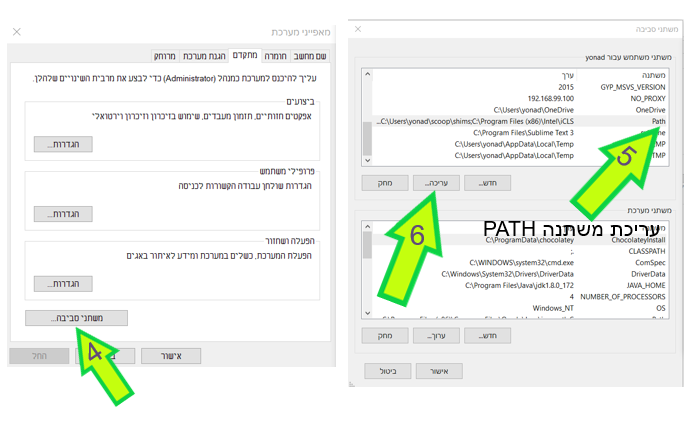
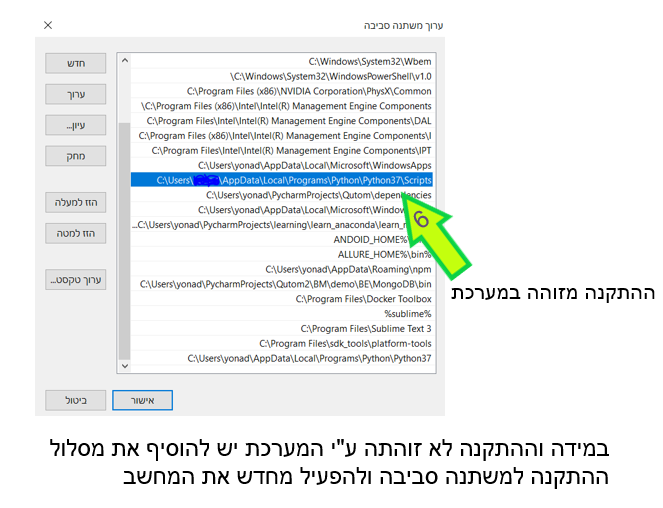
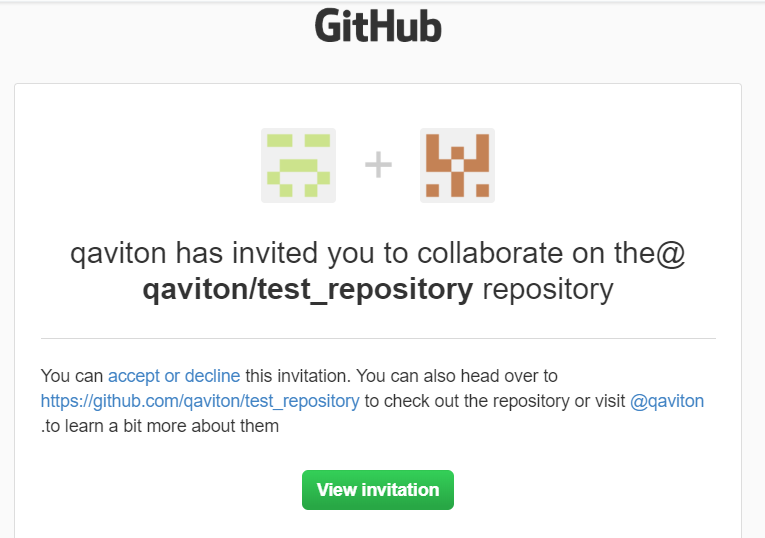
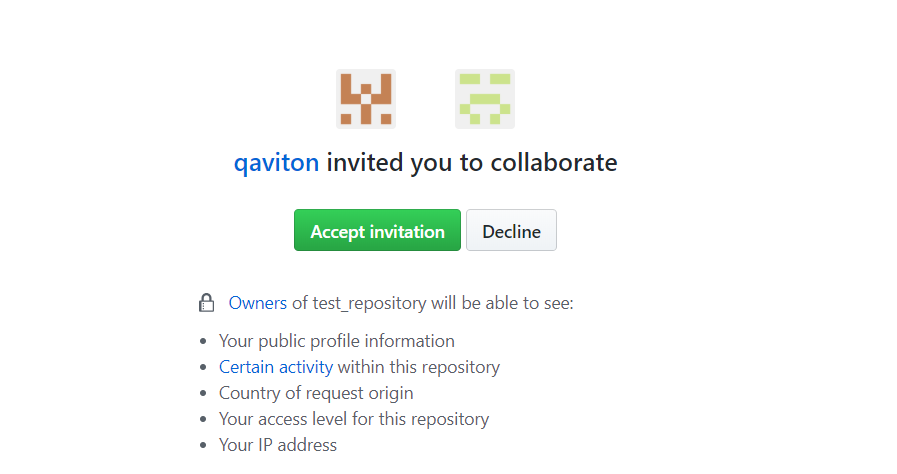
**חלק א' – התקנות ופתיחת משתמשים**

1. התקנת GIT למחשב - https://git-scm.com/downloads
2. להתקין PYTHON 3.7 במחשב- https://www.python.org/downloads/  
     
   במידה ואנחנו מתקינים את התוכנה על מערכת הפעלה וינדוס יש לוודא שההתקנה בוצעה כהלכה בצעדים הבאים:

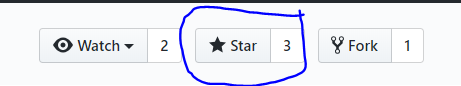
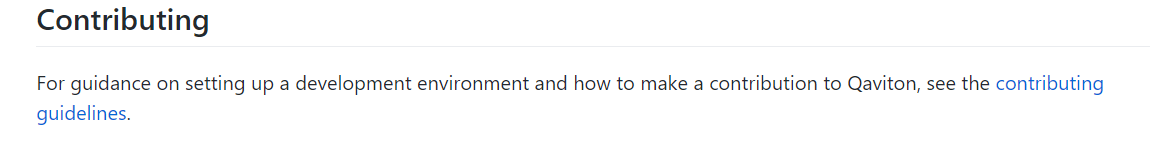
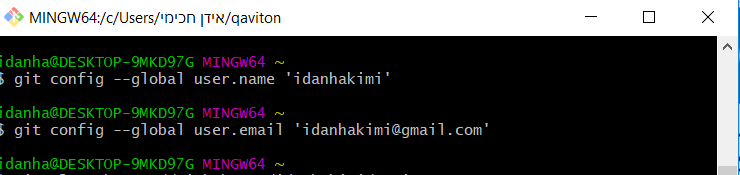
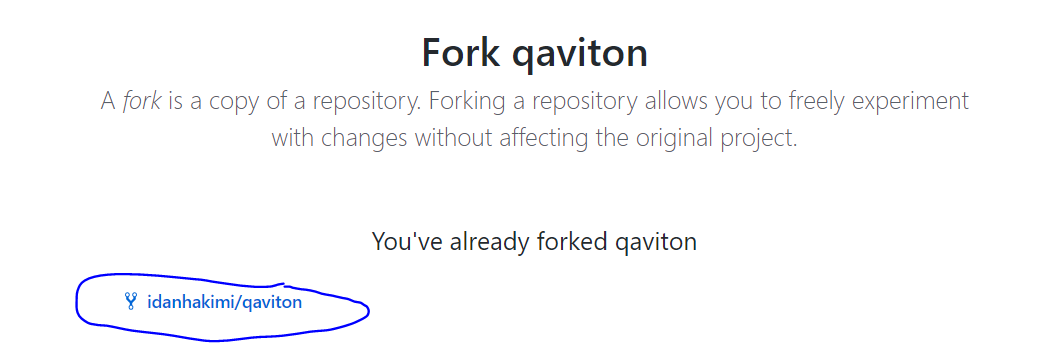
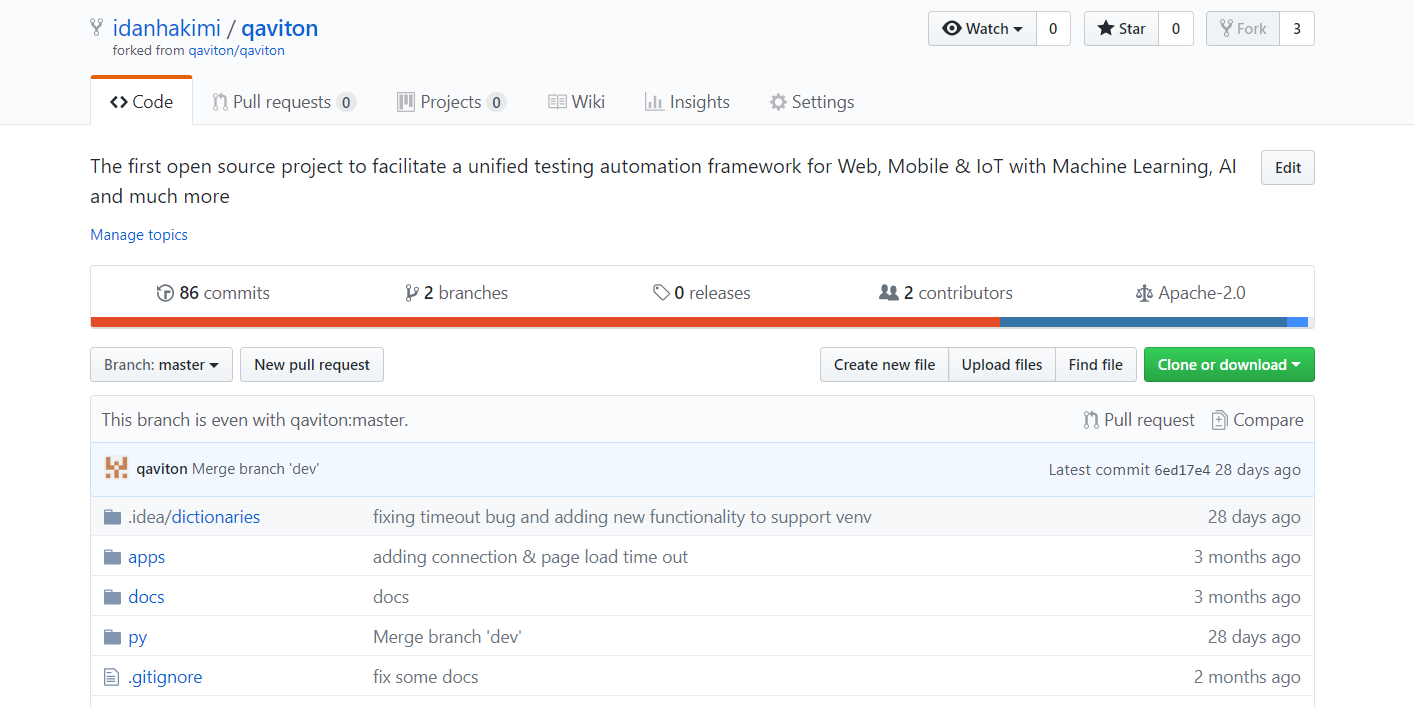
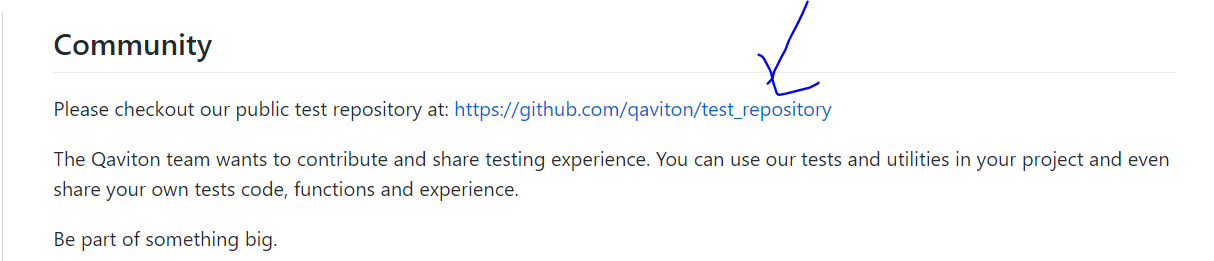


