בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 6 - עיבוד תמונה

להגשה בתאריך 6/12/2017 בשעה 22:00

הקדמה

בתרגיל זה נכתוב תוכנית המזהה זווית של תמונה, ומיישרת את התמונה בהתאם. לצורך פתרון התרגיל עליכם להוריד את הקובץ ex6_helper.py – מודול זה כבר מומש בשבילכם, והוא מכיל מספר פונקציות הדרושות לתרגיל. אל תעשו שום שינוי בקובץ זה!

בנוסף, לרשותכם הקבצים tests.zip ו- corrected.zip. הקובץ tests.zip מכיל מספר קבצי תמונה עליהם תוכלו לבחון את התוכנית הסופית שלכם. שמות קבצי תמונה אלה מסתיימים במספר – המייצג את זווית התמונה. הקובץ corrected.zip מכיל תוצאות של הרצת פתרון בית הספר על הקבצים הנמצאים בקובץ tests.zip עבור ערכים כמתואר בסוף הנחיות התרגיל.

שימו לב, התרגיל מורכב בצורה מובנית ממספר משימות, שבסופו של דבר יתחברו ביחד וירכיבו את המוצר הסופי. לכן עקבו אחר ההוראות והשלבים של התרגיל, והקפידו לכתוב את הקוד שלכם במדויק על פי הנחיות התרגיל. כמו כן, מומלץ בחום לקרוא את כלל התרגיל (ובפרט את סעיף ה"טיפים וההנחיות" בסופו) לפני תחילת הפתרון.

עליכם ליצור קובץ בשם ex6.py בו תממשו את התרגיל. אתם יכולים לכתוב בקובץ ex6.py פונקציות עזר נוספות מלבד אלה הדרושות בתרגיל, ולהשתמש בהן בקוד שלכם. אבל הפונקציות הדרושות בתרגיל חייבות להיכתב בדיוק על פי הדרישות המפורטות להלן.

הקפידו לכתוב תיעוד לקוד שלכם ובפרט לכל פונקציה שאתם כותבים.

הספריה PIL

הקובץ – ex6_helper.py הנתון לכם בתרגיל זה עושה שימוש בספריה PIL של פייתון, ולכן צריך אותה על מנת להריץ את התרגיל.

במעבדת המחשבים של האוניברסיטה (האקוריום) כבר מותקנת ספריה זו, ואין צורך להתקין שום דבר.

כמו כן, הספרייה כלולה ב- WinPython (זמין להורדה ב- http://winpython.github.io). שימו לב – יש לוודא כי האינטרפרטר איתו אתם עובדים הוא זה של WinPython על מנת להנות מעובדה זו!

עבודה עם תמונות בתרגיל

הקובץ ex6_helper.py ממסך את העבודה עם אובייקטי תמונה של פיית'ון. במקום לעבוד ישירות עם אובייקט תמונה, "תמונה" בתרגיל מיוצגת ע"י רשימה של רשימות. כאשר תמונה בגובה 100 פיקסלים וברוחב 200 פיקסלים היא רשימה של 100 רשימות – של 200 פיקסלים כל אחת. קריאה ל- image[row][column] תתן את הפיקסל שבשורה row ובעמודה column, כאשר הפיקסל השמאלי העליון הוא הפיקסל בשורה 0 ועמודה 0.

בתרגיל זה תעבדו רק עם תמונות בגווני אפור, כך שגישה לפיקסל נותנת int בין 0 ל-255, המייצג את בהירות הפיקסל כפי שאנו רואים אותו בתמונה.

הקובץ ex6_helper.py

לצורך התרגיל מסופק לכם הקובץ ex6_helper.py, הממש פונקציות מסוימות להן אתם נדרשים בתרגיל:

```
load_image(image_filename):
```

פונקצייה זו מקבלת את שם קובץ של תמונה (String), ומחזירה את התמונת (כרשימה של רשימות).

```
save(image, filename):
```

פונקצייה זו מקבלת תמונה (רשימה של רשימות) ושומרת אותה בקובץ ששמו (String) הוא filename.

מעבר לכך, לנוחותכם גם הפונקציה:

```
show(image):
```

המציגה תמונה (רשימה של רשימות) למסך.

בקובץ ex6_helper.py גם פונקציות נוספות לשימושכם, אשר יפורטו בהמשך.

