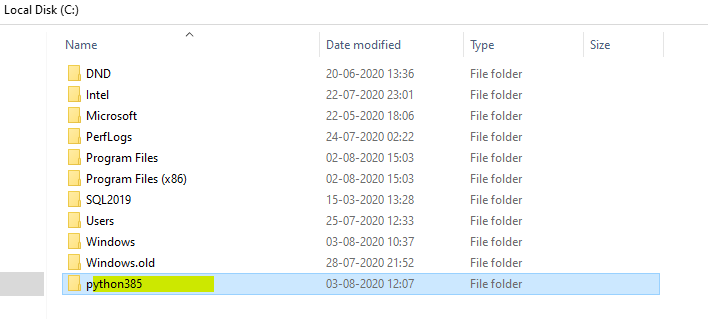
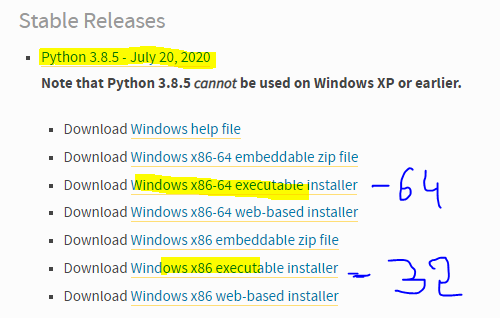
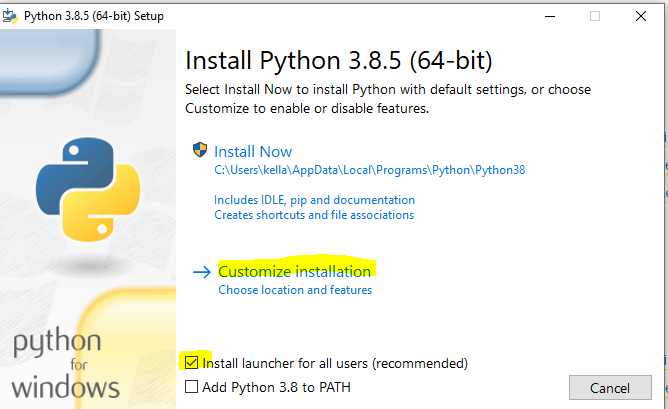
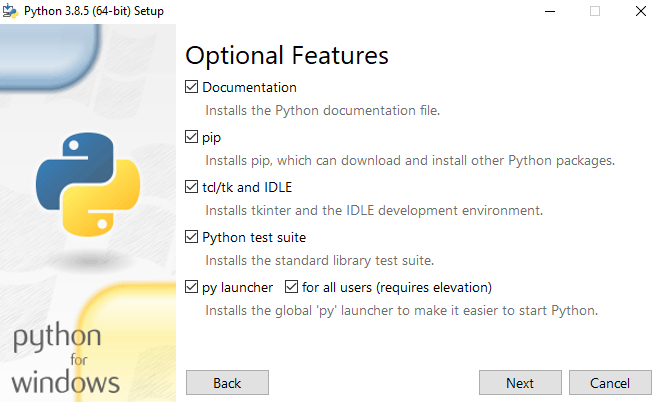
Python Installation:

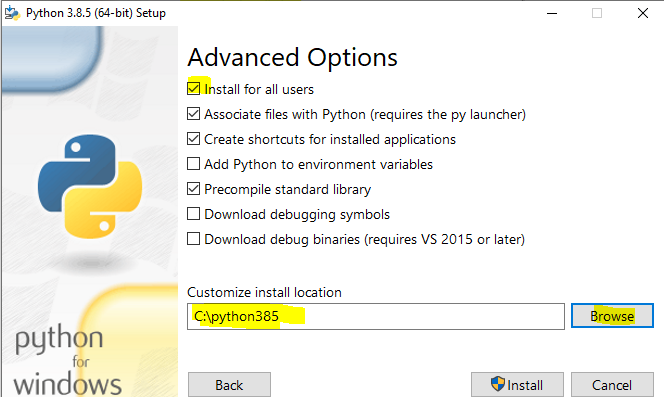
1. Custom Folder we have to create to install python else it will install in program files in C drive. Created Folder in C Drive as **python385**
2. Download the python exe from the official web site

