# תרגיל בית 6

## numpy, image processing

<u>הנחיות כלליות:</u>

- קראו היטב את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- אין לשנות את שמות הפונקציות והמשתנים שכבר מופיעים בקובץ השלד של התרגיל.
- אין לייבא (import) חבילות חיצוניות מעבר לחבילות המיובאות בקובץ השלד של התרגיל.
- אין למחוק את ההערות שמופיעות בשלד.
- **יש לבצע בדיקה עצמית אך ורק בסוף קובץ השלד בתוך התנאי הבא:**
  **if __name__ == "__main__":**
  **בפרט אין להשאיר הדפסות/קריאות לפונקציות בין הפונקציות.**
- **אין להשאיר בקוד נתיבים לקבצים מקומיים.**
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל השאלות יחד בקובץ ex6_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת.
- <u>אופן ביצוע התרגיל:</u> בתרגיל זה עליכם לממש את הפונקציות הנתונות. ניתן להוסיף פונקציות עזר.
- מועד אחרון להגשה: כמפורסם באתר.
- היות ובדיקת התרגילים עשויה להיות אוטומטית, יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).
- <u>בדיקה עצמית:</u> כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה, הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל ו**קלטים נוספים עליהם חשבתם** (וודאו כי הפלט נכון וכי התוכנית אינה קורסת).
- בכל השאלות ניתן להניח את תקינות הקלט על פי המפורט בשאלה.

# שאלה 1 – numpy

בשאלה נעסוק בטבלת מידע המתארת את תוצאות הניסוי של תרופת פלא חדשה לירידה במשקל. במשך מספר חודשים נאספו על-ידי ד"ר גריי נתונים של מדידות משקל חודשיות עבור מספר נסיינים, כך שבעמודה הראשונה מדידות משקל של כל הנסיינים בחודש הראשון לניסוי, בעמודה השנייה המדידות בחודש השני, וכך הלאה.

במהלך השאלה ניתן להניח שמקבלים את טבלת המדידות כמטריצת numpy **ללא שמות של חודשים או משתתפים**. אלה יסופקו לפי הצורך בנפרד. כמו כן, במהלך השאלה לא ניתן להשתמש בלולאות כלל. במקום זאת, נשתמש באפשרויות שמספקת לנו ספריית numpy.

<u>טבלה לדוגמא (רק התאים המודגשים הם חלק ממטריצת המדידות- training_data):</u>

| participant_name / month | November | December | January | February | March | April |
|---|---|---|---|---|---|---|
| Jane | **92.3** | **91.5** | **89.4** | **89.2** | **88.6** | **85.9** |
| Naomi | **104.6** | **102.1** | **100.8** | **98.5** | **96.3** | **94.4** |
| John | **73.2** | **72.0** | **72.6** | **71.4** | **71.2** | **70.9** |
| Moshe | **78.3** | **78.0** | **77.2** | **75.8** | **74.7** | **74.4** |

1. יש לממש את הפונקציה:

   get_highest_weight_loss_participant(training_data, participant_names)

   אשר מקבלת מטריצת numpy של המדידות (training_data) וכן רשימת שמות המשתתפים בניסוי (participant_names). הפונקציה תמצא את המשתתף שירידתו במשקל מתחילת התכנית ועד הסוף הייתה הכי גדולה. **יש לממש את הפונקציה ללא שימוש בלולאות, בשתי שורות לכל היותר.**

   **הערות:**

   - סדר השורות במטריצה תואם את סדר המשתתפים ברשימה. כלומר, השורה ה-i במטריצה שייכת למשתתף ה-i ברשימת המשתתפים בניסוי.
   - ניתן להניח שתמיד קיים משתתף יחיד שערך ירידתו במשקל היא המקסימלית.
   - **שימו לב**, ניתן (אבל לא חובה) להשתמש בפונקציה numpy.argmax, אשר מחזירה את האינדקס בו נמצא הערך המקסימלי.

