תרגיל בית 9

OOP, 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.
- 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 (OOP)

ספרות רומיות הוא שמה של שיטת ספירה אשר הומצאה ברומא העתיקה ונמצאה בשימוש בארצות אירופה עד לתקופה המאורחת של ימי הביניים. בשיטה זו, הספרות מיוצגים על ידי צירופי אותיות מהאלפבית הלטיני (לקריאה נוספת:

(https://en.wikipedia.org/wiki/Roman_numerals)

נרצה לממש מחלקה אשר מאפשרת ייצוג של מספרים בספרות רומיות ותומכת בפעולות אריתמטיות בסיסיות על מספרים אלה.

הספרות הרומיות הבסיסיות הן:

'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000

1. ממשו את המחלקה Roman אשר מאותחלת באמצעות מספר שם (int) אשר מכיל את הערך המספרי שנרצה לייצג באמצעות ספרות רומיות, או באמצעות מחרוזת אשר מכילה מספר הכתוב בספרות רומיות.
   התחילו במימוש של הבנאי (constructor):

def __init__(self, input_value):

הארגומנט input_value מכיל מספר שלם או מחרוזת המכילה מספר הכתוב בספרות רומיות. הבנאי מאתחל שלוש תכונות (attributes):

int_value – שומרת את הערך המספרי בתור מספר שלם (int)
roman_value – שומרת את הייצוג של המספר בספרות רומיות (str)
is_neg – שומרת ערך True אם המספר הוא שלילי (קטן מ 0), ו False אחרת (bool).

שלד התרגיל מכיל מימוש של שתי מתודות: get_roman_from_int ו get_int_from_roman על מנת להמיר מספר שלם ליצוג בספרות רומיות, ולהיפך.

הניחו כי הקלט חוקי ויכול להכיל רק מספר שלם או מחרוזת ב upper-case המכילה יצוג תקין באותיות רומיות. בנוסף, הניחו כי הקלט אינו 0, שכן לא ניתן לייצג 0 בספרות רומיות.

2. כעת נרצה לממש את המתודות __str__ ו __repr__ על מנת לאפשר ייצוג "יפה" של אובייקטים מסוג Roman. הפורמט המבוקש הוא:

- The method of __repr__:
  >>>Roman('V')
  V
  >>>Roman(50)
  L
  >>>Roman('XL')
  XL
- The method of __str__
  >>>print(Roman('L'))

```
The integer value is 50 and the Roman Numeral is denoted by 'L'
>>>print(Roman(50))
The integer value is 50 and the Roman Numeral is denoted by 'L'
>>>print(Roman('XL'))
The integer value is 40 and the Roman Numeral is denoted by 'XL'
>>> print(Roman('-V'))
The integer value is -5 and the Roman Numeral is denoted by '-V'
>>> print(Roman(-5))
The integer value is -5 and the Roman Numeral is denoted by '-V'
```

שימו לב לייצוג של מספרים שליליים כפי שהוא מוצג בשתי הדוגמאות האחרונות.

3. נרצה לתמוך בפעולה שלילה (negation). לצורך כך, עליכם לממש את השירות __neg__.
דוגמא:

```
>>>-Roman(50)
-L
>>>-Roman(V)
-V
```

4. כעת נרצה לממש את המתודות המתאימות לפעולות של סכימה (המתודה __add__) והשוואה (המתודות __lt__ ו __gt__) של ספרות רומיות (עליכם לתמוך אך ורק בפעולות שבהן ה Roman מופיע מצד שמאל לאופרטור). על הפעולות לתמוך בסכימה\השוואה של Roman עם אובייקט מסוג Roman או int.

```
Method: __add__
>>> Roman(5) + Roman(6)
XI
>>> Roman('V') + 6
XI
>>> Roman(5) + Roman(-6)
-I
>>> Roman('V') + (-6)
-I
```

```
Method: __lt__ and __gt__
>>> Roman(5) < Roman(6)
True
>>> Roman(6) < Roman(5)
False
>>> Roman(5) > 6
False
>>> 5 < Roman(6)
True
```

שימו לב למקרה המיוחד שבו תוצאת החיבור נותנת 0. מכיוון שאין יצוג ל 0 בספרות רומיות, עליכם לזרוק ValueError עם מלל לבחירתכם אם תוצאת הסכימה היא 0.

