**<Screen Sharing>**

מסמך עיצוב

<Omri Niri>

<גרסה 1.0>

<27.05.19>

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 27.05.19 | 1.0 | גרסה ראשונית |
|  |  |  |
|  |  |  |
|  |  |  |

**1. הקדמה**

1.1 מטרה

*המסמך מיועד למי שמשתמש במוצר ומנסה להבין כיצד הוא עובד.*

1.2 המוצר

* Screen Sharing
* *המוצר כולל שתי תוכנות מרכזיות הרצות במחשבים שונים. מחשב אחד שהוא השרת מצלם תמונות מסך ושולח למחשב השני שהוא לקוח. המחשב השני קולט את כל פעולות המחשב שמבוצעות על ידי המשתמש (עכבר- תזוזה, לחיצות ומקלדת) ושולח אותן בהתאם לפרוטוקול למחשב השני (השרת).*

1.3 קישור למסמכים קודמים

*מסמך אפיון*

1.4 הגדרות

IP- *כתובת של מחשב. מיוצג על ידי 4 מספרים ונקודות ביניהם (במקרה של* IPv4*)*

Socket- *קצה* *"צינור" התקשורת שדרכו ניתן לתקשר לצד השני: לשלוח ולקבל הודעות.*

Thread- *תהליכון. יחידת הביצוע העצמאית של מערכת ההפעלה.*

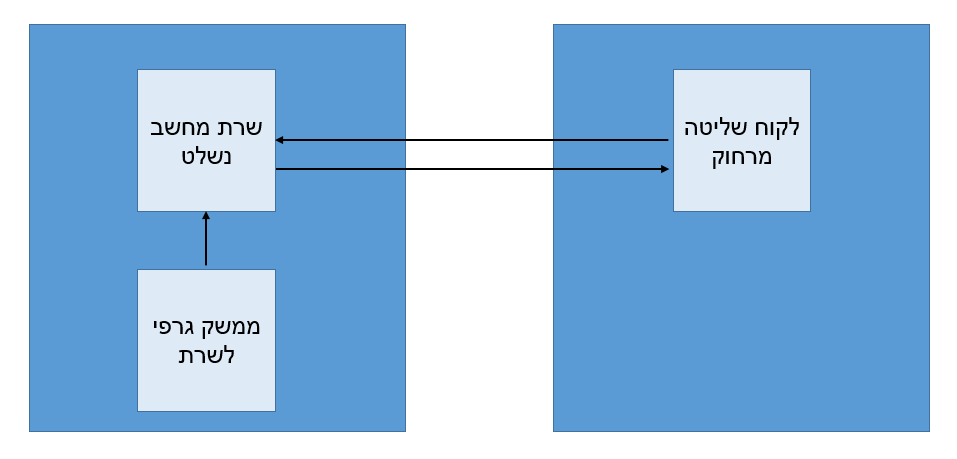
GUI*- ממשק גרפי.*

RGB*- פורמט ייצוג צבעים על המסך על ידי שלושה ערכים (אדום, ירוק וכחול)*

JPEG*- פורמט לייצוג תמונות שמאפשר איבוד מידע.*

1. ארכיטקטורת המערכת

2.1 מבט על



**הפרויקט מחולק לשתי תוכנות עיקריות שרצות כל אחת על מחשב אחר- שרת ולקוח. בהתחלה, תוכנת הלקוח מבקשת מהמשתמש להכניס את כתובת הIP של השרת. אם הכתובת אינה תקנית (למשל ip='john') אז התוכנה מציגה הודעה מתאימה. אם הכתובת תקנית אך אין תשובה התוכנה תציג הודעה מתאימה. אם הכתובת תקנית ויש התחברות אז הGUI נסגר ונוצר חלון חדש שבו יהיה השידור של המסך של מחשב השרת. השרת מחכה ללקוח שהתחבר. בנוסף, השרת מריץ את תוכנת הGUI-CLIENT שתפקידה הוא להציג ממשק משתמש גרפי בשרת. השרת לא יכול בעצמו להריץ ממשק משתמש, משום שבTKINTER יש MAINLOOP ולא ניתן להריץ THREADS במקביל לMAINLOOP. לכן, השרת מריץ תוכנת GUI שמתקשרת איתו. התוכנה תשלח לשרת הודעה מתאימה ברגע שהמשתמש שינה את ההגדות: ביכולתו של המשתמש לקבוע אם בעל המחשב המרוחק יוכל לשלוט במחשבו או לא. ברגע שהלקוח מהמחשב המרוחק מתחבר השרת מתחיל לשלוח תמונות ללקוח בTHREAD נפרד. הלקוח קולט את כל פעולות המחשב (עכבר ומקלדת) באמצעות TKINTER ושולח אותן לשרת לפי הפרוטוקול.**

**לכן, יש שלושה SOCKETS שונים בפרוייקט זה. SOCKET ראשון אחראי אך ורק על שידור התמונות בין הלקוח לשרת. SOCKET שני אחראי על שליחת פקודות שהמשתמש בלקוח ביצע אל השרת. SOCKET שלישי אחראי על שליחת הודעות מהGUI-CLIENT אל השרת כפי שהוסבר למעלה. הרעיון לפצל את הSOCKETS של שידור התמונות ושליחת הפקודות (שניהם בין הSERVER לCLIENT המרוחק) הוא כדי לא ליצור עומס. שליחת התמונות היא איטית, משום שהיא כוללת הרבה מידע ואין צורך להכביד עליה עם שילחה של פקודות.**

2.2 פירוט רכיבי המערכת

שרת מחשב נשלט:

* Server.py
* Class Server
* Def \_\_init\_\_ - הפעולה הבונה של המחלקה. מגדירה את כל המשתנים במחקלה- שלושת הSOCKETS, פרוטוקולים, משתנה השולט ביכולת של הלקוח המרוחק לשלוט, שם של הDLL שבאמצעותו נלקחות תמונות המסך, גודל המסך, הפעולה מהDLL של תמונת המסך, משתנה שאליו נטען המידע של התמונה ומשתנה שאחראי על המשך פעולת הTHREADS.
* Def grab\_screen- פונקציה שלוקחת צילום מסך ומחזירה טיפוס מסוג IMAGE.
* Void ScreenShot- פעולה בשפת C. הפעולה מכינה את המסך וטוענת את כל המידע למשתמש שהיא מקבלת. המידע שהיא טוענת הוא בRGB. הפעולה נכתבה בשפת C ולא בפייתון כמו שאר הפרוייקט משום ששפת C יותר מהירה מפייתון ולכן זמן לקיחת צילום המסך מהיר יותר בC מאשר בפייתון.
* Def send\_stream- הפונקציה שולחת את המידע שהיא מקבלת ללקוח
* Def recv\_control\_data- הפונקציה מקבלת פקודות לפי הפרוטוקול שהמשתמש במחשב המרוחק עושה ומחזירה אותן.
* Def recv\_gui\_data-הפונקציה מקבלת על פי הפרוטוקול מידע

