המדריך למשתתף - PlanetWars

המשימה

לכתוב תכנית מחשב (Bot) שתשחק במשחק PlanetWars ותנצח!

ספר **Python** בעברית

http://data.cyber.org.il/python/python_book.pdf

אתר התחרות

http://planetwars.cyber.org.il/

הרצת משחק בין **Bot**-ים

התקנות

בצעו את ההתקנות הדרושות בהתאם לקובץ ההתקנות שנשלח אליכם לקראת המחנה.

הורידו את **Starter Kit** מעמוד הקבוצה שלכם באתר התחרות.

פתחו את קובץ ה-zip לתיקייה במחשב שלכם, למשל:

```
c:\starter_kit
```

בעזרת ה-Starter Kit תוכלו להריץ על המחשב שלכם משחקים, ולצפות בהם ויזואלית.

הרצת משחק

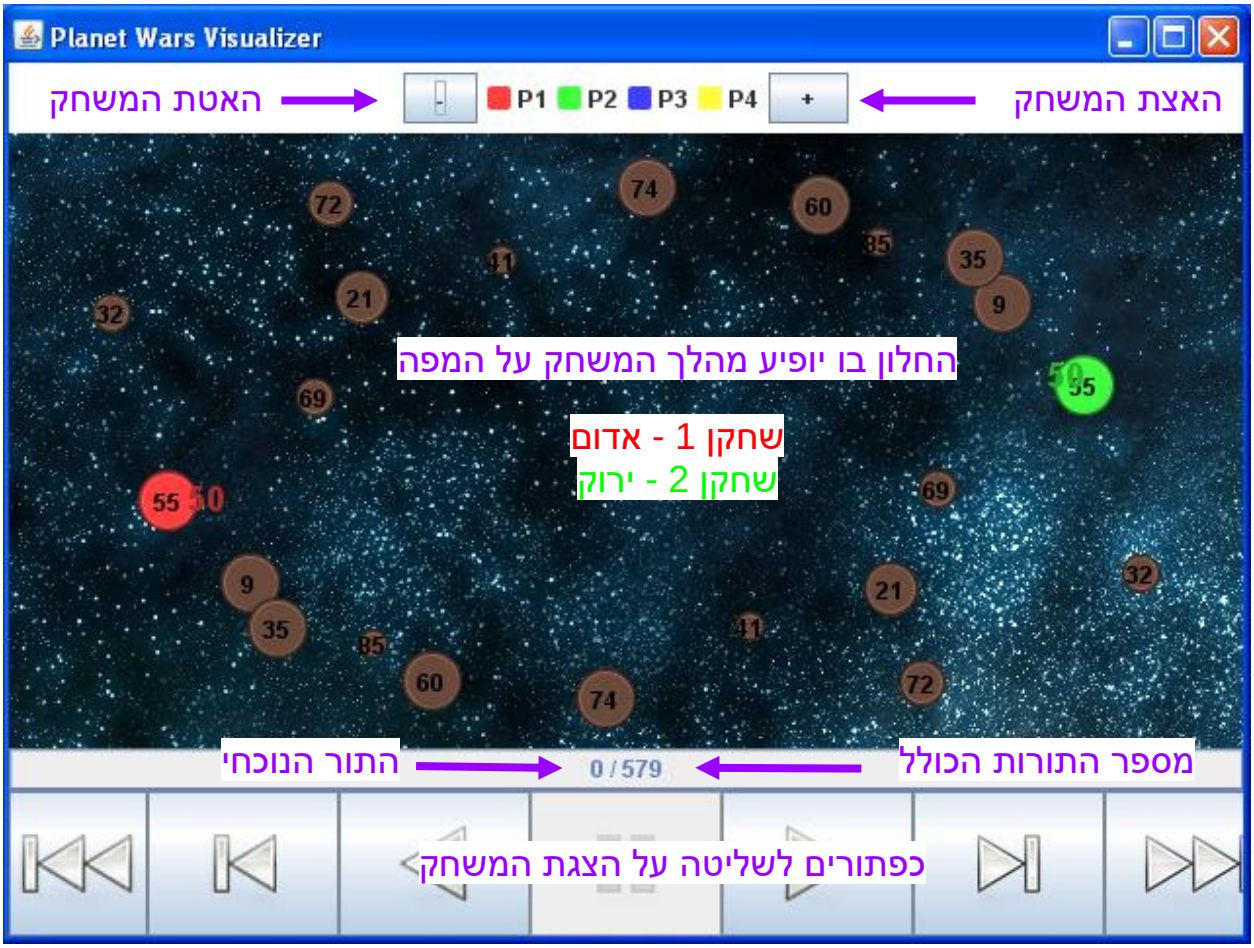
1. פתחו את ה-cmd.
2. היכנסו לתיקייה שאליה פתחתם את ה-Starter Kit. למשל: `cd c:\starter_kit`
3. כתבו:

```
run.bat bots\DemoBot.pyc bots\DemoBot.pyc
```

