

מבוא למדעי המחשב 67101

תרגיל 4 – Hangman

להגשה בתאריך 14.11.18 בשעה 22:00

הקדמה

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_\(game\)](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 הַ (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:

<https://pillow.readthedocs.io/en/latest/installation.html>

(הספרייה כלולה ב-WinPython אשר ניתן להוריד מכאן: <http://winpython.github.io/>)

ניתן להריץ את פתרון בית הספר ע"י הרצת הפקודה: `intro2cs1/bin/ex4/hangman` במחשבי בית הספר.

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 להיעזר (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', '___!_', 'p')
```

תחזיר:

```
'_ppl_'
```

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

אתחול המשחק:

- 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 האותיות אינן גליות.
- איתחול משתנה ששומר את ערך ההודעה למשתמש להיות `DEFAULT_MSG` (מוגדר בקובץ העזר).

מהלך המשחק:

כל עוד 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.

בכל איטרציה של המשחק:

- נציג את המצב הנוכחי ע"י קריאה ל- `display_state` הממומשת ב- `hangman_helper.py`.
- 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 השחקן נבצע את הפעולות הבאות:
 - אם הקלט אינו תקין, כלומר אורכו שונה מאחד או שאינו אות, או שאינו אות קטנה (lowercase) ניתן לפרמטר ההודעה את הערך `NON_VALID_MSG` ונמשיך לחכות לקלט הבא.
 - אחרת, אם האות שנבחרה כבר נבחרה בעבר, ניתן לפרמטר ההודעה את הערך `ALREADY_CHOSEN_MSG` שבקובץ העזר ונשרש למחרוזת הזו את האות שנבחרה (בחירה זו לא תחשב כאות נוספת).
 - 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`.
 - אחרת, האות שנבחרה לא מופיעה במילה, לכן נעדכן את רשימת הניחושים השגויים, נעדכן את ספירת השגיאות וניתן לפרמטר ההודעה את הערך `DEFAULT_MSG`.

הערה: אם חלק ב' לא מומש והקלט הוא בקשת רמז, פרמטר ההודעה למשתמש יהיה `NO_HINTS_MSG`

בסיום המשחק:

נקרא לפונקציה `display_state` כאשר:

- ההודעה תהיה `WIN_MSG` או `LOSS_MSG` שמוגדרות בקובץ העזר כאשר השחקן הצליח לפענח את המילה או לא בהתאמה. במקרה של הפסד, נשרש להודעה את המילה.
- בנוסף נעביר בקריאה את הערך `True` לפרמטר `ask_play` על מנת שיוצג הכפתור עבור משחק חדש
- שליחת הערך `True` לפרמטר `ask_play` מתבצעת על מנת שתוצג השאלה האם להתחיל משחק חדש, שימו לב שבדיקת התשובה עבור שאלה זו צריכה להתבצע בפונקציה הבאה – `main()`.
- שימו לב שהפונקציה `run_single_game` אחראית על ריצה של משחק אחד (כפי ששמה מרמז).

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`.
- הרצת המשחק (ע"י קריאה לפונקציה `run_single_game` שממשתם קודם לכן).
- בסיום כל משחק שואלים את המשתמש אם הוא מעוניין לשחק שוב. אם כן, יתחיל משחק חדש. לשם כך יש להיעזר 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 תהיה 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:

- באותו אורך של התבנית שהזין המשתמש.
- שמכילות אותיות זהות בדיוק באותם מיקומים של האותיות הגליות בתבנית ושאותיות אלו לא נמצאות במיקום אחר במילה המסוננת.
- 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 השלישית 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 (אם גם האות `a` וגם האות `s` מופיעות 10 פעמים, ו-10 זה מספר ההופעות המקסימלי, אז גם `a` וגם `s` תתקבלנה כתשובה נכונה). **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 להיעזר (אבל לא חייבים) 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.

חלק ג'

1. כתבו פונקציה לבדיקת הפונקציה `update_word_pattern`:
- עליכם לממש פונקציה הבודקת את פונקציה `update_word_pattern` שמימשתם בחלק א'.
 - את הפונקציה יש לממש בקובץ `check_update_word_pattern.py`.
 - שם הפונקציה נתון לבחירתכם.
 - עליכם לחשוב על ארבעה קלטים שונים (נסו לחשוב על מקרי קצה) עבור הפונקציה `update_word_pattern`, ועל הפלטים של כל אחד מהקלטים שבחרתם.
- פונקציה זו תריץ את הפונקציה `update_word_pattern` על הקלטים שבחרתם ותבדוק את הפלטים שמתקבלים.
- פונקציה זו תחזיר את הערך `True` אם פלט הפונקציה `update_word_pattern` היה הפלט המצופה עבור כל הבדיקות שבחרתם, ו-`False` אחרת.
- בנוסף הפונקציה תדפיס הודעה אינפורמטיבית על תוצאת הבדיקה:
- לדוגמא, `update_word_pattern` test success אם הבדיקה עברה בהצלחה ו-`update_word_pattern` test fail אם הבדיקה נכשלה.
- לדוגמא, אם בחרנו את אחד מהקלטים לבדיקה להיות הקלט הבא: המילה: `'banana'`, התבנית: `'_a_a_a'` והאות: `'n'`. נצפה שהפונקציה `update_word_pattern` שכתבתם בחלק א' תחזיר את הערך `'_anana'`, ולכן נשאל האם:
`update_word_pattern('banana', '_a_a_a', 'n') == '_anana'`
 - אין צורך לבדוק מקרים של קלט לא תקין.
 - הפונקציה לא מקבלת פרמטרים ומחזירה ערך בוליאני כפי שתואר לעיל.
 - עליכם לכתוב ב- README (תחת הכותרת C – q1) למה בחרתם את הקלטים שבחרתם (מה מטרת הבדיקה עבור כל קלט).
 - בסוף הקובץ `check_update_word_pattern.py` עליכם להוסיף את השורה הבאה (עליה נרחיב בהמשך הקורס):