   <u>דוגמת הרצה על הטבלה למעלה:</u>

```
>>> get_highest_weight_loss_participant(training_data, participant_names)
'Naomi'
```

2. נגדיר "הפרש חודשי" כהפרש המשקל של משתתף מסוים בין **חודש לחודש שאחריו**. יש לממש את הפונקציה get_diff_data(training_data), אשר מקבלת את מטריצת המדידות החודשיות training_data ומחזירה את מטריצת ההפרשים, כך **שבעמודה ה-i יופיע המשקל בחודש ה-i+1 פחות המשקל בחודש**

**ה-i**, ומספר העמודות סה"כ יהיה קטן ב-1 ממספר העמודות במטריצת הקלט. **יש לממש את הפונקציה ללא שימוש בלולאות, בשורה אחת בלבד.**

<u>דוגמת הרצה על הטבלה למעלה:</u>

```
>>> get_diff_data(training_data)
array([[-0.8, -2.1, -0.2, -0.6, -2.7],
       [-2.5, -1.3, -2.3, -2.2, -1.9],
       [-1.2,  0.6, -1.2, -0.2, -0.3],
       [-0.3, -0.8, -1.4, -1.1, -0.3]])
```

3. ד״ר גריי רוצה לבדוק האם יש חודש מסוים שבו משתתפים ירדו יותר במשקל מחודשים אחרים. לצורך כך, יש לממש את הפונקציה get_highest_change_month(training_data, study_months), אשר מקבלת את המדידות החודשיות training_data ואת רשימת החודשים בהם התרחש הניסוי, ומחזירה את החודש שבו התרחש השינוי הגדול ביותר ממנו לחודש שאחריו בסה״כ ע״פ כל המשתתפים. **יש לממש את הפונקציה ללא שימוש בלולאות, בשלוש שורות לכל היותר.**

<u>דוגמת הרצה על הטבלה למעלה:</u>

```
>>> get_highest_change_month(training_data, study_months)
'March'
```

הסבר: סכום ההפרשים בין חודש מרץ לאפריל הינו 5.2-, בעוד שההפרשים בין החודשים האחרים הינם: 4.1-, 5.1-, 3.6-, 4.8-.

4. ד״ר גריי רוצה לבדוק גם האם יש משתתפים שלא ירדו במשקל באופן קונסיסטנטי לאורך כל המחקר. כלומר, היו חודשים בהם הם עלו במשקל או נשארו באותו משקל כמו בחודש הקודם. לצורך כך, יש לממש את הפונקציה get_inconsistent_participants(training_data, participant_names) המקבלת את המדידות החודשיות training_data ואת רשימת שמות המשתתפים, ומחזירה את רשימת המשתתפים שלא ירדו במשקל בכל חודשי המחקר. אם אין משתתפים כאלה, הפונקציה תחזיר רשימה ריקה. **יש לממש את הפונקציה ללא שימוש בלולאות, בארבע שורות לכל היותר.**

<u>דוגמת הרצה על הטבלה למעלה:</u>

```
>>> get_inconsistent_participants(training_data, participant_names)
['John']
```

הסבר: בין דצמבר לינואר המשקל של ג׳ון עלה.

## שאלה 2 – עיבוד תמונה

בתרגיל זה נממש קוד למיזוג תמונות (Image blending) ע״י פירמידות.

כמו שראינו בכיתה, ניתן להשתמש במערכים של Numpy כדי לייצג תמונות.
בתרגיל נעבור עם תמונות grayscale- תמונות בגווני אפור, כמו שראינו בכיתה.
נגדיר תמונת מסיכה (mask) להיות תמונה בינרית- תמונת אפור המורכבת אך ורק מהצבעים שחור ולבן.
**שימו לב- בכל הסעיפים בשאלה זו אין לשנות את התמונות המקוריות (כלומר יש להחזיר תמונות חדשות), ואין להשתמש בלולאות, אלא אם נאמר אחרת.**

