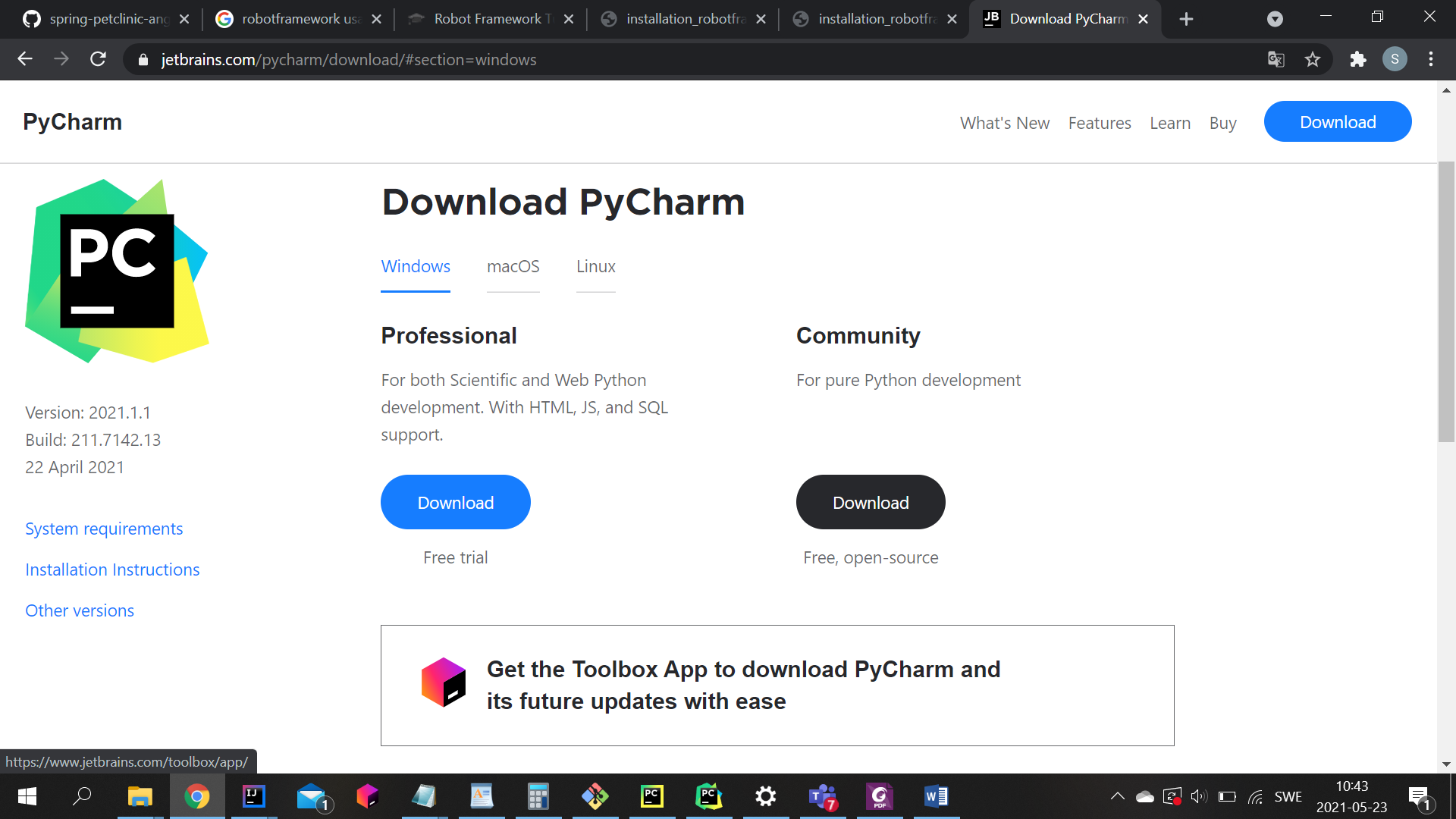
Installation av robotframe work

PyCharm

• Ladda ner installationsfilen här: <https://www.jetbrains.com/pycharm/download/>

