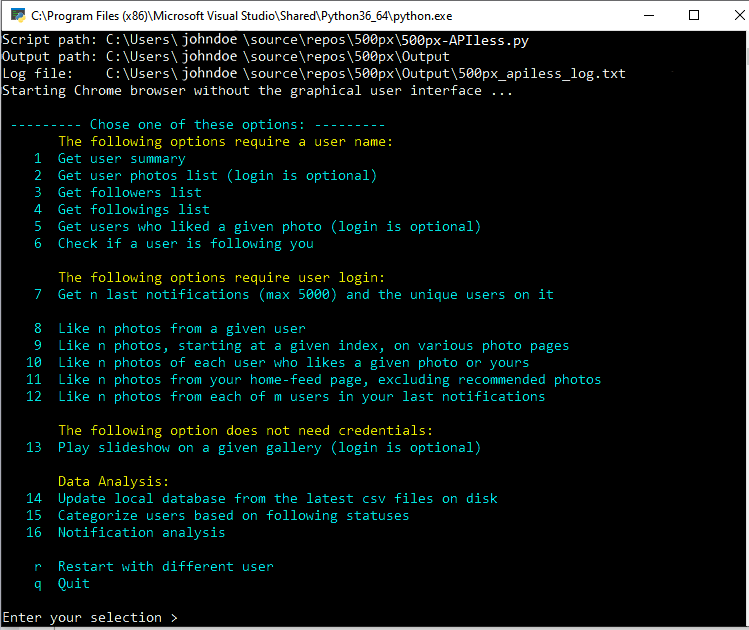
OVERVIEW

Main Menu



The program has four main functionalities:

**Data collection**

The first 7 options are for data collection. The results of these tasks are saved on disk in CSV and HTML formats.

CSV files are used for statistical analysis, and for the automated processes.

HTML files are used for presentation, which are automatically shown after a data collection task is completed.

They are also served as a tool to sort, filter or search the data

**Automated processes**

The options 8 to 12 are the automated processes, or bots, that will perform some actions.

**Entertainment**

Option 13 allows you to play the slideshow of the photos from a preselected or customized gallery

**Data analysis**

To falicitate the analysis tasks, we create a local SQLite database using the csv files obtained in the data collection processes.

We have the analysis for photos, users, and notifications:

Photos:

* List of top photos in terms of Highest Pulse, Views, Likes, Comments, and Featured Galleries
* List of all photos with the detail info arranged in columns, where you can sort, filter, or search

Followers and followings users are categorized into 3 groups:

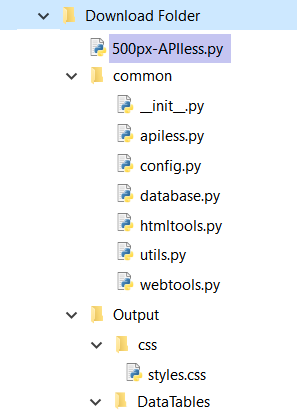
* Your followers that you are also following (reciprocal following)
* Your followers that you do not follow
* Users that you are following but they do not follow you.

Notifications are collected bit by bit over times are combined together into one table in database, which is used for creating statistics, such as:

* All unique users that had interactions with your photos
* For each user, the total number the interactions, total numbers of likes, comments, and featuring your photos
* The following status between you and each user

USAGE

Download and extract (clone) everything in a location on disk, say “Download folder”. You should have this folder structures:

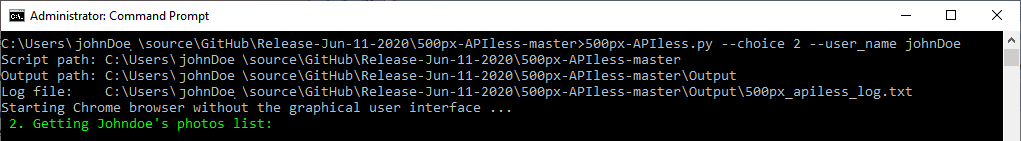


You can start the program in one of three following ways:

1. Directly from the file 500px\_APIless.py. Run the script by double-click it, and follow the main menu.
2. On the command-line window (cmd.exe, the Window Terminal). At the prompt, enter a single task.

For example, to get the photos list:

[full path/]500px-APIless.py --choice 2 --user\_name JohnDoe



1. From a shortcut to the 500px\_APIless.py script, where the Target box in the properties settings is filled with all the needed arguments.

Make many shortcuts as you like, each one for a specific task

For usage examples and a complete syntaxes of command-line switches, go to the end of this document, [here](#ComdLine).

DETAILS

Following are the detailed description of each option and some sample outputs.

