# תרגיל 3: מילונים ועיבוד מידע

| | |
|---|---|
| תאריך פרסום: | 04/11 |
| תאריך הגשה: | 14/11 בשעה 23:59 |
| מתרגל אחראי: | ישעיה צברי |
| משקל תרגיל: | 2 נקודות |
| מטרות העבודה: | מילונים ועיבוד מידע |

## הנחיות כלליות

קראו בעיון את ההנחיות והעבודה. לא יתקבלו ערעורים על טעויות שחרגו מההנחיות.

1. העבודה תבוצע ותוגש ביחידים.
2. מומלץ לקרוא את העבודה כולה לפני שאתם ניגשים לפתרון.
3. עליכם להוריד את קבצי הקוד שמסופק עם התרגיל ולהשלים את הקוד החסר. יש לממש את הפתרון לתרגיל אך ורק באזורים שהוגדרו לשם כך בקובץ!
4. <u>כתיבת קוד קריא</u>:
   - 4.1. השתמשו בשמות משתנים משמעותיים. שימוש בשמות לא משמעותיים עשוי לגרור לפגיעה בציון.
   - 4.2. כתבו תיעוד (הערות) שמסביר את הקוד שלכם. יש לכתוב תיעוד docstring (הערות) בכל פונקציה כפי שנלמד בכיתה.
   - 4.3. אין לכתוב הערות בעברית! עבודה שתכיל טקסט בשפה שאינה אנגלית (או פייתון) תקבל <u>ציון אפס</u> ללא אפשרות ערעור.
5. אין להשתמש בחבילות או במודולים חיצוניים <u>מלבד</u> מה שהוגדר בתרגיל! אם יש ספק ניתן לשאול בפורום המתאים (ראו סעיף 10).
6. יש לכתוב קוד אך ורק באזורים שהוגדרו לשם כך!
7. <u>הנחות על הקלט</u>:
   - 7.1. בכל שאלה יוגדר מה הקלט שהקוד מקבל וניתן להניח כי הקלט שנבדוק מקיים את התנאים הללו. אין להניח הנחות נוספות על הקלט מעבר למה שהוגדר.
   - 7.2. בכל שאלה סיפקנו עבורכם דוגמאות לקלט והפלט הרצוי עבורו. עליכם לערוך בדיקות נוספות לקוד שמימשתם ולא להסתמך על דוגמאות אלו בלבד.
8. <u>בדיקת העבודה</u>:
   - 8.1. העבודה תיבדק באופן אוטומטי ולכן על הפלטים להיות <u>זהים</u> לפלטים שמוגדר בתרגיל.
   - 8.2. טרם ההגשה יש לעבור על המסמך assignments checklist שנמצא במודל.
   - 8.3. מערכת הבדיקות קוראת לפונקציות שהוגדרו בתרגיל בצורה אוטומטית. אין לשנות את חתימות הפונקציות. חריגה מההנחיות תגרור ציון אפס.
9. <u>העתקות</u>:
   - **9.1. <u>אל תעתיקו!</u>**
   - 9.2. העתקת קוד (משנים קודמות, מחברים או מהאינטרנט) אסורה בהחלט. בפרט אין להעביר קוד בין סטודנטים. צוות הקורס ישתמש בכלים אוטומטיים וידניים כדי לזהות העתקות. תלמיד שייתפס בהעתקה יועמד בפני ועדת משמעת (העונש המינימלי לפי תקנון האוניברסיטה הוא כישלון בקורס).
   - 9.3. אנא קראו בעיון את המסמך שהכנו בנושא: https://moodle2.bgu.ac.il/moodle/mod/resource/view.php?id=1922556

10. <u>שאלות על העבודה</u>:

10.1. שאלות בנוגע לעבודה ישאלו בפורום שאלות לתרגיל במודל או בשעות הקבלה של המתרגל האחראי **<u>בלבד</u>** .

10.2. אין לפנות במייל לבודקת התרגילים או למתרגלים אחרים בנוגע לעבודות הגשה. מיילים בנושאים אלו לא יקבלו מענה.

**10.3. לפני ששואלים שאלה בפורום יש לוודא שהשאלה לא נשאלה קודם!** שאלה שחוזרת על שאלה קיימת לא תענה.

10.4. אנו מעודדים סטודנטים לענות על שאלות של סטודנטים אחרים. המתרגל האחראי יאשר שתשובה כזו נכונה.

