**Finger prints project report**

**Global Functions:**

This file defines global functions for Fingerprint project

* **getInputImage (inputPath)**

This function gets a path to image and extracts it into filename, file path and file

prefix.

* **extractFingerDataFromFilename (filename)**

This function gets a filename and extracts data of: person name, hand and finger in

hand.

**Stitcher:**

This file creates panorama images by stitching two images which have enough matching key points.

* **\_\_init\_\_(self):**

*determine if we are using OpenCV v3.X*

* **stitch (self, images, ratio=0.75, reprojThresh=4.0, showMatches=False):**

This function receives two images and checks if they can be used to create a

panorama. If panorama was created, the function will return the stitched image and

visualization when was required.

* **detectAndDescribe (self, image):**

This fun receives an image and returns a tuple with its key points and features.

* **matchKeypoints (self, kpsA, kpsB, featuresA, featuresB, ratio, reprojThresh):**

This function receives key points and features of two images to check matching key

points of a given ratio.

* **drawMatches (self, imageA, imageB, kpsA, kpsB, matches, status):**

This function receives two images and the key point matches and visualizes them.

**Client:**

This file defines the client side of Fingerprint project.

* **trim ()**

Function to crop black border from a PIL image.

* **mse (imageA, imageB):**

This function receives two images and finds the MSE-sum of the squared difference

between the two images. The lower the error, the more "similar" the images are.

* **mseUpdated (imageA, imageB):**

This function receives two images and finds the MSE-sum of the squared difference

between the two images. The lower the error, the more "similar" the images are. In

addition, the difference between the images will be printed.

* **cleanImage (img, range\_start=ZERO, range\_stop=256):**

This function receives an image and a range and cleaned it by the max value of

pixels.

* **findThresholdByRange (img, range\_start=ZERO, range\_stop=256):**

Function to find an image's threshold.

* **findThreshold (img, range\_start=ZERO, range\_stop=256):**

Function to convert an image to black and white colors according to its threshold.

* **findMatchByMSE\_Threshold (source\_img, dbPath=DBpath):**

This function receives an image and looks for matching images in db (according to

MSE value before and after threshold) to create panorama images using the stitcher

function, then it will print those images' names.

* **CreateMatchImgListByMSE (source\_img, dbPath=DBpath):**

Function that creates matching image's list according to the MSE between source

and dest images, will add to the list the images which apply to the condition

mse<THRESHOLD.

* **IsContinue (match\_img\_list):**

A function to ask the user if he wants to continue the process when no matching

images were found by MSE.

* **CreateFullImgList ():**

A function to create a full images list from database.

* **CreateMatchImgListByStitcher (source\_img, match\_img\_list):**

This function creates a list of all images from match\_img\_list that were managed to

be stitches to source image.

* **PrintDetailsOfMatches (match\_list):**

This function will print image's names from the match\_list it has received.

* **temp (source\_img, dbPath=DBpath):**

This function receives an image and looks for matching images in db (according to

MSE value before and after threshold) to create panorama images using the stitcher

function, then it will print those images' names.

* **OpenFile ():**

This function opens a dialog to get a path to image file.

**database:**

This file defines the database of fingerprints images.

* A "for" loops runs over fingerprints files in database and checks if they are standard.  
  If they are, it will save it in the database for new person, or update it for existing person.

**Main:**

This file defines the main part of fingerprints project.

* this session will include a search for matching fingerprints to a source image and displaying panorama images  
  that were managed to be created when there are enough matching key points.

**FLOW CHARTS:**

Main screen -> start -> working space ->

Upload image ->

Sharpness, brightness, rotation, noise cleaning,

Take image by machine -> Taking a photo by finger prints machine (will be developed later)

