

# בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

קורס מבוא למדעי המחשב 67101

## תרגיל 3 – לולאות

להגשה עד יום רביעי, 20 לנובמבר 2019, בשעה 22:00.



# בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

## הקדמה

בתרגיל זה נתרגל שימוש בלולאות ומשתנים. אנא קראו את הבאים לפני תחילת עבודה :

- בתרגיל זה יש להגיש קובץ אחד בשם `ex3.zip`, ובתוכו נמצא הקובץ `ex3.py`.
- חתימות הפונקציות (שם הפונקציה והפרמטרים שלה) ב `ex3.py` צריכות להיות זהות במדויק לחתימות המתוארת במשימות (היזהרו מ `copy – paste` מקובץ זה, כתבו בעצמכם).
- לפני מימוש פונקציה, קראו את כל השאלה. מצורפות דוגמאות והערות למימוש.
- ניתן להוסיף פונקציות נוספות וניתן להשתמש בשאלות מאוחרות יותר בפונקציות שמומשו קודם.
- **סגנון**: הקפידו על תיעוד נאות ובחרו שמות משתנים משמעותיים. הקפידו להשתמש בקבועים (שמות משתנים באותיות גדולות), על פי ההסברים שנלמדו, ורק אם יש בכך צורך.
- **שימוש בכלים שלא נלמדו בקורס**: התרגיל ניתן למימוש קצר ויעיל מאוד, בעזרת כל מיני פונקציות מובנות וספריות מוכנות מראש (למשל Numpy). לכן **(!אסור!)** להשתמש בהן: בתרגיל זה נרצה לתרגל שימוש בלולאות, ואם נשתמש בפונקציות שמייטרות את השאלה אז לא נשיג את מבוקשנו.
- למשל את הפונקציה ב"מימוש מכפלה פנימית" ניתן לממש ב-2 שורות קוד (כולל החתימה) וללא לולאות. לכן אם אין בפתרון שלכם לולאה, נורה אדומה צריכה להידלק. במקרה של ספק, יש לשאול בפורום התרגיל.
- **אין להשתמש באף שאלה בכלי של `list comprehension`.**
- **מותר ורצוי להשתמש בפונקציה `append`.**
- **אסור למילה `"import"` להופיע באף קובץ `py`. בתרגיל זה (שימוש בה עלול לגרור ציון 0).**
- שימו לב מתי יש לקבל קלט מהמשתמש בעזרת הפונקציה `input` (השאלה הראשונה בלבד) ומתי הקלט מתקבל כארגומנט בעת קריאה לפונקציה.
- בכל שאלה מפורט מה ניתן להניח על הקלט - אין צורך לבצע בדיקות תקינות שלא במסגרת ההנחה.
- בכל הסעיפים, כאשר פונקציה צריכה להחזיר ערך, הכוונה היא לשימוש במילה השמורה `return`.
- **בפרט הכוונה אינה להדפיס למסך בעזרת הפונקציה `print`.** שימו לב, אין להדפיס למסך כל הודעה מלבד ההודעות הנדרשות במפורש.
- בפונקציות המקבלות רשימות כקלט – **אין לשנות** את רשימות הקלט עצמה.
- שימו לב כי אתם לא נדרשים לקרוא לפונקציות שכתבתם (מלבד במימוש פונקציות אחרות).
- בכל מטלה קראו את כל השאלה, כולל הערות למימוש, לפני כתיבת הפתרון (לפעמים ההערות נחתכות לעמוד הבא).

# בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

## חלק א' - קבלת קלט מהמשתמש

### 1. קבלת רשימה כקלט

#### הקדמה

הפונקציה `input` מקבלת שורת קלט מהמשתמש וממירה אותה למחרוזת אותה ניתן לשמור לתוך משתנה. לדוגמה, בעת הרצת קטע הקוד הבא התכנה תמתין לקלט מהמשתמש והקלט אותו יכניס המשתמש (למשל, המחרוזת "Hello") יישמר לתוך המשתנה `string_from_user`. במשתנה זה ניתן להשתמש, לדוגמה על מנת להדפיס את ערכו (למסך תודפס המחרוזת "Hello").

```
string_from_user = input()
print(string_from_user)
```

#### הגדרות:

- "רשימה ריקה" – רשימה שאינה מכילה איברים. היא עדיין נחשבת רשימה וגודלה הוא 0.
- "מחרוזת ריקה" – מחרוזת שאינה מכילה תווים, כלומר "" (מירכאות פותחות וסוגרות) או " (גרש פתיחה וסגירה). זוהי עדיין מחרוזת, ואורכה 0. מחרוזת המכילה לפחות רווח אחד אינה ריקה.