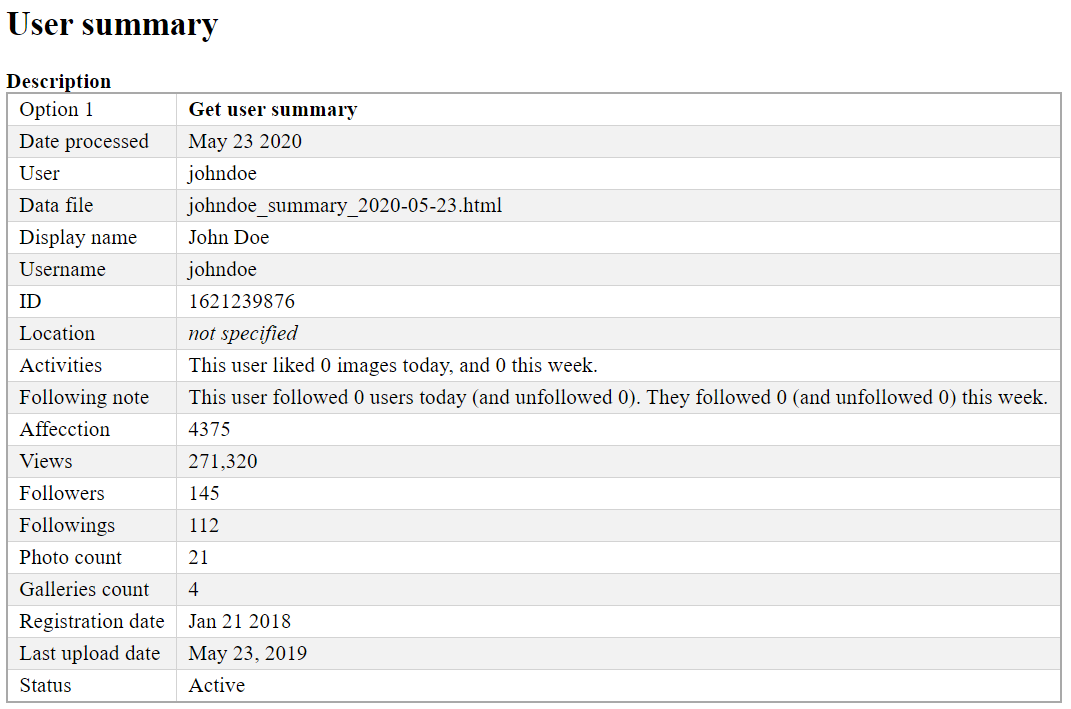
Please note that all the names, photos, titles in this document are not real. They are either auto-generated, or blurred out.

Option 1: Get user summary

Automatic processing:

* Open user home page [https://500px.com/[user\_name](https://500px.com/%5buser_name)]
* Make sure the JavaScript-rendered content is done
* Extract interested data (using built-in lxml library) and
* Use regular expression to extract the JSON part in the document body, and obtain more data from it.

Sample output:



