מבוא למדעי המחשב 67101 – סמסטר א' 2024

תרגיל 9 – תכנות מונחה עצמים

להגשה בתאריך **06/03/2024** בשעה 22:00

מבוא

בתרגיל זה תשתמשו בתכנות מונחה עצמים על מנת לממש גרסה של המשחק "שעת השיא" (rush-hour).

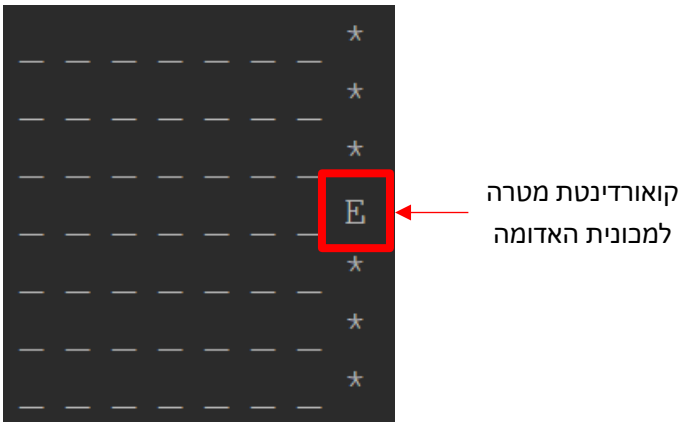
המשחק מורכב מלוח דו-ממדי שעליו ממוקמות מכונית אדומה ומכוניות נוספות. מטרת המשחק היא "לחלץ" את המכונית האדומה מתוך פקק התנועה ולהעביר אותה דרך פתח היציאה.

לוח המשחק

- לוח המשחק מיוצג ע"י מערכת קואורדינטות דו-ממדית.
- מערכת המספור של הקואורדינטות מתחילה מהערך 0. הקואורדינטה (0,0) היא הקואורדינטה השמאלית העליונה של הלוח.
- כל נקודה (row,col) על גבי הלוח מזוהה על ידי צמד קואורדינטות, הראשונה לציון הממד האנכי (row) והשנייה לציון הממד האופקי (col).
- בכל שלב במשחק רק מכונית אחת יכולה לתפוס קואורדינטה נתונה.
- הלוח אותו תממשו יהיה בצורת ריבוע, עם צלע באורך 7. קואורדינטת המטרה שלו תהיה בקואורדינטה האמצעית בצדו הימני של הלוח, כלומר (3,7). שימו לב שקואורדינטה זו היא מחוץ לריבוע הלוח המרכזי ולכן רק מכונית אופקית שהקצה הימני שלה נמצא ב- (3,6) יכולה להגיע לשם על ידי נסיעה ימינה.

דוגמא של לוח המשחק:

(העמודה הימנית, למעט קואורדינטת המטרה, היא רק להמחשה ואינה קיימת במשחק)

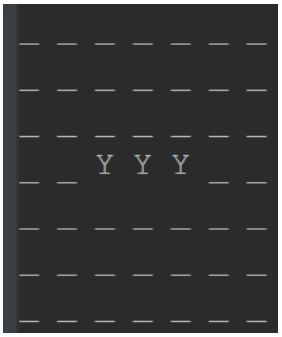


מכונית

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 המכונית הוא מספר שלם בין 2-4)
  - ❖ כיוון (orientation) – מנח המכונית על פני הלוח. הכיוון 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 לקואורדינטת המטרה, אך בגרסה זו, אין הכרח שזאת תהיה דווקא האדומה (משמע, כאשר מכונית כלשהי, לא חשוב מה צבעה, תגיע לקואורדינטת המטרה, הדבר ייחשב ניצחון).
  - ❖ מיקום (location) – כל מכונית נמצאת על פני מספר קואורדינטות בלוח. לשם הנוחות, נייצג את מיקום המכונית באמצעות הקואורדינטה בעלת הערך המינימלי מבין מיקומי המכונית (הקואורדינטה שהכי קרובה לפינה השמאלית העליונה).

    לדוגמא:
    - מכונית באורך 3 בעלת אוריינטציה אופקית, שנמצאת בקואורדינטות [(0,0), (0,1), (0,2)], תיוצג ע"י הקואורדינטה (0,0).
    - מכונית באורך 2 בעלת אוריינטציה אנכית, שנמצאת בקואורדינטות [(2,0), (3,0)], תיוצג ע"י הקואורדינטה (2,0).
- מכונית תסומן בלוח לפי השם, המיקום והאורך שלה. כלומר מכונית צהובה אופקית, באורך 3 במיקום (3,2) תסומן כך:



תנועת מכוניות

במשחק שלנו, 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.
- שימו לב: חלק זה לא מתייחס למחלקה מסוימת. זהו הסבר כללי על תנועת המכוניות במשחק עצמו. מתפקידכם להבין באיזה מהמחלקות יש לבצע כל בדיקה.

משחק מכוניות

המשחק מתנהל באופן הבא:

- הכנת קובץ קונפיגורצית הלוח – על מנת לקבוע את תצורת 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 הפונקציה load_json שנתונה לכם בקובץ העזר- ראו בהמשך) 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. הוא עושה זאת על ידי הזנת קלט שבו הוא מציין איזו מכונית להזיז ובאיזה כיוון. במידה והקלט תקין, המכונית תזוז והלוח יודפס עם השינויים הרלוונטיים. אם הקלט אינו תקין, יש להדפיס הודעת שגיאה קצרה ולחכות לקלט נוסף.
- קלט מהמשתמש - קלט חוקי מהמשתמש יהיה בצורת שני תווים מופרדים בפסיק- שם,כיוון (ללא רווחים כלל). לדוגמא הקלט "Y,d" מציין שהמשתמש בחר להזיז את המכונית הצהובה למטה. יש לוודא שהקלט תקין ומייצג תנועה חוקית במסגרת כללי המשחק.
  - כיוונים חוקיים - u, d, l, r (למעלה, למטה, שמאלה וימינה, בהתאמה)
  - שמות חוקיים - Y, B, O, G, W, R
- סיום מוקדם של המשחק - במידה והשחקן רוצה לצאת מהמשחק לפני סיומו, עליו להזין בקלט את התו "!". פעולה זו תסיים את המשחק.
- סיום המשחק - במידה ומכונית כלשהי מגיעה לקואורדינטת המטרה, המשחק נגמר. עבור הלוח שלנו, זה יקרה אם הקצה הימני של מכונית שנמצאת באוריינטציה מאוזנת נמצא בקואורדינטה (3,7).

JSON

JSON הוא פורמט נפוץ לסידור מידע בקובץ טקסט (JSON). בתרגיל זה עליכם להשתמש בקבצי JSON המכילים את המידע הדרוש לשם הסידור הראשוני של הלוח. הפורמט מאוד דומה למילונים (dict) של Python.

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

משמאל ניתן לראות דוגמא לקובץ JSON מאוד פשוט, המכיל 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 ואוריינטציה. הקובץ הנ"ל מסופק לכם עם קבצי התרגיל.

על מנת לטעון קובץ JSON לתוכנית, יש לבצע את השלבים הבאים (עשינו אותם עבורכם בפונקציה load_json הנמצאת בקובץ העזר):

1. תחילה יש לייבא את הספרייה json.
2. לאחר מכן יש לפתוח את הקובץ לקריאה (באמצעות הפונקציה open).
3. לבסוף יש לקרוא לפונקציה json.load על הקובץ הפתוח, שמחזירה מילון המכיל את תוכן הקובץ.

שימו לב:

- בקבצי JSON אין אפשרות להשתמש ב-tuple, לכן הקואורדינטות מיוצגות ע"י רשימה באורך 2.
- 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, אך הקפידו לשמור אותו עם סיומת של json.
- אנו נבדוק רק קבצי JSON חוקיים מבחינת הפורמט שתיארנו, אז ניתן להניח שהקובץ ייטען ללא שגיאות.
- התוכנית צריכה לקבל את הנתיב של קובץ ה- JSON כארגומנט משורת הפקודה, וניתן להניח שהנתיב שיתקבל הוא תקין ומוביל לקובץ JSON חוקי.

מימוש

בתרגיל ייעשה שימוש בתכנות מונחה עצמים (OOP).

- התרגיל מסופק עם שלד למספר מחלקות. השלד של כל מחלקה מכיל חתימות של מתודות והוא מגדיר את ה-API של אותה המחלקה. יש לממש את כל המתודות המופיעות בשלד, בקבצים המצורפים לתרגיל:
  - car.py – מכיל מחלקה המייצגת מכונית
  - board.py – מכיל מחלקה המייצגת את לוח המשחק
  - game.py – מכיל מחלקה המייצגת משחק

  שימו לב שלכל מתודה יש תיעוד בקובץ שמסביר מה עליה לעשות, מה היא מקבלת ומה היא מחזירה. קראו אותו היטב על מנת להבין כיצד לממש כל מתודה. כמו כן, ייתכן שתצטרכו להוסיף פונקציות/ מתודות נוספות לשם מימוש המשחק כולו.
- בנוסף, מסופקים שני הקבצים הבאים:
  - helper.py – מכיל את פונקציית העזר: `load_json(filename)`, המקבלת נתיב לקובץ הקונפיגורציה ומחזירה מילון שמתאים לערכי הקובץ.
  - car_config.json – קובץ קונפיגורציה לדוגמא.
- אפשר להוסיף מתודות ומשתנים למחלקות הקיימות, <u>אך לא לשנות את חתימות המתודות הקיימות.</u>
- ניתן גם להוסיף מחלקות חדשות אך הן יכולות להסתמך רק על הפונקציות (API) שהגדרנו לכם.

- בכל מחלקה עליכם להוסיף תיעוד במקום המיועד (מתחת לחתימת המחלקה).

הערות

1. לכל מחלקה יש תחום אחריות משלה, והיא לא מכירה את המחלקות האחרות, מלבד ה-API שלהן. למשל: המכונית לא מכירה את הלוח ואת המשחק שלנו, כלומר היא לא מודעת לחוקים של המשחק או של הלוח. לכן, לא מתפקיד המכונית להחליט האם תנועה היא חוקית בתוך הלוח, אבל מכונית כן צריכה להחליט האם התנועה חוקית עבורה. מבחינת האובייקט מכונית, כל שם או אורך מכונית הוא תקין, בניגוד למשחק שלנו, בו יש חוקים מוגדרים בקשר לחוקיות של תכונות מכונית שמשתתפת במשחק. שימו לב לנקודות אלו ולנקודות נוספות במימוש שלכם ובצעו את בדיקות התקינות במקומות המתאימים לכך.
2. עליכם לממש כל מחלקה בצורה עצמאית, **כך שכל אחת מהמחלקות שתממשו תוכל לעבוד עם כל מחלקה אחרת שמישהו בנה, כל עוד היא תואמת ל-API שהוגדרנו** (למשל המחלקה Car שתממשו צריכה לעבוד עם מחלקה Board של מישהו אחר).
3. קלט לא תקין מהמשתמש יכול להיות משני סוגים עיקריים -
   - תו לא חוקי (לדוגמא לנסות להזיז בכיוון g, שזה כיוון לא חוקי).
   - תווים חוקיים המייצגים פעולה לא חוקית (לדוגמא הזזת מכונית אופקית בכיוון אנכי).

