תרגיל 4 חלק א: רקורסיות בסיסי

תאריך פרסום:	11/11
תאריך הגשה:	18/11 בשעה 23:59
מתרגל אחראי:	אוריאל פרידמן
משקל תרגיל:	נקודה אחת
מטרות העבודה:	הכרות עם רקורסיות בסיסיות, המרת לולאות לרקורסיות, עבודה עם פונקציות מעטפת.

הנחיות כלליות

קראו בעיון את ההנחיות והעבודה. לא יתקבלו ערעורים על טעויות שחרגו מההנחיות.

1. העבודה תבוצע ותוגש ביחידים.
2. מומלץ לקרוא את העבודה כולה לפני שאתם ניגשים לפתרון.
3. עליכם להוריד את קבצי הקוד שמסופק עם התרגיל ולהשלים את הקוד החסר. יש לממש את הפתרון לתרגיל אך ורק באזורים שהוגדרו לשם כך בקובץ!
4. כתיבת קוד קריא:
   - 4.1. השתמשו בשמות משתנים משמעותיים. שימוש בשמות לא משמעותיים עשוי לגרור לפגיעה בציון.
   - 4.2. כתבו תיעוד (הערות) שמסביר את הקוד שלכם. יש לכתוב תיעוד docstring (הערות) בכל פונקציה כפי שנלמד בכיתה.
   - 4.3. אין לכתוב הערות בעברית! עבודה שתכיל טקסט בשפה שאינה אנגלית (או פייתון) תקבל ציון אפס ללא אפשרות ערעור.
5. אין להשתמש בחבילות או במודולים חיצוניים מלבד מה שהוגדר בתרגיל! אם יש ספק ניתן לשאול בפורום המתאים (ראו סעיף 10).
6. יש לכתוב קוד אך ורק באזורים שהוגדרו לשם כך!
7. הנחות על הקלט:
   - 7.1. בכל שאלה יוגדר מה הקלט שהקוד מקבל וניתן להניח כי הקלט שנבדוק מקיים את התנאים הללו. אין להניח הנחות נוספות על הקלט מעבר למה שהוגדר.
   - 7.2. בכל שאלה סיפקנו עבורכם דוגמאות לקלט והפלט הרצוי עבורו. עליכם לערוך בדיקות נוספות לקוד שמימשתם ולא להסתמך על דוגמאות אלו בלבד.
8. בדיקת העבודה:
   - 8.1. העבודה תיבדק באופן אוטומטי ולכן על הפלטים להיות זהים לפלטים שמוגדר בתרגיל.
   - 8.2. טרם ההגשה יש לעבור על המסמך assignments checklist שנמצא במודל.
   - 8.3. מערכת הבדיקות קוראת לפונקציות שהוגדרו בתרגיל בצורה אוטומטית. אין לשנות את חתימות הפונקציות. חריגה מההנחיות תגרור ציון אפס.
9. העתקות:
   - **9.1. אל תעתיקו!**
   - 9.2. העתקת קוד (משנים קודמות, מחברים או מהאינטרנט) אסורה בהחלט. בפרט אין להעביר קוד בין סטודנטים. צוות הקורס ישתמש בכלים אוטומטיים וידניים כדי לזהות העתקות. תלמיד שייתפס בהעתקה יועמד בפני ועדת משמעת (העונש המינימלי לפי תקנון האוניברסיטה הוא כישלון בקורס).
   - 9.3. אנא קראו בעיון את המסמך שהכנו בנושא: https://moodle2.bgu.ac.il/moodle/mod/resource/view.php?id=1922556

10. <u>שאלות על העבודה</u>:
    - 10.1 שאלות בנוגע לעבודה ישאלו בפורום שאלות לתרגיל במודל או בשעות הקבלה של המתרגל האחראי **<u>בלבד</u>** .
    - 10.2 אין לפנות במייל לבודקת התרגילים או למתרגלים אחרים בנוגע לעבודות הגשה. מיילים בנושאים אלו לא יקבלו מענה.
    - **10.3 לפני ששואלים שאלה בפורום יש לוודא שהשאלה לא נשאלה קודם!** שאלה שחוזרת על שאלה קיימת לא תענה.
    - 10.4 אנו מעודדים סטודנטים לענות על שאלות של סטודנטים אחרים. המתרגל האחראי יאשר שתשובה כזו נכונה.
11. <u>הגשת העבודה</u>:
    - 11.1 עליכם להוריד את הקבצים מתיקיית "תרגיל בית 4 א" מהמודל. התיקייה תכיל תיקייה נוספת ובה קבצי העבודה וקובץ הוראות. עליכם למלא את הפתרון במקום המתאים ובהתאם להוראות התרגיל.
    - 11.2 שימו לב: בנוסף לקבצי העבודה מצורף קובץ בשם get_id.py. עליכם למלא במקום המתאים בקובץ את תעודת הזהות שלכם. הגשה שלא תכיל את הקובץ הנ"ל עם תעודת הזהות הנכונה לא תיבדק ותקבל ציון אפס!
    - 11.3 את העבודה יש להגיש באמצעות תיבת ההגשה הייעודית במודל.
    - 11.4 פורמט ההגשה הוא להלן:
        - 11.4.1 את התיקייה בשם hw4a שהורדתם מתיקיית העבודה במודל, ובה נמצאים קבצים הקוד שלכם, יש לכווץ לפורמט zip (קבצים אחרים, כגון קבצי rar, יקבלו ציון 0).
        - 11.4.2 השם של התיקייה המכווצת יהיה תעודת הזהות שלכם.
        - 11.4.3 העלו את התיקייה המכווצת לתיבת ההגשה של העבודה.