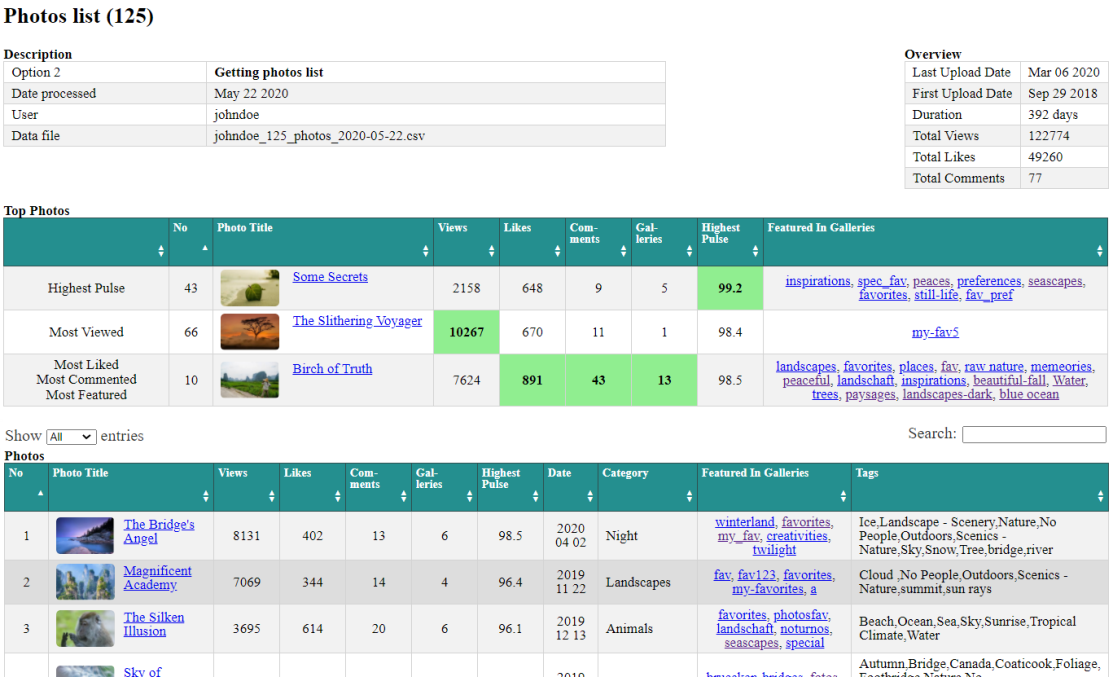
Option 2: Get User Photos list

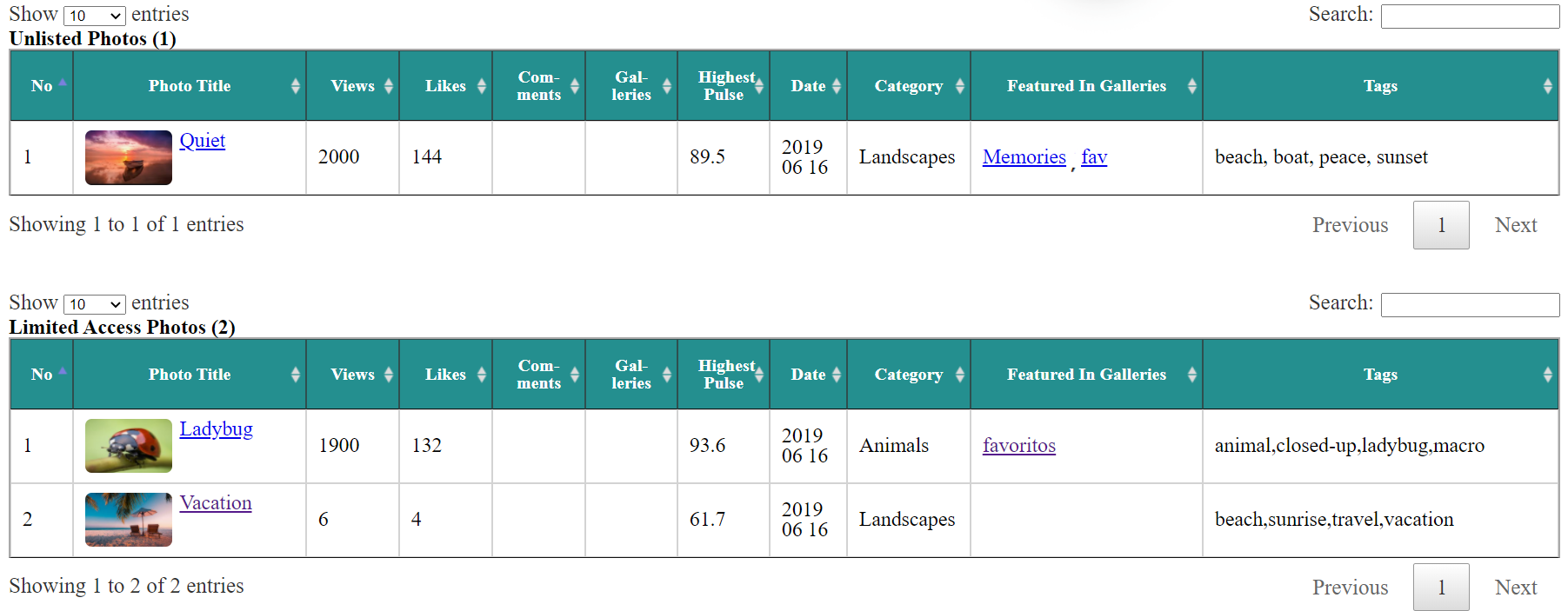
Automatic processing:

* Open user’s photo magager page for Public, Unlisted and Limited Access photos pages. Scroll down until all photos are loaded
* Extract hrefs of all photos on each of three pages.
* Open each photo in each list, parse and extract photo details.
* Write the data to csv and html files. Show the html file in the web browser when done.

Note: If the user provided credentials (optional), the list of the galleries featuring the photo will be shown, and we can extract them.

Sample output:



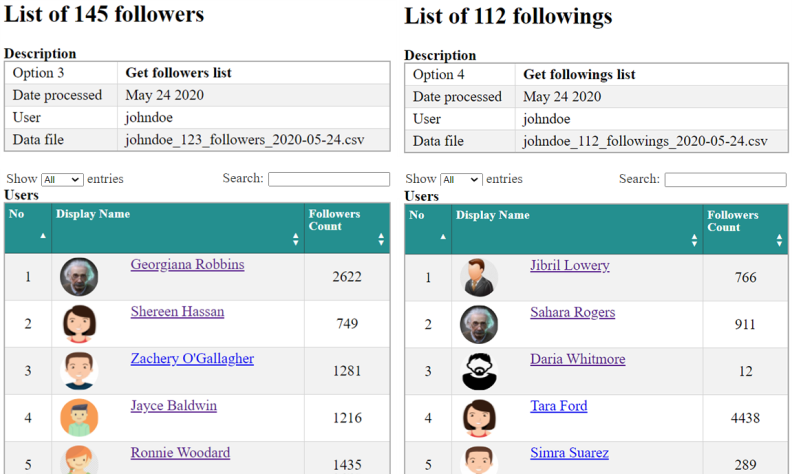


Option 3, and 4: Get Followers and Friends (Followings) list

Automatic processing:

* Open user home page, locate the text “Followers” and click on it to open a model windows containing the list of users
* Keep scrolling down until all items are loaded
* Make sure the JavaScript-rendered content is done
* Extract data and write it to a list.
* Write the list to csv and html file. Show the html file in the web browser when done.