Finger and hand info, user information, DB contact for sustainability check,

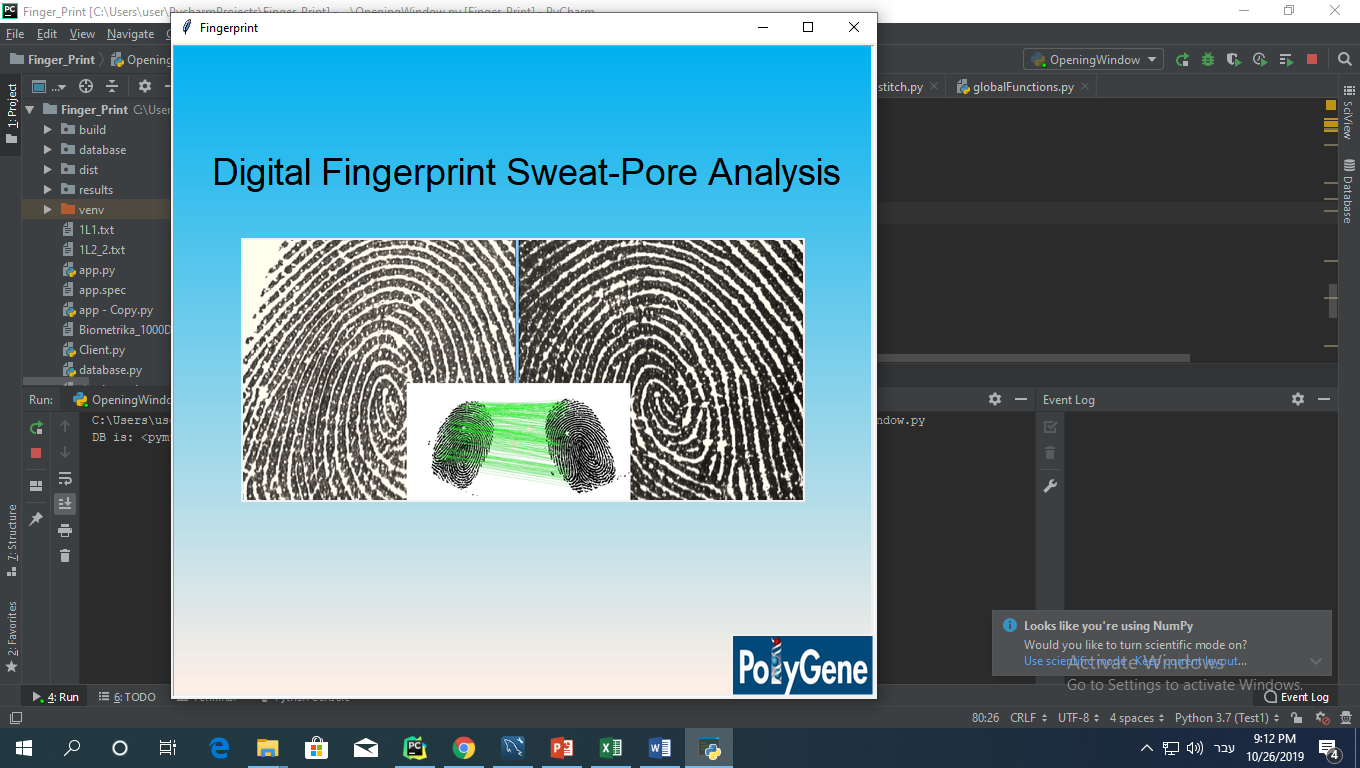
DB contact for sustainability check ->

Contact DB for sustainability testing,

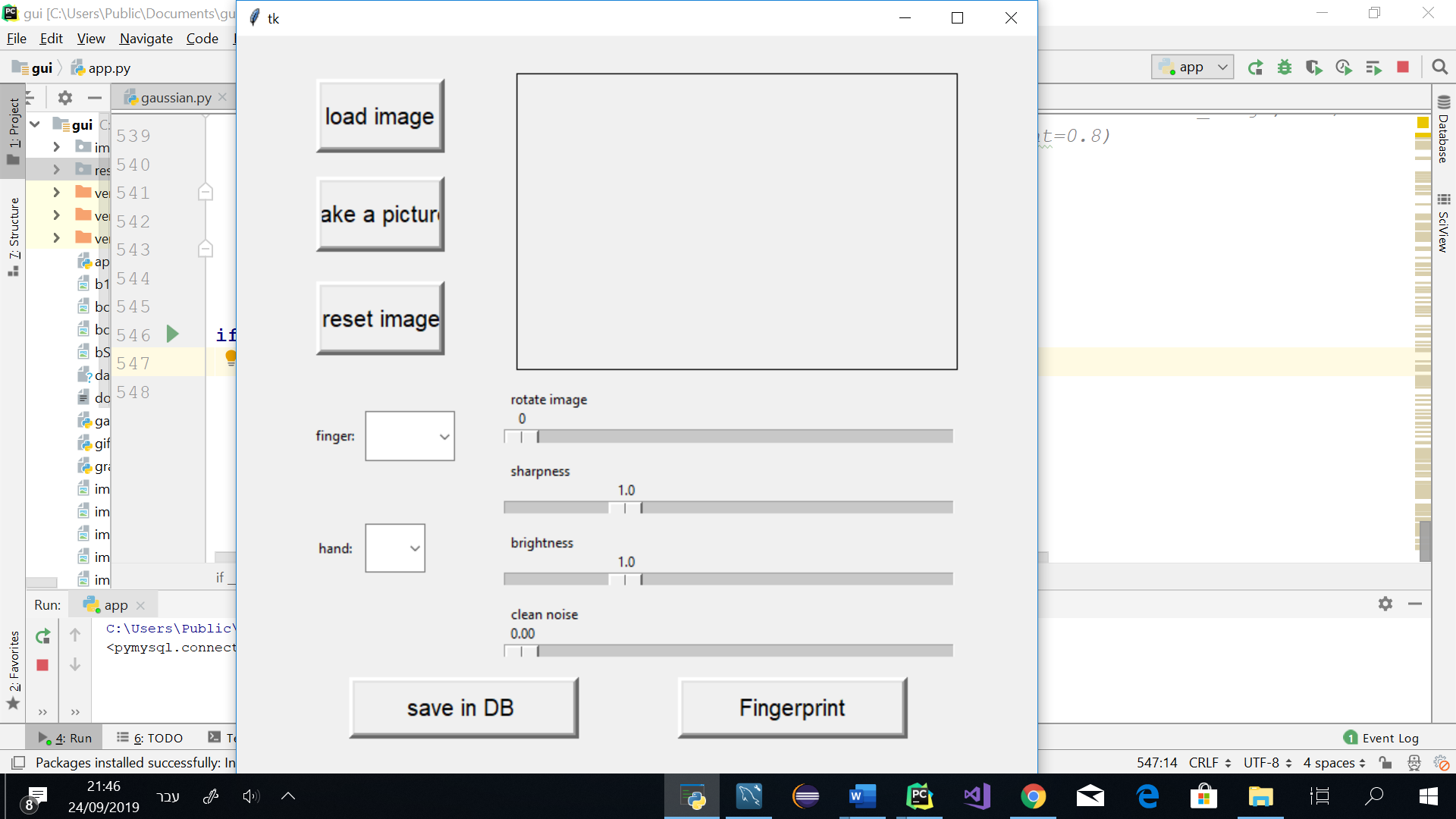
Reset image ->

Reset sharpness values, brightness, rotation, noise cleaning.