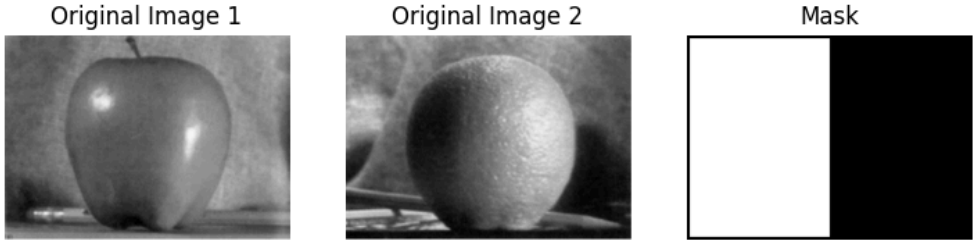
1. ראינו בכיתה שתמונות אפור מיוצגות ע״י מערכים דו-מימדיים (שמימדם (h,w)), כך שכל איבר במערך הוא מספר המייצג את הצבע של הפיקסל שנמצא במיקום שלו בתמונה.
ממשו את הפונקציה read_image(img_path) המקבלת נתיב לתמונה ו-mode לקריאת התמונה. mode יקבל את הערך 'L' המייצג תמונת אפור באופן דיפולטיבי. על הפונקציה להחזיר תמונה המיוצגת ע״י מערך numpy עם ערכים מסוג uint8. לצורך כך, השתמשו בפונקציה imread המיובאת לכם בראש שלד התרגיל עם הפרמטרים img_path, mode.

דוגמאות הרצה לפונקציה read_image:

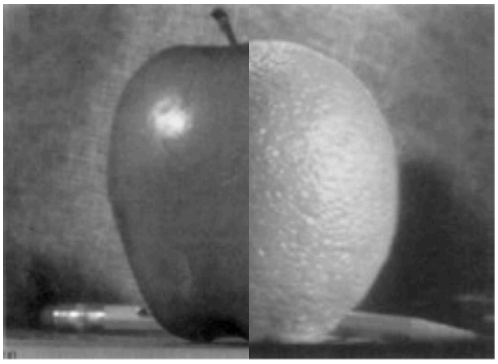
```
>>> read_image('apple.png')
array([[159, 159, 157, ..., 191, 191, 194],
       [157, 157, 156, ..., 179, 180, 184],
       [154, 154, 154, ..., 167, 169, 174],
       ...,
       [146, 147, 148, ..., 153, 154, 162],
       [186, 186, 186, ..., 189, 190, 194],
       [226, 226, 226, ..., 227, 229, 230]], dtype=uint8)
```

2. ממשו את הפונקציה naive_blending(img1, img2, mask) המקבלת 2 תמונות ותמונת מסיכה ומחזירה תמונה חדשה, כך שאיזור המסיכה שערכו 255 יורכב מפיקסלים מהתמונה הראשונה, והאיזור שערכו 0 יורכב מפיקסלים מהתמונה השנייה.

דוגמה: עבור המסיכה הזו והתמונות האלה של תפוח (img1) ותפוז (img2):




הפונקציה תחזיר את התמונה הזו כפלט:



3. בסעיפים הבאים תתבקשו לממש מספר פונקציות שבעזרתן בסוף נוכל לקבל תמונה הממזגת בין שתי תמונות בצורה חלקה ע״י אלגוריתם הנקרא Pyramid Blending.

א. ממשו את הפונקציה blur_and_downsample(img, nb_size) המקבלת תמונה ומספר nb_size.
הפונקציה תטשטש את התמונה ע״י שימוש בתוחלת (mean) על פני חלון סביב כל פיקסל כפעולה מורפולוגית. המספר nb_size יגדיר את גודל החלון סביב הפיקסל.
לאחר הטשטוש, עליכם להקטין את רזולוציית התמונה פי 2 גם ברוחב וגם בגובה, ע״י דגימה של כל פיקסל שני מכל שורה שנייה. למשל, עבור תמונה בגודל 256x256, הפונקציה תחזיר תמונה חדשה בגודל 128x128.
המלצה: העזרו בפונקציות עזר לצורך טשטוש התמונה. **ניתן להשתמש בלולאות למימוש הטשטוש, אבל לא לצורך הקטנת התמונה.**
כשנדפיס את מימד התמונה המקורית ומימד התמונה החדשה, נרצה לראות שהיא קטנה פי 2. למשל עבור תמונת התפוח נקבל:

```
>>> print(im.shape)
(448, 624)
>>> print(blur_and_downsample(im).shape)
(224, 312)
```