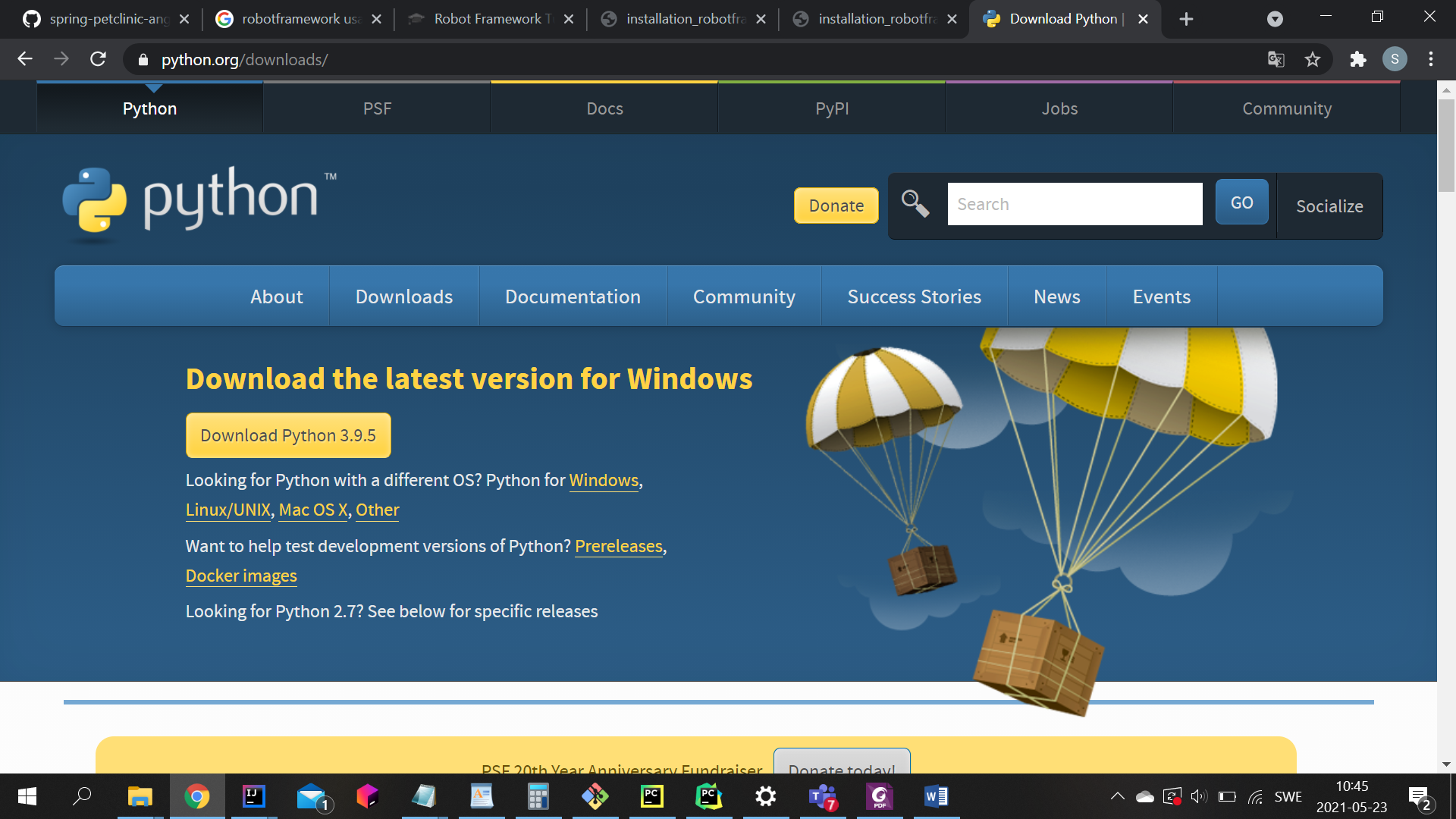


• Kör installeraren och följ stegen.  
• Öppna programmet, klicka på Configure.  
• Klicka på Plugins och skriv in IntelliBot i sökrutan.  
• Välj den plugin som heter IntelliBot @SeleniumLibrary Patched och klicka på Install.  
• Installera även de plugins som heter Robot Framework Support och Run Robot FrameWork TestCase.  
• Starta om PyCharm.

**Python**

• Om du inte redan har Python 3 ska du börja med att skapa en mapp som heter *Python* i din  
 hemmapp: **exempel** C:\Users\user\Python\

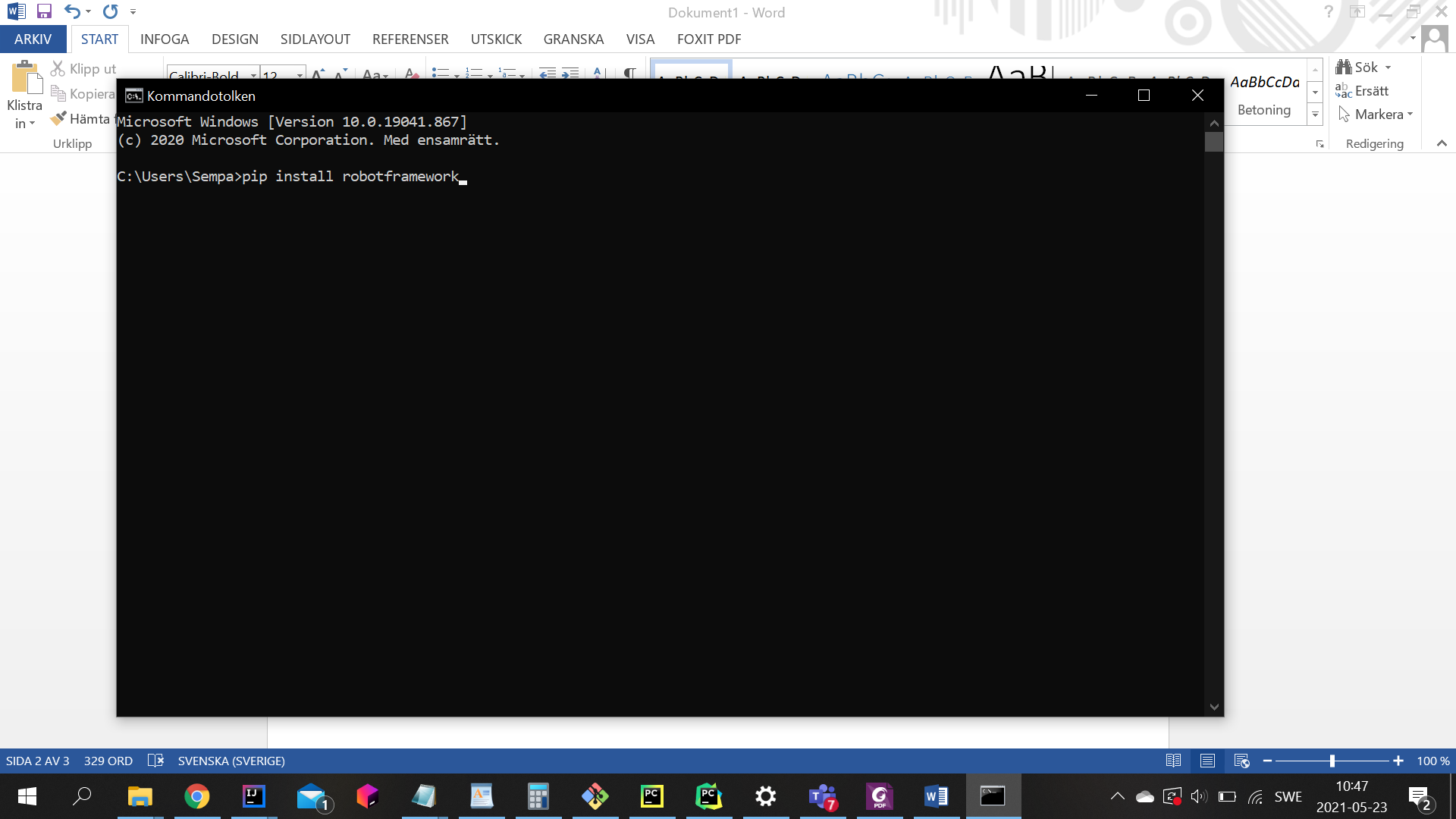
• Ladda ner senaste versionen av Python 3 här: <https://www.python.org/downloads/>



