תרגיל בית 10

Numpy

הנחיות כלליות:

- קראו **בעיון** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל הפתרונות לשאלות יחד בקובץ ex10_012345678.py המצורף לתרגיל, לאחר החלפת הספרות012345678 שבשם הקובץ במספר תעודת הזהות שלכם, כל 9 הספרות כולל ספרת ביקורת.
- אופן ביצוע התרגיל: שימו לב, בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- **אין לשנות את שמות המחלקות, הפונקציות, המתודות והמשתנים שכבר מופיעים בקובץ השלד של התרגיל.**
- **אין למחוק את ההערות שמופיעות בקובץ השלד.**
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון).
- **ניתן להניח שהקלט תקין בהתאם להערות המפורטות בהוראות כל שאלה, אלא אם מצויין אחרת.**
- מועד אחרון להגשה: כמפורסם באתר.

שאלה 1

יוצרי הסדרה הסימפסונ'ס הגיעו למסקנה שהפופולריות של הסדרה דועכת וביקשו מכם לנתח את מצב הסדרה. לרשותכם קובץ המכיל מטריצה של צפיות (במליוני צפיות), כאשר כל שורה היא עונה וכל טור מייצג פרק בעונה לפי הסדר.

שימו לב: השתדלו לעשות שימוש בפקודות Numpy ככל האפשר, בפרט - בסעיפים ב'-ה' אין להשתמש בלולאות או ברקורסיה.

א. ממשו את הפונקציה **read_rating(file_name)**

- הפונקציה מקבלת **את שמו של קובץ נתוני צפיות** וקוראת את תוכנו
- הפונקציה תקרא את נתוני הקובץ למערך דו מימדי של Numpy ותחזיר אותו
- רמז: היעזרו בפקודה np.loadtxt הטוענת את תוכנו של קובץ טקסט בפורמט טבלאי פשוט ישירות למערך של Numpy.
- במקרה של שגיאת IO, יש להדפיס למסך "IO Error encountered" ולהחזיר None.
  אין צורך לבדוק את כמות השורות שמתקבלת בקובץ (ניתן להניח שיש יותר משורה אחת)

הערה: שאר הסעיפים משתמשים במערך המיוצר בסעיף א'

ב. ממשו את הפונקציה **get_mean_rating_for_season(rating_matrix,season)**

- הפונקציה מקבלת מטריצה rating_matrix ומספר עונה season
- הפונקציה תחזיר **מספר בודד** המייצג את ממוצע נתוני הצפיה בעונה המצויינת

הערה: שימו לב כי העונה הראשונה נמצאת בשורה הראשונה (אינדקס 0) ובאופן כללי האינדקס של כל עונה הוא מספר העונה פחות 1. בצורה דומה גם לפרקים.

ג. ממשו את הפונקציה **min_rating_per_episode_num(rating_matrix, episode)**

- הפונקציה מקבלת מטריצה rating_matrix ומספר פרק episode
- הפונקציה תחזיר את **העונה** בה הפרק המצוין קיבל הכי פחות צפיות ואת מספר הצפיות שקיבל.

הפלט יוחזר כ tuple בו האיבר הראשון הוא מספר העונה והשני הוא מספר הצפיות

הדרכה: ניתן להשתמש בפונקציות argmax, argmin אשר מחזירות את האינדקס של הערך המקסימלי/מינימלי (בהתאמה)

ד. ממשו את הפונקציה **most_popular_episode_num(rating_matrix)** המקבלת את המטריצה rating_matrix, מחשבת את ממוצע הצפיות בכל פרק (בכל העונות) ומחזירה את **מספר הפרק** בו ממוצע הצפיות היה מקסימלי בין כל העונות.

הדרכה: גם פה ניתן להשתמש בפונקציות argmax, argmin

ה. ממשו את הפונקציה **plot_season_mean(rating_matrix)** המקבלת את המטריצה rating_matrix, מחשבת את ממוצעי הצפיות בכל העונות ומציגה אותם בגרף עמודות (כל עמודה תייצג ממוצע עונתי). הוסיפו כותרות מתאימות לתרשים ולכל אחד מהצירים.