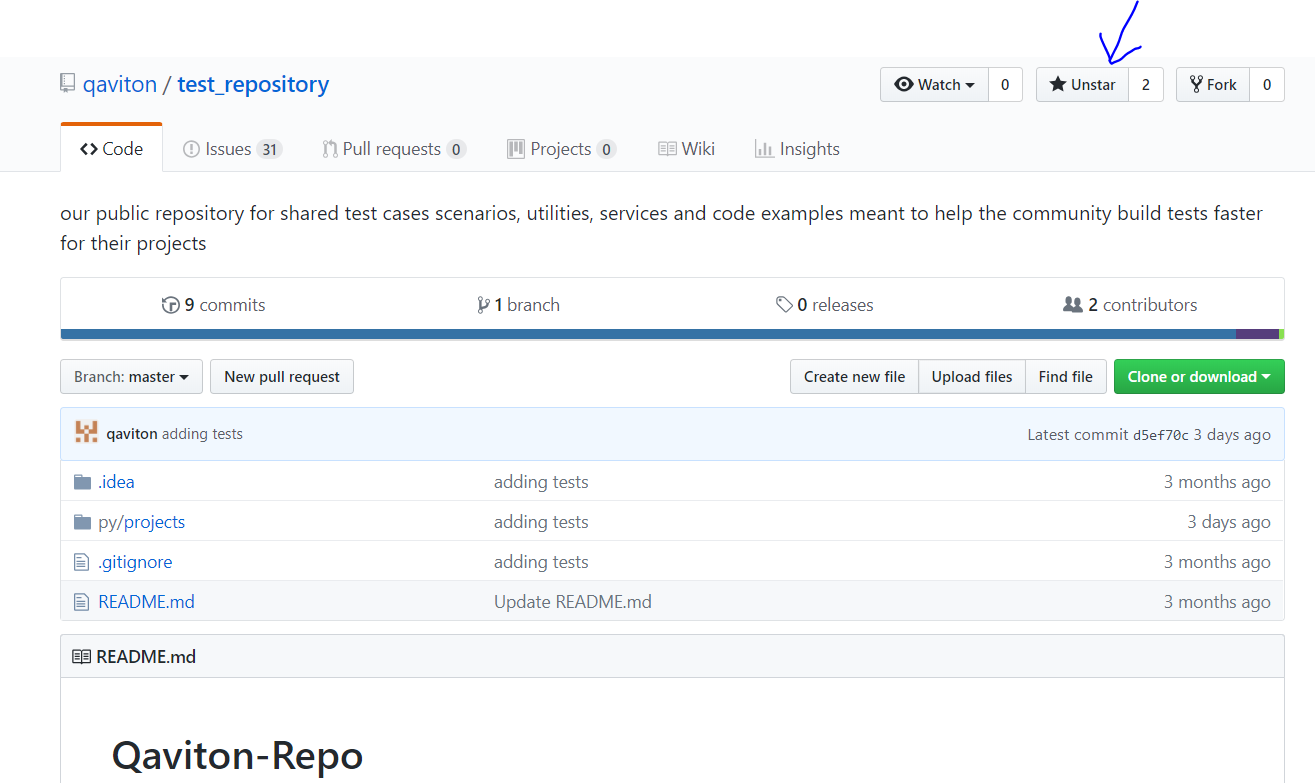


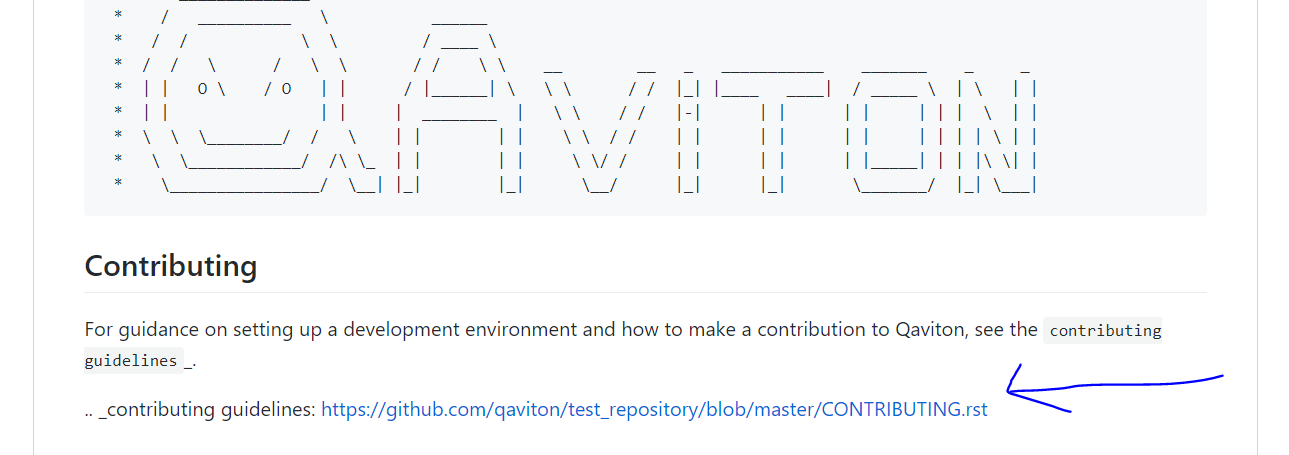


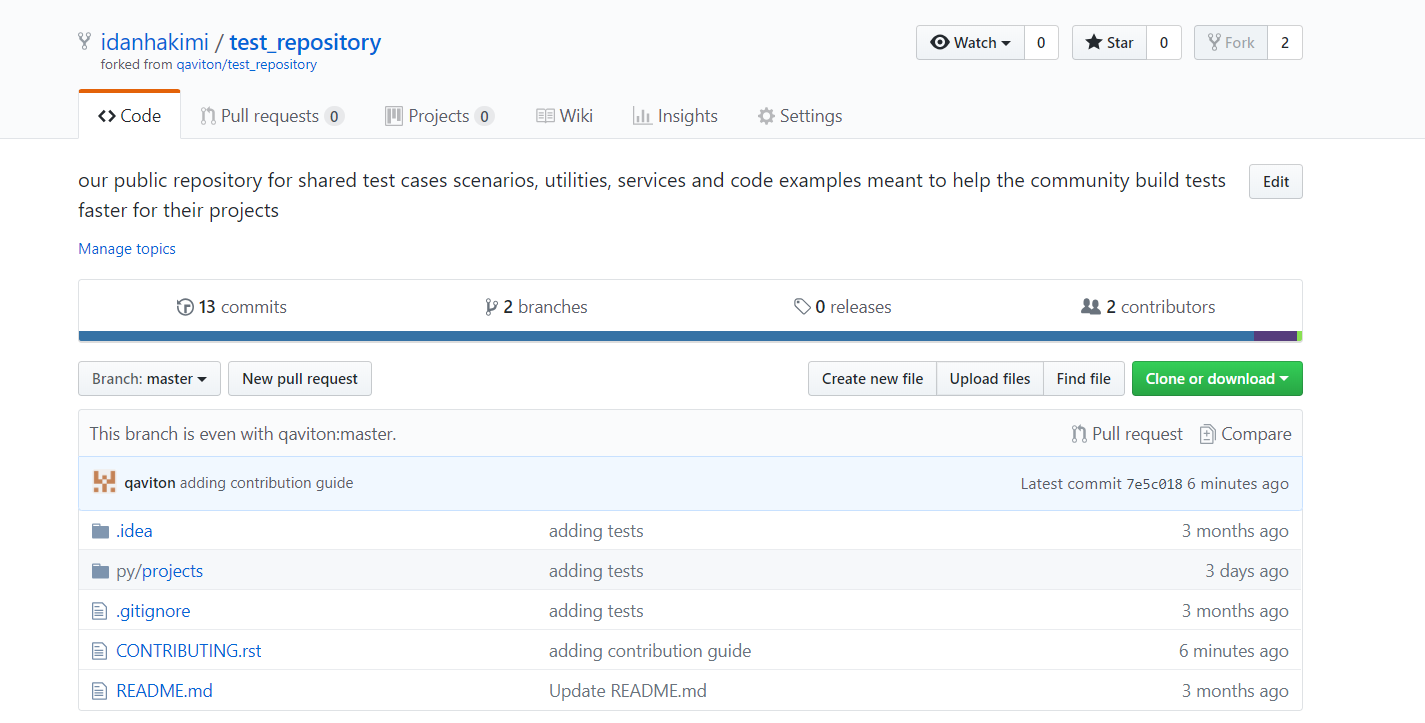
1. להתקין PYCHARM COMMUNITY EDITION - <https://www.jetbrains.com/pycharm/download/>
2. לפתוח משתמש ב-GITHUB
3. לעשות ווידוא של המשתמש במייל.
4. לשלוח בקבוצת OPENSOURCE בואטסאפ את השם משתמש בכדי שנוסיף אתכם כ-CONTRIBUTERS.
5. כעת תאשר במייל את ההזמנה ל-REPOSIDOTY של – QAVITON:  
     
     
     
   

**חלק ב' - FORK & STAR**

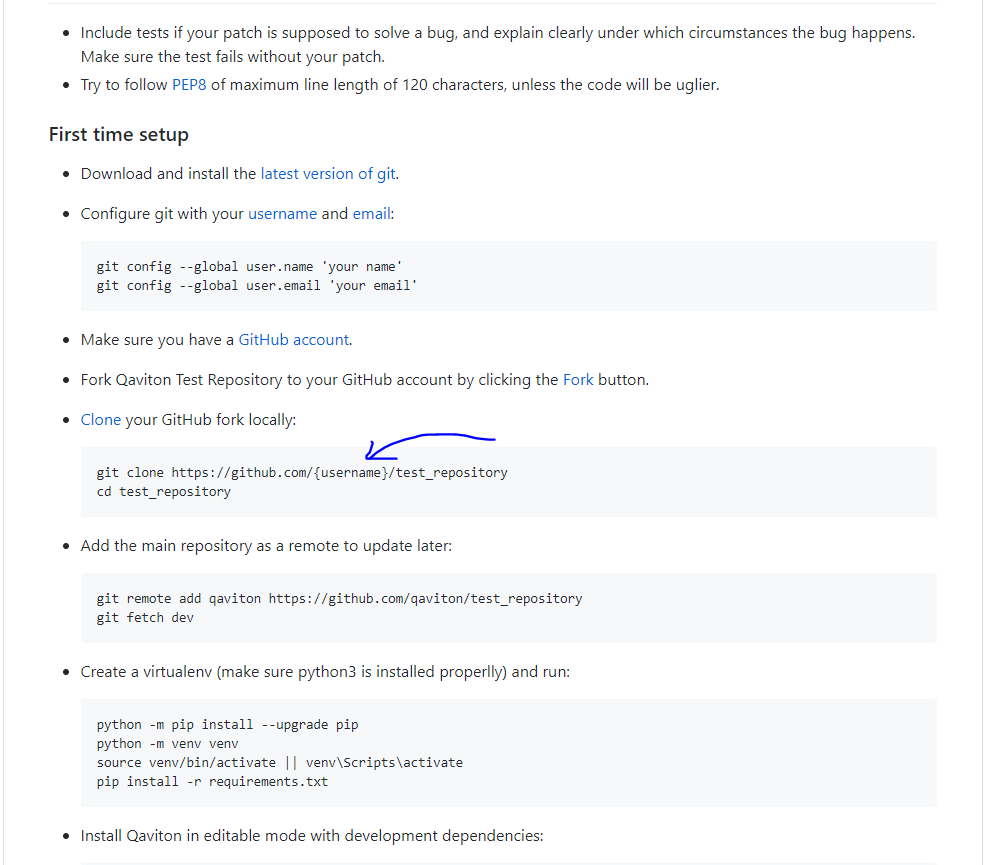
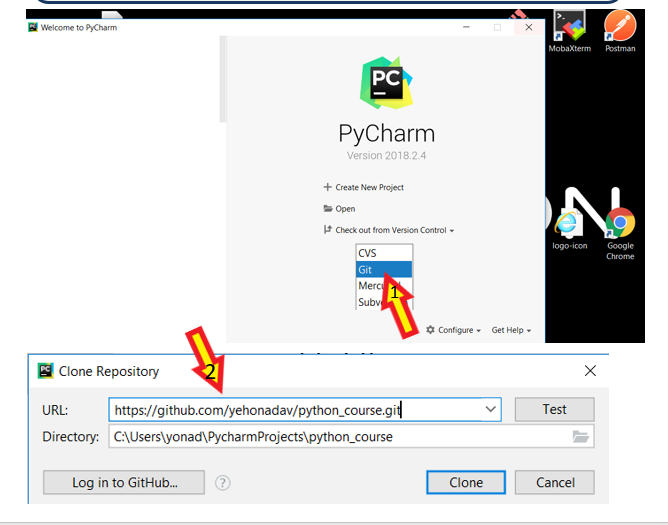
1. להיכנס ללינק הנ"ל:  
   <https://github.com/qaviton/qaviton>
2. לעשות לנו STAR :  
     
   
3. בהמשך הדף תחת קטגוריית "Contributing" ללחוץ על "Contributing Guidelines".  
     
   
4. להגיע למסך של FIRST TIME SETUP אשר נראה כך:  
     
     
   
5. לאחר שהתקנו את ה-GIT ופתחנו משתמש , נפתח את GIT BASH ונריץ את הפקודות אשר נמצאות בנקודה מספר 2.  
     
   
6. לאחר מכן נכנס ל-"FORK".  
   **Fork-** הינו לקחת את הקוד הראשי ולהעתיק אותו אל המשתמש המקומי לעבודה מקומית ללא פגיעה בקוד המקור.
7. לאחר שלחצנו על FORK הוא יקשר את הקוד למשתמש שלנו – כמו שאפשר לראות כאן, המשתמש כבר מקושר לקוד הנ"ל ( במידה ולא כל מה שנדרש זה ללחוץ על שם המשתמש שלנו.)  
   
8. כעת אנו נראה ש-QAVITON נכנס תחת המשתמש שלנו:  
     
   
9. כעת נעבור לדף: <https://github.com/qaviton/test_repository>  
   נחזור על כל הסעיפים בחלק ב' מסעיף 1 עד 8 ( STAR + FORK).  
     