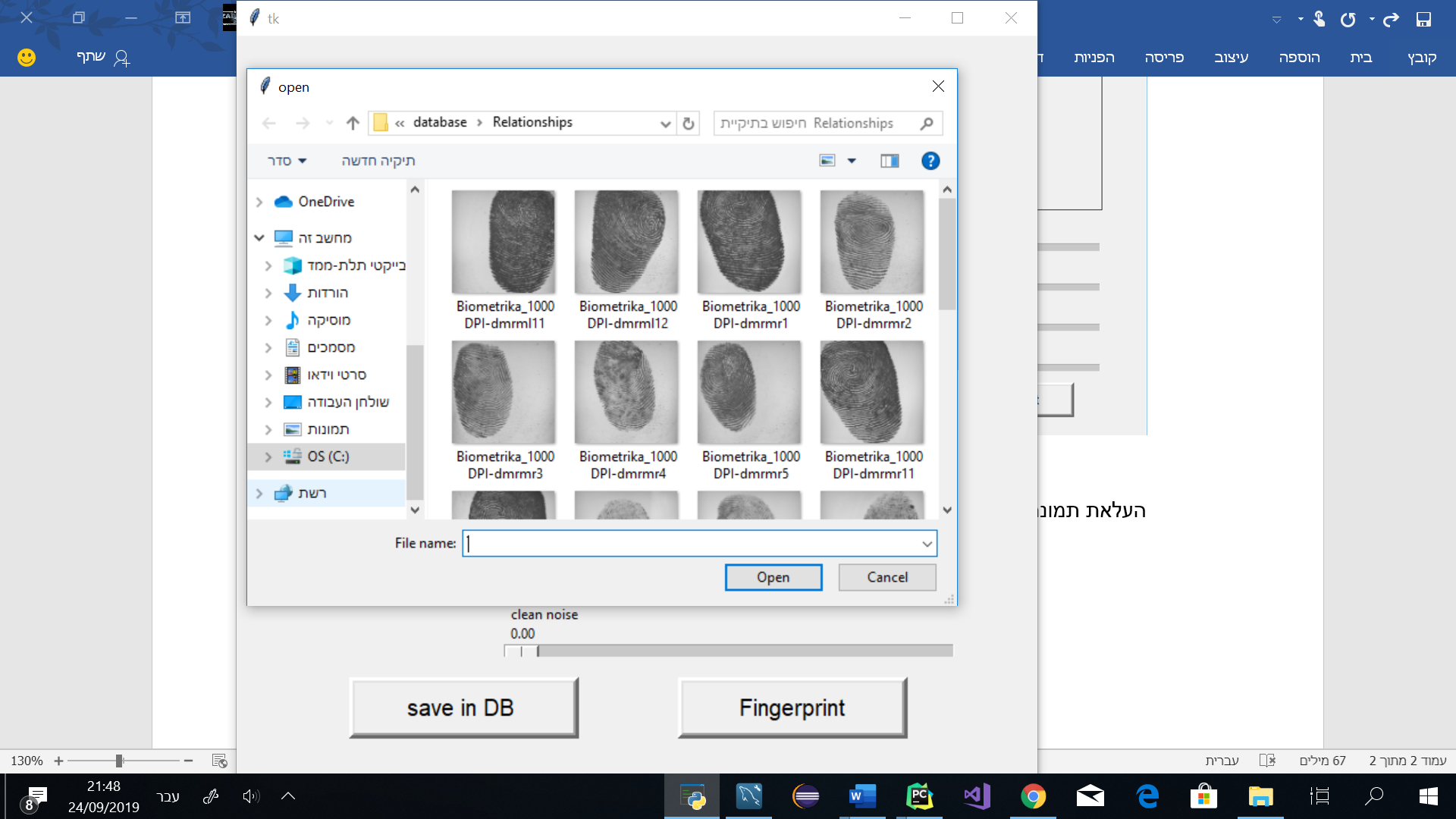
**Main screen**



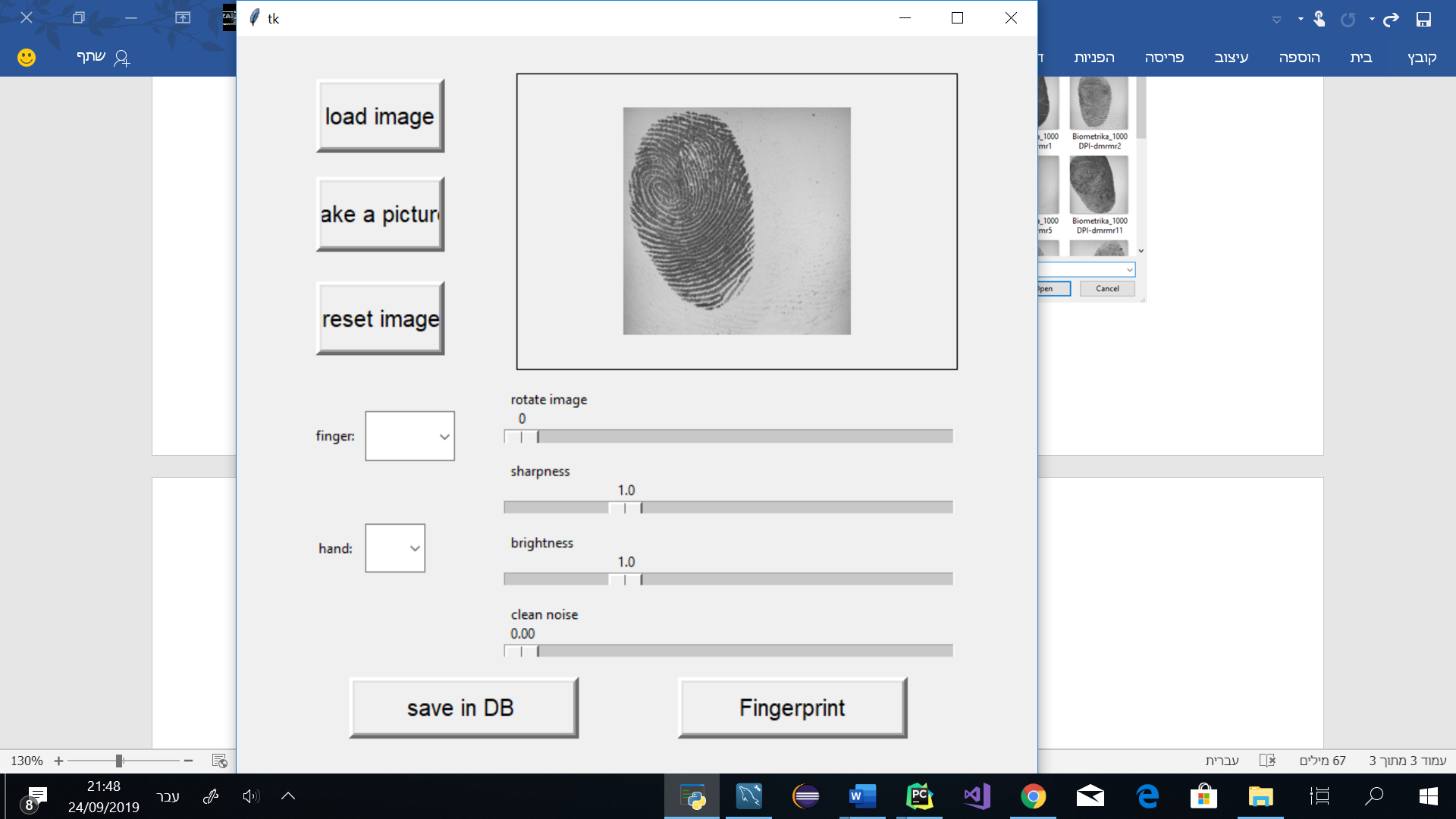
