מבוא למדעי המחשב 67101

תרגיל 4 - רצפים ולולאות

להגשה ב

בתרגיל זה נממש את משחק "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 בויקיפדיה: (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. על מנת שהממשק הגרפי יציג את התמונות כראוי, נעשה שימוש בספרייה PIL של פייתון.
5. הספרייה מותקנת בחוות המחשבים באוניברסיטה, וניתן להתקין אותה בבית. אם התקנתם את פייתון לפי ההוראות שניתנו בתחילת הסמסטר, רשמו בקובץ פייתון את השורה:
   import PIL.Image, PIL.ImageTk , ואם החבילה לא מותקנת, פייצ'ארם יסמנה באדום. לחיצה עם כפתור ימני על הסימון האדום יעלה אפשרות להתקנה של החבילה. אם תתקלו בקשיים תוכלו לפנות ל-Lab-Support לסיוע. בכל מקרה חובה לבדוק את התרגיל על מחשבי החווה בטרם ההגשה!

ניתן להריץ את פתרון בית הספר ע"י הרצת הפקודה **~intro2cs2/bin/ex4/hangman** במחשבי החווה.
שימו לב: בתרגיל זה תעבדו מול API של מחלקה שמימשנו עבורכם, נסו לקרוא את הפונקציות בהם תצטרכו להשתמש מתוך hangman_helper.py ולהבין איך להשתמש בהן - אילו פרמטרים הן מקבלות, מה הן מחזירות, ואם תרגישו חזקים - גם איך הן עובדות. בנוסף, התרגיל מחולק לשני חלקים. אתם יכולים לממש רק את חלק א' ולהריץ את המשחק לבדיקה, אבל כדאי מאוד שתקראו את התרגיל עד הסוף כדי להבין את שאר החלקים בטרם אתם ניגשים לפתרון החלק הראשון.

חלק א

בחלק זה תממשו את המשחק כאשר המחשב מגריל מילה והמשתמש מנסה לגלות אותה.

עליכם ליצור קובץ בשם hangman.py, ולייבא אליו את הקובץ hangman_helper.py בו ממומשות מספר פונקציות בהן תוכלו להעזר (פירוט הפונקציות והסברן בהמשך). כמו כן ודאו כי הורדתם לאותה תיקייה בה אתם עובדים גם את הקובץ words.txt המכיל את רשימת המילים, ואת שבעת קבצי התמונות (hangman0.png עד hangman6.png)

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_'

2. ממשו את הפונקציה *run_single_game*(*words_list*) שמקבלת רשימת מילים, ומריצה את המשחק עצמו. במשחק שלושה שלבים:

<u>אתחול המשחק:</u>

1. הגרלת מילה מתוך רשימת המילים על ידי שימוש בפונקציה get_random_word הממומשת ב hangman_helper.py
2. בשלב זה רשימת הניחושים השגויים היא **ריקה**, אורך התבנית כאורך המילה, וכל אותותיה אינן גלויות.
3. ההודעה למשתמש בתחילת המשחק תהיה DEFAULT_MSG (מתוך קובץ העזר)

<u>מהלך המשחק:</u>

<u>כל עוד</u> לא הסתיים המשחק נבצע **איטרציה** (סבב) נוספת של המשחק. המשחק לא יסתיים כל עוד התבנית לא נחשפה במלואה ומספר הניחושים השגויים קטן מזה שמוגדר במשתנה MAX_ERRORS שבקובץ העזר. שימו לב שברשימת הניחושים השגויים אין חזרות.

<u>בכל איטרציה</u> של המשחק:

1. נציג את המצב הנוכחי ע"י קריאה ל - display_state הממומשת ב - hangman_helper.py.
2. נקבל את הקלט מהמשתמש ע"י קריאה ל - get_input הממומשת בקובץ העזר. פירוט על ערכי ההחזרה נמצא ברשימה בהמשך.
3. אם הקלט הוא ניחוש של השחקן נבצע את הפעולות הבאות:
   a. אם הקלט אינו תקין, כלומר אורכו שונה מאחד או שאינו אות, או שאינו אות קטנה (lowercase) ניתן לפרמטר ההודעה את הערך NON_VALID_MSG וממשיכים לחכות לקלט הבא
   b. אחרת, אם האות שנבחרה כבר נבחרה בעבר, ניתן לפרמטר ההודעה את הערך ALREADY_CHOSE_MSG שבקובץ העזר ונשרשר למחרוזת הזו את האות שנבחרה (בחירה זו לא תחשב כאות נוספת)
   c. אחרת, אם האות שנבחרה מופיעה במילה, יש לעדכן את התבנית ע"י קריאה לפונקציה update_word_pattern שמימשתם קודם ולתת לפרמטר ההודעה את הערך DEFAULT_MSG
   d. אחרת, האות שנבחרה לא מופיעה במילה, לכן נעדכן את רשימת הניחושים השגויים, נעדכן את ספירת השגיאות וניתן לפרמטר ההודעה את הערך DEFAULT_MSG