Sample output:



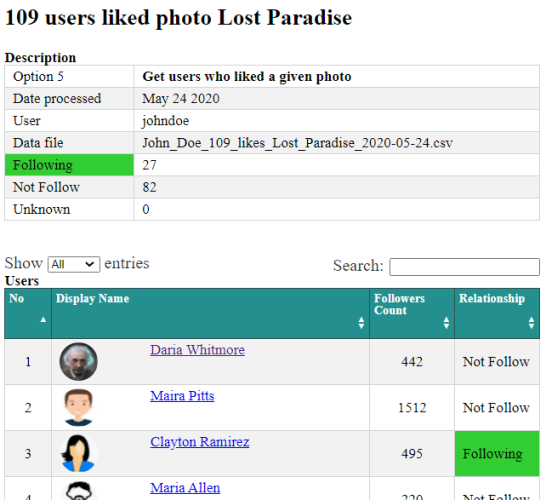
Option 5: Get a list of users who likes a given photo of yours

Automatic processing:

* Ask the user for the link to the photo. Open the link page.
* Make sure the JavaScript-rendered content is done.
* Extract photographer name and the title of the photo
* Locate the Like count number and click on it to open the modal window showing the list of all users who had liked the photo
* Keep scrolling down until all users have been loaded
* Extract info ( Display Name, User Name (with link) and Number of Followers
* Save list in to csv, html files. Show the html file in the web browser when done.

Note: If the user provided credentials (optional ), following status to each of the followers will also be extracted.

Sample output:



Option 6: Check if a user follows you

Automatic processing:

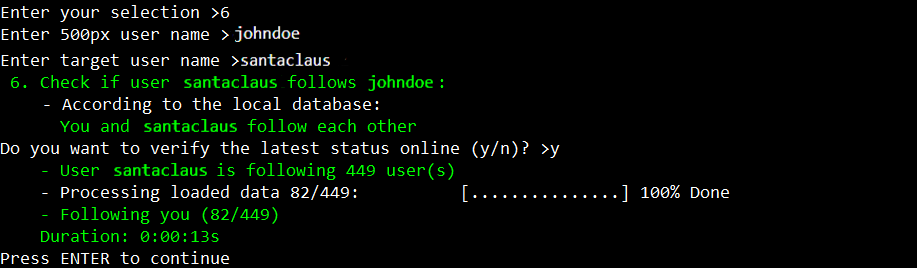
* Ask the main user’s name, and the target user’s name.
* Open the target user’s home page. Locate the “Following” text then click on it to open the modal windows for list of following users.
* Check the loaded list for the main user’s name. If found, show the result on screen then stop, if not found, scroll down to load the next

batch of users and repeat the process

* When all the users are loaded and checked without success. Show the negative result on screen.

Note: this may be a time-consuming process if the target user has many following users, and the target name happens to be near the end of the list.

Sample output:

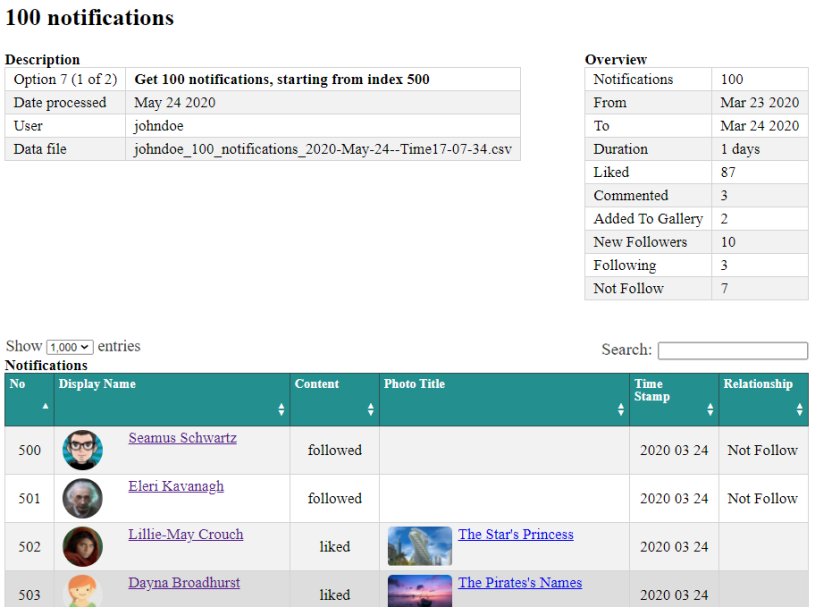


Option 7: Get n notifications details, starting from an index m

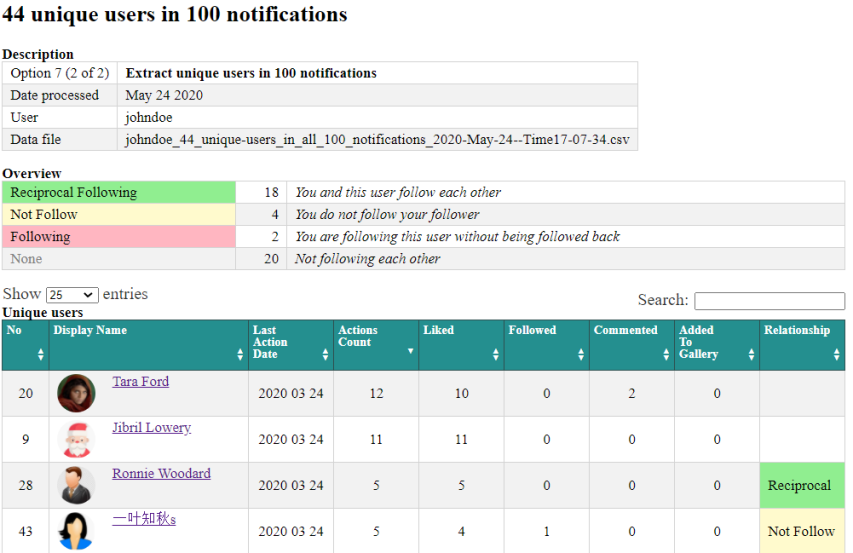
Automatic processing:

* Ask the user for desired number of notifications (n), the start index (m), and to enter the password
* Open the login page, submit the credentials. Then open the user’s notification page
* Scroll down repeatedly until all requested notifications are loaded
* Extract info, write to csv, html files.
* Count the number of appearances of each user on the list. Create another list containing the unique users and the count numbers.
* Show both html files in the web browser when done.