ב. ממשו את הפונקציה build_gaussian_pyramid(img, max_levels, nb_size) המקבלת תמונת אפור, מס׳ שלם המייצג מספר רמות בפירמידה, ומס׳ nb_size המייצג את גודל החלון כפרמטר טשטוש.
הפונקציה תחזיר רשימה, כך שכל איבר ברשימה מכיל תמונה קטנה מזו שלפניה פי 2, ע״י שימוש בפונקציה מהסעיף הקודם. התמונה הראשונה בפירמידה תהיה התמונה המקורית. **במימוש הפונקציה הזו ניתן להשתמש בלולאות.**
דוגמה: עבור קלט שהינו תמונה בגודל (448,624), max_levels=3, nb_size=5, הפונקציה תחזיר רשימה באורך 3 המכילה תמונות בגדלים: [(448,624),(224,312),(112,156)].

תמונות הפירמידה לדוגמה (טשטוש הולך וגדל):

שימו לב שהתמונות כאן מוגדלות לאותו הגודל לצורך המחשת הטשטוש, אבל בפלט הפונקציה שלכם הן יהיו בגדלים שונים, כאשר התמונה באינדקס 0 תהיה הכי גדולה (בגודל התמונה המקורית) ואחריה כל תמונה קטנה פי 2.

ג. ממשו את הפונקציה upsample_and_blur(img, nb_size( המקבלת תמונה ומס׳ nb_size המייצג את גודל החלון כפרמטר טשטוש, ומחזירה תמונה חדשה עם רזולוציה גדולה פי 2 ברוחב ובגובה מתמונת הקלט. לצורך ההגדלה, התחילו בשכפול כל שורה, כך שלמשל תמונה בגודל 128x128 תהפוך לתמונה בגודל 256x128. אז, שכפלו כל פיקסל בכל שורה כך שהתמונה תהפוך לתמונה בגודל 256x256. לאחר ההגדלה, בצעו טשטוש לתמונה עם הפרמטר nb_size.

רמז: לצורך שכפול השורות והעמודות השתמשו בפונקציה `np.repeat`.

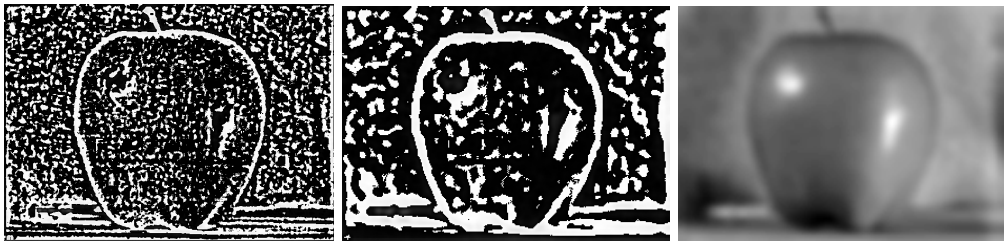
ד. ממשו את הפונקציה build_laplacian_pyramid(img, max_levels, nb_size) המקבלת תמונת אפור, מס׳ שלם המייצג מספר רמות בפירמידה, ומס׳ nb_size המייצג את גודל החלון כפרמטר טשטוש.
הפונקציה תייצר פירמידה גאוסיאנית לתמונה ע״י קריאה לפונקציה מסעיף ב׳:
G_pyramid = build_gaussian_pyramid(img, max_levels, nb_size)

ותחזיר רשימה, כך שכל איבר ברשימה מכיל תמונת לפלסיאן. תמונת הלפלסיאן בכל רמה בפירמידה, הינה תמונת הגאוסיאן ברמה זו, פחות תמונת הגאוסיאן של הרמה הבאה, מוגדלת ע״י הפונקציה שמימשתם בסעיף ג׳. כלומר, אם נסמן ב-L את פירמידת הלפלסיאן, וב-G את פירמידת הגאוסיאן, באינדקס ה-i בין 0 ל-max_levels-2:

$$L[i] = G[i] - upsample(G[i+1])$$

והאיבר האחרון ברשימה יהיה הגאוסיאן האחרון $G[-1]$ ללא שינויים. **במימוש פונקציה זו ניתן להשתמש בלולאות.**

דוגמה לתמונות לפלסיאן (פירמידה עם 3 רמות, nb_size=5):