אם חלק ב' לא מומש והקלט הוא בקשת רמז פרמטר ההודעה למשתמש יהיה NO_HINTS_MSG

<u>בסיום המשחק:</u>
נקרא לפונקציה display_state כאשר:

1. ההודעה תהיה WIN_MSG או LOSS_MSG שמוגדרות בקובץ העזר כאשר השחקן הצליח לפענח את המילה או לא בהתאמה. במקרה של הפסד נשרשר להודעה את המילה.
2. בנוסף נעביר בקריאה את המשתנה ask_play עם הערך True על מנת שיוצג הכפתור עבור משחק חדש.

3. הגדירו את הפונקציה *main*() שאינה מקבלת ואינה מחזירה ערכים ומבצעת את הפעולות הבאות:
   1. טעינת קובץ המילים words.txt לתוך רשימה ע"י שימוש בפונקציה load_words
   2. הרצת המשחק (ע"י קריאה לפונקציה run_single_game שממשתם קודם לכן)
   3. בסיום כל משחק שואלים את המשתמש אם הוא מעוניין לשחק שוב. אם כן יתחיל משחק חדש. לשם כך יש להעזר בפונקציה get_input שבקובץ העזר.

על מנת להריץ את התכנית עליכם לקרוא לפונקציה *start_gui_and_call_main*(*main*) ואחריה לפונקציה *close_gui*() שבקובץ העזר, ע"י הוספת קטע הקוד הבא **בסוף הסקריפט**:

```
if __name__ == "__main__":
  hangman_helper.start_gui_and_call_main(main)
  hangman_helper.close_gui()
```

על מנת להבין לעומק את הקריאה ל - *main*() ניתן לקרוא את הלינק הבא:
https://docs.python.org/3/library/__main__.html

חלק ב

בחלק זה נבצע ניחוש מושכל לרמז שמשתמש יוכל לקבל. הבחירה של אות תהייה זו הנפוצה ביותר מתוך רשימת המילים הרלוונטיות לתבנית.

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 כל אלו שהן:
   a. באותו אורך של התבנית שהזין המשתמש
   b. שמכילות אותיות זהות בדיוק באותם מיקומים של האותיות הגלויות בתבנית ושאותיות אלו לא נמצאות במיקום אחר במילה המסוננת.
   c. לא מכילות אף אות המופיעה ברשימה הניחושים השגויים
   כלומר אם למשל התבנית הנוכחית היא '_ e _ _' ורשימת הניחושים השגויים מכילה את האותיות r ו t נסנן את הרשימה שבידינו לרשימה שמכילה רק מילים באורך ארבע, שהאות השנייה שלהן היא e ולא מופיעה e בשום מקום אחר, ובנוסף לא מופיעות בהן האותיות r ו t

2. ממשו את הפונקציה *choose_letter*(*words*, *pattern*) המקבלת כקלט רשימת מילים (שמתאימה לתבנית הנוכחית) ואת התבנית הנוכחית ומחזירה את האות שמופיעה הכי הרבה ברשימה. אות נספרת כמספר ההופעות שלה סה"כ ולא פעם אחת עבור כל מילה שהיא נמצאת בה. במידה ויש יותר מאות אחת שמופיעה מספר מקסימלי של פעמים ניתן להחזיר כל אות מתוך קבוצת האותיות עם מספר ההופעות המקסימלי. שימו לב שהאות שנבחרת אינה מופיעה כבר בתבנית.
   כדי למצוא את האות הנפוצה ביותר יש לספור את מספר הופעתן של האותיות השונות ברשימת המילים. ניתן להעזר (<u>אבל לא חייבים</u>) בפונקציות letter_to_index ו- index_to_letter שמופיעות בנספח בקובץ זה.
   לדוגמא עבור רשימת המילים grape, strawberry, tomato הפונקציה תחזיר את האות r.

   ניתן להניח כי רשימת המילים אינה ריקה.

3. עדכנו את הפונקציה run_single_game שמימשתם בחלק א:
   a. בקריאה ל- get_input נבדוק האם הערך שהוחזר הוא בקשה לרמז (ראו פירוט של הפונקציה בהמשך).
   b. במידה והמשתמש מעוניין ברמז, נקרא לפונקציה filter_words_list ואחריה לפונקציה choose_letter עם הרשימה המסוננת על מנת לבחור אות.
   c. נציג זאת למשתמש ע"י קריאה ל display_state כאשר ערך ההודעה יהיה HINT_MSG שבקובץ העזר שאליו נשרשר את האות שנבחרה כרמז.