   







**חלק ג' - CLONES**

1. לאחר מכן נמשיך ל-CLONE ונשתמש ב-"GIT BASH" במחשב שלנו או שימוש ב-PYCHARM.  
   במקום ה-'USERNAME' נשים את המשתמש שלנו, כל השאר אותו הדבר:  
   
2. אנו נשתמש ב-PYCHARM , כך זה ייראה:  
     
     
     
     
   כעת פה נשים את הקישור- [https://github.com/{username}/test\_repository.git](https://github.com/%7busername%7d/test_repository.git)
3. בכדי לוודא שאנו נמצאים ב-BRANCH הנכון נריץ את הפקודות הבאות:

git checkout -b dev

git branch -vv

יש לוודא שאנו נמצאים ב- /origin/dev ולא ב- /origin/master :

git branch --set-upstream-to origin/dev

git branch -vv

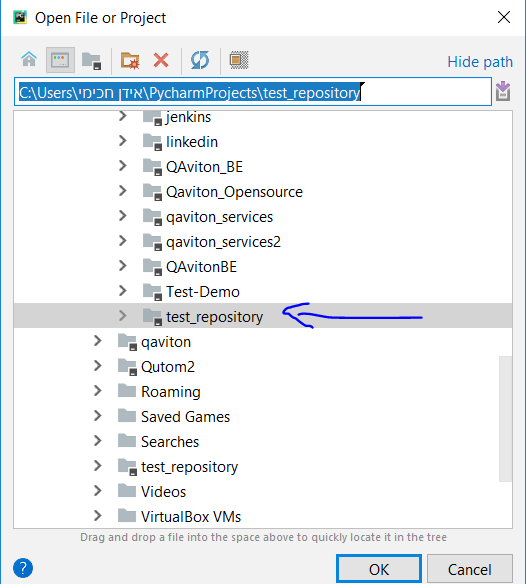
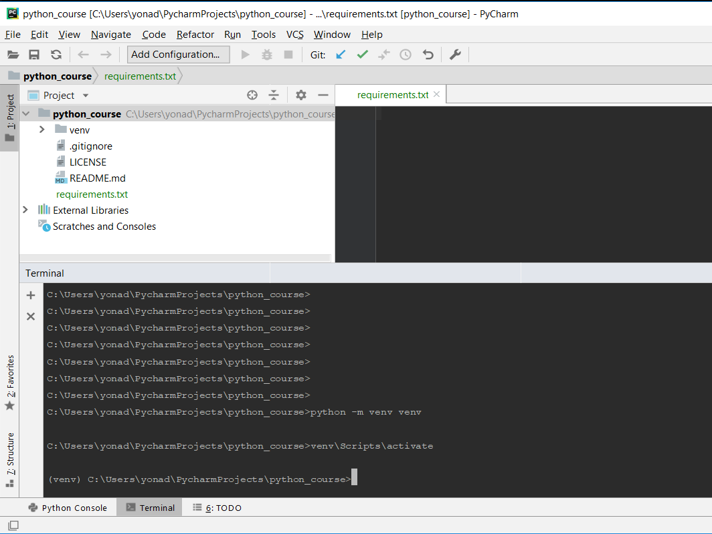
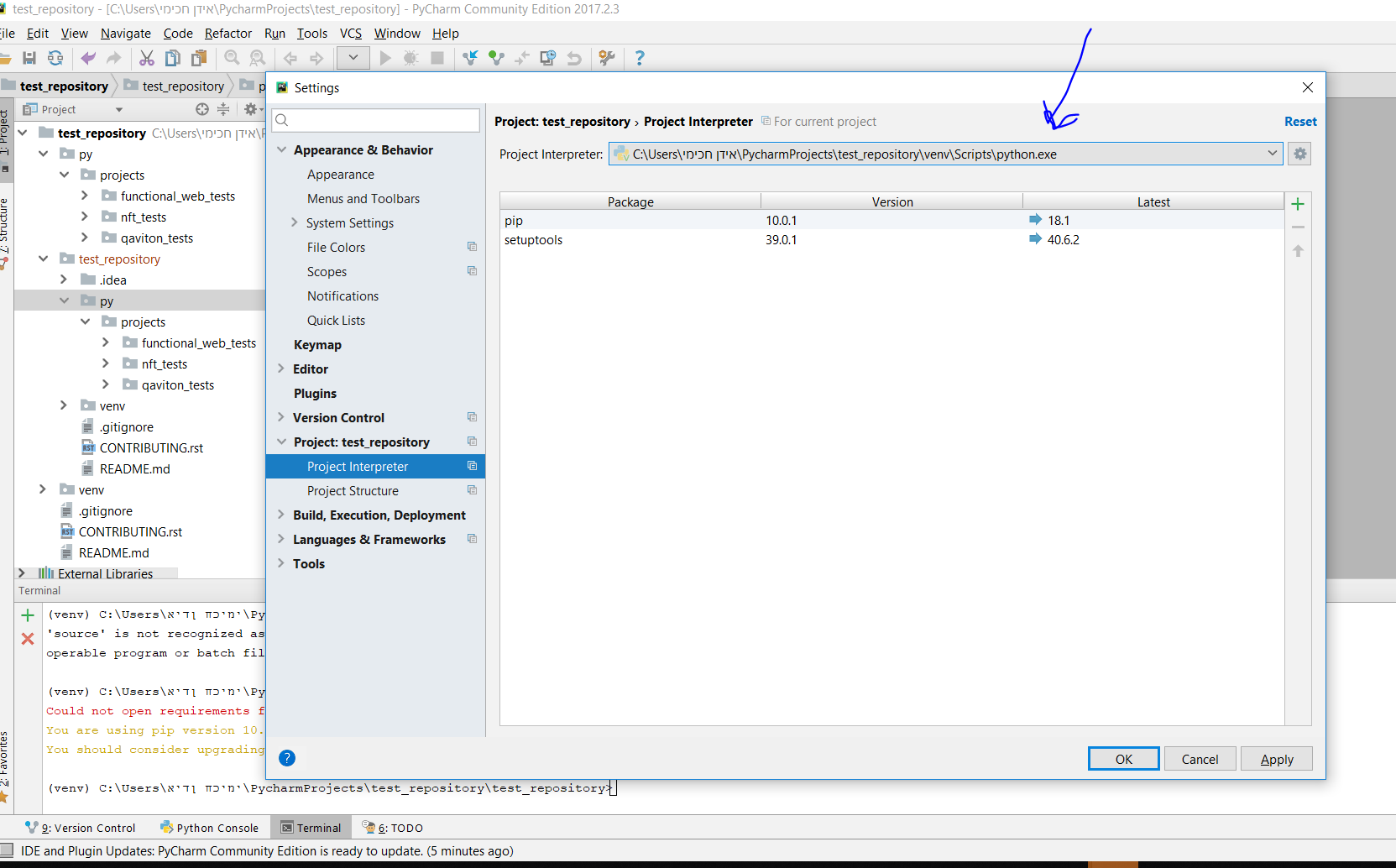
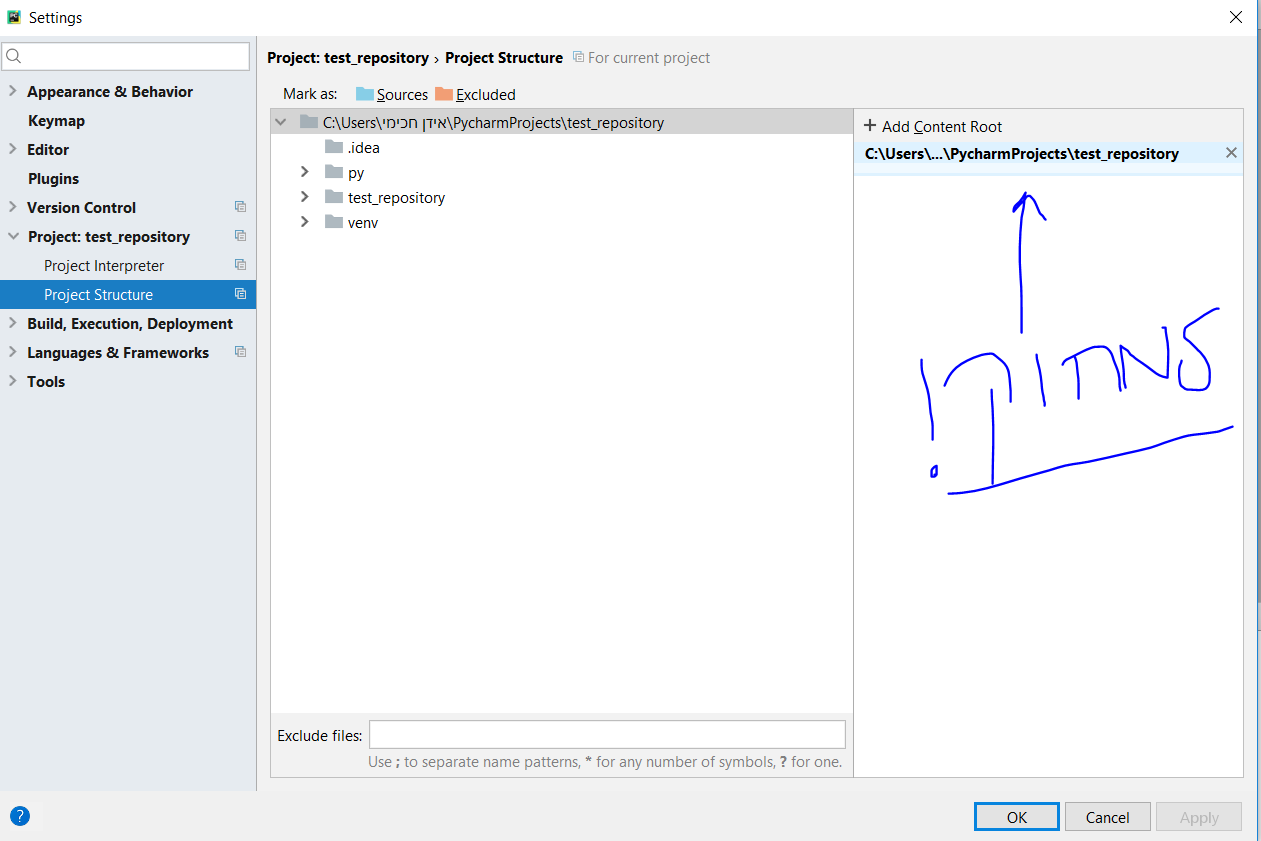
יש לוודא שהכתובת המרוחקת היא על שם המשתמש שיצרנו, לדוגמה: idanhakimi ולא תחת qaviton, נעשה זאת בעזרת הפקודה:

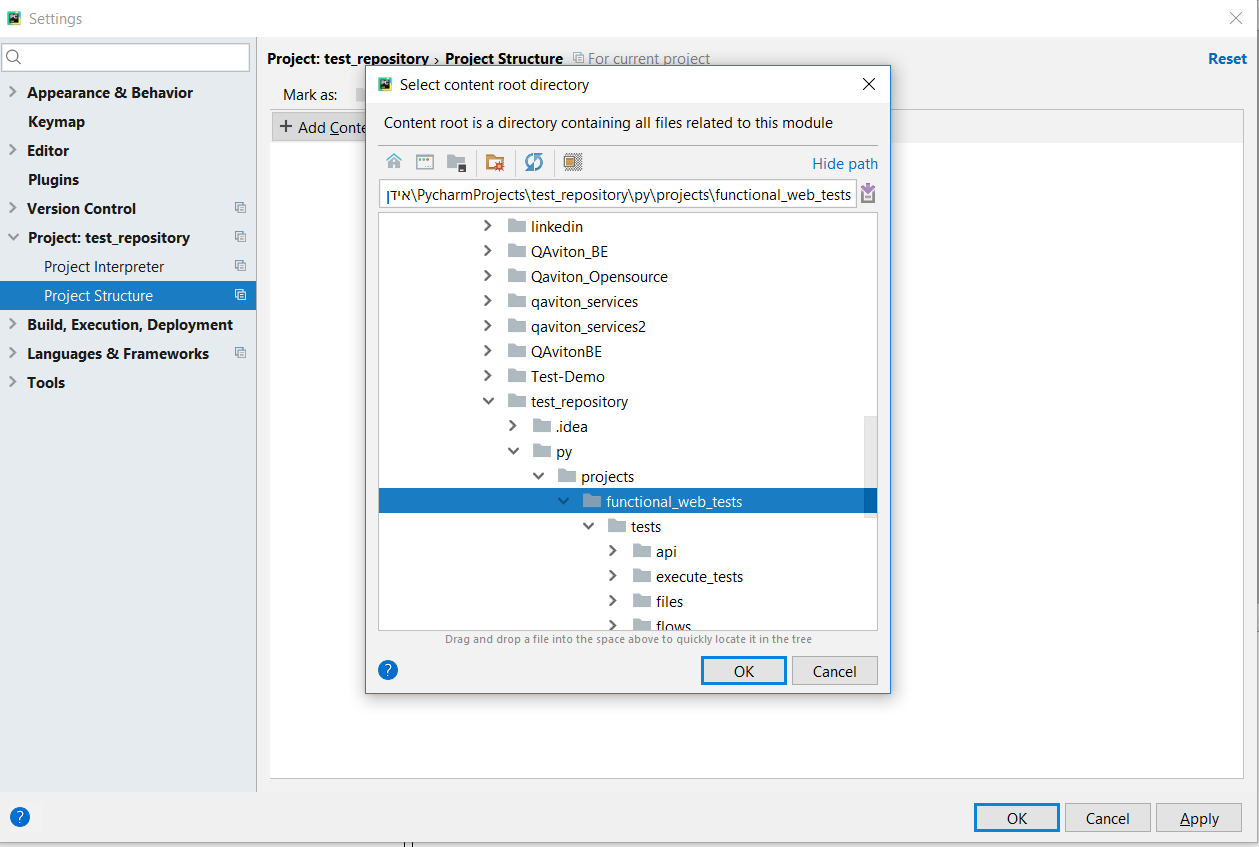
git remote -v

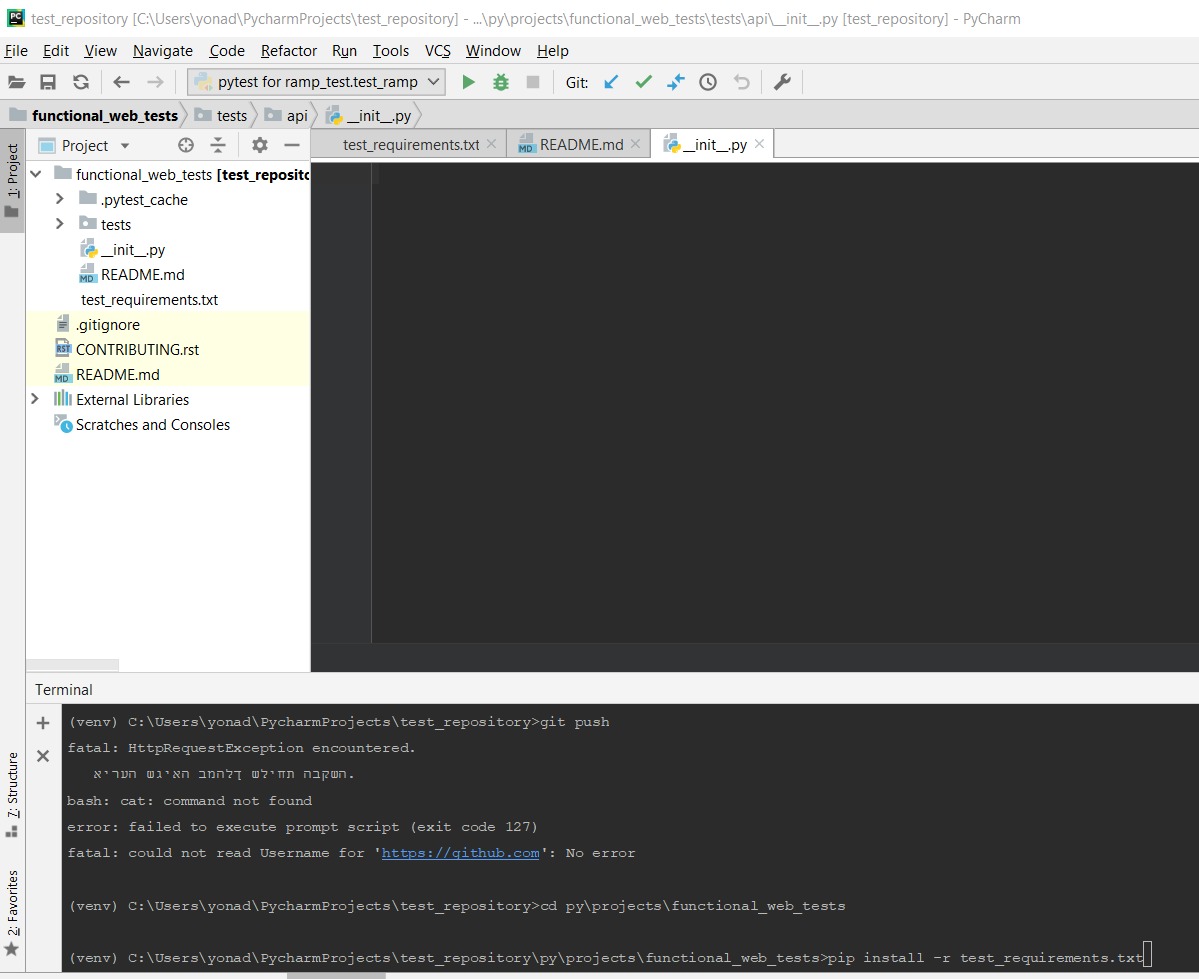
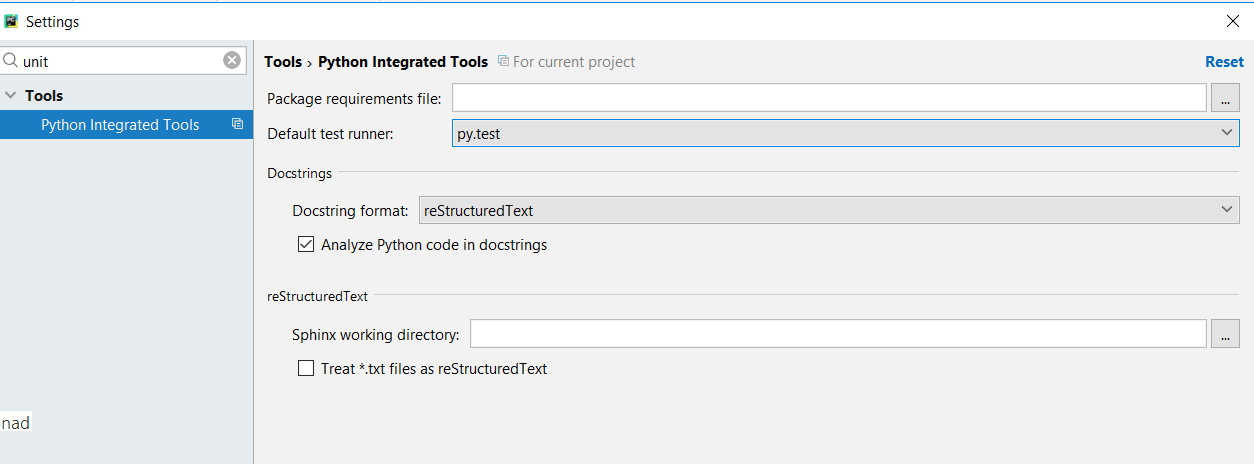
במידה וראינו שאנו לא עובדים מול הכתובת המרוחקת הנכונה, נשתמש בפקודה:

git remote set-url origin <https://github.com/idanhakimi/test_repository.git>

**חלק ד' – הגדרת סביבת עבודה ב-PYCHARM**

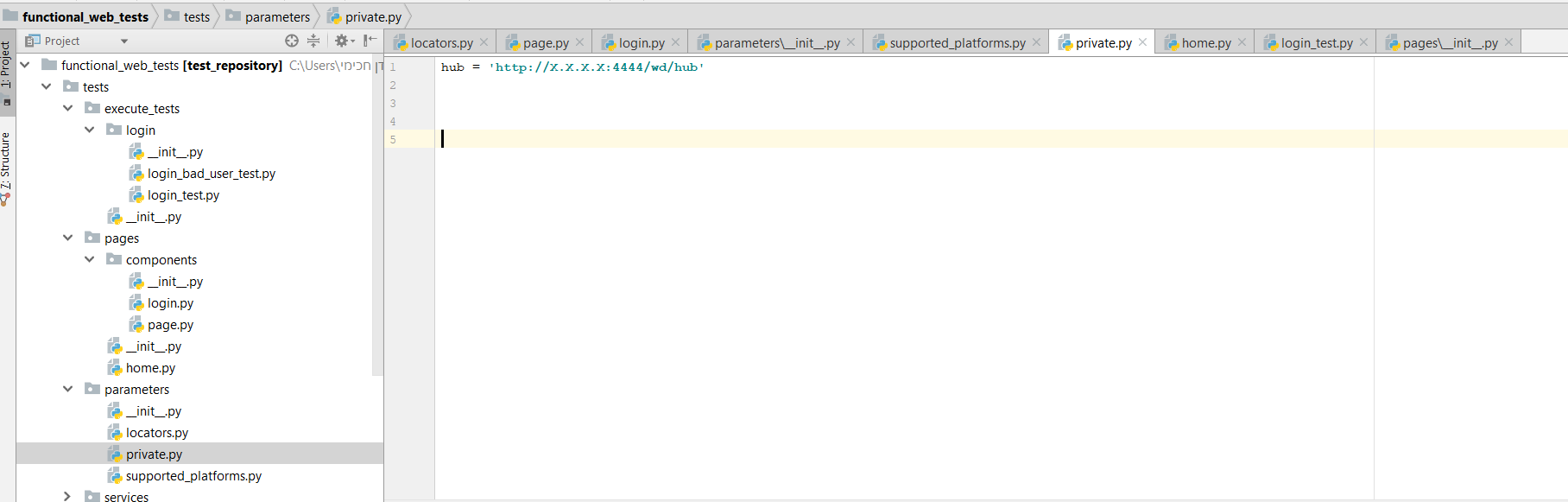
1. כעת נפתח את הפרויקט עם PYCHARM:  
     
   
2. במידה ולא מותקנת לנו סביבה וירטואלית:  
     
     
   לאחר ההתקנה עם הפקודה: python -m venv venv  
     
   אנו נרצה להיכנס לסביבה הוירטואלית ע"י הפקודה:  
   venv\Scripts\activate
3. כעת נפתח את ההגדרות של PYCHARM בצד שמאל למעלה ונגדיר לו INTERPRETER בתור הסביבה הווירטואלית שהתקנו בסעיף הקודם.  
   
4. כעת נגדיר את נתיב הפרוייקט הרצוי.  
   כל שעלינו לעשות הוא:  
   להיכנס ל-PROJECT STRUCTURE ולמחוק את ה-PATH שרשום.  
     
   

לאחר מכן נוסיף "add content root":  
נוסיף את הפרויקט שעליו נרצה לעבוד, כרגע אנו עובדים על "functional\_web\_tests"  


כעת ב-TERMINAL נצטרך ללכת ל-PATH שמתואר באיור ולעשות :  
pip install -r test\_requirments.txt  
  
  
  
6. השלב האחרון הינו הגדרת ה-RUNNER TEST שאיתו נעבוד, במקום "unittests" נשנה ל-"py.test" או "pytest  
  
לאחר מכן נלחץ על אישור ונפתח מחדש את PYCHARM.  
   
  
  
7. כעת כאשר אעבוד עם התיקייה של ה-TESTS, במידה ואני CONTRIBUTER מורשה אצל QAVITON, אני אצטרך להוסיף תחת תיקיית parameters קובץ בשם private.py, אשר יכיל את שורת הקוד הבאה:

hub = **'http://X.X.X.X:4444/wd/hub'**

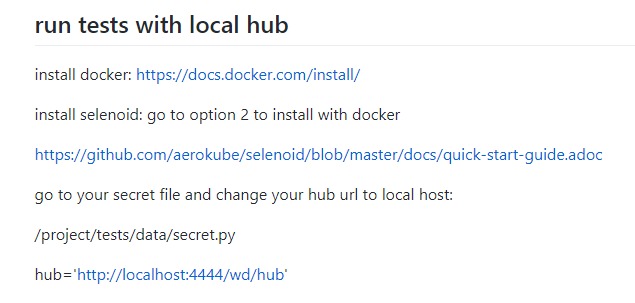
זוהי למעשה כתובת של שרת אשר מספק DRIVER'ים מרוחקים כמו לדוגמה: CHROME,ANDROID,FIREFOX.