חלק ראשון: פילטרים

1. הפונקציה otsu

עליכם לממש את הפונקציה otsu, המחשבת ערך סף אופטימלי לתמונה. חתימת הפונקציה צריכה להיות:

```
def otsu(image):
```

הפונקציה מקבלת את הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות), ומחזירה ערך סף (int) אופטימלי, בפי שראיתם בתרגול 6.

נזכיר כי ערך סף אופטימלי הינו ערך הסף (בין 0 ל- 255) עבורו מתקבלת השונות הפנימית המקסימלית.

נזכיר עוד כי עבור ערך סף threshold, אם נגדיר:

$$\#black = number\ of\ black\ pixels\ (< threshold)$$

$$\#white = number\ of\ white\ pixels\ (\geq threshold)$$

$$mean\_black = average\ value\ of\ black\ pixels$$

$$mean\_white = average\ value\ of\ white\ pixels$$

$$intra\_variance = \#black * \#white * (mean\_black - mean\_white) ** 2$$

2. הפונקציה threshold_filter

עליכם לממש את הפונקציה threshold_filter. חתימת הפונקציה צריכה להיות:

```
def threshold_filter(image):
```

הפונקציה מקבלת את הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות), ומחזירה תמונה בגודל זהה (מבלי לשנות את תמונת המקור) שבה כל פיקסל הוא שחור או לבן (בהתאם להאם ערך הפיקסל בתמונה המקורית נמוך או לא מערך הסף האופטימלי לתמונה).

3. הפונקציה apply_filter

עליכם לממש את הפונקציה apply_filter. חתימת הפונקציה צריכה להיות:

```
def apply_filter(image, filter):
```

הפונקציה מקבלת שני פרמטרים:

א. הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות).

ב. הפרמטר filter המייצג מטריצה בגודל 3*3 (כרשימה של רשימות), שכל איבר ברשימות הפנימיות בו הוא מספר (לא בהכרח שלם!).

הפונקציה מחזירה תמונה new_image בגודל זהה לתמונה המקורית (מבלי לשנות את תמונת המקור), כאשר הפיקסל new_image[row][column] מחושב בצורה הבאה:

מזהים את הפיקסל image[row][column] עם הכניסה המרכזית במטריצה filter, וסוכמים את ערכי שכניו (כולל הפיקסל עצמו) כפול הכניסה המתאימה להם ב- filter. לדוגמא, עבור מטריצת filter הבאה:

-1	-1	-1
-1	8	-1
-1	-1	-1

(כאשר סדר התאים הוא בהתאם לזה בו אנו משתמשים עבור תמונות) נקבל:

new_image[row][column] = -image[row-1][column-1] - image[row-1][column] - image[row-1][column+1] –

- image[row][column-1] + 8*image[row][column] - image[row][column+1] -

- image[row+1][column-1] - image[row+1][column] - image[row+1][column+1]

במידה וסכום זה אינו שלם, יש להתייחס רק לחלק השלם בו. אם הסכום קטן מ- 0: יש להתייחס אל הסכום בערך מוחלט. אם הסכום גדול מ- 255, יש להתייחס אליו כאל 255.

שימו לב: בחישוב ערך לפיקסל x הנמצא על גבולות התמונה, יש להתייחס לערכי פיקסלים הנמצאים מחוץ לגבולות תמונות המקור כאילו היו בעלי ערך זהה לזה של הפיקסל x.

4. הפונקציה detect_edges

עליכם לממש את הפונקציה detect_edges, הממשת אלגוריתם לזיהוי קצוות. חתימת הפונקציה צריכה להיות:

```
def detect_edges(image):
```

הפונקציה מקבלת את הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות), ומחזירה תמונה בגודל זהה (מבלי לשנות את תמונת המקור) בה הערך של כל פיקסל שווה לערך של הפיקסל המתאים לו בתמונת המקור, פחות הממוצע של 8 שכניו בתמונת המקור (לחישוב ערך לפיקסל x הנמצא על גבולות התמונה, נשלים את רשימת 8 שכניו ע"י פיקסלים בעלי ערך זהה ל- x). אם ערך זה שלילי, נתייחס אליו בערך מוחלט.

הנחיה: זכרו כי ניתן להשתמש בכל פונקציה שמימשתם כבר בתרגיל!

