תרגיל בית 9

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 למחוק את ההערות שמופיעות בשלד.
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל השאלות יחד בקובץ ex9_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.
- **אין להשתמש בספריות חיצוניות (ובפונקציות שלהן) מעבר למה שסופק בשלד התרגיל.** כלומר, אין להשתמש בפקודת import. **כל פונקציה שלא דורשת פקודה זו מותרת לשימוש** (כלומר, זו פונקציה שהמתרגם (interpreter) מכיר ללא פקודה זו).
- 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 וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון וכי התוכנית אינה קורסת)
- בכל השאלות ניתן להניח את תקינות הקלט על פי המפורט בשאלה.

שאלה 1

Galactic federationה רצה לבדוק את ההישגים של תכנית אימונים חדשה לחיילים שלה. במשך כמה חודשים רצופים נאספו נתוני החיילים בטבלאות csv, כך שבעמודה הראשונה מופיעים שמות המשתתפים, בשורה הראשונה שמות החודשים, וכל שורה הינה איסוף של הנתונים בסוף כל חודש. (ראו את קובץ הדוגמא המצורף לתרגיל weights.csv, בו נמדדו משקליהם של ארבעה החיילים בקילוגרמים. חודש הדגימה הראשון הוא אוגוסט)

בנוסף, קיימת טבלת csv לנתוני גובה של החיילים כך שבעמודה הראשונה מופיעים שמות המשתתפים, בשורה הראשונה קיימת הכותרת height , ובכל שורה גובה חייל (ראו את קובץ הדוגמא המצורף לתרגיל heights.csv, בו נמדדו גובהם של ארבעה חיילים בסנטימטרים (int). שימו לב כי **לא ניתן להניח** כי סדר השמות (השורות) בקובץ הגבהים זהה לזה שבקובץ המשקלים.

שימו לב כי בסעיפים ב',ד',ה' אין להשתמש בכל בלולאות (for, while).

א. ממשו את הפונקציה load_training_data(weights_file, heights_file):
הפונקציה תקבל את הנתיבים לקבצי ה-csv של המשקלים (weights_file) ושל הגבהים (heights_file).
הפונקציה תחזיר 2 מילונים:
המילון הראשון המכיל שלושה מפתחות מטיפוס מחרוזת הממופים לערכים הבאים:
- "data": ממופה למטריצת numpy המכילה את נתוני הטבלה (ללא שורת החודשים או עמודת שמות המתאמנים).
- "column_names": ממופה לרשימת שמות החודשים לפי סדר העמודות (מהשורה הראשונה ללא התא הראשון), מטיפוס numpy.array של מחרוזות.
- "row_names": ממופה לרשימת המשתתפים לפי סדר השורות (מהעמודה הראשונה ללא התא הראשון), מטיפוס numpy.array של מחרוזות.

המילון השני מכיל בתור ערכי מפתח את שמות החיילים וגובהם של החיילים בתור ערכים (כפי שמופיעים בקובץ נתוני הגובה).

הבהרות:
- ניתן להניח כי קלט קבצי ה*csv* תקין, וכי קיים יותר מחודש אחד (כלומר, יותר מעמודה אחת) בקובץ המשקלים
- ניתן להניח כי טווח החודשים מלא בקובץ המשקלים (אין חודשים חסרים בטווח) וכן כי החודשים מסודרים לפי הסדר עולה
- **לא ניתן להניח** כי סדר השמות (השורות) בקובץ הגבהים זהה לזה שבקובץ המשקלים

קבצים לדוגמא המצורפים לתרגיל:

weights.csv

	August	September	October	November	December	January
Krombopulos	84	81.3	82.8	80.1	77.4	75.2
Michael	79.6	75.2	75	74.3	72.8	71.4
Cornvelious	67.5	66.5	65.3	65.9	65.6	64
Daniel	110.7	108.2	104.1	101	98.3	95.5

heights.csv

	Height
Krombopulos	190
Michael	182
Cornvelious	171
Daniel	193

```
>>> weight_dict, height_dict = load_training_data("weights.csv",
"heights.csv")
>>> print(weight_dict["data"])
[[ 84.   81.3  82.8  80.1  77.4  75.2]
 [ 79.6  75.2  75.   74.3  72.8  71.4]
 [ 67.5  66.5  65.3  65.9  65.6  64. ]
 [110.7 108.2 104.1 101.   98.3  95.5]]

>>> print(weight_dict["column_names"])
['August' 'September' 'October' 'November' 'December' 'January']

>>> print(weight_dict["row_names"])
['Krombopulos ' 'Michael' 'Cornvelious' 'Daniel']

>>> print(height_dict)
{'Krombopulos ': 190, 'Michael': 182, 'Cornvelious': 171, 'Daniel': 193}
```