   נסו למצוא ולטפל בכמה שיותר מקרים מהסוג הנ"ל.
4. ייצור של אובייקט משחק לא אומר בהכרח שמתחילים לשחק בו.
5. אובייקט משחק יכול לסיים משחק, אבל זה לא תפקידו לסיים את ריצת התוכנית. על כן וודאו שאם מישהו מריץ את המשחק שלכם, ומעוניין להריץ קוד נוסף לאחר שהמשחק הסתיים- הוא יוכל לעשות זאת.
6. ייתכן מצב שבו קובץ הקונפיגורציה מכיל מכונית שנמצאת על קואורדינטת המטרה. מצב זה הוא תקין, ואם המכונית תקינה- יש להכניס אותה ללוח (במצב זה המשחק יסתיים לאחר שהוכנסו כל המכוניות).

דגשים לשימוש ב-API

1. אם מתודה מופיעה ב-API, היא חייבת להיות ממומשת, גם אם אתם לא מוצאים לה שימוש בתרגיל.
2. אם מתודה לא מופיעה ב-API, קובץ אחר אינו יכול להניח את קיומה. לדוגמא, מותר (ומומלץ) להגדיר מתודות חדשות ולהשתמש בהן באותה המחלקה, אך לא ניתן לקרוא להן ממחלקה אחרת.
3. מתודה שמחזירה אובייקט (או רשימה) כבר לא שולטת במה שייעשה עם אותו אובייקט. אל תחזירו אובייקט שאתם לא רוצים שישתנה.
4. שימוש באובייקטים של מחלקה לא מחייב את יבוא המחלקה. בדרך כלל יש לייבא את המחלקה רק בשביל לקרוא לבנאי שלה. קריאות לבנאי בתרגיל זה נעשות רק בשורות קוד או בפונקציות המוגנות על ידי התנאי:

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

5. פרטים נוספים בתרגיל זה שכדאי לשים לב אליהם:
   - המכוניות לא מכירות/ מקבלות את הלוח משום גורם.
   - הלוח לא מכיר את חוקי המשחק.
   - אובייקט המשחק אינו יודע (ישירות) מה גודל הלוח.

- באופן עקרוני, למחלקות Car ו- Board אין מגבלה לשמות מכוניות מסוימים או לשמות באורך תו אחד (במילים אחרות, הן לא מכירות את המשחק שלנו).
- המתודה __str__ של המחלקה Board מחייבת יצירה של הצגה סבירה של הלוח. הפורמט המדויק של ההדפסה לא מופיע ב-API ונתון לשיקול דעתכם.

  לאחר שמימשתם את __str__ תוכלו להדפיס את הלוח באמצעות הקריאה print(board), כאשר board הוא אובייקט מסוג Board.
- ניתן להריץ משחק ע"י קריאה לבנאי של Game (יצירת אובייקט מסוג Game), ואז קריאה למתודה play של האובייקט שהתקבל מהבנאי, או משורת הפקודה-

```
python3 game.py <path_to_json>
```

  ניתן להניח שמספר הארגומנטים המתקבלים משורת הפקודה הוא תקין, ושהנתיב לקובץ הקונפיגורציה גם הוא תקין.
- עדיין לא למדנו על Exceptions, ולכן עליכם לבדוק מראש את תקינות הקלט שיישלח לבנאי של המכוניות. חלק מהקלטים הם חוקיים, אך לא במסגרת המשחק שהגדרנו בתרגיל, וחלק אינם חוקיים במסגרת ההגדרה של מכונית (כמו אורך שלילי 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 קבצי השלד הבאים עם המימוש שלכם בתוכם (תזכורת - אפשר להוסיף אך אין לשנות את הקיים בקבצי השלד):

1. car.py
2. board.py
3. game.py

שימו לב!

לתרגיל זה מצורף Quiz המכיל שאלות על זמני ריצה. שאלות אלה הן חובה ואינן במסגרת המעבדה השבועית. חלק מציון התרגיל כולל בדיקה של שאלות אלה.

בהצלחה!