חשיבה מחשובית ותכנות בשפת פייתון
תרגיל בית 6

הנחיות כלליות:

- קראו **היטב** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על הוראות ההגשה המפורסמות במודל. בפרט, יש להגיש את כל השאלות יחד בקובץ ex#_012345678.py המצורף לתרגיל, לאחר החלפת ה# במספר התרגיל והחלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת. למשל, אם מספר תעודת הזהות שלי הינו 112233445 ואני מגיש את תרגיל מספר 1, שם הקובץ שאגיש יהיה ex1_112233445.py.
- 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 (כן כן, עד לרמת הרווח).**
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- **יש לעבוד עם המשתנים שמופיעים בשלד התרגיל ואין לשנות את שמם.** על קטע הקוד של כל שאלה לעבוד ולספק את התוצאה הדרושה עבור קלט שיוזן במשתנים שמופיעים בשלד (המשתנים שלידם סימני שאלה ומחכים לקלט כפי שמופיע בדוגמאות בתרגול). יחד עם זאת, אתם רשאים להוסיף משתנים נוספים כראותם עינכם.
- מומלץ להתעדכן בפורום לגבי שאלות של סטודנטים אחרים וכמובן, במידה ועדיין משהו לא ברור, לשאול בעצמכם.

שימו לבים! (תקף החל מתרגיל בית מספר 5 ואילך)

- ✓ שימו לב 1, החתימה של כל אחת מהפונקציות שעליכם לממש מופיעה כבר בשלד התרגיל ואין לשנותה. עליכם לממש את גוף הפונקציה בלבד. יש למחוק את הפקודה pass (היא נמצאת שם רק כדי שתוכלו להריץ חלקי קוד בלי לשים בהערה את המימושים החסרים) ולהחליף אותה במימוש המתאים.
- ✓ שימו לב 2, בשאלה בה נתבקשתם לממש פונקציה, אין לעשות שום דבר מעבר לכך. ניתן ואף רצוי לקרוא לפונקציה שכתבתם עם מגוון קלטים על מנת לוודא את תקינותה אך אין להשאיר את הקריאות הללו בהגשת התרגיל.
- ✓ שימו לב 3, אף פונקציה לא מדפיסה דבר אלא רק מחזירה ערך כלשהו.

שאלה 1 – ממשו את הפונקציה how_many_common_letters(s1, s2) אשר מקבלת שתי מחרוזות ומחזירה מספר (מטיפוס int) המייצג את סך האותיות המשותפות ביניהן. ניתן להניח שכל אות מופיעה בכל מחרוזת לא יותר מפעם אחת (נסו לחשוב איך היה מושפע הקוד אם לא?). להלן דוגמאות הרצה:

```
>>> x = how_many_common_letters('ban', 'a')
>>> print(x)
1
>>> x = how_many_common_letters('ban', 'nba')
>>> print(x)
3
```

שאלה 2 –

עיבוד שפה טבעית (NLP) הינו תחום במדעי המחשב המתמקד בניתוח ובהבנה על ידי מחשב של טקסט שנכתב בשפה אנושית. אחד השלבים הראשונים בעיבוד טקסט בתחום זה, כולל המרת כל משפט לרשימת המילים המרכיבות אותו (על פי סדר הופעתן בו), הסרת סימני פיסוק ותווים אחרים בין מילים והמרת האותיות באנגלית לאותיות קטנות.

ממשו את הפונקציה clean_word(word, chars_list) המקבלת מחרוזת word ורשימת תווים chars_list. על הפונקציה להפוך את האותיות במחרוזת לאותיות

קטנות (lower case letters), לנקות ממנה את התווים הנמצאים ברשימה char_list (אין להסיר אף תו מעבר לכך), ולהחזיר את התוצאה. להלן דוגמאות הרצה:

```
>>> print(clean_word("hi", ['h', 't']))
i
>>> print(clean_word("#hello", ['o', '#', 'l']))
he
>>> print(clean_word("#hel#lo", ['o', '#', 'l']))
he
```

<u>שאלה 3</u> – ממשו את הפונקציה ciel_list_of_floats_in_place(floats_list) שמקבלת רשימה של מספרים מטיפוס float, ומעגלת כל מספר כלפי מעלה כך תוך שמירה על היותו float. הפונקציה לא מחזירה דבר. אין להשתמש בחבילה math. להלן דוגמאות הרצה:

```
>>> l = [1.2, -4.8, -2.0, 7.8, -10.1]
>>> ciel_list_of_floats_in_place(l)
>>> l
[2.0, -4.0, -2.0, 8.0, -10.0]
```

<u>שאלה 4</u> – clean_words_of_sentence(sentence) המקבלת מחרוזת המכילה משפט באנגלית, ומחזירה רשימה של מילים הכלולות במשפט על פי ההגדרות מטה. ניתן להניח שהמילים במשפט מופרדות על ידי רווח בודד, וכל מילה תכיל לפחות תו אחד שעליכם <u>לא</u> להסיר, כלומר, לא יהיה מצב שמסירים מילה בשלמותה, תמיד יישאר "משהו".

- יש להמיר את כל האותיות באנגלית לאותיות קטנות.
- עליכם להסיר את התווים הבאים: ,.%?!$# (אין להסיר אף תו אחר).
- כל מילה שמופיעה במשפט תופיע בדיוק באותו מיקום ברשימה המוחזרת. אם מילה מסוימת מופיעה יותר מפעם אחת, יש להוסיף אותה לרשימה כמספר הופעותיה תוך שמירה על סדר הופעת המילים ביניהם (דוגמא 2 ברשימת הדוגמאות). להלן דוגמאות הרצה:

```
>>> clean_words_of_sentence("Mr. Stark... I don't feel so #good")
['mr', 'stark', 'i', 'don't', 'feel', 'so', 'good']
>>> clean_words_of_sentence("THE CAKE IS A LIE. THE CA#KE IS A LIE.")
['the', 'cake', 'is', 'a', 'lie', 'the', 'cake', 'is', 'a', 'lie']
>>> clean_words_of_sentence("It's sim$ple, we kill!!! the batman!")
['it's', 'simple', 'we', 'kill', 'the', 'batman']
```

<u>שאלה 5</u> – ממשו את הפונקציה is_valid_pattern(p) המקבלת מחרוזת סוגריים (מחרוזת המכילה סוגריים בלבד) ומחזירה True/False בהתאם להאם מבנה הסוגריים במחרוזת תקין. מבנה סוגריים במחרוזת הינו תקין אם לכל סוגר שמאלי '(' יש מימינו סוגר ימני ')' המתאים לו (כמה דוגמאות שוות אלף מילים). להלן דוגמאות הרצה:

```
>>> is_valid_pattern('()')  # same result for empty string
True
>>> is_valid_pattern('()()')
True
>>> is_valid_pattern('())')
False
>>> is_valid_pattern('(())')
True
>>> is_valid_pattern('))((')
False
>>> is_valid_pattern('())()(')
False
>>> is_valid_pattern('(')
False
```