• Kör installeraren och följ stegen. Ange den nyligen skapade *Python*-mappen när du under  
 installationen (*Advanced Options*) får frågan var du vill spara programmet.  
• Lägg till sökvägen till mappen och till undermappen Scripts som miljövariabler i din PATH:  
 **exempel** C:\Users\user\Python\  
 **exempel** C:\Users\user\Python\Scripts\  
• Öppna en kommandotolk och kör följande kommando:  
 python --version  
• Om programmet har installerats korrekt får du ett resultat som liknar detta (beroende på  
 vilken version du installerade):  
 Python 3.8.1

**Robot Framework**

• Ladda ner Robot Framework via kommandotolken med följande kommando:  
 pip install robotframework



• Robot Framework kommer nu installeras på samma plats som Python. Du behöver därför  
 inte lägga till någon miljövariabel eftersom du redan lagt till de sökvägar som behövs när du  
 installerade Python.  
• Kontrollera att programmet har installerats med hjälp av följande kommando:  
 robot --version  
• Om programmet har installerats korrekt får du ett resultat som liknar detta (beroende på  
 vilken version du installerade):  
Robot Framework 3.1.2 (Python 3.8.1 on win32)  
• Installera Seleniumbiblioteket för Robot Framework. **OBS!** Installera INTE biblioteket med  
 namnet Selenium2Library då det har fasats ut. Använd följande kommando i  
 kommandotolken:  
 pip install robotframework-seleniumlibrary

**Webdriver**

• Skapa en mapp som heter *Webdriver* i din hemmapp:  
 **exempel** C:\Users\user\Webdriver\  
• Öppna Chrome och kontrollera din version genom att gå till följande sida:  
 chrome://settings/help  
• Ladda ner webdriver till den Chrome-version du använder på följande webbsida:  
 https://chromedriver.chromium.org/downloads  
• OBS! Du måste först packa upp filen och sedan spara den i *Webdriver*-mappen.  
• Lägg till sökvägen till mappen som miljövariabel i din PATH:  
 **exempel** C:\Users\user\Webdriver\

Komma igång med Robot Framework och Selenium  
  
**Robot Framework**

Ett Python-baserat ramverk för testautomation med öppen källkod. Används för  
acceptanstestning, acceptanstestdriven utveckling (ATDD) och robotstyrd  
processautomation (RPA). Testfallen kan vara nyckelordsdrivna, beteendedrivna och  
datadrivna vilket bidrar till ökad förståelse för vad som testas.

**Bibliotek**

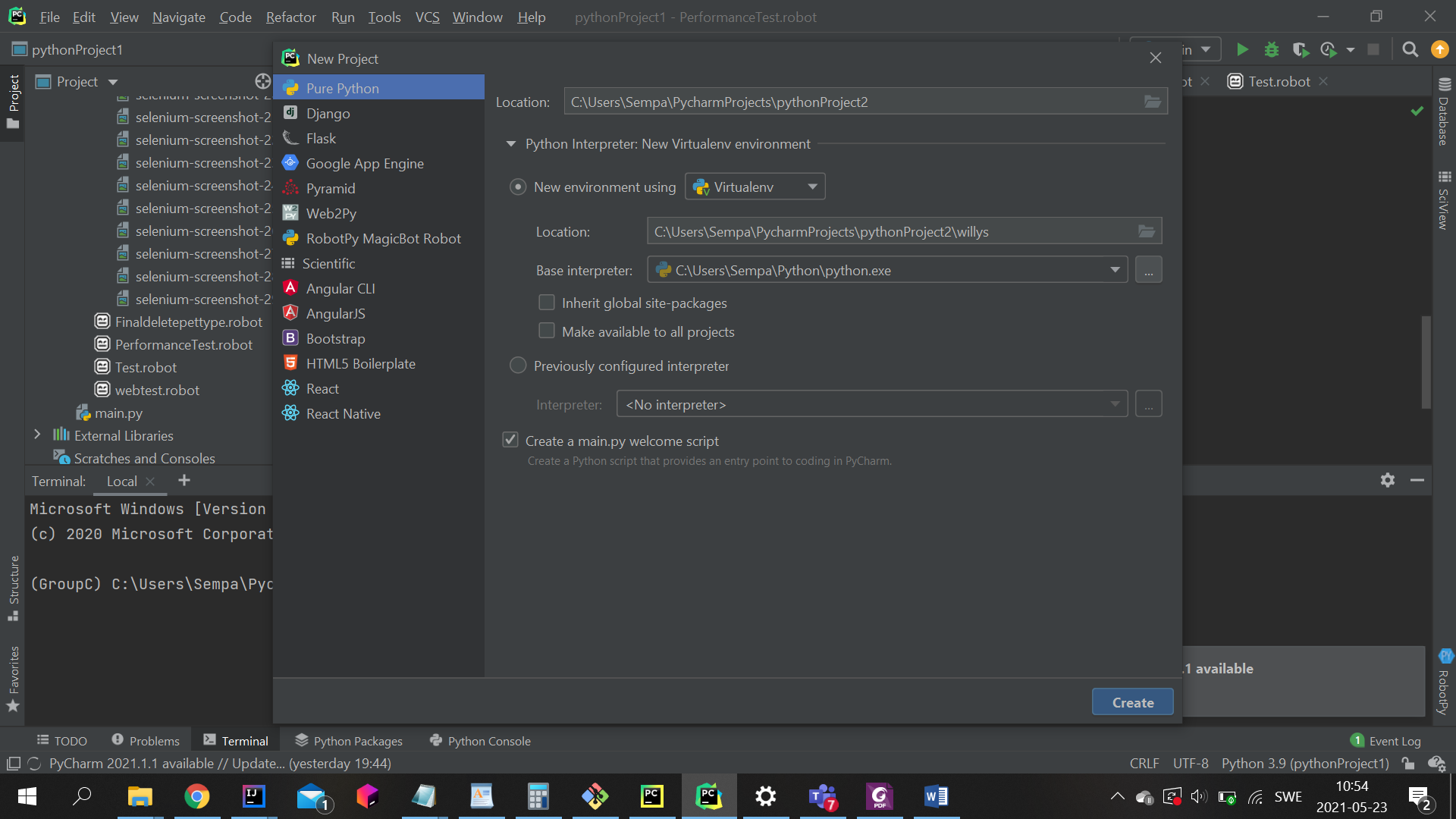
Det finns stöd för att använda externa bibliotek som t.ex. Selenium Library. Det finns också  
möjlighet att koda helt egna bibliotek i Python eller Java om det är något nyckelord man  
saknar.

