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.
- 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).
- ניתן להניח כי הקלט שמקבלות הפונקציות הינו כפי שהוגדר בכל שאלה ואין צורך להתייחס לקלט לא תקין, אלא אם כן נאמר אחרת.
- אין להשתמש בספריה חיצונית כגון numpy לצורך פתרון התרגיל.

שאלה 1

ממשו פונקציה בשם **most_popular_character(my_string)** שמקבלת מחרוזת my_string ומחזירה את האות הכי שכיחה בה. במידה ויש יותר מאות אחת כזו, יש להחזיר את האות הקטנה ביותר מהן (לפי סדר לקסיקוגרפי).

דוגמאות הרצה:

```
>>> most_popular_character("HelloWorld")
'l'
```

הסבר: האות 'l' מופיעה 3 פעמים והאות 'o' פעמיים, שאר האותיות מופיעות פעם אחת בלבד. לכן 'l' היא האות השכיחה ביותר.

```
>>> most_popular_character("gggcccbb")
'c'
```

הסבר: האותיות 'g' ו- 'c' מופיעות 3 פעמים והאות 'b' מופיעה פעמיים. 'c' קטנה יותר בסידור הלקסיקוגרפי מ-'g' ולכן 'c' היא האות השכיחה ביותר.

הדרכה:

להלן דרך אפשרית לפתרון השאלה. עליכם לממש את הצעדים הבאים:

1. בנו מילון המכיל בתוכו את האותיות של המחרוזת ואת מספר המופעים של כל אות.
2. מצאו את הערך הכי שכיח במילון.
3. מצאו את המפתח הקטן מבין הערכים הנ"ל לפי סדר לקסיקוגרפי (במידה ויש כמה).

הערות:

אות גדולה ואות קטנה אינן אותן אותיות (למשל, A קטנה יותר לקסיקוגרפית מ- a).

הנחות:

- ניתן להניח שהפונקציה מקבלת מחרוזת עם לפחות אות אחת.
- ניתן להניח שהמחרוזת מכילה רק אותיות (אנגליות).

שאלה 2

בתרגול ראינו ייצוג של **sparse matrix** באמצעות מילון. בייצוג כזה, עבור כל תא במטריצה שאינו אפס, יאוחסן במילון מפתח מסוג tuple המיצג את קורדינטות התא, והערך ייצג את ערכו של התא במטריצה.

ממשו את הפונקציה **diff_sparse_matrices(lst)** אשר מקבלת רשימה lst של מילונים המייצגים sparse matrices **ומחזירה** מילון המייצג את מטריצת ההפרש.

לדוגמה:

```
>>> diff_sparse_matrices([{(1, 3): 2, (2, 7): 1}, {(1, 3): 6}])
{(1, 3): -4, (2, 7): 1}
>>> diff_sparse_matrices([{(1, 3): 2, (2, 7): 1}, {(1, 3): 2}])
{(2, 7): 1}
>>> diff_sparse_matrices([{(1, 3): 2, (2, 7): 1}, {(1, 3): 6, (9,10): 7}, {(2,7): 0.5, (4,2): 10}])
{(1, 3): -4, (2, 7): 0.5, (9, 10): -7, (4,2): -10}
```

במקרה השני – ערך מטריצת ההפרש במיקום **(1,3)** הוא 0. לכן, ערך זה הוסר מהמילון.

הערות:

- הפונקציה אמורה לקבל רק רשימה המכילה שתיים או יותר מטריצות מהתצורה הנ"ל.
- אורכן לא חייב להיות זהה (תזכורת: ערך שלא מופיע במילון שווה ל-0).
- במידה ויש יותר משתי מטריצות ברשימה, יש להחסיר גם את השלישית והלאה מהראשונה:

Final = A-B-C

הנחות:

- ניתן להניח שמימדי המטריצות זהים (כלומר, מספר השורות ומספר העמודות בכל מטריצה זהה למטריצות האחרות).
- ניתן להניח שהפונקציה מקבלת רשימה של לפחות שתי מטריצות מאותו מימד.

שאלה 3

ממשו פונקציה בשם **find_substring_locations(s, k)** המקבלת מחרוזת s ואורך k של תת-מחרוזת כמספר שלם **ומחזירה** את המילון הבא:

- המפתחות הם כל תתי המחרוזות של המחרוזת s באורך k (אוסף רציף של תווים מתוך s).
- הערך המתאים לכל מפתח הוא רשימה של כל האינדקסים בהם הוא מופיע ב-s (כל מיקום מצויין על ידיי אינדקס התו הראשון של התת-מחרוזת). למשל "ge" מתוך "drge" יופיע במקום 2.