5. מכיוון שאנחנו עובדים רק עם מספרים שלמים, נרצה לתמוך בפעולה של חלוקה ללא שארית (האופרטור // ולא האופרטור /). גם במקרה זה, נתמוך רק בפעולות שבהן Roman מופיע בצידו השמאלי של האופרטור. תוצאת החלוקה היא Roman.
מקרים מיוחדים:
   א. הניחו כי לא תתבצע חלוקה ב 0.
   ב. עליכם לזרוק ValueError עם מלל לבחירתכם אם תוצאת החלוקה ללא שארית היא 0.
פעולת החילוק ללא שארית תעבוד באופן הבא:

```
>>> Roman(6) // Roman(5)
I
```

```
>>> Roman('VI') // 5
I
>>> Roman(6) // Roman('-V')
-II
>>> Roman(6) // -5
-II
>>> 6//-5
-2
>> Roman(2) // Roman(3) # this should raise a ValueError
```

שאלה 2 (Numpy)

משרד הבריאות רצה לבדוק את ההישגים של תכנית אימונים חדשה. במשך כמה חודשים רצופים נאספו נתונים על המשתתפים. הנתונים נאספים בטבלאות csv, כך שבעמודה הראשונה מופיעים שמות המשתתפים, בשורה הראשונה שמות החודשים, וכל שורה הינה איסוף של הנתונים בסוף כל חודש. כלומר, בעמודה השנייה יופיעו הנתונים של כל מועמד בסוף החודש הראשון, בעמודה השנייה בסוף החודש השני וכן הלאה. (ראו את קובץ הדוגמא המצורף לתרגיל weight_input.csv, בו נמדדו משקליהם של ארבעה מתאמנים בקילוגרמים. חודש הדגימה הראשון הוא אוגוסט).

א. ממשו את הפונקציה load_training_data שמקבלת את הנתיב לקובץ ה-csv ומחזירה מילון 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.

דוגמת הרצה על הקובץ המצורף לתרגיל:

	August	September	October	November	December	January
Orit	84	81.3	82.8	80.1	77.4	75.2
Miki	79.6	75.2	75	74.3	72.8	71.4
Roni	67.5	66.5	65.3	65.9	65.6	64
Assaf	110.7	108.2	104.1	101	98.3	95.5

```
>>> data_dict = load_training_data("weight_input.csv")
>>> print(data_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(data_dict["column_names"])
['August' 'September' 'October' 'November' 'December' 'January']
>>> print(data_dict["row_names"])
['Orit' 'Miki' 'Roni' 'Assaf']
```

בסעיפים 2-5 להלן:

- **הפונקציות מקבלות את המילון שנוצר ע״י הפונקציה משאלה 1 על קובץ המשקלים: ובו הערכים המתאימים ל data, column_names, row_names.**
- **יש לממשן <u>ללא שימוש בלולאות</u>.**

2. ממשו את פונקציה get_highest_weight_loss_trainee שמחזירה את שם המשתתף שירידתו במשקל הייתה הכי גדולה מתחילת התכנית ועד סופה (ניתן להניח שיש מתאמן 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 על טבלת הקלט הנתונה:

```
>>> get_highest_weight_loss_trainee(data_dict)
'Assaf'
```

3. נגדיר את ה-'הפרש החודשי' כהפרש עבור משתתף מסויים בין חודש אחד לחודש הקודם לו. ממשו את הפונקציה get_diff_data שמחזירה את מטריצת ההפרשים. בעמודה ה i יופיע ההפרש בין החודש ה i+1 לבין החודש ה i, וסה״כ מספר העמודות במטריצת ההפרשים יהיה קטן ב 1 ממספר העמודות במטריצת הנתונים. <u>שימו לב שמטריצת הקלט לא משתנה</u>.