11. <u>הגשת העבודה</u>:

11.1. עליכם להוריד את הקבצים מתיקיית "תרגיל בית 3" מהמודל. התיקייה תכיל תיקייה נוספת ובה קבצי העבודה וקובץ הוראות. עליכם למלא את הפתרון במקום המתאים ובהתאם להוראות התרגיל.

11.2. שימו לב: בנוסף לקבצי העבודה מצורף קובץ בשם get_id.py. עליכם למלא במקום המתאים בקובץ את תעודת הזהות שלכם. הגשה שלא תכיל את הקובץ הנ"ל עם תעודת הזהות הנכונה לא תיבדק ותקבל ציון אפס!

11.3. את העבודה יש להגיש באמצעות תיבת ההגשה הייעודית במודל.

11.4. פורמט ההגשה הוא להלן:

11.4.1. את התיקייה בשם hw3 שהורדתם מתיקיית העבודה במודל, ובה נמצאים קבצים הקוד שלכם, יש לכווץ לפורמט zip (קבצים אחרים, כגון קבצי rar, יקבלו ציון 0).

11.4.2. השם של התיקייה המכווצת יהיה תעודת הזהות שלכם.

11.4.3. העלו את התיקייה המכווצת לתיבת ההגשה של העבודה.

## שאלה 1

לוח "שחור ופתור" (Nonogram באנגלית) הוא חידת הגיון גרפית שבה יש לגלות ציור חבוי בטבלת משבצות לבנה על ידי השחרה של חלק מהן (ראו ויקיפדיה). בשאלה זו, תממשו פונקציה הבודקת האם פתרון נתון הוא חוקי. בשונה מהגרסה המלאה של המשחק, בגרסה שתממשו, **פתרון חוקי** ימלא אחר שני אילוצים: (1) יכלול לכל היותר רצף אחד של משבצות מושחרות בכל שורה ובכל עמודה. (2) לכל שורה ולכל עמודה יוגדר מספר המציין את אורך רצף המשבצות המושחרות המופיעות לאורך השורה והעמודה בהתאמה.

דוגמה לפתרון חוקי

Columns' constraints

| Rows' constraints | 1 | 1 | 2 | 1 |
|---|---|---|---|---|
| 1 | T | F | F | F |
| 2 | F | T | T | F |
| 2 | F | F | T | T |

דוגמה לפתרון לא חוקי

Columns' constraints

| Rows' constraints | 0 | 1 | 2 | 2 |
|---|---|---|---|---|
| 1 | F | F | F | T |
| 2 | F | T | T | F |
| 2 | F | F | T | T |

לדוגמה באיור העליון, המתאר פתרון חוקי, קיים רצף משבצות מושחרות יחיד בשורה האמצעית באורך 2, בהתאם לאילוץ. לעומת זאת באיור התחתון, המתאר פתרון לא חוקי, האילוץ על רצף המשבצות בעמודה האחרונה מוגדר להיות 2, כמות המשבצות המושחרות אכן 2, אך הן מחולקות לשני רצפים המופרדים ע"י משבצת לבנה ולכן הפתרון אינו חוקי.

בגרסה שתממשו, לוח המשחק יוגדר על ידי רשימה מקוננת בוליאנית, המייצגת מטריצה בגודל n שורות על m עמודות. כל תא במטריצה מייצג משבצת, כאשר משבצת לבנה מיוצגת באמצעות False (כלומר לא הושחרה) ומשבצת מושחרת באמצעות True.

עליכם לממש את הפונקציה:

```
verify_nonogram_board(board, rows_constraints, columns_constraints)
```

המקבלת **פתרון** של לוח "שחור ופתור" (board) ואת אילוצי השורות (rows_constraints) והעמודות (columns_constraints) ובודקת האם הפתרון חוקי.

קלט: הפונקציה מקבלת:

- **board** [List[List[bool]]]– רשימה מקוננת דו-מימדית בגודל n*m עם ערכים בוליאנים המייצגת פתרון לוח "שחור ופתור" כפי שתואר.
- **rows_constraints** [List[int]]– רשימת האילוצים לגבי אורכי רצף המשבצות המושחרות המופיעות לאורך כל שורה.
- **columns_constraints** [List[int]] - רשימת האילוצים לגבי אורכי רצף המשבצות המושחרות המופיעות לאורך כל עמודה.