**Working space**

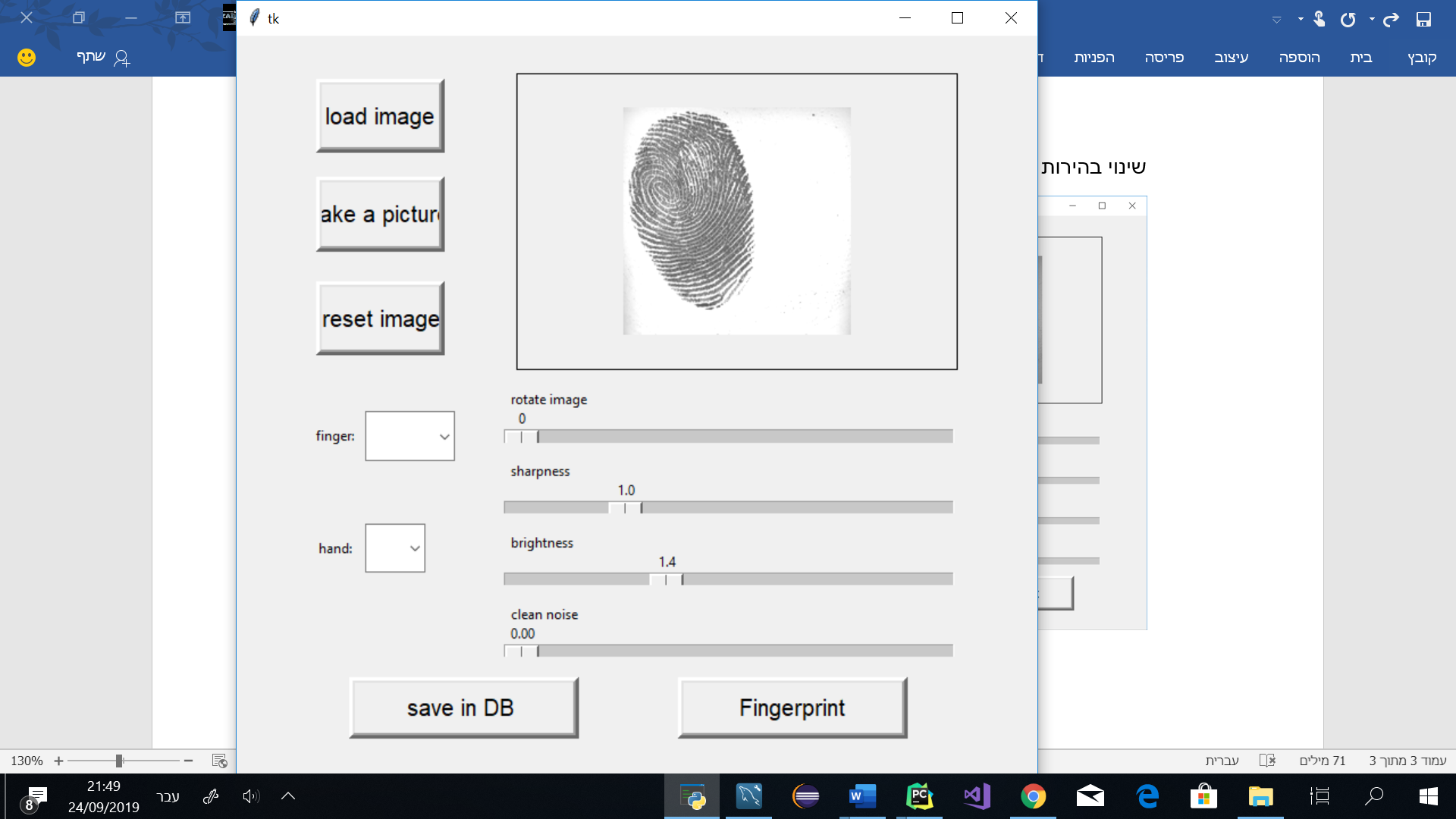


**Upload image**

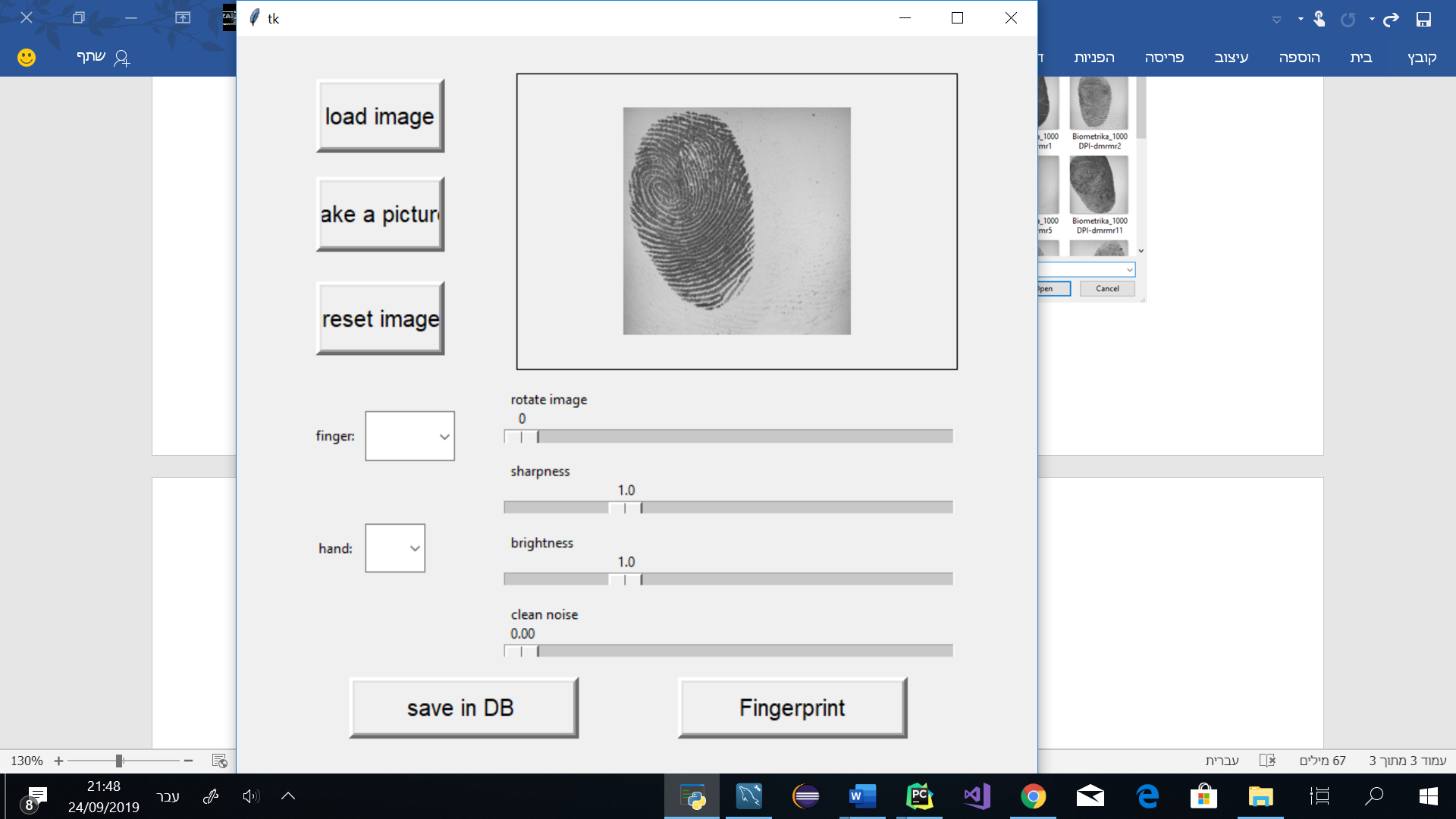


