KAB T!  
פרויקט גמר במגמת סייבר

ארז עתיר  
21310794

תיכון הנדסאים הרצליה

ערן רוזנק

25/04/2020

# תוכן עניינים

[תוכן עניינים 2](#_Toc40579277)

[מבוא 3](#_Toc40579278)

[רקע לפרויקט 3](#_Toc40579279)

[תיאור המוצר 5](#_Toc40579280)

[מבט אישי 6](#_Toc40579281)

[תהליך הפיתוח 6](#_Toc40579282)

[אתגרים בפיתוח 6](#_Toc40579283)

[הערכת הפתרון 7](#_Toc40579284)

[סביבת עבודה 8](#_Toc40579285)

[תיחום הפרויקט 8](#_Toc40579286)

[תיאור ממשק משתמש 9](#_Toc40579287)

[מחשב מורה 9](#_Toc40579288)

[מחשב תלמיד 15](#_Toc40579289)

[מדריך למפתח 19](#_Toc40579290)

[במבט על 19](#_Toc40579291)

[במחשב הסרבר 19](#_Toc40579292)

[קבצים 19](#_Toc40579293)

[עיצוב 20](#_Toc40579294)

[טיפול בריבוי משתתפים 22](#_Toc40579295)

[מחלקה - שחקן 22](#_Toc40579296)

[מהלך משחק – מאחורי הקלעים 23](#_Toc40579297)

[מציאת הסרבר 24](#_Toc40579298)

[textbox.py 24](#_Toc40579299)

[OutputBox 24](#_Toc40579300)

[InputBox 25](#_Toc40579301)

[ButtonBox 28](#_Toc40579302)

[במחשב הקליינט 30](#_Toc40579303)

[קבצים 30](#_Toc40579304)

[מציאת הסרבר 30](#_Toc40579305)

[התחברות 31](#_Toc40579306)

[ספריית הקליינט 31](#_Toc40579307)

[עיצוב 35](#_Toc40579308)

[test\_server.pyw 38](#_Toc40579309)

[קבצים 38](#_Toc40579310)

[עיצוב 38](#_Toc40579311)

[פונקציונאליות 38](#_Toc40579312)

[ממשק משתמש 39](#_Toc40579313)

[בסיסי נתונים 40](#_Toc40579314)

[רפלקציה אישית 41](#_Toc40579315)

# מבוא

במסגרת מגמת סייבר בתוכנית גבהים למדנו קישורים רבים. בכיתה י' למדנו את שפת המכונה אסמבלי וכמו כן התנסנו בעשיית פרויקט גמר. בכיתה י"א עבדנו בשפה פייטון ולמדנו על רשתות ועל התקשורת בן מחשבים. השנה למדנו על מערכת ההפעלה של החשב ועל הקבלת קוד.

עם לימודינו במגמה נדרשנו להכין פרויקט המשלב אלמנטים שונים אותם למדנו בשנים האחרונות. על הפרויקטים עבדנו כל השנה האחרונה או אפילו יותר. אני גאה להציג בפניכם את הפרויקט שלי – קבוט.

במסמך הבא תוכלו לשמוע קצת על תהליך העבודה שלי: מן תמצות קצר של כל הלילות הלבנים, התיסכולים וההצלחות של השנה האחרונה, את קו המחשבה שלי בקוד, את הסיפוק של הפרודקטיביות ואת היאוש מהקורונה.

מקווה שיהיה לכם קריאה מהנה!

# רקע לפרויקט

בכיתה י' קבלנו משימה במהלך השנה בה כל אחד היה צריך ללמד את הכיתה פקודה מסויימת באסמבלי. לאחר מכן כל אחד הכין קהוט על הפקודה שלו שכל הכיתה השתתפה בו. בסוף כיתה י"א, לאחר שסיימנו את החומר של רשתות, קיבלנו משימה דומה בה כל אחד קיבל סיפרייה בפייטון שקשורה לתחום אחר בו לא יצא לנו להתנסות במגמה ועליו ללמד את הכיתה את התחום ואת היישומים שלו בפייטון.  
אני קיבלתי ללמד את הכיתה איך לעבוד בספרייה Pygame.  
לכבוד ימים עברו החלתי לעשות מגניב ולהכין קהוט על הסיפריה שקיבלתי. ההבדל היחידי הוא שלא הכנתי קהוט על פייגיים, הכנתי קהוט בפייגיים. ישבתי שעות רבות וכתבתי בעצמי קהוט בעזרת פייגיים ובעזרת החומר שלמדנו השנה ברשתות.  
בקהוט היה טוויסט נחמד – לא תיכנתי קליינט. את כל תיקשורת הנתונים שצריכה להתבצע בן המחשב המריץ את החידון ובן המחשבים המתחברים לחידון הפכתי ל-2 סיפריות פייטון נוחות לשימוש בשם client ו-server. כל תלמיד קיבל את הסיפריה קליינט, סיפריה שתיכנתתי שמטפלת בעצמה בקלט ופלט של טקסטים בתוך פייגיים, טמפליית של קליינט של הקהוט שעובד דרך הטרמינל ובו חלוקה ברורה לכל המסכים שצריכים להיות בקליינט ומה צריך להיות בכל אחד מהם וסרבר לבדיקה שמריץ משחק מדומה לוקאלית ומאפשר לך לעבור באופן שרירותי בן המסכים של הקליינט. בעזרת מה שלימדתי אותם על השימוש בפייגיים כל אחד היה צריך לתכנת בעצמו את התוכנה שדרכה הוא יתחבר לקהוט שלי. לדבר הוקצה שיעור כפול. בשעה ראשונה לימדתי את התלמידים לעבוד עם פייגיים ושניה ערבתי בניהם ועזרתי לכל אחד לכתוב את הקליינט שלו. התלמידים ממש לא עמדו בזמנים, אבל זאת לא הייתה בעיה, מכיוון שהטמפליית שהם קיבלו ועבדו בתוכו בנוי ככה שכל מסך שהם לא מספיקים לתכנת ממשיך לתפקד כרגיל דרך הטרמינל. אם זאת, ערן ראה שהרבה מהתלמידים מאוד התלהבו מהעשייה והתחילו לתכנת קליינטים ממש מגניבים. ערן אהב את הרעיון ונתן לי שבוע לעבוד עם התלמידים על הליינטים שלהם. במשך כל השיעורים באותו שבועה התלמידים עבדו על הקליינטים שלהם ואני עזרתי והדרכתי את מי שהיה צריך. זה היה שבוע מאוד כיף. לבסוף לרוב הכיתה היה כבר קליינט מוכן ומתפקד ושיחקנו קהוט עם כל הכיתה כאשר כל תלמיד שיחק עם משתמש שהוא כתב בעצמו.

שמתי את הקהוט מאחורי ושכחתי ממנו.

בתחילת שנה התחלתי לעבוד על פרוייקט אחר לגמרי. רעיון שהיה לי בשנה שעברה כחלק מפרוייקט תעשידע שעשינו עם החברה מיקרופוקוס. אני בתקופה זו הייתי רץ הרבה לבד ואהבתי לשמוע תוך כדי מוזיקה ותמיד הייתי בוחר שירים ככה שהקצב שלהם יתאים לקצב שבו אני רץ. הדבר העלה בי רעיון: למה שהטלפון שלי לא יבחר בעצמו שירים לפי הקצב שאני רץ. קראתי איזה מחקר שהראה שכאשר המוזיקה שאליה אתה מקשיב מצטלבת עם קצב הצעדים שלך הדבר יכול לשפר ביצועים בכ-15 אחוז. בדקתי פיתרונון קיימים בשוק וישנם הרבה אפליקציות שמציעות לך שירים לפי ה-BPM שליהם אבל בכולם היה 2 בעיות שהפריעו לי. בכולם השירים הוצעו מתוך מאגר השירים של האפליקציה ולא מתוך סיפריית השירים שלך וכולן הניחו שאתה יודע באיזה BPM אתה רץ. אז זה מה הפרויקט שהתחלתי לתכנת. לפרויקט קראו BitBot ומה שהוא היה אמור לעשות זה להיות פלטפורמה כמו ספוטיפי כזה רק שהיא גם מנתחת לך את כל השירים ומציגה לך את ה-BPM שליהם. בנוסף הקליינט בטלפון היה מזהה את קצב הריצה שלך אוטומטית ומנגן לך שירים בהתאם. החלק הראשון של העבודה היה לתכנת גלאי BPM. בעזרת סיפריות של עבודה עם קול בפייטון הצלחתי לסיים חלק זה תוך שבועיים בערך. הגלאי עבד טוב אבל לא מושלם. הפסקתי לעבוד עליו כדי לבדוק את הדבר הכי חשוב מבחינתי בפרויקט – שהטלפון ידע לזהות לבד את ה-BPM של הצעדים שלך. פה התחילו הבעיות. כניראה שאין אפליקציה קיימת בשוק שעושה דבר כזה מפני שהדבר הינו בלתי אפשרי.  
חודש שלם ניסיתי כל מיני שיטות לזהות את קצב הריצה. שימוש בג'יירו, באקסלומטור ב-GPS, שילובים של השלושה. אף פתרון לא היה מדוייק מספיק בשביל צרכי הפרויקט. השגיאה תמיד הייתה גדולה מידי. לבסוך הגעתי למסכנה שחיישני הטלפון פשוט אינם מדוייקים מספיק בשביל לעשות דבר שכזה. התחלתי להסתכל על פתרונות אחרים כמו לקנות חיישן קסלומטור אחר, או לשים כפתור בנעל ואז לחבר אותם לתוכנה דרך ארדווינו או משהו. בסוף הבנתי שלהזמין משהו כזה מחו"ל יקח מספר שבועות ובזמן הזה אני תקוע בלי כלום. היה לי קשה אבל וויתרתי על הפרויקט. הבנתי שאינו יזים במסגרת הזמן שיש לי.

עברו כבר כחודשיים מתחילת העבודה ולא היה לי כלום.  
אז ישבתי עם לחשוב מה אני יכול לעשות ואז החלטתי שאני אגיש את הקהוט כפרויקט הגמר שלי. השקעתי בקהוט הזה מאות שעות עבודה. היה לי עוד הרבה עבודה עליו כדי להביא אותו מרמה של משהו שעשיתי סתם לכיף וכפעילות לכיתה לרמה של פרויקט גמר, אבל רוב הפרויקט כבר היה מוכן ולפחות אני לא מתחיל מאפס. ככה הקהוט נעשה פרויקט הגמר שלי.  
לפרויקט קראתי Kaboot, כדי לא להסתבך עם זכויות יוצרים כמובן.



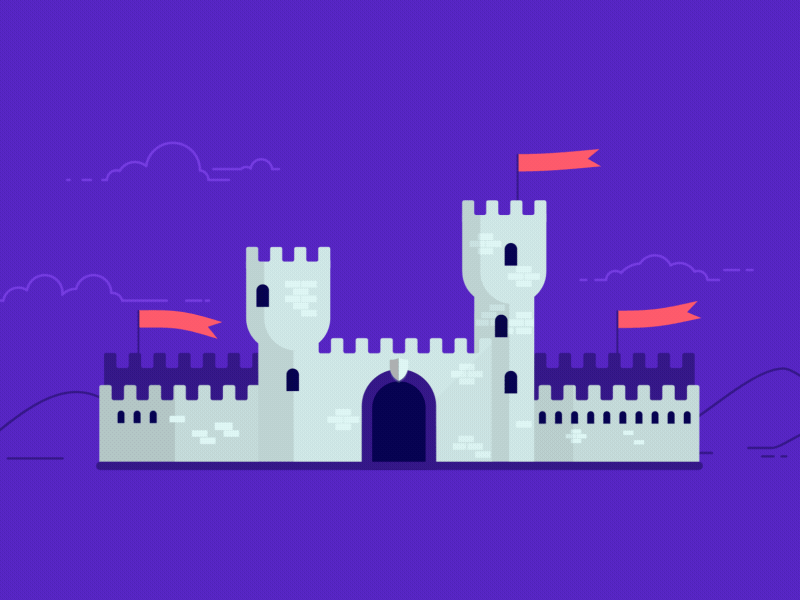
# תיאור המוצר

שם המוצר הוא קבוט. זו היא תוכנת מחשב דרכה מורה יכול לבנות חידונים ושאלונים ולהריץ אותם בכיתה. כל התלמידים בכיתה יכולים להתחבר לחידון ולהתחרות על המקום הראשון. בדומה לפלטפורמה הקיימת - קהוט.

במהלך הפיתוח אחת המטרות המרכזיות בפרויקט הייתה להכין פלטפורמה בה נוח ללמוד ולהתנסות בספרייה פייגיים. להראות לתמידים כמה נוח זה וכמה מהיר מגיעים לתוצר. לתת להם את תחושת הסיפוק שהם בנו משהו מועיל ושימושי דרך הספרייה ושהם כבר משתמשים בו כדי לשחק. בנוסף מעוד אהבתי את הספרייה. היא מאוד נוחה לשימוש ועוד מטרה בפרויקט הייתה ללמוד את הספרייה יותר לעומק ולהתנסות בה.