ב. ממשו את פונקציה get_highest_weight_loss(data_dict) :

הפונקציה תקבל את **מילון** המשקלים (data_dict) (אחד הפלטים מסעיף א').

הפונקציה תחזיר את שם החייל שירידתו במשקל הייתה הכי גדולה מתחילת התכנית ועד סופה (ניתן להניח שיש אחד כזה). כלומר, שההפרש בין משקל ההתחלה למשקל הסיום הגדול ביותר.

ניתן (אבל לא חובה) להשתמש בפונקציה numpy.argmax שמחזירה את האינדקס שבו נמצא הערך המקסימלי.

דוגמת הרצה:

```
>>> print(f'get_highest_weight_loss:\n {get_highest_weight_loss(weights)}\n======')
get_highest_weight_loss:
 Daniel
======
```

ג. BMI (body-mass-index) הוא מדד הנותן הערכה כמותית האם אדם נמצא במשקל תקין, בעודף משקל או בתת משקל (ראה https://en.wikipedia.org/wiki/Body_mass_index).

הbmi מחושב באופן הבא: $\frac{\mathbf{weight}}{\left(\frac{\mathbf{height}}{\mathbf{100}}\right)^{\mathbf{2}}}$.

ממשו את פונקציה get_bmi(weights_data, heights_data):
הפונקציה תקבל את **מילון** המשקלים (weights_data) ו**מילון** הגבהים (heights_data) (פלט סעיף א').
הפונקציה תחזיר את טבלת ה BMI (body-mass-index) של החייל לפי חודשים.

יש לעגל את התוצאה בדיוק של **שתי (2)** ספרות לאחר הנקודה. למשל עבור משקל 72 ק"ג וגובה 182 ס"מ ערך ה BMI הוא ... $\frac{72}{\left(\frac{182}{100}\right)^2} = 21.736505253$ ולכן הערך בטבלה יהיה **21.74**

דוגמת הרצה:

```
>>> print(f'get_bmi:\n {get_bmi(weights, heights)}\n======')
get_bmi:
 [[23.27 22.52 22.94 22.19 21.44 20.83]
 [24.03 22.7  22.64 22.43 21.98 21.56]
 [23.08 22.74 22.33 22.54 22.43 21.89]
 [29.72 29.05 27.95 27.11 26.39 25.64]]
======
```

שימו לב: בסעיף זה (ג') ניתן להשתמש בלולאות. **מימוש ללא לולאות יזכה את הסטודנט בבונוס (תופחת שגיאה אחת בחישוב הציון)!**
רמז: ניתן (אך לא חובה) להשתמש בפונקציות **numpy** הבאות: **meshgrid, apply_along_axis, dstack**

שימו לב באמצעות **meshgrid** ניתן ליצור 2 מטריצות שאחת מייצג את המיקומים בציר ה**X** (העמודות בטבלה) והאחרת את המיקומים בציר ה**Y** (השורות בטבלה). שימוש בפונקציה זו (יחד עם **apply_along_axis** ו**dstack**) יכול לסייע לפתור את השאלה ללא לולאות.
לדוגמא, בהנתן טבלה עם 5 שורות ו3 עמודות:

```
>>> xx,yy= np.meshgrid(np.arange(5),np.arange(3))
>>> xx
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
>>> yy
array([[0, 0, 0, 0, 0],
       [1, 1, 1, 1, 1],
       [2, 2, 2, 2, 2]])
```

התא (ערך התא במיקום **(1,3)** הוא 1 ב**yy** ו3 ב**xx**

ד. נגדיר את ה-'הפרש החודשי' כהפרש הBMI עבור משתתף מסויים בין חודש אחד לחודש הקודם לו.
ממשו את הפונקציה get_bmi_diff(weights_data, heights_data)
הפונקציה תקבל את **מילון** המשקלים (weights_data) ו**מילון** הגבהים (heights_data) (פלט סעיף א').
הפונקציה תחזיר את מטריצת ההפרשים. בעמודה הi יופיע ההפרש בין החודש ה i+1 לבין החודש ה i (במטריצת המשקלים), וסה"כ מספר העמודות במטריצת ההפרשים יהיה קטן ב 1 ממספר העמודות במטריצת הנתונים.
שימו לב לא לשנות את מטריצת הקלט.