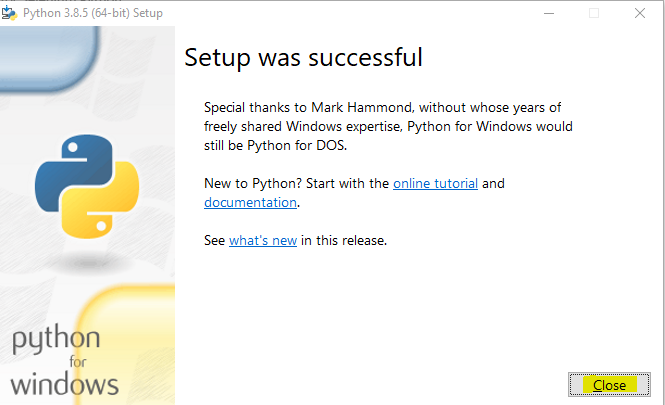
<https://www.python.org/downloads/windows/>



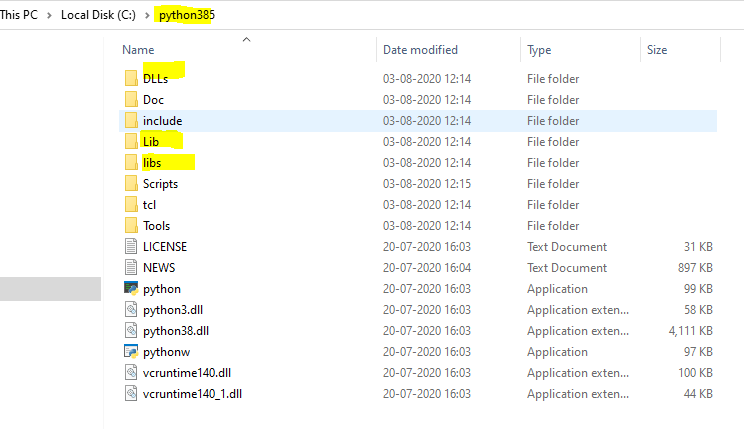
1. Download executable file and start installation as below



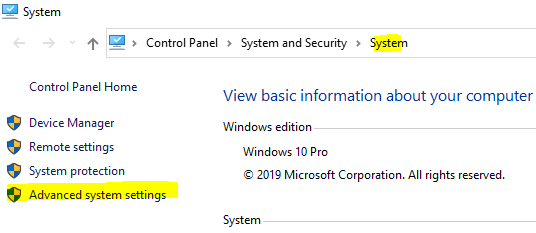




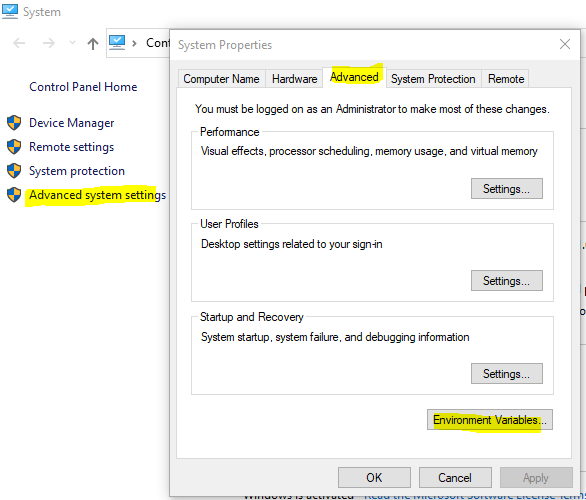
1. Set the path variable to tell the system, where python exist on the system.  
   add the highlighted folders to python path



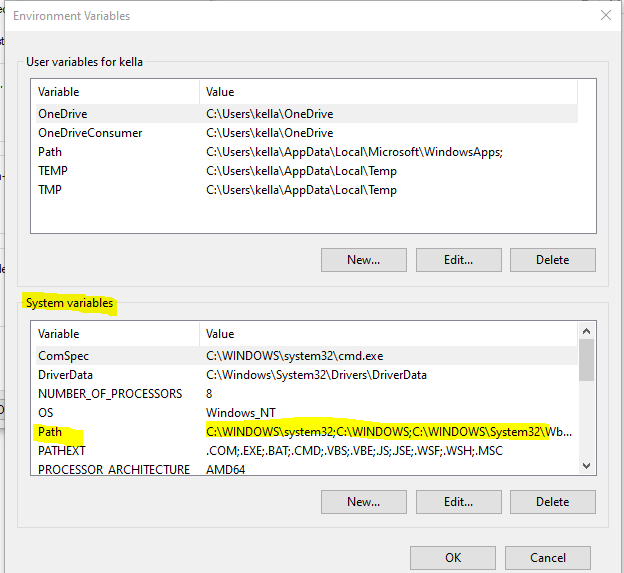
1. Right Click on This Pc ---> Click On Properties ---> Click On Advanced system settings