מהGUI-CLIENT ומחזירה אותו

* Def send\_screen\_shots- הפונקציה פועלת כל עוד הלקוח מחובר ושולחת את תמונות המסך. היא משתמש בפונקציה בשביל לקבל את התמונה עצמה, מגדירה טיפוס BytesIO וטוענת עליו את המידע לפורמט JPEG עם איכות נמוכה כדי שהגודל של התמונה ייקטן ויהיה ניתן לשלוח אותה. לאחר מכן, לוקחים את המידע ומעבירים אותו לפונקציה אשר שולחת אותו.
* Def control\_key- הפונקציה מחלקת את ההודעה שהתקבלה בנושא מקש מקלדת ומיישמת את המקש
* Def control\_click- הפונקציה מחלקת את ההודעה שהתקבלה בנושא לחיצות עכבר ומיישמת את הלחיצה.
* Def control\_motion- הפונקציה מחלקת את ההודעה שהתקבלה בנושא תזוזת עכבר ומיישמת את התזוזה.
* Def control\_scroll- הפונקציה מחלקת את ההודעה שהתקבלה בנושא גלגלת העכבר ומיישמת את הגלגול.
* Def check\_control\_data- הפונקציה בודקת איזו סוג של שליטה נמצא בהודעה וקוראת לפונקציה המתאימה.
* Def check\_gui\_data- הפונקציה בודקת אם המשתמש בתוכנת השרת בחר לאפשר או לא לאפשר שליטה ונותנת ערך מתאים למשתנה.
* Def handle\_gui- הפונקציה מקבלת מידע מהGUI ושולחת אותו לפונקציית היישום.
* Def start- הפונקציה כוללת את הלולאה הראשית, מבצעת LISTEN בכל אחד מן הSOCKETS, מריצה את תוכנת הGUI ומשתמשת בSELECT על מנת לנווט בין הSOCKETS השונים.
* Def close\_connection- הפונקציה סוגרת את כל הSOCKETS.
* Def main- הפונקציה מגדירה טיפוס מסוג SERVER ומריצה אותו. בסיום ההרצה הפונקציה קוראת לסגירת כל הSOCKERS.

**לקוח מחשב שולט**

* Client.py
* Class Client
* Def \_\_init\_\_- הפונקציה מקבלת את הSOCKETS ומגדירה את הROOT. היא עושה BIND לכל פעולות המחשב, מגדירה את הפרוטוקולים ואת הזמן ההתחלתי.
* Def scroll- הפונקציה נקראת כל פעם שהמשתמש משתמש בגלגלת העכבר והיא קוראת לפונקציה ששולחת את המידע לשרת.
* Def left\_click- הפונקציה נקראת כל פעם שהמשתמש לוחץ על הלחצן השמאלי בעכבר והיא קוראת לפונקציה ששולחת את המידע לשרת.
* Def right\_click- הפונקציה נקראת כל פעם שהמשתמש לוחץ על הלחצן הימני בעכבר והיא קוראת לפונקציה ששולחת את המידע לשרת.
* Def key\_down- הפונקציה נקראת כל פעם שהמשתמש לוחץ על מקש במקלדת והיא קוראת לפונקציה ששולחת את המידע לשרת.
* Def motion- הפונקציה נקראת כל פעם שהמשתמש מזיז את העכבר ועבר מפסיק זמן לפי הפונקציה שבודקת זאת והיא קוראת לפונקציה ששולחת את המידע לשרת.
* Def high\_pass\_filter- הפונקציה בודקת שעברו 0.2 שניות או יותר מהפעם האחרונה בה נשלח מיקום העכבר ומחזירה TRUE או FALSE בהתאם ומעדכנת את משתנה הזמן. השימוש בפונקציה זו הוא כדי שלא ייווצר DELAY בעכבר.
* Def recvall- הפונקציה מקבלת את האורך של ההודעה המצופה לפי הפרוטוקול ורצה עד שכל ההודעה מגיעה ואז היא מחזירה אותה.
* Def recv\_data- הפונקציה מקבלת הודעות לפי הפרוטוקול ומחזירה אותן.
* Def send\_control\_commands- הפונקציה מקבלת מידע ושולחת אותו לפי הפרוטוקול לשרת. המידע קשור לשליטת המחשב על ידי הלקוח.
* Def display\_screen- הפונקציה מקבלת מפונקציה אחרת את המידע של התמונות מפונקציה אחרת, יוצרת טיפוס מסוג BytesIO, כותבת לתוכו את המידע ופותחת טיפוס מסוג IMAGE ובמקום לתת בOPEN של IMAGE PATH לתמונה נותנים את הBUFFER. לאחר מכן, יוצרים טיפוס שניתן להצגה בTKINTER (IMAGETK) ומציגים אותו.
* Def correct\_key- הפונקציה בודקת אם אחד מן המקשים הבעייתיים (כתוב בBIND של TKINTER בצורה שונה מPYAUTOGUI) ומתקנת אותו ומחזירה את המקש הרצוי.
* Def correct\_mouse\_position- הפונקציה יוצרת STR הכולל את הX והY עם "," ביניהם ומחזירה את הSTR הרצוי.
* Def start- הפונקציה מריצה את הSELECT וקוראת לפונקציה שמציגה את המסך. הפונקציה רצה עם AFTER. אם יש ERROR כלשהו אז הפונקציה סוגרת את הSOCKETS ואת הROOT ומריצה את הCONNECT CLASS שעליו יורחב בהמשך.
* Def run- הפונקציה מריצה את start עם AFTER.
* Def close\_connection- הפונקציה סוגרת את הSOCKETS ואת הROOT.
* Class OpeningScreen
* Def \_\_init\_\_- פעולה בונה המגדירה את הROOT, ואת הכותרת. באמצעות BIND הפונקציה בודקת אם נלחץ מקש כלשהו וקוראת לפונקציה המתאימה.
* Def run\_client- הפונקציה נקראת כאשר המשתמש לוחץ על איזה כפתור שירצה ואז סוגרת את מסך הפתיחה ומריצה את CONNECT CLASS.
* Class Connect
* Def \_\_init\_\_- פעולה בונה שמגדירה את הSOCKETS ואת הROOT. לSOCKETS היא מגדירה timeout כדי שיהיה אפשר לדעת אם השרת באוויר. בנוסף, היא מגדירה כותרת, מקום להכנסת IP וכפתור אישור.
* Def key\_pressed- ניתן לאשר את הIP גם באמצעות לחיצת כפתור וגם באמצעות ENTER ולכן הפונקציה הזאת בודקת אם המשתמש לחץ ENTER וקוראת (אם כן) לפונקציה המתאימה.
* Def delete\_type- הפונקציה נקראת (עם BIND) כאשר המשתמש לוחץ על המקום של הכנסת הIP ואז מוחקת את הכתוב (ENTER IP).
* Def check\_list- פעולה רקורסיבית המקבלת רשימה ומחזירה TRUE אם ורק אם כל אחד מאבריה הוא STR של מספר ונמצא בתחום המתאים (בין 0 ל255 כולל).
* Def check\_ip- הפונקציה מקבלת IP מהמשתמש ובודקת שהוא הגיוני (באמצעות check\_list). אם לא, מציגה מתאימה ואם כן בודקת שניתן לעשות CONNECT לIP זה. אם לא, מציגה הודעה מתאימה, ואם כן היא סוגרת את הROOT, מגדירה טיפוס מסוג CLIENT ומריצה אותו.
* Def main- הפונקציה מגדירה טיפוס מסוג OpeningScreen ומריצה אותו.