כשהמשכתי עם הקהוט השנה המטרה המרכזית הייתה להביא את המוצר לרמה שהוא עומד לבד וכל אחד יוכל להשתמש בו.

להלן דברים יחודיים למוצר שפיתחתי: במוצר זמן ההצטרפות של הקליינטים לחידון קצר משמעותית מכיוון שאין קוד כניסה אל המשחק. המשחקים מאובטחים בכך שהם חוסמים ניסיונות התחברות שאינם מאותה כיתה. המשחק רץ אוטומטית על הרשת המקומית וכל מי שפותח את התוכנה של הקליינט באותו כיתה מצטרף אליו אוטומטית. בנוסף, כאשר יש שאלה בחידון התשובות מוצגות גם כן על מסך המחשב של המשתמש, ככה הרבה יותר נוח למשתמש לראות את התשובות ולהגיב במהרה.

המערכת מאפשרת למידה חווייתית ומשמעותית לתלמידי בית ספר ומהווה דרך נהדרת להעביר חומר בצורה מעניינת וכיפית בחברות או סתם להנות עם חברים בשאלוני טריוויה.

# מבט אישי

תהליך הפיתוח  
מחקר ראשון שעשיתי היה ללמוד להתמש בספריית PYGAME לצורך הפרוייקט. למדיתי את הספרייה בעיקר דרך המדריך של תוכנית גבהים.

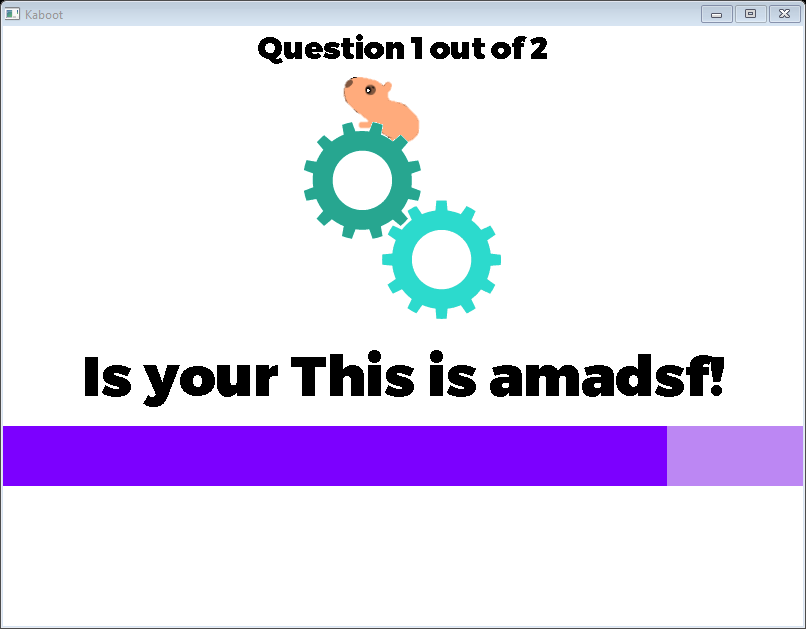
מחקר שני היה ללמוד להשתמש בספריית TKINTER בה השתמשתי להכנת טסט-סרבר וכלי דיבאג אחרים. למדתי מסרטונים באינטרנט. מצאתי תוכנה מאוד שימושית בשם Page שעוזרת בעיצוב ואיתחול של חלונות Tkinter.

לאחר מכן חקרתי על סוגים שונים של DATABASES כדי להחליט באיזה אחד כדאי לי להשתמש לצורך שמירת השאלונים ולאחר שהחלטתי ש-JSON הוא הפורמט שיהיה לי הכי קל לעבוד איתו למדתי דרך סירטונים באינטרנט איך לעבוד עם ספריות שמתעסקות עם קבצי JSON בפייטון.

התחלתי את העבודה בלבנות תשתית תקשורת בן הסרבר, אשר רץ על המחשב בו מריצים את השאלון, ובן הקליינטים, אשר רצים במחשבי המשתתפים בחידון. בניתי פרוטוקלים למציאת הסרבר ברשת המקומית, להתחברות אליו, לשליחת שאלות, תשובות ופקודות אחרות והפכתי את כל התשתית לשתי סיפריות נוחות לשימוש: server ו-client. לאחר מכן התחלתי לעבוד על העיצוב של הסרבר ובניתי אותו מסך מסך תוך כדי שאני משתמש בספריות שתיכנתתי וקלייט בקונסול כדי לבדוק את העבודה ולדבאג. לאחר מכן התשמשתי בספריה TKINTER כדי לבנות תוכנה בשם test\_server. הטסט סרבר מאפשר הרצה של כל פקודות סרבר ושליחתם לקליינט בלי שבאמת קרתה הסיטואציה במשחק, או מבלי שבאמת רץ משחק. השתמשתי בטסט-סרבר כדי להתחיל לעצב ולתכנת את הקליינט מסך מסך. לאחר שסיימתי את הקליינט העברתי את השאלון שהיה מתוכנת בתוך הקוד לקובץ JSON ותכנתתי פונקציות בסרבר שיודעות לקרוא ולפתוח את הקובץ. לאחר מכן התחלתי לעבוד על תוכנה נפרדת שדרכה אתה יכול לערוך את הכתוב בחידון. בתוכנה מסך הדומה למסך של שאלה רצה אבל הכל עריך. לאחר מכן תכנתתי תוכנה שמקשרת בן התוכנה שמריצה חידונים לבין זאתי שעורכת אותם..

אירוע מעניין שהיה במהלך הפיתוח הוא כמובן המשחק שעשינו בו כל אחד תכנת קליינט מעצמו לפרויקט. לקחתי השראה מהעיצוב של הקליינט הכי יפה לאחר מכן כשתיכנתתי קליינט בעצמי.

אני חושב שההתמודדות שלי עם קשיים הייתה בהתחלה ממש גרועה. אני הייתי צריך להבין הרבה מראש שביטבוט לא יעבוד ושאני צריך פרויקט אחר אבל סתם התמעמתי עם זה עוד חודש. חודש של ניסוי וטעיה שלא הוביל לשום מקום. בזבוז זמן.  
אחרי שעברתי לקבוט לא היה הרבה קשיים מאוד משמעותיים.  
נעזרתי באינטרנט כדי לפתור שגיאות מוזרות שלא הבנתי בספריות ושברתי את הראש שלי על באגים עד שהבנתי למה דברים לא עובדים ואז תיקנתי אותם.

הערכת הפתרון  
אני מאוד שמח מהמוצר שיצא. אני חושב שהפרויקט באמת ברמה גבוהה. עבדתי המון זמן על הקוד ועל הסרבר למען שלא יהיו באגים. המוצר מעוצב מאוד יפה ואסטטית.

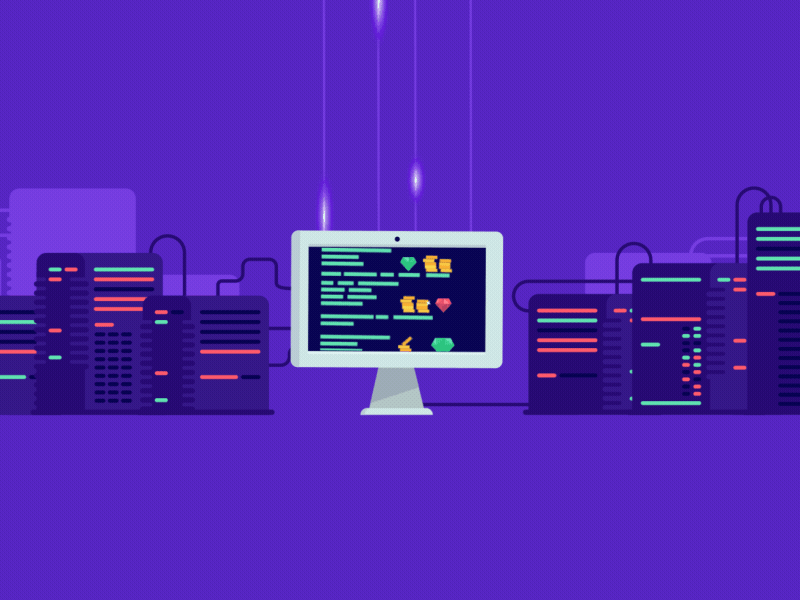
# סביבת עבודה

הפרויקט נכתב כולו בשפת התכנות פייטון.  
ה-IDE בו השתמשתי הוא PyCharm של JetBrains.  
לצורך מעקב אחרי גרסאות ושמירה וכדי לאפשר לי להמשיך לעבוד על הקוד במחשבים של בית ספר השתמשתי ב-GitHub.  
לצורך בקרה של תקשורת הנתונים בן הסרבר לקליינט  
השתמשתי ב-Wireshark.  
כמו כן השתמשתי ב-Tkinter כדי לבנות סרבר מדומה שדרכו יכולתי לבדוק את הקליינט מבלי באמת להריץ משחק.

# תיחום הפרויקט

הפרויקט עוסק בהמון תחומים. נעשה בו שימוש כמעט בכל מה שלמדנו בשנתיים האחרונות במסגרת המגמה למשל בהקבלת קוד, מנעולים, בסאבפרוססים, בממשק תכנות יישומים של ווינדווס, בתקשורת נתונים, TCP, UDP, select, בבסיסי נתונים ובשני ספריות שונות ליצירת GUI, פייגיים ו-Tkinter .

להל"ן הספריות בהם נעשה שימוש בפרויקט:

pygame,  
Tkinter, tkMessageBox, tkFileDialog,  
socket, select, uuid,  
thread, threading, subprocess,  
win32api, win32con, win32gui,  
os,  
sys,  
traceback,  
time,  
json,  
base64

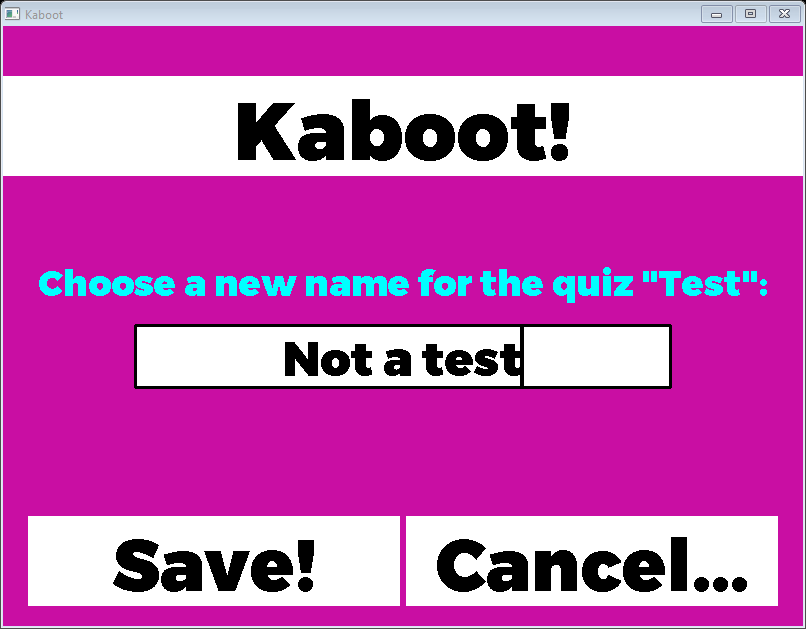
# תיאור ממשק משתמש

למשחק יש שתי תוכנות שונות. תוכנת מרכזית אותה תתקין על מחשב המורה/המחשב שעליו אתה רוצה להריץ את החידון, ותוכנה לשחקנים שמתחברים אל החידון. שני התוכנות מצורפות בווינראר. על מנת להריץ את המשחק בצורה תקינה עליך להעתיק את כל התיקיות מהקובץ לשים אותן באותה תיקיה, שכן בתוכן קבצי שמע, תמונות וספריות שהפרויקט צריך כדי לעבוד. מבלי ששינית כלום בתיקיות, והשארת את כל הקבצים ביחד לחץ על הקובץ Kaboot.py והתחל לשחק!

