בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין מבוא למדעי המחשב 67101

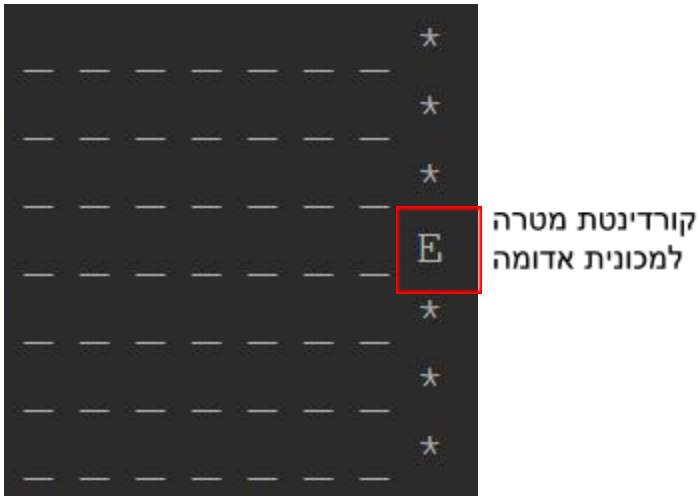
תרגיל 9 - OOP
להגשה בתאריך 1.1.20 בשעה 22:00

מבוא

בתרגיל זה תממשו גרסה של המשחק שעת השיא (rush-hour). המשחק מורכב מלוח דו-מימדי שעליו ממוקמות מכונית אדומה ומכוניות נוספות, מטרת המשחק היא 'לחלץ' את המכונית האדומה מתוך פקק התנועה ולהעביר אותה דרך פתח היציאה.

לוח המשחק

- המשחק אותו תיצרו ממערכת קואורדינטות דו-ממדית. בצורת ריבוע עם אורך צלע 7 (**לא ניתן לשינוי**).
- מערכת המספור של הקואורדינטות מתחילה מהערך 0. הקואורדינטה (0,0) היא הקואורדינטה השמאלית עליונה של הלוח.
- כל נקודה (row,col) על גבי הלוח מזוהה על ידי צמד קואורדינטות, הראשונה לציון המימד האנכי (row) והשנייה לציון הממד האופקי (col).
- קואורדינטת המטרה (פתח היציאה מהחניון) תמיד תהיה בקורדינטה האמצעית (3,7), בצד ימין של הלוח. שימו לב שקורדינטה זו היא מחוץ לריבוע הלוח המרכזי, מה שאומר שרק מכונית אופקית שהקצה הימני שלה נמצא על (3,6) יכולה להגיע לשם על ידי נסיעה ימינה.
- **בכל שלב במשחק רק מכונית אחת יכולה לתפוס קורדינטה נתונה.**



דוגמא של לוח המשחק
(העמודה הימנית, למעט קואורדינטת המטרה,
היא רק להמחשה והיא לא קיימת במשחק)

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין מבוא למדעי המחשב 67101

מכונית

המכונית אותה תייצרו בתרגיל היא אובייקט חד ממדי המונח על לוח המשחק.

- מכונית מאופיינת על ידי :
  - ❖ אורך (length) - מספר התאים אותם תופסת המכונית על פני הממד בו היא מונחת. (הערה: בתרגיל זה אורך המכוניות הוא int בין 2-4)
  - ❖ כיוון (orientation) - מנח המכונית על פני הלוח. כיוון האוריינטציה יכול להיות מאונך (מיוצג ע"י 0) או מאוזן (מיוצג ע"י 1).
  - ❖ שם (name)- בתרגיל זה ישנם שישה שמות חוקיים Y,B,O,W,G,R (המייצגים את הצבעים צהוב, כחול, כתום, לבן, ירוק ואדום בהתאמה). נהוג שהמכונית האדומה היא זו שצריכה להגיע לקורדינטת מטרה, אך בגרסא זו, **אין** הכרח שזאת דווקא תהיה האדומה (משמע, כאשר מכונית כלשהי, לא חשוב מה צבעה, תדרוך בקורדינטת המטרה, הדבר ייחשב ניצחון).
  - ❖ מיקום (location) - כל מכונית נמצאת על פני מספר קואורדינטות בלוח. לשם הנוחות נייצג את מיקום המכונית כמיקום הקואורדינטה בעלת הערך המינימלי מבין מיקומי המכונית (הכי קרוב לפינה שמאלית עליונה).

