appropriate archive folder, creating the folder if it doesn't exist. This script will not recognize image archives if they are named like those served to the library, only standard archive names such as “1234567 - 1234567”. The image files needs to be in the current working directory (and the folder where the image archives are stored). This script will not move files with invalid algae names or duplicates with the same name in the archives.

User options:

**ext=".dng"**

The file extension for the type of file you want to process in quotes. This is case sensitive.

**output\_file="Algae\_filing\_report.csv" #Name of the output file that will be generated**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

**errors="Errors.txt"**

The name of an error file that will be created, listing all of the files that were not processed because they are not valid MICH algae filenames. This can be modified between runs if you don’t want the results of the previous run to be overwritten.

**only\_test=False | True or False.**

This is an option to run the Filing script as a test run. Unless this is set to False, the script will run, but will not create any folders or move any files. The output file will still be created and appear as if the script has actually moved the files.

**duplicate\_type\_images=True #True or False**

This is an option to copy any type specimen images (any images with \_T in them) so that they are filed normally AND copied into a folder called 'Duplicated\_type\_specimens' which will be created in the current working directory.

Input File:

None

Output Files:

This script creates an error report listing any files that were not filed because of invalid filenames. In addition, an output csv file is created with the following columns:

1) Filename 2) Whether or not the file was filled 3) Whether the appropriate archive folder already existed 4) What the appropriate archive is 5) Whether there was a copy of the image with the exact same name already in archive

Algae | **Verify\_existance\_list\_of\_algae\_filenames\_or\_barcodes\_option\_to\_copy.py**

The script takes a target list as an input file and looks through any herbarium image archive directories in the current working directory. It reports any files that match any of the targets. For each target, it reports the number of hits and the file paths for each hit.

The target list can be either a list of 6 or 7 digit barcodes in which case hits will be any file in the archive with that barcode or the target list can be a list of complete file names you are searching for, in which case hits will be only files in archives with that exact name.

There is also an option to copy matching files to a new folder.

User options:

**inputTargetPath="TargetList.txt"**

The input file with a list of targets for which you want to search.

**output\_filename="output.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

**header=True**

Change to True or False depending on if you want a header line specifying the column labels of the output file.

**include\_full\_filepath\_names=True** | True/False

True/False if you would like full file path names for any hits included in the output. If False, only the folder the image is in will be included. If True, the entire path will be output.

**copy\_image\_hits=True** | True/False

If True, all images that hit to one of the targets will be **copied** (not moved) to the folder specified below in "copy\_image\_directory".

**copy\_image\_directory="Target\_images"**

The name of the folder that images that hit a target will be copied to if copy\_image\_hits is set to True.

**require\_targets\_to\_be\_valid\_complete\_filenames=False**

If set to True, the script will verify that each target is a complete, legitimate algae filename before searching for it. Set this to false if you want to target barcodes or other kinds of targets.

Input file:

This script takes in a list of targets, one per line, as an input file. Unless you specify which directory contains your input file, python will look for the input file in your current working directory.

Example input file:

1234560

1234561

1234562

1234563

1234566

1234567

MICH-A-1234564

7654321A

7654321B

654321\_T

654311\_E

MICH-A-1122334.dng

MICH-A-1122333\_T.dng

MICH-A-1001234C\_E\_G\_T\_2.tiff

Output File:

The output file for this script is a comma separated values (csv) file that can be opened in excel or another spreadsheet program. The first column is the target, the second is whether or not there was any matches, the third specifies how many matches there were for that target, and the fourth and later columns are the files names that matched which the target (either the folder+filename or the entire absolute path for the file depending whether you specified **include\_full\_filepath\_names=True**. The output file will be created in your current working directory, unless you specify another directory in the output\_file= user option.

Algae | **change\_file\_names\_based\_on\_list.py**

#This script takes in an csv with two columns. The first is a list of old file names that have been manually curated to the corresponding name in the second column.