#### משימה:

- כתבו פונקציה אשר מקבלת מספרים רבים מהמשתמש, ומחזירה (return) רשימה בה כל הקלטים שהכניס המשתמש ובסופה סכומם. הפונקציה תקבל קלטים רבים מהמשתמש עד אשר יכניס מחרוזת ריקה.
- חתימת הפונקציה הינה: `def input_list():`

#### הערות למימוש:

- שימו לב - רווח יחיד או מספר רווחים הם שונים ממחרוזת ריקה. `" " ≠ ""`.
- הפונקציה תחזיר (return) רשימה המכילה את כל הקלטים שהמשתמש הכניס, כמספרים (ולא כמחרוזות), לפי הסדר. בסוף הרשימה, אחרי המספר האחרון שהזין המשתמש, יופיע סכום כל המספרים שהזין.

- 0 כל תא ברשימה יכיל קלט אחד בדיוק של המשתמש.
- 0 הקלט האחרון של המשתמש (המחרוזת הריקה) לא צריך להיות איבר ברשימה.
- 0 סידור הערכים ברשימה יהיה כך שהקלט הראשון ימצא בתא ה-0 ברשימה והקלט ה-1 ימצא בתא ה-1 ברשימה. במקום ה-n ברשימה ימצא סכום כל המספרים.
- במקרה והקלט הראשון הוא מחרוזת ריקה, תחזיר הפונקציה רשימה עם איבר יחיד 0.
- ניתן להניח שהמשתמש אכן יקליד רק מספרים שלמים.
- אין להשתמש בפונקציה `sum()`.
- מומלץ להשתמש בפונקציה `append`.

## בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

- הפונקציה `input` מחזירה מחרוזת, גם אם מקלידים מספר. אם המשתמש יקליד את המספר 371, אז הפונקציה תחזיר את המחרוזת "371" ולא את המספר 371. כדי לבצע המרה ממחרוזת למספר, השתמשו בפונקציה `int()`.
- יש לקרוא לפונקציה `input` ללא מחרוזת (כלומר אין להדפיס למסך הודעה לבקשת קלט).

דוגמה לשימוש בפונקציה (הטקסט באדום הוא הסבר לתרגיל ולא מופיע בקלט המשתמש)

עבור הקלט:

311

0

-3

2

*# Enter with no input*

הפונקציה תחזיר את הרשימה:

[311, 0, -3, 2, 310]

שימו לב, רשימה של מספרים!

חלק ב' – לולאות ורשימות

### 2. חישוב מכפלה פנימית

רקע: בהמשך, בקורס אלגברה לינארית, תלמדו על מרחבי מכפלה פנימית. באופן כללי, זהו מרחב וקטורי שעליו מוגדרת פעולה הנקראת מכפלה פנימית (או בקיצור מ"פ) – פעולה שעונה על הגדרה מסוימת. אנו נממש מ"פ חשובה ומעניינת, הנקראת המ"פ הסטנדרטית על  $\mathbb{R}^n$ . אנו מתייחסים אל וקטורים כאל רשימות.

הגדרה: בהינתן שתי רשימות של מספרים:  $[x_1, x_2, \dots, x_{n-1}, x_n], [y_1, y_2, \dots, y_{n-1}, y_n]$  המכפלה הפנימית הסטנדרטית שלהן מוגדרת להיות המספר הבא:  $x_1 \cdot y_1 + x_2 \cdot y_2 + \dots + x_n \cdot y_n$ .

משימה:

- כתבו פונקציה המקבלת שתי רשימות של מספרים, ומחזירה את המ"פ הסטנדרטית שלהן.
- חתימת הפונקציה: `def inner_product(vec_1, vec_2):`
- פלט הפונקציה: מספר.

## בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

### הערות למימוש :

- ניתן להניח שהקלט הוא אכן שתי רשימות.
- ניתן להניח שהרשימות מכילות מספרים מסוג `int` או `float` בלבד.
- אם הרשימות אינן באותו האורך, הפונקציה צריכה להחזיר את הערך **None**.
- אם הרשימות ריקות, הפונקציה צריכה להחזיר את הערך 0.