הדרכה: על מנת לקבל גרף דומה לגרף בדוגמא, השתמשו בשורה:

```
plt.bar(np.arange(1,len(mean_episodes)+1),mean_episodes)
```

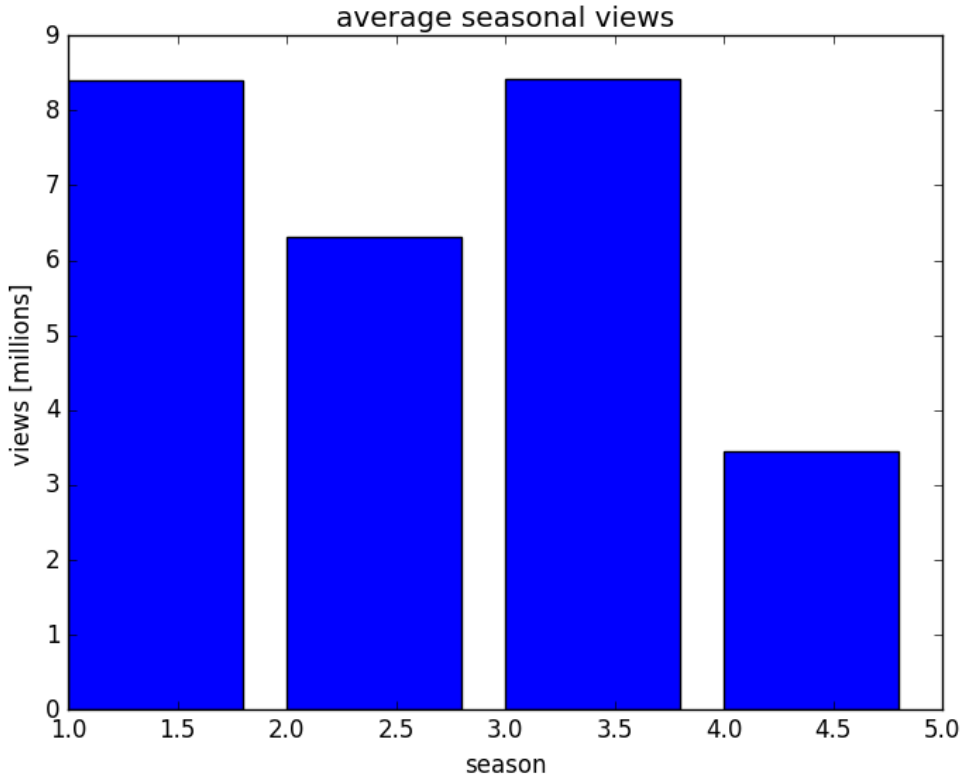
שימו לב: לא יורדו נקודות על חוסר התאמה בין הצירים לדוגמת ההרצה. (כן יורדו נקודות על אי הוספת כותרות מתאימות)

שימו לב שמספרי עונות אמורים להתחיל מ-1 ולא מ-0

הדרכה: צריך להשתמש בפונקציה plt.bar ניתן לראות הסבר על הפונקציה כאן:
https://matplotlib.org/api/pyplot_api.html
שימו לב שיש באתר פירוט על כל סוגי הגרפים

דוגמאות הרצה: (על הקובץ simpson_test.csv)

```
>>> views_matrix = read_rating(input_file)
>>> views_matrix
array([[  8. ,  10.74,   8.37,  12.54,   4.25,   6.47],
       [  5.11,   7.68,   5.84,   6.89,   5.32,   7.02],
       [  9.32,   7.5 ,   8.59,   9.04,   7.03,   9.02],
       [  4.28,   3.29,   2.78,   2.67,   3.93,   3.79]])
>>> get_mean_rating_for_season(views_matrix,2)
6.3099999999999996
>>> min_rating_per_episode_num(views_matrix,4)
(4, 2.6699999999999999)
>>> most_popular_episode_num(views_matrix)
4
```



שאלה 2

בשאלה זו ננתח קובץ מרשם אוכלוסין בפורמט CSV המכיל את נתוני האוכלוסיה של יישובים שונים בארץ בפילוג על פי טווח גילאים.
נניח שקובץ הקלט כולל טבלה דו-מימדית לפי המבנה הבא:

- בשורה הראשונה בטבלה מופיעים שמות טווחי הגילאים
- בעמודה הראשונה משמאל בטבלה מופיעים שמות היישובים
- בעמודה השנייה בטבלה מופיע סך כל התושבים ליישוב
- שאר העמודות מכילות את מספר התושבים לכל קבוצת גיל