- ניתן להשתמש בפונקציה מסעיף ג' (get_bmi) לצורך פתרון סעיף זה

דוגמת הרצה:

```
>>> print(f'get_bmi_diff:\n {get_bmi_diff(weights, heights)}\n======')
get_bmi_diff:
 [[-0.75  0.42 -0.75 -0.75 -0.61]
 [-1.33 -0.06 -0.21 -0.45 -0.42]
 [-0.34 -0.41  0.21 -0.11 -0.54]
 [-0.67 -1.1  -0.84 -0.72 -0.75]]
======
```

ה. כתוב פונקציה get_highest_bmi_loss_month(weights_data, heights_data):

הפונקציה תקבל את **מילון** המשקלים (weights_data) ו**מילון** הגבהים (heights_data) (פלט סעיף א').
הפונקציה תחזיר את שם החודש שבו **הירידה** בBMI הייתה מקסימלית על פני כל החיילים (כלומר, החודש בו סכימת השינויים בBMI נותנת את המספר **הנמוך** ביותר).

- ניתן להשתמש בפונקציות מסעיפים ג' (get_bmi) וד' (get_bmi_diff) לצורך פתרון סעיף זה
- ניתן להניח שיש חודש מקסימלי אחד

דוגמת הרצה:

```
>>> print(f'get_highest_bmi_loss_month:\n {get_highest_bmi_loss_month(weights
, heights)}\n======')
get_highest_bmi_loss_month:
 September
======
```

ו. ידוע ששינוי במשקל הוא יחסי למשקל ההתחלתי. כתוב פונקציה get_relative_bmi_diff_table שתחזיר את טבלת השינוי במשקל, כך שלכל חייל בכל חודש יופיע השינוי היחסי לBMI בחודש הקודם, כלומר, ה'הפרש החודשי' לחלק לBMI בחודש הקודם.

- שימו לב כי קלט הפונקציה הוא נתוני המשקל והגובה כפי שמיוצגים בפלט סעיף 1.
- השתמשו בפונקציה מסעיף 3 לצורך פתרון סעיף זה
- יש לעגל את התוצאה בדיוק של **<u>שלוש</u> (3)** ספרות לאחר הנקודה.

דוגמת הרצה:

```
>>> print(f'get_relative_diff_table:\n {get_relative_diff_table(weights,
heights)}\n======')
get_relative_diff_table:
[[-0.032  0.019 -0.033 -0.034 -0.028]
 [-0.055 -0.003 -0.009 -0.02  -0.019]
 [-0.015 -0.018  0.009 -0.005 -0.024]
 [-0.023 -0.038 -0.03  -0.027 -0.028]]
======
```

למשל, הנתון המודגש בפלט לעיל הוא תוצאת החישוב:

$$\frac{november-october}{october} = \frac{22.43-22.64}{22.64} = -0.00927561837$$

ולאחר עיגול התוצאה: -0.009

שאלה 2

אחד הגדלים החשובים בתורת האינפורמציה הוא **האנטרופיה.** זהו אינדקס המכמת את חוסר הסדר שמכילה מילה, תמונה או אירוע. חישבו על תמונה בעלת גוון אחד בלבד, למשל שחור. תמונה זו היא מאוד "מסודרת" (חישבו כמה מילים צריך כדי לתאר את התמונה הזאת) ואכן ערך האנטרופיה שלה הוא 0 (למה?). לעומת זאת חישבו על תמונה בעלת מנעד רחב של גווני אפור – תמונה זו מכילה המון אינפורמציה – והיא פחות "מסודרת" וערך האנטרופיה שלה גדול.