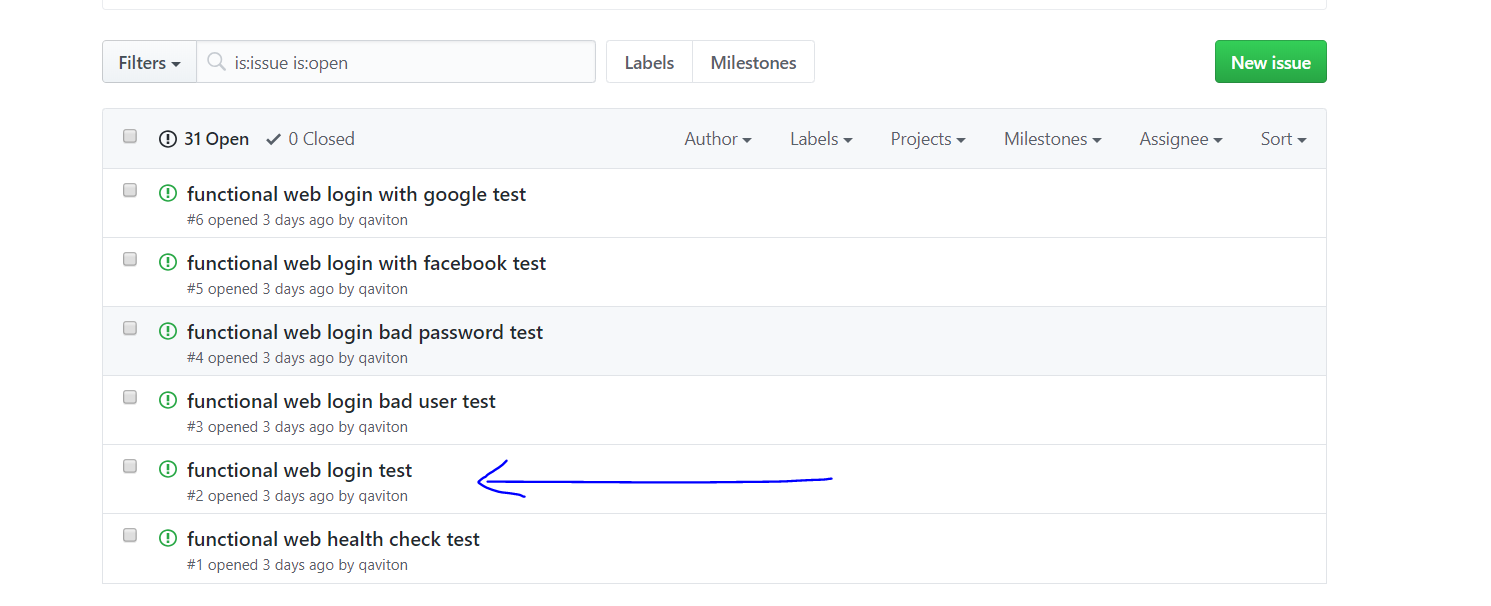
ניתן לעבוד עם שרת מקומי:

1. במידה ואין לנו שרת מרוחק , נוכל להתקין שרת מקומי:

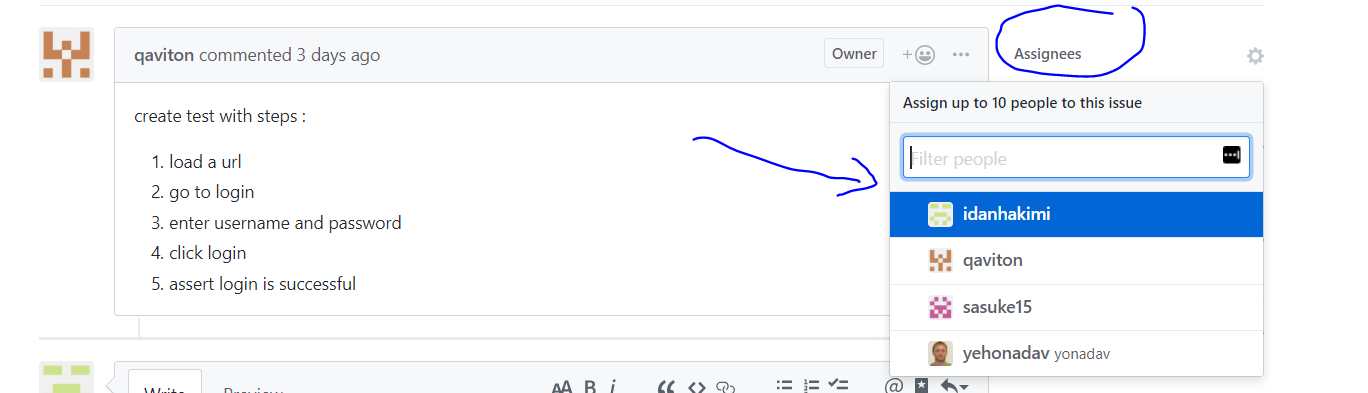
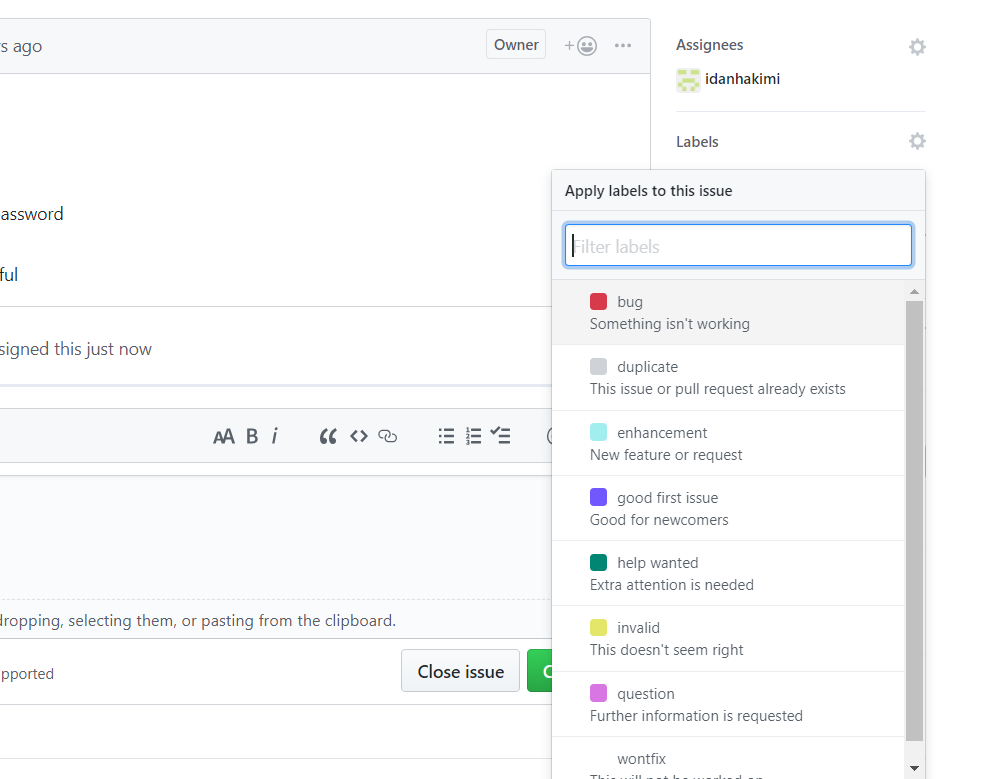
אנו ניכנס לקישור הבא ונלך ל-SECTION המתואר בתמונה:

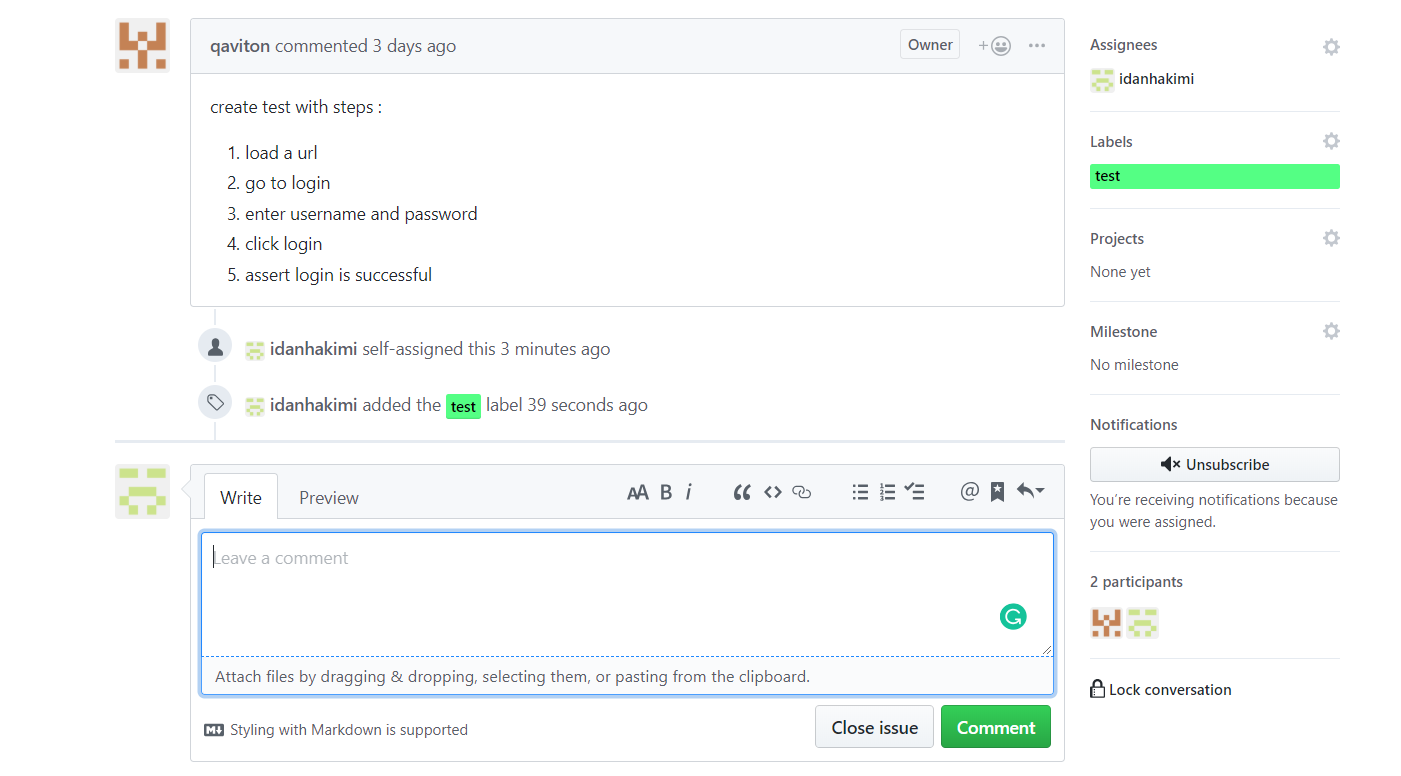
https://github.com/qaviton/qaviton  
  


**חלק ה' – Issues Assignment**

1. כעת נבחר ISSUES (משימות) אותם ניקח מתוך הרשימה הגדולה:  
     
   <https://github.com/qaviton/test_repository/issues>  
     
   

לדוגמה, אני אקח את : "functional web login test".

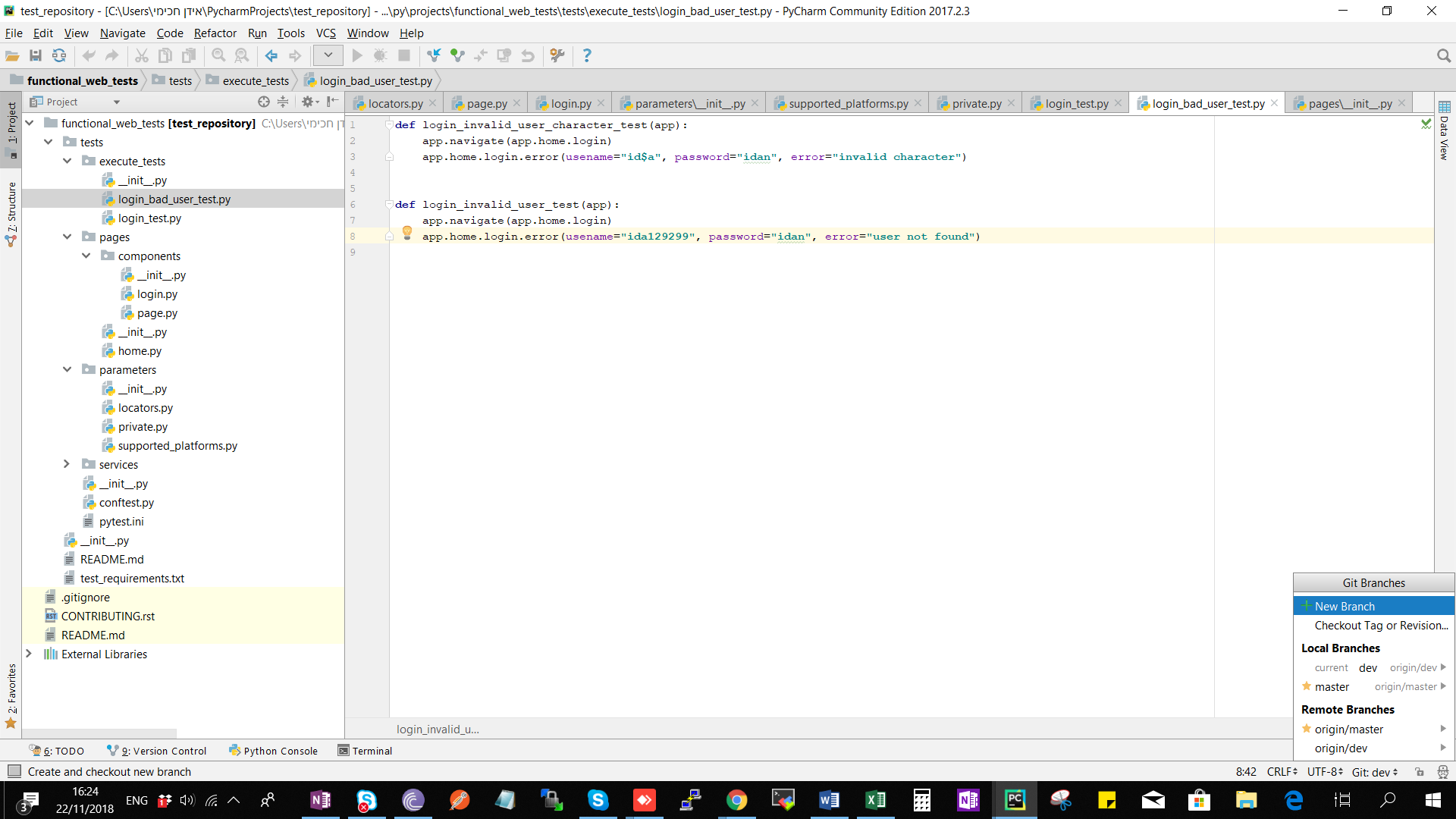
1. כעת משייכים את אותו TEST לאותו אדם שאחראי עליו:  
     