5. הפונקציה downsample_by_3

עליכם לממש את הפונקציה downsample_by_3, המחזירה תמונה שרוחבה וגובהה שניהם קטנים פי 3. חתימת הפונקציה צריכה להיות:

```
def downsample_by_3(image):
```

הפונקציה מקבלת את הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות), ומחזירה תמונה מוקטנת פי 3. הקטנת התמונה נעשית ע"י חלוקת התמונה המקורית לריבועים בגודל 3*3 פיקסלים כל אחד, והחלפת כל אחד מהם בפיקסל יחיד שערכו הוא הערך השלם של ממוצע הפיקסלים בריבוע.

שימו לב: פונקציה זו תיבדק רק עבור מקרים בהם אורך ורוחב התמונה מתחלקים ב- 3 (אך לנוחותכם, מומלץ לממש את הפונקציה כך שתעבוד גם במקרים בהם זה לא המצב).

הנחיה: זכרו כי ניתן להשתמש בכל פונקציה שמימשתם כבר בתרגיל!

חלק שני: מציאת זווית וסיבוב

לפני שניגש לאלגוריתמים למציאת זווית ולסיבוב תמונה, נממש פונקציה המחזירה תמונה מוקטנת – כך שזמן הריצה של יתר האלגוריתמים לא יהיה ארוך מדי.

6. הפונקציה downsample

עליכם לממש את הפונקציה downsample, המחזירה תמונה מוקטנת. חתימת הפונקציה צריכה להיות:

```
def downsample(image, max_diagonal_size):
```

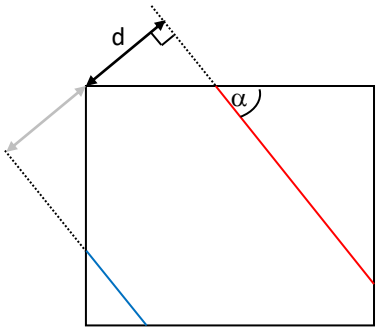
הפונקציה מקבלת שני פרמטרים:

א. הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות).

ב. הפרמטר max_diagonal מספר חיובי (לא בהכרח שלם) שהוא האורך המקסימלי של אלכסון התמונה המוחזרת (למען הסר ספק - אורך האלכסון מחושב ע"פ משפט פיתגורס).

הפונקציה מחזירה תמונה המתקבלת על ידי הקטנת התמונה המקורית פי 3 (בלי לשנות את התמונה המקורית) שוב ושוב עד שאורך האלכסון בתמונה המתקבלת הוא max_diagonal או פחות. הקטנת התמונה תעשה ע"י קריאות חוזרות לפונקציה **downsample_by_3**.

כעת נעבור למצוא את הזווית האופיינית בתמונה. בתמונות רבות ניתן להבחין במספר קווים, ישרים בקירוב, בזוויות שונות. הזווית הדומיננטית בתמונה היא זו המופיעה בצורה המשמעותית ביותר בתמונה, ויכולה להעיד על כך שניתן לסובב את התמונה בזווית זו ולקבל תמונה "מיושרת". על מנת למצוא את הזווית הדומיננטית בתמונה, ראשית נשים לב כי כל ישר החותך את התמונה ניתן לייצוג ע"י זוויתו ומרחקו מראשית הצירים. לדוגמה, בתמונה מופיע קו (מסומן באדום) בזווית α הנמצא במרחק d מראשית הצירים.



שימו לב: עבורה זוויות גדולות מ- 0 וקטנות מ- 90 מעלות, ישנו קו נוסף באותה זווית ובאותו מרחק. במקרה כזה, נתייחס אל הקו הכחול כאל "העליון" מביניהם.

על מנת למצוא את הזווית הדומיננטית בתמונה, ניתן דירוג לכל זווית, ולבסוף נבחר בזווית בעלת הדירוג המקסימלי. זאת בעזרת הפונקציה הבאה:

7. הפונקציה get_angle

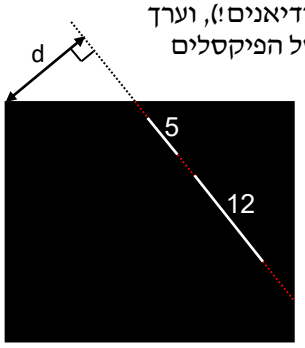
עליכם לממש את הפונקציה get_angle, המחזירה זווית דומיננטית של תמונה. חתימת הפונקציה צריכה להיות:

```
def get_angle(image):
```

