תרגיל בית 4

הנחיות כלליות:

- קראו בעיון את השאלות והקפידו שהתוכניות 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 ביקורת.
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- 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).
- חלק מהשאלות נבדקות באופן אוטומטי. לכן, עליכם לרשום את הקוד שלכם אך ורק במקומות המתאימים לכך בקובץ השלד.
- ניתן להניח כי הקלט שמקבלות הפונקציות תקין (אלא אם נכתב אחרת).
- **אין לשנות שמות פונקציות או משתנים שקיימים בקובץ השלד של התרגיל.**
- אין למחוק את ההערות שמופיעות בשלד.
- אין להשתמש בקריאה לספריות חיצוניות (אסור לעשות **import**).
- מועד אחרון להגשה: כמפורסם באתר.
- **יש לבדוק שהקובץ רץ ב- IDLE לפני ההגשה.**

מילונים

שאלה 1

כתבו פונקציה בשם most_popular_character שמקבלת מחרוזת ומחזירה את האות הכי שכיחה בה. במידה ויש יותר מאות אחת כזו, יש להחזיר את האות הקטנה ביותר מהן (על פי הסדר המילוני של פייתון: אותיות גדולות לפני קטנות. כלומר: A-Z לפני a-z).

דוגמאות הרצה:

most_popular_character('HelloWorld')

'l'

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

most_popular_character('caabb')

'a'

הסבר: האותיות 'a' ו 'b' מופיעות פעמיים, ו 'c' מופיעה פעם אחת. 'a' נבחרה כי היא קטנה מ 'b'.

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 (במידה ויש כמה).

<u>הערות</u>

ניתן להניח שהמחרוזת מקבלת רק אותיות (אנגליות).

במידה ויש רק אות אחת במחרוזת, הפונקציה תחזיר את אותה האות. ניתן להניח שהמחרוזת איננה ריקה.

אותויות גדולות וקטנות הן אותיות נפרדות, כלומר יש לספור את 'a' ו- 'A' בנפרד.

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 של מילונים המייצגים sparse matrices ומחזירה מילון המייצג את מטריצת ההפרש.

לדוגמה:

```
In[2]: diff_sparse_matrices([{(1,3):2, (2,7):1} , {(1,3):6}])
Out[2]: {(1, 3): -4, (2, 7): 1}
In[3]: diff_sparse_matrices([{(1,3):2, (2,7):1} , {(1,3):2}])
Out[3]: {(2, 7): 1}
```

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

<u>הערות:</u>

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

final = A – B - C

שאלה 3

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

- המפתחות הם כל תתי המחרוזות של המחרוזת s, באורך k (אוסף רצוף של תווים מתוך s).
- הערך המתאים לכל מפתח היא 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 במיקום 2.

דוגמת הרצה:

```
In[8]: find_substring_locations('TTAATTAGGGGCGC', 2)
Out[8]:
{'TT': [0, 4],
 'TA': [1, 5],
 'AA': [2],
 'AT': [3],
 'AG': [6],
 'GG': [7, 8, 9],
 'GC': [10, 12],
 'CG': [11]}
In[9]: find_substring_locations('TTAATTAGGGGCGC', 3)
Out[9]:
{'TTA': [0, 4],
 'TAA': [1],
 'AAT': [2],
 'ATT': [3],
 'TAG': [5],
 'AGG': [6],
 'GGG': [7, 8],
 'GGC': [9],
 'GCG': [10],
 'CGC': [11]}
In[10]: find_substring_locations('Hello World', 3)
Out[10]:
{'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 (כלומר במעבר של אות אחת בכל פעם, בלי תלות ב- k), וסופרת כמה פעמים כל תת-מחרוזת הופיעה במחרוזת s.

הפונקציה צריכה לדעת להתמודד עם k בגודל: 1 <= k <= len(s)

ניתן להניח ש s איננה ריקה

שאלה 4

א. כתבו פונקציה בשם courses_per_student(tuples_lst) שמקבלת כקלט רשימה tuples_lst של זוגות (מסוג tuple), כך שהאיבר הראשון בכל זוג הוא שם של סטודנט (str) והאיבר השני הינו שם קורס שסטודנט לומד (str). הפונקציה תחזיר מילון הממפה לכל שם של סטודנט (key) את רשימת הקורסים (list) שאותם הוא לומד (value).

לדוגמא:

<u>Input:</u>

courses_per_student([('Rina', 'Math'), ('Yossi', 'Chemistry'), ('Riki', 'python'), ('Rina', 'pYthon'), ('Yossi', 'biology')])

<u>Output:</u>

{'rina': ['math', 'python'], 'yossi': ['chemistry', 'biology'], 'riki': 'python'}

<u>הערות</u>:

- אם סטודנט לומד יותר מקורס אחד, שמו יופיע ב -tuple_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- tuple_list איננה ריקה.
- על מנת למנוע בלבול בנוגע לשמות הסטודנטים או לשמות הקורסים, יש להפוך אותם לאותיות קטנות, ולהתייחס אליהם כאותו קורס \ סטודנט.

ב. כתבו פונקציה בשם students_per_course(tuple_lst) אשר מקבלת כקלט את הרשימה של שמות הסטודנטים והקורסים אותם הם לומדים (כמו בסעיף א') ומוציאה כפלט מילון עם שמות הסטודנטים כ- keys ומספר הקורסים שכל סטודנט לוקח כ- values.

לדוגמא:

<u>Input:</u>

students_per_course([(‘Rina’, ‘Math’), (‘Yossi’, ‘Chemistry’), (‘Riki’, ‘python’), (‘Rina’, ‘pYthon’), (‘Yossi’, ‘biology’)])

<u>Output:</u>

{'rina': 2, 'yossi': 2, 'riki': 1}

<u>הערות:</u>

ניתן להניח את ההנחות של סעיף א' לגבי הרשימה, ולטפל באיותים אפשריים שונים של שמות הקורסים, בדומה לסעיף א' (כלומר להפוך אותם ל- lowercase).