### 3. בדיקת מונוטוניות של סדרה סופית

#### הגדרה :

בהינתן סדרת מספרים  $(a_n)_{n=0}^N$  בעלת  $N + 1$  איברים  $(a_0, a_1, a_2, \dots, a_{N-1}, a_N)$ :

0. נאמר שהסדרה  $(a_n)_{n=0}^N$  **מונוטונית עולה** אם לכל  $1 \leq n \leq N$  מתקיים:  $a_{n-1} \leq a_n$ .
1. נאמר שהסדרה  $(a_n)_{n=0}^N$  **מונוטונית עולה ממש** אם לכל  $1 \leq n \leq N$  מתקיים:  $a_{n-1} < a_n$ .
2. נאמר שהסדרה  $(a_n)_{n=0}^N$  **מונוטונית יורדת** אם לכל  $1 \leq n \leq N$  מתקיים:  $a_{n-1} \geq a_n$ .
3. נאמר שהסדרה  $(a_n)_{n=0}^N$  **מונוטונית יורדת ממש** אם לכל  $1 \leq n \leq N$  מתקיים:  $a_{n-1} > a_n$ .

#### משימה :

- כתבו פונקציה המקבלת רשימה (של מספרים כלשהם מסוג `float` או `int`, כלומר סדרה  $(a_n)_{n=0}^N$  כלשהי) המחזירה כפלט האם הסדרה עונה לכל אחת מההגדרות הנ"ל.
- **def sequence\_monotonicity(sequence)** : חתימת הפונקציה:
- **פלט הפונקציה :** רשימה בעלת 4 איברים בולאנים : המקום ה-0 ברשימה מתייחס להגדרה 0, מקום 1 להגדרה 1, וכך הלאה. אם הסדרה עונה להגדרה, במיקום המתאים ברשימה יוחזר הערך `True`. אחרת, `False`.

### דוגמאות לקלט ופלט :

`sequence_monotonicity ([1,2,3,4,5,6,7,8]) → [True, True, False, False]`

`sequence_monotonicity ([1,2,2,3]) → [True, False, False, False]`

`sequence_monotonicity ([7.5, 4, 3.141, 0.111]) → [False, False, True, True]`

`sequence_monotonicity ([1, 0, -1, 1]) → [False, False, False, False]`

### הערות למימוש :

- ניתן להניח שהקלט לפונקציה הוא אכן רשימה, וברשימה אין איברים שאינם מסוג `int` או `float`.
- בהינתן רשימה ריקה או רשימה בעלת איבר יחיד, הפונקציה תחזיר `[True, True, True, True]`.
- אין להשתמש בפונקציות `all`, `any`.

## בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

### 4. דוגמאות לסדרות

רקע: דרך טובה להבנת הגדרה היא הסתכלות על דוגמאות. לכן כדי להבין בצורה אינטואיטיבית יותר אילו סדרות מקיימות אילו מארבעת ההגדרות הנ"ל, ניתן דוגמאות.

משימה:

- כתבו פונקציה המקבלת רשימה בת 4 איברים בולאניים (קרי *True* או *False*), ומחזירה רשימה בת 4 מספרים המייצגת סדרה סופית שהיא בעצם דוגמה לסדרה המקיימת את ההגדרות בהתאם להיכן שיש *True* בקלט.
- חתימת הפונקציה: `def monotonicity_inverse(def_bool)`
- פלט הפונקציה: רשימה בת 4 מספרים המייצגת סדרה סופית, כאשר הסדרה עונה להגדרה ה-*i* אם ורק אם ישנו *True* במקום ה-*i* בקלט (כלומר בהתאם ל 4 ההגדרות בסעיף הקודם. דוגמה בהמשך). אם אין רשימה כזאת, הפונקציה תחזיר **None**.

דוגמה לקלטים ופלטים:

- `monotonicity_inverse([True, True, False, False]) → [56.5, 57.5, 63, 84]`
  - זו אכן סדרה עולה ועולה ממש, אבל לא יורדת ולא יורדת ממש
- `monotonicity_inverse([False, True, False, False]) → None`
  - אין דוגמה לסדרה העונה להגדרות הנ"ל.