## מחשב מורה

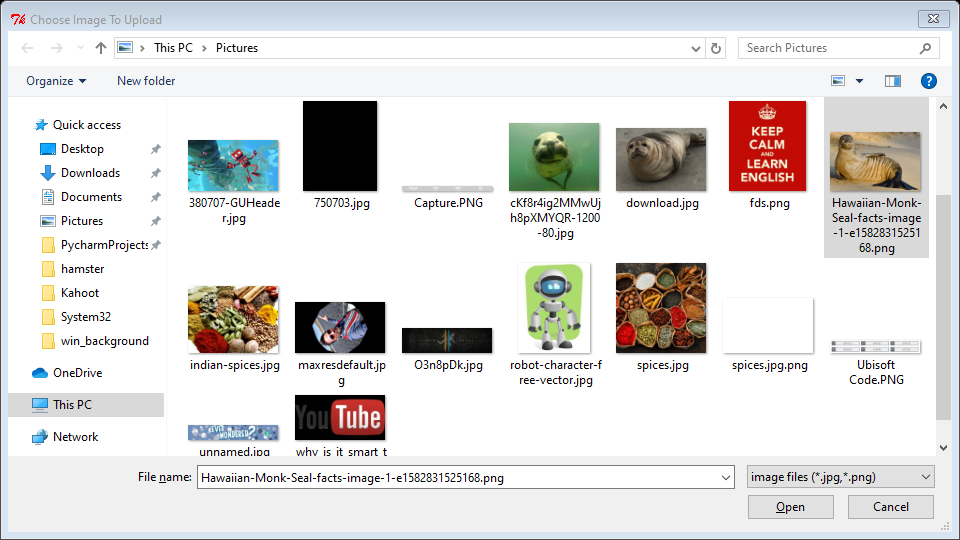
במחשב של המורה כאשר המשתמש פותח את התוכנה Kaboot יפתח לו חלון בו תופיע רשימה של כל החידונים הקיימים במחשב. הוא יכול לדפדף בניהם עם החצים בצד שמאל. הוא גם יכול ללחוץ על חידון כדי לסמן אותו וא\ לבחור לערוך אותו, למחוק אותו, לשנות את שמו או להריץ אותו כמשחק קבוט.

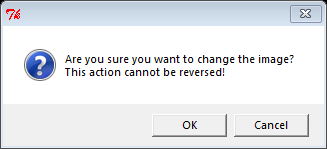




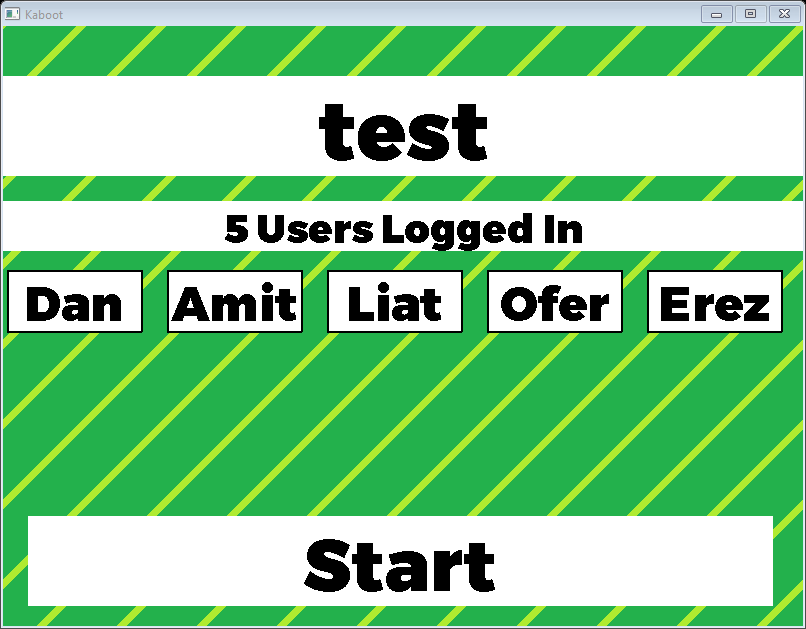
אם הוא נכנס לעריכה של החידון יפתח לו תוכנה בה ניתן לערוך את כל השאלות בחידון.  


כל השדות עריכים. אתה יכול לשנות את המספרים, את התשובות ואת השאלה עצמה. כדי לעשות זאת לחץ על השדה הרצוי וחכה שיופיע שם הסמן העורך כמו שמופיע בתמונה ליד "Yes!". כדי לשנות את התשובה הנכונה לחץ על הצורה שלידה. הצורה תהפוף לירוקה. התשובה שהצורה שלה ירוקה זאת התשובה הנכונה לשאלה. יש גם כן אפשרות לעלות תמונה לשאלה או למחוק את התמונה הקיימת. אם אין תמונה בחידון אז מופיע פשוט הכיתוב KABOOT!. אם בוחרים להעלות תמונה נפתח מסך בו אפשר לבחור את הקובץ התמונה שרוצים לעלות.  
לאחר מכן נפתח חלון בו אפשר להחליט איך לחתוך את התמונה כך שתתאים לגודל הרצוי.  
כמו שניתן לראות בתמונות בעמוד הבא.





אם הוא בחר להריץ את החידון אז מתחיל משחק קבוט ברשת המקומית.

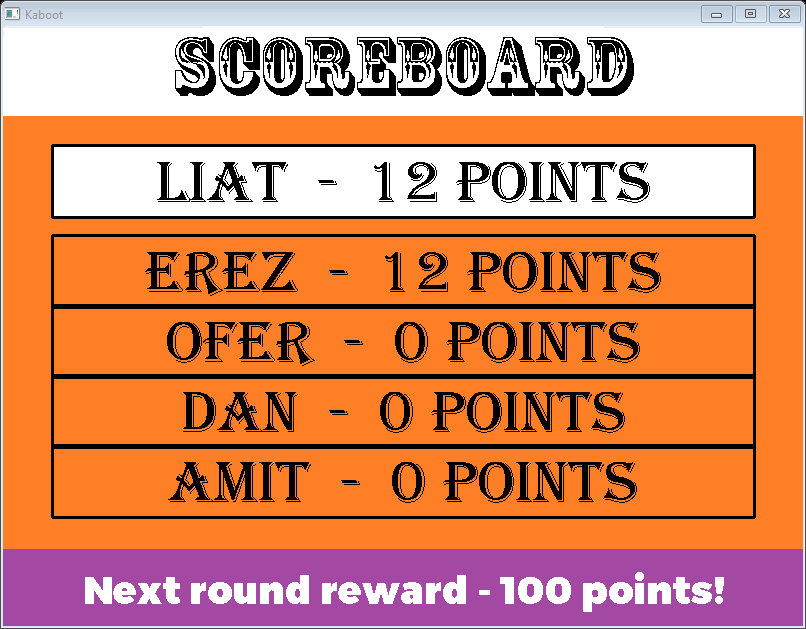
מסך כניסה:  
  
לאחר שכל הכיתה נכנסה למשחק לחץ על הכפתור "Start".

זמן לקרוא את השאלה לפני הרצתה:  


מסך של שאלה:  
  
לאחר שכולם ענו או נגמר הזמן לשאלה יופיע מה התשובה הנכונה וכמה משתתפים ענו על כל שאלה. ריבועי התשובות יוצאים מהמסך במן אנימציה שמדגימה כמה אנשים ענו כל תשובה. בנוסף כתוב בכל ריבוע מספר זה. כל הריבועים נעשים בהירים יותר חוץ מהריבע בו הייתה התשובה הנכונה. לחץ על רווח כדי להתקדם למסך הבא.

A picture containing photo, cat

Description automatically generated

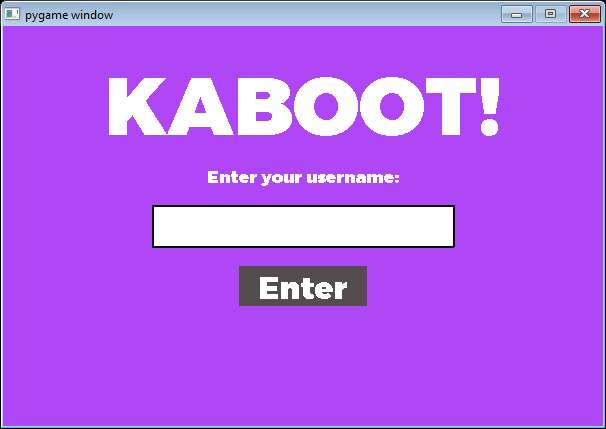
טבלת מובילים:  
  
לחץ על רווח כדי להתקדם שאלה הבאה הבא

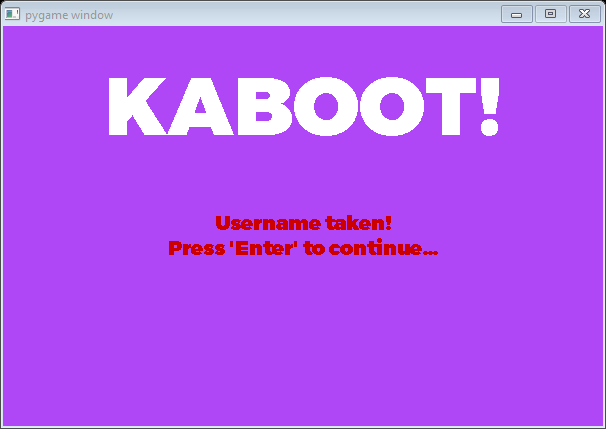
שלושת המסכים האחרונים חוזרים על עצמם שוב ושבו עד שעוברות כל השאלות שבחידון. אחרי השאלה האחרונה אין טבלת מובילים אלה מוכרזים שלושת המנצחים עם אנימציה.A close up of a sign

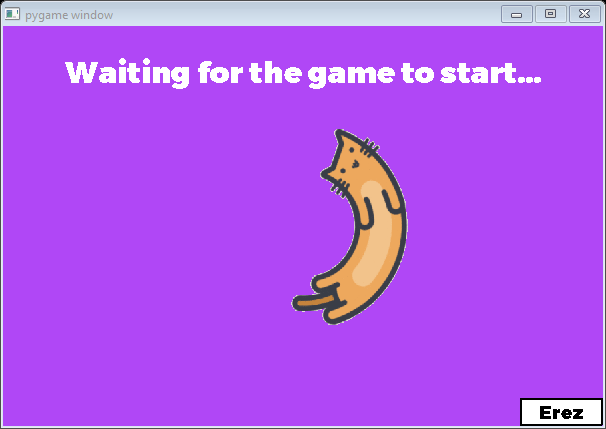
Description automatically generated  
לחץ על רווח כדי לצאת מהמשחק.

## מחשב תלמיד

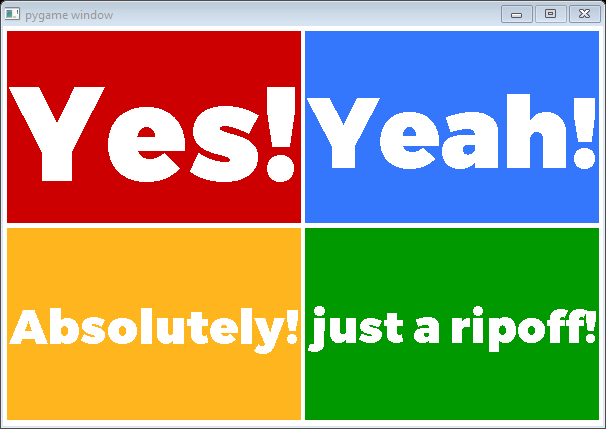
במחשב של התלמידים יש להם תוכנה אחרת ששמה הוא גם כן Kaboot אך כאשר היא נפתח היא אוטומטית מחפשת חידון על הרשת המקומית. אם היא לא מצאה אחד היא מודיע על זה למשתמש ונסגרת.  

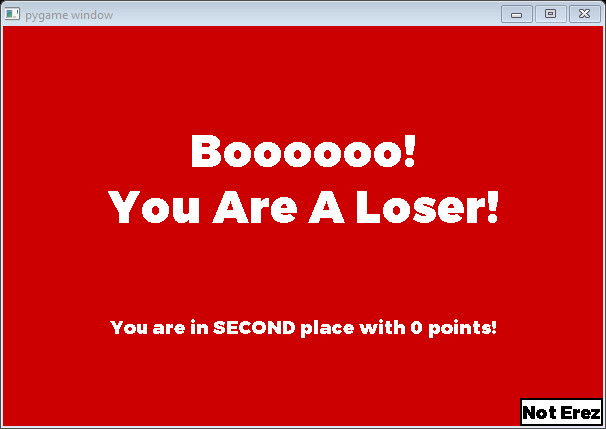

אם נמצא חידון היא מתחברת אליו ומבקשת מהמשתמש לבחור שם משתמש.  


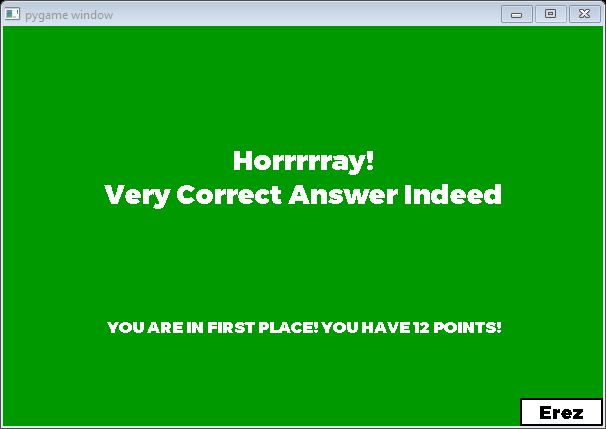
אם שם המשתמש תפוס התלמיד יקבל הודע על כך.  


