פייתון למהנדסים 2023, סמסטר א' תשפ"ג 0509-1820

# 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.

**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). יש לבצע את כל הבדיקות בתוך ה-if:
  if __name__ == '__main__':
- ניתן להניח כי הקלט שמקבלות הפונקציות הינו כפי שהוגדר בכל שאלה ו**אין 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).
- **אין להשתמש בספריה חיצונית כגון numpy לצורך פתרון התרגיל.**

# שאלה 1:

ממשו את הפונקציה בשם sum_divisible_by_k(lst, k) המקבלת רשימה של מספרים בשם lst ומספר חיובי (גדול מאפס) k, ומחזירה את הסכום של כל המספרים ברשימה המתחלקים ב-k.

- במידה ואין מספר המתחלק ב-k (למשל, כאשר הרשימה ריקה) יוחזר 0.

דוגמאות הרצה:

```
>>> result = sum_divisible_by_k([3, 6, 4, 10, 9], 3)
>>> print(result)
18
```

הסבר: מבין כל המספרים ברשימה המספרים 3, 6, 9 מתחלקים ב-3 וסכומם הוא 18.

```
>>> result = sum_divisible_by_k([45.5, 60, 73, 48], 4)
>>> print(result)
108
```

# שאלה 2:

ממשו את הפונקציה בשם mult_odd_digits(n) המקבלת מספר שלם חיובי (גדול מאפס) בשם n ומחזירה את מכפלת הספרות שלו שערכן אי-זוגי.

- אם אין ספרות אי-זוגיות במספר, יוחזר 1.

דוגמאות הרצה:

```
>>> result = mult_odd_digits(5638)
>>> print(result)
15
```

הסבר: הספרות האי-זוגיות ב-5638 הן 5 ו-3. המכפלה שלהן היא 15.

```
>>> result = mult_odd_digits(2048)
>>> print(result)
1
```

הסבר: כיוון שאין ספרות אי-זוגיות ב-2048 מוחזר 1 כברירת מחדל.

```
>>> result = mult_odd_digits(54984127)
>>> print(result)
315
```

# שאלה 3:

ממשו את הפונקציה בשם count_longest_repetition(s, c) המקבלת מחרוזת בשם s ומחרוזת נוספת בשם c המכילה תו בודד. הפונקציה מחזירה את אורך הרצף הארוך ביותר ב-s שמכיל רק את התו הנתון ב-c.

- אם התו 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 התו 'a' בתוך s. ברצף הראשון הוא מופיע פעם אחת (מסומן באדום), ברצף השני הוא מופיע 4 פעמים (מסומן בכחול), וברצף השלישי הוא מופיע פעמיים (מסומן בירוק). הרצף הארוך ביותר מכיל את '4 a' פעמים ולכן יוחזר 4.

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

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

## שאלה 4:

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

- אם lst הוא לא רשימה, יש להחזיר את המספר (1- int)
- אחרת, הפונקציה מחליפה כל איבר ברשימה שהוא מחרוזת (string) במחרוזת מאותיות גדולות בלבד.
  - **אין לשנות איברים שהם לא מחרוזת.**
  - הפונקציה לא מחזירה ערך אלא משנה את הרשימה שהתקבלה כקלט.

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

דוגמאות הרצה:

```
>>> vals = [11, 'TeSt', 3.14, 'cAsE']
>>> upper_strings(vals)
>>> print(vals)
[11, 'TEST', 3.14, 'CASE']
```

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:

'TeSt'  'TEST'
' cAsE '  'CASE'

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

```
>>> vals = [-5, None, True, [1, 'dont change me', 3]]
>>> upper_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 = upper_strings(42)
>>> print(result)
-1

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

>>> result = upper_strings(False)
>>> print(result)
 -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]]

## שאלה 5:

ממשו את הפונקציה בשם div_mat_by_scalar(mat, alpha) המקבלת מטריצה תקנית mat ומספר שלם alpha ומחזירה מטריצה חדשה בעלת ממדים זהים לאלו של mat כאשר כל אחד מאברי המטריצה המוחזרת היא חלוקת (בשלמים) האיבר המתאים ב-mat ב-alpha.

- **יש להחזיר מטריצה חדשה, מבלי לשנות את mat.**
- את החלוקה יש לבצע בעזרת חלוקת שלמים (//) ולא לעגל אחרי החישוב.
- ניתן להניח כי alpha שונה מ-0.
- ניתן להניח שמטריצת הקלט לא ריקה.

דוגמאות הרצה:

```
>>> mat1 = [[2, 4], [6, 8]]
>>> mat2 = div_mat_by_scalar(mat1, 2)
>>> print(mat1)
[[2, 4], [6, 8]]
>>> print(mat2)
[[1, 2], [3, 4]]
>>> div_mat_by_scalar([[10,15], [-3,6]], -5)
[[-2, -3], [0, -2]]
```

## שאלה 6:

ממשו את הפונקציה בשם mat_transpose(mat) המקבלת מטריצה <u>תקנית</u> 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:

$$(1\ 2\ 3\ 4\ \ 5\ 6\ )^T \rightarrow (1\ 3\ 5\ 2\ 4\ 6\ )$$

- **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]]
```