#If the csv indicates that the

#If saving the list from excel, be sure to save using "Comma seperated values" and NOT "CSV UTF-8 Comma Delimited"

User options:

**run\_as\_test=True | True or False**

This is an option to run the script as a test run. Unless this is set to False, the script will run, but will not rename or delete any files. The output file will still be created and appear as if the script has actually renamed the files.

**ext=".CR2"**

The file extension for the type of file you want to process in quotes. For this script, you should include and period. This is case sensitive.

**inputfile="Name-examples.csv"**

The name of the input file for which the script will look. See below for details on how to format the file.

**outputfile="Change\_name\_report.txt"**

The name of the output file that will be created. See below for details on the output file.

**prefix="MICH-A-"**

Specify here what comes before the barcodes in the files you would like to process. This option is included because some users may have an input file that does not include prefixes for the changes they would like to make. However, if your input file contains complete file name changes, this can be set to “” (A set of empty quotes).

**delete\_file="Deleted"**

This option specified the notation you used to specify which files should be deleted. If the entry in the input file matches this, the file specified in column one of the input file will be deleted.

Input file:

The input file for this script is a csv (comma separated value) file with exactly two columns. The first column should specific the current barcode and suffixes of files that you would like to rename (or complete filenames). The second column should specify what barcode and suffixes (or complete filenames) to which you would like those names changed. You can specify files that you would like deleted by the script by using whatever notation specified in the option delete\_file (see User options) in the second column.

Example input file:

737823-01,Deleted

737823-02,737823

737824-01,737824

737824-02,Deleted

737825-01,737825

737825-02,Deleted

737826-01,737826

737826-02,Deleted

737832-01,737832

737832-02,Deleted

737833-01,737833

737833-02,Deleted

676960-01,676960\_1

676960-02,676960\_2

677375-01,Deleted

677375-02,677375

677420-01,677420\_E\_1

677420-02,677420\_E\_2

679255-01,679255\_1

679255-02,679255\_2

679605-01,679605\_1

679605-02,679605\_2

Output file:

The output file for this script is a text file listing all the changes the script made (or would have made if running as a test).

Algae | **curate\_algae\_image\_archives.py**

This script checks that all images in archives are in the correct directory and have valid algae file names for MICH algae specimens.

Run this script while your current working directory the folder than contains the algae image archives you want to investigate.

User options:

**which\_extension="dng"**

This option specifies what file type you want to curate. The script will flag any files that do not have this extension during the curation process. Be case sensitive and don’t include a period.

**output\_file="report\_algae\_curation.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

Input file:

None

Output file:

The output file for this script is a csv with three columns. One column is produced for each issue found by the script. The first indicates the name of the file at issue, the second contains the directory that file is currently in, and the third is a brief note about the issue.

Algae | **list\_barcode\_nums\_in\_archive\_dir.py**

This script makes a list of barcodes without prefixes, extensions or suffix codes as well as the complete file names for the files in the current working directory in which it is run.

User options:

**output\_file\_name="file\_list.csv"**

This is the output file that contains a list of barcodes from files in the current working directory.

Input file:

None

Output file:

The output file for this script is a csv with two columns. The first column is a list of barcode numbers for the files in the current working directory (stripped of all prefixes, suffixes, and file extensions). The second column are the corresponding complete file names for files in the current working directory.

Algae | **replace\_mich\_\_with\_mich-a-.py**

This script goes through all the files in the current working directory and renames any files name with “MICH\_” in them to be “MICH-A-“ which should be present in all MICH algae image file names.

Input files:

None

Output files:

None

Vascular\_plants | **rename\_image\_codes\_lowercase\_to\_uppercase.py**

This script goes through each image archive folder in the current working directory and renames files so that all instances of \_e are changed to \_E and \_g are changed to \_G.

**only\_test=False** | True or False

When this is set to True, this script will print a statement for all the things would move, but will not actually change any file names

**output\_file="renamed\_files.csv"**

Name of the outout csv it makes with the old and new file names

Input file:

None

Output file:

The output file is a text file that contains a list of all the changes made (or that would have been made if it was run as a test).