```
if __name__ == "__main__":
```

```
    (לא מומלץ להעתיק את השורה מקובץ זה, אלא להקלידה בעצמכם)  
    ומתחת לשורה זו, בהזחה של טאב אחד, עליכם לקרוא לפונקציה של שאלה זו.
```

```
כלומר, אם שם הפונקציה היא foo אז סוף הקובץ צריך להיראות כך:
```

```
if __name__ == "__main__":  
    foo()
```

2. על השאלה הבאה יש לענות בקובץ README (תחת הכותרת: section C – q2):

מה הייתם צריכים לשנות בתוכנית שלכם על מנת לשחק את המשחק עם רשימת מילים בעברית, ואותיות עבריות? אין צורך לפרט שינויים שהיו נדרשים בקובץ העזר.

רשימת הפונקציות הממומשות ב- hangman_helper.py

1. `start_gui_and_call_main(main)` פונקציה שמקבלת את פונקציית `main` ומריצה אותו 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 הניחושים השגויים ואת ההודעה למשתמש. במידה ומועבר לפונקציה ערך `ask_play` המשתנה `ask_play` הממשק הגרפי יתעדכן בהתאם. הפונקציה אינה מחזירה ערך.
4. `get_random_word(words_list)` פונקציה המקבלת כקלט רשימת מילים ומחזירה אקראית מתוך הרשימה מילה.
5. `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.
6. `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. check_update_word_pattern.py
3. README

בהצלחה! :)

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

1. `letter_to_index(index)` הפונקציה מקבלת כקלט אות ומחזירה את האינדקס האלפביתי שלה. לדוגמא, עבור הקלט 'a' היא תחזיר 0, עבור הקלט 'b' היא תחזיר 1, עבור הקלט 'c' היא תחזיר 2, וכן הלאה. שימו לב שמניחים שהקלט תקין, כלומר זוהי אות מבין a-z.
2. `Index_to_letter(letter)` הפונקציה פועלת הפוך מ-`letter_to_index`, כלומר הפונקציה תחזיר את האות במיקום האלפאבטי של האינדקס הנתון. לדוגמא, עבור הקלט 0 תחזיר 'a', עבור הקלט 1 תחזיר 'b', וכן הלאה. שימו לב שגם כאם מניחים שהקלט תקין 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)
```