Sample output:



100 notifications starting from index 500



Unique users from n notifications and the number of theirs appearances

Option 8: Like n photos of a given user

Automatic processing:

* Ask user for some inputs, as in figure below.
* Login, then open the given user’s home page
* Locate the first photo, click on like-icon if it is not yet liked, or move to the next photo if it is
* Continue until the desired number of photos is reached, scrolling down the page if needed

Note about the counting of the desired number (n):

When you process a photo, if it is already liked, you will count it as if it was done by the process. For example, if the request is three photos, and you found that in the first three photos in the list, you already liked two of them, then you just need to like one more.

If you want to change this designed behavior, there is a boolean named include\_already\_liked\_photos\_in\_count.

Set it to False. Then the process will continue until all n photos are liked, or until there are no more photos to process.

Sample result:

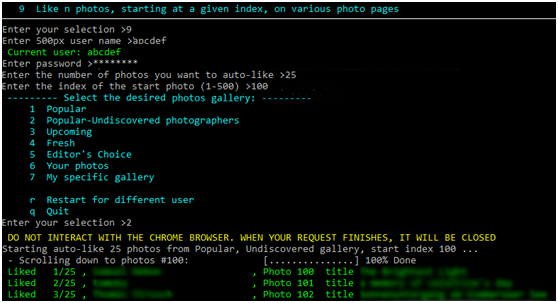


Option 9: Like n photos on various photo pages, starting from a given index

Automatic processing:

The process is similar to the previous option (option 8: Like n photos of a given user). But it starts with the given photo page instead of the user home page.

Sample result: Like 25 photos, starting at photo #100, on the page Popular of Undiscovered photographers

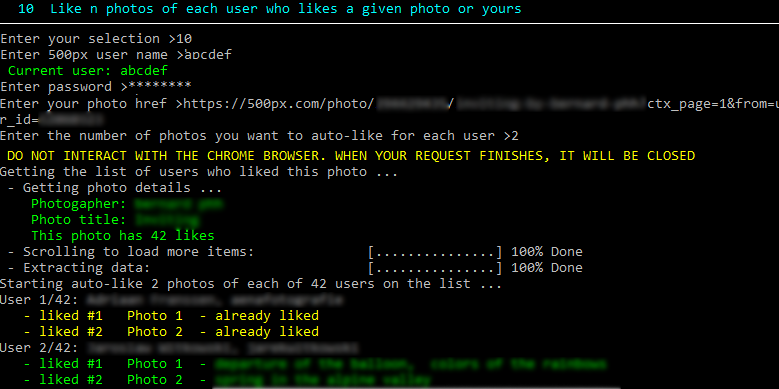


Option 10: Like n photos of each user who liked a given photo

Automatic processing:

* First step, do as in option 5 : get a list of users who likes a given photo
* Second step, for each user in the list, do as in option 8: like n photos of a given user

Sample result: Auto-like 1 photo of each user who liked a given photo, skipping already-liked photos



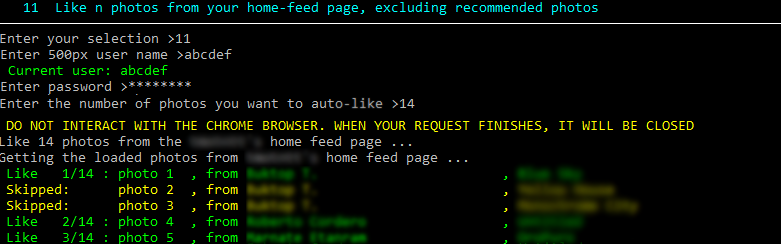
Option 11: Like n photos from the user’s home feed page, excluding the recommended photos

Automatic processing:

* Open the user home page, locate the 500px logo and click on it to open the user’s home feed page
* Get the list of all <img> elements on the page
* Remove from the list unwanted images such as thumbnails, avatar, recommended photos
* For each image elements on the list, like it if it is not yet liked, then go to the next image

Note: there are users who may upload 5, 10 photos at a time. We don’t want to like all of them. So the program will only process the first photo and will skip all consecutive photos of the same users.