הסבר: קובץ ההרצה run.bat מצפה לקבל שמות של שני קבצי Python שהם Bot-ים במשחק. במקרה הזה, אנחנו נריץ את אותו ה-Bot שבקובץ DemoBot.pyc נגד עצמו.
באופן כללי, תוכלו כמובן להריץ קבצי Bot-ים אחרים:

```
run.bat <Player One Bot Path> <player Two Bot Path>
```

4. לאחר הרצת run.bat, המשחק בין ה-Bot-ים ירוץ ב-cmd, ותוכלו לראות בסופו של דבר מי ניצח, וכן הדפסות Debug שתרשמו בקוד של ה-Bot שלכם (אל דאגה, נסביר זאת בהמשך).
5. מיד לאחר סיום הרצת המשחק ב-cmd, ייפתח לכם חלון המציג את מהלך המשחק בצורה גרפית:


Planet Wars Visualizer
P1 P2 P3 P4
האצת המשחק
האטת המשחק
החלון בו יופיע מהלך המשחק על המפה
שחקן 1 - אדום
שחקן 2 - ירוק
מספר התורות הכולל
התור הנוכחי
0 / 579
כפתורים לשליטה על הצגת המשחק

Toutorial למימוש **Bot**

פרק זה נצעד את צעדינו הראשונים בכתיבת Bot ב-Python.

לפני הכל, פתחו את עורך הטקסט האהוב עליכם (למשל Sublime Text) וצרו קובץ חדש בשם TutorialBot.py ושימרו אותו בתיקיית ה-Starterkit.

הקפידו להעתיק בעצמכם כל שורת קוד שמופיעה כאן אל הקובץ שיצרתם, והריצו לאחר כל שינוי. תוכלו לשחק מול DemoBot.pyc שהגיע ביחד עם ה-Starter Kit.

האסטרטגיה שבה נשתמש היא נאיבית, בסיסית מאוד, וכנראה הולכת להפסיד רוב הזמן:

- אם יש צי חלליות שלי טסות בחלל:
  - אל תעשה כלום
- אחרת:
  - אם אין לי כוכב בשליטתי:
    - אל תעשה כלום
  - אם יש פלנטה נייטראלית כלשהו:
    - תקוף את הפלנטה הניטראלית הראשונה עם חצי מהחלליות מהפלנטה הראשונה שלי
  - אחרת, אם יש פלנטת אויב כלשהו:
    - תקוף את פלנטת האויב הראשונה עם חצי מהחלליות מהפלנטה הראשונה שלי

כנראה, לא ברור מה הכוונה ב"ראשון" כשאנחנו מתייחסים לפלנטה ניטראלית או פלנטת אויב - נבין זאת בהמשך.

על מנת שה-Bot שלנו יוכל לשחק, הוא צריך לממש פונקציה בשם `do_turn`. הפונקציה תיקרא בכל תור של המשחק. היא מקבלת את הפרמטר pw שמאפשר לנו לדעת מה מצב המפה ולתת הוראות לביצוע (כלומר, לשלוח ציי חלליות לתקיפה).