דוגמת הרצה:

```
>>> find_substring_locations('TTAATTAGGGGCGC', 2)
{'TT': [0, 4], 'TA': [1, 5], 'AA': [2], 'AT': [3], 'AG': [6], 'GG': [7, 8, 9], 'GC': [10, 12], 'CG': [11]}
>>> find_substring_locations('TTAATTAGGGGCGC', 3)
{'TTA': [0, 4], 'TAA': [1], 'AAT': [2], 'ATT': [3], 'TAG': [5], 'AGG': [6], 'GGG': [7, 8], 'GGC': [9], 'GCG': [10], 'CGC': [11]}
>>> find_substring_locations('Hello World', 3)
{'Hel': [0], 'ell': [1], 'llo': [2], 'lo ': [3], 'o W': [4], ' Wo': [5], 'Wor': [6], 'orl': [7], 'rld': [8]}
```

כפי שניתן לראות, הפונקציה מחלקת את המחרוזת לתת-מחרוזות באורך k, בצורה רציפה, ומוצאת את כל מיקומי ההתחלה של כל תת-מחרוזת במחרוזת s.

הפונקציה צריכה לדעת להתמודד עם k בגודל: $1 \leq k \leq len(s)$

הערות:

האינדקסים ברשימות אינם שליליים (כלומר, האינדקסים הם גדולים או שווים ל-0).

שאלה 4

א. ממשו פונקציה בשם **courses_per_student(tuples_lst)** המקבלת כקלט רשימה tuples_lst של זוגות (מסוג tuple), כך שהאיבר הראשון בכל זוג הוא שם של סטודנטית (מסוג מחרוזת) והאיבר השני הינו שם הקורס שהסטודנטית לומדת (str). הפונקציה **תחזיר** מילון הממפה לכל שם של סטודנטית (key) את רשימת הקורסים (list) שאותם היא לומדת (value).

לדוגמה:

>>> courses_per_student([('Tom', 'Math'), ('Oxana', 'Chemistry'), ('Scoobydoo', 'python'), ('Tom', 'pYthon'), ('Oxana', 'biology')])

{'tom': ['math', 'python'], 'oxana': ['chemistry', 'biology'], 'scoobydoo': ['python']}

הערות:

- אם סטודנטית לומדת יותר מקורס אחד, שמה תופיע ב-tuples_lst ביותר מזוג אחד, אך עם שמות של קורסים שונים.
- אין חשיבות לסדר האיברים (שמות הקורסים) ברשימות המופיעות כ-values במילון הפלט.
- ניתן להניח ש-tuples_lst איננה ריקה.
- ניתן להניח שאין tuples זהים ברשימה tuples_lst, כלומר זוגות זהים בהם שם הסטודנטית ושם הקורס זהים.
- על מנת למנוע בלבול בנוגע לשמות הסטודנטים או לשמות הקורסים, יש להפוך אותם לאותיות קטנות ולהתייחס אליהם כאותו קורס \ סטודנטית.

ב. ממשו פונקציה בשם **num_courses_per_student(stud_dict)** אשר מקבלת כקלט את המילון stud_dict שהוא פלט הפונקציה courses_per_student שמימשתם בסעיף א' (ראו דוגמה בהמשך). הפונקציה **תשנה** את המילון stud_dict כך שיכיל את שם הסטודנטית כ-key ואת **מספר** הקורסים שהסטודנטית לומדת כ-value. **שימו לב,** הפונקציה **אינה מבצעת** return (כלומר, **אסור** לפונקציה להחזיר את המילון).

לדוגמה:

```
>>> stud_dict = courses_per_student([('Tom', 'Math'), ('Oxana', 'Chemistry'), ('Scoobydoo', 'python'),   ('Tom', 'pYthon'), ('Oxana', 'biology')])

>>> stud_dict

{'tom': ['math', 'python'], 'oxana': ['chemistry', 'biology'], 'scoobydoo': ['python']}

>>> num_courses_per_student(stud_dict)

>>> stud_dict

{'tom': 2, 'oxana': 2, 'scoobydoo': 1}
```

הערות:

ניתן להניח ש- stud_dict מכיל לפחות זוג אחד של שם סטודנטית (key) ורשימת הקורסים שהיא לומדת (value).

בהצלחה!