    לדוגמא:
    - מכונית בעלת אוריינטציה אופקית אשר גודלה 3 והיא נמצאת בקואורדינטות [(0,0),(0,1),(0,2)], הקואורדינטה המינימלית שלה היא (0,0)
    - מכונית בעלת אוריינטציה אנכית אשר גודלה 2 והיא נמצאת בקואורדינטות [(2,0),(3,0)] הקואורדינטה המינימלית שלה היא (2,0).
    - מכונית תסומן בלוח לפי השם, המיקום והאורך שלה. כלומר מכונית צהובה אופקית, באורך 3 במיקום (3,2) תסומן כך-

תנועת מכוניות

מכוניות נוסעות הלוך ושוב על גבי הלוח בממד האוריינטציה שלהן בלבד. כלומר, מכונית בעלת אוריינטציה מאוזנת נוסעת ימינה ושמאלה בלבד ומכונית בעלת אוריינטציה אנכית נוסעת למעלה ולמטה בלבד.

- האוריינטציה של מכונית נקבעת בעת אתחולה ולא משתנה במהלך המשחק.
- משחק המכוניות מתנהל על ידי שחקן בודד אשר מזיז באופן סדרתי את המכוניות על פני הלוח.
- בכל תור, השחקן מזיז מכונית **אחת** לתא סמוך בהתאם למנח המכונית (כלומר מכונית אופקית יכולה לזוז צעד בודד ימינה או שמאלה, ומכונית אנכית יכולה לזוז צעד בודד למעלה או למטה).
- מכונית לא יכולה לחרוג מגבולות הלוח, ולא יכולה 'לדרוס' מכונית אחרת, כלומר תנועת מכונית אפשרית רק לכיוון תאים **בלוח** שאינם תפוסים.

- עליכם לוודא שהקלט המתקבל על ידי המשתמש עונה על כל הדרישות הנ"ל. **במקרה שלא, עליכם להדפיס שגיאה קצרה.**

משחק מכוניות

משחק מכוניות מתנהל באופן הבא :

- הכנת קובץ קונפיגורציית הלוח - על מנת לקבוע את קונפיגורציית הלוח ההתחלתית עליכם להכין קובץ JSON (הסבר בהמשך) המכיל את המידע על שם, אורך, אוריינטציה ומיקום המכוניות.
  - אין צורך לוודא כי קונפיגורציה התחלתית מסוימת היא פתירה, כלומר שניתן להגיע ממנה לסיום המשחק.
- יצירת הלוח הראשוני - התוכנית צריכה לקרוא את הקובץ (בעזרת פונקצית load_json. ראו נספח על קובץ העזר) ולמקם את כל המכוניות על הלוח (לפי הסדר שהתקבל במילון).
אם קיימות מכוניות עם פרמטרים לא חוקיים (לדוגמא, מיקומים לא חוקיים, שזו תכונה של הלוח, או גדלים/שמות לא חוקיים, שהן תכונות של המשחק, וכו'), דלגו עליהן והכניסו למשחק רק מכוניות עם פרמטרים חוקיים.
- מהלך משחק - בכל תור המשתמש מזיז מכונית אחת, צעד אחד (=ריבוע אחד). הוא עושה זאת על ידי בחירת המכונית הרצויה ואת כיוון ההזזה. במידה והקלט תקין, המכונית תזוז. שימו לב שעליכם לוודא תקינות הקלט. אם הקלט אינו תקין הדפיסו על כך הודעת שגיאה קצרה וחכו לקלט נוסף. אם הקלט תקין, הלוח יודפס עם השינויים הרלוונטיים.
- קלט מהמשתמש - קלט חוקי מהמשתמש יהיה בצורת שני תווים מופרדים בפסיק- שם,כיוון. לדוגמא "Y,d" עבור המכונית הצהובה וכיוון למטה. **ללא רווחים בכלל**. צריך לוודא תקינות (לדוגמא שלא מנסים להזיז מכונית אופקית בכיוון אנכי וכו') .
  - כיוונים חוקיים - u,d,l,r
  - שמות חוקיים - Y,B,O,G,W,R
- סיום מוקדם של המשחק - במידה והשחקן רוצה לצאת מהמשחק לפני סיומו ניתן להכניס לשורת קלט את התו "!". הכנסת התו ולחיצת אנטר תסיים את המשחק.
- סיום המשחק - במידה ומכונית כלשהי מגיעה לקורדינטה הסמוכה ליציאה, כלומר - קצה המכונית (**מכונית באוריינטציה מאוזנת**) הימני נמצא ב(3,7), המשחק נגמר.

