תרגיל בית 3

פונקציות ומטריצות

הנחיות כלליות:

- קראו **היטב** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל השאלות יחד בקובץ ex3_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת.
- מועד אחרון להגשה: כמפורסם באתר.
- יש להקפיד על פלטים מדויקים על פי הדוגמאות.
- אין למחוק את ההערות שמופיעות בשלד.
- **אין לשנות את שמות הפונקציות והמשתנים שמופיעים בקובץ השלד של התרגיל.** עם זאת, אתם רשאים להוסיף משתנים ופונקציות נוספות כראות עינכם.
- **אין להשתמש בספריות חיצוניות (ובפונקציות שלהן).** כלומר, אין להשתמש בפקודת import. **כל פונקציה שלא דורשת פקודה זו מותרת לשימוש** (כלומר, זו פונקציה שהמתרגם (interpreter) מכיר ללא פקודה זו).

הנחיות מיוחדות לתרגיל זה:

- אופן ביצוע התרגיל: בתרגיל זה עליכם לכתוב את תוכן הפונקציות הנתונות ע"י החלפת המילה השמורה pass (המסמלת שורה שלא עושה כלום) במימוש שלכם. (אין חובה להוסיף קריאות שמפעילות את הפונקציות לצורך ההגשה אך ניתן להוסיף כאלו באזור המיועד בקובץ השלד – ראו סעיף הבא).
- בדיקה עצמית: הכנסנו לנוחיותכם את הדוגמאות כבדיקות בסוף קובץ השלד. כל בדיקה שעוברת בהצלחה מדפיסה True. הדוגמאות לא בהכרח מכסות את כל המקרים ולכן הריצו את הפונקציות שלכם על מגוון קלטים שונים כדי לוודא את נכונותן (וודאו כי הפלט נכון וכי התוכנית אינה קורסת). **אין למחוק את הטסטים.**
- ניתן להניח כי הקלט שמקבלות הפונקציות הינו כפי שהוגדר בכל שאלה ואין צורך להתייחס לקלט לא תקין, אלא אם כן נאמר אחרת.
- **בכל השאלות אין להדפיס (print) את הפלט אלא להחזיר (return) אותו!** (עם זאת, ייתכנו מקרים בהם אין דרישה להחזיר ערך מסוים).

שאלה 1:

ממשו את הפונקציה בשם mult_residuals_of_k(lst, k) המקבלת רשימה של מספרים (מטיפוס float או int) בשם lst ומספר חיובי (גדול מאפס) נוסף k (גם הוא יכול להיות מטיפוס float או int). הפונקציה **תחזיר** מספר מטיפוס float המייצג את מכפלת השאריות של המספרים ברשימה שאינם מתחלקים ב-k.

- במידה ואין מספר שאינו מתחלק ב-k (למשל, כאשר הרשימה ריקה) יוחזר 1.0.

דוגמאות הרצה:

```
>>> result = mult_residuals_of_k([3, 6, 4, 11, 9], 3)
>>> print(result)
2.0
```

הסבר: מבין כל המספרים ברשימה למספר 4 ישנה שארית 1 ולמספר 11 ישנה שארית 2. לשאר המספרים אין שארית. 2*1=2 ולכן התשובה היא 2.

```
>>> result = mult_residuals_of_k([45.5, 60, 74, 48], 4)
>>> print(result)
3.0
```

החישוב: 1.5*1

שאלה 2:

ממשו את הפונקציה בשם sum_even_digits(n) המקבלת מספר שלם (מטיפוס int) וחיובי (גדול מאפס) בשם n . הפונקציה **תחזיר** את סכום הספרות שלו שערכן זוגי.

- אם אין ספרות זוגיות במספר, יוחזר 0.

דוגמאות הרצה:

```
>>> result = sum_even_digits(5638)
>>> print(result)
14
```

הסבר: הספרות זוגיות ב-5638 הן 8 ו-6. והסכום שלהן הוא 14.

```
>>> result = sum_even_digits(137)
>>> print(result)
0
```

הסבר: כיוון שאין ספרות זוגיות ב-137 מוחזר 0 כברירת מחדל.

```
>>> result = sum_even_digits(54984127)
>>> print(result)
18

>>> result = sum_even_digits(6)
```

```
>>> print(result)
6
```

שאלה 3:

ממשו את הפונקציה בשם count_longest_repetition(s, c) המקבלת מחרוזת בשם s ומחרוזת נוספת בשם c המכילה אות בודדת באלפבית האנגלי. הפונקציה **תחזיר** את אורך הרצף הארוך ביותר ב-s שמכיל רק את התו הנתון ב-c **כאות גדולה או קטנה**.

- אם האות הנתון ב-c לא מופיע בתוך s יש להחזיר 0.

דוגמאות הרצה:

```
>>> s = 'eabbaAAacccaaddd'
>>> result = count_longest_repetition(s, 'a')
>>> print(result)
4
```

הסבר: יש שלושה רצפים של האות 'a' בתוך s. ברצף הראשון הוא מופיע פעם אחת (מסומן באדום), ברצף השני הוא מופיע 4 פעמים (מסומן בכחול) - **כאות קטנה וגדולה**, וברצף השלישי הוא מופיע פעמיים (מסומן בירוק). הרצף הארוך הוא 4 ולכן יוחזר 4. משיקולים דומים, count_longest_repetition(s, 'A') יחזיר תוצאה זהה.

```
>>> result =  count_longest_repetition ('CCCccc', 'c')
>>> print(result)
6

>>> result = count_longest_repetition ('ab!de', 'z')
>>> print(result)
0

>>> result = count_longest_repetition ('', 'z')
>>> print(result)
0
```

שאלה 4:

ממשו את הפונקציה בשם lower_strings(lst) המקבלת פרמטר בודד בשם lst ופועלת באופן הבא:

- אם lst הוא לא רשימה, הפונקציה לא תשנה את הקלט **ותחזיר** את המספר (int) 1-
- אחרת, הפונקציה מחליפה כל איבר ברשימה שהוא מחרוזת (string) במחרוזת מאותיות קטנות בלבד. **המחרוזת יכולה להכיל כל תו ASCII.**
  - **אין לשנות איברים שהם לא מחרוזת.**
  - הפונקציה **לא תחזיר** כלום אלא רק תשנה את הרשימה הנתונה כקלט (in-place)
    - חשבו, במידה והפונקציה לא מחזירה כלום, איזה ערך יוחזר כברירת מחדל?