הנחיות ספציפיות לעבודה:

עבודה זו היא עבודה ראשונה מתוך זוג עבודות ברקורסיה המתמקדת בנושאים של המרת לולאה לרקורסיה ופונקציות מעטפת.
חלק גדול מן הבעיות ניתן לפתור בקלות בלולאה ואת חלקן אף פתרנו כך בכיתה אך אנו דורשים פתרונות רקורסיביים.

אלא אם נאמר אחרת:

- ניתן לממש פונקציות מעטפת בכל מקום בו לא נכתב בפירוש אחרת.
- בכל מקום בו אתם כותבים פונקציית מעטפת שמה יהיה השם שתואר בסעיף – פונקציית העבודה יכולה להיות בכל שם (משמעותי) אחר שבחרתם לנכון.
- ניתן לכתוב פונקציות עזר (ללא לולאות) אלא אם נאמר בפירוש אחרת.
- אין להשתמש בפרמטרי ברירת מחדל בהגדרה של פונקציה.
- אין להשתמש בלולאות (אתם מתרגלים רקורסיה).
- שימו לב כי גם פקודות על אוספים כגון in, max, min או sum מהוות לולאות ולא ניתן להשתמש בהן בתרגיל זה.

שאלה 1

כפי שראינו בשיעור, פלינדרום הוא טקסט או מספר שקריאתו מימין לשמאל ומשמאל לימין היא זהה.

סעיף א'

עליכם לממש את הפונקציה:

`is_palindrom_no_helper(st)`

המקבלת **מחרוזת st** ומחזירה האם היא פלינדרום או לא.

קלט: הפונקציה מקבלת:

- **`st[str]` – מחרוזת.**

פלט: הפונקציה מחזירה ערך בוליאני True כאשר המחרוזת הינה פלינדרום ו-False אחרת.

הנחות קלט:

- St הוא מחרוזת, לא None.

דגשים והערות:

- **בסעיף זה בלבד אין לממש פונקציית מעטפת.**
- **בסעיף זה בלבד אין להשתמש בפונקציית / פונקציות עזר.**

דוגמאות:

1.
```
>> print(is_palindrom_no_helper('abba'))
True
```

2.
```
>> print(is_palindrom_no_helper('aba'))
True
```

3.
```
>> print(is_palindrom_no_helper('ima'))
False
```

4.
```
>> print(is_palindrom_no_helper('??!!@!!??'))
True
```

5.
```
>> print(is_palindrom_no_helper(' asdsa '))
True
```

6.
```
>> print(is_palindrom_no_helper('asdsa '))
False
```

סעיף ב'

בסעיף הקודם היה מימוש לא יעיל בשימוש בזיכרון, בסעיף זה נממש פתרון יעיל יותר בעזרת פונקציית מעטפת.
עליכם לממש את הפונקציה:

`is_palindrom_with_helper(st)`

המקבלת **מחרוזת st** ומחזירה האם היא פלינדרום או לא.

קלט: הפונקציה מקבלת:

- **`st[str]` – מחרוזת.**

פלט: הפונקציה מחזירה ערך בוליאני True כאשר המחרוזת הינה פלינדרום ו-False אחרת.

הנחות קלט:

- St הוא מחרוזת, לא None.

דגשים והערות:

- שימו לב: בסעיף זה אין להשתמש ב-slicing, מכיוון שזו פעולה שלמעשה תשכפל את המחרוזת.

דוגמאות:

1.

```
>> print(is_palindrom_with_helper(' asdsa '))
True
```

2.

```
>> print(is_palindrom_with_helper('asdsa '))
False
```

סעיף ג'

עליכם לממש את הפונקציה:

`is_palindrom_no_spaces(st)`

המקבלת **מחרוזת st** ומחזירה האם היא פלינדרום או לא – ללא התחשבות בתו רווח.
הכוונה בפלינדרום ללא התחשבות ברווח היא שאם לאחר צמצום רווחים מהמחרוזת מגיעים לפלינדרום גם אם המחרוזת אינה פלינדרום לפני צמצום רווחים – המחרוזת תיחשב לפלינדרום (ראו דוגמא שנייה).

<u>קלט:</u> הפונקציה מקבלת:

- **`st[str]` – מחרוזת.**

<u>פלט:</u> הפונקציה מחזירה ערך בוליאני True כאשר המחרוזת הינה פלינדרום ללא התחשבות ברווחים ו-False אחרת.