     
     
   ניתן לסווג את ה-TEST לפי תגיות ( לא חובה אבל אם כבר שימו בתגית- "test"):  
     
   

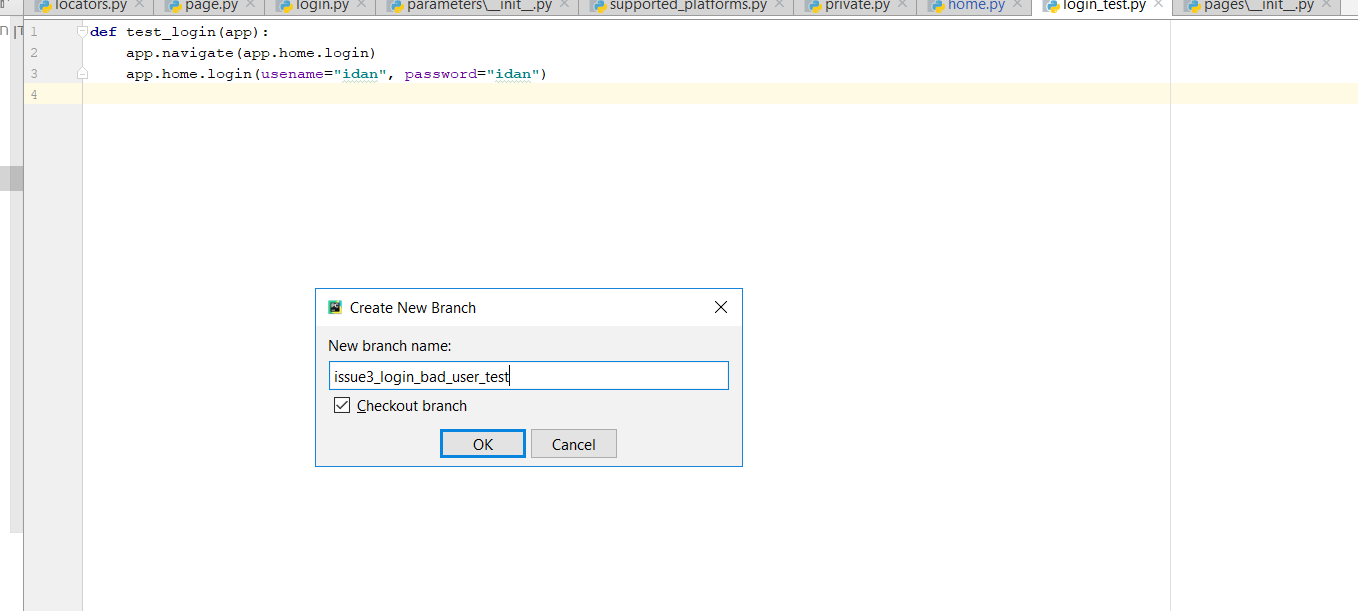
כעת ניתן להוסיף comments לאורך הדרך וכאשר נגמור את אותו TEST אפשר לסגור אותו:  
  


1. לאחר שלקחנו על עצמנו ISSUE ויש לנו את כל ה-CLONE במחשב שלנו, אפשר להתחיל לבנות את הטסט הרלוונטי.

**חלק ה' – Open new branch**

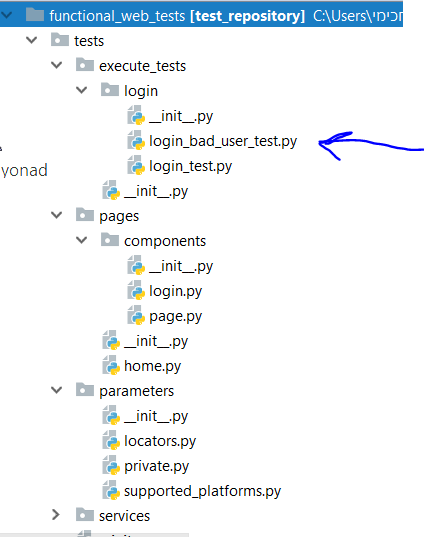
1. בכל פעם שנרצה לקחת משימה, נייצר BRANCH מקומי חדש וזמני עבור המשימה ונכלול בשם שלו את מספר ה-ISSUE ושם ה-ISSUE:



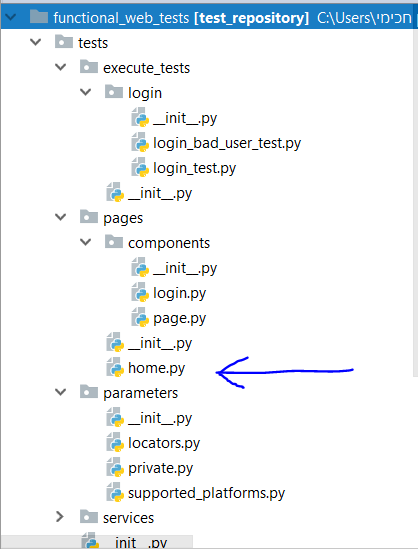


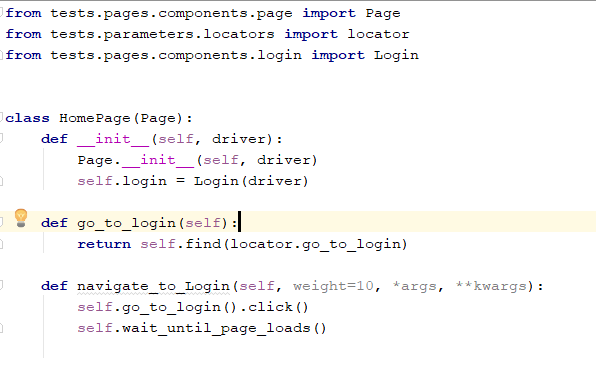
**חלק ו' – בניית טסטים**

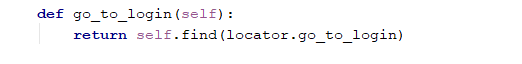
1. אז כך תיראה התיקייה וכל התוכניות של הטסטים שאנו רוצים שתעבדו עליהם כרגע, כמו שניתן לראות ב- Execute\_tests יהיו את כל הטסטים שאנחנו עושים, כל פעם שתיקחו טסט תפתחו שם קובץ מסוג פייתון עם סיומת .PY עם השם התואם את כותרת ה-ISSUE שלקחתם ב-GITHUB עם \_ במקום רווחים.  
     
   \*\*במידה וה-ISSUE מכיל מספר טסטים, הקובץ גם יכיל מספר טסטים.

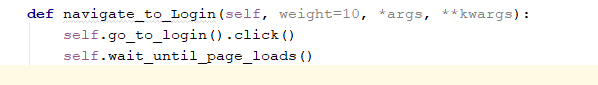
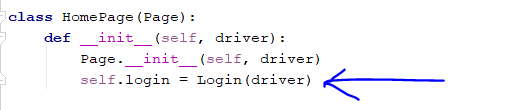


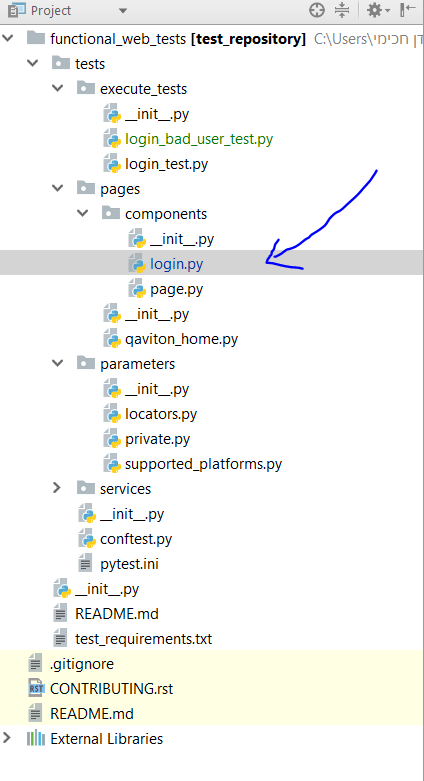
לאחר מכן פה נגדיר את הדפים שאיתם נרצה לעבוד, אנו עובדים עם דף הבית כרגע:



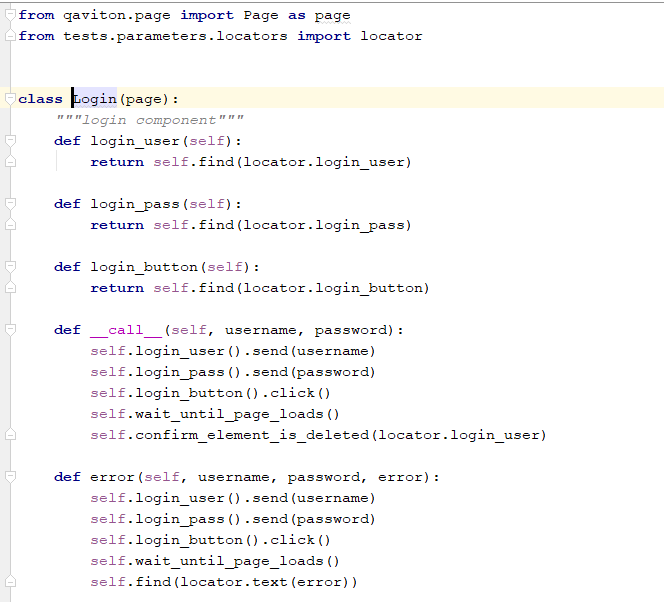


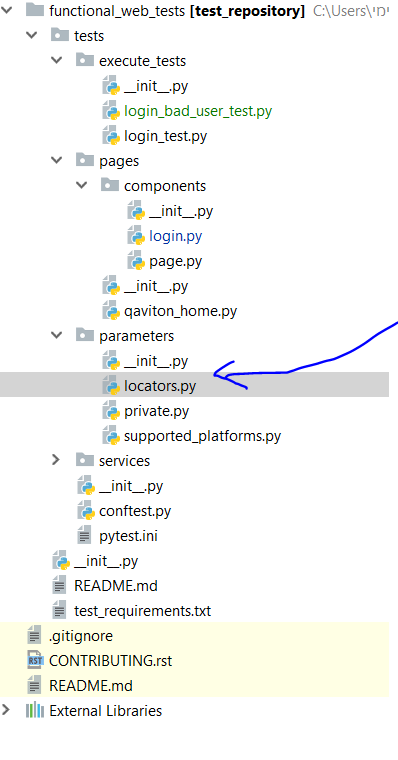
הדף מכיל : Elements, Components, Navigation functions.  
את האלמנטים אנו מגדירים בעזרת locator שמחזיק רשימה של כל האלמנטים באפליקציה.  
על מנת שהדף יחזיר לנו את האלמנט אנו מגדירים פונקציה בדף אשר בעלת שם זהה לשם של הכפתור ומחזירים את האלמנט באמצעות פונקציית FIND.  
  


את ה-NAVIGATIONS נגדיר כאשר נרצה לנווט בין דפים או COMPONENTS.  
  
  
  
את ה-COMPONENTS נגדיר כאשר נרצה לפרק את הדף לחלקים שניתן למחזר בתוך דפים אחרים.  
  