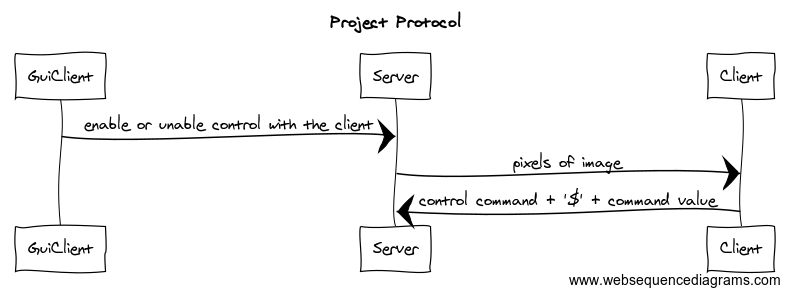
**ממשק גרפי לשרת**

* Gui\_client.py
* Class OpeningScreen
* Def \_\_init\_\_- פעולה בונה המגדירה את הROOT, כותרת וBIND למקשי המקלדת.
* Def run\_client- ברגע שהמשתמש לוחץ על איזה כפתור שירצה פונקציה זו תיקרא, תסגור את הROOT, תגדיר טיפוס מסוג ControlClient ותריץ אותו.
* Class ControlClient
* Def \_\_init\_\_- פעולה בונה המגדירה את ההודעות לשליטה, ROOT, כותרת ושני כפתורים שרק אחד מהם יכול להיות לחוץ והראשון שמאפשר שליטה לחוץ כברירת המחדל.
* Def connect\_to\_server- הפונקציה מתחברת לשרת.
* Def send\_data- הפונקציה מקבלת מידע ושולחת אותו לפי הפרוטוקול.
* Def enable\_control- ברגע שהמשתמש לוחץ על הפתור של אישור שליטה פונקציה זו נקראת ושולחת לשרת הודעה מתאימה באמצעות פונקציית השליחה.
* Def unable\_control- ברגע שהמשתמש לוחץ על הפתור של ביטול שליטה פונקציה זו נקראת ושולחת לשרת הודעה מתאימה באמצעות פונקציית השליחה.
* Def main- הפונקציה מגדירה טיפוס מסוג OpeningScreen ומריצה אותו.

2.3 דיון בנושא העיצוב הנבחר

פיצול למספר SOCKETS נעשה משום שצריך לשלוח כל הזמן מידע על הSOCKET של הSTREAM ותמונות הן בעלות מידע רב ולכן השליחה איטית. לכן, לא רציתי והעמיס עוד יותר על הSTREAM-SOCKET.

שימוש בPython 3.6 ולא בPython 2.7 נעשה משום שPILLOW (ספרייה) שינו את המחלקה IMAGE כך שלא חייבים לשמור לתוך קובץ או לקרוא מתוכו על מנת ליצור טיפוס מסוג IMAGE. חשוב לפרויקט, משום שקריאה מהדיסק לוקחת המון זמן ביחס לקריאה מBUFFER כמו של BytesIO.

1. עיצוב נתונים ופרוטוקולים

*ה*gui\_client *שולח הודעות* לserver *בנוגע לאפשר או לא לאפשר את שליטת* הclient*. ה*server *שולח הודעות ל*client *של צילומי המסך. ה*client *שולח הודעות ל*server *לפי הפקודות של המשתמש (אם הלקוח הזיז את העכבר, לחץ על מקש מסוים).*

*דוגמאות ל*commands:

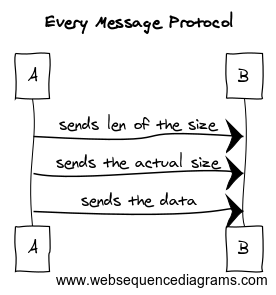
Key$f

Click$right

Motion$100,100

Scroll$-30

*כל הודעה שנשלחת מועברת לפי הפרוטוקול הבא:*



ראשית, מחשבים את האורך של ההודעה. לאחר מכן מחשבים את האורך של אורך הודעה. שולחים אורך זה, שולחים את האורך האמיתי ואז שולחים את ההודעה עצמה.

דוגמא:

Message = 'hi'

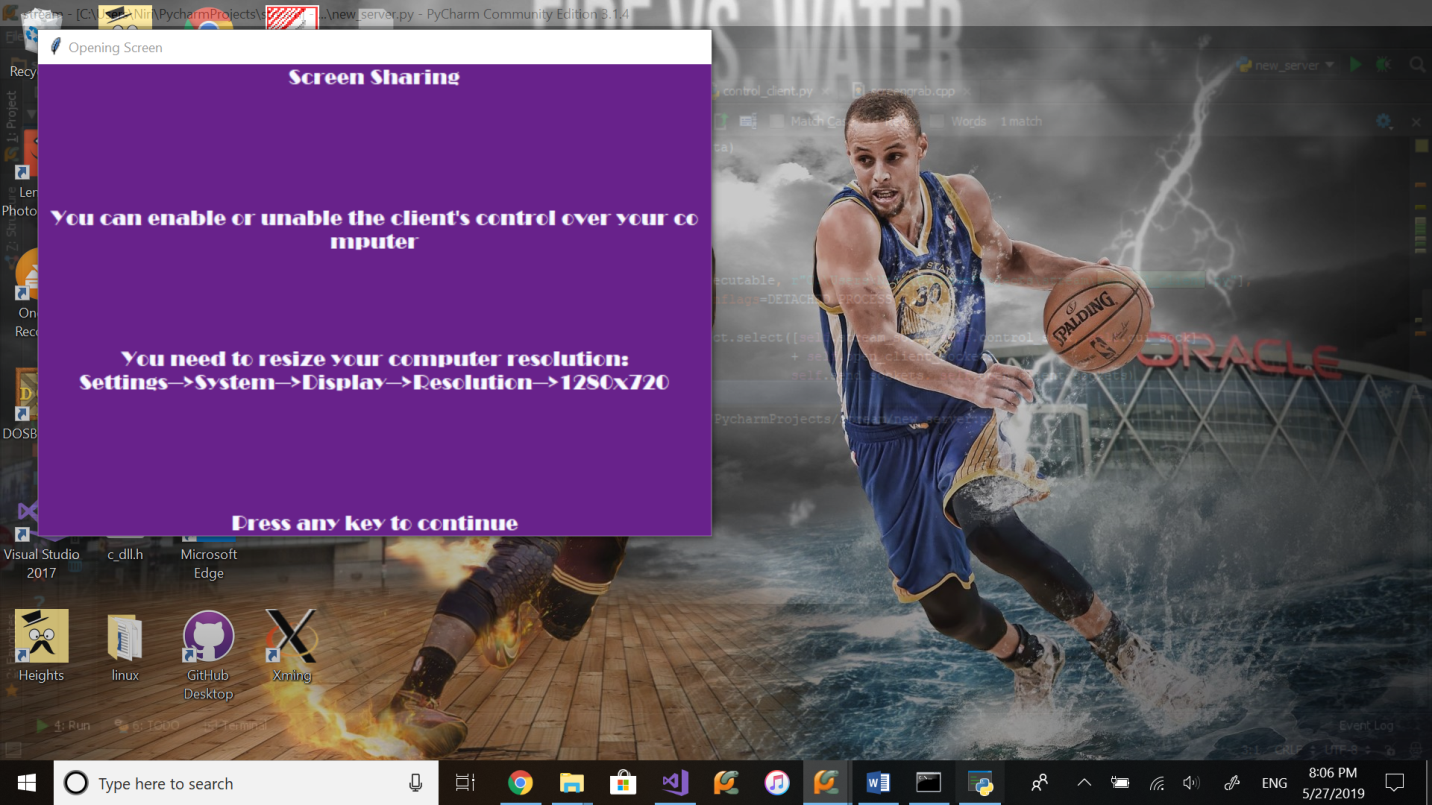
Size = str(len(Message)) = "2"