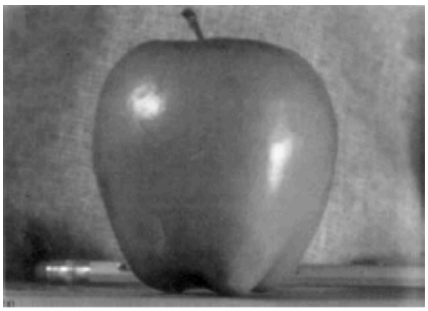


שימו לב שהתמונות כאן מוגדלות לאותו הגודל לצורך המחשה, אבל בפלט הפונקציה שלכם הן יהיו בגדלים שונים, כאשר התמונה באינדקס 0 תהיה הכי גדולה (בגודל התמונה המקורית) ואחריה כל תמונה קטנה פי 2. **בשלב זה אין לתקן מצבי overflow בפעולות בין התמונות.**

ה. ממשו את הפונקציה laplacian_pyramid_to_image(laplacian_pyramid, nb_size) המקבלת פירמידת לפלסיאן ומס׳ nb_size המייצג את גודל החלון כפרמטר טשטוש. הפונקציה תשחזר את התמונה המקורית מפירמידת הלפלסיאן ע״י חיבור כל תמונות הלפלסיאן, כך שבכל שלב, החל מהתמונה האחרונה (הקטנה ביותר), מגדילים את תמונת הלפלסיאן הנוכחית ומחברים אליה את תמונת הלפלסיאן הנמצאת בשלב הקודם.

כלומר, נגדיר את התמונה ההתחלתית להיות תמונת הלפלסיאן האחרונה, ואז בצורה איטרטיבית, עבור i בין max_levels-2 ל-0: $img = L[i] + upsample(img)$. **במימוש פונקציה זו ניתן להשתמש בלולאות.**

בסוף נחזיר את התמונה המשוחזרת. מימד התמונה המשוחזרת צריך להיות כמו מימד התמונה המקורית. דוגמה לשחזור תמונת התפוח:



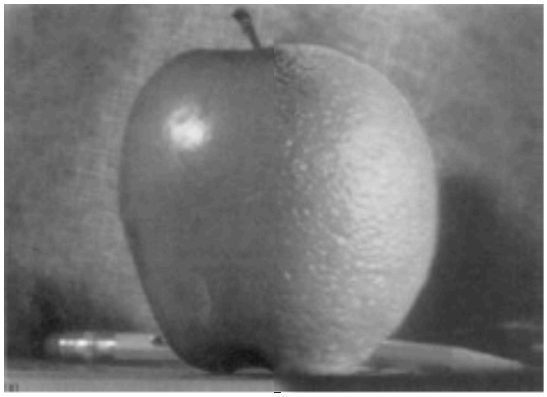
ו. נחבר הכל ביחד!
ממשו את הפונקציה pyramid_blending(img1, img2, mask, max_levels, nb_size) המקבלת שתי תמונות אפור, תמונת מסיכה, מס׳ רמות בפירמידה, ומס׳ nb_size המייצג את גודל החלון כפרמטר טשטוש.
הפונקציה תחזיר תמונה ממוזגת של שתי התמונות לפי המסיכה, כך שאיזור המסיכה שערכו 255 יורכב מפיקסלים מהתמונה הראשונה, והאיזור שערכו 0 יורכב מפיקסלים מהתמונה השנייה, והמעבר בין התמונות יהיה חלק.
בפונקציה זו נממש את האלגוריתם הבא המשתמש בפונקציות שמימשתם בסעיפים הקודמים:
1. נבנה פירמידת לפלסיאן לכל תמונה מבין img1, img2
2. נבנה פירמידת גאוסיאן למסיכה mask. **עם זאת**, נבחין כי לאחר פעולות טשטוש יהיו במסיכה ערכים "אפורים", לא רק 0 ו-255. **כדי לתקן זאת**, נתייחס לכל פיקסל שערכו לא 0 (שחור מוחלט) כ-255 (לבן מוחלט).
3. ניצור פירמידת לפלסיאן חדשה, המשלבת בין תמונות הלפלסיאן בכל רמה בהתאם למסיכה. כלומר, עבור i בין 0 ל-max_levels-1 הערך בפירמידה החדשה $L_B$ יהיה:

$$L_B[i] = mask \bullet L_1[i] + (1 - mask) \bullet L_2[i]$$

לצורך כך ניתן להשתמש בפונקציה naive_blending שמימשתם בתחילת התרגיל.
4. נשתמש בפונקציה מהסעיף הקודם laplacian_pyramid_to_image עם פירמידת הלפלסיאן החדשה כדי לקבל את התמונה הממוזגת.
5. נחזיר את התמונה הממוזגת.