מימוש

בתרגיל ייעשה מימוש בתכנות מונחה עצמים (OOP).

- התרגיל מסופק עם שלד למספר מחלקות. יש לממש את כל הפונקציות של המחלקות, כמפורט להלן.
הקבצים המסופקים לתרגיל הם :
  - car.py - מכיל מחלקה המייצגת מכונית אשר יש לממש.
  - game.py - מכיל מחלקה המייצגת משחק בעלת פונקציות אשר יש לממש.
  - helper.py - מכיל פונקצית עזר (ראו נספח בסוף התרגיל)
  - board.py - מכיל מחלקה המייצגת את לוח המשחק בעלת פונקציות אשר יש לממש.
  - car_config.json - קובץ json לדוגמא

- אפשר להוסיף פונקציות ומשתנים למחלקות הקיימות, אך לא לשנות את חתימות הפונקציות הקיימות.
- בכל מחלקה עליכם להוסיף תיעוד במקום המיועד (ממש מתחת לחתימת המחלקה)
- לשם הדפסת הלוח, אתם יכולים לבצע את הקריאה print(board). כאשר board הוא אובייקט מסוג Board (שימו לב שאכן מימשתם את פונקציית __str__ שלו לפני ביצוע קריאה זו). שימו לב, **אין הנחיות מפורשות לצורה שבה הלוח צריך להיות מודפס**

JSON

JSON הוא פורמט נפוץ לסידור מידע בקובץ טקסט (JSON). בתרגיל זה עליכם להכין קבצי JSON המכילים את המידע הדרוש לשם הסידור הראשוני של הלוח. הפורמט מאוד דומה למילונים של פייתון. משמאל ניתן לראות דוגמא לקובץ JSON מאוד פשוט, וספציפית מכיל גם תוכן המותאם לתרגיל. אפשר לראות שבכל שורה יש לנו מפתח, שבמקרה הזה הוא שם המכונית. לכל מפתח יש ערך שהוא רשימה עם שאר הנתונים של המכונית, אורך, מיקום, אוריינטציה. הקובץ הנ"ל מסופק לכם עם קבצי התרגיל.

```
{
    "O":[2,[2,3],0],
    "R":[2,[0,0],1]
}
```

על מנת לטעון קובץ JSON לתוך תוכנית עליכם תחילה לייבא את ספריית json. לאחר מכן צריך לפתוח את הקובץ לקריאה (בעזרת open) ולהשתמש בפונקציה load של ספריית json על הקובץ הפתוח. דבר זה יתן לכם מילון ובו תוכן קובץ ה-JSON. עשינו זאת עבורכם בפונקציה load_json הנמצאת בקובץ העזר (ראו נספח).

שימו לב שב-JSON אין טאפל, ולכן יחזיר רשימות.

- הפורמט הזה, שבו המפתח הוא השם והערך הוא רשימה שבה יש אורך, מיקום ואוריינטציה (בסדר הזה), הוא **מחייב** ועליכם לתמוך בו.
- את הקובץ ניתן לערוך בכל עורך טקסט, רק חשוב לשמור אותו עם סיומת של json.
- אנו נבדוק רק קבצי JSON חוקיים מבחינת פורמט ה-JSON אז ניתן להניח שהקובץ יטען בלי שגיאות.
- התוכנית צריכה לקבל את הנתיב של קובץ הJSON כארגומנט ואז יהיה ניתן להשתמש בו דרך sys.argv (כבר ראיתם זאת בעבר)