הערות למימוש:

- ניתן להניח כי הקלט הוא רשימה בעלת 4 איברים בולאניים (קרי *True* או *False*).
- ניתן לכתוב את התשובות המוחזרות במפורש (אין הכרח לחשבן בתוך הפונקציה).
- שימו לב: ישנם  $2^4$  קלטים חוקיים לפונקציה.
- חישבו: עד כמה ניתן לוודא את נכונות הפונקציה בעזרת הסעיף הקודם?

### 5. הדפסת מספרים ראשוניים:

רקע: היה היה לכאורה סטודנט חרוץ ושמו אספי המתמטיקאי. אספי המתמטיקאי מאוד אהב מספרים ראשוניים, אך עצלנותו מנעה ממנו למצוא אותם - שכן אין הוא מכיר שיטה קלה לחשבם. בשאלה זו נעזור לאספי המתמטיקאי, שעדיין לא עבר את קורס "מבוא למדעי המחשב" לבנות פונקציה המחזירה מספרים ראשוניים.

משימה:

- כתבו פונקציה המקבלת מספר שלם *n* ומחזירה רשימה של *n* המספרים הראשוניים הראשונים לפי סדר, החל מ 2 (כולל).
- חתימת הפונקציה: `def primes_for_asafi(n)`

## בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

### דוגמה לקלטים ופלטים:

`primes_for_asafi(3)` → [2, 3, 5]

`primes_for_asafi(1)` → [2]

`primes_for_asafi(5)` → [2, 3, 5, 7, 11]

`primes_for_asafi(6)` → [2, 3, 5, 7, 11, 13]

### הערות למימוש:

- ניתן להניח כי הקלט  $n$  הוא אכן מספר שלם גדול או שווה ל 1.
- ממשו את הפונקציה באופן יעיל במידת האפשר.
- חישבו: האם אפשר לממש את הפונקציה ע"י נוסחה סגורה כללית למספרים ראשוניים? אם כן, שתפו אותנו בפורום (האם קיימת פונקציה חח"ע ועל  $\mathbb{N}$  ל  $\mathbb{N}_{prime}$ ? אם יש, ניתן לבטאה בצורה אלמנטרית?)

### חלק ג' – לולאות מקוננות

#### 6. חיבור וקטורים

רקע: כשתלמדו על מרחבים וקטורים, תיווכחו לדעת כי פעולת החיבור במרחב וקטורי היא פעולה חשובה ושימושית. זו בדיוק הפעולה שנממש, במרחב הוקטורי החשוב והידוע,  $\mathbb{R}^n$ , עם המ"פ הסטנדרטית.

הגדרה: בהינתן שתי רשימות של מספרים:  $[x_1, x_2, \dots, x_{n-1}, x_n], [y_1, y_2, \dots, y_{n-1}, y_n]$  נגדיר את החיבור הוקטורי של הרשימות להיות הרשימה:  $[x_1 + y_1, x_2 + y_2, \dots, x_n + y_n]$  (כלומר חיבור קואורדינטה קואורדינטה).

### משימה:

- כתבו פונקציה המקבלת רשימה של וקטורים (כלומר רשימה של רשימות) ומחזירה את הסכום הוקטורי שלהם.
- def** `sum_of_vectors(vec_lst)`: חתימת הפונקציה:
- פלט הפונקציה: וקטור (רשימה) המהווה את סכום הוקטורים בקלט.

### דוגמאות לקלטים ופלטים:

`sum_of_vectors([[1,1], [1,3]])` → [2, 4]

`sum_of_vectors([[1,1, 1], [1,0, 0], [0, 0, 100]])` → [2, 1, 101]

`sum_of_vectors([[1,1,1,1,1], [1,1,1,1,1]])` → [2, 2, 2, 2, 2]

### הערות למימוש:

- ניתן להניח כי הקלט הוא אכן רשימה של רשימות.
- אם הקלט היא רשימה ריקה, הפונקציה תחזיר **None**.

## בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

- ניתן להניח כי כל אחת מהרשימות שבקלט היא רשימת מספרים מסוג int או float, והרשימות באותו האורך.
- אם הרשימות בקלט ריקות, הפונקציה תחזיר [].

### 7. בדיקת אורתוגונליות

רקע: כנאמר לעיל, לאחר שתלמדו על מרחבי מכפלה פנימית, תלמדו על המושג הנקרא אורתוגונליות. ההגדרה אומרת ששני וקטורים אורתוגונליים זה לזה, אם המכפלה הפנימית ביניהם היא 0. באופן כללי, אפשר לחשוב על זה כ"ניצבות של וקטורים זה לזה".