רמז: ניתן להשתמש בפונקציה type כדי לקבל את הטיפוס של ערך מסוים ולבצע השוואה בין טיפוסים.

דוגמאות הרצה:

```
>>> vals = [11, 'Rick137', 3.14, 'moRTy’]
>>> result=lower_strings(vals)
>>> print(vals)
[11, 'rick137', 3.14, 'morty']
>>> print(result)
None
```

הסבר: רשימת הקלט מכילה שני איברים שהם מחרוזת ושני איברים מטיפוס מספרי (int ו-float).
הפונקציה החליפה את כל אחת משתי המחרוזות ברשימה במחרוזת חדשה שמכילה את אותם התווים, רק באותיות גדולות:

Rick137' → rick137'
moRTy' → 'morty'

פרט לכך, שני האיברים האחרים ברשימה, שהם לא מטיפוס מחרוזת, נותרו אותו הדבר.

מכיוון שלא החזרנו כלום בפונקציה, ערך ההחזר בברירת המחדל הוא None.

```
>>> vals = [-5, None, True, [1, Dont change me', 3]]
>>> lower_strings(vals)
>>> print(vals)
[-5, None, True, [1, 'Dont change me', 3]]
```

הסבר: במקרה הזה אף איבר ברשימה הוא לא מחרוזת ולכן אף איבר לא שונה (האיבר האחרון ברשימה הוא גם רשימה ולכן לפי הגדרת הפונקציה אין לשנותו, אפילו שהוא מכיל בתוכו מחרוזת).

```
>>> result = lower_strings(42)
>>> print(result)
-1

>>> result = lower_strings('im not a list')
>>> print(result)
-1

>>> result = lower_strings(False)
>>> print(result)
 -1
```

הסבר: במקרים לעיל הקלט אינו רשימה ולכן יוחזר -1.

מטריצות:

בשאלות הבאות נעבוד עם רשימות דו ממדיות של מספרים ממשיים לטובת ייצוג מטריצות. מטריצה שממדיה n על m (כלומר, יש לה n שורות שאורך כל אחת מהן m) תיוצג על ידי רשימה של n רשימות באורך m, שכל אחת מהן מייצגת שורה אחת.
לדוגמה, מטריצה בעלת 3 שורות ו-2 עמודות שערכיה הם:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

תיוצג על ידי הרשימה:

[[1, 2], [3, 4], [5, 6]]

דוגמא נוספת, מטריצה בעלת שורה אחת שערכיה הם:

$$\begin{pmatrix} 5 & 8 & 3 \end{pmatrix}$$

תיוצג על ידי הרשימה:

[[5, 8, 3]]

שאלה 5:

ממשו את הפונקציה בשם mult_mat_by_scalar(mat, alpha) המקבלת מטריצה תקנית mat ומספר alpha (מטיפוס int). הפונקציה **תחזיר** מטריצה חדשה בעלת ממדים זהים לאלו של mat כאשר כל אחד מאברי המטריצה המוחזרת היא מכפלת האיבר המתאים ב-mat ב-alpha.

- **יש להחזיר מטריצה חדשה, מבלי לשנות את mat.**
- ניתן להניח שמטריצת הקלט לא ריקה.

דוגמאות הרצה:

```
>>> mat1 = [[2, 5], [6, 9]]
>>> mat2 = mult_mat_by_scalar(mat1, 2)
>>> print(mat1)
[[2, 5], [6, 9]]
>>> print(mat2)
[[4, 10], [12, 18]]
>>> mult_mat_by_scalar([[10,15], [-3,6]], -5)
[[-50, -75], [15, -30]]
```

שאלה 6:

ממשו את הפונקציה בשם mat_transpose(mat) המקבלת מטריצה תקנית mat ומחזירה מטריצה חדשה שהיא המטריצה המשוחלפת של mat. שחלוף מטריצה היא פעולת ההחלפה בין השורות והעמודות של

מטריצה נתונה. הפעולה מקבלת מטריצה בת n שורות ו-m עמודות, ומחזירה מטריצה בת m שורות ו-n עמודות, שבמקום ה-(i, j) שלה נמצא האיבר ה-(j, i) של המטריצה המקורית.

הנה דוגמה למטריצה והמטריצה המשוחלפת שלה:

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}^T \rightarrow \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

- **יש <u>להחזיר</u> מטריצה חדשה, מבלי לשנות את mat.**
- ניתן להניח שמטריצת הקלט לא ריקה.

<u>דוגמאות הרצה:</u>

```
>>> mat = [[1,2],[3,4],[5,6]]
>>> mat_T = mat_transpose(mat)
>>> print(mat)
[[1, 2], [3, 4], [5, 6]]
>>> print(mat_T)
[[1, 3, 5], [2, 4, 6]]

>>> mat2 = [[0, 1, 2], [10, 11, 12], [20, 21, 22]]
>>> mat2_T = mat_transpose(mat2)
>>> print(mat2_T)
[[0, 10, 20], [1, 11, 21], [2, 12, 22]]
```