אם הוא חוקי היא הוא יופיע כמשתמש בחידון ויכנס למסך טעינה עד שיתחיל החידון.  


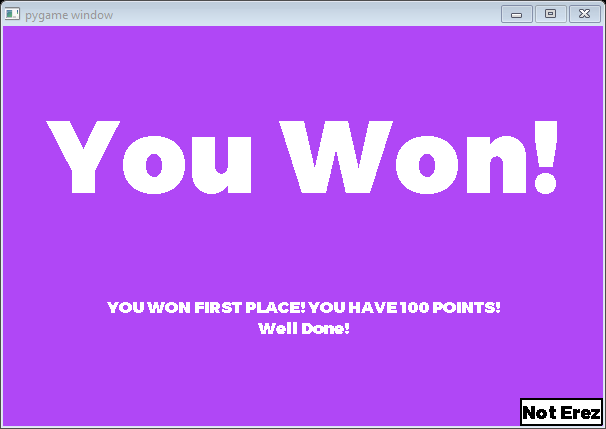
מצד ימין למטה מופיע שם המשתמש.

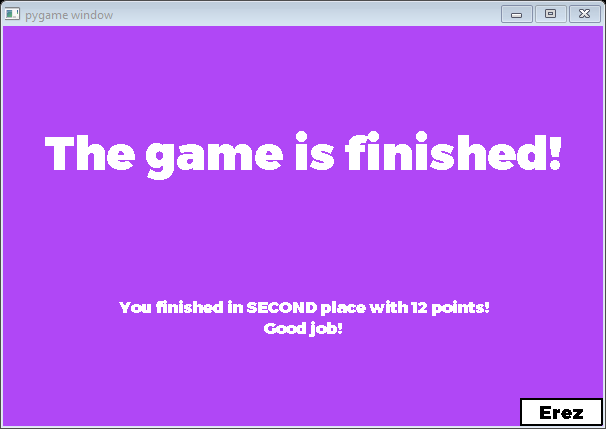
ברגע שיתחילו את החידון תופיע שאלה על הלוח עם זמן לקרוא אותה. ברגע שנגמר הזמן יופיעו ארבע תשובות על המסך של התלמיד. התלמיד צריך ללחוץ עם העכבר על התשובה שהוא חושב שהיא נכונה.   


יופיע לו מסך שמודיע לו אם הוא צדק או טעה. אחרי כל השאלות יופיע לו מסך שמודיע שהסתיים המשחק ובאיזה מקום הוא.



לאחר מכן שלושת המסכים האחרונים חוזרים על עצמם לכל השאלות שבחידון. אחרי השאלה האחרונה מופיע מסך שמודיע למשתמש שנגמר המשחק, באיזה מקום הוא סיים את המשחק ואם כמה נקודות.





# מדריך למפתח

## במבט על

על מחשב אחד, בוא מוצג השאלון, מורץ שרת מרובה משתמשים אליו מתחברים כל שחקני המשחק.

סרבר

קליינט

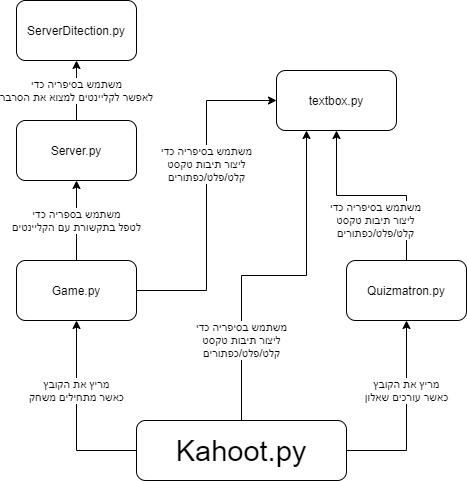
קליינט

קליינט

קליינט

קליינט

קליינט

****

## במחשב הסרבר

### קבצים

### עיצוב

במחשב שבו מורץ הסרבר מנהל המשחק. המשחק מתוכנת ספריה pygame. רוב הקוד בקובץ Kahoot.py אחראי על עיצוב המסכים שכן תיקשורת הנתונים והמעקב אחרי שחקני המשחק כולו נעשה בספריות. הנה הפונקציה שמעצבת את מסך השאלה במשחק הקהוט.

def load\_question(screen, question, photo, answers, qtime):  
 """ ***:param*** screen: gets screen to print on  
 ***:param*** question: gets the question  
 ***:param*** photo: gets a photo if there is one  
 ***:param*** answers: gets all possible answers  
 ***:param*** correct\_answer: gets the number of the correct answer  
 ***:param*** qtime: gets the time for the question  
 ***:param*** qpoints: gets maximum points for answering the question  
 ***:return***: Did player exit the game  
 """ global users, Running  
 # image rc = pygame.image.load(IMAGES\_DIR + "main\\red\_correct.png")  
 rc = resfix(rc)  
 bc = pygame.image.load(IMAGES\_DIR + "main\\blue\_correct.png")  
 bc = resfix(bc)  
 yc = pygame.image.load(IMAGES\_DIR + "main\\orange\_correct.png")  
 yc = resfix(yc)  
 gc = pygame.image.load(IMAGES\_DIR + "main\\green\_correct.png")  
 gc = resfix(gc)  
 a = [0, 367, 403, 367, 0, 484, 403, 484]  
 Rstartx, Rstarty, Bstartx, Bstarty, Ystartx, Ystarty, Gstartx, Gstarty = [int(a[x]/800.\*WIDTH) if x % 2 == 0 else int(a[x]/600.\*HEIGHT) for x in range(len(a))]  
 addedimg = None  
 if photo:  
 addedimg = pygame.transform.scale(pygame.image.load("./files/temp." + photo), (int((665-143)/800.\*WIDTH), int((334-70)/600.\*HEIGHT)))  
 # question  
 question\_text = textbox.OutputBox(screen, question, (WIDTH, int(70/600.\*HEIGHT)), (0, 0), (255, 255, 255), 0, (), (0, 0, 0), "files\\montserrat\\Montserrat-Black.otf") # time  
 start\_time = time.time() time\_passed = time.time() - start\_time answer\_boxes = []  
 for y in range(4):  
 answer\_boxes.append(textbox.OutputBox(screen, text=answers[y], size=(int(335/800.\*WIDTH), int(105/600.\*HEIGHT)), place=(int(int(60/800.\*WIDTH) + (WIDTH / 2) \* (y % 2)), int(372/600.\*HEIGHT) + int(120/600.\*HEIGHT) \* int(y / 2)),  
 color=None, text\_color=WHITE, font="files\\montserrat\\Montserrat-Black.otf")) timerText = textbox.OutputBox(screen, text=str(qtime), size=(int((753-693-6)/800.\*WIDTH), int((235-175)/600.\*HEIGHT)), place=(int((43+3)/800.\*WIDTH), int(175/600.\*HEIGHT)),  
 color=None, text\_color=WHITE, font="files\\montserrat\\Montserrat-Black.otf")  
 timerTextHeader = textbox.OutputBox(screen, text=" Seconds:", size=(int(142/800.\*WIDTH), int((237-177+100)/600.\*HEIGHT)), place=(0, int(177/600.\*HEIGHT) - int((237-177+50)/600.\*HEIGHT)),  
 color=None, text\_color=BLACK, font="files\\montserrat\\Montserrat-Black.otf") answerText = textbox.OutputBox(screen, text=str(0), size=(int((753-693-6)/800.\*WIDTH), int((235-175)/600.\*HEIGHT)), place=(int((693+3)/800.\*WIDTH), int(175/600.\*HEIGHT)),  
 color=None, text\_color=WHITE, font="files\\montserrat\\Montserrat-Black.otf")  
 answerTextHeader = textbox.OutputBox(screen, text="Answers: ", size=(int((800-664)/800.\*WIDTH), int((235-175+100)/600.\*HEIGHT)), place=(int(664/800.\*WIDTH), int(175/600.\*HEIGHT) - int((235-175+50)/600.\*HEIGHT)),  
 color=None, text\_color=BLACK, font="files\\montserrat\\Montserrat-Black.otf") pygame.mixer.music.load(OST\_DIR + "question.mp3")  
 pygame.mixer.music.set\_volume(0.4)  
 pygame.mixer.music.play(-1)  
 answers\_amount = 0  
 while time\_passed < qtime and answers\_amount < len(users):  
 events = pygame.event.get()  
 answers\_amount = Server.receive()  
 for event in events:  
 if event.type == pygame.QUIT:  
 pygame.mixer.music.stop() Running = False  
 exit()  
 return True  
 if event.type == pygame.KEYDOWN:  
 # If pressed key is ESC quit program  
 if event.key == pygame.K\_ESCAPE: Running = False  
 exit()  
 return True screen.fill(WHITE)  
 pygame.draw.circle(screen, (201, 14, 163), (int(723 / 800. \* WIDTH), int(206 / 600. \* HEIGHT)), 30)  
 pygame.draw.circle(screen, (201, 14, 163), (int(74 / 800. \* WIDTH), int(206 / 600. \* HEIGHT)), 30)  
 if addedimg:  
 screen.blit(addedimg, (int(143/800.\*WIDTH), int(75/600.\*HEIGHT)))  
 else:  
 textbox.OutputBox(screen, text=" KABOOT! ", size=(int((665-143)/800.\*WIDTH), int((334-75)/600.\*HEIGHT)), place=(int(143/800.\*WIDTH), int(75/600.\*HEIGHT)),  
 color=(201, 14, 163), text\_color=WHITE, font="files\\montserrat\\Montserrat-Black.otf").draw() screen.blit(rc, (Rstartx, Rstarty))  
 screen.blit(bc, (Bstartx, Bstarty))  
 screen.blit(yc, (Ystartx, Ystarty))  
 screen.blit(gc, (Gstartx, Gstarty)) #check\_for\_place(screen, events) # question  
 question\_text.draw() # answers  
 for answer in answer\_boxes:  
 answer.draw() # timer  
 time\_passed = time.time() - start\_time  
 timerText.text = str(int(qtime - time\_passed))  
 timerText.draw()  
 timerTextHeader.draw() answerText.text = str(int(answers\_amount)) if answers\_amount is not None else "0"  
 answerText.draw()  
 answerTextHeader.draw() pygame.display.flip()  
 pygame.mixer.music.stop()  
 return False

עד לכאן הכל זה קביעה של דברים סטאטים במסך כמו, מיקום של טקסטבוקסים, לקיחת תמונות מקבצים, הפעלת מוזיקה ועוד.

חלק זה הוא הלולאה של הרצת המסך עצמו. כאן מדפיסים את כל האובייקטים. הלולאה רצה כל עוד לא עבר הזמן ולא ענו כל המשתתפים

פעולה שקיימת בכל לולאה ואחראית להמשך הפעולה של הסרבר

### טיפול בריבוי משתתפים

התפעול של הסרבר, אסיפת התשובות שמירת הנקודות של השחקינים, כל אלה שקופים לקובץ המרכזי שכן הוא אחראי רק על עיצוב. הוא מקבל פיד מהסרבר ומשאיר אותו בחיים על ידי שימוש הפונקציה Server.receive(). הסרבר משתמש ב-socktets כדי לתקשר עם המשתמשים ומתפל במספר רב שליהם על ידי שימוש בפונקציה select. התקשורת נעשת ב-TCP.

def receive():  
 *"""  
 receives all of the players answers and saves them but doesn't respond yet in addition to answering mandatory requests.* ***:return****: how many players answered yet  
 """* try:  
 if open\_client\_sockets:  
 global \_\_server\_socket  
 rlist, wlist, xlist = select([\_\_server\_socket] + open\_client\_sockets, open\_client\_sockets, open\_client\_sockets, 0.1)  
 for current\_socket in rlist:  
 \_\_single\_user(current\_socket)  
 for current\_socket in xlist:  
 open\_client\_sockets.remove(current\_socket)  
 \_\_send\_\_mandatory(wlist)  
 return answersanount  
 except Exception:  
 \_error()

def \_\_send\_\_mandatory(wlist):  
 *"""  
 Sends waiting messages that are queued to be sent, but only if the client is writable* ***:param*** *wlist: a list of all the writable client sockets* ***:return****: None  
 """* global \_\_mandatory  
 for message in \_\_mandatory:  
 try:  
 client\_socket, data = message  
 if client\_socket in wlist:  
 client\_socket.send(data + '\n')  
 \_\_mandatory.remove(message)  
 except socket.error:  
 open\_client\_sockets.remove(client\_socket)