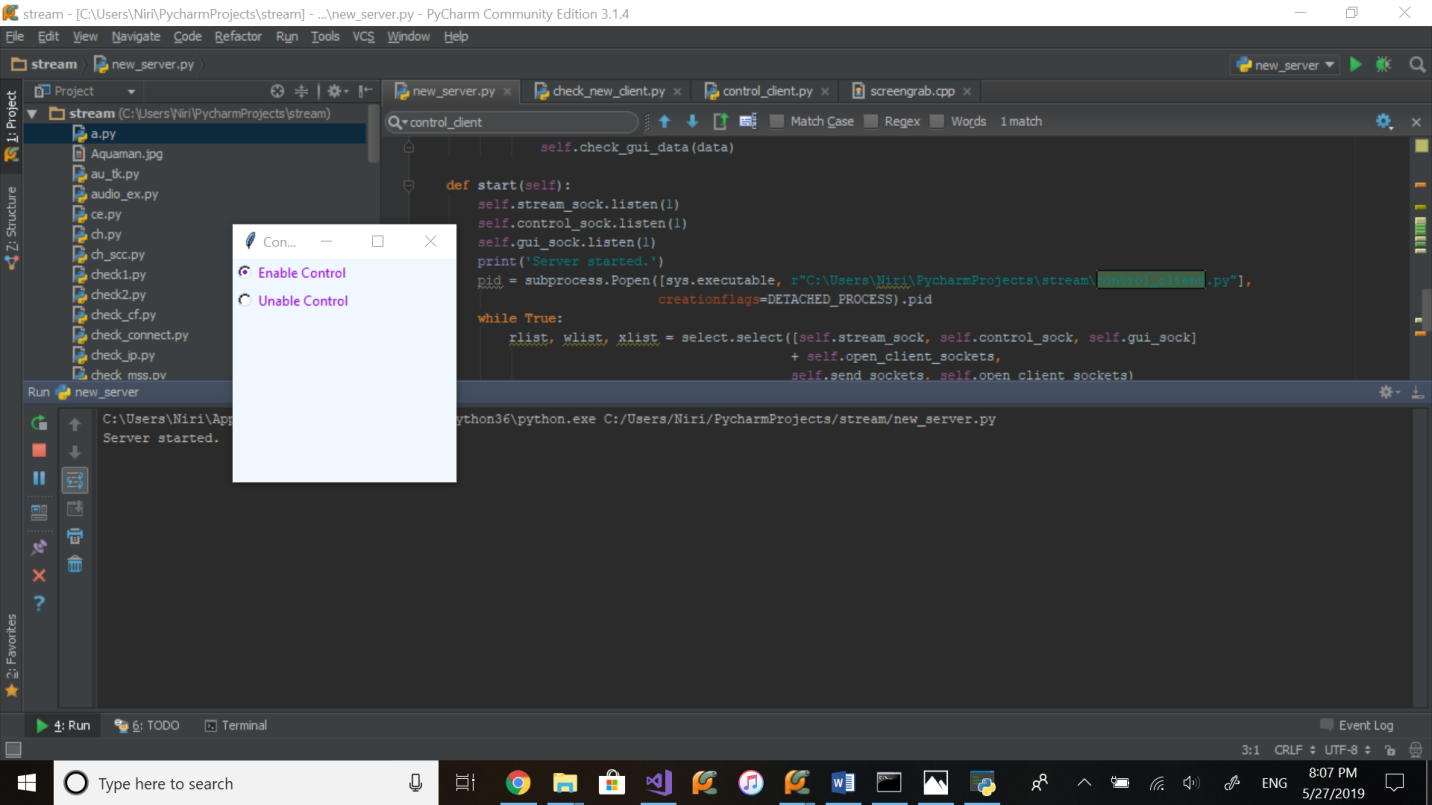
Size\_len = str(len(Size)) = "1"

ואז שולחים את Size\_len, אחרי זה את Size ואז את Message.