Vascular\_plants | **File\_vascular\_plant\_images\_into\_archives\_checking\_if\_duplicates.py**

Files vascular plant images into archive folders after checking that the names are properly formatted MICH vascular plant names and checking for duplicates in both existing normal archive folders and folders formatted to be served to the library. It won’t file anything if that name already exists in an archive folder.

Run so that your current working directory contains the vascular plant image archives. Images to file should also be in the current working directory.

Creates a report (a spreadsheet of **tab** separate values) of what if did OR would do if running a test.

User options:

**only\_test=True** | True or False

This is an option to run the Filing script as a test run. Unless this is set to False, the script will run, but will not create any folders or move any files. The output file will still be created and appear as if the script has actually moved the files.

**enable\_reading\_metadata=False** | True or False

Requires the program Exifread 2.1.2 to be installed for the version of python you are using. If this is needed, the module can be installed with pip. Contact IT about how to do this. Allows the image creation date for duplicated files to be output. With option will do nothing if **overwrite\_all\_copy\_files\_with\_ones\_in\_base\_dir** is set to True. If set to False, the script will still run, even if Exifread is not installed.

**image\_extension=".dng"**

The file extension for the type of file you want to process in quotes. This is case sensitive.

Include a period.

**output\_file="Record\_of\_images\_filed.tsv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten. See below for details on the output file.

**overwrite\_all\_copy\_files\_with\_ones\_in\_base\_dir=False**

If set to true, Image files will be filed, overwritting those that are currently in those dirs. Be careful with this.

Input files:

None

Output files:

The output file for this script is a file of tab separated values that can be opened in excel. The spreadsheet contains eight columns. Which are as follows: 1) The filename of the file 2) The barcode of the file followed by a comma 3) whether or not the file was filed (or would have been if a test) 4) Whether or not there was an issue with the file name 5) The location to which the file should be filed 6) Whether or not the image file is a duplicate to a file already in the archives 7) If duplicated, the date the image already in the archives was taken (Requires exifread module to be installed) 8) If duplicated, the date the image that would be filed was taken (Requires exifread module to be installed).

Vascular\_plants | **Verify\_existance\_list\_of\_vascular\_plant\_filenames\_option\_to\_copy.py**

The script takes a target list as an input file and looks through any herbarium image archive directories in the current working directory. It reports any files that match any of the targets. For each target, it reports the number of hits and the file paths for each hit.

The target list can be either a list of 7 digit barcodes in which case hits will be any file in the archive with that barcode or the target list can be a list of complete file names you are searching for, in which case hits will be only files in archives with that exact name.

There is also an option to copy matching files to a new folder.

Your current working directory should be the directory with the vascular image archives that you would like to search.

User options:

**inputTargetPath="TargetList.txt"**

The input file with a list of targets for which you want to search.

**output\_filename="output.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

**header=True** | True/False

Change to True or False depending on if you want a header line specifying the column labels of the output file.

**include\_full\_filepath\_names=True** | True/False

True/False if you would like full file path names for any hits included in the output. If False, only the folder the image is in will be included. If True, the entire path will be output.

**copy\_image\_hits=False** | True/False

If True, all images that hit to one of the targets will be **copied** (not moved) to the folder specified below by **copy\_image\_directory**.

**copy\_image\_directory="Target\_images"**

The name of the folder that images that hit a target will be copied to if copy\_image\_hits is set to True.

Input files:

The input file with a list of MICH vascular plant image file names (or barcodes) for which you would like to search. This script takes in a list of targets, one per line, as an input file. Unless you specify which directory contains your input file, python will look for the input file in your current working directory.