כתבו לקובץ TutorialBot.py את הקוד הבא (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
    pass
```

שימרו, והריצו את ה-Bot. תוכלו בוודאי לראות שה-Bot שלכם לא עושה כלום, בעוד DemoBot כובש פלנטה אחר פלנטה. ה-Bot שיצרתם לא עושה דבר בכל תור, כיוון שלא מימשנו דבר בפונקציה `do_turn`.
ראשית, נממש את החלק הבא מהאסטרטגיה:

- אם אין לי כוכב בשליטתינו:
  - אל תעשה כלום

לשם כך, נרצה לבדוק אם יש פלנטה בשליטתינו. הפונקציה הבאה מחזירה לנו את רשימה עם כל הפלנטות שבשליטתי:

```
pw.my_planets()
```

כדי לבדוק אם יש בכלל פלנטות בשליטתינו, נוכל לבדוק את אורך הרשימה:

```
len(pw.my_planets())
```

ועל מנת לסיים את פעולת הפונקציה, נשתמש בפקודה הבאה:

```
return
```

ונרצה לסיים את פעולת הפונקציה אם האורך הוא 0 (אין פלנטות בשליטתי). כלומר:

```
if len(pw.my_planets()) == 0:
      return
```

נרצה שהחלק הזה יופיע כבר בתחילת הפונקציה, כי לפי האסטרטגיה שלנו - ברגע שאין לנו כוכבים בשליטתנו, אין לנו לאיזה כוכב לתת הוראות - אנחנו לא עושים דבר, כלומר מסיימים את ריצת הפונקציה.

הפונקציה do_turn עד כה (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
      if len(pw.my_planets()) == 0:
            return
```

נרצה כעת להמשיך בפיתוח האסטרטגיה הבאה:

- אם אין לי כוכב בשליטתינו:
  - אל תעשה כלום
- אם יש פלנטה נייטראלית כלשהו:
  - תקוף את הפלנטה הניטראלית הראשונה עם חצי מהחלליות מהפלנטה הראשונה שלי

לשם כך, נרצה לבדוק אם יש פלנטה ניטראלית. הפונקציה הבאה מחזירה לנו את רשימה עם כל הפלנטות הניטראליות:

```
pw.neutral_planets()
```

כדי לבדוק אם יש בכלל פלנטות ניטראליות, נוכל לבדוק את אורך הרשימה:

```
len(pw.neutral_planets())
```

נרצה שהאורך יהיה לפחות 1, כלומר שיש לפחות פלנטה ניטראלית אחת.

נשמור למשתנה dest, את הפלנטה הניטראלית הראשונה, כלומר במקום ה-0 במערך. הסדר של הפלנטות נקבע בזמן הרצת המשחק, הוא יכול להשתנות בין תורות ומשחקים שונים. לצורך העניין, תוכלו לחשוב על כך שאנחנו בוחרים פלנטה ניטראלית כלשהי .
ולכן:

```
dest = pw.neutral_planets()[0]
```

הפונקציה do_turn עד כה (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
      if len(pw.my_planets()) == 0:
            return

      if len(pw.neutral_planets()) >= 1:
            dest = pw.neutral_planets()[0]
```

כל מה שאנחנו יודעים כרגע זה את פלנטת היעד. כדי לגשת לרשימת הפלנטות שבשליטתנו, נשתמש בפונקציה:
```
pw.my_planets()
```

ונשמור במשתנה source, את הפלנטה הראשונה שבשליטתי (באותו מובן של "ראשון" ממקודם):
```
source = pw.my_planets()[0]
```

כעת, נרצה לחשב את מספר הספינות לשליחה. אנחנו רוצים שחצי מהחלליות בפלנטה שבשליטתי, כלומר הפלנטה ששמורה במשתנה source, יישלחו אל פלנטת היעד. את מספר הספינות ב-source נוכל לקבל בעזרת הפונקציה:
```
source.num_ships()
```

נשמור למשתנה num_ships את מחצית החלליות:
```
num_ships = source.num_ships() / 2
```

אז מה יש לנו עד עכשיו?
- פלנטת היעד במשתנה dest
- פלנטת המקור במשנה source
- מספר החלליות לשליחה במשתנה num_ships

כל מה שנותר הוא לקרוא לפונקציה issue_order שתיתן את הפקודה לתקיפה:
```
pw.issue_order(source, dest, num_ships)
```

ואם נחבר את הכל, נקבל את ה-Bot הבא (וודאו שההזחה כמו במסמך):
```
def do_turn(pw):
    if len(pw.my_planets()) == 0:
        return