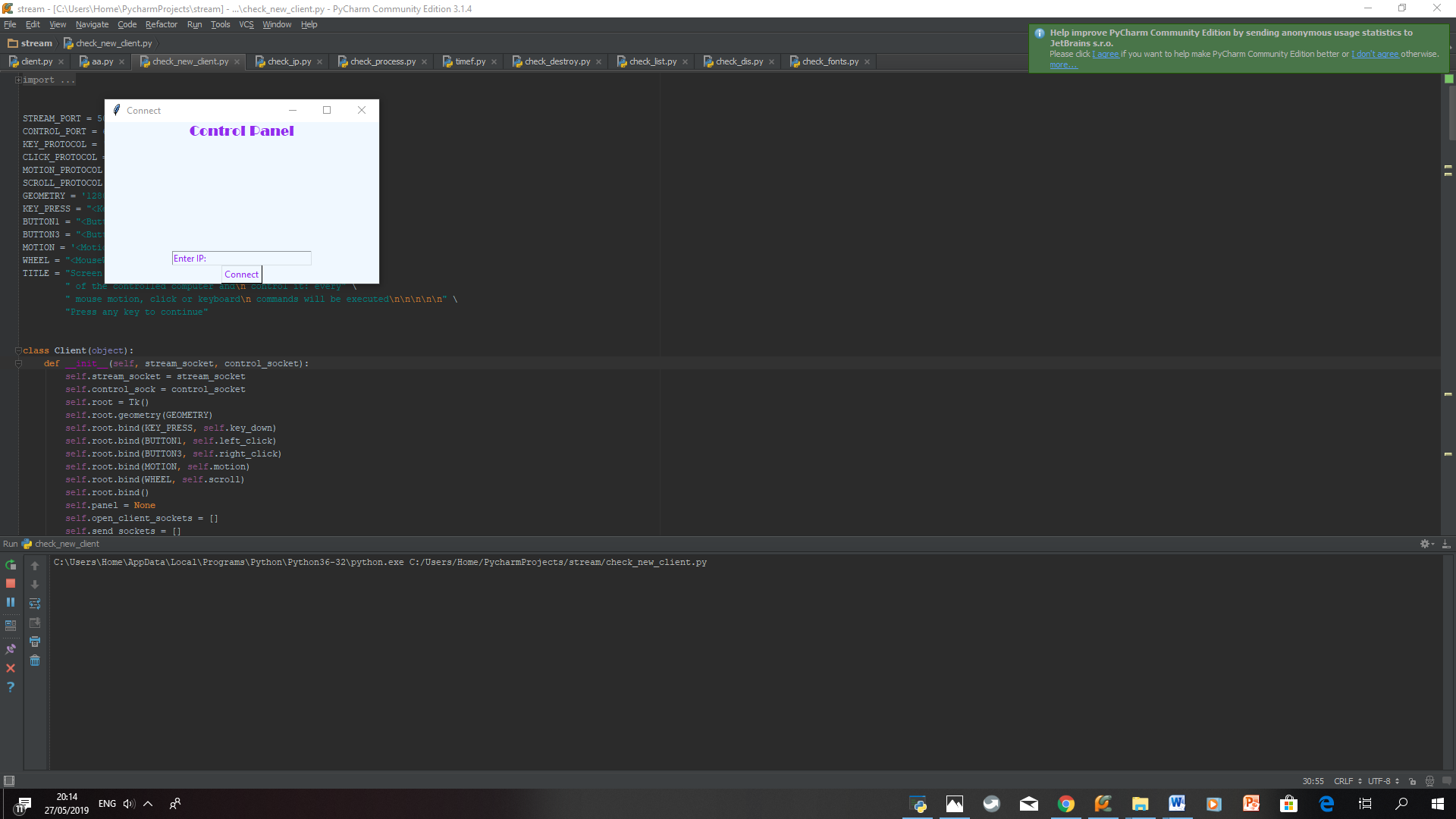
4. ממשק משתמש

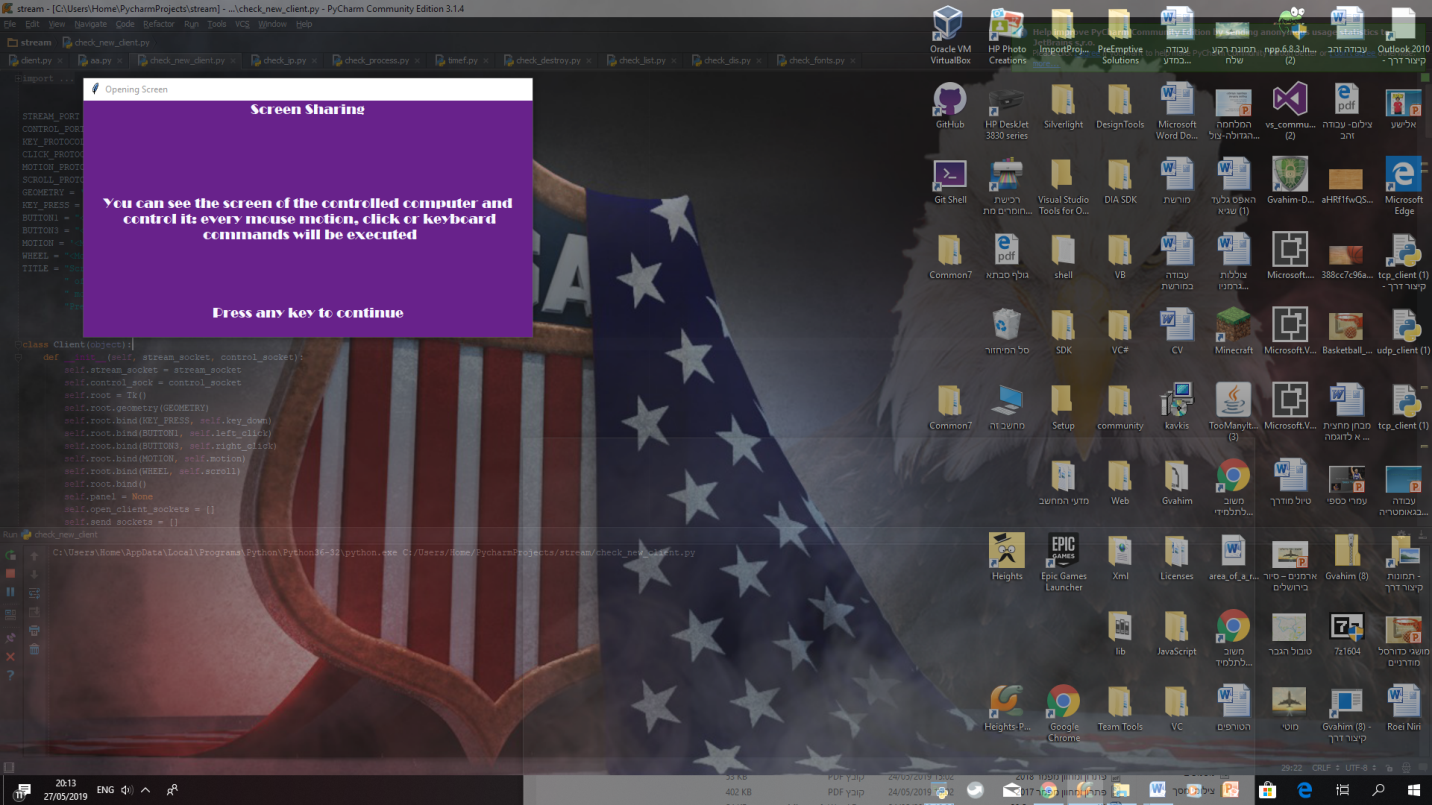
שרת מראה מסך פתיחה עם הסבר ראשוני: **לאחר שהמשתמש לחץ על מקש כלשהו:**