**במימוש פונקציה זו ניתן להשתמש בלולאות.**

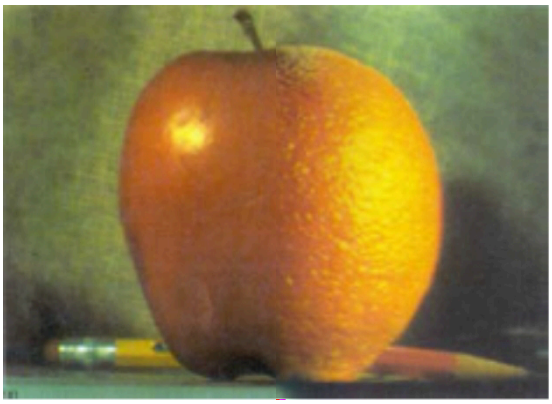
דוגמה לתמונות התפוח והתפוז ממוזגות:



ז. **בונוס (ללא ציון):** מיזוג תמונות בצבע.
עד כה השתמשנו בתמונות אפור. כדי לעבוד עם תמונות צבע, יש לעבוד עם ייצוג שונה של תמונות. ייצוג נפוץ לתמונות צבע נקרא RGB, שמשמעותו (Red, Green, Blue), כך שבהתאמה כל פיקסל בתמונה מיוצג ע״י 3 ערכים המתאימים לכל אחד מהצבעים אדום, ירוק וכחול, והשילוב ביניהם מגדיר את הצבע שלו.
כלומר, תמונות RGB מיוצגות ע״י מערכים תלת-מימדיים שמימדם (h,w,3). כך למשל, (0,0,0) מייצג שחור, (255,0,0) מייצג אדום, (128,0,128) מייצג סגול, ו-(255,255,255) מייצג לבן.

- כדי לעבוד עם תמונות RGB נצטרך לקרוא לפונק׳ read_image משאלה 1 עם 'mode='RGB עבור תמונות צבע. ודאו שעבור 'mode=='L, הפונקציה שלכם מחזירה תמונת אפור, ועבור mode == 'RGB'', היא מחזירה תמונת RGB.
- כדי למזג תמונת RGB, נצטרך להפעיל את פונקציית המיזוג שלנו מסעיף ו׳ על כל ערוץ צבע בנפרד.
  ממשו פונקציה אחרונה pyramid_blending_RGB_image(img1, img2, mask, max_levels, nb_size) המפעילה את הפונקציה מהסעיף הקודם pyramid_blending על כל ערוץ צבע בתמונת RGB (כלומר פעם אחת על אדום, פעם אחת על ירוק, ופעם אחת על כחול), ומחברת את כל תמונות התוצאה לכדי תמונה אחת.
  רמז: כל תוצאה בודדת תהיה במימד (h,w) ועל התוצאה הסופית להיות במימד (h,w,3).

בסוף, כשנפעיל את הפונקציה מהסעיף הזה על שלשת התפוח-תפוז-מסיכה מתחילת התרגיל (בצבע), נקבל את התוצאה הזו:



**כלי עזר לפיתוח:** השתמשו ב-matplotlib.pyplot המיובא בראש שלד התרגיל כ-plt על מנת להציג את התמונות שלכם במהלך הפיתוח וודאו שאתם מקבלים את התמונות הרצויות. בתחתית השלד (ב-main, איזור ה"טסטים") תוכלו למצוא דוגמאות לשימוש ב-plt.
אם לא תרצו להציג אותם, תוכלו להפוך את המשתנה `plot_flag` בראש ה-main ל-False.

# בהצלחה!