Sample result: Auto-like (15) photos from the user’s home feed page, excluding the recommended photos and skipping consecutive photos of the same users

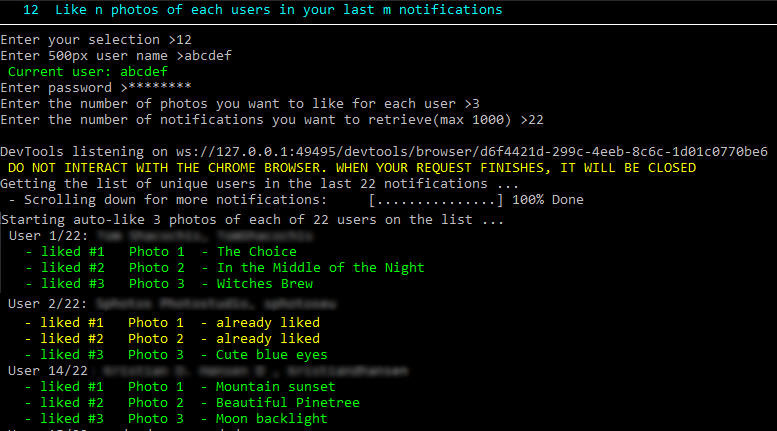


Option 12: Like n photos of each user in the last m notifications

Automatic processing: In general, there are two steps:

* First step, do as in option 6: get list of unique user who generated last **m** notifications
* Second step, for each user in the list, do as option 8: auto-like n photos of a given user

Sample result: Auto-like 2 photos of each user in the last 10 notifications



Option 13: Playing slide show on a selected gallery

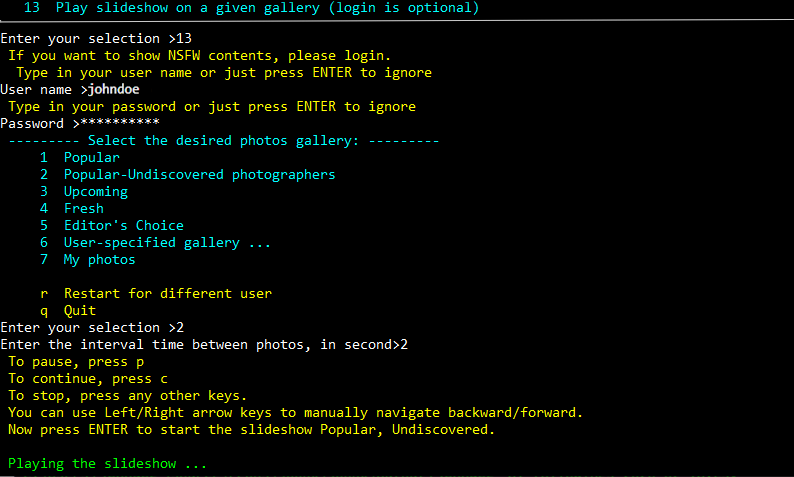
Automatic processing:

* The user is asked to select or enter a desired gallery, and the time interval between photos
* Open the given gallery page in full-screen browser, with scroll bar and info bar hidden.
* Click on the first photo of the gallery to open the photo page
* Locate the Expand icon at the top right corner, click on it to expand the photo
* After the interval time, send the Right Arrow key press to advance to the next photo
* During the slideshow, the user can press **p** to pause, **c** to continue, or any other keysto stop
* The user can use Left/Right arrow keys to navigate backward/forward any time.

Note:

* The user is asked to select one in 5 public galleries, as shown in figure below, or enter a specific photo gallery link
* Specific gallery may be a personal gallery of any user, or a category-filtered gallery. For example, a link of the popular gallery, with three categories family, people, travel selected, is: <https://500px.com/popular/family-people-travel>
* If you want to show the NSFW (not-suitable-for-work) contents, the on-screen instruction will give you a chance to login. (We expect the logged-in user already has the option “Show NSFW” turned on in the settings)
* Logged-in user will have “Your Photos” included in the preselect public galleries

Sample result:



Play a slideshow from a preselected or user-specified gallery

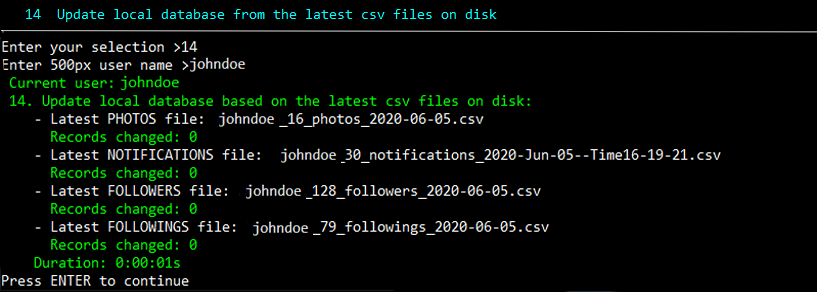
Option 14: Create/Update local database

This option takes the latest csv files on disk and create (or update) the local SQLite database.