Example input file:

1234560

1234561

1234562

1234563

1234566

1234567

MICH-V-1122334.dng

MICH-V-1122333\_T.dng

MICH-V-1001234C\_E\_G\_T\_2.tiff

Output files:

The output file for this script is a comma separated values (csv) file that can be opened in excel or another spreadsheet program. The first column is the target, the second is whether or not there was any matches, the third specifies how many matches there were for that target, and the fourth and later columns are the files names that matched which the target (either the folder+filename or the entire absolute path for the file depending whether you specified **include\_full\_filepath\_names=True**. The output file will be created in your current working directory, unless you specify another directory in the output\_file= user option.

Vascular\_plants | **curate\_vascular\_image\_archives.py**

This script checks that all images in archives are in the correct directory and have valid vascular plant file names for MICH vascular plant specimens.

Run this script while your current working directory the folder than contains the vascular plant image archives you want to investigate.

User options:

**which\_extension="dng"**

This option specifies what file type you want to curate. The script will flag any files that do not have this extension during the curation process. Be case sensitive and don’t include a period.

**output\_file="report\_curation.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

Input file:

None

Output file:

The output file for this script is a csv with three columns. One column is produced for each issue found by the script. The first indicates the name of the file at issue, the second contains the directory that file is currently in, and the third is a brief note about the issue.

General | **Unfile\_list\_of\_barcodes.py**

This script takes a list of barcode numbers or full file names, looks through herbarium archives and moves any targets out of the archives to a new folder. Then, those files can be manually checked and deleted as necessary.

User options:

**inputTargetPath="TargetList.txt"**

The input file with a list of MICH Image file names or barcodes for which to search.

The script will look for this file in the current working directory, unless another location is specified.

**output\_filename="unfile\_output.csv"**

The name of the file that this script will output.

**include\_full\_filepath\_names=True** | True/False

Specifies whether or not you would like full file path names for any hits included in the output. If False, only the folder the image is in will be included, rather than the full file path.

**only\_test=True |** True/False

Will create output but not actually move any files if set to True.

**unfile\_directory="Unfiled\_images"**

The name of the folder that images will be moved into.

Input files:

The input file with a list of MICH barcodes (or complete image file names) for which you would like to search. This script takes in a list of targets, one per line, as an input file. Unless you specify which directory contains your input file, python will look for the input file in your current working directory.

Example input file:

1234560

1234561

1234562

1234563

1234566

1234567

MICH-V-1122334.dng

MICH-V-1122333\_T.dng

MICH-V-1001234C\_E\_G\_T\_2.tiff

Output files:

The output file for this script is a comma separated values (csv) file that can be opened in excel or another spreadsheet program. The first column is the target, the second is whether or not there was any matches, the third specifies how many matches there were for that target, and the fourth and later columns are the files names that matched which the target (either the folder+filename or the entire absolute path for the file depending whether you specified **include\_full\_filepath\_names=True**. The output file will be created in your current working directory, unless you specify another directory in the output\_file= user option.

Fungi | **Verify\_existance\_list\_of\_fungi\_filenames\_option\_to\_copy.py**

The script takes a target list as an input file and looks through any herbarium image archive directories in the current working directory. It reports any files that match any of the targets. For each target, it reports the number of hits and the file paths for each hit.

The target list can be either a list of barcodes in which case hits will be any file in the archive with that barcode or the target list can be a list of complete file names you are searching for, in which case hits will be only files in archives with that exact name.

There is also an option to copy matching files to a new folder.

Your current working directory should be the directory with the vascular image archives that you would like to search.

User options:

**inputTargetPath="TargetList.txt"**

The input file with a list of targets for which you want to search.

**output\_filename="output.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

**header=True** | True/False

Change to True or False depending on if you want a header line specifying the column labels of the output file.

**include\_full\_filepath\_names=True** | True/False

True/False if you would like full file path names for any hits included in the output. If False, only the folder the image is in will be included. If True, the entire path will be output.

**copy\_image\_hits=False** | True/False

If True, all images that hit to one of the targets will be **copied** (not moved) to the folder specified below by **copy\_image\_directory**.