    if len(pw.neutral_planets()) >= 1:
        dest = pw.neutral_planets()[0]

    source = pw.my_planets()[0]

    num_ships = source.num_ships() / 2

    pw.issue_order(source, dest, num_ships)
```

אם תריצו את ה-Bot, תגלו שהוא תוקף רק פלנטות ניטראליות (בדיוק כמו שכתבנו אותו). נוסיף את החלק שמטפל בפלנטות אויב, כלומר את את האסטרטגיה הבאה:
- אם אין לי כוכב בשליטתינו:
  - אל תעשה כלום
- אם יש פלנטה נייטראלית כלשהו:
  - תקוף את הפלנטה הניטראלית הראשונה עם חצי מהחלליות מהפלנטה הראשונה שלי
- אחרת, אם יש פלנטת אייב כלשהו:

- תקוף את פלנטת האויב הראשונה עם חצי מהחלליות מהפלנטה הראשונה שלי

בעצם, נוכל להעתיק את קטע הקוד הבא:

```
if len(pw.neutral_planets()) >= 1:
      dest = pw.neutral_planets()[0]
```

ולהחליף את שתי המקומות שכתוב בהם Neutral במילה Enemy, ונקבל את הבחירה של פלנטת האויב הראשונה! כלומר:

```
if len(pw.enemy_planets()) >= 1:
      dest = pw.enemy_planets()[0]
```

נשים את ה-if-ים במקום הנכון, כדי שקודם נתקוף פלנטה ניטראלית, ורק אם אין אחת כזו, אז נתקוף פלנטת אויב:

```
if len(pw.neutral_planets()) >= 1:
      dest = pw.neutral_planets()[0]

else:
      if len(pw.enemy_planets()) >= 1:
            dest = pw.enemy_planets()[0]
```

אין צורך לשנות את החלק שבוחר פלנטת מכקור, את מספר הספינות ומבצע את התקיפה. בסופו של דבר, קיבלנו את פונקציית do_turn הבאה (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
      if len(pw.my_planets()) == 0:
            return

      if len(pw.neutral_planets()) >= 1:
            dest = pw.neutral_planets()[0]

      else:
            if len(pw.enemy_planets()) >= 1:
                  dest = pw.enemy_planets()[0]

      source = pw.my_planets()[0]

      num_ships = source.num_ships() / 2

      pw.issue_order(source, dest, num_ships)
```

נותר לנו לממש את החלק הבא באסטרטגיה לנו:

- אם יש צי חלליות שלי טסות בחלל:
  - אל תעשה כלום

נצטרך לבדוק אם יש צי חלליות שלנו שטס בחלל. נוכל להשתמש בפונקציה הבאה שמחזירה רשימה של כל ציי

הספינות שלי שטסות:

```
pw.my_fleets()
```

כדי לבדוק אם יש בכלל צי חלליות שלי שטס בחלל, נוכל לבדוק את אורך הרשימה:

```
len(pw.my_fleets())
```

ועל מנת לסיים את פעולת הפונקציה, נשתמש בפקודה הבאה:

```
return
```

כלומר, החלק שמטפל בציי החלליות לפי האסטרטגיה שלנו הוא:

```
if len(pw.my_fleets()) >= 1:
      return
```

נרצה שהחלק הזה יופיע כבר בתחילת הפונקציה, כי לפי האסטרטגיה שלנו - ברגע שיש צי חלליות שלנו באוויר - אנחנו לא עושים דבר, כלומר מסיימים את ריצת הפונקציה.