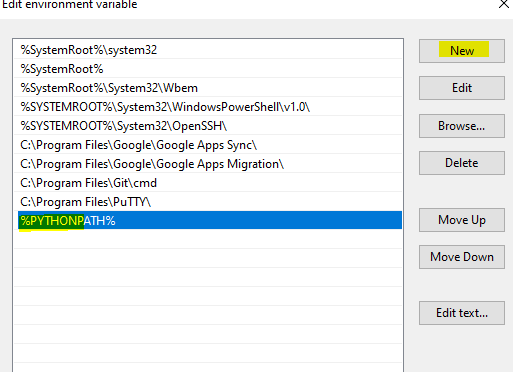
1. Click on environment variables



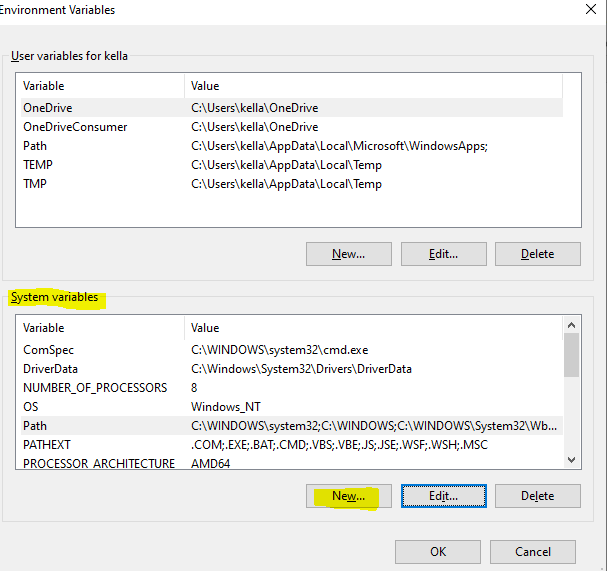
1. Under system variables , Select **Path** click on EDIT button



1. Add a variable called PYTHONPATH, by clicking the NEW and click OK

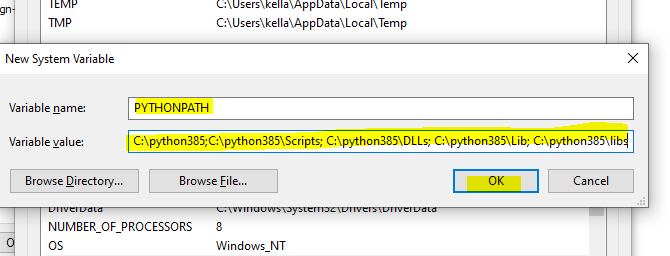


1. Now add the python directories to it, by clicking NEW button



1. Add the following path to variables  
   variableName: PYTHONPATH

Varaible value: C:\python385;C:\python385\Scripts; C:\python385\DLLs; C:\python385\Lib; C:\python385\libs