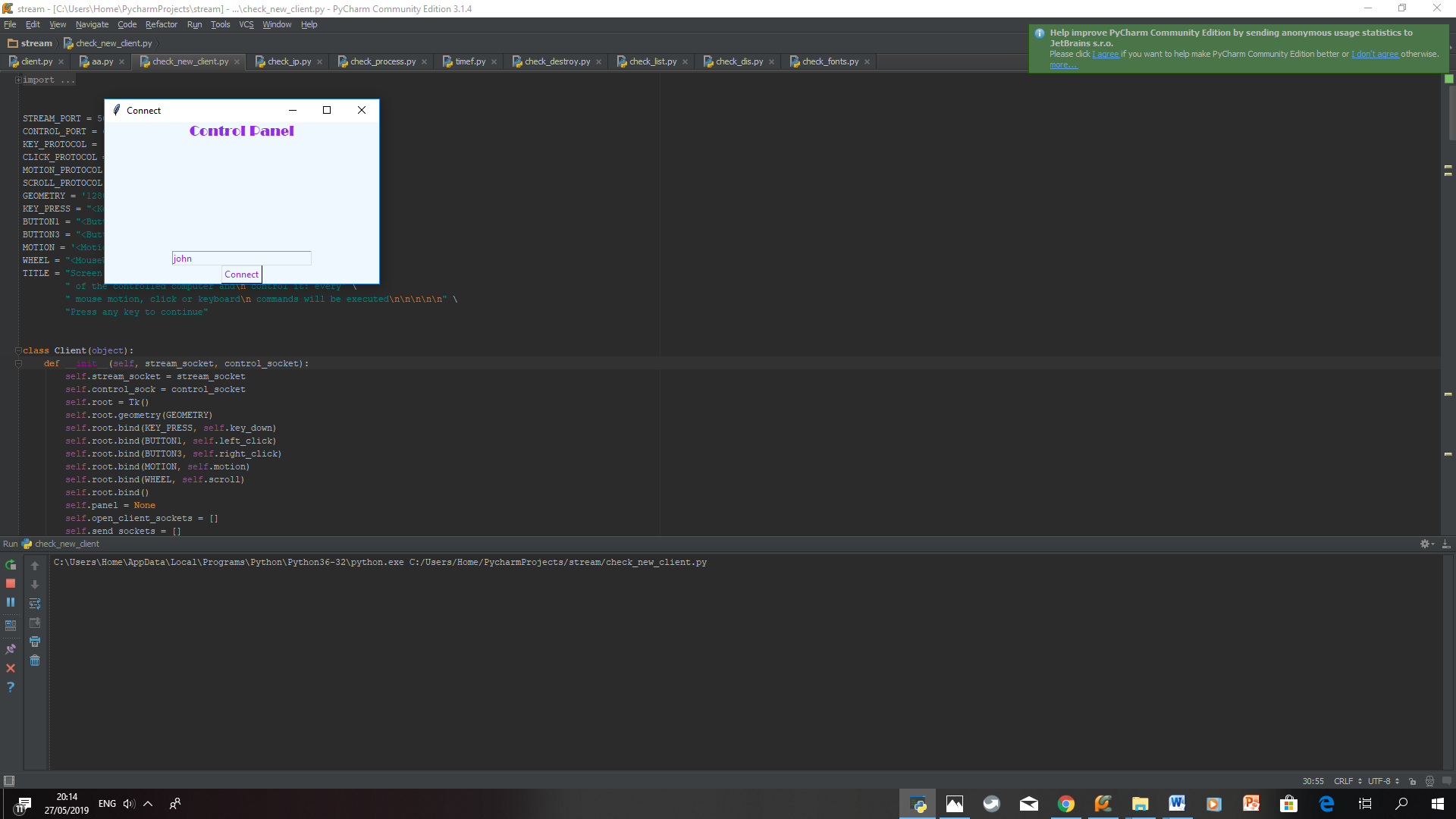
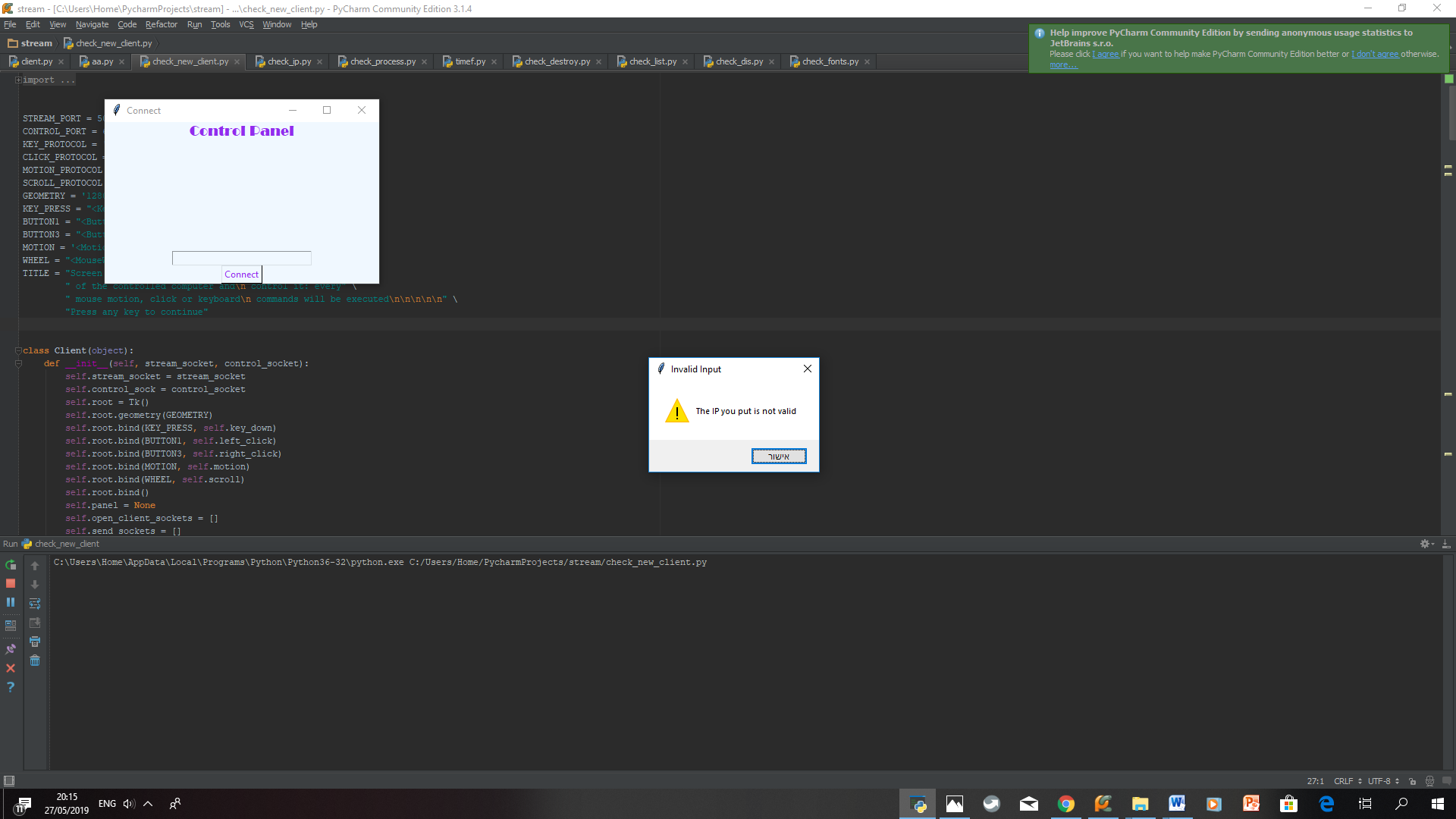