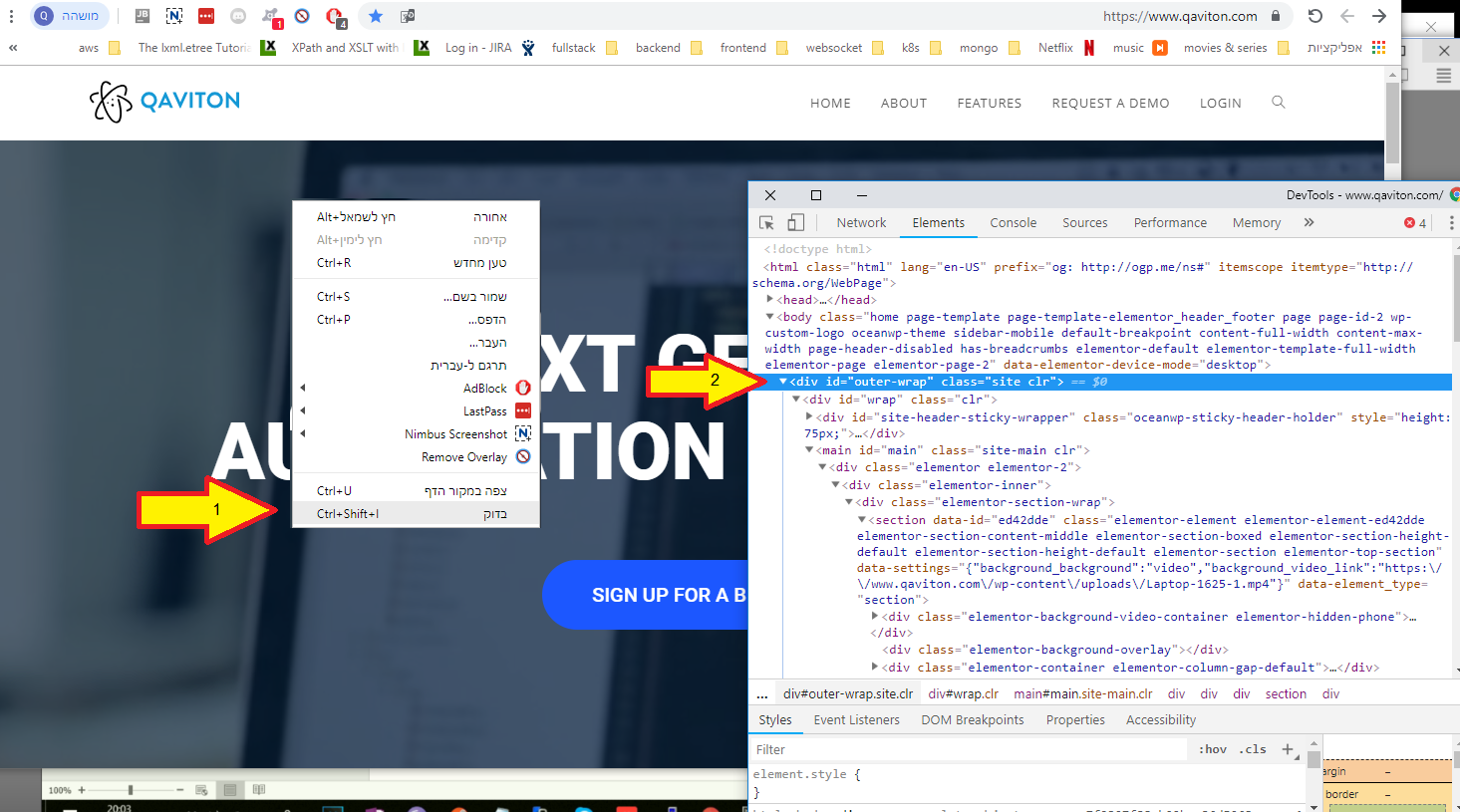
כעת נוכל להגדיר את ה-COMPONENTS:  
  


COMPONENT מתנהג בדיוק כמו דף , יש לו כפתורים , פונקציות ו-NAVIGATION.

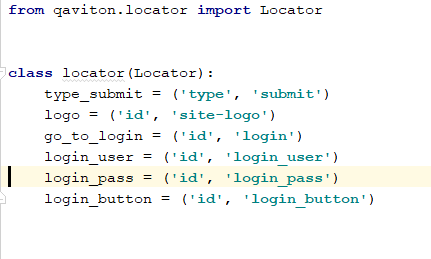


כאן נוכל להגדיר את הכפתורים עצמם שנרצה לבדוק ולמדל :  
  


ניתן למצוא כפתורים של האפליקציה באמצעות כלי הבדיקה של הדפדפן (Inspect/בדיקה):



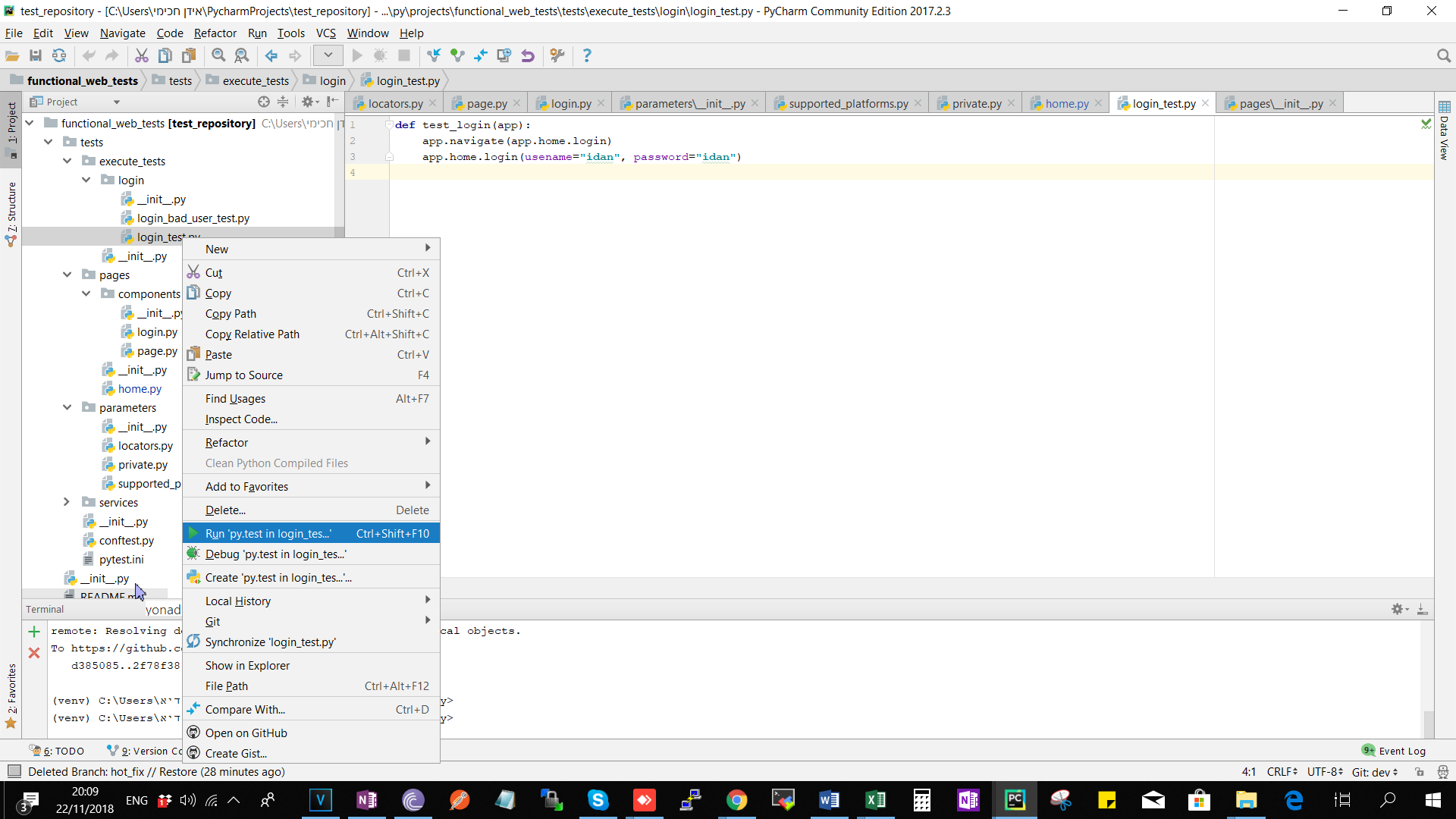
לאחר שמצאנו את הכפתור נגדיר אותו באמצעות תכונות ייעודיות לו, לדוגמה: id , בקובץ ה-LOCATOR:



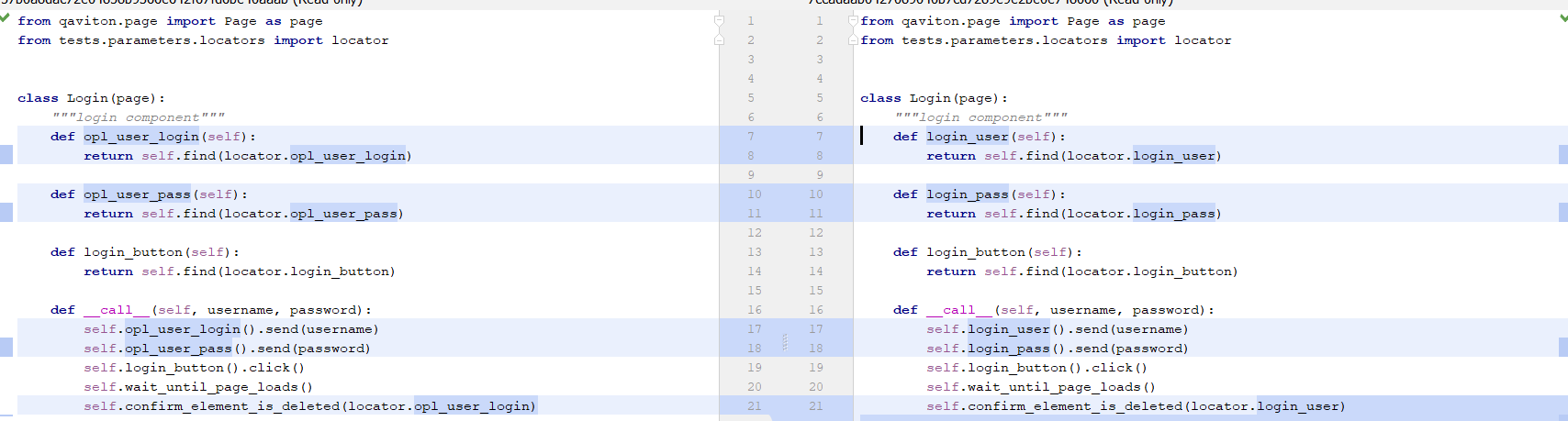
לאחר שבנינו את הדפים, COMPONENTS, כפתורים ופונקציות , נייצר קבצי טסט:



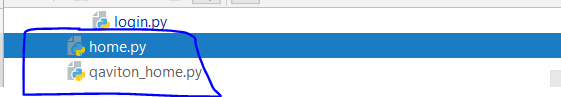
לאחר בניית הטסט נריץ אותו באמצעות מקש ימני ולחיצה על 'RUN':



<https://www.jetbrains.com/help/pycharm/pytest.html>  
  
לאחר שהטסט עבר בהצלחה, נרצה לשנות את שמות וערכי הפרמטרים שיהיו גנריים בשביל משתמשים.