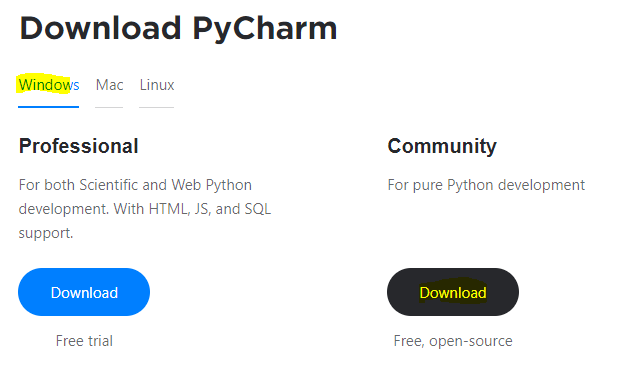
Click ok on all the open windows

1. Restart the system to reflect the changes
2. Check python properly installed or not by using command prompt

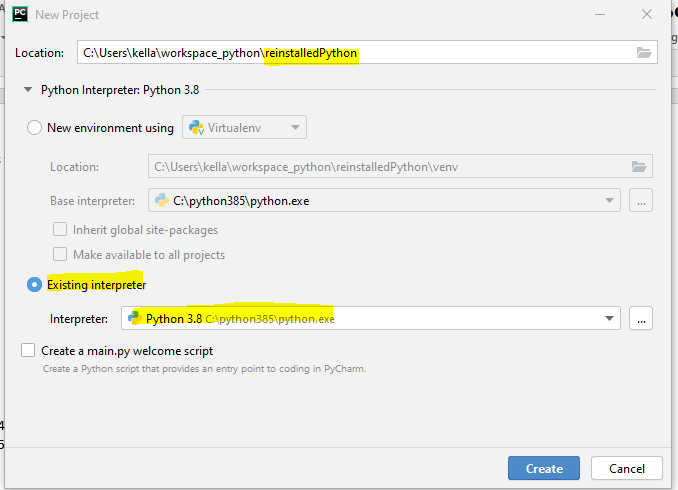


1. Install the Pycharm

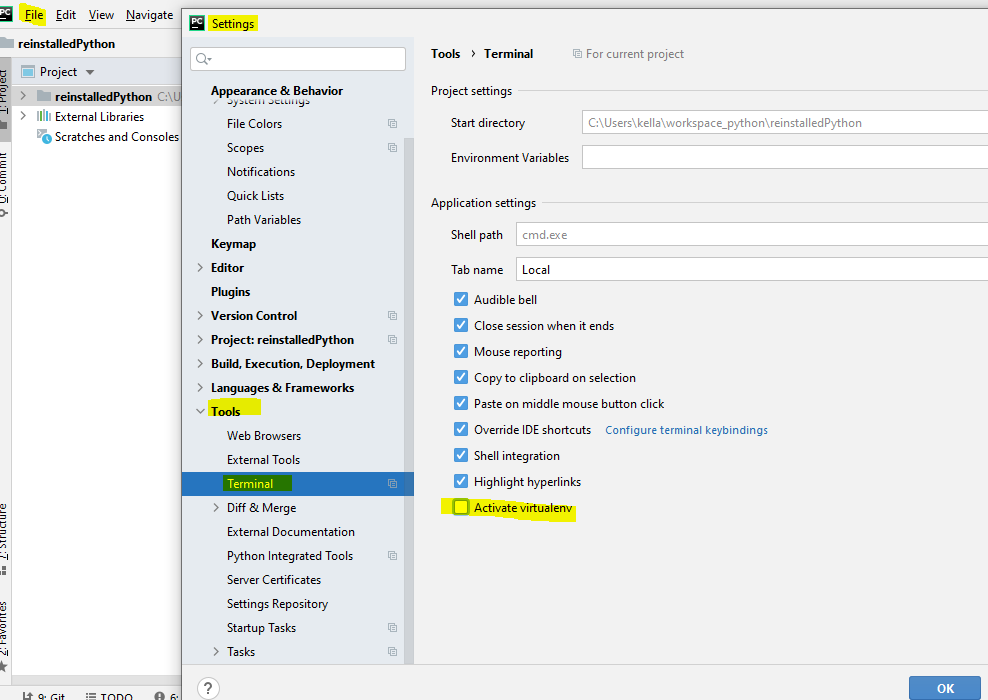
<https://www.jetbrains.com/pycharm/download/#section=windows>



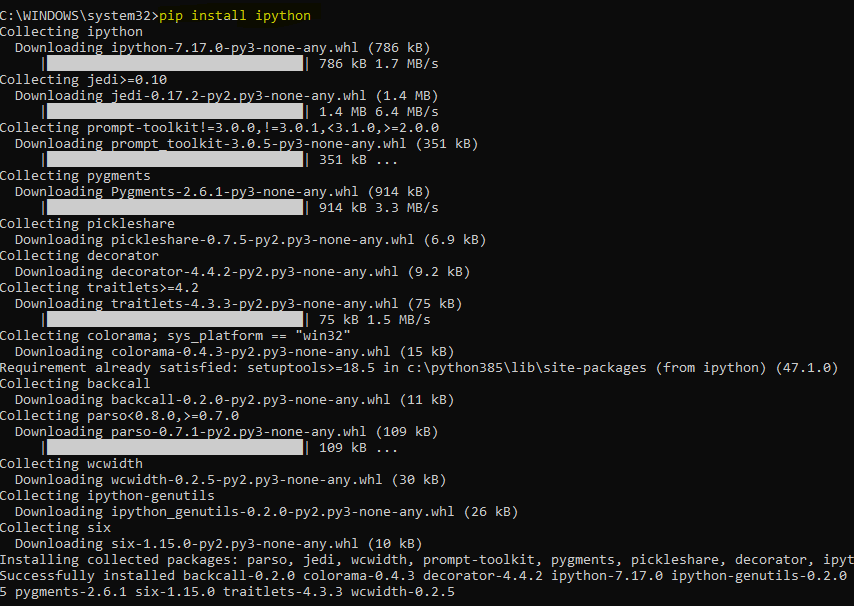
1. Install by clicking PyCharm exe file and click on okay for all the default settings
2. Create a new project and set the existing python interpreter