רשימת הפונקציות הממומשות ב hangman_helper.py

1. *start_gui_and_call_main*(*main*) פונקציה שמקבלת כפרמטר את פונקציית main ומריצה אותו ואת הממשק הגרפי במקביל.
2. *load_words*() פונקציה שלא מקבלת קלט, ומחזיר את רשימת המילים המופיעות ב words.txt
   display_state(*pattern*, *error_count*, *wrong_guess_lst*, *msg*, *ask_play = False*) פונקציה זו מציגה את המצב הנוכחי: את התבנית, את הציור הרלוונטי של האיש התלוי (עפ"י המשתנה error_count). את רשימת הניחושים השגויים ואת ההודעה למשתמש. במידה ומועברים לפונקציה ערך שונה עבור המשתנה ask_play הממשק הגרפי יתעדכן בהתאם. הפונקציה אינה מחזירה ערך.
3. *get_random_word*(*word_list*) פונקציה מקבלת כקלט רשימת מילים ומחזירה אקראית מתוך רשימת המלים
4. *get_input*() הפונקציה מחזירה קלט מהמשתמש שהוזן דרך הממשק הגרפי. הקלט יכול להיות אות בקשה לרמז, או בקשה למשחק חדש.
   הפונקציה מחזירה זוג (tuple) כאשר האיבר הראשון הוא סוג הקלט, כלומר אחד מהמשתנים HINT, LETTER, או PLAY_AGAIN המוגדרים בקובץ. האיבר השני יהיה האות במקרה שהקלט היא אות, None אם האיבר הראשון הוא רמז ו - True/False אם האיבר הראשון הוא משחק חדש.
5. *close_gui*() הפונקציה סוגרת את הממשק הגרפי

רשימת המשתנים הגלובליים לשימושכם המוגדרים ב - hangman_helper.py:

```
1. MAX_ERRORS = 6
2. WIN_MSG = 'Correct guess, this is the word!!!'
3. LOSS_MSG = 'You have run out of guesses, the word was: '
4. ALREADY_CHOSEN_MSG = 'You have already chosen '
5. NON_VALID_MSG = 'Please enter a valid letter'
6. HINT_MSG = 'Consider choosing: '
7. NO_HINTS_MSG = 'Hints not supported'
8. DEFAULT_MSG = ''
9. HINT = 1
10. LETTER = 2
11. PLAY_AGAIN = 3
```

*התכנית אמורה לפעול בצורה נכונה **גם במידה** וערכי הקבועים הגלובליים מוחלפים.*

הגשת התרגיל:

עליכם להגיש קובץ zip הנקרא ex4.zip ומכיל את הקבצים הבאים:

1. hangman.py

2. README

שימו לב! גם בתרגיל זה ישנן בדיקות אוטומטיות. חלק מהבדיקות יפעלו גם בזמן הגשה - **חובה עליכם** לוודא כי הפריסבמיט חוזר תקין ושאין בו שגיאות. להזכירכם, תרגילים שנכשלים בפריסבמיט יקבלו 0 בבדיקה האוטומטית.

בהצלחה!

נספח - פונקציות שיכולות לעזור אך ממש לא חובה להשתמש בהן:

1. *letter_to_index*(*index*) הפונקציה מקבלת כקלט אות ומחזירה את האינדקס האלפאביתי שלה. לדוגמא עבור הקלט 'a' היא תחזיר 0, עבור הקלט 'b' היא תחזיר 1 עבור 'c' תחזיר 2 וכן הלאה. שימו לב שמניחים שהקלט תקין, כלומר זוהי אות מבין ל a - z.
2. *index_to_letter*(*letter*) הפונקציה פועלת הפוך מ letter_to_index, כלומר הפונקציה תחזיר את האות במיקום האלפאבתי של האינדקס הנתון. לדוגמא עבור הקלט 0 תחזיר 'a', עבור הקלט 'b' תחזיר 1 וכן הלאה. שימו לב שגם כאם מניחים שהקלט תקיות כלומר מספר בין 0 ל 25.

```
CHAR_A = 97

def letter_to_index(letter):
    """
    Return the index of the given letter in an alphabet list.
    """
    return ord(letter.lower()) - CHAR_A

def index_to_letter(index):
    """
    Return the letter corresponding to the given index.
    """
    return chr(index + CHAR_A)
```