# תרגיל בית מספר 2 - להגשה עד 23 בנובמבר בשעה 23:55

**קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.**

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- השתמשו בקובץ השלד skeleton2.py כבסיס לקובץ ה py אותו אתם מגישים.

**לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.**

- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם 012345678.pdf ו- 012345678.py.

- הקפידו לענות על כל מה שנשאלתם.

- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.
להנחיה זו מטרה כפולה:
  1. על מנת שנוכל לבדוק את התרגילים שלכם בזמן סביר.
  2. כדי להרגיל אתכם להבעת טיעונים באופן מתומצת ויעיל, ללא פרטים חסרים מצד אחד אך ללא עודף בלתי הכרחי מצד שני. זוהי פרקטיקה חשובה במדעי המחשב.

## **שאלה 1**

אלגוריתם שמבצע את פעולתו **בַּמָּקום** (in-place algorithm) הוא אלגוריתם שמעתיק לכל היותר כמות קבועה מהקלט שלו למשתני עזר (ב"כמות קבועה" הכוונה לכמות שאינה תלויה בגודל הקלט).
בפרט, אלגוריתם כזה לא יכול ליצור העתק של הקלט שלו (מדוע?), אלא משנה אותו במידת הצורך.

ראשית, נבחן בהקשר זה את פעולת היפוך סדר באיברים של רשימה נתונה בשם lst.

א. האם הפקודה lst = lst[ : :-1] הופכת את סדר האיברים **במקום**? הסבירו בקצרה את תשובתכם.
ב. האם הפקודה lst.reverse() (המתודה reverse של המחלקה list) הופכת את סדר האיברים **במקום**? מה מחזירה הפונקציה? הסבירו בקצרה את תשובתכם.
ג. האם ניתן לכתוב פונקצית reverse **במקום** עבור מחרוזת? הסבירו.
ד. הוסיפו לקובץ השלד המצורף מימוש לפונקציה reverse_sublist(lst,start,end), אשר מקבלת רשימה ושני אינדקסים, והופכת **במקום** את סדר האיברים בחלק הרשימה שמתחיל באינדקס start ונגמר באינדקס end-1.
לשם הפשטות ניתן להניח כי 0<=start<end<=len(lst).
דוגמאות הרצה:

```
>>> lst = [1, 2, 3, 4, 5]
>>> reverse_sublist (lst,0,4)
>>> lst
[4, 3, 2, 1, 5]
>>> lst = ["a","b"]
>>> reverse_sublist (lst,0,1)
>>> lst
["a", "b"]
```

כעת נדון בפעולת ה k-סיבוב של רשימה.
בהנתן פרמטר ה k -סיבוב, פעולת ה k-סיבוב על רשימה באורך m מוגדרת באופן הבא:
הרשימה תסודר מחדש כך שאיבר שהופיע באינדקס i יועתק לאינדקס (i+k)mod m.
לדוגמה,
אם נבצע על הרשימה [1,2,3,4,5] 1-סיבוב נקבל [5,1,2,3,4]
אם נבצע על הרשימה [1,2,3,4,5] 2-סיבוב נקבל [4,5,1,2,3]
ואם נבצע על הרשימה [1,2,3,4,5] (1-)-סיבוב נקבל [2,3,4,5,1].
שימו לב שפעולת הסיבוב מעתיקה לתוך כל תא איבר יחיד. בנוסף, הפרמטר k חייב להיות שלם, אבל יכול להיות שלילי (כמו בדוגמה האחרונה).