### מחלקה - שחקן

בשביל המעקב אחרי כל השחקנים משתמש הסרבר בקלאס של שחקן:

class Player:  
 def \_\_init\_\_(self, name, client\_socket):  
 self.name = name  
 self.socket = client\_socket  
 self.score = 0  
 self.answer = None  
 self.connected = True  
  
 def add\_score(self, points):  
 self.score += points  
  
 def submit\_answer(self, answer):  
 self.answer = answer  
  
 def delete\_answer(self):  
 self.answer = None

### מהלך משחק – מאחורי הקלעים

הקליינט יכול לשלוח 5 פקודות שונות כשהפורמט בהם הקליינט שולח את הפקודה הוא: "Command: data".  
הפקודות אותן הקליינט יכול לשלוח הם:  
login – ניסיון להיכנס למשחק, מחזיר לקליינט OK, אם הוא יצתרף בהצלחה ו-TAKEN, אם שם המשתמש שלו תפוס  
answer – תשובה לשאלה הנוכחית  
get\_score – מחזיר לקליינט את כמות הנקודות שיש לו  
get\_place – את המקום שלו  
get\_behind – את שם המשתמש של המשתמש שמקום אחד לפניו

כאשר מתקבל קלט מאחד הקליינטים הסרבר מטפל בו כפי שניתן לראות בפונקציה הבאה:

def \_\_handle\_client\_request(client\_socket, request):  
 *"""  
 Appends a proper respond to the clients request* ***:param*** *client\_socket: the user's socket* ***:param*** *request: what was received from the user* ***:return****: none  
 """* global \_\_mandatory, open\_client\_sockets, answersanount  
 player = [x for x in \_\_players if x.socket == client\_socket]  
 if player:  
 player = player[0]  
 if request[:len("login: ")] == "login: ":  
 if request.split("login: ")[1]:  
 if request.split("login: ")[1] not in [x.name for x in \_\_players]:  
 \_\_players.append(Player(request.split("login: ")[1], client\_socket))  
 \_\_mandatory.append((client\_socket, 'OK'))  
 else:  
 for player in \_\_players:  
 if player.name == request.split("login: ")[1] and not player.connected:  
 player.socket = client\_socket  
 player.connected = True  
 \_\_mandatory.append((client\_socket, 'OK'))  
 \_\_mandatory.append((client\_socket, 'TAKEN')) elif request[:len("answer: ")] == "answer: ":  
 if player.answer is None:  
 player.submit\_answer(int(request.split("answer: ")[1]))  
 answersanount += 1 elif request == "get\_score":  
 \_\_mandatory.append((player.socket, "score: " + str(player.score))) elif request == "get\_behind":  
 if \_\_players.index(player) > 0:  
 \_\_mandatory.append((player.socket, "behind: " + \_\_players[\_\_players.index(player)-1].name)) elif request == "get\_place":  
 \_\_mandatory.append((player.socket, "place: " + str(\_\_players.index(player)+1)))

### מציאת הסרבר

מה שחסר לנו רק להבין לפני שנעבור לבחון את הקוד שאץ בקליינטים זה כיצד הקליינט יודע על איזה מחשב ברשת המקומית רץ השרת. לשם זה תיכנתתי ספרייה בשם ServerDitection. ספריה זאת משתמשת בפרוטוקול UDP כדי לצעוק קריעות ברודקסאט, לכל המחשבים ברשת המקומית, "פה הסרבר".

def server\_emitter():  
 try:  
 server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM, socket.IPPROTO\_UDP)  
 server.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1)  
 # Set a timeout so the socket does not block  
 # indefinitely when trying to receive data.  
 server.settimeout(0.2)  
 server.bind(("", 44444))  
 message = b"Here Be Server: " + IP  
 subnet = subnet\_calculator()  
 while True:  
 server.sendto(message, (subnet, 37020))  
 time.sleep(0.3)

## textbox.py

זוהי ספרייה מאוד שימושית שתכנתתי בא משתמש גם הסרבר וגם הקליינט והיא מטפלת בהדפסה של טקסטים בפייגים לבד. בסיפריה שלוש אובייקטים, אחד להדפסה של טקסט, אחד לקליטה של טקסט ואחד ליצירת כפתורים.   
ישנם שינויים מזעריים בן הטקסט בוקס שבסרבר לבן האחד שבקליינט בשל צרכים שנבעו ספציפית בכל אחד המודולים, לדוגמא רק בסרבר יש אופציה לעשות תיבת קלט שקולטת רק מספרים, רק בקליינט יש אופציה לעשות תיבת קלט שמוגבלת במספר תווים. את הספרייה קיבלו גם הילדים בפעילות שעשיתי בכיתה כדי להקל על חייהם.

### OutputBox

מטפל בהדפסה של טקסטים למסך. ממקם את הטקסט במרכז הקופסא באופן אוטומטי. בוחר את גודל הפונט המקסימלי ככה שהטקסט יכנס בגבולות הקופסא באופן אוטומטי. עושה ירידת שורה באופן אוטומטי. מקל מאוד על השימוש בטקסט בפייגיים שכן ברגיל קשה מאוד לדעת כמה פיקסלים יתפוס הטקסט לפי גודל הגופן שלו.

class OutputBox:  
 *"""  
 Creates an input box which you can use to get the username for the first screen.* ***:param*** *screen: the surface you want to draw on* ***:param*** *text: the text you want to print* ***:param*** *size: (x, y) -> where x is the length of the box and y is the height* ***:param*** *place: (x, y) -> where x is the x-coordinate of the box and y is the y-coordinate, in relation to 'screen'* ***:param*** *color: (R, G, B) of the background color of the box* ***:param*** *border\_width: the width of the border, 0 for no border* ***:param*** *border\_color: (R, G, B) of the border color* ***:param*** *text\_color: (R, G, B) of the text color* ***:param*** *font: the name of the font for the text  
 """* def \_\_init\_\_(self, screen, text, size, place, color=(255, 255, 255), border\_width=0, border\_color=(0, 0, 0), text\_color=(0, 0, 0), font=None):  
 self.font\_size = size[1]  
 self.\_screen = screen  
 self.text = text  
 self.size = size  
 self.place = place  
 self.color = color  
 self.border\_width = border\_width  
 self.border\_color = border\_color  
 self.text\_color = text\_color  
 self.font = font  
 self.initiated\_font = None  
  
 def draw(self):  
 *"""  
 Call this inside the loop in order to draw the textbox to the screen  
 """* if self.color:  
 pygame.draw.rect(self.\_screen, self.color, (self.place, self.size))  
 if self.border\_width:  
 pygame.draw.rect(self.\_screen, self.border\_color, (tuple([x-self.border\_width/2 for x in self.place]), tuple([x+self.border\_width for x in self.size])), self.border\_width)  
 font\_size = self.font\_size  
 text\_font = pygame.font.Font(self.font, font\_size)  
 while max([text\_font.size(self.text.split("\n")[x])[0] for x in range(len(self.text.split("\n")))]) >= self.size[0] or (text\_font.size(self.text)[1])\*len(self.text.split("\n")) >= self.size[1]:  
 font\_size -= 1  
 text\_font = pygame.font.Font(self.font, font\_size)  
 self.font\_size = font\_size  
 self.initiated\_font = text\_font  
 linumber = 0  
 for line in self.text.split("\n"):  
 printext = text\_font.render(line, False, self.text\_color)  
 textW, textH = text\_font.size(self.text.split("\n")[linumber])  
 self.\_screen.blit(printext, (self.place[0] + self.size[0]/2 - textW/2, self.place[1] + self.size[1]/2 - (textH/2)\*len(self.text.split("\n")) + textH\*linumber))  
 linumber += 1

### InputBox

מטפל בקליטה מהקלדת. אפשר לכתוב אותיות מספרים, להתשמש בשיפט, קאפסלוק, בחצים, הום, אנד, אנטר, דליט ועוד. כמעט את כל הפונקציות האפשריות במקלדת תיכנתתי לתוך הסיפריה ככה שיהיה קל כל פעם לבקש קלט מהשתמש. לתכנת את הספרייה היה קצת סיפור שכן כל ההתייחסות של פייגים למקלדת היא כהופעות של אירועים, אבל בשביל להרכיב מכל האבנטים האלה קלט אחד, שיהיה אפשר לערוך מהתחלה אמצע וסוף, שיהיה סממן על מקום העריכה הייתי צריך לעבוד קשה. לבדוק כל מיני מקשים מיוחדים, לתזמן דיליי כמו שצריך...

class InputBox:  
 *"""  
 Creates an input box which you can use to get the username for the first screen.* ***:param*** *screen: the surface you want to draw on* ***:param*** *size: (x, y) -> where x is the length of the box and y is the height* ***:param*** *place: (x, y) -> where x is the x-coordinate of the box and y is the y-coordinate, in relation to 'screen'* ***:param*** *color: (R, G, B) of the background color of the box* ***:param*** *border\_width: the width of the border, 0 for no border* ***:param*** *border\_color: (R, G, B) of the border color* ***:param*** *text\_color: (R, G, B) of the text color* ***:param*** *font: the name of the font for the text  
 """* def \_\_init\_\_(self, screen, size, place, color=(255, 255, 255), border\_width=0, border\_color=(0, 0, 0), text\_color=(0, 0, 0), font="Arial", numeric=False, placeholder=""):  
 self.\_\_start = time.time()  
 self.inputted\_text = placeholder  
 self.\_\_keys = {letter: time.time() for letter in [chr(let) for let in range(97, 123) + range(48, 58) + [39, 44, 45, 46, 47, 59, 61, 91, 92, 93] + [8, 13, 32, 127]] + ["<-", "->"]}  
 self.\_\_toggle = False  
 self.\_\_toggle\_\_spot = -1  
 self.\_\_last\_size = 0  
 self.\_screen = screen  
 self.size = size  
 self.place = place  
 self.color = color  
 self.border\_width = border\_width  
 self.border\_color = border\_color  
 self.text\_color = text\_color  
 self.font = font  
 self.is\_numeric = numeric  
 self.display = OutputBox(screen, "", size, place, color, border\_width, border\_color, text\_color, font)  
 def draw(self):  
 *"""  
 Call this inside the loop in order to draw the textbox to the screen  
 """* if self.color:  
 pygame.draw.rect(self.\_screen, self.color, (self.place, self.size))  
 if self.border\_width:  
 pygame.draw.rect(self.\_screen, self.border\_color, (tuple([x-self.border\_width/2 for x in self.place]), tuple([x+self.border\_width for x in self.size])), self.border\_width)  
 last = self.\_\_toggle  
 if pygame.mouse.get\_pressed()[0]:  
 self.\_\_toggle = True if all(p <= x <= p+s for x, s, p in zip(pygame.mouse.get\_pos(), self.size, self.place)) else False  
 self.\_\_toggle\_\_spot = -1 if last != self.\_\_toggle else self.\_\_toggle\_\_spot  
 text\_to\_print = self.inputted\_text  
 backspaced = False  
 if self.\_\_toggle:  
 pressed = pygame.key.get\_pressed()  
 #if 1 in pressed:  
 # if pressed.index(1) != 300:  
 # print pressed.index(1)  
 shift = 97-65 if any(pressed[303:305]) else 0  
 temp = range(97, 123) + range(48, 58) + [8, 13, 32, 39, 45, 44, 46, 47, 61, 59, 91, 92, 93, 127, 275, 276, 278, 279] if not self.is\_numeric else range(48, 58) + [8, 127, 275, 276, 278, 279]  
 for key in temp:  
 if pressed[key]:  
 if key in [275, 276]:  
 key = "<-" if key == 276 else "->"  
 if time.time() - self.\_\_keys[key] >= 0.1:  
 self.\_\_keys[key] = time.time()  
 if key == "<-":  
 self.\_\_toggle\_\_spot = self.\_\_toggle\_\_spot - 1 if self.\_\_toggle\_\_spot\*-1 <= len(self.inputted\_text) else self.\_\_toggle\_\_spot  
 else:  
 self.\_\_toggle\_\_spot = self.\_\_toggle\_\_spot + 1 if self.\_\_toggle\_\_spot < -1 else self.\_\_toggle\_\_spot  
 backspaced = True  
 elif key in [278, 279]:  
 key = "<-" if key == 278 else "->"  
 self.\_\_toggle\_\_spot = -1 if key == "->" else len(self.inputted\_text)\*-1-1  
 backspaced = True  
 elif time.time() - self.\_\_keys[chr(key)] >= 0.4 and key not in [8, 127, 275, 276]:  
 self.\_\_keys[chr(key)] = time.time()  
 if 97 <= key <= 122:  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text += chr(key-shift)  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + chr(key-shift) + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 elif key == 13:  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text += "\n"  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + "\n" + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 elif 48 <= key <= 57 and shift and not self.is\_numeric:  
 add = ")!@#$%^&\*("[key-48]  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text += add  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + add + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 elif key in [39, 44, 45, 46, 47, 59, 61, 91, 92, 93] and shift:  
 c = "\_" if chr(key) == "-" else ""  
 c = "+" if chr(key) == "=" else c  
 c = "{" if chr(key) == "[" else c  
 c = "}" if chr(key) == "]" else c  
 c = ":" if chr(key) == ";" else c  
 c = "\"" if chr(key) == "'" else c  
 c = "|" if chr(key) == "\\" else c  
 c = "<" if chr(key) == "," else c  
 c = ">" if chr(key) == "." else c  
 c = "?" if chr(key) == "/" else c  
 if c:  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text += c  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + c + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 else:  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text += chr(key)  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + chr(key) + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 elif key == 8:  
 if time.time() - self.\_\_keys[chr(key)] >= 0.1:  
 self.\_\_keys[chr(key)] = time.time()  
 if self.\_\_toggle\_\_spot == -1:  
 self.inputted\_text = self.inputted\_text[:-1]  
 else:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot] + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 self.display.font\_size += 100  
 backspaced = True  
 elif key == 127:  
 if time.time() - self.\_\_keys[chr(key)] >= 0.1:  
 self.\_\_keys[chr(key)] = time.time()  
 if self.\_\_toggle\_\_spot == -2:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1]  
 self.\_\_toggle\_\_spot += 1  
 elif self.\_\_toggle\_\_spot != -1:  
 self.inputted\_text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + self.inputted\_text[self.\_\_toggle\_\_spot+2:]  
 self.\_\_toggle\_\_spot += 1  
 self.display.font\_size += 100  
 backspaced = True  
 else:  
 if key in [278, 279]:  
 pass  
 elif key in [275, 276]:  
 key = "<-" if key == 276 else "->"  
 self.\_\_keys[key] = 0  
 else:  
 self.\_\_keys[chr(key)] = 0  
 self.display.text = self.inputted\_text  
 self.display.draw()  
 font\_size = self.display.font\_size  
 text\_font = self.display.initiated\_font  
 linumber = 0  
 if self.\_\_toggle\_\_spot == -1:  
 text = self.inputted\_text + "\a"  
 else:  
 text = self.inputted\_text[:self.\_\_toggle\_\_spot+1] + "\a" + self.inputted\_text[self.\_\_toggle\_\_spot+1:]  
 for line in text.split("\n"):  
 if "\a" in line:  
 if self.\_\_toggle and int((time.time() - self.\_\_start)\*2) % 2 == 0 or backspaced:  
 textW, textH = text\_font.size(line.replace("\a", ""))  
 toggle = line.index('\a') - len(line.replace("\a", "")) - 1  
 W, H = text\_font.size(line.replace("\a", "")[toggle+1:])  
 W = 0 if toggle == -1 else W  
 pygame.draw.line(self.\_screen, self.text\_color, (self.place[0] + self.size[0]/2 + textW/2 - W, self.place[1] + self.size[1]/2 - (textH/2)\*len(text.split("\n")) + textH\*linumber),  
 (self.place[0] + self.size[0]/2 + textW/2 - W, self.place[1] + self.size[1]/2 - (textH/2)\*len(text.split("\n")) + textH\*linumber + textH), font\_size/10)  
 linumber += 1  
 def get\_input(self):  
 *"""  
 Call this after the loop is done and the user has submitted his input in order to get it* ***:return*** *The user's input  
 """* return self.inputted\_text  
 def is\_toggled(self):  
 *"""* ***:return*** *True if the text box is currently toggled and False otherwise  
 """* if self.\_\_toggle:  
 return True  
 return False