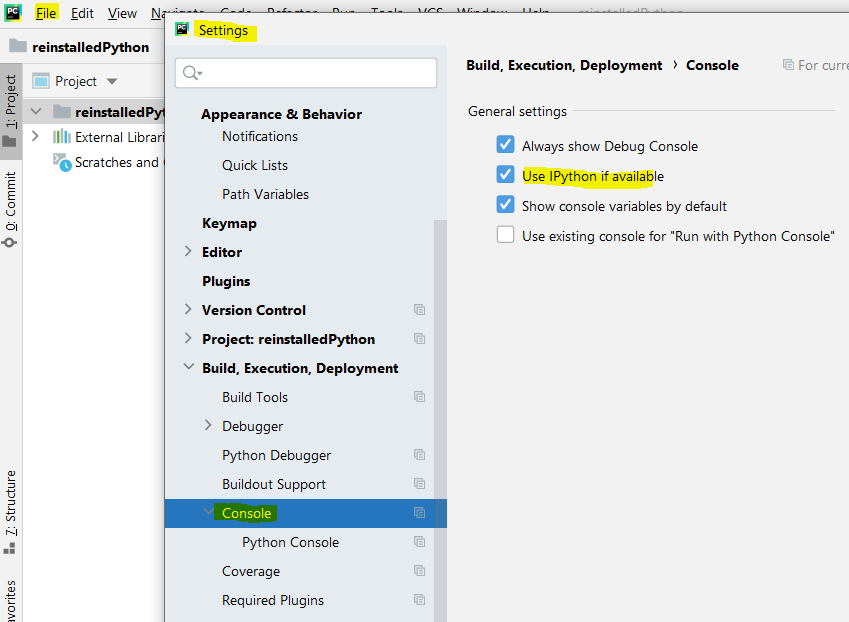
1. Disable the virtual environment for all the project File🡪settings🡪tools🡪Terminal🡪uncheck Active Virtual Env, click on Apply button



1. Install the wheel with help of pip
2. Install ipython with help of pip, run command prompt as Admin and execute “pip install ipython”