ה. הוסיפו לקובץ השלד מימוש לפונקציה rotate1(lst), אשר מקבלת רשימה ומבצעת 1-סיבוב של אבריה **במקום**.
דוגמת הרצה:

```
>>> lst = [1, 2, 3, 4, 5]
>>> rotate1 (lst)
>>> lst
[5, 1, 2, 3, 4]
```

ו. הוסיפו לקובץ השלד מימוש לפונקציה rotatek_v1(lst,k), אשר מקבלת רשימה ומבצעת סיבוב-k של אבריה **במקום** באמצעות קריאה (אחת או יותר) לפונקציה rotate1(). ניתן להניח ש k שלם. על מספר הקריאות ל- rotate1 להיות מינימלי.
דוגמת הרצה:

```
>>> lst = [1, 2, 3, 4, 5]
>>> rotatek_v1 (lst,2)
>>> lst
[4, 5, 1, 2, 3]
```

ז. (סעיף בונוס, 5 נק') הוסיפו לקובץ השלד מימוש לפונקציה rotatek_v2(list,k), אשר מקבלת רשימה ומבצעת סיבוב-k של אבריה **במקום** באמצעות קריאה (אחת או יותר) לפונקציה reverse_sublist().
פלט לדוגמה: בדיוק כמו בסעיף הקודם, בשינוי שם הפונקציה.

הנחיות הגשה:

- בקובץ ה pdf הגישו:
  - את התשובות לסעיפים א',ב',ג'.
- בקובץ ה py הגישו:
  - את הפונקציות שמימשתם בסעיפים ד',ה',ו',ז'.

## שאלה 2

בשאלה זו ננתח את מספר פעולות הכפל שמתבצעות בחישוב $a^b$.

הפונקציה הבאה (שראינו בכיתה) מקבלת שני פרמטרים a,b ומחשבת את $a^b$.

```
def power(a,b):
    """ computes a**b using iterated squaring """
    result=1
    while b>0: # b is nonzero
        if b % 2 == 1: #  b is odd
            result = result*a   # (פעולת כפל אחת)
        a = a*a  # (פעולת כפל אחת)
        b = b//2
    return result
```

נרצה לספור כמה פעולות כפל מתבצעות ע"י הפונקציה power (השורות בהן מבוצעת פעולת כפל מסומנות לנוחיותכם בקוד שלעיל).

נתונים לנו שני מספרים עשרוניים: a שהייצוג הבינארי שלו מכיל n ביטים, ו- b שהייצוג הבינארי שלו מכיל m ביטים.

א. מהו המספר (המדויק) הקטן ביותר האפשרי של פעולות כפל שמתבצעות ע"י power, כפונקציה של n ו/או m?
מהו המספר (המדויק) הגדול ביותר האפשרי של פעולות כפל שמתבצעות ע"י power, כפונקציה של n ו/או m?

שימו לב שהתשובות בשני הסעיפים צריכות להיות כלליות (עבור כל n ו- m), ולא עבור מספרים ספציפיים. בכל אחד מהסעיפים הללו הסבירו את התשובה, וציינו מה צריך להיות המבנה של a ו/או b כדי שנקבל מספר של פעולות כפי שציינתם.

ב. השלימו בקובץ השלד את הפונקציה power_new שבהינתן a, b מחשבת את $a^b$ באותה סיבוכיות זמן כמו הפונקציה power שלמדנו. יש להשלים 3 שורות בלבד.

```
def power_new(a,b):
    """ computes a**b using iterated squaring """
    result = 1
    b_bin = bin(b)[2:]
    reverse_b_bin = b_bin[: :-1]
    for  bit in reverse_b_bin:
        _____________________________________________
        _____________________________________________
        _____________________________________________
    return result
```

## שאלה 3

בשאלה זו נעסוק בהיבטים שונים של ייצוג מספרים בבסיסים שונים.

א. כמה ספרות יש למספר $7^{1111}$ בייצוג עשרוני ?

ענו על השאלה בשתי דרכים שונות: (1) תוך שימוש בנוסחה שראיתם בשיעור (הרצאה 5, שקף 10), (2) באמצעות הפיכת המספר מ-integer ל-string, ומדידת אורך המחרוזת באמצעות ()len.