בסעיף זה נרצה לחשב את האנטרופיה של תמונה בגווני אפור ובכך לכמת את אי הסדר שבתמונה. הנוסחא לחישוב אנטרופיה היא:

$$S = \sum_{i=0}^{N} -P_i \cdot log_2 P_i$$

כאשר P_i היא השכיחות לקבל גוון אפור כלשהו בין 0 ל 255 ($P_i = \frac{\text{מספר מופעי ערך הפיקסל } i}{\text{מספר הפיקסלים בתמונה}}$). במילים אחרות P_i הם הערכים של ההיסטוגרמה המנורמלת של התמונה. N הוא מספר גווני האפור.

לדוגמא: נחשב את האנטרופיה של תמונה בעלת 4 פיקסלים. 2 פיקסלים בצבע שחור. ו2 פיקסלים בצבע לבן.

$$S = \sum_{i=0}^{N} -P_i \cdot log_2 P_i = \frac{-1}{2} \cdot log_2\left(\frac{1}{2}\right) - \frac{1}{2} \cdot log_2\left(\frac{1}{2}\right) = 1$$

במקרה הספציפי הזה, השכיחות של פיקסל לבן או שחור היא חצי.

ממשו את הפונקציה def compute_entropy(img) אשר מקבלת שם של תמונה img (string) ומחזירה את האנטרופיה (float) של תמונת גווני אפור.

- עליכם להתעלם מערכי P_i שהם אפס, (כלומר, לא לקח אותם בחשבון בסכום האנטרופיה).
- ניתן להניח כי הקלט תקין – תמונה בגודל כלשהו עם גווני אפור בין 0 ל 255.
- ניתן להניח שהתמונה מכילה יותר מגוון אפור אחד.
- רמז: ניתן אך לא חובה להשתמש ב *np.bincount*
- יש לעגל אל התוצאה לדיוק של **ארבע (4)** ספרות אחרי הנקודה

דוגמת הרצה (משלד התרגיל):

```
>>>print(compute_entropy("rick_and_morty_gray.png"))
>>> 7.0982
```

הבהרות:

- אין להשתמש בחבילות פייתון מעבר לאלו המיובאות לכם בשלד התרגיל

שאלה 3

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.

בשאלה זו נהפוך את התמונות לבינאריות (כלומר, שחור-לבן ללא גווני אפור) ונכווץ את הגודל שהתמונות תפוסות הזכרון. לצורך כך, נשתמש באלגוריתם בשם Run-length encoder (RLE).

האלגוריתם RLE. זה מקודד מטריצות של אפסים ואחדות למספרים המייצגים את אורך הסדרות המתחלפות ביניהן, כאשר **מניחים שהסדרה הראשונה היא סדרת אפסים**.

דוגמאות:

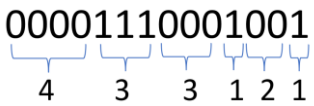
השורה הבאה:

00001110001001

שורה זו תקודד ל

433121

הסבר:



והשורה הבאה:

1100001110001001

שורה זו תקודד ל

02433121

הסבר:

שימו לב כי בדוגמא זו הקידוד מתחיל מ0, מכיוון שהשורה אותה מקודדים מתחילה ב1. זאת מכיוון שהמספר הראשון מקודד את מספר האפסים המתחילים את השורה.

הרחבה נוספת על האלגוריתם RLE, תוכלו למצוא קישור הבא:
https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:digital-information/xcae6f4a7ff015e7d:data-compression/a/simple-image-compression

א. כתוב פונקציה בשם load_image_as_matrix(img_path):
הפונקציה תקבל את הנתיב לקובץ (מסוג מחרוזת/str).
הפונקציה תחזיר מטריצת numpy דו מימדית המייצגת את ערכי הפיקסלים של התמונה.

ב. כתוב פונקציה בשם binarize_matrix המקבלת כקלט את המטריצה הדו-מימדית (הפלט מסעיף א') ומחזירה מטריצה דו מימדית **חדשה** המייצגת את הפיקסלים שבמטריצה המקורית **בשחור-לבן בלבד ללא ערכי אפור** בצורה הבאה:
- ערכי האפור קטנים מ128 יקבלו אתה ערך 0
- כל ערכי האפור מ128 (כולל) ומעלה יקבלו את הערך 1

הפונקציה תחזיר מטריצת numpy דו מימדית המייצגת את ערכי הפיקסלים של התמונה.