For now, it will have 4 tables : photos, followers users, followings users, and notifications

The database is saved at Output folder under the name **500px\_[user name].db**

There are many open source, freeware visual tool that we can use to view, design and edit SQLite database file. One of them is **DB Browser for SQLite**

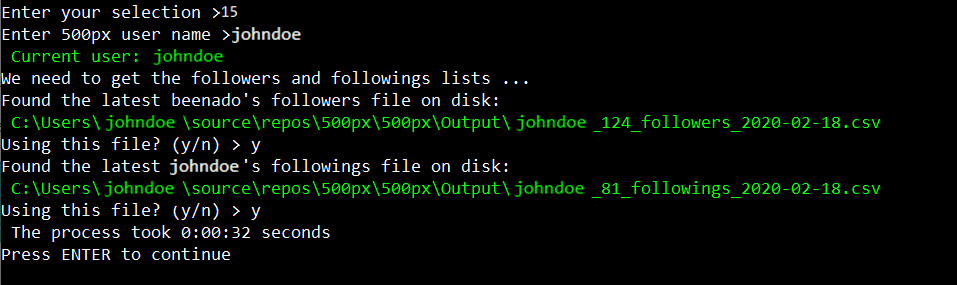


Option 15: Categorize users based on following statuses

From the Followers and Followings users lists, you categorize users into groups based on theirs following status with you.

You may choose to use the latest, previously extracted csv files, or to start from scratch

C:\Users\bernardp\Documents\GitHub\ForDocument\15.data_analysis.png



Sample output:

The results come in 4 separated lists, one for each group and one that combines all users in one list.



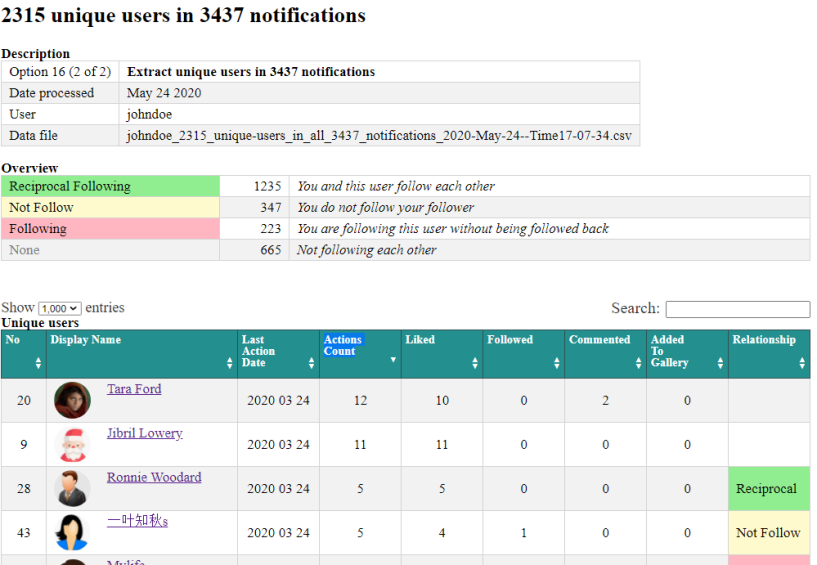
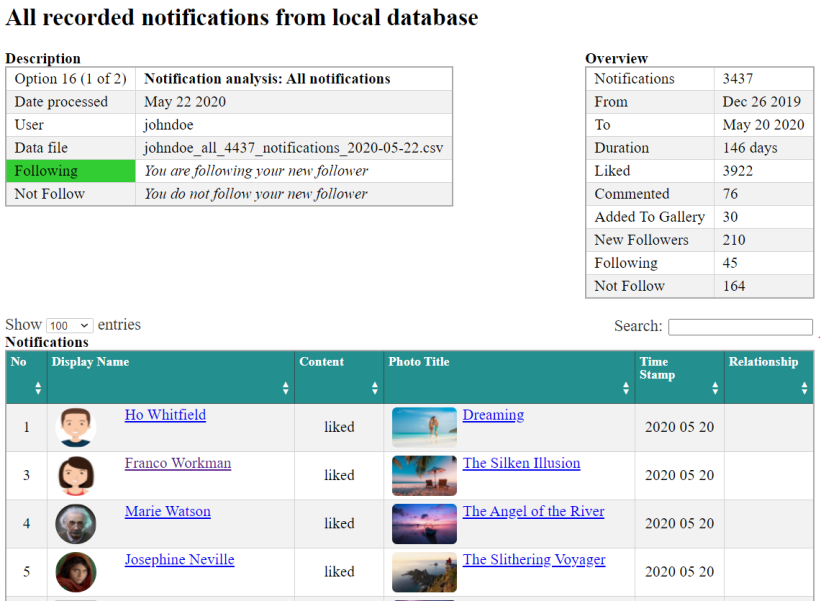
Fig. All users with theirs relationships in one table

Option 16: Notification analysis:

This option takes all the notifications csv files on disk and create (or update) the local SQLite database.

The process produces 2 files:

1. List of all notifications that have been extracted over times
2. List of the unique users and the analysis data.

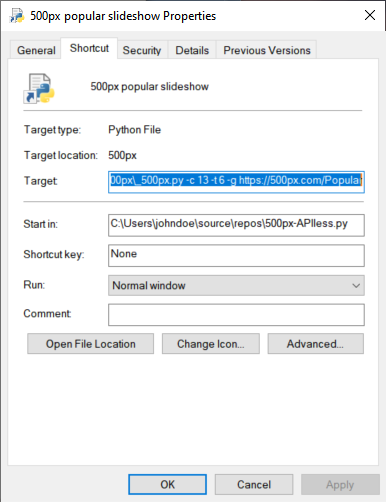


Command-line arguments

You can run un-attendedly a specific task by creating a shortcut to the script, then append the arguments to the Target box.

