בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101 – סמסטר ב' 2022
תרגיל 4 – Hangman
להגשה בתאריך **06/04/2022** בשעה 22:00

הקדמה

בתרגיל זה נממש וריאציה של המשחק "האיש התלוי" (hangman).

מטרת המשחק היא לנחש נכונה מילה או ביטוי שנבחרו על ידי אחד השחקנים באמצעות ניחוש של האותיות המרכיבות אותם.

בשלב ראשון, אחד השחקנים בוחר מילה, ורושם קווים אופקיים אחד ליד השני כמספר האותיות. השחקן האחר מנחש אותיות: אם האות שניחש מופיעה במילה שבחר השחקן הראשון, אז השחקן חושף את האות בכל המקומות שבהם היא מופיעה. אם האות שניחש שגויה, השחקן הראשון מצייר חלק אחד מתוך עמוד תלייה שעליו תלוי אדם ורושם את האות השגויה בצד. על השחקן המנחש להצליח לנחש את המילה בטרם ישלים השחקן הראשון את עמוד התלייה.

ניתן לקרוא עוד בוויקיפדיה: https://en.wikipedia.org/wiki/Hangman_(game)

במימוש שלנו יתקיימו התנאים הבאים:

1. המילה שצריך לנחש היא אחת ומורכבת רק מאותיות שהן Lower Case.
2. האותיות בתבנית שאינן גלויות ייוצגו ע"י התו _ (קו תחתון).
3. התבנית, המילה והאותיות מיוצגות בתור מחרוזות.
4. השחקן יכול לבחור לנחש אות אחת מהתבנית, או את המילה המלאה.
5. לשחקן מוקצות מספר נקודות בתחילת כל סבב משחקים.

   כל ניחוש עולה לו נקודה, אך הוא יכול להרוויח נקודות נוספות אם הניחושים נכונים.
6. השחקן יכול לבקש רמז המציע לו מילים אפשריות לפתרון, כל בקשה לרמז עולה לשחקן נקודה.
7. לא נצייר עמוד תלייה, בניגוד למשחק המקורי.

ניתן להריץ את פתרון בית הספר ע"י הרצת הפקודה: **~intro2cs2/bin/ex4/hangman** במחשבי בית הספר.

שימו לב: אנו ממליצים מאוד לקרוא את כל הוראות התרגיל המפורטות במסמך זה לפחות פעם אחת לפני שאתם ניגשים לפתרון!

בנוסף הקפידו על כתיבת קוד קריא על פי חוקי ה-Coding Style שהוצגו בקורס!

חלק א'

בחלק זה תממשו את המשחק כאשר המחשב מגריל מילה והמשתמש מנסה לגלות אותה. לאחר שהמשחק מסתיים, השחקן יכול לבחור לשחק את המשחק שוב ושוב.

עליכם ליצור קובץ בשם **hangman.py**, ולייבא אליו את הקובץ **hangman_ helper.py** בו ממומשות מספר פונקציות בהן תוכלו להיעזר (פירוט הפונקציות והסברן בהמשך). כמו כן, ודאו כי הורדתם לאותה תיקייה בה אתם עובדים גם את הקובץ **words.txt** המכיל את רשימת המילים.

1. ממשו את הפונקציה **update_word_pattern(word, pattern, letter)** המקבלת כפרמטרים את המילה, התבנית הנוכחית ואות, ומחזירה תבנית מעודכנת המכילה את אותה אות.

לדוגמא:

update_word_pattern('apple' , '___l_' , 'p')

תחזיר:

'_ppl_'

ניתן להניח שהתבנית המתקבלת כפרמטר תואמת למילה (כלומר שאורכי התבנית והמילה זהים, וכן האותיות שנחשפו בתבנית תואמות לאותן אותיות במילה).

2. ממשו את הפונקציה **run_single_game(words_list, score)** שמקבלת רשימת מילים ומספר נקודות איתן השחקן מתחיל את המשחק, ומריצה משחק <u>אחד</u> (כפי ששמה מרמז). על הפונקציה להחזיר בסיום המשחק את מספר הנקודות של השחקן בסוף המשחק. בכל משחק שלושה שלבים:

<u>אתחול המשחק:</u>

- הגרלת מילה מתוך רשימת המילים על ידי שימוש בפונקציה **get_random_word** הממומשת ב-**hangman_helper.py**.
- בשלב זה רשימת הניחושים השגויים היא ריקה, אורך התבנית כאורך המילה, וכל אותיותיה אינן גלויות.