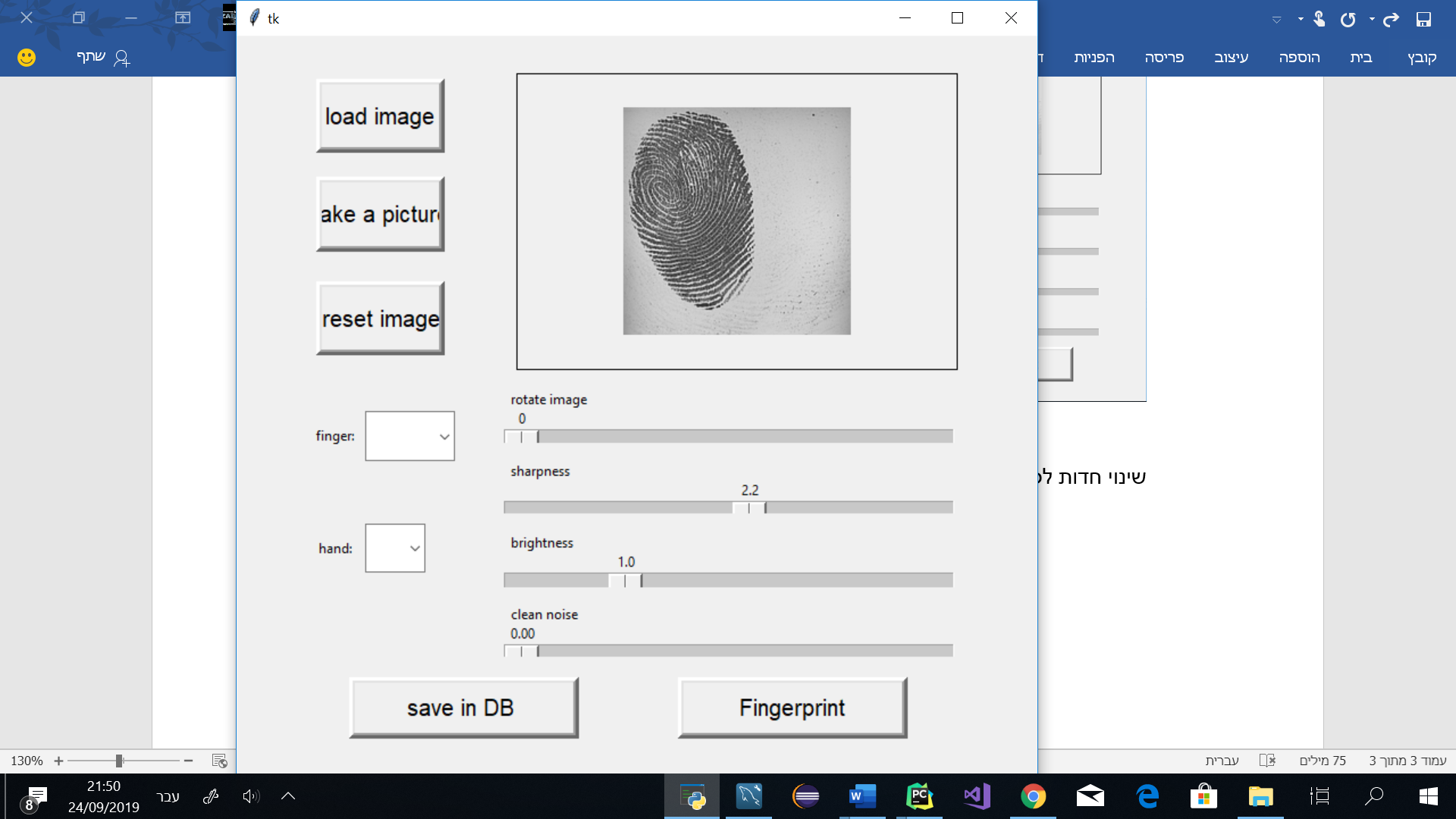
**Change brightness before and after**