<u>הנחות קלט:</u>

- St הוא מחרוזת, לא None.

<u>דגשים והערות:</u>

- בדומה לסעיף הקודם, גם פה אין להשתמש ב-slicing.
- בסעיף זה אין להשתמש בפונקציות של מחרוזות למעט len.

<u>דוגמאות:</u>

1.

```
>> print(is_palindrom_no_spaces('??!!@!!??'))
True
```

2.

```
>> print(is_palindrom_no_spaces(' asdsa          '))
True
```

3.

```
>> print(is_palindrom_no_spaces('ima'))
False
```

שאלה 2

ממוצע משוקלל הוא ממוצע בו לכל ציון יש משקל והמשקל משפיע על הממוצע הסופי. בהינתן סדרה של ציונים x_i ומשקלים תואמים w_i הציון הממוצע המשוקלל יהיה:

עוד על ממוצע משקולל.

$$\bar{x} = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i}$$

ממשו את הפונקציה:

`weighted_average(ls)`

אשר מקבלת ציונים ומשקלים ומחזירה את הממוצע המשוקלל של הציונים.

קלט:

- **`grades[list[tuple[int,float]]`** – רשימה מקוננת של ציונים, כל איבר ב-grades מכיל tuple של 2 אברים: ציון (אינדקס 0) ומשקל (אינדקס 1).
  לדוגמא הרשימה המקוננת [(89,4.5),(95,5)] כוללת ציון 89 במשקל 4.5 וציון 95 במשקל 5.

פלט: הפונקציה מחזירה מספר ממשי שערכו הוא הממוצע המשוקלל, עם עד 2 ספרות אחרי הנקודה, מעוגל (ניתן להשתמש בפונקציית round).

הנחות קלט:

- grades - רשימה מקוננת, כל אבר בתוכה הוא tuple המכיל ציון תקין.
- grades מכילה לפחות איבר אחד.
- grades אינה None.

דוגמאות:

1.

```
>> print(weighted_average([(89,4.5),(95,5)]))
92.16
```

2.

```
>> print(weighted_average([(89,4.5)]))
89.0
```

שאלה 3

מספר ראשוני הוא מספר שלם וחיובי המתחלק ללא שארית בעצמו ובאחד בלבד, 1 נחשב למספר שאינו ראשוני (מספר מיוחד).

ממשו את הפונקציה:

`is_prime(num)`

המקבלת מספר שלם חיובי num ומחזירה האם הוא ראשוני או לא.

קלט:

- **num [int]** – מספר שלם וחיובי גדול מ 0.

פלט: הפונקציה מחזירה True אם num ראשוני ומחזירה False אחרת.

הנחות קלט:

- **num** מספר שלם חיובי – גדול ממש מ 0.

דוגמאות:

1.
```
>> print(is_prime(4))
False
```

2.
```
>> print(is_prime(13))
True
```

3.
```
>> print(is_prime(169))
False
```

4.
```
>> print(is_prime(23))
True
```

שאלה 4

מספר משוכלל הוא מספר שלם וחיובי השווה לסכום כל מחלקיו מלבד המספר עצמו. לדוגמא 6 הוא מספר משוכלל מכיוון שסכום כל המחלקים (מלבד המספר עצמו) הוא 6: 1+2+3=6 , ניתן לקרוא כאן עוד על מספרים מושלמים.

ממשו את הפונקציה:

`is_perfect(num)`

המקבלת מספר שלם חיובי num ומחזירה האם הוא מושלם או לא.

קלט:

- **num [int]** – מספר שלם חיובי.

פלט: הפונקציה מחזירה True אם num מספר מושלם ו- False אחרת.

הנחות קלט:

- **num** שלם חיובי גדול ממש מ - 0.

דגשים והערות:

- שימוש בכל המספרים המושלמים שנמצאו עד היום ובדיקה האם הקלט הוא כזה – תפסול את הסעיף.

דוגמאות:

1.
```
>> print(is_perfect(8))
False
```

2.
```
>> print(is_perfect(6))
True
```

3.
```
>> print(is_perfect(169))
False
```

4.
```
>> print(is_perfect(28))
True
```

שאלה 5

מספר "7 בום" יוגדר כמספר חיובי ושלם המתחלק ב 7 ללא שארית או שלפחות אחת מספרותיו היא 7.

ממשו את הפונקציה:

`is_7_boom(num)`

המקבלת מספר שלם חיובי num ומחזירה האם הוא "7 בום" או לא.

קלט:

- **num [int]** – מספר שלם וחיובי.

פלט: הפונקציה מחזירה True אם num הוא מספר "7 בום" ו False אחרת.

הנחות קלט:

- **num** שלם חיובי גדול ממש מ 0.

דוגמאות:

1.
```
>> print(is_7_boom(8))
False
```

2.
```
>> print(is_7_boom(14))
True
```

3.
```
>> print(is_7_boom(143))
False
```

4.
```
>> print(is_7_boom(47741))
True
```

5.
```
>> print(is_7_boom(70))
True
```

בהצלחה !