פלט: הפונקציה מחזירה ערך בוליאני True כאשר הפתרון חוקי ו-False אחרת.

הנחות קלט:

- board, rows_constraints, columns_constraints אינם None.
- כלל האיברים ברשימה המקוננת board הם מסוג bool .
- כלל האיברים ברשימות האילוצים הם integers עם ערך לא-שלילי.

- אורך רשימות האילוצים תואם לגודל הלוח.
- אורך כל השורות זהה.
- אורך כל העמודות זהה.

<u>דגשים והערות:</u>

- ניתן לממש פונקציות עזר מעבר לדרישות התרגיל.

<u>דוגמאות:</u>

.1

```
board2 = [
    [False, True, True, False],
    [True, True, True, False],
    [False, False, True, True],
]
```

```
board1 = [
    [True, False, False],
    [True, True, False],
    [False, False, True],
]
```

```
>> ans1=verify_nonogram_board(board1, [1,2,1], [2,1,1])
>> ans2=verify_nonogram_board(board2, [2,3,2],[1,2,3,1])
>> print("ans1: " + ans1 + "|ans 2: " + ans2)
"ans1: True|ans2: True"
```

הפונקציה `verify_nonogram_board` תחזיר True עבור שני הפתרונות החוקיים.

.2

```
board3 = [
    [True, True, False, True],
    [True, True, True, False],
    [False, False, True, True],
]
```

```
>> ans3=verify_nonogram_board(board3, [3,3,2],[2,2,2,2])
>> print("ans3: " + ans3)
"ans3: False"
```

הפונקציה `verify_nonogram_board` תחזיר False עבור הפתרון הלא חוקי.
הפתרון הנ"ל אינו חוקי משתי סיבות; המשבצות המושחרות בשורה הראשונה אינן רצופות והמשבצות המושחרות בעמודה האחרונה אינן רצופות. כל סיבה **לבדה** מהווה צידוק מספק לפסילת הפתרון.

.3

```
board4 = [
    [True, True, False, False],
    [True, True, True, False],
    [False, False, True, False],
]
```

```
>> ans4=verify_nonogram_board(board4, [2,3,1],[2,2,2,1])
>> print("ans4: " + ans4)
```

```
"ans4: False"
```

הפונקציה `verify_nonogram_board` תחזיר False עבור הפתרון הלא חוקי.
הפתרון הנ"ל אינו חוקי מכיוון שמספר המשבצות המושחרות הרצופות בעמודה האחרונה (אין משבצות מושחרות) לא תואם לאילוץ העמודה האחרונה (`column_constraints[3] = 1`).

## שאלה 2

אותיות גדולות (capital letters) בשפה האנגלית מוצבות בתחילתו של כל משפט ובתחילתו של כל שם. לעיתים, ע"מ לשקף טון דיבור או דגש, ישתמש הכותב באותיות גדולות ע"מ להדגיש מילים מסוימות או חלקים מסוימים במילה. בשאלה זו תממשו ניתוח בסיסי של מחרוזות בניסיון להעריך את כוונת המחבר.

### סעיף א'

ממשו את הפונקציה:

**`get_all_capital_letters(text)`**

אשר מקבלת מחרוזת (text) ומחזירה רשימה של כל האותיות הגדולות המופיעות בטקסט **לפי סדר הופעתן**.

<u>קלט:</u>

- **`[str] text`** - הפונקציה מקבלת את הארגומנט `text` מסוג מחרוזת (string) המייצג טקסט כלשהו וערכו אינו None.

<u>פלט:</u> הפונקציה מחזירה רשימה הכוללת את כל האותיות הגדולות ב-`text` לפי סדר הופעתן.

<u>דוגמאות:</u>

```
>> ans = get_all_capital_letters(text='DorMaMmU')
>> print(ans)
"['D', 'M', 'M', 'U']"

>> ans = get_all_capital_letters(text='Thanos Vs. Dormammu!?')
>> print(ans)
"['T', 'V', 'D']"
```

## סעיף ב'

נגדיר תו *משמעותי* להיות כל תו שהוא אות מהאלפבית האנגלי, רווח או שורה חדשה לא נחשבים כחלק מהאלפבית. בנוסף, נגדיר *מילה נקייה* כמילה שאינה מכילה **תו לא משמעותי** כלל.