דוגמא לטבלת הקלט:

City	total	0-9	10-19	20-35	36-59	60-300
Tel Aviv	15323	1324	2210	6342	3571	1876
Lod	12145	825	2341	3512	4211	1256
Eilat	8857	654	1712	2111	3524	856

לצורך ניהול הנתונים נכתוב מחלקה בשם PopulationData על פי הפירוט הבא:

א. כתבו את בנאי המחלקה PopulationData
חתימת הבנאי __init__(self, file_name)

- הבנאי יקבל את שם הקובץ של מרשם האוכלוסין ויקרא אותו
- כל אובייקט מהמחלקה יכיל את השדות הבאים:
  i. **cities** מערך חד מימדי (של מחרוזות) הכולל את שמות היישובים
  ii. **age_list** מערך דו מימדי המציין את טווחי הגילאים (מערך בן שתי שורות כאשר השורה הראשונה מכילה את ערכי המינימום של טווחי הגיל והשורה השנייה מכילה את ערכי המקסימום) של טווח הגילאים.
  דוגמא:לטווחי גילאים 0-2,3-11,12-19,20-39,40-200
  [[0,3,12,20,40],[2,11,19,39,200]]
  iii. **total_array** מערך חד מימדי (של integers) המכיל את סכום התושבים הכולל בכל יישוב
  iv. **population_mat** מערך דו מימדי (של integers) המכיל את מספר התושבים בכל טווח גיל בכל יישוב (נתוני הטבלה המספריים שאינם חלק מכותרות הטבלה) **.**
  v. **percentage_mat** מערך דו מעמדי (של floats) המכיל את החלק היחסי של (סכום כל שורה הוא 1) התושבים בכל טווח גיל בכל יישוב (מיוצר במתודה בסעיף ג')
- **על בנאי המחלקה להשתמש במתודות מסעיפים ב' ו-ג'** ולהדפיס את הערך החוזר מהמתודה בסעיף ב'
- בסעיף זה אין צורך לטפל בשגיאות IO

ב. בעמודה המכילה את סך כל התושבים ליישוב בקובץ הנתונים חלו מספר תקלות וחלק מהתוצאות יצאו שגויות. ממשו מתודה בשם **fix_total_array(self)** על פי הפירוט הבא:

- המתודה תשתמש במערך פיזור הגילאים (**population_mat**) ובוקטור סך כל הערכים (**total_array**) ותוודא את תקינותו (כל ערך בווקטור מכיל את סכום השורה המתאימה לו)
- אם הערך אינו תקין יש לתקנו.
- על הפונקציה ל**סכום את ההפרשים** של כל המקרים בהם הוקטור הכיל תוצאה גבוהה מהרצוי (error_high) ואת כל המקרים בהם הוקטור הכיל תוצאה נמוכה מהרצוי (error_low) ולהחזיר את שניהם **כ-tuple (**error_high**,** error_low).
- יש להשתמש במתודה בתוך הבנאי של המחלקה **ולהדפיס אותם בבנאי**
  ההדפסה תיראה כך:
  total error high was *error_high*
  total error low was *error_low*

ג. ממשו את המתודה **build_percentage_matrix(self)** שמייצרת מערך דו מימדי שיכיל את החלק (כאשר סך כל החלקים בשורה הוא 1) היחסי של כל קבוצת גיל באוכלוסיה

- המערך יהיה בגודל זהה למערך **population_mat**
- יש לשלב את המתודה בבנאי של המחלקה

דוגמא למטריצה המוחזרת (על פי הקלט שבדוגמא למעלה)

0.08640606	0.1442276	0.4138876	0.2330484	0.1224303
0.06792919	0.1927542	0.2891725	0.346727	0.103417
0.0738399	0.1932934	0.2383426	0.3978774	0.09664672

ד. ממשו מתודות שבודקות עבור כל קבוצת גיל את העיר בה היא דומיננטית ביותר

- ממשו את המתודה **max_city_age_group_percentage (self)** המחזירה רשימה המציגה עבור כל קבוצת גיל את היישוב בו קבוצת גיל זו מהווה את **החלק** הכי גבוה (ביחס לאחוז בערים האחרות)
- ממשו את המתודה **max_city_age_group_quantative(self)** המחזירה רשימה המציגה עבור כל קבוצת גיל את היישוב בו קבוצת גיל זו הכי גדולה (מספרית ביחס לערים האחרות)

ה. משפחה רוצה לעלות לארץ ולמצוא את היישוב שבו יש לחברי המשפחה הכי הרבה סיכוי למצוא חברים.

