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).
- אין לשנות את שמות הפונקציות והמשתנים 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.

- **הרצת טסטר: יחד עם התרגיל קיבלתם קובץ טסטר בשם CodeTests.py . לאחר שפתרתם את כל השאלות, אתם יכולים להעתיק בסוף קובץ הפתרון את הטקסט מקובץ הטסטר. הוא כולל מספר בדיקות בסיסיות כדי לראות שניתן לקרוא לפונקציות. אם הבדיקות עברו בהצלחה, יודפס הטקסט: Congrats!!! All preliminary tests passed!. שימו לב, הטסטר אינו בודק את נכונות התשובות אלא רק שחתימות הפונקציות מומשו נכון.**

- **את התרגיל יש להגיש ללא הטסטר**

פייתון למהנדסים 0509-1820 , סמסטר א' תשע"ז 2016-2017

שאלה 1

ממשו פונקציה בשם **find_tomato** המקבלת רשימת מחרוזות בשם items, ומחזירה את האינדקס של האיבר הראשון שערכו שווה ל:"tomato", ללא חשיבות לגודל האותיות (case insensitive). במידה ואין איבר כזה, הפונקציה תחזיר 1-. **אין להשתמש בפונקציה המובנית index.**

דוגמאות הרצה:

```
>>> find_tomato(["apple","orange","tomato"])
2
>>> find_tomato(["GrapeS","TomAto","pineApple"])
1
>>> find_tomato(["lemon","apple","banana","grapes"])
-1
```

שאלה 2

ממשו פונקציה בשם **even_vowels** אשר מקבלת מחרוזת בשם sentence ובודקת האם מספר אותיות התנועה במחרוזת הוא זוגי (אותיות התנועה באנגלית הן: "AEIOU"). שימו לב שבשאלה זו אין משמעות לגודל האותיות (case insensitive), כלומר "a" תיחשב כאות תנועה וכן גם "A". הפונקציה תחזיר True אם מספר אותיות התנועה במחרוזת הנתונה הוא זוגי ו-False אחרת. עבור התרגיל, המספר 0 נחשב למספר זוגי.

דוגמאות הרצה:

```
>>> even_vowels("Strawberry and Apple")
False
>>> even_vowels("Strawberry and Bannana")
True
>>> even_vowels("through")
True
```

שאלה 3

ממשו פונקציה בשם **three_div** המקבלת מספר בשם number ומחזירה את מספר הספרות שמתחלק ב-3. לדוגמא, עבור המספר 23456 הפונקציה תחזיר 2 כי המספרים 3 ו-6 מתחלקים ב-3 ללא שארית, ואילו 2,4,5 לא. (לצורך התרגיל, אם x%3==0 אז נאמר ש-x מתחלק ב-3)

דוגמאות הרצה:

```
>>> three_div(349867549)
4
>>> three_div(245720)
1
```

שאלה 4

מחרוזת מתארת שם אם היא מהצורה:

(first name)(space)(last name)

כאשר השם הפרטי ושם המשפחה הינם רצפי אותיות באנגלית המתחילים באות גדולה, ואחריה אותיות קטנות (ללא תווים אחרים שאינם אותיות).

דוגמאות למחרוזות המתארות שם:

Harry Potter

Abcd Efghi

דוגמאות למחרוזות שאינן מתארות שם:

David2 Sela

Bracha

Simcha shlonsky

Sara e. Cohen

ממשו פונקציה בשם **is_name** המקבלת מחרוזת בשם test_string ומחזירה True אם המחרוזת מתארת שם ו-False אם המחרוזת אינה מתארת שם.

דוגמאות הרצה:

```
>>> is_name("Rihanna")
False
```

```
>>> is_name("Kylie Kristen Jenner")
False
>>> is_name("Shkl Maef")
True
>>> is_name("D3fr Shaul")
False
```

שאלה 5

בשאלה זו נכתוב תוכנית לניתוח רשימת קניות המאפשרת גם את התאמת הרשימה לתקציב נתון. הרשימה תמומש בתור רשימה של רשימות כאשר כל רשימה פנימית מייצגת מוצר. האיבר הראשון בכל רשימה פנימית מייצג את שם המוצר והאיבר השני הוא מספר ממשי המייצג את מחיר המוצר בשקלים. דוגמא לרשימת קניות: [["milk",7.90],["bread",5],["pasta",10.5]]

a) ממשו פונקציה בשם **find_max** המקבלת רשימה בשם grocery_list המייצגת רשימת קניות כמוגדר למעלה, ומחזירה את האינדקס של המוצר היקר ביותר ברשימה. אם מספר מוצרים מתומחרים במחיר הגבוה ביותר, היא תחזיר את המוצר הראשון ברשימה בעל המחיר הגבוה ביותר. **אין להשתמש בפונקציה המובנית max.**

b) ממשו פונקציה בשם **check_list** המקבלת רשימה בשם grocery_list המייצגת רשימת קניות כמוגדר למעלה, וגם מספר (maximal_budget) המייצג את התקציב הזמין. הפונקציה תחזיר True אם סכום מחירי המוצרים ברשימה הוא קטן או שווה לתקציב המקסימלי, ו-False אחרת. **אין להשתמש בפונקציה המובנית sum.**

c) ממשו פונקציה בשם **adjust_to_recession** שמקבלת רשימת קניות (grocery_list) וכן מספר (maximal_budget) ובודקת אם הרשימה עומדת בתקציב. אם לא, הפונקציה תתאים את רשימת הקניות לתקציב הנתון על ידי הסרה של המוצר היקר ביותר ברשימה שוב ושוב (אם יש יותר ממוצר אחד בעל המחיר המקסימלי , יש להוציא את הראשון מביניהם מהרשימה), עד שהרשימה תעמוד בתקציב (כלומר סכום מחירי הפריטים לא יעלה על maximal_budget).
עבור כל מוצר שהורדתם מהרשימה יש להדפיס "You cannot afford __" ולבסוף אם יש מוצרים ברשימה המעודכנת יש להדפיס את תכולת הרשימה המעודכנת ואת המשפט "Go Shop!". אם הרשימה ריקה יש להדפיס "Empty list".