הערות

1. בכל אחד מהקבצים ישנה מחלקה ובה פונקציות קיימות שעליכם לממש, קיראו היטב את תיעוד הפונקציות הקיימות על מנת להבין איך לממש אותם. בנוסף תצטרכו להוסיף פונקציות נוספות לשם מימוש המשחק כולו.
2. ניתן גם להוסיף מחלקות חדשות אך הן יכולות להסתמך **רק** על הפונקציות (API) שהגדרנו לכם.
3. קלט לא תקין יכול להיות משני סוגים עיקריים -
    a. תו לא חוקי (לדוגמא לנסות להזיז בכיוון g, שזה אינו כיוון חוקי).

b. תווים חוקיים, אבל הפעולה שמנסים לעשות לא חוקית (לדוגמא להזיז מכונית אופקית בכיוון אנכי).

נסו למצוא ולטפל בכמה שיותר מקרים הנ"ל.

דגשים לשימוש ב-API

1. אם פונקציה מופיעה ב-API, היא חייבת להיות מיושמת, גם אם אתם לא מוצאים לה שימוש בתרגיל.
2. אם פונקציה לא מופיעה ב-API, קובץ אחר **אינו יכול להניח את קיומה**. אנחנו יכולים להחליף כל אחד מהקבצים בקובץ העומד בתנאי ה-API, והתכנית צריכה להמשיך לעבוד. לדוגמא, אתם לא יכולים להוסיף פונקציה חדשה במחלקה כלשהי ולהשתמש בה ממחלקה אחרת.
3. מתודה שמחזירה אובייקט (או רשימה) כבר לא שולטת במה שיעשה עם אותו אובייקט. אל תחזירו אובייקט שאתם לא רוצים שישתנה.
4. שימוש באובייקטים של מחלקה לא מחייבת את יבוא המחלקה. בדרך כלל צריכים לייבא את המחלקה רק בשביל לקרוא לבנאי שלה. **קריאות לבנאי בתרגיל זה נעשות רק בשורות קוד או פונקציות המוגנות על ידי התנאי "__name__ == "__main__**
5. פרטים נוספים בתרגיל זה שכדאי לשים לב אליהם:
   a. המכוניות לא מכירות/מקבלות את הלוח משום גורם.
   b. הלוח לא מכיר את חוקי המשחק.
   c. אובייקט המשחק אינו יודע (ישירות) מה גודל הלוח.
   d. באופן עקרוני, למחלקות Car ו- Board אין מגבלה לשמות מסוימים או לשמות באורך תו אחד (במילים אחרות, הם לא מכירות את המשחק שלנו).
   e. הפונקציה __str__ של מחלקת Board מחייבת יצירה של הצגה סבירה של לוח. קוראים חיצונים לא יכולים להניח את הפורמט המדויק של ההדפסה, שלא מופיע ב-API
   f. ניתן להריץ משחק ע"י קריאה לבנאי של Game (יצירת אובייקט מסוג Game), ודרכו לפונקציה play, או משורת הפקודה -

'python3 Game.py [path_to_json]'

   g. לא למדתם על Exceptions, ולכן אתם צריכים לבדוק מראש את הקלט שיישלח לבנאי של המכוניות. חלק מהקלטים הם חוקיים, אך לא במסגרת המשחק שהגדרנו בתרגיל, וחלק אינם חוקיים (כמו אורך שלילי של מכונית) במסגרת ההגדרה של מכונית. את שתי האפשרויות צריך לבדוק מראש. יש לבדוק את הקלט מראש גם אם אתם מיישמים Exceptions לצורך זה.

הוראות הגשה

יש להגיש קובץ zip יחיד ששמו ex9.zip המכיל את הקבצים :

קבצי השלד עם המימוש שלכם בתוכם (תזכורת - אפשר להוסיף אך אין לשנות את הקיים בקבצי השלד)

a. car.py
b. board.py
c. game.py

נספח - פונקציות בקובץ העזר - helper.py

1. *הפונקציה load_json(filename). מקבלת path לקובץ json ומחזירה מילון שמתאים לערכי הקובץ.*