הפונקציה מקבלת את הפרמטר image המייצג תמונת שחור-לבן (כרשימה של רשימות), ומחזירה את ערך הזווית הדומיננטית (במעלות) בתמונה. על מנת לתת דירוג לכל אחת מהזוויות, עליכם להשתמש בפונקציה

```
pixels_on_line(image, angle, distance, top=True)
```

הממומשת עבורם בקובץ ex6_helper.py. פונקציה זו מקבלת כפרמטרים תמונה, זווית (ברדיאנים!), וערך בוליאני המתאר האם מדובר בקו ה"עליון" (במידה וזה רלוונטי), ומחזירה רשימה סדורה של הפיקסלים המופיעים בתמונה ונמצאים על קו זה.



לכל קו כזה ניתן דירוג בצורה הבאה:
נמצא את אורכי כל הקווים המופיעים בתמונה (כפיקסלים לבנים) הנמצאים עליו, ונסכום את ריבועי האורכים הנ"ל. כך, בדוגמא שמשמאל:
הדירוג שיינתן לקו האדום יהיה 25+144=169.

על מנת למצוא את אורכי הקווים המופיעים על קו ישר, ראשית נגדיר כי שני פיקסלים שמרחקם האחד מהשני הוא לכל היותר **2** ייחשבו כפיקסלים רצופים לצורך חישוב אורך הקווים. כך נתגבר על אי רציפויות נקודתיות. (נזכור:
המרחק בין הפיקסל image[row1][column1] והפיקסל image[row2][column2]
הוא: $\sqrt[2]{(row1-row2)^2+(column1-column2)^2}$)

כעת, על מנת למצוא את הערך של ישר מסויים, נעבור על הפיקסלים שעליו עד למציאת הפיקסל הלבן הראשון, ונמשיך עד מציאת הפיקסל הלבן האחרון כך שבינו ובין הפיקסל הראשון יש סדרה של פיקסלים לבנים שבה כל שני פיקסל עוקבים הם רצופים (כלומר, מרחק הפיקסל הראשון מהשני הוא לכל היותר 2, מרחק השני מהשלישי לכל היותר 2, וכן הלאה עד לאחרון). כעת ניתן לחשב את המרחק בין הפיקסל הראשון לבין האחרון – וזה אורך

הקו. עתה נמשיך לעבור על הפיקסלים שעל הישר עד למציאת הפיקסל הלבן הבא, שמהווה את הפיקסל הראשון בקו הבא, וכן הלאה.

בשביל לתת דירוג לזווית מסויימת, נסכום את דירוגי כל הישר המתאימים לזווית זו – כאשר לצורך מעבר על כל הישרים המתאימים לזווית נעבור על כל כל הישרים בזווית זו שמרחקם מהראשית הוא לכל היותר אורך האלכסון (במימוש תרגיל זה יש לעבור על אורכים אלה בקפיצות של 1).

על מנת למצוא את הזווית הדומיננטית, יש לעבור על כלל הזוויות בין 0 ל- 179 מעלות (בקפיצות של מעלה אחת), ולבחור בזווית בעלת הדירוג הגבוה ביותר.

שימו לב כי עבור זוויות מסוימות יש יותר מישר אחד באותו מרחק, כמפורט מעלה.

8. הפונקציה rotate

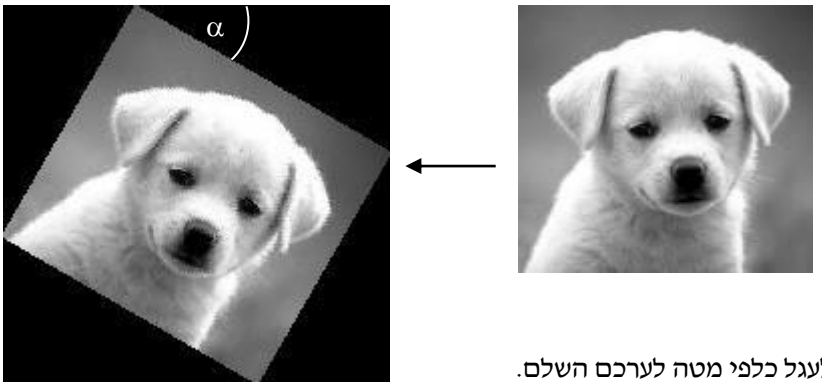
עליכם לממש את הפונקציה rotate, המחזירה תמונה מסובבת. חתימת הפונקציה צריכה להיות:

```
def rotate(image, angle):
```

הפונקציה מקבלת שני פרמטרים:

א. הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות)
ב. הפרמטר angle שהוא הזווית (מספר לא בהכרח שלם) במעלות בה אנו רוצים לסובב את התמונה.