דוגמת הרצה על טבלת הקלט הנתונה:

```
>>> get_diff_data(data_dict)
array([[-2.7,  1.5, -2.7, -2.7, -2.2],
       [-4.4, -0.2, -0.7, -1.5, -1.4],
       [-1. , -1.2,  0.6, -0.3, -1.6],
       [-2.5, -4.1, -3.1, -2.7, -2.8]])
```

4. כתוב פונקציה get_highest_loss_month שמחזירה את שם החודש שבו סכום ההפרשים החודשיים על פני כל המועמדים הוא הגדול ביותר (כלומר, **שההורדה** במשקל הייתה מקסימלית). שימו לב שהפונקציה מקבלת את data, columns_names, row_names (ולא את מטריצת הפלט מסעיף 3). <u>השתמשו בפונקציה מסעיף 3</u>.

דוגמת הרצה: במטריצת הדוגמא, בחודש ספטמבר המתאמנים הורידו ביחד 10.6 ק"ג, סכום שיותר גדול מאשר בשאר החודשים.

```
>>> get_highest_loss_month(data_dict)
'September'
```

(ניתן להניח שיש חודש אחד כזה)

5. ידוע ששינוי במשקל הוא יחסי למשקל ההתחלתי. כתוב פונקציה get_relative_diff_table שתחזיר את טבלת השינוי במשקל, כך שלכל משתתף בכל חודש יופיע השינוי היחסי למשקלו בחודש הקודם, כלומר, ה'הפרש החודשי' לחלק למשקל בחודש הקודם. שימו לב שהפונקציה מקבלת את data, columns_names, row_names (ולא את מטריצת הפלט מסעיף 3). <u>השתמשו בפונקציה מסעיף 3</u>.

דוגמת הרצה:

```
>>> get_relative_diff_table(data_dict)
array([[-0.03214286,  0.01845018, -0.0326087 , -0.03370787, -0.02842377],
       [-0.05527638, -0.00265957, -0.00933333, -0.02018843, -0.01923077],
       [-0.01481481, -0.01804511,  0.00918836, -0.00455235, -0.02439024],
       [-0.02258356, -0.03789279, -0.02977906, -0.02673267, -0.02848423]])
```

למשל, הנתון המסומן בפלט לעיל, הוא תוצאת החישוב:

$$\frac{november-october}{october} = \frac{74.3-75}{75} = -0.0093333$$

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.

השתמשו בחבילה imageio בכדי לטעון את התמונות.

א. אחד הגדלים החשובים בתחום התקשורת היא האנטרופיה. האנטרופיה היא דרך לכמת 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. 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.
- 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*

דוגמת הרצה:

```
>>>print(compute_entropy('cameraman.tif'))
>>>7.009716283345514
```

ב. כאשר אנו מגדילים תמונה, אנחנו צריכים למלא בצורה כלשהי את הפיקסלים החדשים בעזרת הפיקסלים הישנים. אחת השיטות הבסיסיות לעשות זאת היא באמצעות עיקרון "השכן הקרוב". ערך הפיקסל בתמונה המוגדלת יקבל את ערך הפיקסל הקרוב ביותר לפיקסל הנ"ל בתמונה המקורית.

הנוסחא לפיקסל בתמונה המוגדלת תהיה:

$$Bigpixelvalue[i,j] = smallpixelvalue[floor(i * \frac{ySmallSize}{yBigSize}), floor(j * \frac{xSmallSize}{xBigSize}))]$$

כאשר xBigSize הוא הגודל בציר x (מספר עמודות) של התמונה המוגדלת וכן הלאה לגבי שאר הערכים.
j ו i הם האינדקסים של הפיקסלים בתמונה המוגדלת. Floor הינו עיגול כלפי מטה לערך השלם.

ממשו את הפונקציה nearest_enlarge(img,a) אשר מקבלת שני פרמטרים: שם של תמונה img (string) וערך הגדלה a גדול מ1 (int). הפונקציה מחזירה תמונה (מערך ndarray דו מימדי) מוגדלת כאשר מספר הפיקסלים בכל אחד ממימדי התמונה מוגדל פי a.

דוגמת הרצה:

```
>>>I=nearest_enlarge('cameraman.tif',2)
>>>plt.figure()
>>>plt.imshow(I,cmap = plt.cm.gray)
>>>plt.show()
```