צרפו לקובץ ה pdf את התשובה, וכן שתי שורות קוד בודדות – אחת לכל אופן פתרון - שמבצעות את החישוב.

הדרכה: מומלץ להשתמש בפונקציה log(x,base) של המודול math, אשר מחשבת את הלוגריתם של x בבסיס base.

ב. ממשו את הפונקציה add_hex(A, B), אשר מקבלת שתי מחרוזות המייצגות מספרים בייצוג הֶקְסַדצימלי (כלומר בבסיס 16), ומחזירה את המחרוזת שמייצגת את תוצאת החיבור שלהם, גם היא בייצוג הקסדצימלי. ניתן להניח כי מחרוזות הקלט מייצגות מספרים חוקיים בבסיס 16, ולפיכך כוללות רק ספרות ואת האותיות a,b,c,d,e,f (שימו לב: האותיות ניתנות ב lower case).

דוגמת הרצה:

```
>>> add_hex("a5", "17")
'bc'
```

להלן המחשה של אלגוריתם החיבור A+B על מספרים בבסיס 16 (בדומה לחיבור מספרים עשרוניים עם נשא (carry)):

```
        1    (carried digits)
        1 f 5 (A)
+         5 a (B)
-------------
=       2 4 f
```

<u>הנחיה</u>:

1. הטיפול בספרות ההקסדצימליות (חלקן ספרות "רגילות" וחלקן אותיות) לא יתבצע באמצעות סדרה של משפטי תנאי, אלא באמצעות המרה מיידית, בדומה למה שראיתם בתרגול 3.

2. אין להמיר את המספרים A או B מבסיס לבסיס. בפרט, אין להשתמש כלל בפונקציה int של פייתון, בפונקציה convert_base שראינו בתרגול וכו'. יש לממש את האלגוריתם בהתאם להמחשה: ישירות באמצעות לולאות.

## שאלה 4

בהינתן טבעי n, כל טבעי k **<u>שקטן ממש</u>** מ-n ומחלק את n נקרא **מחלק ממש** של n.

**סדרת מחלקים** היא סדרת מספרים, שהראשון שבהם הוא מספר טבעי כלשהו, וכל איבר שווה לסכום המחלקים ממש של קודמו בסדרה. אם מגיעים ל- 0 – הסדרה מסתיימת (שכן ל- 0 אין מחלקים).

למשל, להלן סדרת מחלקים שמתחילה ב- 45:

$$45 \mapsto 33 \mapsto 15 \mapsto 9 \mapsto 4 \mapsto 3 \mapsto 1 \mapsto 0$$

סדרות מחלקים נחלקות ל- 3 סוגים:

1. סדרות שמסתיימות ב- 0 (סדרות סופיות).
2. סדרות שלא מגיעות ל- 0 לעולם (אינסופיות), שנכנסות למעגל מחזורי כלשהו.
3. סדרות שלא מגיעות ל- 0 לעולם (אינסופיות), שלא נכנסות למעגל מחזורי כלשהו (סדרות כאלו חייבות להיות בלתי חסומות).

ישנן סדרות שלא ידוע לאיזה סוג הן שייכות. למעשה לא ידוע אם בכלל יש סדרות מהסוג השלישי – זוהי שאלה פתוחה במתמטיקה.

א. הסבירו בקצרה מדוע יש אינסוף סדרות מהסוג הראשון (רמז: מספרים ראשוניים), ומדוע יש סדרות מהסוג השני (רמז: מספרים מושלמים).

ב. ידוע שכל המספרים בתחום בין 1 ל- 275 (כולל) מתחילים סדרה ששייכת לאחד משני הסוגים הראשונים הנ"ל (לגבי 276 אגב, לא ידוע כיום לאיזה סוג הוא שייך...). נרצה לדעת כמה מתוכם הם התחלה של סדרה סופית (מסוג 1).