לבסוף הגרסה הסופית של ה-Bot שלנו (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
      if len(pw.my_fleets()) >= 1:
            return

      if len(pw.my_planets()) == 0:
            return

      if len(pw.neutral_planets()) >= 1:
            dest = pw.neutral_planets()[0]
      else:
            if len(pw.enemy_planets()) >= 1:
                  dest = pw.enemy_planets()[0]

      source = pw.my_planets()

      num_ships = source.num_ships() / 2

      pw.issue_order(source, dest, num_ships)
```

נציג עוד פונקציה אחת שתעזור לנו לדבג את ה-Bot שלנו בזמן הפיתוח שלו:

```
pw.debug(message)
```

הפונקציה מקבלת מחרוזת כפרמטר (`message`) ומדפיסה אותו ב-cmd בזמן ריצת התוכנית. כך, תוכלו לבדוק מה התוכנית שלכם עושה לא בסדר, ולתקן זאת. למשל, בקטע קוד הבא של ה-Bot, אנחנו נדפיס ל-cmd את כמות החלליות שאנחנו שולחים בכל צי (וודאו שההזחה כמו במסמך):

```
def do_turn(pw):
      if len(pw.my_fleets()) >= 1:
```

```
        return

if len(pw.my_planets()) == 0:
        return

if len(pw.neutral_planets()) >= 1:
        dest = pw.neutral_planets()[0]
else:
        if len(pw.enemy_planets()) >= 1:
                dest = pw.enemy_planets()[0]

source = pw.my_planets()[0]

num_ships = source.num_ships() / 2
pw.debug('Num Ships: ' + str(num_ships))

pw.issue_order(source, dest, num_ships)
```

הנה, סיימתם את ה-Bot הראשון שלכם. הוא דיי לא מוצלח.
יאללה, לעבודה!

Reference למימוש Bot

על מנת לכתוב Bot משלכם במשחק PlanetWars, עליכם ליצור קובץ עם השם **MyBot.py** שבו תכתוב את הקוד. בקובץ חייבת להיות הפונקציה **do_turn**, שתופעל בכל תור של המשחק. בפונקציה, תוכלו לגשת למידע על הפלנטות והספינות שכרגע במשחק, ולתת הוראות לשליחת ציי ספינות מפלנטה אחת לאחרת.

הפונקציה do_turn מקבלת את הפרמטר pw, כלומר הפונקציה צריכה להיות מהצורה:

```
def do_turn(pw):
      # --- YOUR CODE HERE ---
```

המידע שניתן לקבל מהפרמטר pw נוגע לפלנטות ולציי החלליות שבמפה. למשל, ע"י קריאה לפונקציה:

```
planets = pw.planets()
```

נקבל במשתנה planet רשימה של כל הפלנטות שבמפה (נייטרלים, של השחקן ושל האויב). נוכל לגשת לתא הראשון ברשימה:

```
first_planet = planet[0]
```

ואז במשתנה first_planet יהיה אובייקט מסוג Planet. בעזרת אובייקט מסוג זה ניתן לגשת למידע הנוגע לפלנטה (כמו מיהו שליט הפלנטה? כמה חלליות יש בפלנטה? ועוד...). למשל, ע"י קריאה לפונקציה:

```
x_first_planet = first_planet.x()
```

נקבל במשתנה x_first_planet את קורדינאטת ה-X של הפלנטה הראשונה במפה.

בטבלה הבאה נמצאות כל הפונקציות של אובייקט מסוג Planet והסבר קצר על מה הן מחזירות:

Class Planet:	פלנטה - הסבר הפונקציות
planet_id()	המספר הסידורי של הפלנטה
owner()	שליט הפלנטה (0 - ניטרלי, 1 - שחקן, 2 - אויב)
num_ships()	מספר החלליות שבפלנטה
growth_rate()	קצב גידול החלליות בכל תור בפלנטה
x()	קורדינאטת ה-X של הפלנטה
y()	קורדינאטת ה-Y של הפלנטה

באופן דומה, ניתן לקבל מ-pw את כל ציי החלליות שנמצאות כרגע על המפה, בעזרת הקריאה לפונקציה:

```
fleets = pw.fleets()
```

נקבל במשתנה fleets רשימה של כל ציי הספינות שנמצאות על המפה (של השחקן ושל האויב). נניח וכרגע יש שני ציי ספינות שנעים במפה, נוכל לגשת לצי השני:

```
second_fleet = fleets[1]
```

ואז במשתנה `second_fleet` יהיה אובייקט מסוג `Fleet`. בעזרת אובייקט מסוג זה ניתן לגשת למידע הנוגע לצי החלליות (כמוו מי שלח את הצי החלליות? מהי פלנטת היעד שלו? כמה חלליות יש בצי? ועוד...). למשל, ע"י קריאה לפונקציה:

```
second_turns_remaining = seconed_fleet.turns_remaining()
```

נקבל במשתנה `second_turns_remaining` את מספר התורות שנותרו עד להגעתו של צי הספינות השני לפלנטת היעד.

בטבלה הבאה נמצאות כל הפונקציות של אובייקט מסוג `Fleet` והסבר קצר על מה הן מחזירות:

Class Fleet:	צי חלליות - הסבר הפונקציות
owner()	בעל צי החלליות (1 - שחקן, 2 - אויב)
num_ships()	מספר החלליות שבצי
source_planet()	פלנטת המקור של צי החלליות
destination_planet()	פלנטת היעד של צי החלליות
total_trip_length()	אורך המרחק של הטיסה
turns_remaining()	מספר התורות שנותרו עד הגעה לפלנטה היעד

לאובייקט `pw` ישנן פונקציות נוספות מעבר לשתיים שראינו למעלה: `Planets`|`Fleets`-. פונקציות אלו מאפשרות לגשת למידע נוסף או מסונן.

`pw.planets()`

מחזירה רשימה של כל הפלנטות (שלי + ניטראלי + אויב) שבמפה

`pw.fleets()`

מחזירה רשימה של כל ציי החלליות (שלי + אויב) שבמפה

`pw.issue_order(source_planet, destination_planet, num_ships)`

שלח **`num_ships`** מפלנטת מקור **`source_planet`** לפלנטת יעד **`destination_planet`**

זוהי בעצם הפונקציה היחידה שבעזרתה נותנים פקודות לביצוע. אם לא קוראים לפונקציה ב-do_turn, אז ה-Bot לא יבצע כל פעולה.

ה-Bot מפסיד את המשחק אוטומטית, במידה ו:

- פלנטת המקור זהה לפלנטת היעד
- נשלחות יותר חלליות מאשר הכמות שנמצאות בפלנטת המקור