על מנת לנתח טקסט ממקורות שונים באופן אחיד, נהוג לערוך עיבוד מקדים לטקסט לפני שימוש בשיטות מתקדמות לעיבוד שפה טבעית. תחילה נהוג להסיר מהטקסט תווים לא משמעותיים, ואז להפריד בין מילים. אחרי הניקוי, נרצה לפרק את הטקסט כדי לייצר מהמילים רשימה של מילים נקיות. משמע, בטקסט המקורי, בין כל מילה נקייה למילה נקייה מפריד התו רווח בודד [ ]. המילים הנקיות נשמרות ללא רווחים.

ממשו את הפונקציה:

**`split_text_to_tokens(text)`**

אשר מקבלת מחרוזת טקסט (text) ומחזירה רשימה של **כל המילים** המופיעות ב-`text` כמילים נקיות המסודרות על פי סדר הופעתן בטקסט.

<u>קלט:</u> הפונקציה מקבלת פרמטר בודד:

- **`text - [str]`** ארגומנט מטיפוס מחרוזת שערכו אינו None.

<u>פלט:</u> הפונקציה מחזירה רשימה של המילים הנקיות בטקסט כמחרוזות בסדר הופעתן במקור.

<u>דגשים והערות:</u>

- מילה לאחר ניקוי שאורכה 0, לא תוחזר כחלק מרשימת המילים הנקיות.

<u>דוגמאות:</u>

```
>> text1 = "   Dormam4mu I've co|me to bar+gain"
>> ans = split_text_to_tokens(text=text1)
>> print(ans)
"['Dormammu', 'Ive', 'come', 'to', 'bargain']"
# notice the empty spaces in the beginning of text1
>> text2 = 'you a^re so fun"ny'
>> ans = split_text_to_tokens(text=text2)
>> print(ans)
"['you', 'are', 'so', 'funny']"
```

סעיף ג'

נגדיר ציון של מילה כאחוז האותיות הגדולות מתוך כלל התווים במילה.

ע"מ להעריך את הטון אליו התכוונו מחברים שונים בכותבם טקסטים מגוונים, נדרג טקסטים בעזרת ציון המנורמל בין 0-1. נגדיר את הציון של טקסט כלשהו, כציון הממוצע של כלל המילים (נקיות כמוגדר בסעיפים הקודמים) בטקסט.

ממשו את הפונקציה:

**`grade_text_tone(text)`**

אשר מקבלת מחרוזת טקסט (text) ומחזירה **מחרוזת** המייצגת את הציון של הטקסט בדיוק של ארבע נקודות עשרוניות.

קלט: הפונקציה מקבלת פרמטר בודד:

- **`text [str]`** – ארגומנט מטיפוס מחרוזת שערכו אינו None.

פלט: הפונקציה מחזירה מחרוזת המייצגת ציון מספרי בין 0-1 עם דיוק של 4 נקודות אחרי הנקודה העשרונית.

דגשים והערות:

- **שימו לב**, הפונקציה מבצעת ניקוי של הטקסט (כפי שהוגדר בסעיפים הנ"ל) לפני חישוב הציון.
- אין לחשב ציון למילה בעלת הערך "" (מחרוזת ריקה) או להכלילו בממוצע לציון הטקסט כולו.
- אם המחרוזת text לאחר ניקוי, אינה מכילה מילים, ציון המחרוזת כולה הוא המחרוזת '0.0000'.

דוגמאות:

.1

```
>> text1 =" DormaMMu I've COme to barGAin"
>> ans = grade_text_tone(text=text1)
>> print(ans)
"0.2988"
```

הסבר:

לאחר פירוק וניקוי המילים בטקסט נחשב **לחמש** מילים את הציון. שימו לב שאת הרווח בתחילת המשפט לא נחשיב כמילה (ולא נחשב לו ציון). המילה הראשונה תקבל את הציון 3/8 שכן ישנן 3 אותיות גדולות מתוך 8. המילה השנייה תקבל את הציון 1/3, השלישית 2/4, הרביעית 0/2 והחמישית 2/7. נמצע את ציוני המילים (סה"כ חמש) והתוצאה הסופית שתתקבל היא:

$$\frac{\frac{3}{8}+\frac{1}{3}+\frac{2}{4}+\frac{0}{2}+\frac{2}{7}}{5} = \mathbf{0.2988}095238095238$$

.2

```
>> text2 ='**'
>> ans = grade_text_tone(text=text2)
>> print(ans)
"0.0000"
```

## שאלה 3

במכללת EliEliyahu פנו אליכם על מנת שתממשו את מערכת הניקוד האוטומטית.