ומחזירה תמונה חדשה (מבלי לשנות את התמונה המקורית), בה נמצאת התמונה המקורית כשהיא מסובבת בזווית angle סביב מרכזה. נזכיר כי סיבוב בזווית α סביב הראשית (0,0) מעביר את הנקודה (y,x) לנקודה ($\sin(\alpha)*x+\cos(\alpha)*y$, $\cos(\alpha)*x-\sin(\alpha)*y$). על מנת להשלים את התמונה המתקבלת מהסיבוב, פיקסלים בתמונת היעד (לאחר הסיבוב) שאין פיקסל בתמונה המקורית שמתאים להם – יקבלו את הערך 0 כפי שניתן לראות בדוגמא שמשמאל, בה מתקבלת תמונת כלב כקלט ומסובבת בזוית של α=30 מעלות.



שימו לב:

א. לבניית תמונת היעד,יש לעבור על כל אחד מהפיקסלים בה, ועבור כל אחד מהם לחשב מה הפיקסל בתמונת המקור שמתאים לו
ב. במידה ומתקבלים ערכים לא שלמים, יש לעגל כלפי מטה לערכם השלם.
ג. לצורך מימוש פונקצייה זו, תוכלו להשתמש בפונקציה

```
get_diagonal_angle(image)
```

הממומשת עבורכם בקובץ ex6_helper.py, שמקבלת כפרמטר תמונה, ומחזירה את הזוית (במעלות) של האלכסון בה.

כעת נוכל לבצע את תהליך התיקון המלא:

9. הפונקציה make_correction

עליכם לממש את הפונקציה make_correction, המזהה זווית דומיננטית בתמונה ומחזירה תמונה מתוקנת באותה זווית. חתימת הפונקציה צריכה להיות:

```
def make_correction(image, max_diagonal):
```

הפונקציה מקבלת 2 פרמטרים:

א. הפרמטר image המייצג תמונה בגווני אפור (כרשימה של רשימות).

ב. הפרמטר max_diagonal מספר חיובי (לא בהכרח שלם) שהוא האורך המקסימלי של האלכסון בתמונה עליה יפעל האלגוריתם לזיהוי הזווית הדומיננטית (כפי שיפורט מייד).

ראשית, הפונקצייה מוצאת את הזווית הדומיננטית בתמונה בצורה הבאה:

א. ראשית, על מנת שהאלגוריתם יעבוד בזמן ריצה סביר – נעבור לעבוד עם יצירת תמונה מוקטנת, שאורך אלכסונה לכל היותר max_diagonal
ב. שנית, נעבור לתמונת שחור-לבן
ג. על תמונת השחור-לבן נפעיל את מסנן זיהוי קצוות
ד. גם את תמונת זיהוי הקצוות נעביר לתמונת שחור-שחר
ה. על תמונת הקצוות בשחור-לבן נפעיל את הפונקציה get_angle לקבלת הזווית הדומיננטית בתמונה.

כעת, לאחר שמצאנו את הזווית הדומיננטית α, נחזיר תמונה בה תמונת המקור מתוקנת על ידי סיבוב בזווית ההפוכה α-.

10. הרצת התרגיל

בנוסף לפונקציות שעליכם למלא בקובץ ex6.py , עליכם לכתוב בו מקטע קוד שמריץ את התוכנית ושומר את התוצאה בקובץ (לצורך כך, תצטרכו להשתמש בפקודות מהקובץ ex6_helper.py). על מקטע קוד זה לרוץ רק כאשר מריצים את התוכנית משורת הפקודות. הרצת התוכנית מתבצעת על ידי הפקודה:

python ex6.py <image_source> <output_name> <max_diagonal >

כאשר:

image_source - string שהוא השם של תמונת המקור.

output_name – string שהוא השם של התמונת המתוקנת הנשמרת.

max_diagonal - מספר חיובי שהוא האורך המקסימלי של האלכסון בתמונה עליה ירוץ האלגוריתם למציאת הזווית הדומיננטית בתמונה (המפורט בסעיף הקודם).