- **הקפידו שהפלט יהיה בדיוק על פי הדוגמאות שבהמשך.**
- **אין להשתמש בפונקציות המובנות max,sum, אבל רצוי להשתמש בפונקציות מסעיפים a,b.**
- **שימו לב שהפונקציה לא מחזירה אף ערך, אלא מעדכנת את הרשימה בתוך הפונקציה כפי שראינו בשיעור.**

דוגמאות הרצה:

```
>>> find_max([["bread",5],["banana",16.5],["milk",7.9],["lettuce",12]])
1
>>> check_list([["bread",5],“banana",16.5],[“milk",7.9],[“lettuce",12]],20)
False
>>> check_list([[“bread",5],[“banana",16.5],[“milk",7.9],[“lettuce",12]],50)
True
>>> adjust_to_recession([[“bread",5],[“banana",16.5],[“milk",7.9],[“lettuce",12]],20)
You cannot afford banana
You cannot afford lettuce
bread ,milk
Go Shop!
>>> adjust_to_recession([[“bread",5],[“banana",16.5],[“milk",7.9],[“lettuce",16.5]],4)
You cannot afford banana
You cannot afford lettuce
You cannot afford milk
You cannot afford bread
Empty List
>>> adjust_to_recession([[“bread",5],[“banana",16.5],[“milk",7.9],[“lettuce",12]],50)
bread ,banana ,milk ,lettuce
Go Shop!
```

מטריצות

בשאלות הבאות נעבוד עם מטריצות של מספרים שלמים. המטריצות ייוצגו בצורה הבאה: מטריצה שמימדיה הם m x n כלומר מטריצה עם m שורות ו-n עמודות, תיוצג על ידי רשימה של רשימות, כך שברשימה הראשית יהיו m רשימות באורך n. כך לדוגמא המטריצה:

$$A_{3x2} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

תיוצג ע"י הרשימה [[1,2],[3,4],[5,6]].

שאלה 6

ממשו פונקציה בשם **max_column** המקבלת מטריצה בשם mat תוך שימוש בייצוג שהוצג לעיל, ומחזירה את אינדקס העמודה בעלת סכום הערכים המקסימלי. אם המטריצה ריקה יש להחזיר 1-. אם יש כמה עמודות עם אותו סכום מקסימלי, יש להחזיר את אינדקס העמודה הראשונה מביניהן.

דוגמאות הרצה:

```
>>> max_column([[1,2,3],[2,3,1],[4,1,2]])
0
>>> max_column([[1,2,3],[2,3,1],[1,1,2]])
1
>>> max_column([[1,2,3],[2,3,1],[1,4,2]])
1
```

שאלה 7

ממשו את הפונקציה **create_matrix(m,n)** המקבלת 2 מספרים שלמים n,m ומחזירה מטריצה בעלת m שורות ו-n עמודות, כאשר במקום ה -(i,j) של המטריצה היא תאותחל לערך (i-1)*n+j לדוגמא בהינתן 2,2 המטריצה תהיה [[1,2],[3,4]].

תזכורת: אינדקסי המטריצה מתחילים מהמספר 1, ואילו אינדקסי הרשימה בפייתון מתחילים מ-0

```
>>> create_matrix(2,3)
[[1, 2, 3], [4, 5, 6]]
>>> create_matrix(5,3)
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15]]
```

שאלה 8

נתונה מטריצה $A_{m\ x\ n}$ בעלת m שורות ו-n עמודות, ומטריצה $B_{n\ x\ p}$ בעלת n שורות ו-p עמודות.

<u>מכפלת מטריצות:</u> מכפלת מטריצות $A_{m\ x\ n}\times B_{n\ x\ p}$ היא פעולה המקבלת שתי מטריצות A_{mXn}, B_{nXp} ומחזירה את מכפלתן: מטריצה AB שגודלה m x p . מספר העמודות במטריצה השמאלית (A) חייב להיות זהה למספר השורות במטריצה מימין (B). איברי המטריצה החדשה מוגדרים כך: $(AB)_{i,j} = \sum_{k=1}^{n} A_{i,k}\, B_{k,j}$

ממשו פונקציה בשם **multiply_matrix**, המקבלת שתי רשימות מקוננות (A,B) המייצגות מטריצות (אפשר להניח שהרשימות אכן מייצגות מטריצות חוקיות, שאבריהן מספרים (int או float)). הפונקציה תוודא שניתן להכפיל את המטריצות, כלומר שמספר העמודות במטריצה A שווה למספר השורות במטריצה B, אם לא ניתן היא תחזיר מחרוזת שגיאה:

'Error, Matrices cannot be multiplied'

במידה שניתן להכפיל את המטריצות, הפונקציה תחזיר רשימה מקוננת המייצגת את מכפלת המטריצות AB.

דוגמאות הרצה:

```
>>> multiply_matrix([[1,2,3],[4,5,6]],[[7,8],[9,10],[11,12]])
[[58, 64], [139, 154]]
>>> multiply_matrix([[1,2,3]],[[2,3,4],[2,3,4]])
'Error, Matrices cannot be multiplied'
```