For example, to run a slideshow of popular photos:

C:\Users\johndoe\source\repos\500px\500px\\_500px.py -c 13 -t 6 -g https://500px.com/Popular



Shortcut settings: Create a shortcut to 500px-APIless.px,

put it on the desktop, right-click the shortcut and choose Properties, then append the arguments to the Target box

Following are available arguments switches.

There are two forms of switches, shorthand switches that start with a hyphen “-” or the full switches which start with double hyphen “--”.

You can use either forms, or mix them, as shown in example 1 below.

Short format Full format Default value

-c --choice ( 0 )

-u --user\_name (“”)

-d --password (“”)

-p --photo\_href (“”)

-g --gallery\_href (“”)

-l --number\_of\_photos\_to\_be\_liked ( 1 )

-i --index\_of\_start\_photo ( 1 )

-n --number\_of\_notifications ( 200 )

-a --target\_user\_name ( “”)

-t --time\_interval ( 4 )

Examples:

1 Get user statistics (recent activities, last upload date, registration date ...:

Full: [full path/]500px-APIless.py --choice 1 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 1 -u JohnDoe

Mixed: [full path/]500px-APIless.py --choice 1 -u JohnDoe

2 Get user photos list:

Full: [full path/]500px-APIless.py --choice 2 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 2 -u JohnDoe

3 Get followers list:

Full: [full path/]500px-APIless.py --choice 3 --user\_name JohnDoe --password \*\*\*\*\*\*

Short: [full path/]500px-APIless.py -c 3 -u JohnDoe -p \*\*\*\*\*\*

4 Get following list:

Full: [full path/]500px-APIless.py --choice 4 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 4 -u JohnDoe

5 Get a list of users who liked a given photo:

Full: [full path/]500px-APIless.py --choice 5 --user\_name JohnDoe –photo\_href https://500px.com/....

Short: [full path/]500px-APIless.py -c 5 -u JohnDoe -p “https://500px.com/....”

6 Check if a user follows you:

Full: [full path/]500px-APIless.py --choice 7 --user\_name JohnDoe -target\_user\_name santaclaus

Short: [full path/]500px-APIless.py -c 7 -u JohnDoe - a santaclaus

7 Get n last notifications details (max 1000):

Full: [full path/]500px-APIless.py --choice 7 --user\_name JohnDoe --password \*\*\*\*\*\* --number\_of\_ notifications 500

Short: [full path/]500px-APIless.py -c 7 -u JohnDoe -d \*\*\*\*\*\* -n 500

8 Like n photos from a given user:

Full: [full path/]500px-APIless.py --choice 8 --user\_name JohnDoe --password \*\*\*\*\*\*

--number\_of\_photos\_to\_be\_liked 50 –target\_user\_name JaneDoe

Short: [full path/]500px-APIless.py -c 8 -u JohnDoe -d \*\*\*\*\*\* -n 500 -a JaneDoe

9 Like n photos, starting at a given index, on various photo pages:

Full: [full path/]500px-APIless.py --choice 9 --user\_name JohnDoe --password \*\*\*\*\*\* --number\_of\_photos\_to\_be\_liked 50

--index\_of\_start\_photo 1 –gallary\_href “https://500px.com/popular/travel”

Short: [full path/]500px-APIless.py -c 9 -u JohnDoe -d \*\*\*\*\*\* -n 50 -I 1 -g https://500px.com/popular/travel

10 Like n photos of each user who likes a given photo or yours:

Full: [full path/]500px-APIless.py --choice 10 --user\_name JohnDoe --password \*\*\*\*\*\* --number\_of\_photos\_to\_be\_liked 50

--photo\_href [a photo link]

Short: [full path/]500px-APIless.py -c 10 -u JohnDoe -d \*\*\*\*\*\* -n 50 -p [a photo link]

11 Like n photos from your home-feed page, excluding recommended photos:

Full: [full path/]500px-APIless.py --choice 11 --user\_name JohnDoe --password \*\*\*\*\*\* --number\_of\_photos\_to\_be\_liked 50

Short: [full path/]500px-APIless.py -c 11 -u JohnDoe -d \*\*\*\*\*\* - l 50

12 Like n photos of each users in your last m notifications:

Full: [full path/]500px-APIless.py --choice 12 --user\_name JohnDoe --password \*\*\*\*\*\*

--number\_of\_photos\_to\_be\_liked 50 -number\_of\_notifications 200

Short: [full path/]500px-APIless.py -c 12 -u JohnDoe -d \*\*\*\* -l 50 -n 200

13 Play a slideshow:

Full: [full path/]500px-APIless.py --choice 13 --user\_name JohnDoe --time\_interval 5 –gallery\_href https://500px.com/Popular

Short: [full path/]500px-APIless.py -c 13 -u JohnDoe -t 5 -g https://500px.com/Popular

14 Update local database from the latest csv files on disk:

Full: [full path/]500px-APIless.py --choice 14 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 14 -u JohnDoe

15 Categorize users based on following statuses:

Full: [full path/]500px-APIless.py --choice 15 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 15 -u JohnDoe

16 Notifications analysis:

Full: [full path/]500px-APIless.py --choice 16 --user\_name JohnDoe

Short: [full path/]500px-APIless.py -c 13 -u JohnDoe