- פלנטת המקור אינה בשליטת השחקן

ניתן לקרוא לפונקציה `issue_order` ככל שתרצו בתור יחיד, כל עוד מספר החלליות הכולל בציים שנשלחים מפלנטה, לא גדולים מכמות החלליות שנמצאות בה.

`pw.debug(message)`

מדפיס למסך את המחרוזת **message**
פונקציה חשובה מאוד לצורך דיבוג.

`pw.distance(source_planet, destination_planet)`

מחזירה את המרחק בין פלנטה **`source_planet`** לפלנטה **`destination_planet`**

`pw.turn_number()`

מחזירה את מספר התור הנוכחי במשחק
ספירת התורות מתחילה מ-1

`pw.num_planets()`

מחזירה את מספר הפלנטות שבמפה

`pw.my_planets()`

מחזירה רשימה של כל הפלנטות שבשליטתי שבמפה

`pw.neutral_planets()`

מחזירה רשימה של כל הפלנטות הניטראליות שבמפה

`pw.enemy_planets()`

מחזירה רשימה של כל הפלנטות שבשליטת האויב שבמפה

`pw.not_my_planets()`

מחזירה רשימה של כל הפלנטות שלא בשליטתי (נירטאלי + אויב) שבמפה

`pw.get_planet(planet_id)`

מחזירה אובייקט פלנטה עם מספר סידורי `planet_id`

`pw.num_fleets()`

מחזירה את מספר ציי החלליות שבמפה

`pw.my_fleets()`

מחזירה רשימה של כל ציי החלליות שלי שבמפה

`pw.enemy_fleets()`

מחזירה רשימה של כל ציי החלליות של האויב שבמפה

`pw.get_fleet(fleet_id)`

מחזירה אובייקט צי חלליות עם מספר סידורי `fleet_id`

חוקים (רשימה חלקית)

- משחק מסתיים אם מתקיים לפחות אחד מהתנאים הבאים:
  - נותר רק שחקן אחד במשחק
