

הקדמה

בתרגיל זה תממשו את המשחק [Boggle](#) (כללי המשחק בתרגיל חורגים במעט מוויקיפדיה). את תרגיל זה **חובה** להגיש בזוגות. על כל זוג להגיש את התרגיל ממשתמש אחד בלבד! יש להגיש גם את קבצי העזר.

חוקי המשחק Boggle

לוח המשחק:


בבוגל, לוח 4X4 של קוביות, כך שעל כל קוביה רשומה אות. בתחילת המשחק, מטילים כל אחת מהקוביות וכך מקבלים לוח אקראי של 4X4, שבו אותיות.

מטרת המשחק:

היא למצוא כמה שיותר מילים על הלוח בפרק זמן מסוים. ישנו רק שחקן אחד במשחק.

מילה חוקית:

היא מילה המופיעה במילון והמורכבת מטיול על לוח המשחק המתחיל באחת האותיות ועובר לאותיות שכנות. אות שכנה תחשב לאות המופיעה צמוד לאות הנוכחית באחד מ-8 הכיוונים (למעלה, למטה, ימינה, שמאלה או אחד מארבעת האלכסונים). מותר להשתמש באותה הקובייה בשתי מילים שונות, אבל אין להשתמש פעמיים באותה הקובייה במהלך אותה המילה.

בדומה לחוקים הרשמיים, קיימת אות שבנויה משתי אותיות QU. היא יכולה להופיע על קובייה כמו כל אות אחרת אך בבניית המילה היא כוללת את שתי האותיות QU ברצף. 

הניקוד הכולל של השחקן:

כל מילה מזכה בניקוד ריבועי באורך המילה. כלומר, מילה באורך n שמופיעה במילון תזכה ב n^2 נקודות.

אי אפשר לקבל ניקוד פעמיים על אותה המילה, גם אם היא מופיעה מספר פעמים על הלוח.

זמן המשחק:

לשחקן יש 3 דקות להשיג כמה שיותר נקודות.

דוגמאות למילים חוקיות

הדוגמה הבאה ממחישה את הכתוב לעיל (האותיות עצמם נבחרו לשם המחשה בלבד ולא בעזרת קובץ העזר):

D	C	B	A
G	A	D	E
T	J	Y	T
N	M	F	I

בטבלה הנ"ל המילה BED היא חוקית היא מתחילה באות B בתא (0,2) וממשיכה למטה וימינה לאות E בתא (1,3) ומסתיימת באות D בתא (1,2). המילה תזכה ב-9 נקודות. בנוסף המילה FIT גם היא חוקית. היא מתחילה באות F בתא (3,2), ממשיכה לאות I בתא (3,3) ומסתיימת באות T בתא (2,3).

פונקציות למימוש

עליכם לממש את הפונקציות הבאות בקובץ ששמו `ex12_utils.py`. קובץ שלד עם חתימות הפונקציות מסופק לכם כחלק מהתרגיל. ניתן להוסיף לקובץ פונקציות עזר. מותר לייבא את הקובץ `ex12_utils.py` ולהשתמש בפונקציות ממנו בשאר הקבצים בפרויקט, אך אין חובה לעשות זאת.

`:load_words_dict(file_path)`

הפונקציה מקבלת שם של קובץ המכיל מילים עבור המשחק. הפורמט של הקובץ זהה לפורמט של קובץ המילים המסופק לכם, `boggle_dict.txt`.

הפונקציה מחזירה מילון שהמפתחות (`keys`) שלו הם כל המילים מהקובץ `boggle_dict.txt`. הערכים (`values`) של המילים הם תמיד `True`.

לדוגמא, אם קובץ המילים מכיל את שלושת המילים 'apple', 'orange', 'pear', המילון שיוחזר הוא:

`{ 'apple': True, 'orange': True, 'pear': True }`

(נקודה למחשבה: מה היתרונות ומה החסרונות של שמירת המילים במילון בהשוואה לשמירתן ברשימה? מתי תרצו להשתמש במילון שהפונקציה מחזירה?)