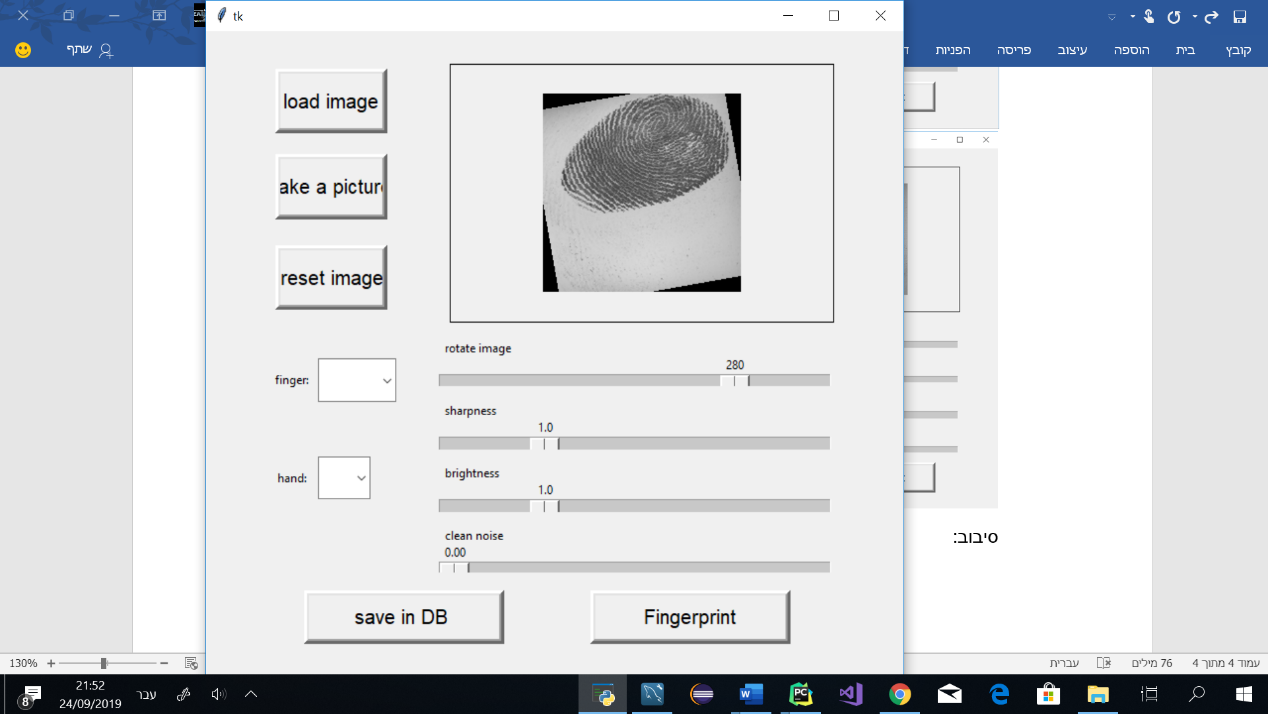


**Change sharpness before and after**