הגדרה: בהינתן שתי רשימות של מספרים:  $[x_1, x_2, \dots, x_{n-1}, x_n]$ ,  $[y_1, y_2, \dots, y_{n-1}, y_n]$  נאמר שהרשימות ניצבות זו לזו, אם מתקיים:  $x_1 \cdot y_1 + x_2 \cdot y_2 + \dots + x_n \cdot y_n = 0$ .

משימה:

- כתבו פונקציה המקבלת רשימה של וקטורים (כלומר רשימה של רשימות) ומחזירה את **מספר הזוגות** של הרשימות הניצבות זו לזו.
- חתימת הפונקציה: `def num_of_orthogonal(vectors)`
- פלט הפונקציה: מספר הזוגות של הרשימות הניצבות זו לזו ברשימה.

דוגמה לקלטים ופלטים:

`num_of_orthogonal([[1,0,0], [0,1,0], [0,0,1]])` → 3

`num_of_orthogonal([[0,0,0], [0,1,0], [0,0,1]])` → 3

`num_of_orthogonal([[0, 0], [1, 2], [10, 5]])` → 2

`num_of_orthogonal([[1,1,1,1], [2,1,3,3], [0, 0, 100, 33], [8, 8, 8, 1.5], [9,9,9,9]])` → 0

הערות למימוש:

- ניתן להניח כי הקלט הוא אכן רשימה של רשימות, ולא ריק.
- ניתן להניח כי כל אחת מהרשימות שבקלט היא רשימת מספרים מסוג int או float, והרשימות באותו האורך, ולא ריקות.
- **שימו לב לכפילויות**: כל זוג סופרים **פעם אחת** ולא יותר. אם ספרתם פעמיים, חלקו ב-2.
- השתמשו בפונקציה הקודמת שרשמתם לחישוב המכפלה הפנימית בין שתי רשימות. המנעו מכפילויות קוד ושמרו על קוד נאה (אולי מישוהו יקרא אותו?).
- כשאתם בודקים את עצמכם, היזהרו לא להתבלבל משימוש במספרים לא שלמים. למה? למשל אם תסכמו את 0.0001 בפייתון שמונה פעמים, לא בהכרח תקבלו 0.0008. התעלמו ממקרים כאלו, וכדי להבין למה זה קורה – חכו לקורסים הבאים.



# בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

## חלק ד' – הגשת שאלות תאורטיות

בחלק זה תדרשו לענות על שאלות תאורטיות.

היכנסו לשאלון **Ex3 - Quiz** במודל, ענו והגישו את השאלות (פירוט במודל).

### 8. \* התנדבות לקהילה – סעיף רשות (ללא ניקוד):

לא הכל נמדד בנקודות או בכסף. אם חפץ בכם הדבר, אנא פיתחו דיון בפורום התרגיל, והסבירו לחברכם לקורס ולכלל את הבא :  
נניח שבשאלה הראשונה אין להניח מהו הקלט אותו יקליד המשתמש. כלומר כבר מימשנו את הפונקציה, וכעת המשתמש מנסה להקליד כל מיני תווים מוזרים : אותיות, סימנים וכיו. מה יקרה בפועל לתכנה שכתבתם אם המשתמש יכניס תווים? למה זה קורה? ומהי לדעתכם הדרך היעילה ביותר להתמודד עם זה?

## הוראות הגשה

עליכם להגיש את הקובץ **ex3.zip** (בלבד) בקישור ההגשה של תרגיל 3 דרך אתר הקורס (moodle). ההגשה היא עד המועד הנקוב בראש הקובץ.

**ex3.zip** צריך לכלול את הקבצים הבאים בלבד :

1. **ex3.py**

ודאו שבקובץ ה **py**. שאתם מגישים אין קריאות הרצה לפונקציות שכתבתם.

## הנחיות כלליות בנוגע להגשה

- הנכם רשאים להגיש תרגילים דרך מערכת ההגשות באתר הקורס מספר רב של פעמים. ההגשה האחרונה בלבד היא זו שקובעת ושותיבדק.
- לאחר הגשת התרגיל, ניתן ומומלץ להוריד את התרגיל המוגש ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.
- באחריותכם לוודא כי – PDF הבדיקות נראה כמו שצריך.
- קראו היטב את קובץ נהלי הקורס לגבי הנחיות נוספות להגשת התרגילים.

**בהצלחה!**