תרגיל בית 4

מילונים

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 ex4_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת.
- 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 וכי התוכנית אינה קורסת).
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- **אין לשנות את שמות המשתנים שכבר מופיעים בקובץ השלד של התרגיל. יש לעבוד עם המשתנים שמופיעים בשלד התרגיל.** על הקוד של כל שאלה לעבוד ולספק את התוצאה הדרושה עבור קלט שיוזן במשתנים שמופיעים בשלד (המשתנים שלידם סימני שאלה ומחכים לקלט כפי שראינו בדוגמא מהתרגול). יחד עם זאת, אתם רשאים להוסיף משתנים נוספים כראותם עינכם.
- **שימו לב מתי מבקשים מכם להחזיר ערך מפונקציה (return) ומתי מבקשים להדפיס למסך (print).**
- אין למחוק את ההערות שמופיעות בשלד.

שאלה 1:

ממשו את הפונקציה ***lists_to_simple_dict(names, phones)*** אשר מקבלת רשימה של שמות (ללא חזרות) ורשימה של מספרי טלפון באותו האורך, ומחזירה מילון הממפה לכל שם את מספר הטלפון שלו.

- ניתן להניח שרשימות הקלט הן באותו אורך, אך הן עשויות להיות ריקות (במקרה זה יש להחזיר מילון ריק).
- names היא רשימה של מחרוזות המייצגות שמות ו-phones היא רשימה של מחרוזות המייצגות מספרי טלפון. כל מספר טלפון ברשימה phones הינו מספר הטלפון של האדם ששמו נמצא במיקום המקביל ברשימה names (כלומר מספר הטלפון במיקום i כלשהו ברשימה phones הינו מספר הטלפון של האדם ששמו מאוחסן במיקום i ברשימה names).
- במילון המוחזר כל מפתח יהיה מחרוזת המייצגת שם, וכל ערך יהיה מחרוזת המייצגת את מספר הטלפון שמתאים לשם זה.

דוגמת הרצה:

```
>>> names = ['Alon','Michal','Yael','Ofir']
>>> phones = ['123','234','345','456']
>>> dic = lists_to_simple_dict(names, phones)
>>> print (dic)
{'Alon': '123', 'Michal': '234', 'Yael': '345', 'Ofir': '456'}
```

שאלה 2:

ממשו את הפונקציה ***lists_to_complex_dict(names, phones)*** אשר מקבלת רשימה של שמות (ייתכנו חזרות) ורשימה של מספרי טלפון באותו האורך, ומחזירה מילון הממפה לכל שם את רשימת המחרוזות המייצגות את מספרי הטלפון שלו.

בסעיף זה, ייתכן ואותו שם יופיע מספר פעמים ברשימה names.

- ניתן להניח שרשימות הקלט הן באותו אורך, אך הן עשויות להיות ריקות (במקרה זה יש להחזיר מילון ריק).
- names היא רשימה של מחרוזות המייצגות שמות ו-phones היא רשימה של מחרוזות המייצגות מספרי טלפון. כל מספר טלפון ברשימה phones הינו מספר הטלפון של האדם ששמו נמצא במיקום המקביל ברשימה names (כלומר מספר הטלפון במיקום i כלשהו ברשימה phones הינו מספר הטלפון של האדם ששמו מאוחסן במיקום i ברשימה names).
- במילון המוחזר כל מפתח יהיה מחרוזת המייצגת שם, וכל ערך יהיה **רשימה** של מחרוזות המייצגת את מספרי הטלפון שמתאימים לשם זה.

דוגמת הרצה:

```
>>> names =
['Alon','Michal','Alon','Yael','Ofir','Ofir','Ofir']
>>> phones = ['123','234','124','345','456','457','458']
>>> d=lists_to_complex_dict(names, phones)
>>> print (d)
{'Alon': ['123', '124'], 'Michal': ['234'], 'Yael': ['345'],
'Ofir': ['456', '457', '458']}
```

שאלה 3:

ממשו את הפונקציה ***get_frequent_locations(text, locations)*** אשר מקבלת מחרוזת בשם text המכילה מילים באנגלית המופרדות על ידי רווח בודד, ורשימה של שמות-מקומות. הפונקציה תחזיר רשימה של 2 שמות-המקומות שמוזכרים הכי הרבה פעמים בטקסט.