:is_valid_path(board, path, words)

הפונקציה מקבלת את הפרמטרים הבאים:

- board – לוח המשחק, בפורמט שמתקבל מהפונקציה randomize_board בקובץ boggle_board_randomizer.py המסופק לכם.
- path – מסלול על לוח המשחק. המסלול בנוי מרשימה של tuples. כל tuple מכיל שני איברים המייצגים את השורה בלוח שמתאימה לקוביה ואת העמודה בלוח שמתאימה לקוביה, בסדר זה. לדוגמא, המסלול [(3, 1), (3, 2), (2, 1)] מתחיל בקוביה הממוקמת בשורה 3 בעמודה 1, עובר לקוביה שממוקמת בשורה 3 בעמודה 2 ומסתיים בקוביה הממוקמת בשורה 2 בעמודה 1. שימו לב שהפינה השמאלית העליונה של הלוח היא באינדקס (0,0).

- words – מילון המכיל את כל המילים החוקיות במשחק, בפורמט המוגדר בפונקציה load_words_dict.

הפונקציה בודקת אם המסלול הוא מסלול חוקי שמייצג מילה הקיימת במילון. אם כן, הפונקציה מחזירה את המילה שמייצג המסלול. אם המסלול אינו חוקי או שהמילה המתאימה לו אינה קיימת במילון, הפונקציה תחזיר None.

:find_length_n_words(n, board, words)

הפונקציה מקבלת את הפרמטרים הבאים:

- n - מספר טבעי בין 3 ל-16 שמייצג את אורך המילים שיש למצוא
 - board - ייצוג של לוח המשחק, בפורמט שמתקבל מהפונקציה randomize_board בקובץ boggle_board_randomizer.py המסופק לכם.
 - words – מילון המכיל את כל המילים החוקיות במשחק, בפורמט המוגדר בפונקציה load_words_dict.
- הפונקציה מחזירה רשימה של זוגות (כל זוג הוא tuple) של מילים ומסלולים. האיבר הראשון בכל זוג הוא מחרוזת שמייצגת מילה. האיבר השני הוא רשימה שמתארת את המסלול שמתאים למילה הזאת. למשל, הזוג שמתאים למילה BED בלוח לדוגמא (שנמצא תחת הסעיף "דוגמאות למילים חוקיות") הוא ((0,2),(1,3),(1,2)).
- הרשימה שהפונקציה מחזירה צריכה לכלול איבר אחד עבור כל מילה חוקית באורך n שקיימת על הלוח. למשל, הקריאה לפונקציה עם n=4 תחזיר את הזוגות שמתאימים לכל המילים החוקיות באורך 4 הקיימות בלוח.

אלמנטים חזותיים

המשימה העיקרית הנה ליצור את כל רכיבי ה-UI המרכיבים את המשחק בשימוש במודול tkinter. המשחק שלכם חייב להיות נוח ואינטואיטיבי לשימוש ע"י השחקן. בפרט, עליכם לקיים את הדרישות הבאות:

1. המשחק צריך לאפשר בקלות לבחור את המילה בעזרת העכבר (אין לעשות שימוש בהקלדה לשם כך).
2. לוח המשחק צריך להראות באופן ברור את המילה הנוכחית לפי הקוביות שהשתמש בחר. סדר האותיות צריך להיות לפי סדר בחירת הקוביות.
3. כשהתוכנית מתחילה, השעון לא מתחיל מיד אלא רק כשהשחקן בחר להתחיל (והלוח גם לא גלוי לו עד שהשעון מתחיל, כדי שהשחקן לא יחפש מילים ללא אובדן זמן בשעון). יש להציג בבירור את הזמן הנותר.
4. בסיום המשחק התוכנית לא תסתיים, אלא תציע לשחקן לשחק שנית.
5. יש להציג את הניקוד שצובר השחקן באופן ברור וכן את רשימת המילים שכבר מצא.
6. על התוכנית לעשות שימוש ברכיבי GUI בלבד ואין להדפיס בעזרת print או לשמור קבצים.

העיצוב החזותי והממשק המדויק נתונים לשיקולכם.

בנוסף של עד 5 נק' ינתן למי שיממש ממשק יפה במיוחד, או יוסיף פונקציונליות מיוחדת לממשק (לשיקול דעתו של הבודק). לשם כך, הגישו בנוסף קובץ README המפרט את התוספות האישיות שלכם.

פרטיים טכניים

פורמט ההרצה של המשחק:

הפקודה `python3 boggle.py` תריץ את המשחק.