### סעיף א'

בסוף כל סמסטר, כלל הסטודנטים (מנוסח בלשון זכר אך פונה לכלל המגדרים והמינים וכו') מגישים חיבור. כל חיבור שייך לסטודנט אחד בלבד ומוזן למערכת הניקוד האוטומטית. מכיוון שבאנגלית נהוג ששמות פרטיים ושמות משפחה **מתחילים** באות גדולה ושאר האותיות בשם הינם באות קטנה, נאכוף נוהג זה.

ממשו את הפונקציה:

**`register_students_submissions(students_raw_submissions)`**

המקבלת רשימה של שמות הסטודנטים והחיבור שלהם (students_raw_submissions) ומחזירה מילון המכיל את שם הסטודנטים כמפתח ואת החיבור המתאים שלהן כערך.

<u>קלט:</u> הפונקציה מקבלת פרמטר בודד:

- **`[List[str]] students_raw_submissions`** - משתנה מסוג רשימה שערכה אינו None, כאשר כל אלמנט ברשימה הוא מחרוזת (string). כל מחרוזת מורכבת משם הסטודנט ומהחיבור שלו, מופרדים באמצעות התו "|" (קו אנכי).

<u>פלט:</u> הפונקציה מחזירה משתנה מסוג מילון בו כל מפתח הוא שם סטודנט (כמחרוזת) וכל ערך הוא החיבור שהגיש הסטודנט (כמחרוזת).

<u>הנחות קלט:</u>

- מבנה כל string ברשימה `students_raw_submissions` תואם למבנה הבא: "`FirstName LastName|A short or long assay`".
- כל **שם סטודנט** (המחרוזת עד לקו האנכי) מורכב משם פרטי ושם משפחה המופרדים באמצעות רווח בודד בלבד.

- **החיבור** כתוב באנגלית ואינו מכיל אף מופע של התו "|" (קו אנכי).

דגשים והערות:

- יש לאכוף **תקינות שמות הסטודנטים** של אות גדולה ראשונה ושאר האותיות קטנות (כמתואר בהסבר הפותח) ולהמיר כל שם פרטי ושם משפחה שאינו עומד באילוץ למבנה התקין.
- ייתכן וסטודנט הגיש יותר מפעם אחת חיבור (שמות שחוזרים על עצמם במחרוזות שונות ברשימה), עליכם לשמור את ההגשה האחרונה בלבד (הרשימה students_raw_submissions שומרת על סדר ההזנה/הגשה).

<u>דוגמה:</u>

```
>> students_raw_submissions = [
    "Ada Lovelace|Who wrote the fiRst Algorithm??",
    "Allen turing|I cracked tHe Enigma machine",
    "Rick SanChez|I don't make mistakes",
    "Ada lovelace|I wrOte the fiRst Algorithm"]
>> ans=register_students_submissions(students_raw_submissions)
>> print(ans)
" {'Ada Lovelace': "I wrOte the fiRst Algorithm",
 'Allen Turing': "I cracked tHe Enigma machine",
 'Rick Sanchez': "I don't make mistakes"}
"
```

### סעיף ב'

עבור כל חיבור, המרצה מעוניין לתת ציון לעבודה. דרישת הבסיס לכל חיבור היא אורכו: הסטודנטים נדרשו לחיבור באורך 2-10 מילים (**כולל שתי קצוות הטווח**) בלבד. כל חיבור שאינו באורך מתאים, יקבל את הציון "F" כstring. כל חיבור שעומד בדרישת הבסיס של אורך החיבור יקבל את הניקוד שהוגדר כממוצע אחוז האותיות הגדולות בכל מילה בחיבור, כפי שהוגדר בשאלה 2 סעיף ג'.