**לקוח מראה מסך פתיחה: לאחר שהמשתמש לחץ על מקש כלשהו:**

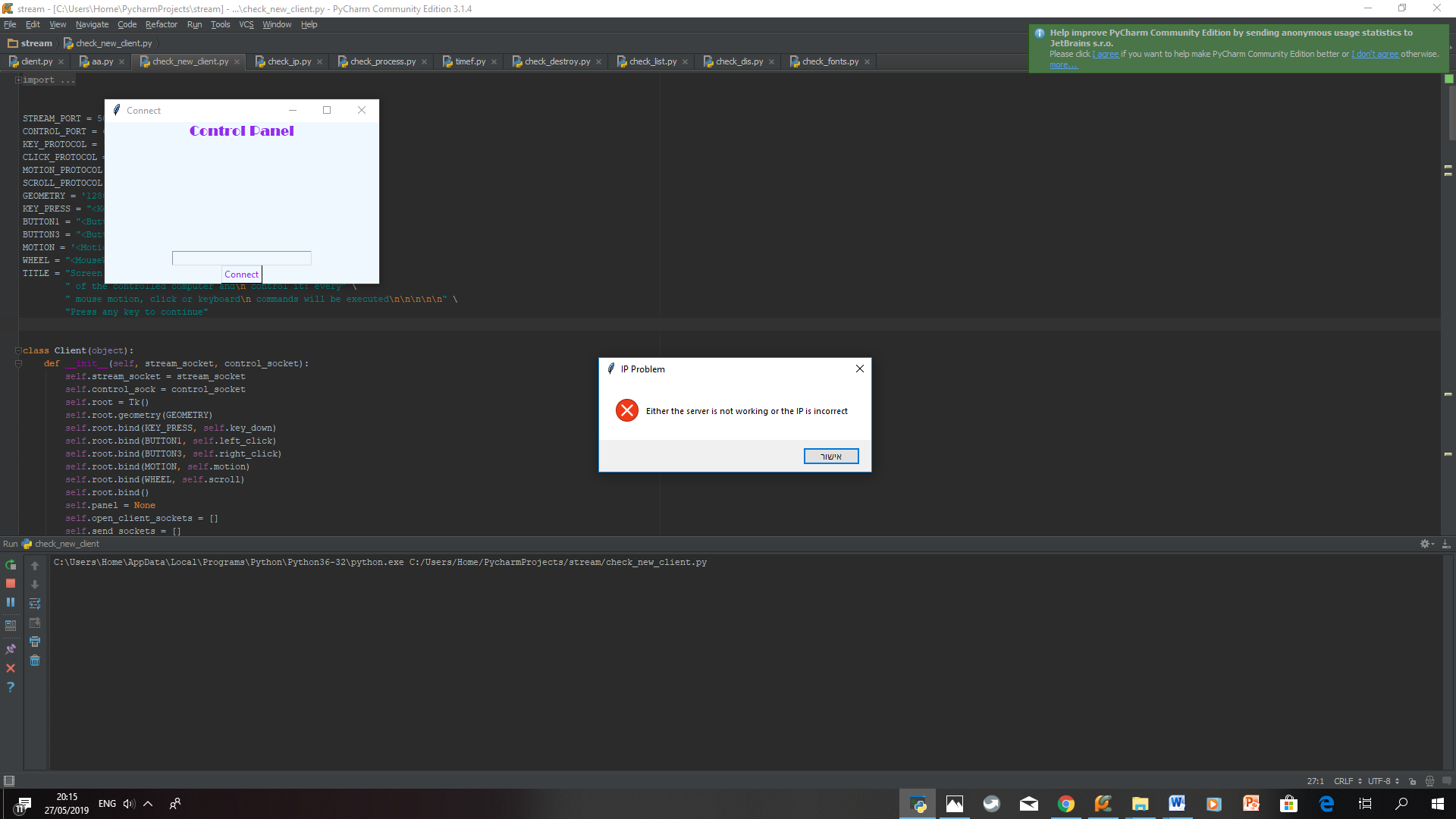


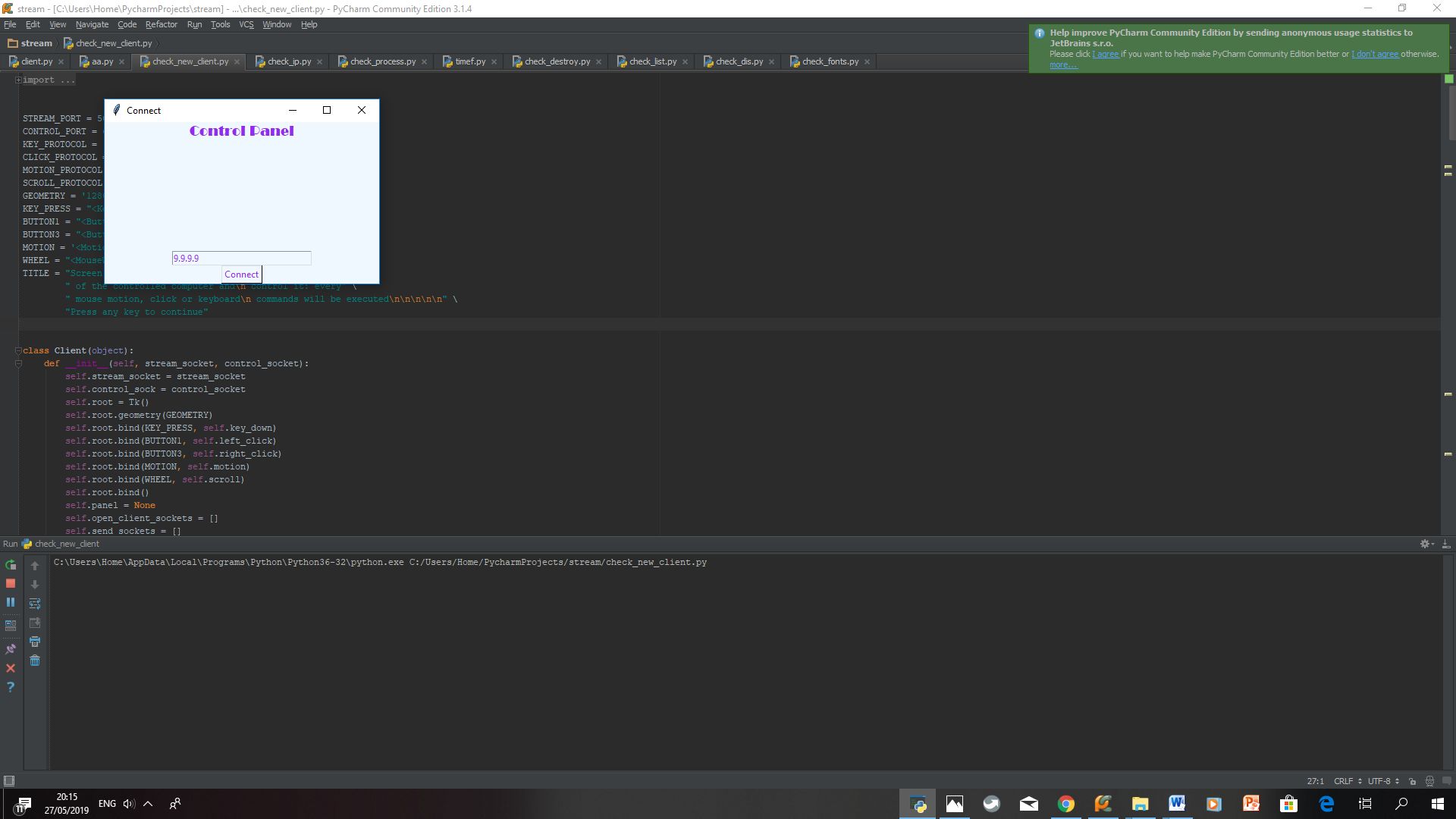


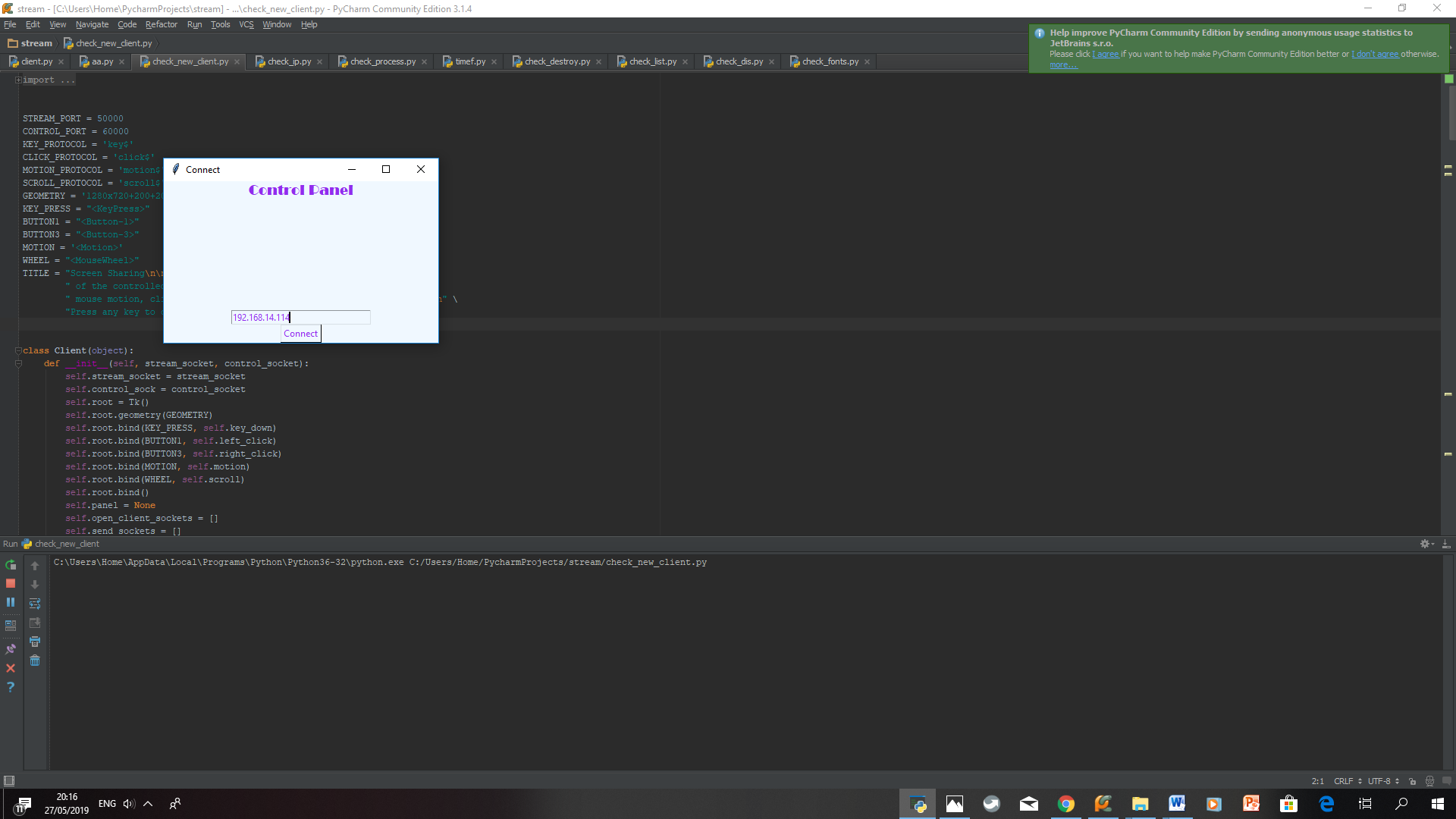
**המשתמש מנסה להכניס IP לא תקין: שמוצגת לאחר מכן:**



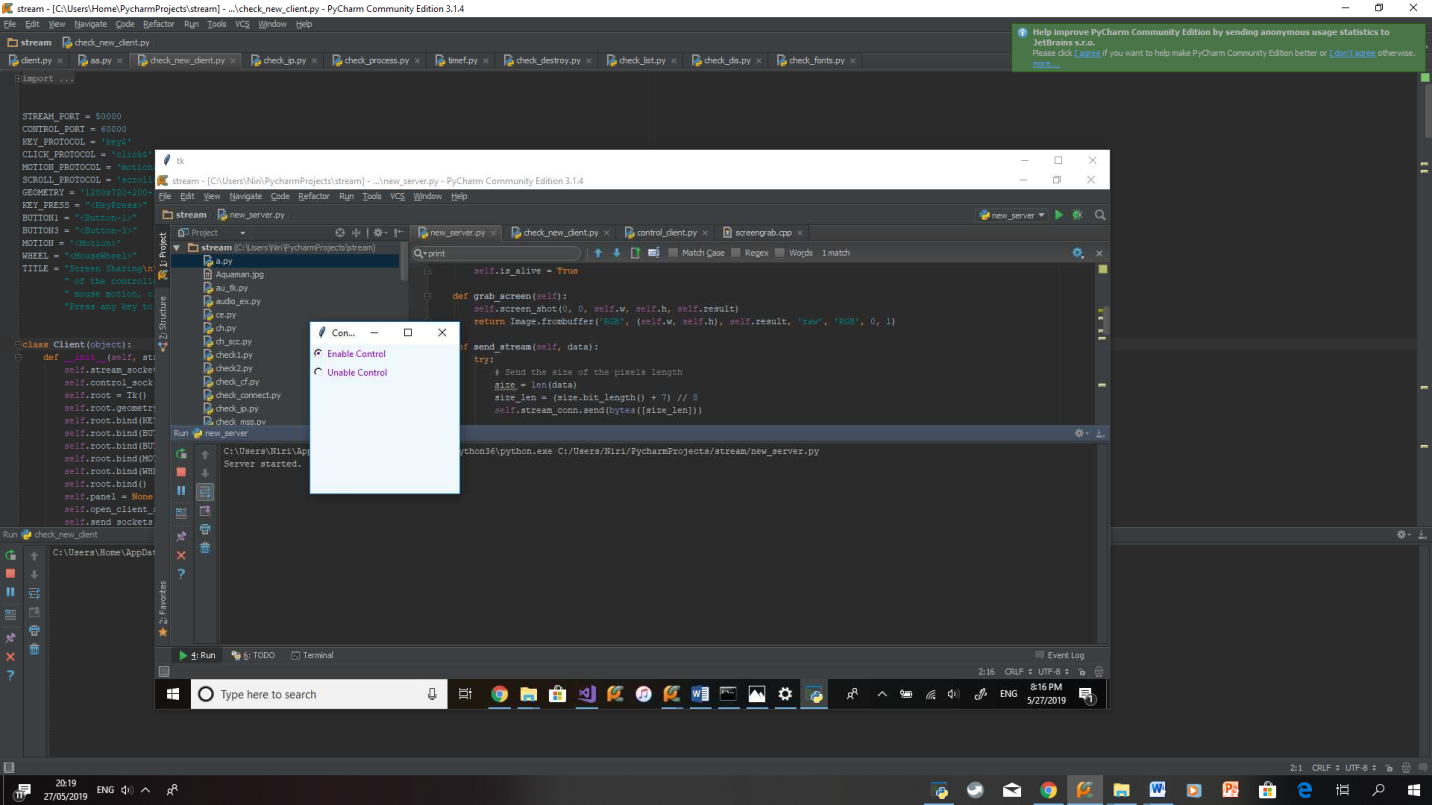
**המשתמש מנסה להכניס IP שונה מהIP של השרת: ההודעה שמוצגת לאחר מכן:**





**הלקוח מכניס את הIP של השרת:**

**המסך של השרת מוצג אצל הלקוח והלקוח יכול לשלוט במחשב השרת:**



5. נספחים

**המערכת מוגבלת ללקוח אחד ושרת אחד. זאת משום שמחשב אחד שולט במחשב אחר ולא יותר. לאחר שלקוח מתנתק משרת, השרת לא נסגר, אלא מחכה ללקוח חדש. לאחר ששרת נסגר, הלקוח לא נסגר, אלא סוגר את מסך השידור ומראה את המסך בו ניתן להכניס את כתובת הIP להתחברות נוספת עם אותו שרת או עם שרת אחר.**

**בחקר נמצא שJPEG זה פורמט של תמונות שניתן לאבד מידע ובכך להקטין את הגודל. בכך, ניתן לשלוח את המידע בגודל קטן יותר.**

הוראות הפעלה והתקנה

**יש להתקין באמצעות: pip install**

Socket

Pyautogui

PIL

Io

Select

Threading

Ctypes

Subprocess

Sys

Tkinter

Time

הפעלה:

**בצד של המחשב שנשלט: צריך באותה תיקייה את: server.py, gui\_client.py, SceenGrabber.dll**

**ואז יש להריץ את server.py**

**בצד של המחשב המרוחק ששולט מריצים את client.py**