<u>מהלך המשחק:</u>

בכל סבב של המשחק:

- נציג את המצב הנוכחי ע"י קריאה ל- **display_state** הממומשת ב- **hangman_helper.py** (ראו פירוט לגבי הפרמטרים שהפונקציה מקבלת בהמשך). שימו לב כי אחד הפרמטרים של פונקציה זו הוא הודעה המודפסת למשתמש. עליכם לקבוע את פרמטר ההודעה הנ"ל להודעה מתאימה בהתאם למצב המשחק לאחר האיטרציה הקודמת של המשחק (בסבב הראשון תוכלו לקבוע את הפרמטר להודעה מתאימה לבחירתכם)

- נקבל את הקלט מהמשתמש ע"י קריאה ל- **get_input** הממומשת בקובץ העזר. פירוט על ערכי ההחזרה נמצא ברשימה בהמשך.
- אם הקלט הוא ניחוש של אות נבצע את הפעולות הבאות:
  - אם הקלט אינו תקין, כלומר אורכו גדול מאחד או שאינו אות, או שאינו אות קטנה (lowercase) נקבע את פרמטר ההודעה למשתמש עם הודעה לבחירתכם המציינת שהקלט אינו תקין ונעבור לסבב הבא.
  - אחרת, אם האות שנבחרה כבר נבחרה בעבר, ניתן לפרמטר ההודעה למשתמש הודעה מתאימה לבחירתכם המציינת זאת ולאחר מכן נעבור לסבב הבא.
  - אחרת, נפחית נקודה אחת למשתמש. כמו כן:
    - אם האות שנבחרה מופיעה במילה, יש לעדכן את התבנית ע"י קריאה לפונקציה **update_word_pattern** שמימשתם קודם. בנוסף אם האות מופיעה במילה n פעמים, השחקן מקבל n*(n+1)//2) נקודות סה"כ. לאחר מכן נעבור לסבב הבא.
    - אחרת, האות שנבחרה לא מופיעה במילה, לכן נעדכן את רשימת הניחושים השגויים. לאחר מכן נעבור לסבב הבא.
- אם הקלט הוא ניחוש של מילה, נפחית נקודה אחת למשתמש. אם המילה שהשחקן ניחש נכונה, וכתוצאה מהניחוש שלו נחשפו n אותיות (כולל מופעים חוזרים של אותה אות), השחקן מקבל n*(n+1)//2) נקודות. למשל, אם התבנית הנוכחית היא 'a___e' והשחקן ניחש נכונה שהמילה היא 'apple', הוא ירוויח (3*4)//2 נקודות.

הערה 1: הקפידו על כך שאין חזרה על אותיות ברשימת הניחושים השגויים שביצע השחקן.

הערה 2: אם חלק ב' עדיין לא מומש בשלב זה והקלט הוא בקשת רמז, הגדירו את פרמטר ההודעה למשתמש בתור הודעה לבחירתכם שהרמז כרגע אינו נתמך, ועברו לסבב הבא.

הערה 3: שימו לב שעל פי הנוסחה הנ"ל לחישוב כמות הנקודות ששחקן מקבל עבור ניחוש נכון של אות או מילה, עדיף לשחקן לנחש מילה בשלמותה אם הוא יודע את התשובה ולא כל אות בנפרד.

המשחק מסתיים כאשר התבנית נחשפה במלואה, או שמספר הנקודות של השחקן הוא 0. אחרת, נמשיך לסבב נוסף של המשחק.

<u>בסיום המשחק:</u>

נקרא לפונקציה **display_state** כאשר:

- הודעה המודיעה לשחקן על ניצחון או הפסד תשלח כפרמטר (msg) לפונקציה. במקרה של הפסד, נשרשר להודעה את המילה אותה אמור היה השחקן לנחש.