דוגמא להרצת ex6.py עם פרמטרים שונים:

```
python3 ex6.py test.jpg corrected.jpg 330
```

אם נניח כי הקובץ ex6.py נמצא בסיפריית הבית "./" , אזי על ההרצה הנ"ל ליצור תמונה מתוקנת לתמונה "./test.jpg" ולשמור אותה בקובץ "./corrected.jpg" (ללא הצגה למסך). להזכירכם, הפרמטרים בהרצת קובץ פייתון שמורים ב- sys.argv.

הפתרונות המופיעים בקובץ corrected.zip הם פתרונות המתקבלים כתוצאה מהרצת התוכנית עבור ערך max_diagonal של 330.

במידה והקובץ ex6.py מורץ עם **מספר פרמטרים שונה מהדרוש**, עליכם להדפיס למסך הודעה אינפורמטיבית המפרטת את הדרך הנכונה להקרוא לקובץ. לדוגמא, תוכלו להדפיס את ההודעה:

"Wrong number of parameters. The correct usage is:

ex6.py <image_source> <output > <max_diagonal>"

שימו לב: כאשר מייבאים את הקובץ ex6.py (על ידי import) הרצת התוכנית והשמירה לקובץ לא אמורים להתבצע!

אתם יכולים להניח תקינות של הקלט (כלומר שכל הנתונים בשורת הפקודה תקינים).

טיפים והנחיות

- להזכירכם, **בכל מקום** בתרגיל בו מדובר על "**תמונה**" הכוונה היא ל**רשימה של רשימות**.
- על מנת שהחישובים השונים יתבצעו במהירות – מומלץ לעבוד עם תמונות קטנות במהלך פתרון התרגיל.
- כמו כן – בתהליך הפתרון, מומלץ לקבל חיווי (הדפסת הודעה, או הצגת תמונה, לדוגמא) בשלבים שונים של ריצת התוכנית. כך תוכלו לדעת שהאלגוריתם "מתקדם" ומה מתקבל כתוצאה מהפעלת הפונקציות שכתבתם. **אך שימו לב לא להשאיר חיוויים כאלה כ"זבל" בתרגיל הסופי**!
- ניתן להניח תקינות הקלטים לכל אחד מהסעיפים (בהתאם להגדרת הפרטנית של כל סעיף). בפרט, ניתן להניח כי כל התמונות ניתנות בפורמט תקין (רשימה של רשימות, שבכל אחת מהן אותו מספר פיקסלים), וכי כל הרשימות הן אכן רשימות לא ריקות.
- **אין לשנות את האובייקטים המתקבלים כפרמטרים באף אחת מהפונקציות.**
- הקפידו לכתוב את הקוד שלכם בצורה מדויקת וברורה, ולשים לב לאינדקסים וגבולות של לולאות וקונטיינרים.
- לצורך פיתרון התרגיל תוכלו גם להשתמש במודולים sys ו-copy כפי שראיתם בתרגולים.
- ניתן להשתמש בכל הפונקציות המובנות של פייתון. אך **אין להשתמש במודול numpy או PIL.**
- הרצת הקוד על מחשבים שונים עשוייה לתת תוצאות מעט שונות עקב מימושים שונים למודול PIL. במידה ואתם נתקלים בשוני בין תוצאת ריצת התוכנית שלכם לבין תוצאות פתרונות בית הספר - מומלץ לוודא את ריצת התוכנית שלכם על מחשבי בית הספר למול תוצאת פתרון בית הספר.

נהלי הגשה

הלינק להגשה של התרגיל הוא תחת השם : ex6

בתרגיל זה עליכם להגיש את הקבצים הבאים :

1. ex6.py – עם המימושים שלכם לפונקציות.
2. README (על פי פורמט ה-README לדוגמא שיש באתר הקורס, ועל פי ההנחיות לכתיבת README המפורטות בקובץ נהלי הקורס).

יש להגיש קובץ zip הנקרא ex6.zip המכיל בדיוק את שני הקבצים הנ"ל.

בהצלחה! ☺