בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 8 - OOP

להגשה בתאריך 22/12/2017 בשעה 14:00

מבוא

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

תיאור התרגיל מורכב משני חלקים - בחלק המבוא מסופק תיאור מילולי של מהלך המשחק, חוקיו ואופן ניהולו. בחלקו השני של התרגיל ("מימוש") מופיעות הדרישות המפורשות על קטעי הקוד אשר יש לממש והנחיות לגבי בניית המחלקות, הגדרת האינטראקציה ביניהן ודוגמאות לפלטים תקינים של התכנית.

לוח המשחק

לוח המשחק מורכב ממערכת קואורדינטות דו-ממדית.

- לוח המשחק הוא ריבועי, כלומר אורכו שווה תמיד לרוחבו. לכן, ניתן לזהות לוח עם מידת אורך יחידה המייצגת את אורך צלע הריבוע.
- מערכת המספור של הקואורדינטות מתחילה מהערך - 0.
  - הקואורדינטה (0, 0) היא הקואורדינטה השמאלית עליונה של הלוח.
- כל נקודה (x, y) על גבי הלוח מזוהה על ידי צמד קואורדינטות, הראשונה לציון המימד האופקי (x) והשנייה לציון הממד האנכי (y).
- כך, בדוגמה שלהלן מוצג לוח משחק מגודל 5 שהקואורדינטה (1, 3) שלו מסומנת בכוכב.

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מכונית

מכונית היא אובייקט חד ממדי המונח על לוח המשחק.

- מכונית מאופיינת על ידי :

a. אורך - מספר התאים אותם תופסת המכונית על פני הממד בו היא מונחת. (<u>הערה</u>: בתרגיל זה אורך המכוניות הוא 2-4)

b. כיוון (אוריינטציה) - מנח המכונית על פני הלוח. כיוון האוריינטציה יכול להיות מאונך או מאוזן.

c. מיקום על פני הלוח - כל מכונית נמצאת על פני מספר קואורדינטות בלוח. לשם הנוחות נייצג את מיקום המכונית כמיקום הקואורדינטה בעלת הערך המינימלי מבין מיקומי המכונית.

- כדי לקבוע מה ערך הקואורדינטה המינימלי (ראו דוגמה בתת הסעיף הבא), נשווה את ערכי הקואורדינטות בממד האוריינטציה של המכונית.

- לדוגמה מכונית בעלת אוריינטציה אופקית אשר גודלה 3 והיא נמצאת בקואורדינטות [(0,0),(0,1),(0,2)], הקואורדינטה המינימלית שלה היא (0,0).
- לדוגמה מכונית בעלת אוריינטציה אנכית אשר גודלה 2 והיא נמצאת בקואורדינטות [(2,0),(3,0)] הקואורדינטה המינימלית שלה היא (2,0).

d. כך לדוגמה באיור המצורף להלן (של לוח משחק בעל אורך צלע של 6):

- המכונית התכלת היא בעלת אוריינטציה אופקית, אורכה הוא 2, היא נמצאת על התאים (4, 1) ו-(5, 1) על גבי הלוח ולשם נוחות נאמר כי מיקומה הוא (4, 1).
- המכונית הוורודה היא בעלת אוריינטציה אנכית, אורכה הוא 2, היא נמצאת על התאים (1, 3) ו-(1, 4) על גבי הלוח ולשם נוחות נאמר כי מיקומה הוא (1, 3).

- הן בשלב אתחול המשחק ובכל שלב במהלך המשחק, בכל קואורדינטה יכולה להימצא מכונית אחת בלבד.

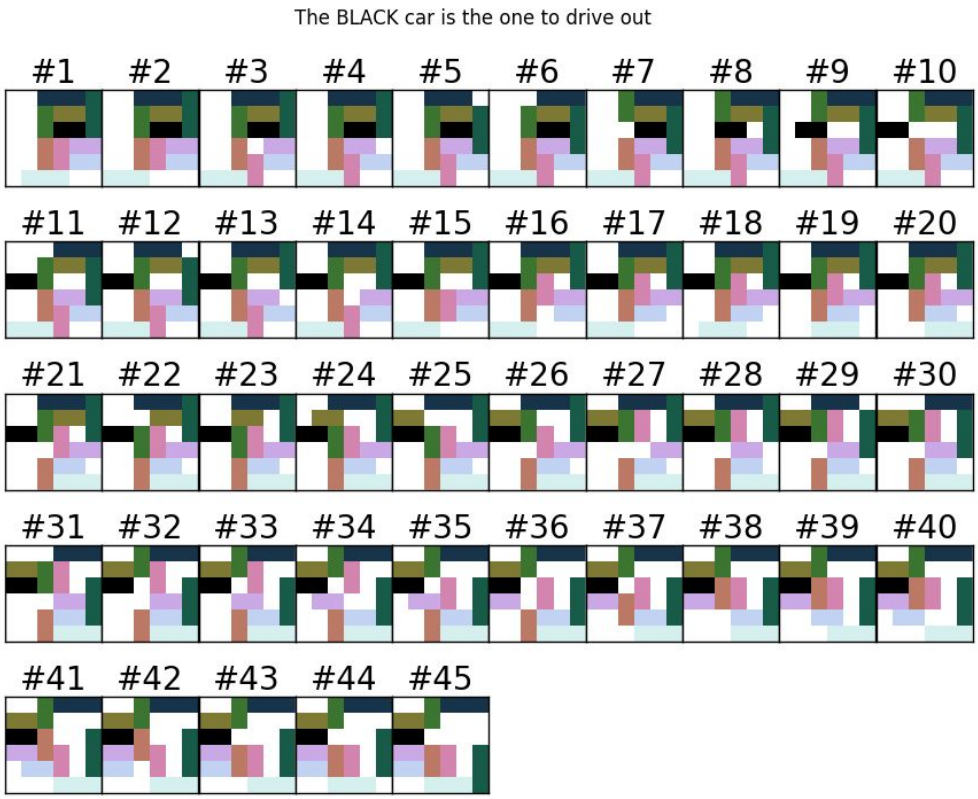
תנועת מכוניות

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 מוגדרת בעת אתחולה ולא משתנה במהלך המשחק.
- משחק המכוניות מתנהל על ידי שחקן בודד אשר מזיז באופן סדרתי את המכוניות על פני הלוח. בכל שלב, השחקן מזיז מכונית יחידה 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.

- מטרת המשחק היא לחלץ את המכוניות האדומה מהפקק, כלומר להזיז את כלל המכוניות על פני הלוח באופן שיאפשר את הסעת המכונית האדומה דרך פתח היציאה. פתח היציאה הוא תא בודד הנמצא על על פני אחת מצלעות הלוח. פתח זה מוגדר בתחילת המשחק ולא משתנה לכל אורך המשחק.

בתרשים המצורף, מוצג מהלך של משחק עם לוח בגודל 6x6. המכונית ה**שחורה** היא זו שצריך לחלץ מהפקק ופתח היציאה נמצא על הלוח בקואורדינטה (2,5) כלומר בשורה השלישית בתא הימני ביותר. בכל שלב, מכונית אחת נוסעת צעד אחד בלבד בכיוון המנח שלה. המשחק מסתיים כאשר המכונית השחורה יכולה לצאת דרך פתח היציאה.



משחק מכוניות

משחק מכוניות מתנהל באופן הבא :

- **<u>אתחול הלוח</u>**

- אתחול לוח ריק בגודל 6x6
- השמה של המכונית האדומה על גבי הלוח

- **<u>השמה של המכוניות</u>** בקשת קלט מהמשתמש של מספר המכוניות הרצוי

עבור כל מכונית:

- הדפסת המצב הנוכחי של הלוח
- קבלת קלט מהמשתמש לגבי: צבע המכונית, אורך המכונית, מיקום המכונית והאוריינטציה של המכונית.

- בדיקת תקינות של הקלט (לדוגמה שאין השמה של מכונית על גבי תא בלוח שכבר תפוס) והשמה של המכוניות על פני הלוח

- **<u>כל עוד המכונית האדומה לא הגיעה לפתח היציאה</u>** :

- הדפסת הלוח
- בקשת קלט של צבע המכונית שאותה רוצים להסיע.
- בקשת קלט של כיוון הנסיעה
- בדיקת תקינות הקלט ועדכון מיקום המכוניות על פני הלוח

- בסיום המשחק יש לדווח למשתמש כי המשחק הסתיים בהצלחה.

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 :
    - <u>car.py</u> - מכיל מחלקה המייצגת מכונית אשר יש לממש.
    - <u>game.py</u> - מכיל מחלקה המייצגת משחק בעלת פונקציות אשר יש לממש.
    - <u>game_helper.py</u> - מכיל פונקציות עזר הממומשות עבורכם.
    - <u>board.py</u> - מכיל מחלקה המייצגת את לוח המשחק בעלת פונקציות אשר יש לממש.
- ניתן להוסיף מחלקות נוספות (שאתם תיצרו) לצורך פתרון התרגיל.
- ניתן להוסיף פונקציות ומשתנים למחלקות הקיימות אך לא לשנות את חתימות הפונקציות הקיימות.

Car

מחלקה המייצגת מכונית. הפונקציה הבאה מופיעה בשלד המחלקה ויש לממש אותה :

```
def __init__(self, color, length, location, orientation):
```

פונקציית האתחול של מכונית. מקבלת את:

i. צבע המכונית
ii. מיקום המכונית (ערכי x ו - y של הקואורדינטה המינימאלית).
iii. אורך המכונית.
iv. האוריינטציה של המכונית.

Board

מחלקה המייצגת את לוח המשחק. הפונקציות הבאות מופיעות בשלד המחלקה ויש לממש אותן:

```
def init(self, cars, exit_board, size=6)
```

פונקציית האתחול של הלוח. מקבלת את:

- גודל צלע הלוח - הלוח הוא ריבועי (ברירת המחדל של גודל הלוח הוא 6).
- פתח היציאה מהלוח (touple של ערכי x ו-y של נקודה על שפת הלוח דרכה המכונית האדומה צריכה לצאת)

- רשימה/מילון/מבנה נתונים אחר של האובייקטים מהמחלקה Car שנמצאים על הלוח. שימו לב, ניתן לאתחל את הלוח עם רשימה ריקה של מכוניות.

```
def add_car (self, car):
```

פונקציה המקבלת אובייקט מסוג מכונית ומוסיפה אותה ללוח המשחק.
במקרה שבו המכונית הוספה בהצלחה, הפונקציה תדפיס הודעה על כך ותחזיר True, ובמקרה של כישלון הפונקציה תדפיס הודעה על כך שמכונית לא הוספה ללוח ותחזיר False.

```
def is_empty (self , location):
```

פונקציה המקבלת קואורדינטה של תא בלוח (tuple של ערכי x ו - y) ומחזירה True אם התא ריק ו- False אחרת.

```
def move(self, car, direction):
```

פונקציה המקבלת אובייקט מסוג מכונית ואובייקט מסוג כיוון, ומסיעה את המכונית בכיוון המבוקש.
הפונקציה צריכה לוודא את תקינות קלט הכיוון, וכן אם התא שאליו מנסים להזיז את המכונית פנוי. (ניתן להשתמש בפונקציות עזר). במקרה של הצלחה, הפונקציה תעדכן את המיקום של המכונית בהתאם לתזוזה ותחזיר True. במקרה של כישלון, הפונקציה תדפיס הודעה המפרטת את הסיבה לכישלון ותחזיר False.

```
def __repr__(self):
```

פונקציה המחזירה מחרוזת המספקת תיאור של המצב הנוכחי של הלוח (הלוח והמכוניות הממוקמות עליו). אין להשתמש בערכי ההחזרה של פונקציה זו למימוש הלוגיקה של המשחק.

Game

מחלקה המייצגת משחק מכוניות.

```
def __init__(self, board):
```

פונקציית האתחול של המחלקה. מקבלת:

- אובייקט מסוג לוח משחק.

```
def single_turn (self):
```

פונקציה המממשת תור יחיד.

a. אופן המימוש של פונקציה זו נתון לשיקול דעתכם וניתן לממש את הלוגיקה המפורטת להלן במספר פונקציות נפרדות.

בתור יחיד מתבצעות הפעולות :

i. הדפסת הלוח למסך (באמצעות הפקודה `print(self.board)`. קריאה זו אפשרית רק לאחר מימוש הפונקציה `__repr__` של המחלקה board)
ii. קבלת קלט מהמשתמש של: צבע המכונית אותה רוצים להזיז.
iii. במידה ולא קיימת מכונית בצבע המבוקש על גבי הלוח - הדפסה של הודעת שגיאה וחזרה לסעיף הקודם.
iv. קבלת קלט מהמשתמש לגבי הכיוון שאליו רוצים להזיז את המכונית.

- את קבלת הקלט מהמשתמש יש לבצע באמצעות הפונקציה `get_direction` מהקובץ game_helper. (הפונקציה מחזירה אובייקט מהמחלקה Direction).

v. בדיקה אם ניתן להזיז את המכונית לכיוון המבוקש (כלומר שהתא אליו מנסים להזיז את המכונית נמצא בגבולות הלוח והוא פנוי).

vi. במידה והתזוזה אפשרית, עדכון של מיקום המכונית והחזרה של true, אחרת - הדפסה של הודעת שגיאה והחזרה של False.

```
def play(self):
```

b. הפונקציה הראשית אשר מנהלת את המשחק.

i. בתחילת המשחק :

1. יש להדפיס למסך הודעת פתיחה.
2. יש להדפיס למסך את הלוח ההתחלתי.
3. יש להוסיף ללוח המשחק את המכונית האדומה (במקום מתאים, כלומר במיקום ובאוריינטציה שיאפשרו את יציאתה דרך פתח היציאה)
4. יש לבקש קלט מהמשתמש לגבי מספר המכוניות שאותו הוא מעוניין למקם על גבי הלוח. את קבלת הקלט יש לבצע באמצעות קריאה לפונקציה `get_num_cars` שבקובץ `game_helper` (פונקציה זו ממומשת עבורכם ואין צורך לממש אותה בעצמכם).
5. יש לבקש קלט מהמשתמש לגבי צבע המכונית, אורך המכונית, מיקום המכונית ומנח המכונית. את קבלת הקלט יש לבצע באמצעות קריאה לפונקציה `get_car_input` שבקובץ `game_helper` (פונקציה זו ממומשת עבורכם ואין צורך לממש אותה בעצמכם).
6. להוסיף את המכוניות ללוח (באמצעות קריאה לפונקציה `add_car` של המחלקה Board).

ii. כל עוד המכונית האדומה לא הגיע לתא המייצג את היציאה מהלוח, ביצוע תור בודד (על ידי קריאה לפונקציה `single_turn`). **שימו לב -** לא לכל משחק קיים פתרון. עבור אתחולים מסוימים של מכוניות על הלוח, אין אפשרות להוביל את המכונית האדומה דרך פתח היציאה. בתרגיל זה, **אינכם** נדרשים לטפל באופן מיוחד במקרים אלו.

iii. לאחר שהמכונית הגיע ליציאה - הדפסת הודעה למשתמש ויציאה מהמשחק (לצורך ההדפסה יש לקרוא לפונקציה `report_gameover` (שבקובץ game_helper.py):

```
'Congratulations! you released the red car!'
```

<u>שימו לב</u> עיצוב נכון (ומחייב לתרגיל) בסגנון תכנות מונחה עצמים הוא הפרדה בין מימוש של כל הישויות במשחק ובפרט בין המחלקה Game והמחלקות Car, Board. המחלקה Game מקבלת אובייקט מסוג Board בקונסטקרטור שלה (בפונקציה __init__) וכל מה ש - Game יכולה להניח על הלוח הוא שהוא מממש את הפונקציות הנתונות בספסיפיקציה של המחלקה Board (הפונקציות add_car , is_free, move וכל הפונקציות הפרטיות כמפורט בתרגיל) ושהן מקבלות קלט ומחזירות פלט כמפורט בתרגיל. אך, **אסור** ל - Game להניח דבר על המימוש עצמו של הפונקציות הללו.

סגנון זה נקרא **design by contract**. מתכנת א' (אתם) אחראי על בניית המחלקה Game ומתכנת ב' (שוב, אתם) אחראי על בניית המחלקה Board. כל מה שמעניין את מתכנת א' זה שהמחלקה Board תחזיר פלט תקין

עבור קלט תקין (לדוגמה שתחזיר True כאשר קוראים לפונקציה - is_free רק אם התא המבוקש אכן פנוי) אבל לא אכפת לו מה הפלט הזה. בפרט, אם מתכנת ב' מחליט להשתגע וקובע שמעכשיו כל פעם שקוראים ל - move המכונית זזה שני תאים קדימה/זזה בכיוון רנדומלי/לא זזה לשום מקום, שהמכוניות הן בצורת ריבוע או שהן מונחות באלכסון זו בעיה של מתכנת ב' ומתכנת א' לא צריך לדאוג לזה. כל מה שמתכנת א' צריך לדעת הוא מה סוג הקלט שהוא מצופה להעביר כשהוא קורא לפונקציות של Board ומה סוג הפלט שהוא אמור לקבל.
עיקרון נוסף שיש להקפיד עליו הוא **אינקפסולציה**.על פי עקרון זה, מחלקות לא חושפות את המשתנים הפרטיים שלהן למחלקות אחרות. במקום זאת, ניהול המשתנים הפרטיים של מחלקה מתבצע מבחוץ באמצעות פונקציות אשר המחלקה בוחרת לחשוף. כך לדוגמה, אם המחלקה Board מכילה רשימה של מכוניות הנמצאות כעת על הלוח ואתם מעוניינים להסיר את המכונית השלישית - גישה ישירה אל הרשימה ומחיקת המכונית:

```
del my_game.car_list[2]
```

תהיה פגיעה בעקרון האינקפסולציה כיוון שרשימת המכוניות היא משתנה פרטי של Board ולא צריך לאפשר אליו גישה מבחוץ. במקום, עדיף להגדיר פונקציה פומבית אשר תפקידה להסיר את המכונית מהלוח ולקרוא לה.:

```
my_game.remove_car (car_index=2)
```

הוראות הגשה

יש להגיש קובץ zip יחיד ששמו ex8 המכיל את הקבצים :

1. README
2. 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

משחק לדוגמה

משחק זה מדגים מהלך משחק עם לוח בגודל 5 בו יוצבו שתי מכוניות (צהובה וורודה - p, y) באורכים 4 ו-3 בהתאמה.

```
Welcome to rush Hour game
Insert number of cars that you want to add to the board (1 - 5): 2
['0', '_', '_', '_', 'R', '_', '_']
['1', '_', '_', '_', 'R', '_', '_']
['2', '_', '_', '_', '_', '_', '_']
['3', '_', '_', '_', '_', '_', '_']
['4', '_', '_', '_', '_', '_', '_']
['5', '_', '_', '_', '_', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
Insert a color of car to add to board. available colors are:
b,g,o,p,w,y: p
Insert a car length (2, 3 or 4): 3
```

```
Insert x,y coordinates of car location: 5,0
Insert car orientation (0 - vertical, 1 - horizontal): 1
A car was added successfully!
['0', '_', '_', '_', 'R', '_', '_']
['1', '_', '_', '_', 'R', '_', '_']
['2', '_', '_', '_', '_', '_', '_']
['3', '_', '_', '_', '_', '_', '_']
['4', '_', '_', '_', '_', '_', '_']
['5', 'p', 'p', 'p', '_', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
Insert a color of car to add to board. available colors are:
b,g,o,w,y: w
Insert a car length (2, 3 or 4): 1,5
Not a valid car length
Insert a car length (2, 3 or 4): 4
Insert x,y coordinates of car location: 0,5
Insert car orientation (0 - vertical, 1 - horizontal): 0
A car was added successfully!
['0', '_', '_', '_', 'R', '_', 'w']
['1', '_', '_', '_', 'R', '_', 'w']
['2', '_', '_', '_', '_', '_', 'w']
['3', '_', '_', '_', '_', '_', 'w']
['4', '_', '_', '_', '_', '_', '_']
['5', 'p', 'p', 'p', '_', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? R
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 8
['0', '_', '_', '_', '_', '_', 'w']
['1', '_', '_', '_', 'R', '_', 'w']
['2', '_', '_', '_', 'R', '_', 'w']
['3', '_', '_', '_', '_', '_', 'w']
['4', '_', '_', '_', '_', '_', '_']
['5', 'p', 'p', 'p', '_', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? p
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 6
['0', '_', '_', '_', '_', '_', 'w']
['1', '_', '_', '_', 'R', '_', 'w']
['2', '_', '_', '_', 'R', '_', 'w']
['3', '_', '_', '_', '_', '_', 'w']
['4', '_', '_', '_', '_', '_', '_']
```

```
['5', '_', 'p', 'p', 'p', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? R
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 8
['0', '_', '_', '_', '_', '_', 'w']
['1', '_', '_', '_', '_', '_', 'w']
['2', '_', '_', '_', 'R', '_', 'w']
['3', '_', '_', '_', 'R', '_', 'w']
['4', '_', '_', '_', '_', '_', '_']
['5', '_', 'p', 'p', 'p', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? R
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 8
['0', '_', '_', '_', '_', '_', 'w']
['1', '_', '_', '_', '_', '_', 'w']
['2', '_', '_', '_', '_', '_', 'w']
['3', '_', '_', '_', 'R', '_', 'w']
['4', '_', '_', '_', 'R', '_', '_']
['5', '_', 'p', 'p', 'p', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? p
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 4
['0', '_', '_', '_', '_', '_', 'w']
['1', '_', '_', '_', '_', '_', 'w']
['2', '_', '_', '_', '_', '_', 'w']
['3', '_', '_', '_', 'R', '_', 'w']
['4', '_', '_', '_', 'R', '_', '_']
['5', 'p', 'p', 'p', '_', '_', '_']
['-', '0', '1', '2', 'E', '4', '5']
What color car would you like to move? R
Insert the direction you would like to move the car. (up = 2, down =
8 right = 6, left = 4): 8
Congratulations! you released the red car!
```