תרגיל בית 9

Numpy (image processing) & Pandas

<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.
- שימו לב כי הרצת טסטים עבור בסעיפים/שאלות שלא מומשו עלולה לתת שגיאה

שאלה 1

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 2

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

דוגמאות:

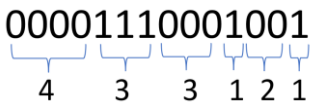
השורה הבאה:

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. ממשו 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. ממשו 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)
[[0. 1. 0.]
[0. 1. 0.]]
```

ג. בסעיף זה נממש את אלגוריתם RLE **בשינוי קל**.
ממשו 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.]]

>>> 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)
[1 1 2 1 1] (2, 3)
```

ד. ממשו 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 וקטור התמונה הדחוסה 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)
[[0. 1. 0.]
[0. 1. 0.]]
```

ה. ממשו 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

שאלה 3

ריק (Rick) מסתובב בכל רחבי היקומים (Universes) במצוד אחר מפלצות המטילות אימה על תושביהן. הוא בוחר את יעדו לפי חישוב קר של עלות מול תועלת.

במהלך השאלה, נעזר בשני קבצים המכילים את כלל היעדים והמפלצות אותם יש לחסל. הקבצים בפורמט CSV ומכילים את המידע הבא:

קובץ ראשון:

- בשורה הראשונה מופיעים שמות עמודות המידע בסדר הבא:
  - Universe - העמודה הראשונה; מכילה את שם היקום במצוקה.
  - Expenses - העמודה השלישית; מכילה את עלות המסע לאותו היקום ברכב חלל.
  - Duration - העמודה הרביעית; מכילה את מספר הימים שייקח להשלים את המסע (שימו לב שיכול להיות מספר לא שלם).
  - Teleporting – העמודה החמישית; מכילה אינדיקציה (ערך בוליאני) האם ניתן לעשות teleporting לאותו היקום (כלומר, להיות משוגר לשם באופן מהיר ללא צורך ברכב חלל).
- שאר השורות מכילות את המידע הרלוונטי בהתאם לעמודות כמצוין לעיל (ראו את טבלת הדוגמא למטה)
- לנוחיותכם הטבלה הבאה נמצאת כקובץ בשם "travels.csv" בפורמט csv בין קבצי התרגיל שקיבלתם:

Universe	Expenses	Duration	Teleporting
Gaia	250	2.5	FALSE
Flip Flop	500	3	TRUE
Tiny	150	7	TRUE
Buya	60	0.5	FALSE
Zool	250	5	TRUE

קובץ שני:

- בשורה הראשונה מופיעים שמות עמודות המידע בסדר הבא:
  - Monster - העמודה הראשונה; מכילה את שם המפלצת שיש לחסל.
  - Bounty - העמודה השנייה; מכילה את הגמול על חיסול המפלצת.
  - Universe - העמודה השלישית; מכילה את שם היקום במצוקה.
- שאר השורות מכילות את המידע הרלוונטי בהתאם לעמודות כמצוין לעיל (ראו את טבלת הדוגמא למטה)

לנוחיותכם הטבלה הבאה נמצאת כקובץ בשם "missions.csv" בפורמט csv בין קבצי התרגיל שקיבלתם:

Monster	Universe	Bounty
Creepy	Gaia	1000

Breepy	Flip Flop	150
Bripy	Flip Flop	150
Burpy	Tiny	500
Glupglap	Buya	250
Grumpy	Buya	500

בכל הסעיפים ניתן להניח כי הקלט תקין

א. ממשו את הפונקציה:
read _files(travels_file,missions_file)

- הפונקציה מקבלת את שמם של קבצי המידע (מטיפוס מחרוזת).
- הפונקציה תחזיר 2 אובייקטי dataframe (DF) של החבילה **pandas** לפי הסדר הבא:
  - אובייקט ראשון: טבלת המסעות בעלת 4 עמודות: Universe,Expenses,Duration,Teleporting.
  - אובייקט שני: טבלת המשימות; בעלת 3 עמודות: Monster,Bounty,Universe
- ניתן להניח כי **במידה** ושמות הקבצים קיימים, ערכיהם יהיו תקינים ומכילים לפחות שורת משימות אחת (ושורת כותרות לעמודות).
- רמז: ניתן (אך לא חייבים) להיעזר בפקודה pd.read_csv על מנת לטעון את הקובץ לטבלה של pandas. הסתכלו בתיעוד הפונקציה באינטרנט בכדי להעביר את הארגומנטים המתאימים להשלמת השאלה.

פלט קבצי הדוגמא:

```
=== A ===
Travels:
   Universe  Expenses  Duration  Teleporting
0      Gaia       250       2.5        False
1  Flip Flop      500       3.0         True
2      Tiny       150       7.0         True
3      Buya        60       0.5        False
4      Zool       250       5.0         True
Missions:
   Monster  Bounty   Universe
0   Creepy     280       Gaia
1   Breepy     150  Flip Flop
2    Bripy     150  Flip Flop
3    Burpy     500       Tiny
4 Glupglap      25       Buya
5   Grumpy      50       Buya
```

```
6    Grumpi    300      Zool
```

ב.

1. ממשו את הפונקציה:
merge_tables(df_travels,df_missions)

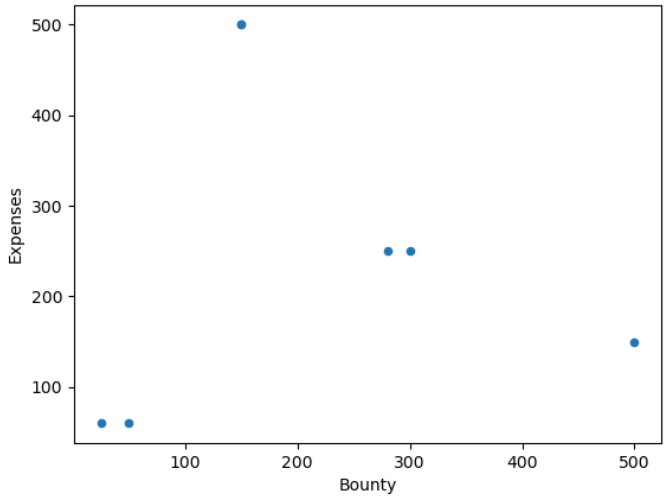
- הפונקציה מקבלת את פלט סעיף א' - שני אובייקטי DF (הראשון של המסעות והשני של המשימות)
- הפונקציה תמזג את 2 האובייקטים לפי עמודת שמות היקומים, ותחזיר את האובייקט הממוזג (מטיפוס DF).
- שימו לב כי בטבלה travels כל יקום מופיע פעם אחת בלבד. לעומת זאת, בטבלה missions יקום יכול להופיע מספר פעמים.
- הניחו כי לא קיים יקום המופיע בטבלה אחת בלבד.
- רמז: העזרו בפקודה merge
- **אין להשתמש בלולאות.**
- **יש לממש את הפונקציה בשורה אחת בלבד.**

פלט קבצי הדוגמא:

```
===  B  ===
   Universe  Expenses  Duration  Teleporting  Monster  Bounty
0     Gaia      250      2.5       False    Creepy    280
1  Flip Flop    500      3.0        True    Breepy    150
2  Flip Flop    500      3.0        True     Bripy    150
3     Tiny      150      7.0        True     Burpy    500
4     Buya       60      0.5       False  Glupglap     25
5     Buya       60      0.5       False   Grumpy      50
6     Zool      250      5.0        True   Grumpi     300
```

2. **ממשו את הפונקציה:**
scatter_plot(df_merged)

- בסעיף זה נצייר דיאגרמה מסוג scatter plot:
  - לקריאה כללית ראו: https://en.wikipedia.org/wiki/Scatter_plot
  - למימוש scatter plot באמצעות pandas ראו: https://pandas.pydata.org/pandas-docs/version/0.25.0/reference/api/pandas.DataFrame.plot.scatter.html
- הפונקציה מקבלת את פלט סעיף ב.1.
- הפונקציה תצייר scatter plot, **ותחזיר את הax** (אובייקט מסוג SubplotBase) עליו התבצע הציור.
- להלן תוצאת הציור המבוקשת:



ג. ממשו את הפונקציה:

add_gain_per_monster (df_merged)

- קלט הפונקציה הינו טבלת pandas הפלט מסעיף ב'.
- על הפונקציה להוסיף **לטבלה הקיימת** עמודה בשם Gain המכילה את הרווח מחיסול כל מפלצת. הרווח נמדד לפי תגמול (Bounty) בניקוי ההוצאות נסיעה (Expenses).
- **לדוגמא:** בהינתן מערך כפי שמופיעה בטבלה לעיל, כדאיות המשימה ביקום Gaia היא 750 מכיוון ש-$280 - 250 = 30$
- העמודה החדשה תיקרא "Gain"
- בשורות בהם לא ניתן להסיק את ערך gain עקב ערכים חסרים (בBountry או בExpenses) יש לשים **בעמודה Gain בלבד** את הערך 0.
- **אין להשתמש בלולאות.**
- **יש לממש את הפונקציה בשורה אחת בלבד.**
- **הפונקציה לא צריכה להחזיר כלום.**

ערך הטבלה לאחר העדכון (לפי קבצי הדוגמא):

```
===  C  ===
   Universe  Expenses  Duration  Teleporting  Monster  Bounty  Gain
0     Gaia      250       2.5       False    Creepy    280    30
1  Flip Flop    500       3.0        True    Breepy    150   -350
2  Flip Flop    500       3.0        True    Bripy     150   -350
3     Tiny      150       7.0        True    Burpy     500   350
4     Buya       60       0.5       False  Glupglap     25   -35
5     Buya       60       0.5       False   Grumpy      50   -10
6     Zool      250       5.0        True   Grumpi     300    50
```

על מנת ליעל את פעולותיו, החליט ריק כי לכל יקום בו ייסע, יהרוג את כל המפלצות באותה הנסיעה, וכך ישקיע זמן נסיעה (Duration) ישלם הוצאות נסיעה (Expenses) פעם אחת בלבד עבור חיסול כל המפלצות מאותו היקום.

ד. ממשו את הפונקציה:
daily_gain_per_universe(df_merged)

- הקלט לפונקציה הינו **הטבלה שהוחזרה בסעיף ב'.**
- על הפונקציה להחזיר עמודה מסוג **סדרה** (Series, טבלת DataFrame חד מימדית).
  - כותרת פלט הסדרה היא Universe.
  - האינקסים של העמודה הם שמות היקום
  - הערכים של העמודה הם הרווח הנקי היומי **עבור כל יקום**
  - טיפוס הערכים בסדרה הוא float64.
- הרווח היומי הנקי עבור כל יקום מחושב סך הרווחים מחיסול **כל מפלצות** באותו היקום (כפי שחישבתם בסעיף ג') חלקי סך הימים אשר יידרשו להשלמת **כלל המשימות** באותו היקום.
- **לדוגמא:** בהינתן הטבלאות בדוגמא לעיל, קיימת משימה אחת ביקום Gaia וכדאיותה היא 12 מכיוון ש-$\frac{280-250}{2.5} = 12$.
- **דוגמא נוספת:** בהינתן הטבלאות בדוגמא לעיל, קיימות 2 משימות ביקום Buya ולכן כדאיות יקום זה היא 30 מכיוון ש-$\frac{25+50-60}{0.5} = 30$.
- רמז: השתמשו בgroupby
- **אין להשתמש בלולאות.**
- **יש לממש את הפונקציה בשורה אחת בלבד.**

פלט מקבצי הדוגמא:

```
===  D  ===
Universe
Buya        30.000000
Flip Flop  -66.666667
Gaia        12.000000
Tiny        50.000000
Zool        10.000000
dtype: float64
```

ה. ממשו את הפונקציה:
drop_nonprofitable_universes (df_merged)

- קלט לפונקציה הינו הטבלה **שהחוזרה בסעיף ב'**.
- על הפונקציה להחזיר טבלה חדשה המכילה את כל היקומים עבורם סך הרווח יומי חיובי (כפי חושב בסעיף הקודם).
- לדוגמא, עבור ביקום Flip Flop קיימות 2 מפלצות, אך סך התגמול שלהן (300) קטן מהוצאות נסיעה ליקום זה (500), ולכן השורות המכילות את היקום Flip Flip יוסרו מהטבלה.
- במידה וקיימת יותר ממפלצת אחת באותו היקום, הוצאות המסע יחושבו פעם אחת בלבד.
- ניתן (ורצוי) להשתמש בפונקציות שמימשתם בסעיפים הקודמים.

```
===E===
 Universe  Expenses  Duration  Teleporting   Monster  Bounty
0     Gaia     250       2.5       False    Creepy     280
3     Tiny     150       7.0        True     Burpy     500
4     Buya      60       0.5       False  Glupglap      25
5     Buya      60       0.5       False   Grumpy      50
6     Zool     250       5.0        True   Grumpi     300
```

ו. ממשו את הפונקציה:
mean_duration(df_5)

- קלט לפונקציה הינו הטבלה **שהוחזרה בסעיף ה'**.
- את ממוצע זמן הנסיעות הנדרש עבור חיסול כלל המפלצות
- תזכורת: במידה וקיימת יותר ממפלצת אחת באותו היקום, הוצאות המסע יחושבו פעם אחת בלבד.
- **אין להשתמש בלולאות**
- **יש לכתוב את גוף הפונקציה בשורה אחת.**

```
===F===
3.75
```

ז. ממשו את הפונקציה:
drop_least_daily_lucrative_universe(df_5)

- קלט לפונקציה הינו סדרת הפלט של מסעיף ה'.
- על הפונקציה להחזיר 3 פלטים (מופרדים באמצעות פסיק) בסדר הבא:
  - את **שם** היקום (מסוג מחרוזת) בו הרווח היומי (שחושב סעיף הקודם) הכי פחות כדאי (הכי נמוך).
  - ערך הרווח היומי הכי פחות כדאי (מטיפוס float).
  - **עותק** של טבלת הקלט ללא השורה של היקום בעל הרווח הנמוך ביותר.
- ניתן להניח כי קיים יקום אחד בעל רווח יומי מינימלי.

```
===G===
('Zool', 10.0,  Universe  Expenses  Duration  Teleporting   Monster  Bounty
0     Gaia     250       2.5       False    Creepy     280
3     Tiny     150       7.0        True     Burpy     500
4     Buya      60       0.5       False  Glupglap      25
5     Buya      60       0.5       False   Grumpy      50 )
```

ח. ריק השיג טלפורטר (portal gun), וכעת הוא יכול לשגר את עצמו באמצעות teleporting לכל יקום אשר המכיל True בעמודה teleporting.
השימוש בteleporting מקצר את משך המסע (duration) בחצי. לדוגמא, מסע לTiny יתקצר מ7 ל3.5.

ממשו את הפונקציה **drop_least_daily_lucrative_universe_with_teleporting(df_merged)**

קלט הפונקציה הוא טבלת הפלט מסעיף ב'.
על הפונקציה לפתור שוב את סעיף ז' תוך הנחה שריק השתמש ב portal gun לכל יקום אליו יכול לעשות זאת.

הנחיות:

- תחילה, הפונקציה תעדכן את העמודה Duration לפי עבור כל שורה בה הערך Teleporting הוא True ולאחר מכן תבצע את שאר החישוב.
- ניתן (ואף רצוי) להשתמש בסעיפים הקודמים לפתרון סעיף זה.

```
===H===
('Gaia', 12.0,  Universe  Expenses  Duration  Teleporting  Monster  Bounty
3        Tiny     150      3.5       True     Burpy     500
4        Buya      60      0.5      False  Glupglap      25
5        Buya      60      0.5      False    Grumpy      50
6        Zool     250      2.5       True    Grumpi     300 )
```