  - עברו 1000 תורות

- אם מסיימים את התור ה-1000, השחקן עם הכי הרבה חלליות מנצח

- אם ה-Bot של שחקן קורס או מבצע פעולה לא חוקית - הוא מפסיד

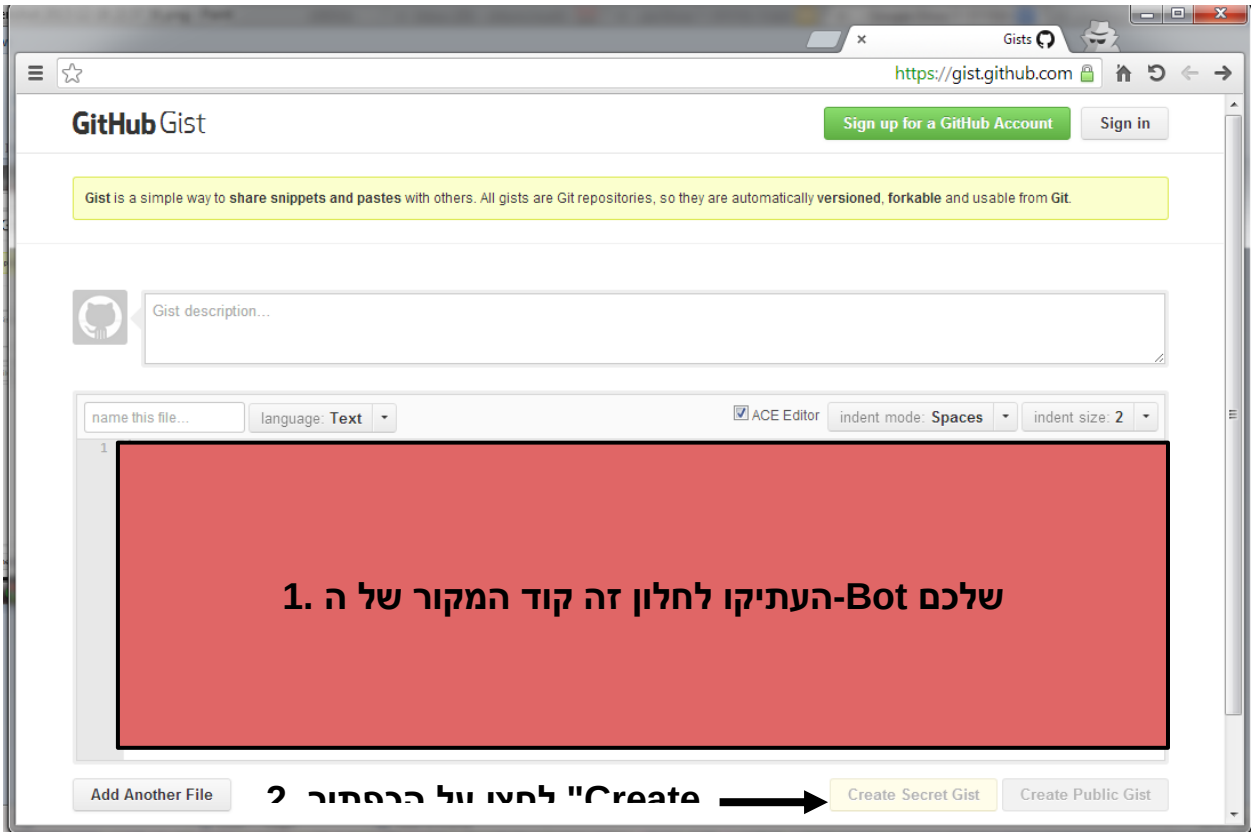
- ניקוד לסיבוב
  - 2 - ניצחון
  - 1 - תיקו
  - 0 - הפסד

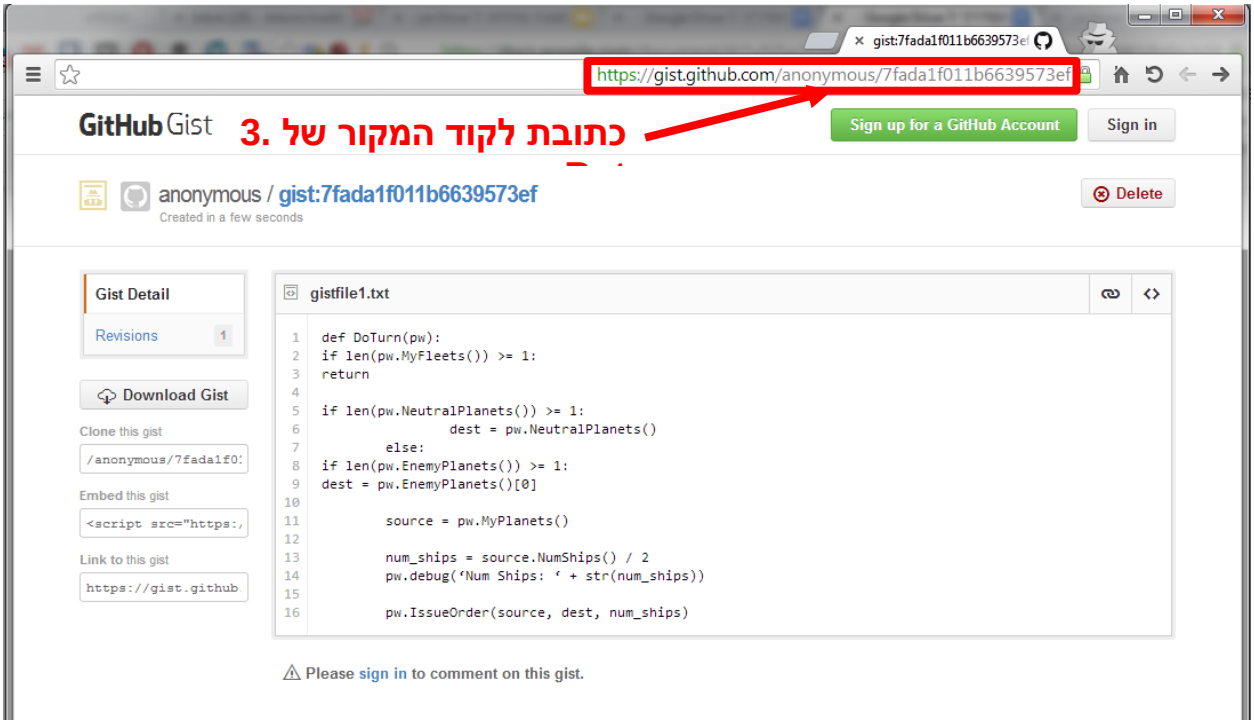
העלאת קוד המקור של ה-Bot ל-Gist

על מנת שה-Bot שכתבתם יוכל להשתתף בתחרות, עליכם להעלות את קוד המקור שלו לאינטרנט, כדי שיהיה נגיש למנוע המשחק. אנחנו נשתמש בשירות Gist של Github שמאפשר לשתף טקסט וקטעי קוד (המכונים Gist-ים) בצורה פשוטה ומהירה.

שימו לב: על מנת להשתמש בשירות Gist עליכם להיות בעלי משתמש ב-github ומחוברים. במידה ואין לכם משתמש - פתחו אחד באמצעות לחיצה על לחצן ה-sign in.

היכנסו לכתובת https://gist.github.com, בצילומי המסך הבאים תוכלו ללמוד את השלבים לשמירת טקסט/קטעי קוד וקבלת כתובת שמפנה אליו:





<u>הנחיות</u>

- את כתובת ה-Bot תצטרכו להכניס באתר התחרות.
- עבור כל גרסה חדשה של ה-Bot שתרצו שתתמודד בתחרות, תצטרכו ליצור Gist חדש, שעבורו תקבלו כתובת חדשה שגם אותם יש לעדכן באתר התחרות.
- במידה ותרצו, תוכלו להירשם ל-Github. כך, תוכלו לערוך את אותו ה-Gist בלי שכתובתו תתחלף (ולא תצטרכו לעדכן את הכתובת באתר התחרות).
- לאחר שעדכנתם את פרטי ה-Gist באתר, תוכלו במסך הראשי ללחוץ על "הורדה לבדיקה" כדי לוודא שהקובץ שהעלתם עובד והוא הקובץ הנכון.