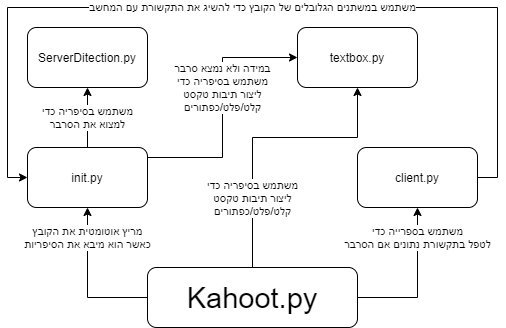
### ButtonBox

משש לייצירת כפתורים על המסך. לכפתור מופיע אווטליין כאשר העכבר מעליו ויש בו פונקציה שמודיע אם מאז הפעם האחרונה שבדקת הייתה לחיצה על הכפתור.

class ButtonBox:  
 *"""  
 Creates an input box which you can use to get the username for the first screen.* ***:param*** *screen: the surface you want to draw on* ***:param*** *text: the text you want to print* ***:param*** *size: (x, y) -> where x is the length of the box and y is the height* ***:param*** *place: (x, y) -> where x is the x-coordinate of the box and y is the y-coordinate, in relation to 'screen'* ***:param*** *color: (R, G, B) of the background color of the box* ***:param*** *border\_width: the width of the border, 0 for no border* ***:param*** *border\_color: (R, G, B) of the border color* ***:param*** *text\_color: (R, G, B) of the text color* ***:param*** *font: the name of the font for the text  
 """* def \_\_init\_\_(self, screen, text, size, place, color=(255, 255, 255), border\_width=3, border\_color=(0, 0, 0), text\_color=(0, 0, 0), font=None, mouse=True):  
 self.\_\_font\_size = size[1]  
 self.\_screen = screen  
 self.text = text  
 self.size = size  
 self.place = place  
 self.color = color  
 self.border\_width = border\_width  
 self.border\_color = border\_color  
 self.text\_color = text\_color  
 self.font = font  
 self.clicked = False  
 self.highlighted = False  
 self.mouse = mouse  
  
 def draw(self):  
 *"""  
 Call this inside the loop in order to draw the textbox to the screen  
 """* if self.color:  
 pygame.draw.rect(self.\_screen, self.color, (self.place, self.size))  
 x, y = pygame.mouse.get\_pos()  
 if self.place[0] < x < self.place[0] + self.size[0] and self.place[1] < y < self.place[1] + self.size[1]:  
 if self.border\_color:  
 pygame.draw.rect(self.\_screen, self.border\_color, (tuple([x-self.border\_width/2 for x in self.place]), tuple([x+self.border\_width for x in self.size])), self.border\_width)  
 if self.mouse:  
 pygame.mouse.set\_cursor(\*pygame.cursors.broken\_x)  
 self.highlighted = True  
 if pygame.mouse.get\_pressed()[0]:  
 self.clicked = True  
 else:  
 if self.highlighted:  
 pygame.mouse.set\_cursor(\*pygame.cursors.arrow)  
 self.highlighted = False  
 font\_size = self.\_\_font\_size  
 text\_font = pygame.font.Font(self.font, font\_size)  
 while max([text\_font.size(self.text.split("\n")[x])[0] for x in range(len(self.text.split("\n")))]) >= self.size[0] or (text\_font.size(self.text)[1])\*len(self.text.split("\n")) >= self.size[1]:  
 font\_size -= 1  
 text\_font = pygame.font.Font(self.font, font\_size)  
 self.\_\_font\_size = font\_size  
 linumber = 0  
 for line in self.text.split("\n"):  
 printext = text\_font.render(line, False, self.text\_color)  
 textW, textH = text\_font.size(self.text.split("\n")[linumber])  
 self.\_screen.blit(printext, (self.place[0] + self.size[0]/2 - textW/2, self.place[1] + self.size[1]/2 - (textH/2)\*len(self.text.split("\n")) + textH\*linumber))  
 linumber += 1  
 def was\_clicked(self):  
 temp = self.clicked  
 self.clicked = False  
 return temp  
 def is\_highlighted(self):  
 return self.highlighted

## במחשב הקליינט

### קבצים



### מציאת הסרבר

קודם כל צריך המחשב לדעת היכן נמצא הסרבר כדי שיוכל להתחבר אליו. לשם כך הוא מחכה עד שהוא יקבל קריאת "פה הסרבר". ברגע שהוא מקבל קריאה הוא מתחבר למחשב ששלח לו הודעה זאת. אם לא מתקבלת קריאה מופיע למשתמש כלב עצוב בגשם. המחשב מריץ פקודה blocking שמאזינה לסרבר ב-Thread נפרד. אם עובר 0.7 שניות ולא התקבלה קריאה המחשב עוצר את ה-Thread ומודיע שאין משחק. אם כן התקבלה קריאה הוא שומר אותה ב-data\_to\_return ומחזיר אותה.

def server\_scout():  
 try:  
 t = threading.Thread(None, check)  
 t.daemon = True  
 t.start()  
 start = time.time()  
 while time.time() - start <= 0.7:  
 if data\_to\_return:  
 return data\_to\_return.split("Here Be Server: ")[1]  
 if goodbye:  
 exit()  
 if data\_to\_return:  
 return data\_to\_return.split("Here Be Server: ")[1]  
 else:  
 return None

def check():  
 try:  
 global data\_to\_return, goodbye  
 client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM) # UDP  
 client.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1)  
 client.bind(("", 37020))  
 data, addr = client.recvfrom(1024)  
 if data:  
 if "Here Be Server: " in data and len(data.split("Here Be Server: ")) == 2:  
 data\_to\_return = data

### התחברות

IP = ServerDitection.server\_scout()  
 if IP:  
 my\_socket = socket.socket()  
 my\_socket.connect((IP, 23))  
 else:

כלב עצוב (ראה בממשק משתמש)

### ספריית הקליינט

ספרייה זו אותה קיבלו חברי לכיתה מטפלת בכל תקשורת הנתונים עם השרת ובה פונקציות המאפשרות למשתמש לעושת login, לשלוח תשובה, לבדוק כמה זמן נשאר לשאלה, כמה ניקוד יש למשתמש ועוד. בכל המידע המתקבל מהסרבר מטפלת הספרייה בעצמה ושומרת כמידע נגיש למשתמש.

#-----------------------Imports-----------------------  
import os  
import sys  
sys.path.insert(0, os.getcwd()+'/files')  
import ServerDitection  
import traceback  
import socket  
from select import select  
import time  
import Libraries  
sys.dont\_write\_bytecode = True  
#-----------------------Globals-----------------------  
IP = Libraries.IP  
my\_socket = Libraries.my\_socket  
username = None  
game\_finished = False  
new\_question = False  
timer = None  
answered\_current = False  
  
answer = None  
score = None  
behind = None  
place = None  
answers = None  
#-----------------------Library-----------------------  
def login(name):  
 *"""  
 Login to the our Kaboot server.* ***:param*** *name: your choosen username. len(name) <= 14* ***:return****: True -> Connected successfully  
 False - > Username was taken  
 """* if '\r' in name or '\n' in name or '\t' in name:  
 raise Exception("Usernames are not allowed to contain line breaks or tabs")  
 if len(name) > 14:  
 raise Exception("Name should be 14, characters or less!\nWe didn't give enough fuck to make our GUI more responsive.")  
 try:  
 global my\_socket, username  
 if not my\_socket:  
 IP = "127.0.0.1"  
 my\_socket = socket.socket()  
 my\_socket.connect((IP, 23))  
  
 my\_socket.send("login: " + name + "\n")  
 data = ''  
 while '\n' not in data:  
 data += my\_socket.recv(1)  
 if data == "OK\n":  
 username = name  
 return True  
 else:  
 return False  
 except Exception:  
 raise Exception("Login failed!\nPlease make sure the \"test\_server\" is up")  
def send\_answer(your\_answer=None):  
 *"""  
 Sends your answer to the current question to the server.* ***:return*** *None  
 """* if your\_answer:  
 if your\_answer < 1 or your\_answer > 4:  
 raise Exception("Answer can only be a number between 1 and 4!\nYou've chosen " + str(your\_answer) + "!")  
 global my\_socket, answered\_current  
 handle\_server()  
 if not question\_received():  
 raise Exception("You are out of time")  
 if answered\_current:  
 raise Exception("You have already answered the current question!")  
 my\_socket.send("answer: " + str(your\_answer) + "\n")  
 answered\_current = True  
def question\_received():  
 *"""* ***:return*** *True -> A question has started  
 False -> New question didn't start yet  
 """* global new\_question  
 handle\_server()  
 return new\_question  
def time\_left():  
 *"""* ***:return*** *How much seconds are left for the current question  
 """* global timer  
 handle\_server()  
 if not timer:  
 raise Exception("No question currently in progress")  
 return timer - int(time.time())  
def time\_is\_up():  
 *"""  
 checks if the question has finished* ***:return*** *True -> if it did  
 False -> if it didn't  
 """* global timer  
 handle\_server()  
 if not timer:  
 return True  
 return timer - int(time.time()) < 0  
def get\_answers():  
 *"""  
 checks if the question has finished* ***:return*** *True -> if it did  
 False -> if it didn't  
 """* global answers  
 handle\_server()  
 if not timer:  
 raise Exception("No question currently in progress")  
 return answers  
def result():  
 *"""  
 checks if your answer was correct or wrong* ***:return*** *True -> it was correct  
 False -> it was wrong/no answer was received  
 """* global answer  
 handle\_server()  
 if answer is None:  
 raise Exception("Question is still in progress!")  
 return answer  
def get\_score():  
 *"""  
 get your current score* ***:return*** *the amount of points you have collected so far  
 """* global my\_socket, score  
 my\_socket.send('get\_score\n')  
 time.sleep(0.3)  
 attemps = 1000000  
 while not handle\_server()['score']:  
 attemps -= 1  
 if attemps < 0:  
 raise Exception("Server didn't respond.")  
 return score  
def get\_behind():  
 *"""  
 antagonize yourself by knowing who is this cunt right in front of you* ***:return*** *HIM  
 or None if you are in first place  
 """* global my\_socket, score  
 if get\_place() > 1:  
 my\_socket.send('get\_behind\n')  
 time.sleep(0.3)  
 attemps = 1000000  
 while not handle\_server()['behind']:  
 attemps -= 1  
 if attemps < 0:  
 raise Exception("Server didn't respond.")  
 return behind  
 else:  
 return None  
def get\_place():  
 *"""  
 tells you how well you did in compare to other student in your class* ***:returns*** *your place  
 """* global my\_socket, score  
 my\_socket.send('get\_place\n')  
 time.sleep(0.3)  
 attemps = 1000000  
 while not handle\_server()['place']:  
 attemps -= 1  
 if attemps < 0:  
 raise Exception("Server didn't respond.")  
 return place  
def end\_game():  
 *"""  
 Prepare your final\_screen because this game is done, maybe...  
 you can still preform 'get\_place', 'get\_score', or 'get\_behind' functions after game is done.* ***:return*** *True -> game is finished  
 False -> not yet  
 """* global game\_finished  
 handle\_server()  
 return game\_finished  
#----------------------Functions----------------------  
#--------------------Back End Shit--------------------  
#--------------------Do Not Touch!--------------------  
def handle\_server():  
 *"""  
 updates the server connection* ***:return****: a dictionary containing everything that was received. {result, score, behind, place}  
 """* global my\_socket, game\_finished, new\_question, timer, behind, score, place, answer, answered\_current, answers  
 rlist, wlist, xlist = select([my\_socket], [my\_socket], [my\_socket], 0.1)  
 recieved = {'result': False, 'score': False, 'behind': False, 'place': False}  
 if rlist:  
 data = ''  
 while '\n' not in data:  
 data += rlist[0].recv(1)  
 data = data.replace('\n', '')  
 if data == "True":  
 answer = True  
 recieved['result'] = True  
 new\_question = False  
 timer = None  
 answered\_current = False  
 elif data == "False":  
 answer = False  
 recieved['result'] = True  
 new\_question = False  
 timer = None  
 answered\_current = False  
  