- ממשו את המתודה **find_friendly_neighborhood(self,family)** המקבלת מערך בשם family המכיל את גילאי חברי המשפחה. המתודה תחשב עבור כל יישוב את מכפלת האחוזים של קבוצות הגילאים האלה (אם שני חברי משפחה באותה קבוצת גיל הקבוצה תחושב פעמיים).
- הפונקציה תייצר מערך שיכיל את התוצאות ותחזיר את שם היישוב בו המכפלה היא מקסימלית (ניתן להניח שיש רק יישוב אחד כזה).

דוגמא (על פי הקלט למעלה):

find_friendly_neighborhood(self,arr[45,40,7,3,13])

חישוב לדוגמא (נראה על תל אביב):

Age_group(36-59)^2* Age_group(0-9)^2* Age_group(10-19)=
23.30484^2*8.640606^2*14.42276=5.84829E-05

ו. ממשו את המתודה **boxplot_age_group_percentage(self,family)** המציגה למסך boxplot של התפלגות האחוזים של כל קבוצת גיל כאשר כל עמודה תייצג את התפלגות האחוזים של קבוצת הגיל בכל הערים.

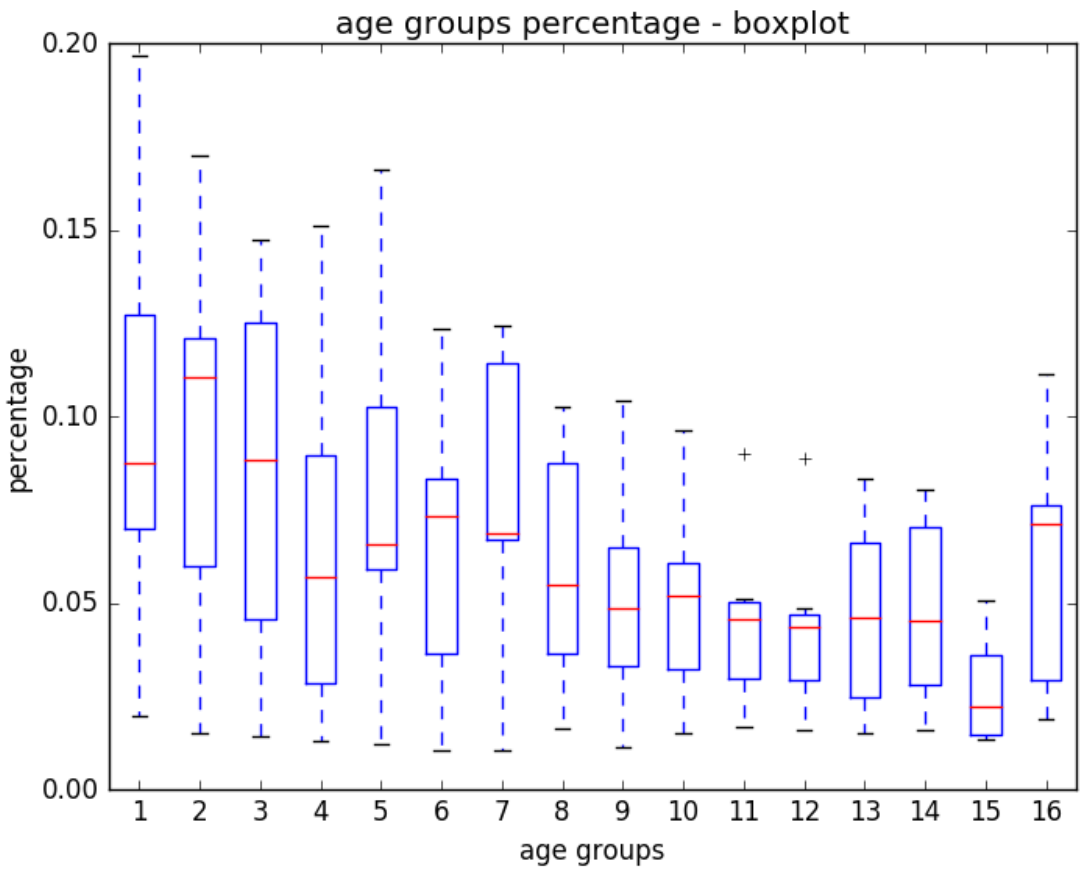
הדרכה: צריך להשתמש בפונקציה plt.boxplot ניתן לראות הסבר על הפונקציה כאן:
https://matplotlib.org/api/pyplot_api.html
שימו לב שיש באתר פירוט על כל סוגי הגרפים

