

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101 – סמסטר א' 2022/23

תרגיל 11 – Boggle

להגשה בתאריך 24/01/2023 בשעה 22:00

שימו לב שבשונה משאר התרגילים- ההגשה היא ביום שלישי!

הקדמה

בתרגיל זה תממשו את המשחק [Boggle](#) (כללי המשחק בתרגיל חורגים במעט מוויקיפדיה). את התרגיל חובה להגיש בזוגות. על כל זוג להגיש את התרגיל ממשמש אחד בלבד! (הסבר על הגשה בזוגות מופיע ב"נהלי הגשה" שבסוף מסמך זה). **יש להגיש גם את קבצי העזר.** תרגיל זה הוא תרגיל כפול- כלומר משקלו בממוצע התרגילים יהיה כפול ממשקל התרגילים האחרים, בהתאם למפורט בנהלי הקורס.

חוקי המשחק Boggle

לוח המשחק:

ב-Boggle, לוח 4×4 של קוביות, שעל כל קובייה רשומה אות. נגדיר מיקום של אות על הלוח על-ידי זוג אינדקסים (x, y) , כך ש- y מציינ את מספר השורה בה האות נמצאת על הלוח, ו- x את מספר העמודה. האות הנמצאת בפינה השמאלית-עליונה של הלוח תמוקם במיקום $(0, 0)$, ואילו האות בפינה הימנית-תחתונה של הלוח תמוקם במיקום $(3, 3)$. בתחילת המשחק, מגרילים לוח אקראי בגודל 4×4 , שבו אותיות.

מהלך המשחק:

במשחק ישנו שחקן אחד בלבד, ומרגע תחילת המשחק יש לשחקן 3 דקות להשיג כמה שיותר נקודות, על-ידי מציאת מילים חוקיות על הלוח.

מילה חוקית:

היא מילה המופיעה במילון ומורכבת מטיוול 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 עבור שתי מילים שונות, אבל **אין להשתמש פעמיים באותה הקובייה עבור אותה המילה.**

בדומה לחוקים הרשמיים, קיים זוג מיוחד של אותיות **QU**, שנתייחס אליו בתור אות **Q**. הוא יכול להופיע על קובייה כמו כל אות אחרת, אך בבניית המילה, אם נבחר את הקובייה שמכילה אותו- המילה אותה נרכיב תכלול את שתי האותיות **QU** ברצף.

הניקוד הכולל של השחקן:

כל מילה מזכה בניקוד ריבועי באורך המסלול. כלומר, מילה שמופיעה במילון שנמצאת על מסלול באורך n תזכה ב n^2 נקודות. אי אפשר לקבל ניקוד פעמיים על אותה המילה, גם אם היא מופיעה מספר פעמים על הלוח.

דוגמאות למילים חוקיות

הדוגמה הבאה ממחישה את הכתוב לעיל (האותיות עצמן 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 זו, המילה BED היא מילה חוקית והיא מתחילה באות B בתא (0,2), ממשיכה 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

במהלך התרגיל נשתמש בהגדרות הבאות:

- לוח משחק** - רשימה של רשימות של מחרוזות בפורמט שמתקבל מהפונקציה `randomize_board` בקובץ `boggle_board_randomizer.py` המסופק לכם.
 שימו לב! הלוח לא חייב להיות לוח שיכול להתקבל על ידי הפונקציה הנ"ל עם הקוביות הסטנדרטיות.
- מסלול על לוח המשחק** - מסלול בנוי מרשימה של `tuples`. כל `tuple` מכיל שני איברים המייצגים את השורה בלוח שמתאימה לקובייה ואת העמודה בלוח שמתאימה לקובייה, בסדר זה. לדוגמה, המסלול [(3,1),(3,2),(2,1)] מתחיל בקובייה הממוקמת בשורה 3 בעמודה 1, ממשיך לקובייה שממוקמת בשורה 3 בעמודה 2 ומסתיים בקובייה הממוקמת בשורה 2 בעמודה 1. שימו לב שהמיקום של הפינה השמאלית העליונה של הלוח הוא (0,0).
- אוסף מילים** - אובייקט המכיל את כל המילים החוקיות במשחק. אוסף זה יכול להיות רשימה, `tuple`, מילון, או כל אובייקט אחר שאפשר לעשות עליו איטרציה. ניתן להשתמש בפעולת `in` לבדוק הכלה. אורך מילה הוא מספר שלם חיובי.

עליכם לממש את הפונקציות הבאות בקובץ ששמו `ex11_utils.py`.

קובץ שלד עם חתימות הפונקציות מסופק לכם כחלק מהתרגיל. ניתן להוסיף לקובץ פונקציות עזר.