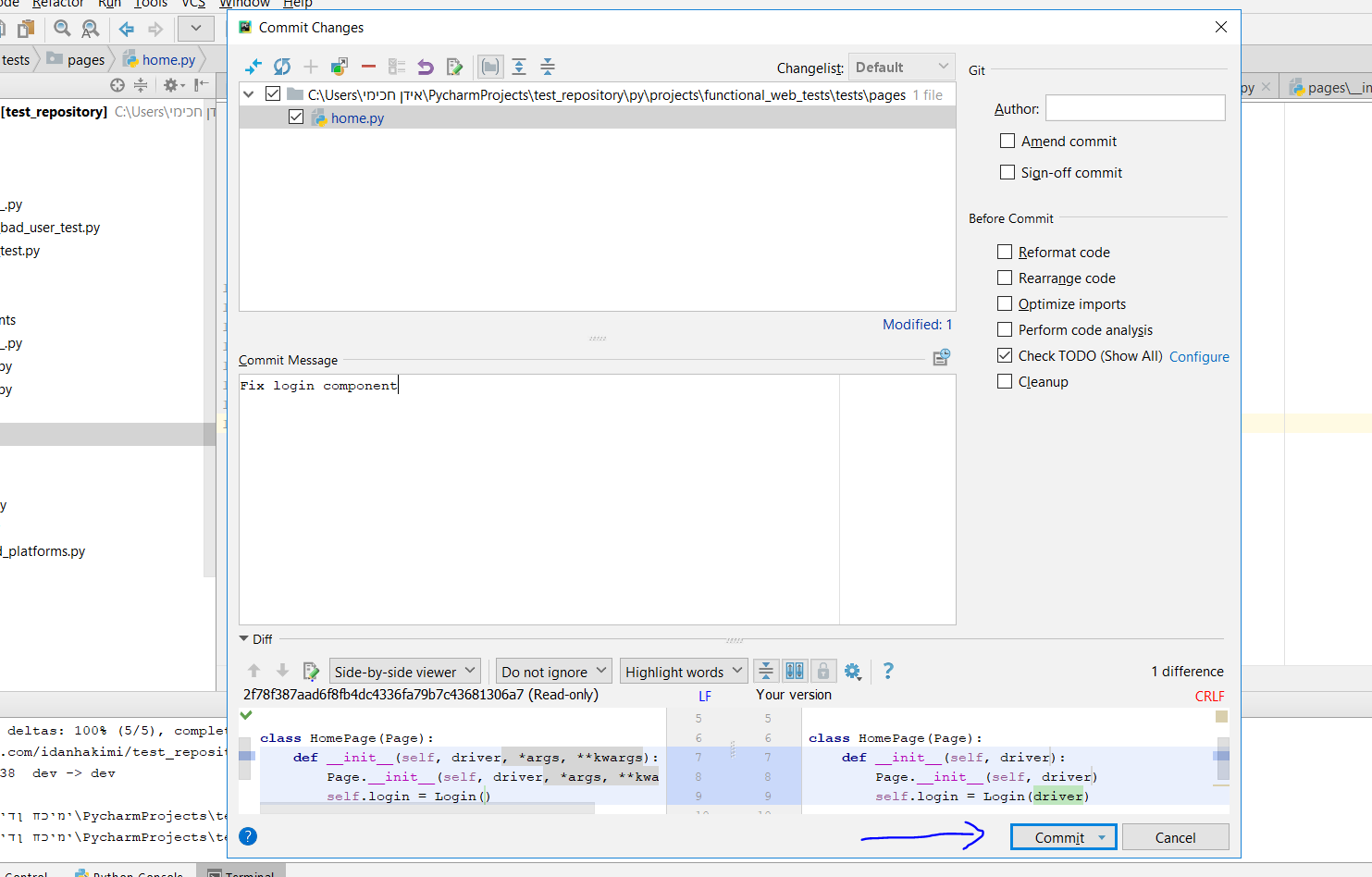






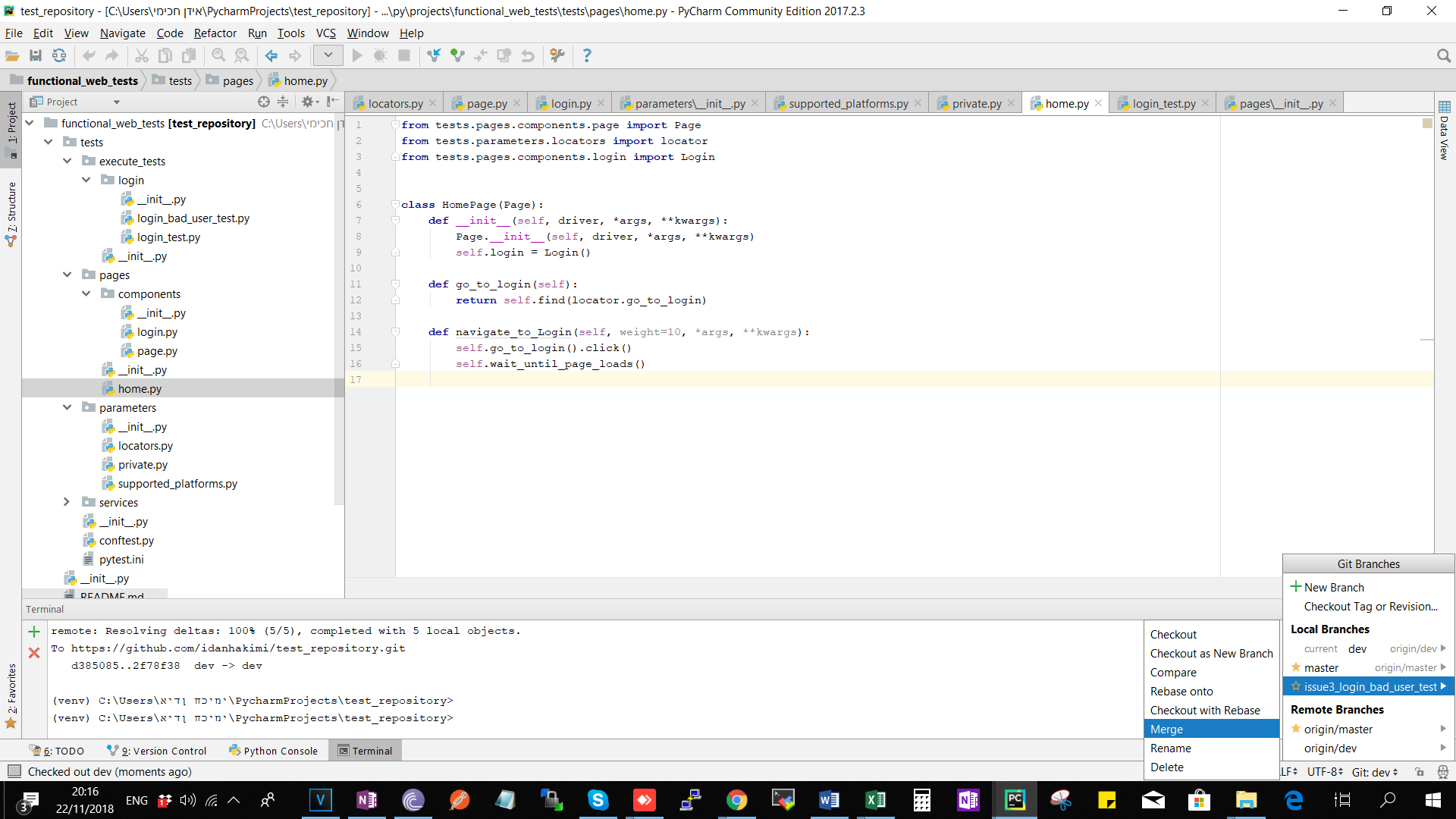
לאחר מכן נדחוף COMMIT, עם הודעה רלוונטית עבור ה-TEST :

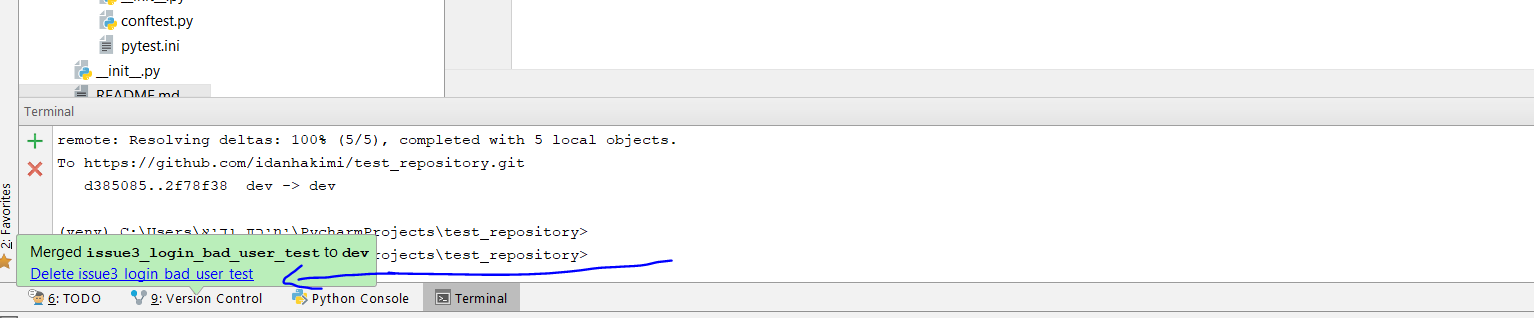


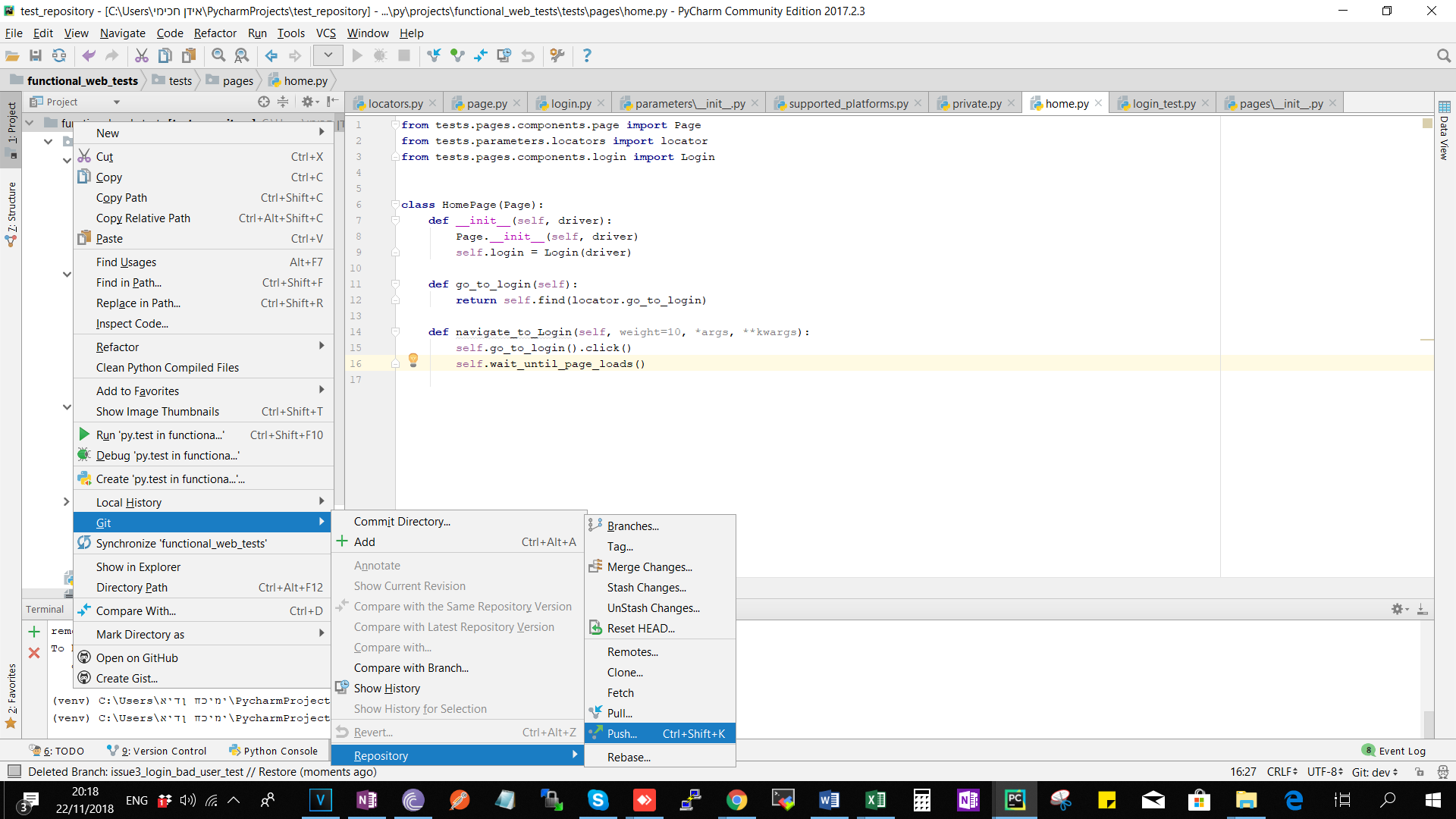


ולבצע MERGE לתוך-BRANCH DEV ולמחוק את ה-BRANCH הזמני.







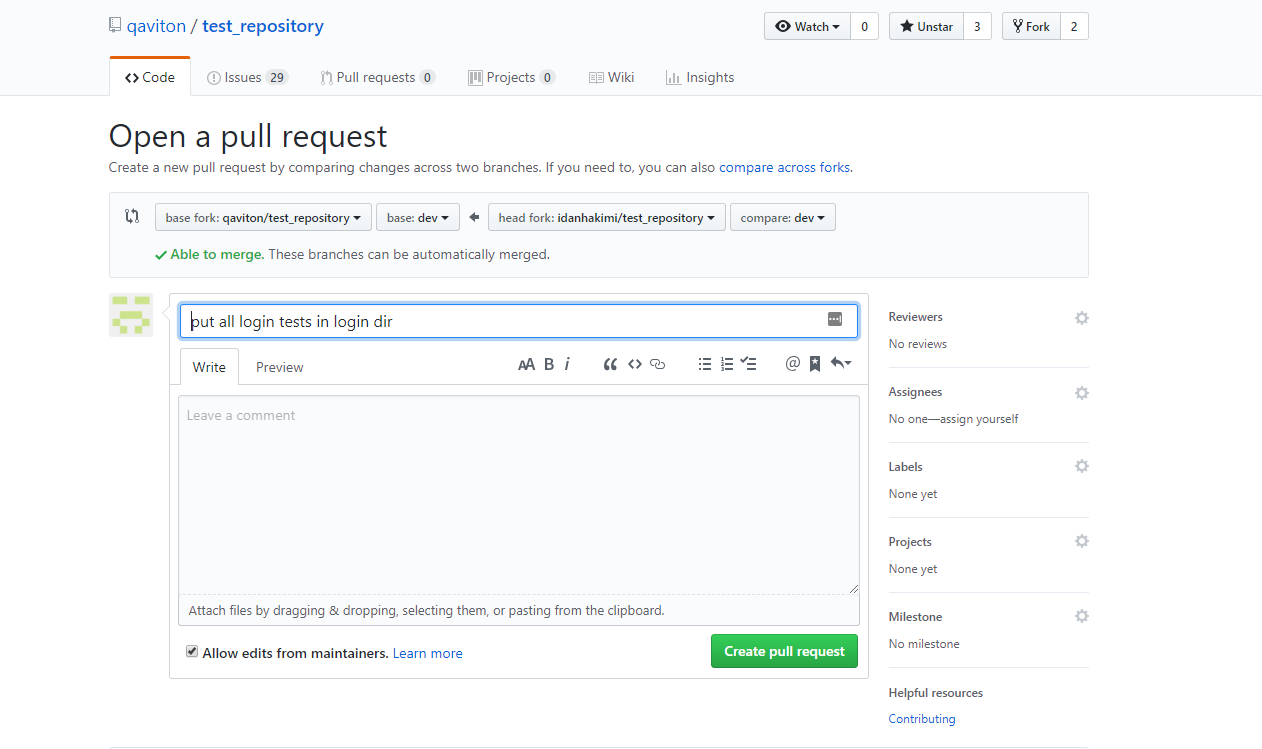


רואים כמה זה פשוט? במקום לכתובת טסט ב-20 שורות, כתבנו אותו רק ב-3 שורות בודדות, אחד ל-NAVIGATION והשנייה הטסט עצמו.

**חלק ז' – Pull Request**

לאחר ה- PUSH אנו עושים PULL REQUEST ב-GITHUB:





ברגע שעשיתם PULL, כעת האישור יתבצע ע"י מנהלי QAVITON OPENSOURCE, להלן האישור של המנהלים- **החלק הנ"ל לא לגעת בו ולא לאשר!**