- סדר שמות-המקומות ברשימה המוחזרת יהיה ממוין לפי שכיחות הופעת שמות-המקומות בטקסט – באיבר הראשון ברשימה יופיע שם-המקום הכי נפוץ ובאיבר השני ברשימה יופיע שם-המקום השני הכי נפוץ.
- ניתן להניח שהטקסט כולל לפחות 2 שמות-מקומות.
- במידה וכמה שמות-מקומות מופיעים אותו מספר פעמים בטקסט, אין חשיבות לסדר ביניהם ברשימה המוחזרת.
- היעזרו ברשימת שמות-המקומות כדי לזהות מילים בטקסט שמציינות שם-מקום, והיעזרו במילון כדי לחשב את שכיחות הופעת שמות-המקומות בטקסט.

דוגמאות:

```
>>>my_bio = "I was born in Rehovot raised in Haifa but went to
school in Jerusalem before returning to Haifa only to miss
Jerusalem and dream about Jerusalem"
>>>locations = ['Jerusalem','Haifa','Rehovot','Sitriya']
>>>print (get_frequent_locations(my_bio,locations))
['Jerusalem', 'Haifa']
>>>news = "The Washington Post reported that California and
Florida gave Trump more votes than Arizona although California
and Florida were not the focus of the campaign"
>>>locations = ['Washington','Arizona','California','Florida']
>>>print (get_frequent_locations(news,locations))
['California', 'Florida']
```

שאלה 4:

בכיתה ראינו ייצוג חסכוני בזיכרון עבור מטריצות דלילות שהינן מטריצות שמרבית איבריהן הם אפס (sparse matrix). במקום להחזיק את כל איברי המטריצה בזיכרון כרשימה של רשימות, נשתמש במילון בו נשמור רק את האיברים ששונים מאפס (כערכי המילון) ואת מיקומם במטריצה (כמפתחות המילון). עבור כל איבר במטריצה ששונה מאפס, נשמור במילון כמפתח את מיקום האיבר (tuple הכולל את אינדקס השורה ואת אינדקס העמודה) וכערך נשמור את האיבר במטריצה.

למשל, בהינתן המטריצה הדלילה באה (המיוצגת ע"י רשימה של רשימות):

```
[[0, 0, 0, 1, 0],
 [0, 0, 0, 0, 0],
 [0, 2, 0, 0, 0],
 [0, 0, 0, 0, 0],
 [0, 0, 0, 3, 0]]
```

נוכל לייצג אותה כמילון כך:

```
{(0, 3): 1, (2, 1): 2, (4, 3): 3}
```

למשל, המפתח (0,3) הינו tuple המייצג את המיקום שורה 0 ועמודה 3 וערך האיבר שם הוא 1.

ממשו את הפונקציה sum_sparse_matrices(mat1,mat2) אשר מקבלת שני מילונים המייצגים מטריצות דלילות באותם המימדים, ומחזירה מילון המייצג את מטריצת הסכום. במטריצת הפלט, כל איבר יהיה הסכום של האיברים במיקום המקביל במטריצות הקלט.

- ניתן להניח ששתי מטריצות תקינות ושהינן בעלות מימדים זהים.

<u>דוגמת הרצה:</u>

```
>>> mat1 = {(0,1): 2, (2,5): 1, (3,3): -10}
>>> mat2 = {(0,0): 5, (3,3): 8}
>>> print (sum_sparse_matrices(mat1, mat2))
{(0, 1): 2, (2, 5): 1, (0, 0): 5, (3, 3): -2}
```

בהצלחה !