מבוא למדעי המחשב 67101
תרגיל 4 – רצפים ולולאות
להגשה בתאריך 18/11/2015 בשעה 22:00

בתרגיל זה נממש את משחק ה"איש 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 הפעמים שבהן 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 אך ורק מאותיות שהן lower case
2. בתבנית אותיות שאינן גלויות ייוצגו ע"י התו _ (קו תחתון)
3. התבנית, המילה והאותיות מיוצגים כמחרוזות
4. על מנת שהממשק הגרפי יציג את התמונות כראוי, נעשה שימוש בסיפרייה PIL של פייתון. הספרייה מותקנת בחוות המחשבים באוניברסיטה, וניתן להתקין אותה בבית: http://pillow.readthedocs.org/en/latest/installation.html (הסיפרייה כלולה ב WinPython אשר ניתן להורדה כאן)

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_'
```

(בדוגמא לעיל ישנם רווחים בין הקווים התחתוניים, אך זה לצורך ההדגמה. בפועל במימוש שלכם אין רווחים בתבנית בין תו לתו).

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

<u>איתחול 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, התבנית ההתחלתית ריקה ואורכה כאורך המילה.
3. בנוסף, עליכם להגדיר משתנה של הודעה למשתמש ולתת לו את הערך של המשתנה DEFAULT_MSG שבקובץ העזר.

<u>מהלך המשחק:</u>

כל עוד לא הסתיים המשחק נבצע איטראציה (סבב) נוספת של המשחק. המשחק לא יסתיים כל עוד התבנית לא נחשפה במלואה ומספר הניחושים השגויים קטן מזה שמוגדר במשתנה MAX_ERRORS בקובץ העזר. שימו לב שברשימת הניחושים השגויים אין חזרות.

בכל איטראציה 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 אות נבצע את הפעולות הבאות:
   i. אם הקלט אינו תקין, כלומר אורכו יותר מאחד או שאינו אות, או שאינו אות קטנה (lowercase) ניתן למשתנה ההודעה את הערך NON_VALID_MSG וממשיכים לחכות לקלט הבא
   ii. אחרת, אם האות שנבחרה כבר נבחרה בעבר, ניתן למשתנה ההודעה את הערך ALREADY_CHOSEN_MSG שבקובץ העזר ונשרשר למחרוזת הזו את האות שנבחרה (בחירה זו לא תחשב כטעות נוספת)
   iii. אחרת, אם האות שנבחרה מופיעה במילה, יש לעדכן את התבנית ע"י קריאה לפונקציה **update_word_pattern** שמימשתם קודם ולתת למשתנה ההודעה את הערך DEFAULT_MSG
   iv. אחרת האות שנבחרה לא מופיעה במילה, לכן נעדכן את רשימת הניחושים השגויים במידה ויש צורך, נעדכן את ספירת השגיאות וניתן למשתנה ההודעה את הערך DEFAULT_MSG

<u>בסיום המשחק:</u>

נקרא לפונקציה **display_state** כאשר:

1. ההודעה תהייה 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 פונקציית **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 האם הוא מעוניין לשחק שוב. אם כן יתחיל משחק חדש. לשם כך יש להעזר בפונקציה **get_input** שבקובץ העזר.

על מנת להריץ את התוכנית 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 לmain: https://docs.python.org/3/library/__main__.html

<u>חלק ב</u>

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

1. ממשו את הפונקציה **filter_words_list**(words, pattern, wrong_guess_lst) המקבלת כקלט את רשימת המילים, התבנית הנוכחית ורשימת הניחושים השגויים, ומחזירה רשימה חדשה שמכילה רק חלק מהמילים שברשימת הקלט שיכולות להתאים לתבנית ולניחושים הקודמים.

מתוך רשימת כל המילים נסנן את כל אלו שהן:

1. באותו אורך של התבנית שהזין המשתמש
2. שמכילות אותיות זהות בדיוק באותם מיקומים של האותיות הגלויות בתבנית
3. 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. 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.

3. עדכנו את הפונקציה **run_single_game** שמימשתם בחלק א:
   i. בקריאה ל **get_input** נבדוק האם הערך שהוחזר הוא בקשה לרמז (ראו פירוט של הפונקציה בהמשך).
   ii. במידה והמשתמש מעוניין ברמז, נקרא לפונקציה **filter_words_list** ואחריה לפונקציה **choose_letter** עם הרשימה המסוננת על מנת לבחור אות.

iii. 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 אותה ואת הממשק הגרפי במקביל.

2. **load_words**() פונקציה שלא מקבלת קלט, ומחזיר את רשימת המילים המופיעות ב words.txt

3. **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 ניחושים השגויים ואת ההודעה למשתמש. במידה ומועברים לפונקציה ערך שונה עבור המשתנה ask_play הממשק הגרפי יתעדכן בהתאם. הפונקציה אינה מחזירה ערך.

1. **get_random_word**(words_list) הפונקציה מקבלת כקלט רשימת מילים ומחזירה מילה אקראית מתוך הרשימה
2. **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 היה אות, ו True במקרים שבהם הקלט הוא בקשה לרמז או למשחק חדש.
3. **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. DEFAULT_MSG = ''
8. HINT = 1
9. LETTER = 2
10. PLAY_AGAIN = 3
```

הגשת התרגיל:

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

1. hangman.py
2. README

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 a ל z.
2. **index_to_letter**(letter) הפונקציה פועלת הפוך מ **letter_to_index**, כלומר הפונקציה תחזיר את האות במיקום האלפאביתי של האינדקס הנתון. לדוגמא עבור הקלט 0 תחזיר 'a', עבור הקלט 'b' תחזיר 1 וכן הלאה. שימו לב שגם כאן מניחים שהקלט תקין, כלומר מספר בין 0 ל 25.

```
__INIT_ALPHA_ORDER__ = 97

def letter_to_index(letter):
    '''
    Returns the index of the given letter in an alphabet list
    '''
    return ord(letter.lower()) - __INIT_ALPHA_ORDER__

def index_to_letter(index):
    '''
    Returns the index of the given letter in an alphabet list
    '''
    return chr(index + __INIT_ALPHA_ORDER__)
```