תרגיל בית 3

הנחיות כלליות:

- קראו בעיון את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל הפתרונות לשאלות יחד בקובץ ex3_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז. שלכם, כל 9 הספרות כולל ספרת ביקורת.
- מועד אחרון להגשה: כמפורסם באתר.
- בדיקה עצמית: 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).
- הנחות: ניתן להניח את תקינות הקלט במפורט בשאלה אלא אם נאמר אחרת. ניתן להניח שכל הרשימות והמחרוזות בתרגיל אינן ריקות אלא אם צוין במפורש שיש לטפל במקרה זה.
- תזכורת: החזרת ערך (ע"י פונקציה) שונה מהדפסה, כלומר return ≠ print.
- אין לשנות את שמות הפונקציות שכבר מופיעים בקובץ השלד של התרגיל.
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- אופן ביצוע התרגיל: שימו לב, בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף. מימוש ריק של כל הפונקציות מופיע בקובץ התרגיל. עליכם להחליף את השורה של pass בכל אחת מהפונקציות בתרגיל במימוש שלכם של השאלה.
- **אין למחוק את ההערות שמופיעות בשלד.**

שאלה 1

בשאלה זו שני סעיפים. המספרים בשאלה זו הינם שלמים חיוביים ואינם מכילים את הספרה 0.

1. ממשו את הפונקציה reverse_num(num) המקבלת מספר (שלם חיובי ללא 0) בשם num ומחזירה את המספר ההפוך, כלומר מספר חדש שבו הספרות בסדר ההפוך.

דוגמאות,

```
>>> reverse_num(1234)
4321
>>> reverse_num(89823567)
76532898
```

2. מספר יקרא *פלינדרום* אם ניתן לקרוא אותו באותו אופן מימין לשמאל או משמאל לימין. ממשו את הפונקציה is_palindrom_num(num) המקבלת מספר (שלם חיובי ללא 0) בשם num ומחזירה True אם הוא פלינדרום ו-False אחרת.

דוגמאות,

```
>>> is_palindrom_num(1234)
False
>>> is_palindrom_num(2334332)
True
>>> is_palindrom_num(1221)
True
>>> is_palindrom_num(2634332)
False
```

רמז: ניתן להשתמש בפונקציה מסעיף 1.

שאלה 2

ממשו את הפונקציה בשם mult_even_digits(n), המקבלת מספר שלם חיובי ממש בשם n, ומחזירה את מכפלת הספרות שלו שערכן זוגי. אם אין ספרות זוגיות במספר יוחזר 1. דוגמאות הרצה:

```
>>> mult_even_digits(1223)
4
>>> mult_even_digits(33)
1
>>> mult_even_digits(324769)
48
```

הסבר: בדוגמא הראשונה 2*2=4, בשניה אין ספרות זוגיות לכן מוחזר 1, בשלישית 2*4*6=48.

שאלה 3

כתבו פונקציה בשם fix_grades(grades) המקבלת רשימה (לא ריקה) של ציונים (מספרים מסוג int בין 0 ל-100) בשם grades, ומבצעת את שתי הפעולות הבאות:

a. **משנה** את הרשימה grades (אין ליצור רשימה חדשה – יש לשנות את ערכי הרשימה הקיימת) כמפורט: ציונים בין 0 ל-50 יהפכו ל-0, ציונים בין 51 ל-60 יהפכו ל-60 וציונים מעל 60 ישארו ללא שינוי.
b. **מחזירה** את מספר הציונים העוברים לאחר השינוי שתואר ב-a, כאשר ציון עובר הוא 60 ומעלה.

דוגמת הרצה:

```
>>> grades = [19, 90, 11, 51, 50]
>>> fix_grades(grades)
2
>>> print grades
[0, 90, 0, 60, 0]
```

שאלה 4

שתי רשימות מספרים ללא חזרות יקראו *פרמוטציה* אחת של השניה אם הן מכילות בדיוק את אותם איברים.
ממשו את הפונקציה is_permutation(lst1,lst2) המקבלת שתי רשימות של מספרים ללא חזרות lst1 ו- lst2. אם שתי הרשימות אינן פרמוטציה אחת של השניה, יש להחזיר False. אם הן כן פרמוטציה אחת של השניה, יש להחזיר רשימה של מיקומי האיברים של רשימה lst2 ברשימה lst1.
ניתן להניח שהרשימות לא ריקות, מכילות מספרים בלבד ואינן כוללות חזרות (כלומר, כל מספר מופיע רק פעם אחת ברשימה).