**copy\_image\_directory="Target\_images"**

The name of the folder that images that hit a target will be copied to if copy\_image\_hits is set to True.

Input files:

The input file with a list of MICH fungi image file names (or barcodes) for which you would like to search. This script takes in a list of targets, one per line, as an input file. Unless you specify which directory contains your input file, python will look for the input file in your current working directory.

Example input file:

1234560

1234561

1234562

1234563

1234566

1234567

MICH-F-1122334.dng

MICH-F-11\_T.dng

MICH-F-120\_2.dng

MICH-F-1001234C\_E\_G\_T\_2.tiff

Output files:

The output file for this script is a comma separated values (csv) file that can be opened in excel or another spreadsheet program. The first column is the target, the second is whether or not there was any matches, the third specifies how many matches there were for that target, and the fourth and later columns are the files names that matched which the target (either the folder+filename or the entire absolute path for the file depending whether you specified **include\_full\_filepath\_names=True**. The output file will be created in your current working directory, unless you specify another directory in the output\_file= user option.

Fungi | **File\_fungi\_images\_into\_archives\_checking\_if\_duplicates.py**

Files fungi images into archive folders after checking that the names are properly formatted MICH fungi names and checking for duplicates in existing archive folders. It won’t file anything if that name already exists in an archive folder, unless you set it to overwrite in the script.

Run so that your current working directory contains the vascular plant image archives. Images to file should also be in the current working directory.

Creates a report (a spreadsheet of **tab** separate values) of what if did OR would do if running a test.

User options:

**only\_test=True** | True or False

This is an option to run the Filing script as a test run. Unless this is set to False, the script will run, but will not create any folders or move any files. The output file will still be created and appear as if the script has actually moved the files.

**enable\_reading\_metadata=False** | True or False

Requires the program Exifread 2.1.2 to be installed for the version of python you are using. If this is needed, the module can be installed with pip. Contact IT about how to do this. Allows the image creation date for duplicated files to be output. With option will do nothing if **overwrite\_all\_copy\_files\_with\_ones\_in\_base\_dir** is set to True. If set to False, the script will still run, even if Exifread is not installed.

**image\_extension=".dng"**

The file extension for the type of file you want to process in quotes. This is case sensitive.

Include a period.

**output\_file="Record\_of\_images\_filed.tsv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten. See below for details on the output file.

**overwrite\_all\_copy\_files\_with\_ones\_in\_base\_dir=False**

If set to true, Image files will be filed, overwriting those that are currently in those dirs. Be careful with this.

Input files:

None

Output files:

The output file for this script is a file of tab separated values that can be opened in excel. The spreadsheet contains eight columns. Which are as follows: 1) The filename of the file 2) The barcode of the file followed by a comma 3) whether or not the file was filed (or would have been if a test) 4) Whether or not there was an issue with the file name 5) The location to which the file should be filed 6) Whether or not the image file is a duplicate to a file already in the archives 7) If duplicated, the date the image already in the archives was taken (Requires exifread module to be installed) 8) If duplicated, the date the image that would be filed was taken (Requires exifread module to be installed).

Fungi | **curate\_vascular\_image\_archives.py**

This script checks that all images in archives are in the correct directory and have valid fungi file names for MICH fungi specimens.

Run this script while your current working directory the folder than contains the fungi image archives you want to investigate.

User options:

**which\_extension="dng"**

This option specifies what file type you want to curate. The script will flag any files that do not have this extension during the curation process. Be case sensitive and don’t include a period.

**output\_file="report\_curation.csv"**

The name of the output file that this script will create. This can be modified between runs if you don’t want the results of the previous run to be overwritten, otherwise, if the same output file name is specified for two runs, the older version will be overwritten.

Input file:

None

Output file:

The output file for this script is a csv with three columns. One column is produced for each issue found by the script. The first indicates the name of the file at issue, the second contains the directory that file is currently in, and the third is a brief note about the issue.