דוגמאת הרצה:

```
>>> img_original=np.array([[10,200,110],[0,205,100]])
>>> img_binarized =binarize_matrix(img_original)
>>> print(img_binarized)
array([[0., 1., 0.],
       [0., 1., 0.]])
```

ג. בסעיף זה נממש את אלגוריתם RLE **בשינוי קל**.
ממש את הפונקציה compress_flatten_rle(mat):
הפונקציה תקבל ייצוג **בינארי** מטריציוני בnumpy של התמונה (הפלט מסעיף ב').
הפונקציה **תשטח** את המטריצה לכדי וקטור באמצעות הפונקציה flatten בnumpy), ותריץ את האלגוריתם RLE על הוקטור (כפי שמוסבר בקישור בסעיף הקודם).
דוגמא להפעלת הפונקציה flatten:

```
>>> mat=np.array([[0,1,0],[0,1,0]])
>>> print(mat, mat.shape)
[[0., 1., 0.]
 [0., 1., 0.]] (2, 3)

>>> mat_flatten= mat.flatten()
>>> print(mat_flatten, mat_flatten.shape)
[0 1 0 0 1 0] (6,)
```

הפונקציה compress_flatten_rle(mat) תחזיר tuple המכיל שני משתנים:
1) וקטור (חד ממדי) בnumpy המייצג את התמונה המשוטחת **לאחר הכיווץ**
2) tuple המכיל את **אורך ורוחב המטריצה המקורית**.

דוגמאת הרצה:

```
>>> mat_rle_compressed, shape = compress_flatten_rle(img_binarized)
>>> print(mat_rle_compressed, shape)
(array([1, 1, 2, 1, 1], dtype=int64), (2, 3))
```

ד. כתבו פונקציה בשם decompress_flatten_rle(mat_rle_compressed, shape):
הפונקציה תקבל את הוקטור המייצג את התמונה **המשוטחת הדחוסה** (mat_rle_compressed), ו-tuple המכיל את אורך ורוחב המטריצה המקורית (shape) (הפלטים מסעיף ג').
הפונקציה תמיר את המטריצה הוקטור התמונה הדחוס חזרה **למטריצה הבינארית** בייצוג מטריציוני של numpy, ותחזיר את המטריצה.

- **שימו לב שפלט הפונקציה decompress_flatten_rle צריך להיות זהה לחלוטין לפלט הפונקציה binarize_matrix.**

דוגמאת הרצה:

```
>>> mat_rle_decompressed=decompress_flatten_rle(mat_rle_compressed, shape)
>>> print(mat_rle_decompressed)
array([[0., 1., 0.],
       [0., 1., 0.]])
```

ה. כתבו פונקציה בשם calc_compression_ratio המקבלת את ייצוג התמונה המכווץ (האיבר הראשון בפלט הפונקציה compress_flatten_rle) ואת התמונה הבינארית המקורית (פלט הפונקציה binarize_matrix) ומחזירה את יחס הכיווץ של התמונה. יחס הכיווץ הוא מספר האיברים (int) שהיו שמורים במטריצה המכווצת לעומת המקורית:

$$\frac{number\ of\ entries\ in\ compressed\ matrix}{number\ of\ entries\ in\ original\ matrix}$$

יש לעגל את התוצאה בדיוק של **<u>שתי</u> (2)** ספרות לאחר הנקודה.
דוגמת הרצה:

```
>>> print(f'calc_compression_ratio: {calc_compression_ratio(mat_rle_compresse
d, img_binarized)}')
calc_compression_ratio: 0.83
```

הערה: אנו מניחים כי הקלט בסעיף זה הוא תמונה **דו מימדית** (ללא מימד הצבע). כדי לבדוק סעיף זה על תמונות נוספות מלבד אלו שסיפקנו לכם יחד עם התרגיל, יש להפוך תמונות צבעוניות תלת מימדיות לתמונות שחור לבן דו מימדיות. דרך אחת לעשות זאת היא להשתמש בשורות הבאות:

```
from PIL import Image, ImageOps
ImageOps.grayscale(Image.open('rick_and_morty_color.png')).save('rick_and_morty_gray.png')
```

שימו לב שזהו רק עיבוד מקדים ושורות אלו לא אמורות להיות חלק מהתכנית שלכם