**Selenium Library**

Ett bibliotek till Robot Framework som används för att testa webbsidor. Det använder  
Selenium Webdriver-moduler för att kontrollera webbläsaren. Biblioteket sköter all  
interaktion med testobjektet med hjälp av olika nyckelord

Skapa projekt

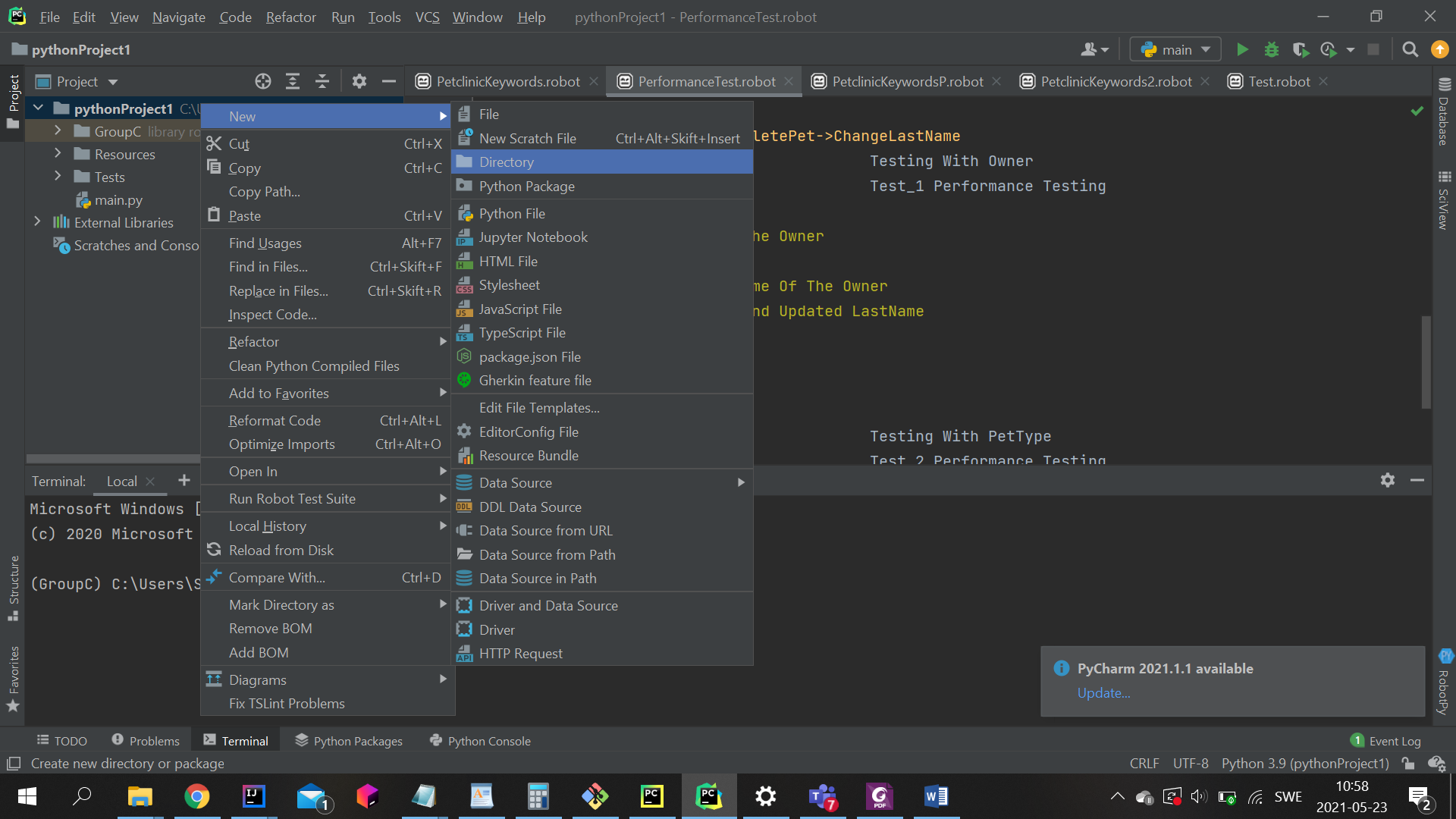
• Öppna PyCharm  
• Klicka på *Create New Project*• Välj location vart projektet ska sparas



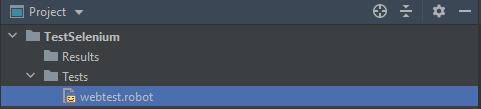
• Döp projektet till ett passande namn

• Klicka på Create.

• Högerklicka på mappen med projektets namn till vänster, klicka på *New* och på  
*Directory*

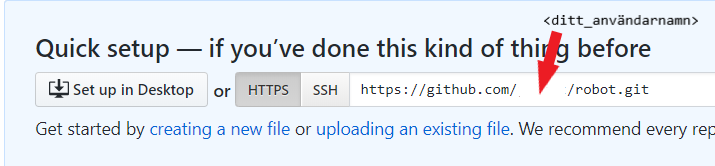


• Döp mappen till *Tests* och tryck på *Enter*• Gör en mapp till och döp den till *Results*• Högerklicka på mappen *Tests*, klicka på *New* och välj sedan *File*• Döp filen till webtest.robot  
• Nu ska mapp- och filstrukturen se ut enligt följande:



Koppla projektet till Github

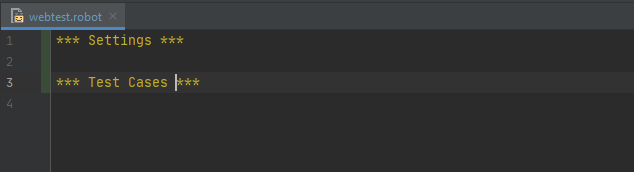
• Skapa ett nytt **publikt** repo på Github som du kallar för *r*• Kopiera adressen till ditt repo, du hittar den på Github:



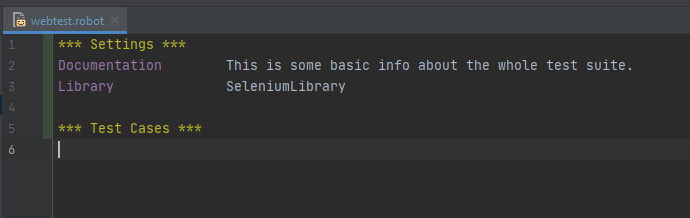
* Du kan nu antingen koppla ditt PyCharm-projekt till ditt Git-repo genom IDE:n likt vi gjort tidigare med IntelliJ.  
  o I VCS-menyn, välj *Import into Version Control*, och välj *Create Git repository*.  
  o Tryck på *Ok*.  
  o I VCS-menyn, välj *Git*, tryck sedan på *Remotes...*o Tryck på *+*, skriv in adressen till ditt repo. Tryck på *Ok.* Tryck på *Ok.*o Högerklicka på *Tests*, välj *Git*, sedan *Add*.  
  o I menyn längst upp till höger finns nu en knapp med en grön bock. Tryck på  
   den för att commita. Skriv in ett lämpligt meddelande i Commit-rutan som  
   dyker upp och tryck på *Commit.*o I samma meny uppe till höger, tryck på den gröna pilen för att pusha.  
  o Se på Github hur ditt repo uppdaterats
* 

Skapa testfall

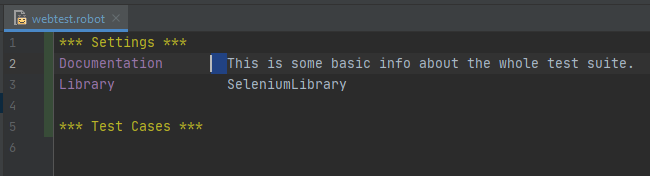
• I filen används en särskild syntax och data delas in i olika sektioner. Börja med att  
 ange rubrikerna *Settings* och *Test Cases* enligt följande:



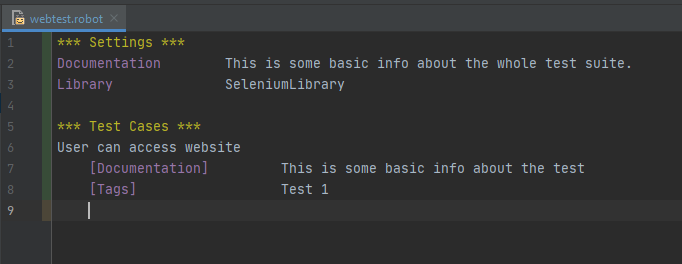
• Under rubriken *Settings* anges vilket bibliotek som ska importeras och man kan även  
 lägga till information om testsviten under Documentation, se nedan:



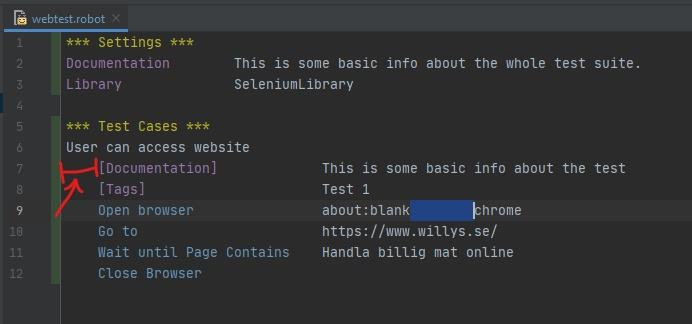
• Viktigt att notera är att det behövs **minst** två mellanslag mellan olika poster då dessa  
 fungerar som separator. Endast ett mellanslag leder till att programmet inte kan  
 tolka det som står i filen

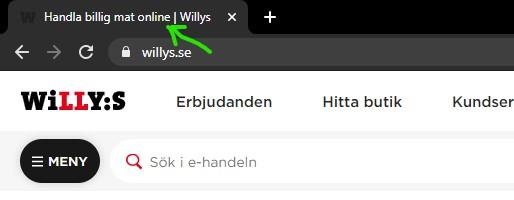


* Lägg till Documentation och Tags för att enklare kunna skriva förklaring till testet samt kunna hitta det enklare

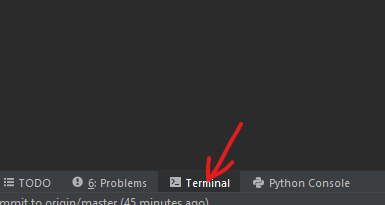


• I våra testfall ska vi använda nyckelord från SeleniumLibrary-biblioteket för att  
 instruera Webdrivern hur testerna ska utföras. Nedan ses hela testfallet för att  
 användaren ska navigera till webbsidan willys.se och för att det också ska verifieras  
 att sidan har laddats. Detta gör man genom att vänta på att texten *Handla billig mat  
 online* blir synlig.  
• Skriv in texten nedan i PyCharm och notera att det behövs minst två mellanslag  
 mellan *about:blank* och *chrome* i nyckelordet *Open Browser*. Lägg också märke till  
 indenteringen av *[Documentation]*, *[Tags]* och alla nyckelord som ligger under  
 testfallets namn (se den röda pilen):

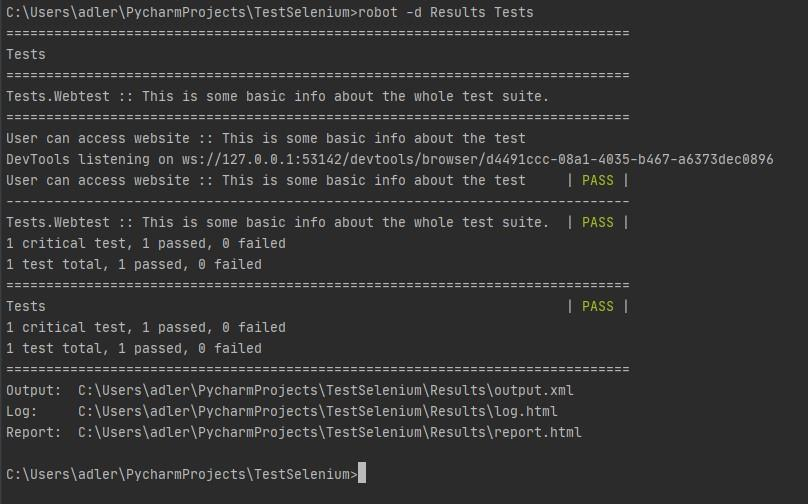




• Det är nu möjligt att köra testet. För att göra det öppnar man terminalen genom att  
 klicka på *Terminal* längst ner till vänster i PyCharm:



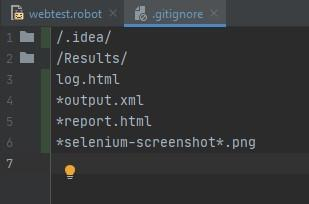
• Använd följande kommando för att köra testet:  
 robot -d Results Tests (Namnet på filen).robot

• I terminalen ska det nu, förutsatt att testet går igenom, se ut på följande sätt:  


• Resultat från körningen ska nu finnas i mappen *Results.*

Skapa -gitignore-fil

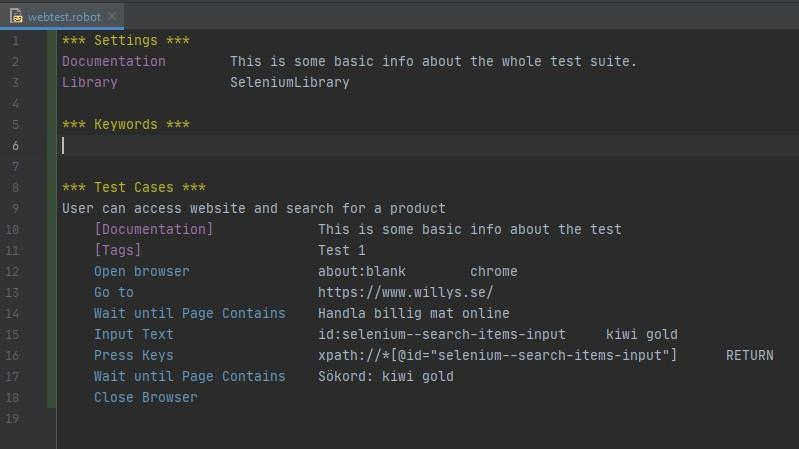
• Högerklicka på Projektmappen, klicka på *New* och välj sedan *File*• Döp filen till *.gitignore* (missa inte punkten)  
• Klicka *Add* när Pycharm frågar om du vill lägga till filen till Git  
• Lägg till följande i *.gitignore*-filen:



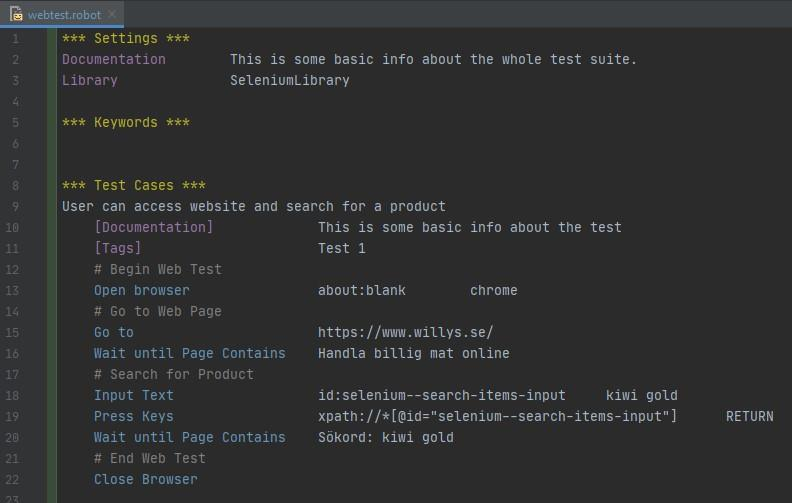
• Detta gör att git inte bryr sig om dessa filer och mappar, så att vi inte råkar committa  
 dem  
• Gör nu en commit av .*gitignore*-filen

Utöka testfallet med nyckelord

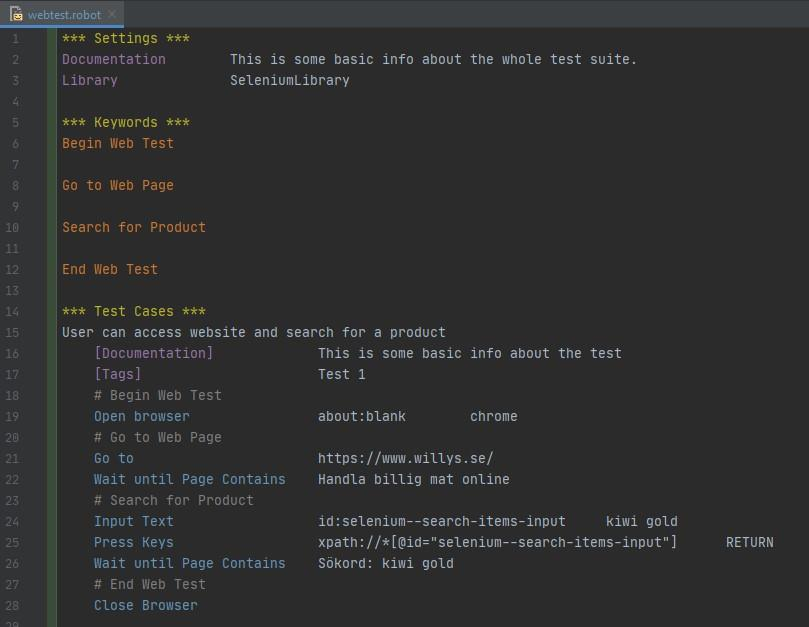
• Det befintliga testet ska nu göras ännu tydligare genom att vi skapar egna nyckelord.  
 Till att börja med ska sektionen *Keywords* läggas till:



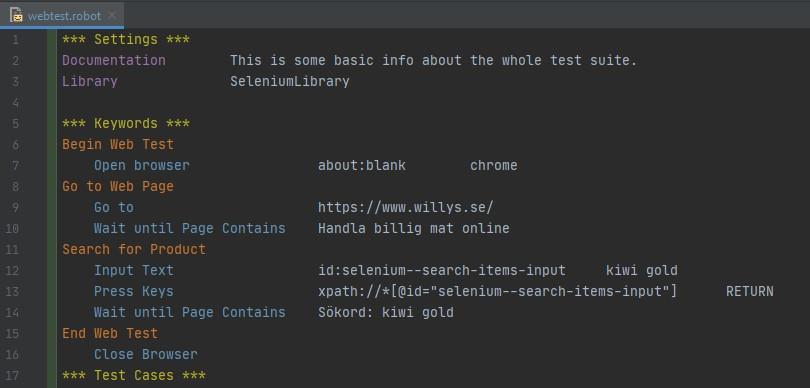
• Därefter ska testet delas in i olika delar beroende på vad som utförs. För att det ska  
 bli enklare kan man börja med att skriva kommentarer:

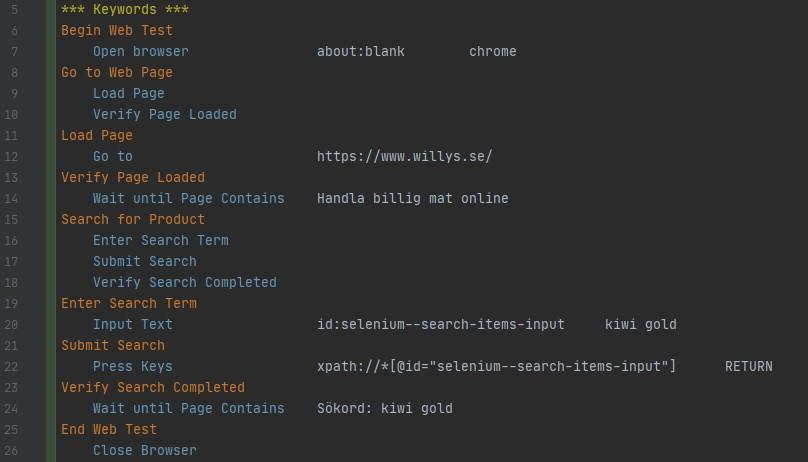


• Kopiera sedan kommentarerna (OBS! Ta inte med brädgårdstecknet) och klistra in  
 dem under *Keywords*:



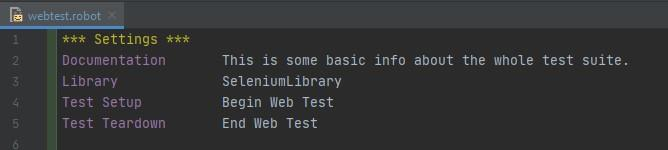
• Klipp sedan ut allt som ligger under kommentarerna i testfallet och klistra in under  
respektive rubrik i *Keywords*:

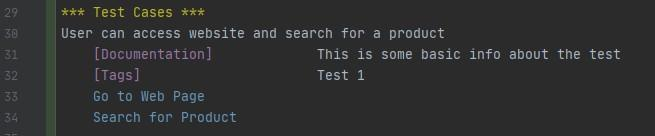


• Ta bort alla brädgårdsstecken framför rubrikerna i testfallet enligt nedan:  
• Det är möjligt att bryta ner nyckelorden ännu mer på följande sätt:  


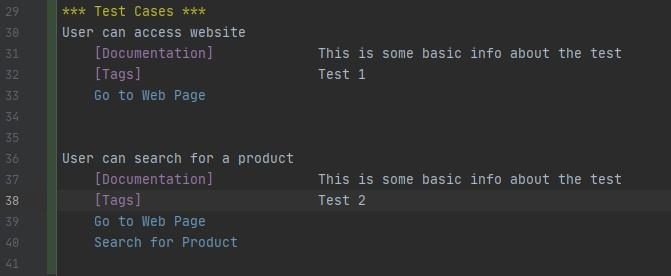
• På detta sättet kan vi använda våra Keywords på mängder av olika sätt, där Keywords kan  
 kalla på andra Keywords, t.ex. som här kallar *Go to Web Page* på keywords *Load Page* och  
 *Verify Page Loaded*. Se det som nested Keywords.

Lägg till Test Setup och Test Teardown  
• Nästa steg är att lägga till en *Test Setup* och *Test Teardown*. Med hjälp av dessa  
 beskriver man vad som ska hända före och efter varje test. I det här fallet är det  
 lämpligt att *Test Setup* öppnar vald webbläsare och att *Test Teardown* stänger  
 webbläsaren. Dessa steg flyttas därför från själva testfallet upp till sektionen *Settings* enligt följande bild:



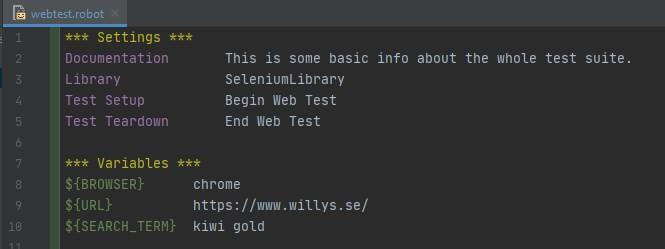
• Testet ser nu ut så här men fungerar fortfarande på samma sätt som tidigare:  


• För att se hur det fungerar med flera testfall kan det första testfallet göras om till  
 två:

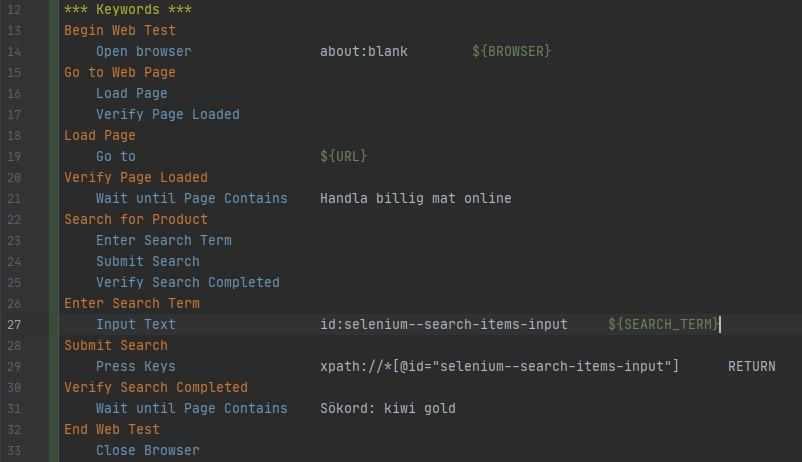


Lägg till variabler

• Nu ska sektionen *Variables* läggas till. Med hjälp av variabler kan man ersätta  
 hårdkodade värden vilket bland annat gör det lättare att återanvända t.ex. nyckelord  
 utan att behöva modifiera dem. Istället ger man helt enkelt variabeln ett nytt värde.  
• Det finns olika typer av variabler. De variabler som ska läggas till nu är globala, d.v.s.  
 alla nyckelord och testfall kan komma åt dem. Lägg till följande:



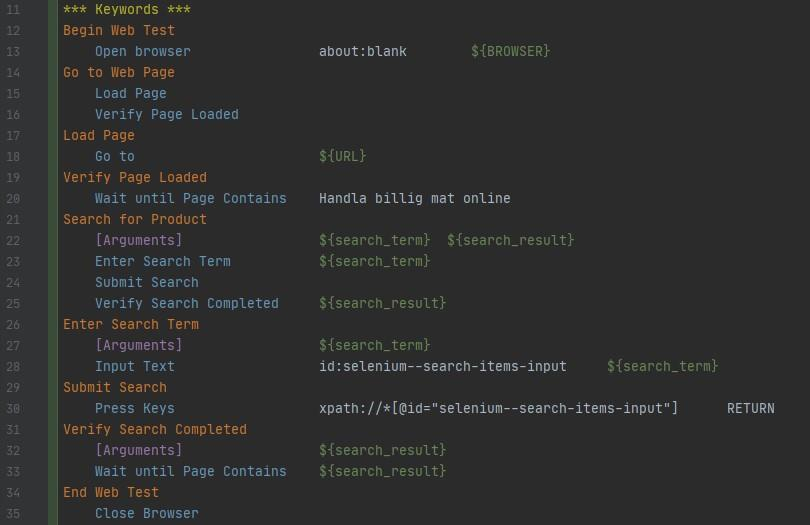
• Tänk på att det måste vara minst två mellanslag mellan variabelnamnet och dess  
 värde för att det ska fungera.  
• Ersätt nu värdena med variabler i de nyckelord där det är möjligt:



• Skulle man nu vilja byta till en annan webbläsare ändrar man helt enkelt variabelns  
 värde i sektionen *Variables*. Överallt där *${BROWSER}* används ändras följaktligen  
 värdet. Mer information om variabler finns i dokumentationen.

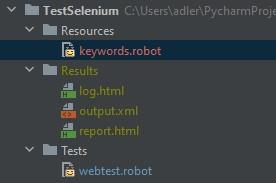
• Det är även möjligt att använda lokala variabler, d.v.s. variabler som endast är  
 åtkomliga i det nyckelord där de deklarerats. Notera att man i Robot Framework gör  
 skillnad mellan versaler och gemener i variabelnamnen och att det senare indikerar  
 att det är lokala variabler.

• Om man exempelvis skulle vilja kunna söka efter olika produkter i olika testfall och  
 återanvända *Search for Product*-nyckelordet är det möjligt genom att göra på  
 följande sätt:

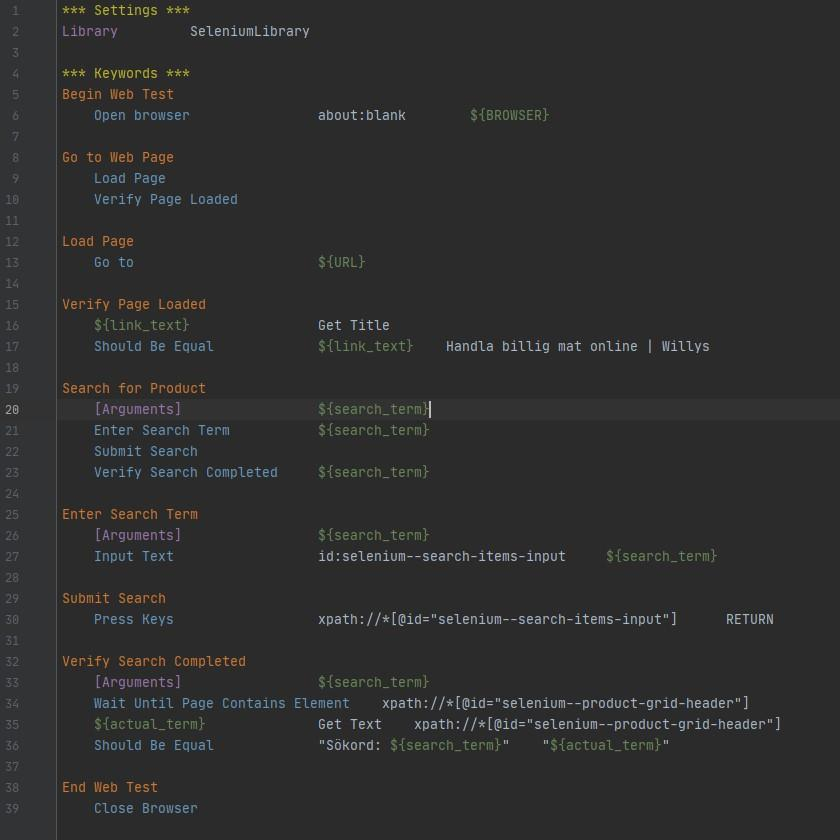


Skapa fil för nyckelord

• Slutligen kan projektet göras ännu lättare att överblicka genom att separera  
 nyckelorden från testfallen och lägga dem i en egen fil. Skapa en ny mapp som heter  
 *Resources* och skapa en fil i mappen med namnet *keywords.robot*:



• Klipp ut hela sektionen *Keywords* och lägg in i den nya filen:



• I filen *webtest.robot* behöver man nu hänvisa till *keywords.robot* för att nyckelorden  
 ska hittas vilket görs genom att skriva in sökvägen till filen i *Settings*. Vill man dela  
 upp nyckelorden ytterligare går det utmärkt att lägga dem i flera filer så länge som  
 man hänvisar till alla från filen med testfallen.

