*REAL TIME FACE BLURING CODE*

Μαρίνης Νικόλαος 2118159

# Libraries:

* Cv2
* Numpy as np
* Keyboard
* Time
* Daytime

# Functions:

## RESIZE:

* Δέχεται την εικόνα ή αλλιώς το frame και το καινούργιο width που θα είναι ίσο με 500. Κάνει unpack το ύψος και το πλάτος. Υπολογίζει το ratio κάνοντας την διαίρεση και μέσο αυτού υπολογίζει το νέο πλάτος. Επιστρέφει την αλλαγή μεγέθους για την εικόνα με το νέο ύψος και πλάτος.
* \_ είναι dummy μεταβλητή για όταν κάνουμε unpack, να πάρει τιμές που δεν χρειαζόμαστε.

## GENERATE DATA:

* Η συνάρτηση δημιουργήθηκε στο αρχείο DataBase\_Generator.py.
* Κάνουμε import τα pandas.
* Δημιουργούμε τις στήλες (header) με ονόματα id, username, password και clearance.
* Δημιουργούμε τα δεδομένα για τους χρήστες.
* Μέσω των pandas δημιουργούμε το dataFrame και ύστερα το επιστρέφουμε.

## LOGGING SYSTEM:

* Δημιουργήθηκε αρχείο *Logging\_System.py* στο οποίο έχει γίνει import η βιβλιοθήκη logging.
* Στο αρχείο αυτό δημιουργήθηκε συνάρτηση logData η οποία θα καταγράφει το id του κάθε χρήστη όταν συνδέεται καθώς και πότε έκανα λήψη της θόλωσης όπου αυτό γίνεται και τέλος θα καταγράφει πότε πραγματοποίησε έξοδο.
* LOGGING: Από την βιβλιοθήκη “logging” χρησιμοποιήθηκαν οι συναρτήσεις “logging.shutdown”, “logging.basicConfig” και “logging.info”.
  + LOGGING.SHUTDOWN: Με το πέρας του προγράμματος καλείτε αυτή η συνάρτηση για να ενημερώσει το σύστημα να διακόψει την λειτουργία κάνοντας “flush” όλους τους handlers που έχουν απομείνει.
  + LOGGING.BASICCONFIG: Δημιουργεί το αρχείο στο οποίο θα γράψουμε το log σε κάθε περίπτωση.
  + LOGGING.INFO: Προσθέτει την πληροφορία που θέλουμε στο log αρχείο που βρισκόμαστε.
* Κάνουμε μετατροπή των παραμέτρων που δέχεται η συνάρτηση σε μορφή string. Αν το clearance είναι 0 τότε κάνουμε καταγραφή μόνο πότε έκανε είσοδο και έξοδο ο χρήστης αλλιώς πότε έκανε είσοδο, έξοδο και πότε έκανε λήψη τις θόλωσης.
* Η καταγραφή των timestamps επειδή είναι σε μορφή λίστας γίνεται με την βοήθεια join, και map

## LOGIN:

* Δημιουργήθηκε αρχείο *Login\_System.py* στο οποίο έχουν γίνει import οι βιβλιοθήκες sys, tkinter για να κάνουμε διακοπή στην λειτουργία του προγράμματος και εμφάνιση των GUI αντίστοιχα και δημιουργήθηκε συνάρτηση *login(),* η οποία θα πραγματοποιεί του κατάλληλους ελέγχους κατά την διάρκεια της σύνδεσης και θα επιστρέφει το ID του συνδεδεμένου χρήστη.
  + Η συνάρτηση δέχεται το data frame των users. Δημιουργούμε μεταβλητές logged\_in (ID του τρέχον χρήστη) και clearance (εξουσιοδότηση του συνδεδεμένου χρήστη).
  + Δημιουργούμε flags, correct\_username, correct\_password, pw\_check και error.
  + Ζητάμε από τον χρήστη να εισάγει το username του όσο το correct\_username = 0.
  + Δημιουργούμε tkinter παράθυρο και κάνουμε withdraw τα root παράθυρα και ζητάμε από τον χρήστη μέσω της συνάρτησης simpledialog.askstring να κάνει input το username του.
  + Κάνουμε iterate μέσα σε στο data frame και ελέγχουμε κατά πόσο υπάρχει το username που έδωσε ο χρήστης (αν υπάρχει αλλαγή του flag, correct\_username). Αν υπάρχει το username κάνουμε επανάληψη όσο το flag pw\_check είναι ίσο με 0, ζητάμε από τον χρήστη τον κωδικό με όμοιο τρόπο (GUI) και αν ο κωδικός αντιστοιχεί στο όνομα που έδωσε αλλάζουμε τα flags (correct\_password, pw\_check), αλλιώς εμφανίζουμε κατάλληλα μηνύματα μέσω των messagebox συναρτήσεων messagebox.showerror και προσθέτουμε 1 στο error κάθε φορά που κάνει λάθος τον κωδικό και όταν κάνει 3 λάθη τότε το πρόγραμμα τερματίζει. Αν πραγματοποιηθεί η είσοδος και τα flags είναι και τα δύο 1 τότε αποθηκεύουμε το ID του τρέχον χρήστη στην μεταβλητή logged\_in και την εξουσιοδότησή του στην μεταβλητή clearance, κάνουμε επαναφορά του correct\_password, εμφανίζουμε κατάλληλο μήνυμα σε περίπτωση λανθασμένου ονόματος ή ορθής εισόδου με τα messagebox.\* και επιστρέφουμε το logged\_in, clearance.