קבצי עזר:

- `boggle_dict.txt` – קובץ עם מילים חוקיות שעליכם לטעון למשחק. הקובץ יימצא באותה התיקייה עם המימוש שלכם. אין לשנותו אך יש להגישו.
- `boggle_board_randomizer.py` – קוד בפייטון שמגריל לוח. יש לייבא אותו ולקרוא לפונקציה `randomize_board` שאינה מקבלת פרמטרים. אין לשנותו ואתם חייבים להגריל לוחות דרכו (שימו לב לאות המיוחדת QU). הלוח המוגרל ניתן כרשימה דו מימדית. לשם הבהרה, הקואורדינטה (0,0) נמצאת בפינה

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

השמאלית העליונה של הלוח, בשורה שמתחתיה הקואורדינטה השמאלית ביותר היא (1,0). הקובץ יימצא באותה תיקייה עם המימוש שלכם. אין לשנותו אך יש להגישו.

אין צורך לוודא את התקינות של המילים בקובץ המילון. כמו כן, אין צורך להתמודד עם מקרה בו הקובץ ריק. אין צורך לוודא את התקינות של הלוח שמחזירה הפונקציה `randomize_board`. אין להסתמך על כך שהלוח שתקבלו תמיד מיוצר באמצעות הקוביות שנתונות לכם. בפרט, אי אפשר להניח שQU הוא צמד האותיות היחיד שמופיע כזוג – ייתכן שהקוד ייבדק עם קוביות בהן יש צירופים כפולים נוספים.

ייבוא חבילות של פייתון:

מותר לייבא כל חבילה של פייתון שמותקנת על מחשבי בית הספר. **חובתכם לוודא שהקוד שלכם רץ באופן תקין על מחשבי בית הספר** לפני ההגשה.

אם אתם לא יודעים אם חבילה מסוימת מותקנת על מחשבי בית הספר, תוכלו להיכנס לטרמינל ולהקליד את הפקודה:
`python3 -c "import <module name>"`

אם הספרייה אינה מותקנת, תודפס שגיאה מסוג `ModuleNotFoundError` למסך הטרמינל.

למשל: `python3 -c "import numpy"`.

דגשים למימוש

יתקיימו ראיונות על התרגיל אותו מימשתם, לכן עליכם לשים לב לעקרונות התכנות שנלמדו בקורס. נסו להפריד את החלקים הגרפיים של המשחק מהחלקיים הלוגיים. לדוגמה מומלץ שהלוח עצמו יהיה נפרד מהמשחק - עליו להיות אחראי על הרכיבים הגרפיים בלבד בעוד המשחק צריך להיות אחראי על הלוגיקה (כמו למשל ספירת הנקודות).

באופן כללי, כדאי קודם ליצור את החלקים הלוגיים, לבדוק אותם ולאחר מכן לעבור למימוש החלקים הגרפיים. בנוסף, נסו לפרק את הרכיבים הגרפיים למחלקות שונות, עם ממשק (API) פשוט לכל מחלקה שפונה החוצה למחלקות האחרות.

נהלי הגשה

את תרגיל זה כאמור חייבים להגיש בזוגות. לפני ההגשה ותחת הלינק המיועד להגשה, עליכם לפתוח קבוצה ב-moodle. אחד השותפים ייצור את הקבוצה על ידי

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

הזנת שם השותף השני (שימו לב להוסיף את השם בדיוק כפי שהוא מופיע ב-moodle, כולל אותיות גדולות וקטנות במקומות הנכונים). השותף (שלא יצר את הקבוצה) יוכל לראות שרשמו אותו על ידי כניסה לקישור ההגשה וצפייה בשם בן הזוג. כדאי לרשום את בן הזוג בשלב מוקדם (אין צורך להגיש את התרגיל בפועל על מנת להירשם כזוג). כמו כן, וודאו כי הקבוצה אינה מכילה יותר משני שותפים. עליכם להגיש קובץ נוסף בשם **AUTHORS (ללא כל סיומת)**. קובץ זה יכיל שורה אחת ובו הלוגינים (CSE login) של שני הסטודנטים המגישים, מופרד ע"י פסיק. כך:

minniemouse,mickeymouse

ודאו כי הגשתם קובץ **AUTHORS** תקין! אי הגשה של קובץ **AUTHORS** תקין תגרור הורדה בציון.

במידה ואתם מגישים קובץ **README** גם הוא צריך להיות **ללא כל סיומת**.

עליכם להגיש את הקבצים הבאים:

1. boggle.py
2. ex12_utils.py
3. boggle_dict.txt - ללא שינוי
4. boggle_board_randomizer.py - ללא שינוי
5. README (בכדי לקבל את הבונוס)
6. AUTHORS
7. קבצים נוספים הנחוצים להרצת התוכנית אם כתבתם כאלה

בהצלחה!