- על הפונקציה **run_single_game** להחזיר את מספר הנקודות של השחקן בסוף הסיבוב הנוכחי.

3. הגדירו את הפונקציה **()main** שאינה מקבלת ואינה מחזירה ערכים ומבצעת את הפעולות הבאות:

- טעינת קובץ המילים words.txt לתוך רשימה ע"י שימוש בפונקציה **load_words** שנמצאת בקובץ העזר.
- נריץ את המשחק (ע"י קריאה לפונקציה **run_single_game** שמימשתם קודם לכן) עם כמות נקודות התחלתית השווה בערכה לערך המשתנה **POINTS_INITIAL** שנמצא בקובץ העזר.
- לאחר שהמשחק נגמר השחקן יכול לבחור לשחק משחק נוסף. במידה והוא ניצח במשחק האחרון, כמות הנקודות החיובית ממשיכה איתו לסבב הבא. נתאר את ההתנהגות בפירוט:
  - במידה והשחקן סיים את המשחק האחרון עם כמות נקודות חיובית, השתמשו בפונקציה **play_again** על מנת להציג למשתמש הודעה לבחירתכם. ההודעה תציין את: מספר המשחקים ששיחק עד כה, כמות הנקודות שצבר עד כה, וכן תשאל האם להמשיך למשחק נוסף. במידה והשחקן בוחר להמשיך, נתחיל משחק חדש עם אותה כמות הנקודות איתה השחקן סיים את המשחק הקודם, ונקדם את מספר המשחקים ששיחק. אחרת, נסיים את התוכנית.
  - אם השחקן סיים את המשחק האחרון עם 0 נקודות, השתמשו בפונקציה **play_again** על מנת להציג למשתמש הודעה לבחירתכם. ההודעה תציין את: מספר המשחקים ששיחק עד כה, כמות הנקודות שצבר עד כה, וכן תשאל האם **להתחיל סבב משחקים חדש.** במידה והשחקן בוחר להתחיל סבב משחקים חדש, נתחיל לספור מחדש את מספר המשחקים שהשחקן שיחק, נאתחל מחדש את כמות הנקודות של השחקן להיות **POINTS_INITIAL** ונתחיל משחק חדש. אחרת, נסיים את התוכנית.

על מנת להריץ את התוכנית עליכם לקרוא לפונקציה **()main** שמימשתם קודם לכן, ע"י הוספת קטע הקוד הבא בסוף הסקריפט:

```
If __name__ == "__main__":
        main()
```

חלק ב'

בחלק זה נוסיף לשחקן את האפשרות לבקש רמז המציע מילים אפשריות שיכולות להתאים לתבנית. בקשת רמז עולה לשחקן נקודה אחת.

1. ממשו את הפונקציה **filter_words_list(words, pattern, wrong_guess_lst)** המקבלת כקלט רשימה של מילים, תבנית ורשימת ניחושים שגויים, ומחזירה רשימה חדשה שמכילה רק את המילים ברשימת הקלט שיכולות להתאים לתבנית ולניחושים הקודמים.

   מתוך רשימת כל המילים נסנן את המילים שהן:
   - באותו אורך של התבנית.
   - שמכילות אותיות זהות בדיוק באותם מיקומים של האותיות הגלויות בתבנית ושאותיות אלו לא נמצאות במיקום אחר במילה המסוננת.
   - לא מכילות אף אחת מהאותיות המופיעות ברשימת הניחושים השגויים.

לדוגמה, אם התבנית הנוכחית היא "**d** _ _ _ _ **a** _ _ _ _" ורשימת הניחושים שלנו מכילה את האותיות ['b','c'] אז רשימת מילים מסוננות אפשרית היא ['**d**elegating', '**d**erogation', '**d**ishwasher']. יש לממש את הפונקציה בצורה כזו שרשימת הפלט תשמר את הסדר של המילים כפי שהופיעו ברשימת הקלט.