לשם כך עליכם להשלים בקובץ השלד את הפונקציות הבאות:

- הפונקציה sum_divisors(*n*), אשר מקבלת מספר שלם חיובי *n*, ומחזירה את סכום המחלקים-ממש שלו.

דוגמת הרצה:

```
>>> sum_divisors(4)
3
>>> sum_divisors(220)
284
```

<u>הנחייה מחייבת</u>: בפונקציה לא תהיה יותר מלולאה אחת. עבור קלט *n*, על הלולאה שבפונקציה לבצע לא יותר מ- $\sqrt{n}$ איטרציות. פונקציות שמבצעות מספר איטרציות בסדר גודל גבוה יותר (למשל בערך $n/2$ איטרציות) אינן עומדות בתנאי זה.

- הפונקציה is_finite(n) שמחזירה True אם סדרה המחלקים שמתחילה במספר *n* היא סופית, כלומר שייכת לסוג 1 הנ"ל.

דוגמת הרצה:

```
>>> is_finite(4)
True
>>> is_finite(220)
False
```

<u>הנחיה מחייבת</u>: השתמשו ב- sum_divisors.

- הפונקציה cnt_finite(limit) שמחזירה כמה מספרים בין 1 ל- limit (כולל) הם התחלה של סדרה סופית.

  <u>הנחיה מחייבת</u>: הפונקציה תקרא ל- is_finite, וזו בתורה תקרא כאמור ל- sum_divisors.

בקובץ ה- pdf רישמו את התשובה עבור limit=275.

ג. התבוננו שוב בהגדרת הפלט של is_finite. אמיר הציע לדרוש גם שהפונקציה תחזיר False אם הסדרה איננה מסוג 1. מיכל התנגדה להוספת דרישה זו לתרגיל. מדוע?

## <u>שאלה 5</u>

בשאלה זו נבצע פעולת חישוב על רשימה של מספרים. כיוון שרשימת המספרים עשויה להיות ארוכה מאד, הכרחי לממש את הפעולה באופן יעיל.

נגדיר סכום מתחלף (alternating sum) של סדרת מספרים $a_0, a_1, \dots, a_{n-1}$ כתוצאת החישוב
$\sum_{i=0}^{n-1}(-1)^n a_i = a_0 - a_1 + a_2 - a_3 \dots$.

שימו לב: אנו מגדירים את סדר הספרות במספר משמאל לימין, ולכן בכל רצף עוקב הספרה השמאלית ביותר היא זו שתסומן כ $a_0$ ותפתח את הסכום בסימן חיובי.
לדוגמה, המספר 43805 מכיל את הסכומים המתחלפים באורך 3 הבאים:
4-3+8
3-8+0
8-0+5

עליכם למצוא במספר $3^{200,000}$ , מבין כל הסכומים המתחלפים של 50,000 (חמישים-אלף) ספרות סמוכות, את המקסימלי.

על הריצה להסתיים תוך לכל היותר דקה אחת על מחשב סביר (למשל המחשבים בחוות המחשבים בבניין שרייבר).

בקובץ השלד ממשו את הקוד בפונקציה altsum_digits(n, d), שמחזירה את הסכום המתחלף המקסימלי של d ספרות סמוכות במספר n.

דוגמת הרצה:

```
>>> altsum_digits(5**36, 12)
18
```

הדרכה:
נסו לחשוב בתחילה כיצד הייתם מחפשים באופן יעיל את הסכום הרגיל (לא מתחלף) של d ספרות עוקבות. בפרט, שימו לב שעל מנת שהתכנית תרוץ מספיק מהר, לא ניתן לממש את הפתרון הטריוויאלי, אשר סוכם בנפרד כל תת סדרה רציפה באורך d. חשבו איך בהינתן הסכום של d הספרות הראשונות ניתן לחשב ביעילות את הסכום של d הספרות אשר מתחילות בספרה השנייה בסדרה.

## <u>סוף.</u>