ממשו את הפונקציה:

**`grade_students_submissions(students_submissions)`**

המקבלת מילון של שמות הסטודנטים והחיבור שלהם (students_submissions) ומחזירה מילון עם ציוני החיבור לכל סטודנט.

ציון כל חיבור יהיה 'F' (<u>כמחרוזת</u>) אם הוא אינו עומד בדרישת הבסיס שהוגדרה (אורך), או **מחרוזת** המייצגת את הציון של הטקסט בדיוק של ארבע נקודות עשרוניות.

<u>קלט:</u> הפונקציה מקבלת פרמטר בודד:

- **students_submissions [dict[str, str]]** - משתנה מסוג מילון שערכו אינו None, מכיל את שמות הסטודנטים כמפתח ואת החיבור אותו כתבו כערך.

פלט: הפונקציה מחזירה מילון עם שם הסטודנט כמפתח (מחרוזת) וציון החיבור של הסטודנט (מחרוזת).

הנחות קלט:

- בסעיף זה, שמות הסטודנטים המשמשים כמפתחות עומדים באילוץ אות גדולה ראשונה וכל שאר האותיות קטנות (כמוגדר בסעיף הקודם).
- כל **שם סטודנט** (המחרוזת עד לקו האנכי) מורכב משם פרטי ושם משפחה המופרדים באמצעות רווח בודד בלבד.
- כל הערכים והמפתחות במילון הם משתנים מטיפוס string וערכם אינו None.

דוגמה:

```
>> students_submissions = {'Ada Lovelace': 'Hi DarK MoOn',
                        'Allen Turing': 'I',
                        'Rick Sanchez': "I don't CONFOUND TO THESE
CONSTRAINTS OF ARTIFICIAL LANGUAGE AND OR MIND CONTROL"}

>> ans= grade_students_submissions(students_submissions)
>> print(ans)
"
{'Ada Lovelace': '0.5000',
'Allen Turing': 'F',
'Rick Sanchez': "F"}
"
```

## סעיף ג'

המכללה ביקשה לחשב את שכיחות המילים המופיעות בכלל החיבורים של הסטודנטים. שתי מילים יוגדרו זהות אם הגרסה הנקייה שלהן זהה עד כדי הפיכה של אותיות קטנות לגדולות או ההיפך (ראו דוגמה).

ממשו את הפונקציה:

**calculate_tokens_frequencies(students_submissions)**

המקבלת מילון של שמות הסטודנטים והחיבור שלהם (students_submissions) ומחזירה מילון עבור המילים המופיעות בחיבורים (כמפתחות) והשכיחות שלהן (כערכים).

קלט: הפונקציה מקבלת פרמטר בודד:

- **`[dict[str, str]] students_submissions`** - משתנה מסוג מילון שערכו אינו None, מכיל את שמות הסטודנטים כמפתח ואת החיבור אותו כתבו כערך.

פלט: הפונקציה מחזירה מילון המורכב מהמילים הייחודיות כמפתחות (**בגרסתן הנקייה ובאותיות קטנות בלבד**) המופיעות בחיבורים, ואת כמות הפעמים שהמילה הופיעה בכל החיבורים כinteger.

הנחות קלט:

- שמות הסטודנטים המשמשים כמפתחות עומדים באילוץ אות גדולה ראשונה וכל שאר האותיות קטנות (כמוגדר בסעיף הקודם).
- כל שם סטודנט מורכב משם פרטי ושם משפחה המופרדים באמצעות רווח בודד בלבד.
- כל הערכים והמפתחות במילון הם משתנים מטיפוס string וערכם אינו None.

דוגמה:

```
>> students_submissions = {
'Ada Lovelace': 'I wrOte the fiRst Algorithm',
'Allen Turing': 'I cracked the enigma machine',
'Rick Sanchez': "I went back in time and cRACked the e'nigma MachIne
too"}
>> ans= calculate_tokens_frequencies(students_submissions)
>> print(ans)
"
{'i': 3, 'wrote': 1, 'the': 3,
'first': 1, 'algorithm': 1, 'cracked': 2,
'enigma': 2, 'machine': 2, 'went': 1,
'back': 1, 'in': 1, 'time': 1,
'and': 1, 'too': 1}
"
```

# בהצלחה !