2. עדכנו את הפונקציה **run_single_game** שמימשתם בחלק א':
   - במידה והשחקן ביקש רמז:
     - נפחית מניקוד השחקן נקודה אחת.
     - נקרא לפונקציה **filter_words_list** שמימשתם קודם לכן כדי לקבל את רשימת המילים המסוננות שמתאימות לתבנית הנוכחית.
     - במידה ואורך הרשימה גדול מערך המשתנה **HINT_LENGTH** אשר בקובץ העזר, ניקח מתוכה את תת הרשימה הבאה עם **HINT_LENGTH** איברים בלבד מהמיקומים הבאים ברשימה המקורית (n הוא אורך הרשימה המקורית):

0, n//HINT_LENGTH, (2*n)//HINT_LENGTH, ..., ((HINT_LENGTH - 1)*n)//HINT_LENGTH

     - נקרא לפונקציה **show_suggestions** שבקובץ העזר עם רשימת מילות ההצעה.
   - שימו לב שאם השחקן ביקש רמז כשברשותו רק נקודה אחת, בקשת הרמז בהכרח תגרום להפסד של השחקן. במקרה כזה בקשת הרמז תגרום להפסד רק לאחר הצגת הרמז.

רשימת הפונקציות הממומשות ב- **hangman_helper.py**

1. **load_words()** - פונקציה שלא מקבלת קלט, ומחזירה את רשימת המילים המופיעות ב-**words.txt**.
2. **get_random_word(words_list)** - פונקציה המקבלת כקלט רשימת מילים ומחזירה מילה אקראית מתוך הרשימה.
3. **display_state(pattern, wrong_guess_lst, points, msg)** - פונקציה זו מציגה את המצב הנוכחי: את התבנית, את רשימת הניחושים השגויים, את כמות הנקודות הנוכחית ואת ההודעה למשתמש. הפונקציה אינה מחזירה ערך.
4. **get_input()** - הפונקציה מחזירה קלט מהמשתמש שהוזן על-ידו. הקלט יכול להיות ניחוש אות, ניחוש מילה או בקשה לרמז. הפונקציה מחזירה זוג (tuple) כאשר האיבר הראשון הוא סוג הקלט, כלומר אחד מהמשתנים LETTER, WORD, HINT המוגדרים בקובץ. האיבר השני יהיה האות במקרה שהקלט הוא אות, מילה במקרה שהקלט הוא מילה, והערך None אם הקלט הוא הוא רמז.
5. **show_suggestions(matches)** - פונקציה המקבלת רשימת מילים מוצעות לפתרון, ומדפיסה אותן למשתמש.
6. **play_again(msg)** - פונקציה המקבלת כקלט הודעה (מחרוזת) שמהווה שאלה למשתמש אם להתחיל משחק חדש, אותה היא מדפיסה, לאחר מכן היא מקבלת כקלט מהמשתמש את בחירתו, ומחזירה True/False בהתאם לתשובתו (ערך בוליאני).

רשימת המשתנים הגלובליים לשימושכם המוגדרים ב **hangman_helper.py**

1. **POINTS_INITIAL**: מספר הנקודות ההתחלתי איתו שחקן מתחיל סדרת משחקים חדשה.
2. **HINT_LENGTH**: כמות המילים המוצעות אשר יוצגו בבקשת רמז.
3. **LETTER**: מציין שסוג הקלט הוא ניחוש אות. יכול לחזור בתור הערך הראשון מבין ערכי ההחזרה של הפונקציה **get_input** שבקובץ העזר.
4. **WORD**: מציין שסוג הקלט הוא ניחוש מילה. יכול לחזור בתור הערך הראשון מבין ערכי ההחזרה של הפונקציה **get_input** שבקובץ העזר.
5. **HINT**: מציין שסוג הקלט הוא רמז. יכול לחזור בתור הערך הראשון מבין ערכי ההחזרה של הפונקציה **get_input** שבקובץ העזר.

שימו לב: התוכנית אמורה לפעול בצורה נכונה גם במידה וערכי הקבועים הגלובליים מוחלפים מאלו שקיבלתם בקובץ העזר.

הגשת התרגיל:

עליכם להגיש קובץ zip הנקרא ex4.zip ומכיל את הקובץ **hangman.py** בלבד. שימו לב שאין להגיש את הקובץ **hangman_helper.py**.

בהצלחה!