דוגמאות הרצה: (על הקובץ population_data_test.csv)

```
>>> pop = pop_data(input_file)
```

```
total error high was 1610
total error low was 200392
>>>pop.population_mat
array([[10386,  9868,  8951,  8337,  7791,  6662,  5707,  4764,  4224,
        3656,  3376,  3164,  2246,  2921,  1391,  3208],
       [ 2482,  1937,  1797,  1651,  1544,  1315, 13584, 12977, 13216,
        12211, 11437, 11251, 10553, 10208,  6446, 14092],
       [21069, 18200, 15780, 16206, 17827,  1933,  1153,  1753,  1241,
        1644,  1813,  1695,  1605,  1735,  1465,  2016],
       [ 3778,  2292,  2004,  1223,  2892,  4002,  5436,  4512,  3095,
        2951,  2044,  2029,  2051,  2027,  1001,  3299],
       [ 1216,  1502,  1936,  1247,  1444,  1602,  1496,  1621,  1330,
        1209,  1086,  1069,  1198,  1363,   851,  1730],
       [ 2115,  2666,  1116,  1997,  1281,  1327,  1655,  1017,  1139,
        1257,  1232,  1048,  1886,  1890,   799,  1715],
       [ 8856,  8408,  9637,  1926,  7554,  8105,  8152,  2041,  1279,
        1488,  1364,  1447,  1525,  1495,   887,  1442]])
>>> pop.percentage_mat
array([[ 0.11985875,  0.11388081,  0.10329825,  0.09621244,  0.08991137,
        0.07688224,  0.06586115,  0.05497853,  0.04874671,  0.04219176,
        0.03896044,  0.03651387,  0.02591977,  0.03370955,  0.01605272,
        0.03702165],
       [ 0.01958943,  0.01528796,  0.014183 ,  0.01303068,  0.01218617,
        0.01037877,  0.10721304,  0.10242224,  0.10430857,  0.09637651,
        0.09026764,  0.08879961,  0.08329058,  0.08056764,  0.05087568,
        0.11122248],
       [ 0.19665842,  0.16987912,  0.1472908 ,  0.15126709,  0.16639754,
        0.01804266,  0.01076212,  0.01636253,  0.01158352,  0.01534513,
        0.01692257,  0.01582116,  0.0149811 ,  0.01619452,  0.01367434,
        0.01881738],
       [ 0.0846402 ,  0.05134869,  0.0448965 ,  0.02739941,  0.06479075,
        0.08965857,  0.12178511,  0.10108433,  0.06933865,  0.06611255,
        0.04579263,  0.04545658,  0.04594946,  0.04541178,  0.02242584,
        0.07390895],
       [ 0.05552511,  0.06858447,  0.08840183,  0.05694064,  0.06593607,
        0.07315068,  0.0683105 ,  0.07401826,  0.06073059,  0.05520548,
        0.04958904,  0.04881279,  0.0547032 ,  0.06223744,  0.03885845,
        0.07899543],
       [ 0.08761392,  0.11043911,  0.04623032,  0.08272577,  0.05306545,
        0.054971  ,  0.06855841,  0.04212925,  0.0471831 ,  0.05207125,
        0.05103563,  0.04341342,  0.07812759,  0.07829329,  0.03309859,
        0.07104391],
       [ 0.13498765,  0.12815901,  0.14689205,  0.02935707,  0.11514191,
        0.12354053,  0.12425693,  0.03110996,  0.01949517,  0.02268085,
```

```
        0.02079078,  0.02205591,  0.02324483,  0.02278755,  0.0135201 ,
        0.0219797 ]])
>>> pop.max_city_age_group_percentage()
['Haifa', 'Haifa', 'Haifa', 'Haifa', 'Haifa', "Rishon Le'zion", "Rishon Le'zion", 'Ashdod', 'Ashdod',
'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod']
>>> pop.max_city_age_group_quantative()
['Haifa', 'Haifa', 'Haifa', 'Haifa', 'Haifa', "Rishon Le'zion", 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod',
'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod', 'Ashdod']
>>> pop.find_friendly_neighborhood([51,56,63])
'Ashdod'
>>> pop.find_friendly_neighborhood([11,36,76])
'Netanya'
```



שאלה 3 – עיבוד תמונה

ניתן לייצג תמונות בגווני אפור כמערך ndarray דו מימדי בו כל איבר הינו מספר בטווח 0-255 כאשר 0 מייצג שחור ו-255 מייצג לבן.