דוגמאות:

```
>>> is_permutation([4,2,8,3],[2,4,3,8])
[1, 0, 3, 2]
>>> is_permutation([4,2,8,3,5],[2,4,1,3,8])
False
>>> is_permutation([2,4,3],[2,4,3])
[0, 1, 2]
```

הסבר: בדוגמא הראשונה, שתי הרשימות זהות עד כדי סדר האיברים. האיבר הראשון ברשימה השניה נמצא במקום 1 ברשימה הראשונה, האיבר השני ברשימה השניה נמצא במקום 0 ברשימה הראשונה, האיבר השלישי ברשימה השניה נמצא במקום 3 ברשימה הראשונה והאיבר האחרון ברשימה השניה נמצא במקום 2 ברשימה הראשונה. בדוגמא השניה הרשימות אין פרמוטציה אחת של השניה שכן יש 5 בראשונה ולא בשניה כמו גם 3 בשניה אבל לא בראשונה. בדוגמא השלישית הרשימות זהות.

מטריצות

בשאלות הבאות נעבוד עם מטריצות של מספרים שלמים. המטריצות ייוצגו בצורה הבאה: מטריצה שמימדיה m על n (כלומר, יש לה m שורות שאורך כל אחת מהן n) תיוצג על ידי רשימה של רשימות, שכל אחת מהן מייצגת שורה אחת
כך לדוגמא, המטריצה שבעלת 3 שורות ו-2 עמודות שערכיה הם

1 2
3 4
5 6

תיוצג על ידי הרשימה

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

ניתן להניח שהמטריצות בשאלות הבאות אינן ריקות.

שאלה 5

ממשו את הפונקציה add_matrices(mat1,mat2) שמקבלת שתי מטריצות mat1 ו-mat2 המיוצגות בעזרת רשימה של רשימות. הפונקציה תחזיר מטריצה (רשימה של רשימות) שמייצגת את חיבור שתי המטריצות (חיבור איבר איבר), כלומר, במקום ה- i,j במטריצת הפלט יהיה סכום האיברים i,j במטריצות mat1,mat2. ניתן להניח שהמטריצות mat1,mat2 באותם מימדים.

לדוגמא:

```
>>> add_matrices([[1,2,3],[4,5,6]],[[11,12,13],[14,15,16]])
[[12, 14, 16], [18, 20, 22]]
>>> add_matrices([[1,-2],[-4,5]],[[3,2],[5,-4]])
[[4, 0], [1, 1]]
```

<u>שאלה 6</u>

כתבו פונקציה sum_matrices(mat_lst) שמקבלת רשימה (לא ריקה) של מטריצות (כל מטריצה היא בעצמה רשימה של רשימות). על הפונקציה להחזיר מטריצה שהיא סכום של כל המטריצות ברשימה. כלומר, כל איבר במטריצת הפלט הוא סכום האיברים המתאימים של כל המטריצות ברשימה. ניתן להניח שכל המטריצות ברשימה הן באותם מימדים.

לדוגמא,

```
>>> sum_matrices([[[1,-2],[-4,5]],[[3,2],[5,-4]]])
[[4, 0], [1, 1]]
>>>
sum_matrices([[[1,2,3],[4,5,6]],[[11,12,13],[14,15,16]],[[21,22,23],[24,25,26]],[[31,3
2,13],[34,35,36]]])
[[64, 68, 52], [76, 80, 84]]
```

בדוגמא הראשונה הרשימה מכילה שתי מטריצות כמו בדוגמא משאלה 5 ולכן סכומן זהה. הדוגמא השניה מכילה רשימה של ארבע מטריצות.

רמז: ניתן להשתמש בפונקציה משאלה 5

בהצלחה!