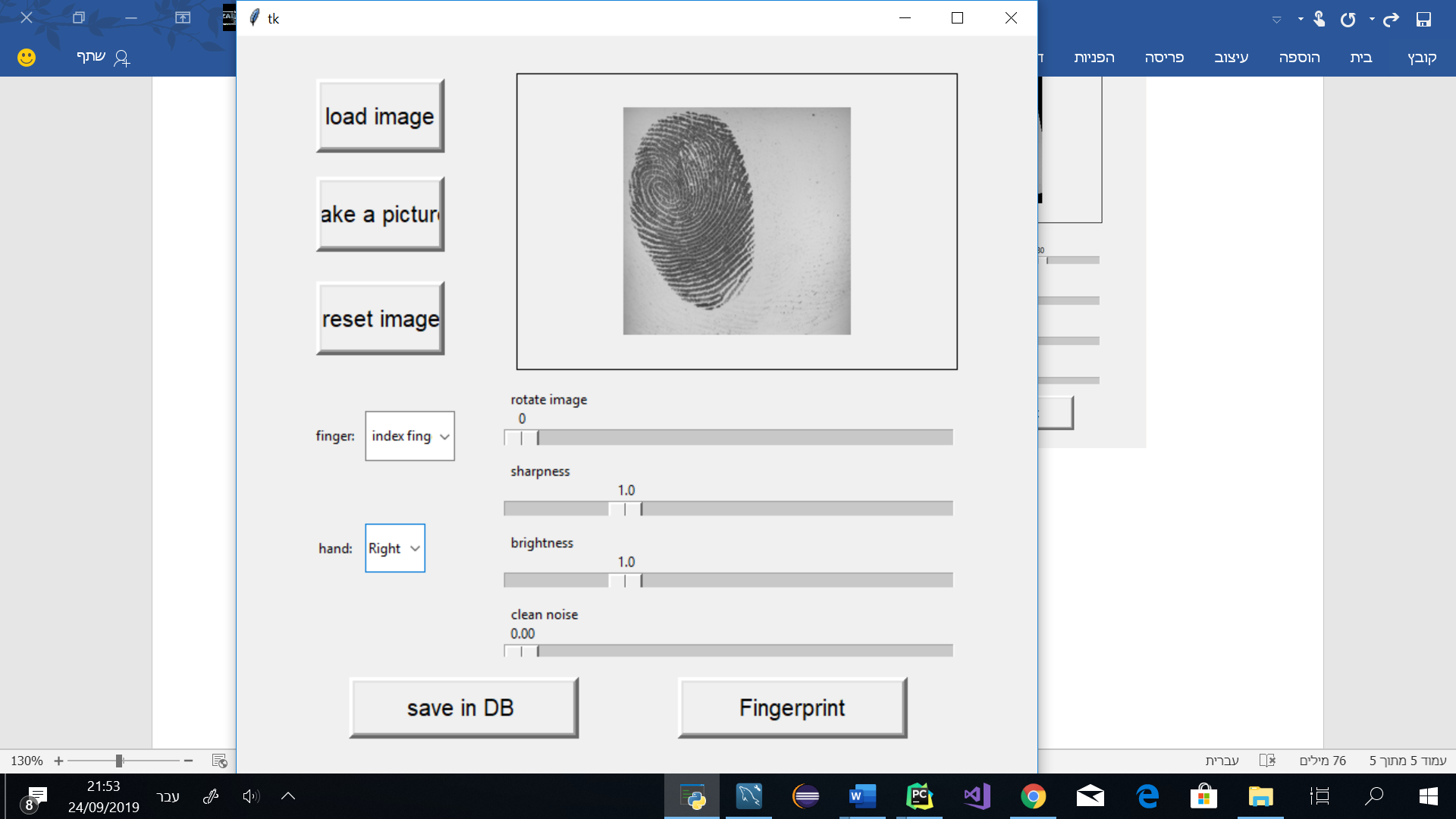


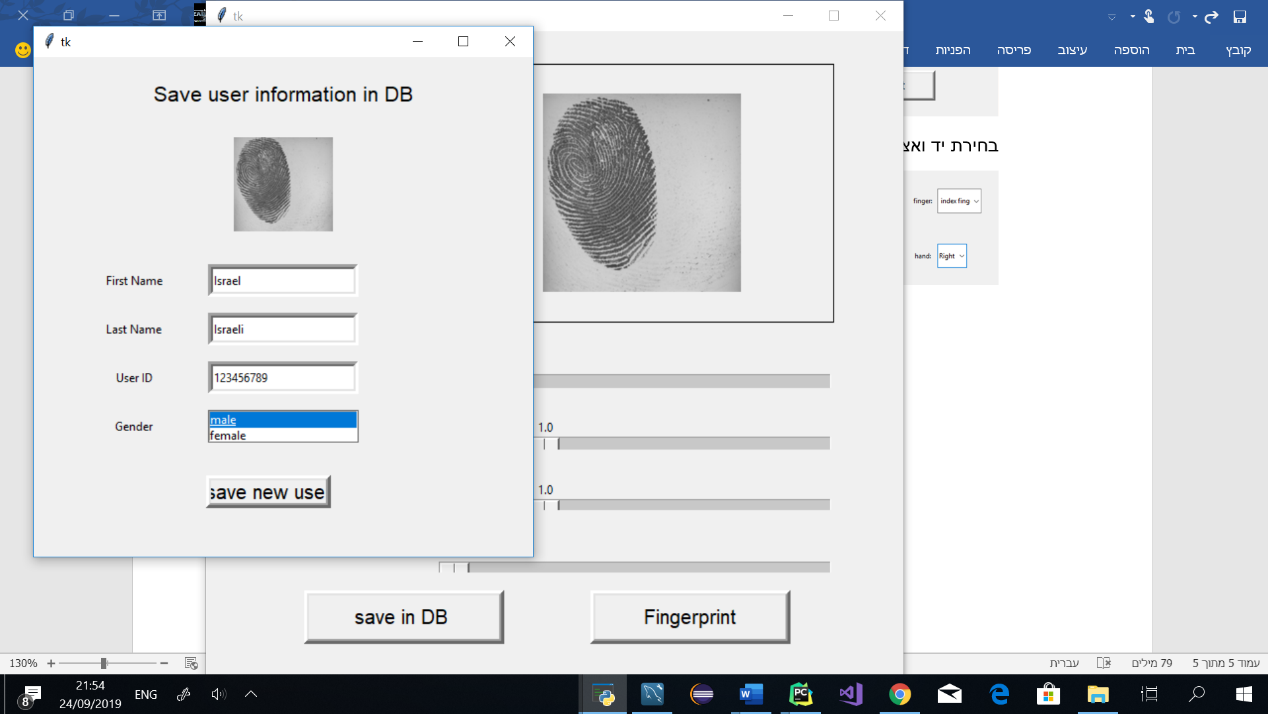


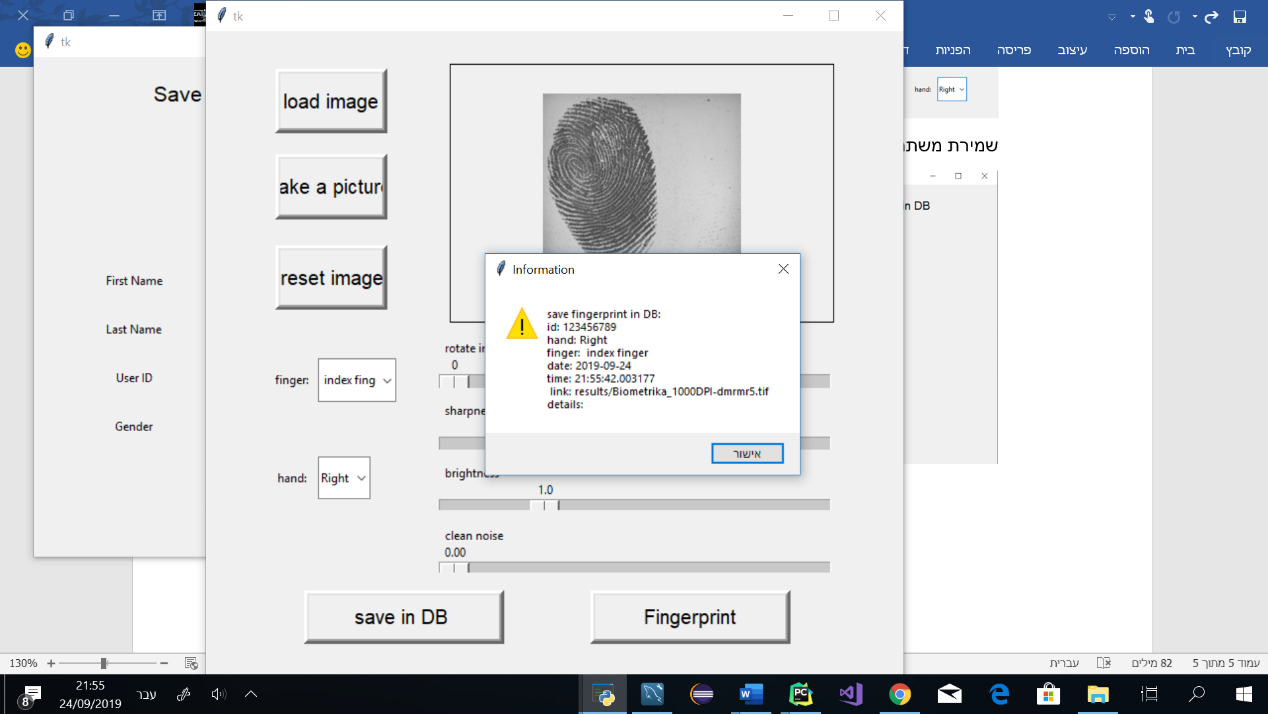
**Rotation**



**Choose hand and finger**



**Save new user**



**Check image matching**