מותר לייבא את הקובץ `ex11_utils.py` ולהשתמש בפונקציות ממנו בשאר הקבצים בפרויקט, אך אין חובה לעשות זאת.


`is_valid_path(board, path, words):`

הפונקציה מקבלת את הפרמטרים הבאים:

- **board** - לוח משחק.
- **path** - מסלול על לוח המשחק.
- **words** - אוסף מילים.

הפונקציה בודקת אם המסלול הוא מסלול חוקי המתאר מילה הקיימת באוסף המילים. אם כן, הפונקציה מחזירה את

המילה שנמצאה. אם המסלול אינו חוקי או שהמילה המתאימה לו אינה קיימת במילון, הפונקציה תחזיר `None`.

`find_length_n_paths(n, board, words):` 

הפונקציה מקבלת את הפרמטרים הבאים:

- **n** - מספר שלם חיובי. מייצג את אורך המסלולים אותם יש למצוא.
- **board** - לוח משחק.
- **words** - אוסף מילים.

הפונקציה מחזירה רשימה של כל המסלולים החוקיים באורך n המתארים מילים באוסף המילים. אם יש מספר מסלולים

חוקיים באורך n שמתארים את אותה המילה, יש להחזיר את כולם.

`find_length_n_words(n, board, words):`

הפונקציה מקבלת את הפרמטרים הבאים:

- **n** - מספר שלם חיובי. מייצג את אורך המילים אותן יש למצוא.
- **board** - לוח משחק.
- **words** - אוסף מילים.

הפונקציה מחזירה רשימה של כל המסלולים החוקיים המתארים מילים באוסף המילים שהן באורך n. אם יש מספר

מסלולים חוקיים לאותה מילה, יש להחזיר את כולם.

`max_score_paths(board, words):`

הפונקציה מקבלת את הפרמטרים הבאים:

- **board** - לוח משחק.
- **words** - אוסף מילים.

הפונקציה מחזירה רשימה של מסלולים חוקיים המספקים את הניקוד המירבי למשחק עבור הלוח ואוסף המילים

שניתנים. שימו לב שאין לכלול יותר ממסלול אחד עבור אותה מילה ושיכולים להיות מסלולים בעלי ניקוד שונה עבור אותה

מילה. 

הערות כלליות:

- אין לשנות את הקלט בכל הפונקציות הללו.
- שימו לב ליעילות. כל הפונקציות אמורות לרוץ מהר על לוחות סטנדרטיים. חישבו עבור אילו לוחות היעילות תרד.
- יש תכונות משותפות בין הפונקציות שתוארו, למרות שלא בטוח שכדאי להשתמש בהן כפונקציות עזר באופן ישיר. הקפידו להימנע מכפל קוד.

אלמנטים חזותיים

- המשימה העיקרית בתרגיל זה היא ליצור את כל רכיבי ה-GUI המרכיבים את המשחק בעזרת שימוש 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 שנותר.
 4. בסיום המשחק התוכנית לא תסתיים, אלא תציע לשחקן לשחק משחק נוסף.
 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' יינתן 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.

פרטים 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.

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

- `boggle_board_randomizer.py` – מכיל פונקציה שמגרילה לוח משחק. יש לייבא אותו ולקרוא לפונקציה `randomize_board` ללא פרמטרים. **אין לשנות את הקובץ** ואתם חייבים להגריל לוחות באמצעות פונקציה זו **(שימו לב לאות המיוחדת QU)**. הלוח המוגרל ניתן כרשימה דו מימדית. לשם הבהרה, הקואורדינטה (0,0) נמצאת בפינה 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 ש-QU הוא צמד האותיות היחיד שמופיע כזוג** – ייתכן שהקוד ייבדק עם קוביות בהן יש צירופים כפולים נוספים (ויותר מקובייה אחת, עם יותר מאות אחת).

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

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.

ראיונות

- על תרגיל זה ייערכו ראיונות עם מדריכי המעבדה. בראיונות תתבקשו בין היתר להסביר את השיקולים שהיו לכם בעת כתיבת המחלקות השונות והממשקים ביניהן, את ההחלטות שקיבלתם באשר לקביעת המקום בקוד בו יופיעו הפעולות השונות, וכו'.
- כחלק מהראיון יינתן ציון לסגנון התכנות שלכם (תיעוד, שמות משתנים, פונקציות ומתודות אלגנטיות, וכד')

נהלי הגשה

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

וודאו 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 קובץ בשם **ex11.zip** (בלבד), והוא צריך לכלול את הקבצים הבאים:

1. boggle.py
2. ex11_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!