 elif data[:len("new: ")] == "new: ":  
 timer = int(time.time()) + int(data.split(": ")[1].split("[")[0]) - 1  
 new\_question = True  
 answers = data.split("['")[1].split("']")[0].replace("\"", "\'").replace("\\n", "\n").split("', '")  
 answer = None  
  
 elif data[:len("score: ")] == "score: ":  
 score = int(data.split(": ")[1])  
 recieved['score'] = True  
  
 elif data[:len("place: ")] == "place: ":  
 place = int(data.split(": ")[1])  
 recieved['place'] = True  
  
 elif data[:len("behind: ")] == "behind: ":  
 behind = data.split(": ")[1]  
 recieved['behind'] = True  
  
 elif data == "game\_finished":  
 game\_finished = True  
 return recieved

### עיצוב

כל עיצוב ה-GUI של הקליינט נעשה בקובץ Kahoot.py. בהתחלה הקובץ הכיל טמפליית של התלמידים היו צריכים למלא בעצמם. מכיוון שכבר ראינו בסרבר כיצד נראה עיצוב טיפיקלי של מסך בפייגיים עם שימוש בטקסטבוקס ועוד נסתכל רק על הטמפליית.

#-----------------------Imports-----------------------  
from Libraries import textbox  
from Libraries import client  
#-----------------------Globals-----------------------  
username = None  
screen = None  
  
#-------------------------Main-------------------------  
def main():  
 *"""  
 Don't worry! We have got your back!  
 All of the hard work was already done for you.  
 Our server and your client already have a programmed protocol which you can access by our very helpful "client" library.  
 All that is left for you to do is to program your own unique User Interface, on which the client would run.  
 How fun is that?!  
  
 Everything is already sorted out in to functions. Each function is its own screen and should be programmed separately and independently,  
 But do not open a new window for every function, that's just bad manners...  
  
 To your assist, we have programmed two libraries:  
 - client  
 - textbox  
 To see a library's abilities enter its file and read its fucking manual, below the #---Library--- stamp  
  
 That is all,  
 GOOD LUCK, Amigo!  
 """* while not client.login(login\_screen()):  
 username\_taken()  
 loading("Waiting for the game to start...", lambda x: client.question\_received())  
 while not client.end\_game():  
 if client.question\_received():  
 client.send\_answer(main\_screen())  
 if client.question\_received():  
 loading("Waiting for question to end...", lambda x: not client.question\_received())  
 else:  
 results\_screen(client.result(), lambda x: client.question\_received() or client.end\_game())  
 finish\_screen()  
#----------------------Functions----------------------  
def login\_screen():  
 *"""  
 Welcome, amigo!  
 There is still a lot of work ahead of us, but I promise, it will be fun!  
  
 TASK I:  
 Program a login screen which gets a username as an input from the user and returns it.  
 You can use the textbox library which we programmed just for you in order to create an "InputBox"* ***:return*** *a username  
 """* global username, screen  
 username = raw\_input("Choose username: ") # <--- delete this  
 return username  
def username\_taken():  
 *"""  
 Ooooooooooooooooooooooooops!  
 Looks like you and another amigo were planing on using the same username and you weren't fast enough... What a shameful lost!  
  
 TASK II:  
 Notify the user the username he had chosen is already used and he should chose a different one, Thou don't input it at this screen* ***:return*** *None  
 """* print "Username is taken!" # <--- delete this  
def main\_screen():  
 *"""  
 That was fun! wasn't it?  
 Now, let's really get to business!  
  
 This is the screen where the user can choose an answer while a question is up.  
 BE CAREFUL!  
 If the user doesn't have enough time to answer the question the function must return None.  
 The server won't accept answers which were submitted late, and the client would crush!  
  
 TASK III:  
 Program the main screen.  
 Screen must contain a way to input your answer - a number between 1, and 4  
 Do not wait for the "time\_is\_up" if an answer is chosen before hands, return it at the moment  
 The "client" library has a bunch of function you can use to decorate the screen* ***:return*** *The answer the user had choosen, 1 <= number <= 4  
 OR None if he didn't have enough time  
 """* answer = int(raw\_input("Your answer: ")) # <--- delete this  
 while not client.time\_is\_up():  
 return answer # <---------------- write here  
 return None  
def results\_screen(correct, wait\_until):  
 *"""  
 Are you a secrete genius?  
 Or the class's biggest foul?  
 Well, amigo, today we find out!* ***:param*** *correct: True for mastermind, False otherwise* ***:param*** *wait\_until: exit condition. don't touch!  
  
 TASK IV:  
 Program a "Horrrrray! Very Correct Answer Indeed." and a "Boooooo! You Are A Loser!" screens  
 The "client" library has a bunch of function you can use to decorate the screen* ***:return*** *None  
 """* if correct:  
 print "Your answer was correct" # <--- delete this  
 while not wait\_until(''):  
 pass # <-------------------------- wright here  
 else:  
 print "Your answer was wrong" # <--- delete this  
 while not wait\_until(''):  
 pass # <------------------------ wright here  
def loading(message, wait\_until):  
 *"""  
 HOLD UP! Why are you in a rush! Chill, amigo. Somethings in life take time, you know.  
 Like what, you ask?!  
 For example, When you login to our Kaboot but we are still waiting for other players to join in.  
 Or when you finish a question but there is still time left to answer it.* ***:param*** *message: the reason for the delay  
 will be one of the following:  
 - "Waiting for the game to start..."  
 - "Waiting for question to end..."* ***:param*** *wait\_until: exit condition. don't touch!  
  
 TASK V:  
 Program a loading screen so everyone would know they need to calm the fuck down once in a while.* ***:return*** *nothing  
 """* print message # <--------------- delete this  
 while not wait\_until(''):  
 pass # <-------------------- wright here  
def finish\_screen():  
 *"""  
 Well, it's been a pleasure working with you,  
 but everything eventually must come to an end.  
 Before our farewells, there is still one last thing.  
  
 TASK VI:  
 Make an ending screen indicating that the game is over.  
 Make sure you let your client know what place he ended up getting.  
 Do not close the program until the user says so!* ***:return*** *None  
 """* print "\nThe game is finished!\nYou won " + str(client.get\_place()) + " place with " + str(client.get\_score()) + " points!\nGood bye!\n" # <---- delete this  
 raw\_input("\nPress 'Enter' to exit...") # In my example, a simple "raw\_input()" is used to wait until the user wants to exit the program  
def gui\_is\_ready():  
 *"""  
 That's it, amigo!  
 Farewell!  
  
 Task VII:  
 Contact one of the project managers and tell them you have finished you GUI.  
 We want to appreciate how well made it is, and also to instruct you how to connect to the real Kaboot.  
 """* pass  
if \_\_name\_\_ == '\_\_main\_\_':  
 gui\_is\_ready()  
 main()

## test\_server.pyw

את השרת המדומה העברתי לתמידים כדי שיוכלו להריץ משחקים מדומים על המחשב שליהם כדי לבדוק שהקליינט שלהם עובד כמו שצריך ולא קורס. בקובץ הראשוני ששלחתי להם הקליינט לא חיפש סרבר ב-ServerDitection אלה חיפש רק משחקים על אותו מחשב, 127.0.0.1, רק אחרי שכולם סיימו את הקליינטים שלהם ושיחקנו את המשחק האמיתי הנחתי אותם איך לשנות שורה אחת בקובץ init.py של הסיפריות כדי שיחפש משחקים ברשת המקומית. עד אז כולם עבדו עם השרת המדומה. השרת המדומה מאפשר מלעבור באופן שרירותי בן כל המסכים של הקליינט כדי לבדוק שהם עובדים. לאחר מכן השתמשתי בשרת המדומה בעצמי המון כדי לתכנת את הקליינט שלי.  
השרת המדומה מתוכנת בסיפריה Tkinter.

### קבצים

C:\Users\Erez Atir\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\C27EA383.tmp

### עיצוב

כל העיצוב של השרת המדומה נעשה בקובץ test\_server.py ועוצב על ידי שימוש בתוכנה הסופר יעלה – Page.  
התוכנה יוצרת את הדף לבד וממקמת הכל במקומו. אני תיכנתתי את הקובץ test\_server\_support.py כדי לטפל בתקשורת נתונים ובאינטרקציה עם המשתמש. את הפונקציה בקובץ שמשאירה את החיבור בן השרת ללקוח פעיל אני מריץ ב-Thread נפרד בלולאה תמידית.

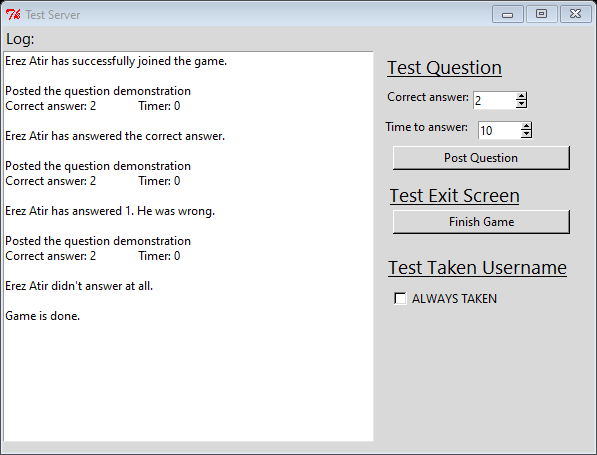
import test\_server\_support  
def vp\_start\_gui():  
 *"""Starting point when module is the main routine."""* global val, w, root  
 root = tk.Tk()  
 root.resizable(False, False)  
 root.after(400, test\_server\_support.login)  
 thread.start\_new\_thread(test\_server\_support.update, ())  
 test\_server\_support.set\_Tk\_var()  
 top = Toplevel1 (root)  
 Server = test\_server\_support.init(root, top)  
 root.mainloop()  
 Server.end\_game()  
 Server.new\_question(0)  
 timer = time.time()  
 while time.time() - timer <= 1:  
 Server.receive()