נרצה לקחת שתי תמונות בגווני אפור ולשלב אותן זו בזו.

שימו לב: השתדלו לעשות שימוש בפקודות Numpy ככל האפשר, אין להשתמש בלולאות או ברקורסיה.

א. ממשו את הפונקציה **fix_contrast(image)** המקבלת תמונה (מערך דו מימדי) בגוונים של אפור ומוודאת כי הניגודיות של התמונה מקסימלית.

- הפונקציה תבדוק כי הערך המינימלי בתמונה הוא 0 והמקסימלי הוא 255
- אם טווח הערכים איננו מקסימלי, הפונקציה תמתח את טווח הערכים כך שיהיה מקסימלי (0-255)
- **הפונקציה תחזיר תמונה (מערך דו מימדי) מתוקנת**

הדרכה: יש להחסיר את הערך המינימלי מכולם (על מנת שהערכים יתחילו ב-0) ואז להכפיל ב255 חלקי הערך המקסימלי (החדש לאחר ההחסרה)

$$\text{New_pixel_val} = \frac{(\text{old_pixel_val} - \text{min_val_in_image}) * 255}{\text{max_val_in_image} - \text{min_val_in_image}}$$

ב. ממשו את הפונקציה **merge_images(image1,image2)**

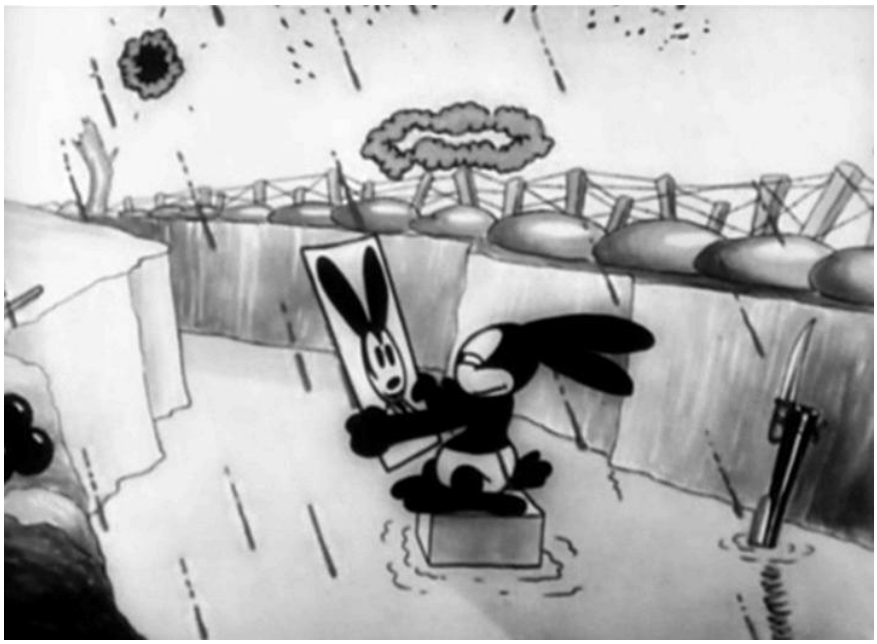
- הפונקציה תקבל שתי תמונות (מערכים דו מימדיים) באותו גודל
- תתקן את הניגודיות שלהן (עם הפונקציה מסעיף א')
- תחבר אותן על ידי השוואה בינהן ובחירת הפיקסל הכהה יותר לתמונה המאוחדת ותחזיר את התמונה החדשה

מצורפים קבצי תמונה עליהן ניתן לבדוק את הקוד, וקובץ התוצאה

דוגמאות הרצה:

תמונה עם טווח מצומצם:

תמונה מתוקנת:



תמונות לפני איחוד:

אחרי איחוד: