מבוא למדעי המחשב (234221) סמסטר חורף 2019-2020

תרגיל בית 3 - פונקציות ו-Top Down Design

מועד אחרון להגשה: **יום ב' – 30.12.2019 23:59**

על כל שאלה בתרגיל זה יש לפנות במייל ו/או להגיע לאחת משעות הקבלה של צוות הקורס.
הרשימה של המתרגלים, מיילים ושעות קבלה מופיעה באתר הקורס תחת הקישור STAFF.
מתרגל אחראי על התרגיל: אליה טורנר
Email: eliaturner11@gmail.com
* יש לציין את מספר תרגיל הבית, מספר הקורס ואת מספר הסטודנט בנושא ההודעה.

בדוגמאות המוצגות בשתי השאלות, הטקסט **האפור** מייצג את פלט התוכנית והטקסט **הירוק** מייצג את קלט המשתמש.

הוראות הגשה:

- ההגשות בבודדים בלבד!
- יש להגיש קובץ בודד hw3q1.py המכיל את התוכנית הדרושה.
- לפני ההגשה, יש לבדוק את נכונות התוכניות שכתבתם ע"י שימוש ב-redirection של קלט ופלט התוכנית והשוואה עם DiffMerge לקבצי ה- input output לדוגמא אשר יופיעו באתר הקורס (בסמוך לתרגיל – בחרו את הקבצים שמתאימים למערכת ההפעלה בה אתם משתמשים).
- במידה וקיימים הבדלים – תקנו אותם בתוכנית המקורית וחזרו על השלבים עד לקבלת פלט זהה.
- הפלטים שמוצגים בקובץ זה הם להמחשה בלבד! הפורמט המדויק של הקלט נמצא אך ורק בקבצי הדוגמא! אם יש הבדלים בין השניים, היצמדו למה שמופיע בטסטים שהועלו לאתר.
- בנוסף, כמו בתרגיל בית הקודם יש לצרף קובץ students.txt המכיל את שמכם **באנגלית**, את תעודת הזהות ואת כתובת האימייל שלכם.
- יש ליצור קובץ hw3.zip שבתוכו הקבצים אשר ציינו לעיל – **וודאו שאין תיקיות בקובץ הזיפ!**

שימו לב!: בכל שעורי הבית, יש להגיש אלקטרונית. אחרי המסך בו יש להכניס ת.ז. וסיסמא, יהיה מסך נוסף בו ניתן להעלות את הקובץ hw3.zip. **בסוף, תקבלו אישור שההגשה עברה בהצלחה, עליכם לשמור על קוד זה עד שציונכם התפרסם!**

בהצלחה !

<u>הנחיות לתכנון וכתיבת קוד: חשוב לקרוא לפני התרגיל!</u>

בתרגיל זה המטרה היא לתרגל אתכם בפירוק בעיה לגורמים קלים לתכנות (ותכנון). לצורך כך, ישנן כמה מגבלות על כתיבת הקוד.

אי עמידה בדרישות אלה עלול לגרור הורדה בציון הסופי של התרגיל

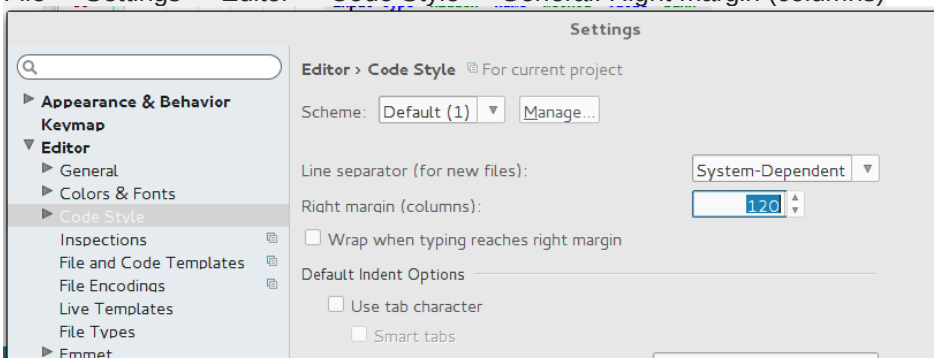
1. אורך כל פונקציה לא יעלה על **16** שורות קוד.

   <u>חשוב:</u> עמידה בתנאי זה דורשת תכנון מוקדם של הקוד! יינתן פירוט בהמשך

   *בעזרת תכנון נכון אפשר להגביל את אורך הפונקציה המקסימלית ל-13.

2. רוחב כל שורה (כולל הערות והזחות) לא יעלה על 75 תווים. ניתן להגביל אורך של שורה ב-Pycharm ע"י המסלול הבא:

File -> Settings -> Editor -> Code Style -> General: Right margin (columns)



3. לפני כל פונקציה, יש לכתוב הערה (בקצרה) מה הפונקציה עושה
   a. ההערה צריכה להסביר מה הפונקציה עושה ולא כיצד היא עושה. כלומר, מבלי להכנס לפרטי המימוש של הפונקציה. לדוגמה: "הפונקציה מכניסה את x ו-y למערך move" אינו הסבר, בעוד ש-"הפונקציה שומרת את המהלך האחרון במערך moves" הוא כן הסבר.
   b. ההערות צריכה להיות באנגלית
   c. ההערה צריכה להופיע לפני מימוש הפונקציה
   d. יש לתאר לכל פונקציה בתיעוד את כל הפרמטרים שמקבלת ואילו פרמטרים מחזירה.
4. חובה לתת שמות משמעותיים לפונקציות ולמשתנים. כלומר, גם בלי ההערות הקוד צריך להיות *self-explanatory*.
5. אסור לשכפל קוד שלא לצורך!
6. הקוד שלכם צריך להיות גמיש לשינויים – לדוגמא, שינוי גודל רצף האסימונים שצריך להשיג כדי לנצח (במקום 4 המקורי), שינוי התווים של השחקנים, שינוי אורך ושורה של לוח. באופן כללי – הקפידו על שימוש במשתנים, כל ערך קבוע שחוזר בתוכנית יותר מפעם אחת אמור לעורר את חשדכם!

<u>הגדרות לגבי אורך הפונקציה:</u>

1. מגבלת השורות תקפה לגבי כל פונקציה בקוד, כולל ה-main.
2. אסור לכתוב שתי פקודות בשורה ולהפרידן ע"י ;
3. כותרת הפונקציה, שורה המכילה הערות בלבד או שורה המכילה else בלבד אינה נחשבת לספירת אורך הפונקציה – כל שאר השורות נחשבות.
4. שורה ארוכה שנשברה ל-2 (כדי שרוחב השורה יהיה קטן מ-75 תווים) נחשבת כ-2 שורות.

שאלה 1- צוללות:

בתרגיל הזה נממש את המשחק צוללות.
במשחק משתתפים שני שחקנים. לכל שחקן יש לוח בגודל NxN שעליו ממקם צוללות קרב. כל שחקן רואה את הצוללות שמיקם בלוח שלו, אבל לא יכול לראות את מיקום צוללות היריב.
מטרת המשחק היא להתקיף את כל צוללות היריב.
במהלך המשחק , כל שחקן בתורו בוחר איזה משבצת לתקוף אצל היריב. לשם כך, לכל שחקן יש לוח נוסף שהוא לוח המעקב אחר היריב. אותו לוח מעקב יתמלא באינפורמציה הנכונה – אם בנקודה שהותקפה נמצאה צוללת יסומן V, אחרת יסומן X. צוללת נחשבת טבעה אם תקפו את כולה.
מומלץ להתנסות במשחק לפני שניגשים לקרוא את שאר ההוראות. דוגמא למשחק אונליין נמצאת בקישור הבא:

http://en.battleship-game.org/

בתרגיל זה, המשחק יהיה בין מחשב למשתמש. בתורות של המשתמש, נקודות התקיפה יתקבלו כקלט, ובתורות של המחשב נקודות התקיפה יוגרלו.
כדי להגריל מהלכים עבור המחשב נשתמש בספריה random. כדי שנהיה מסוגלים לשחזר הרצות נשתמש בכל דוגמא ב-seed מסוים, שיתקבל כקלט מהמשתמש בתחילת התוכנית.
כדי להגדיר seed ריצה נשתמש בפקודה random.seed(seed) כאשר seed הוא ארגומט מסוג int.
כדי להגריל אינדקסים בלוח נשתמש בפקודה random.randint(start,stop). הפקודה מגרילה מספר שלם בטווח start עד stop **כולל**!

פתיחת המשחק:

בתחילת המשחק תוצג הודעת פתיחה:

```
Welcome to Battleship!
```

התוכנית תבקש מהמשתמש seed:

```
Please enter seed:
```

אתחול לוח:

לכל שחקן יש מספר מוגדר של צוללות מכל גודל שעליו למקם בלוח.
בטסטים אתם תבדקו על לוח בגודל 10x10 ולכל שחקן יהיו:

- ארבע צוללות בגודל 1
- שלוש צוללות בגודל 2
- שתי צוללות בגודל 3
- צוללת אחת בגודל 4

לצורך המסמך בלבד ומפאת חוסר מקום נניח שלכל שחקן יש שתי צוללות בגודל 2 וצוללת אחת בגודל 3. נסמן את גודל הצוללת ב-size לצורך ההסבר.

גם המשתמש וגם המחשב צריכים לבחור את המיקומים של הצוללות.
כדי למקם כל צוללת יש להזין שלושה פרמטרים: מספר טור (מיקום על ציר x) מספר שורה (מיקום על ציר y) וכיוון אנכי (vertical) או אופקי (horizontal).
אם הכיוון הוא אנכי, הצוללת תמוקם במקומות (x,y)-(x,y+size-1).

אם הכיוון הוא אופקי, הצוללת תמוקם במקומות (x,y)-(x+size-1,y).

<u>עבור המשתמש:</u>
המשתמש יזין את כל הנתונים הנ"ל עבור כל צוללת בנפרד.
הקלט יינתן בפורמט הבא:

x,y orientation

כאשר x,y הן הקאורדינטות כפי שהוסבר למעלה, ו-orientation זו אות 'h'/'v'.
לפני בקשת קלט עבור כל צוללת בגודל * יודפס לוח המשחק של המשתמש ולאחר מכן תודפס ההודעה:

```
Enter location for Battleship of size *:
```

במקרה שיש שגיאה כלשהי בערכי הצוללות יש להדפיס:

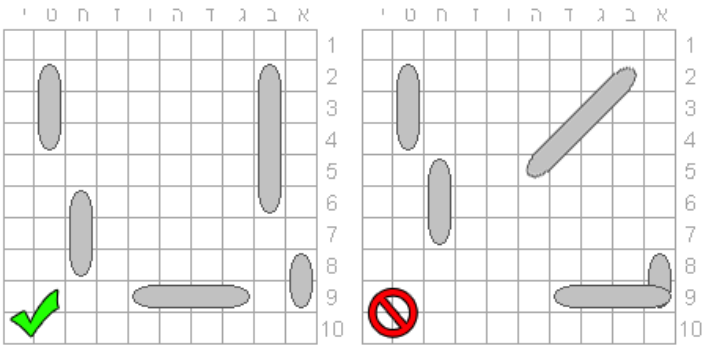
```
Invalid input
```

<u>עבור המחשב:</u>
אתחול הלוח יעשה בדומה למשתמש, רק שבמקום לקלוט מהמשתמש, המיקום והכיוון של כל צוללת יוגרלו ע"י הפונקציה randint כפי שהוסבר למעלה. הקואורדינטות יוגרלו בטווח [0,N-1] כאשר N הוא גודל הלוח. לאחר מכן הכיוון יוגרל ע"י הגרלה של מספר 0/1, 0 מתאים ל-'h' ו-1 מתאים ל-'v'. ההגרלות חייבות להתבצע בדיוק בסדר הזה!!! אחרת הפלט בטסטים לא יתאם לפלט של הקוד שלכם.
במקרה של שגיאה כלשהי בהגרלת ערכי הצוללת, יש להגריל את הערכים מההתחלה בלי להדפיס הודעה כלשהי.

<u>שגיאות אפשריות:</u>

1. מיקום הצוללת חורג מגבולות הלוח. יש לשים לב גם להתחשב בגודל הצוללת ובאוריינטציה שלה. למשל, עבור N=5 וקלט 'h' 4,4 וגודל צוללת 2, אמנם הקאורדינטות הן בטווח הלוח אבל הצוללת עצמה תחרוג מהלוח.
2. אם הצוללת שנבחרה חופפת לצוללת אחרת שכבר מוקמה, כלומר קיים תא ששייך לצוללת שמוקמה בעבר, והצוללת הנוכחית כוללת גם את תא זה.
3. אם הצוללת מוצבת בסמוך לצוללת אחרת שכבר הוצבה, כולל באלכסון.

להלן המחשה מוויקיפדיה- צד ימין מתאר הצבות לא תקינות של צוללות, וצד שמאל מתאר הצבות תקינות של צוללות (ויקיפדיה):



דוגמא לאתחול המשחק:

בדוגמא הנ״ל N=5 ויש 2 צוללות בגודל 2 וצוללת אחת בגודל 3:

```
Welcome to Battleship!
Please enter seed:
4
Your current board:
   0 1 2 3 4
   - - - - - -
0 |
1 |
2 |
3 |
4 |

Enter location for Battleship of size 2:
4,0 v
Your current board:
   0 1 2 3 4
   - - - - -
0 |       *
1 |       *
2 |
3 |
4 |

Enter location for Battleship of size 2:
4,2 v
Invalid input
Please enter location for Battleship of size 2 again:
4,4 v
Invalid input
Please enter location for Battleship of size 2 again:
4,1 v
```

ביצוע התקפות:

המשחק הוא בתורות כאשר המשתמש מתחיל ראשון.

<u>ביצוע תור המשתמש:</u>
בתחילת כל תור של השחקן יודפס מצב המשחק בסדר הבא:
קודם כל לוח המעקב של השחקן על המחשב (בפרט בהתחלה הוא יוצג כלוח ריק) ואחריו יוצג לוח המעקב של המחשב עם מיקום צוללות השחקן.
תא בלוח המעקב שהתגלתה בו צוללת יסומן ע״י ’V’ ואילו תא שהתגלה שאין בו צוללת יסומן ע״י ’X’.
חלקי צוללות השחקן שמוצגים על גבי לוח המעקב של המחשב ועדיין לא הותקפו על ידו יסומנו ב-’*’.

מתחת להדפסת מצב המשחק, תוצג הודעה לשחקן הנוכחי, שתבקש ממנו להזין את המיקום בלוח היריב שבו הוא רוצה לתקוף.

```
Your following table:
   0 1 2 3 4
   - - - - -
0 |
1 |
2 |
3 |
4 |

The computer's following table:
   0 1 2 3 4
   - - - - -
0 | * *
```

השחקן יזין את המיקום שברצונו לתקוף בצורה הבאה: x,y (בדומה להזנת המיקומים בעת שיבוץ הצוללות).

במקרה של שגיאה כלשהי בבחירת מיקום התקיפה יש להדפיס את ההודעה:

```
Error: Invalid attack...
Please try again:
```

ולקלוט מיקום בשנית.

ביצוע תור המחשב:
לאחר שהשחקן יבצע את תורו באופן תקין, המחשב יבצע את תורו בצורה רנדומלית.
ביצוע מהלכי המחשב יהיה באופן דומה לאתחול לוח המחשב: המחשב יגריל את קאורדינטות x ו-y לפי הסדר בתחום [0,N-1] עד אשר המהלך שיתקבל יהיה תקין.

שגיאות אפשריות בבחירת מיקום התקיפה:
1. הזנת ערך החורג מגבולות הלוח (רק במקרה של המשתמש).
2. תקיפה במקום שכבר הותקף קודם לכן.

הטבעת צוללת:

לאחר כל תור התוכנית תדפיס הודעה במידה וצוללת כלשהי הותקפה בשלמותה.
אם הצוללת של המשתמש טבעה:

```
Your battleship has been drowned.
```

אם הצוללת של המחשב טבעה"

```
The computer's battleship has been drowned.
```

בכל מקרה, לאחר מכן יודפס מספר הצוללות שנותר להטביע:

```
*/3 battleships remain!
```

כאשר * מייצג את מספר הצוללות שנותרו, ו-3 במקרה זה מייצג את סך כל הצוללות במשחק.

סיום המשחק:

כפי שנאמר, המנצח במשחק יהיה מי שהטביע את כל צוללות היריב.
במידה והמשתמש ניצח, תודפס ההודעה:

```
Congrats! You are the winner :)
```

ובמידה והמחשב ניצח, תודפס ההודעה:

```
Game over! The computer won the fight :(
```

הנחות על הקלט:

ניתן להניח שכל קלט שתקבלו יהיה משחק מלא מההתחלה ועד הסוף. כלומר לעולם לא תתרחש שגיאת EOF.