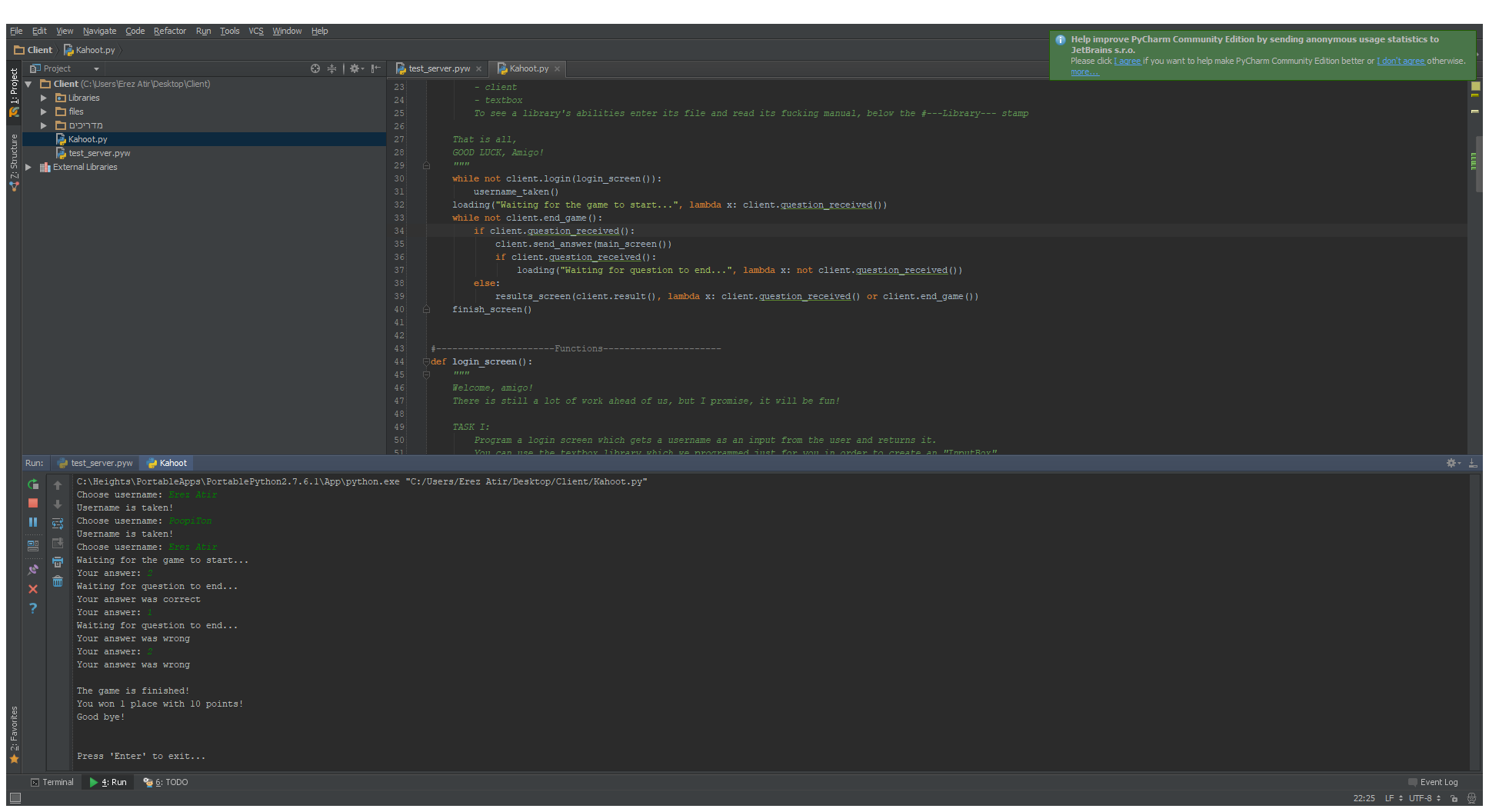
### פונקציונאליות

את כל הפונקציונאליות של המקשים שונים, הפועול שקורות כאשר לוחצים על המקש, תיכנתתי בקובץ test\_server\_support.py בשימוש בספרייה פייגיים. לדוגמא הקוד ששולח שאלה חדשה לקליינט.

def new\_question():  
 global root, player, correct, timer, spinbox1, spinbox2  
 if not timer:  
 if player:  
 try:  
 correct = int(spinbox1.get())  
 timer = int(time.time()) + int(spinbox2.get())  
 Server.new\_question(spinbox2.get())  
 w.log.config(state="normal")  
 w.log.insert("end", "Posted the question demonstration\nCorrect answer: %s Timer: %s\n\n" % (correct, timer - int(time.time())))  
 w.log.config(state="disabled")  
 w.Button1.config(state="disable")  
 root.after(150, receive)  
 sys.stdout.flush()  
 except Exception:  
 tkMessageBox.showerror("Error!", "You must choose the correct answer and the duration of the question!")  
 else:  
 tkMessageBox.showerror("Error!", "You haven't successfully joined in yet!")  
 else:  
 tkMessageBox.showerror("Error!", "There is currently another question in progress!")

### ממשק משתמש

כאשר אתה מתכנת קליינט משל עצמך בטמפלייט ששלחתי, לפני הרצתו הרץ קודם את השרת המדומה. לאחר שחלון השרת המדומה נפתח הרץ את הקליינט שתכנתת. השתמש בפונקציות השונות של השרת המדומה כדי לנווט בן החלונות השונים של הקליינט שלך ולדמות משחק קבוט אמיתי. בדוק שכל חלונות הקליינט עובדים כראוי.

ב-Log ניתן לראות משחק מדומה שנערך עם השחקן Erez Atir. שלוש פעמים השתמשתי בפונקציה Test Question כדי לשלוח שאלה שהתשובה הנכונה אליה היא 2 ויש 10 שניות לענות עליה. כאשר עשיתי זאת הופיע הכיתוב בלוג שלי והזמן בטיימר החל לרדת מ10 עד ל-0. בפעם הראשונה עניתי מהקליינט 2, בפעם השנייה 1 ובפעם השלישית עניתי 2 אבל רק אחרי שניגמר הזמן בטיימר. לאחר מכן לחצתי על הלחצן Finish Game והועברתי בקליינט מסך של סיום המשחק. עם אני אלחץ עם העכבר ואסמן וי בקופסא ליד הכיתוב ALWAYS TAKEN אני אגרום לכך שלא משנה עם איזה שם משתמש אני אתחבר אקבל שגיאה ששם המשתמש הזה תפוס, וככה אוכל לבדוק את החלון בקליינט שמודיע על שגיאה זו.

משמאל ניתן לראות את המשחק מנקודת המבט של הקליינט הדיפולטי   
שמגיע בטמפלייט של הקליינט ורץ דרך הטרמינל של פייצ'ארם.

ניתו לראות בשני ניסיונות ההתחברות הראשונים היה מסומן וי בקופסא  
ALWAYS TAKEN. לאחר מכן הורדתי אותו והתחברתי עם השם Erez Atir.

עניתי 2 בשאלה הראשונה, 1 בשנייה ו-2 בשלישית אבל רק אחרי שנגמר הזמן.

לאחר מכן לחצתי על הכפתור Finish Game וקיבלתי הודעה שמודיע על כך  
ביחד עם הנתונים שלי מהמשחק. בשרת המדומה תמיד מסיימים מקום ראשון  
כי תמיד יש רק משתתף אחד, וכל שאלה שווה באופן אוטומטי 10 נקודות.

לכן קיבלתי הודע שסיימתי את המשחק במקום ראשון עם 10 נקודות,  
מהשאלה הראשונה.

# בסיסי נתונים

בפרויקט שלי אין צורך בשימוש נרחב של בסיסי נתונים. הדברים היחידים שאותם צריך לשמור זה את החידונים עצמם בצורה שתהיה קלה לערוך ויעלה. לשם כך השתמשתי בספריית פייטון לעבוד עם קיבצי JSON.

החידונים שמורים כולם כקבצי JSON בהם מופיע כל המידע הדרוש כדי להריץ את החידון.  
החידונים שמורים בתיקיה quizes.

הנה חידון לדוגמא:

{  
 "Questions": [  
 {  
 "time to answer": 40,   
 "photo": "/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAUDBAQEAwUEBAQFBQU..........m0UVQj/2Q==",   
 "question": "Is this the best game you ever played in your life?",   
 "answers": [  
 "Yes!",   
 "Yeah!",   
 "Absolutely!",   
 "just a ripoff!"  
 ],   
 "time to read": 10,   
 "points": 12,   
 "correct answer": 1,   
 "image file type": "jpg"  
 },   
 {  
 "time to answer": 140,  
 "photo": null,   
 "question": "Is the sky blue?",  
 "answers": [  
 "Yes!",   
 "I don't know…",  
 "I think so.",  
 "Well,\nyou tell me!"  
 ],   
 "time to read": 10,   
 "points": 22,  
 "correct answer": 2,   
 "image file type": null  
 }  
 ]  
}

# רפלקציה אישית

אני אכתוב רפלקציה אישית לגירסא הסופית של הספר פרוייקט. כרגע אין לי איך לסכם את העבודה שלי מכיוון שלא סיימתי אותה עדיין אלה אני רק ב-95% אחוז.  
מה שנשאר לי לעשות זה לכתוב את התוכנה שמחבר בן Game.py ל-Quizmatron.py.  
כשאסיים את זה אוסיף עליה קצת במדריך למפתח, אכתוב רפלקציה אישית על הליך העבודה ואז אעשה בבליוגרפיה, בא אנסה לחפש את סירטונים שמהם למדתי Tkinter ו-JSON, למרות שאיני בטוח שאזכור באיזה סרטונים השתמשתי.  
לנספחים אוסיף את המצגת שהכנתי כדי ללמד את הכיתה פייגיים, את הספר לימוד של גבהים בפייגיים וקישוור ל-Page.

אתגרים בפיתוח  
אני חושב שהאתגר הכי משמעותי בפיתוח היה כתיבת ספר הפרויקט.  
נאמר לנו שוב ושוב במהלך השנה שלא לחכות עם כתיבת ספר הפרויקט רק לסיום העבודה כי בסוף אתם תעבדו עד הרגע האחרון ולא יהיה לכם זמן לכתוב ספר פרויקט ושהדרך הכי נכונה לעשות את זה היא להתחיל את ספר הפרויקט עם תחילת העבודה ולעדכן אותו תוך כדי במהלך העבודה. אז כמובן שמה שעשיתי זה חיכיתי עם כתיבת ספר הפרויקט לרגע האחרון. אמרתי לעצמי שגם ככה אני לא יכול לכתוב את ספר הפרויקט תוך כדי פיתוח כי הרבה מהפיתוח נעשה בכלל בשנה שעברה. לפי הלו"ז שקיבלנו היה לנו זמן לכתוב את תיק הפרויקט עד לראשון ביוני אבל בגלל כל בלגן משרד החינוך דרש לקבל את תיקי הפרויקט קודם לכן. קיבלנו הודעה שאנחנו צריכים להגיש את ספר הפרויקט ב-19/04 רק ב-15/04. לא היה לי כלום עדיין. אני כותב את כל ספר הפרויקט מאפס בארבע ימים הקרובים.

זה לא הצורה היחידה בה נפגעתי מהקורונה. קצב העבודה שלי על הפרויקט היה מעולה לפני הקורונה. הייתי יושב כמעט כל יום בערב על הפרויקט כמה שעות לפני שהלכתי לישון. בצורה אירונית ביטול בית ספר והאיסור על יציאה מהבית ממש פגעו בהתקדמות בפרויקט. מצד אחד, כן אני כל הזמן בבית והייתי יכול לעבוד על הפרויקט אבל מצד שני הדבר מאוד פגע בשגרה שלי. אני ועוד הרבה ילדים נכנסו קצת לדיכאון מהבידוד הזה. פתאום אין קשר עם המורים, אני כבר לא רואה את החברים שלי, את החברה שלי, את הילדים מהכיתה, לא יוצא להתאמן, קם כל יום בשש בערב ונשאר בפיג'מה כל יום (תכנית כל הלילה). זה היה לי מאוד קשה לחיות ככה ולא מצאתי מוטיבציה לעבוד על הפרויקט. הייתי הרבה יותר פרודקטיבי אם היה לי שיגרה או לפחות יכולת לצאת מהבית לנשום אוויר.

היה הרבה קושי בלקחת פרויקט שהתחלתי בכלל בשנה שעברה ולהתחיל לעבוד עליו שוב. שבועות ראשונים של העבודה השנה היו רק על להבין מה עשיתי שנה שעברה. הקוד לא מחולק טוב לקבצים ומחלקות כמו שהייתי עושה אם הייתי בונה אותו השנה מאפס ולקח לי הרבה זמן להבין איפה אני עומד ומה נשאר לי לעשות.  
לדוגמא, הקבוט רץ רק על רזולוציה מוסימת ורציתי שכל אחד יוכל לשים אותו על מסך מלא בלי שהכל יזוז מהמקום. אז הייתי צריך לעבור על אלפי שורות קוד משנה שעברה ולשנות בכל מקום שאני שם משהו על המסך ככה שהוא לא ישים את זה בפיקסל ספציפי אלה באחוז מסויים מהמסך. זו הייתה עבודה מאוד סיזיפית וארוכה.  
עוד דוגמא למשהו שנשאר לי לעשות השנה זה שהחידון ששיחקנו היה מתוכנת בתוך הקוד. רציתי לעשות שכל שאלון יהיה בקובץ נפרד. רציתי שתיהיה תוכנה שיוצרת שאולונים ועורכת שאלונים קיימים. רציתי שיוכלו לשתף שאלונים.  
וכמובן שלא היה קליינט, שכן הקטע היה שהתלמידים בכיתה שלי תיכנתו קליינטים בעצמם. אז עכשיו השתמשתי בטמפלייט משנה שעברה כדי גם כן לתכנת קליינט לקבוט