## LOGIN GUI:

* Δημιουργήθηκε αρχείο Login\_GUI.py στο οποίο έγινε import η βιβλιοθήκη tkinter για να κάνουμε εμφάνιση του gui για την είσοδο του χρήστη.
* Δημιουργήθηκε συνάρτηση loginGUI() μέσα στην οποίο δημιουργήσαμε συνάρτηση closeWindow που θα λειτουργήσει σαν command όταν πατηθεί το κουμπί και θα κλείσει το παράθυρο.
* Δημιουργούμε το παράθυρο με το geometry και τίτλο. Στην συνέχεια δημιουργούμε το label και entry widget για το username και το password αντίστοιχα (το password θα εμφανίζεται ως “\*” για censorship). Ύστερα δημιουργούμε το login button Και τα κάνουμε όλα pack για να εμφανιστούνε στην οθόνη με την main.loop.
* Κάνουμε get το username και το password και τα επιστρέφουμε.

## MAIN:

* Αποθηκεύουμε το dataFrame που δημιουργούμε μέσω τις συνάρτησης generateData στην μεταβλητή user\_data.
* Πραγματοποιούμε σύνδεση μέσω της συνάρτησης login, και αποθηκεύουμε το ΙD και την εξουσιοδότηση του συνδεδεμένου χρήστη.
* Αφού πραγματοποιηθεί σύνδεση αποθηκεύουμε την ώρα μέσω της συνάρτηση datetime.datetime.now() και δηλώνουμε το timestamp σε κενή λίστα (σε περίπτωση που ο χρήστης αποφασίσει να μην κάνει άρση της θόλωσης)
* Χρησιμοποιούμε Haar Cascade (haarcascade\_frontalface\_alt.xml) για την αναγνώριση προσώπων από το OpenCV.
* Με την μεταβλητή capture θα κάνουμε εγγραφή με την χρήση τις κάμερας. Χρήση cv2.CAP\_DSHOW (direct show) στις παραμέτρους (specifying the video source).
* Ελέγχουμε αν έχει ανοίξει σωστά.
* Κάνουμε while loop για να εξετάσουμε κάθε ένα frame.
* Κάνουμε ανάγνωση το frame και καλούμε την συνάρτηση resize για να αλλάξουμε το μέγεθός του.
* Γίνεται χρήση της συνάρτησης detectMultiScale() μέσω του cascade για να εντοπιστούν τα πρόσωπα σε κάθε στιγμιότυπο. Η συνάρτηση έχει παραμέτρους:
  + scaleFactor: parameter specifying how much the image size is reduced at each image scale. Makes the faces detectable by the algorithm at any scale.
  + Το κάθε frame.
  + minNeighbors: parameter specifying how many neighbors each candidate rectangle should have to retain it.
* Κάνουμε loop για κάθε πρόσωπο στην λίστα detections και κάνουμε unpack τις τιμές των μεταβλητών σε κάθε επανάληψη.
* Γίνεται χρήση τις Gaussian συνάρτησης του OpenCV η οποία δέχεται ως παραμέτρους το source (input image), το ksize (gaussian kernel size, height width.) και το border type. Το αποτέλεσμα αποθηκεύεται στο frame.
  + Για το source, κάνουμε slicing ξεκινώντας από το y μέχρι το y + height και από το x μέχρι το x + width).

Μέσω της cv2.rectangle δημιουργούμε ένα περίγραμμα ορίων. Δέχεται ως παραμέτρους την εικόνα. Τις αρχικές συντεταγμένες, τις τελικές συντεταγμένες, το bgr χρώμα και το πάχος του περιγράμματος. Επιστρέφει την εικόνα.

* Εμφανίζουμε την εικόνα μέσω της συνάρτησης cv2.imshow
* Cv2.waitKey(1): κάνουμε display τα frames με καθυστέρηση 1 millisecond και αν ισούται με 27 βγαίνουμε από την επανάληψη.
* Ελέγχουμε αν το πλήκτρο “-” είναι πατημένο και αν η εξουσιοδότηση ισούται με “εξουσιοδότηση 1”, ξεκινάμε την χρονομέτρηση μέσω της συνάρτησης time.time() και όμοια με πριν με την χρήση τις daytime.daytime.now αποθηκεύουμε την στιγμή που ο χρήστης έκανε άρση τις θόλωσης και την αποθηκεύουμε στην λίστα timestamp. Όμοια επανάληψη χωρίς να κάνουμε θόλωση των προσώπων και αν ο συνολικός χρόνος – τις αρχής είναι μεγαλύτερο ή ισούται του 30 σημαίνει πως έχουν περάσει 30 δευτερόλεπτα άρα κάνουμε break(επιστρέφουμε σε θόλωση). Αν πατηθεί το πλήκτρο “esc” τότε γίνεται ξανά θόλωση.
* Ελέγχουμε αν το πλήκτρο “-” είναι πατημένο και αν η εξουσιοδότηση ισούται με “εξουσιοδότηση 2” όμοια με πριν με την χρήση τις daytime.daytime.now αποθηκεύουμε την στιγμή που ο χρήστης έκανε άρση τις θόλωσης και την αποθηκεύουμε στην λίστα timestamp και κάνουμε όμοια επανάληψη χωρίς να κάνουμε θόλωση των προσώπων. Αν πατηθεί το πλήκτρο “esc” τότε γίνεται ξανά θόλωση.
* Ελέγχουμε αν το πλήκτρο “`” είναι πατημένο και αν πατηθεί τότε βγαίνουμε από την επανάληψη.
* Παίρνουμε όμοια την ώρα που έκανε logout ο χρήστης και καλούμε την συνάρτηση logData με τις μεταβλητές user\_ID, loginTime, logoutTime, timestamp και clearance.
* Τέλος κάνουμε release το capture και κλείνουμε όλα τα παράθυρα.