ניתן להניח שהקלט נכון מבחינה סינטקטית. כלומר שגיאות אפשריות לקלט, אשר פורטו למעלה, יהיו רק בקשר לערכים שמתקבלים, לא תבדקו על שגיאות בפורמט הקלט.

קובץ hw3q1_template.py

לשימושכם אנו מצרפים קובץ תבנית המהווה המלצה לבסיס מימוש המשחק.
בקובץ ניתן למצוא:

- מספר משתנים גלובאליים שהגדרנו כדי לפשט את המימוש.
- פונקציות הדפסה ממומשות.
- חתימות של פונקציות שאנו ממליצים לממש.

אנו ממליצים בחום להתחיל עם מה שנמצא בקובץ ולא לממש לבד, לשיקולכם. מי שבוחר לעבוד עם הקובץ יש להקפיד לשנות את השם שלו לפני ההגשה (למחוק את הסיומת _template).

```
Welcome to Battleship!
Please enter seed:
4
Your current board:
  0 1 2 3 4
  - - - - -
0 |
1 |
2 |
3 |
4 |

Enter location for Battleship of size 2:
4,0 v
Your current board:
  0 1 2 3 4
  - - - - -
0 |     *
1 |     *
2 |
3 |
4 |

Enter location for Battleship of size 2:
4,2 v
ERROR: Invalid location
Please enter location for Battleship of size 2 again:
5,5 v
ERROR: Invalid location
Please enter location for Battleship of size 2 again:
```

```
4,3 v
Your current board:
   0 1 2 3 4
   - - - - -
0 |       *
1 |       *
2 |
3 |       *
4 |       *

Enter location for Battleship of size 3:
0,2 h
All battleships have been located successfully!
Your following table:
   0 1 2 3 4
   - - - - -
0 |
1 |
2 |
3 |
4 |

The computer's following table:
   0 1 2 3 4
   - - - - -
0 |       *
1 |       *
2 | * * *
3 |       *
4 |       *

It's your turn!
Enter location for attack:
2,2
Your following table:
   0 1 2 3 4
   - - - - -
0 |
1 |
2 |     V
3 |
4 |
```

```
The computer's following table:
   0 1 2 3 4
   - - - - -
0 |       *
1 |       *
2 | * * *
3 |       *
4 |     X *

It's your turn!
Enter location for attack:
3,2
Your following table:
   0 1 2 3 4
   - - - - -
0 |
1 |
2 |     V X
3 |
4 |

The computer's following table:
   0 1 2 3 4
   - - - - -
0 |       *
1 |       *
2 | * * *
3 |     X   *
4 |     X *

It's your turn!
Enter location for attack:
1,2
The computer's battleship has been drowned.
2/3 battleships remain!
Your following table:
   0 1 2 3 4
   - - - - -
0 |
1 |
2 |   V V X
3 |
4 |
```

```
The computer's following table:
  0 1 2 3 4
  - - - - -
0 |       *
1 |     X *
2 | * * *
3 |   X   *
4 |     X *

It's your turn!
Enter location for attack:
2,0
Your following table:
  0 1 2 3 4
  - - - - -
0 |   X
1 |
2 |  V V X
3 |
4 |

The computer's following table:
  0 1 2 3 4
  - - - - -
0 |       *
1 |     X *
2 | * V *
3 |   X   *
4 |     X *

It's your turn!
Enter location for attack:
2,4
Your following table:
  0 1 2 3 4
  - - - - -
0 |   X
1 |
2 |  V V X
3 |
4 |   X
```

```
The computer's following table:
  0 1 2 3 4
  - - - - -
0 |   X   *
1 |     X *
2 | * V *
3 |   X   *
4 |     X *

It's your turn!
Enter location for attack:
4,3
Your following table:
  0 1 2 3 4
  - - - - -
0 |   X
1 |
2 |  V V X
3 |       V
4 |   X

The computer's following table:
  0 1 2 3 4
  - - - - -
0 | X   X   *
1 |     X *
2 | * V *
3 |   X   *
4 |     X *

It's your turn!
Enter location for attack:
4,2
Your following table:
  0 1 2 3 4
  - - - - -
0 |   X
1 |
2 |  V V X V
3 |       V
4 |   X

The computer's following table:
  0 1 2 3 4
```