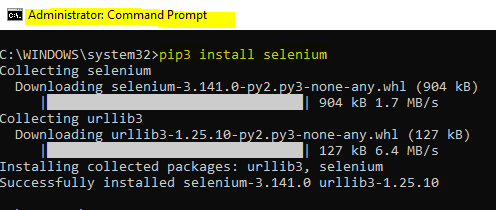


1. Use ipython in pycharm



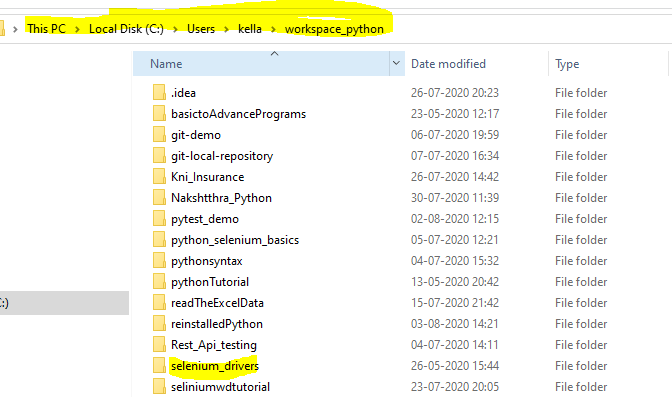
1. Install the selenium web driver

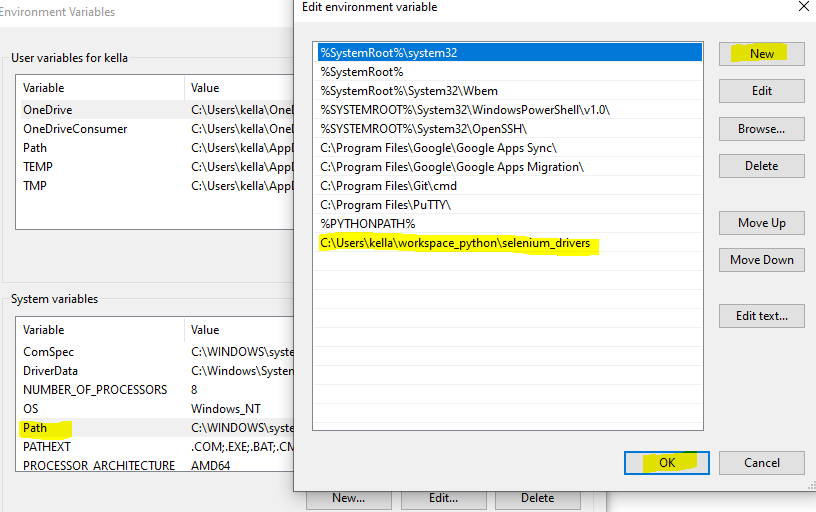
Open command prompt as admin



1. Set the drivers in system environment variables

Download all the drivers and put it under one folder, folder should be in the project location

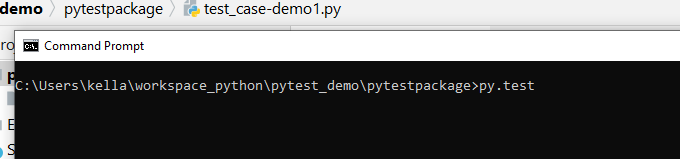


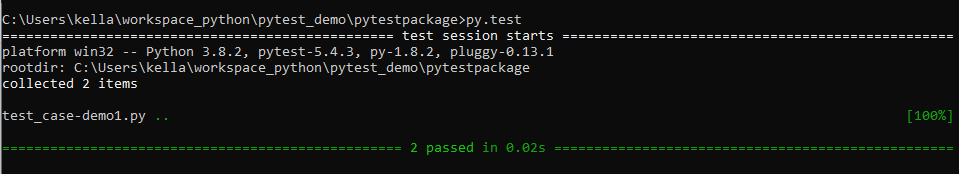
1. thisPC-->Properties-->Advanced System Settings 🡪click On Environment variables  
   select “**PATH**” click on edit , then add the drivers path.

PyTest:

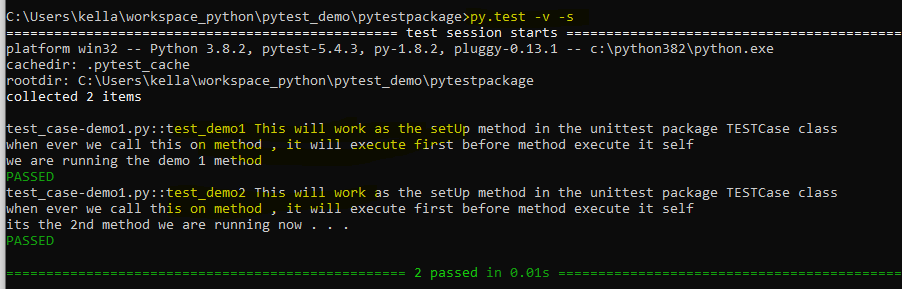
Inorder to run or execute the python file from the command prompt or terminal of pycharm

* We have to go to the project location
* Go to the directory/package folder



* Run the command “py.test” – this will collect all the methods and execute them , it will print only when there is a failed cases else it will return the how many methods executed successfully
* 
* If you want to see the messages, then we have to execute that in the following way

Py.test -v -s here -v means verbosity , -s means print the statements



* **File names should start or end with “test”**, as in test\_example.py or example\_test.py
* **Class name should start with “Test”**, as in TestExample
* **Test method names should start with “test\_”**, as in test\_example
* **Pytest Fixtures:** initialize test functions. They provide a fixed baseline so that tests execute reliably and produce consistent, repeatable, results.
* fixtures have explicit names and are activated by declaring their use from test functions, modules, classes or whole projects.
* fixtures are implemented in a modular manner, as each fixture name triggers a *fixture function* which can itself use other fixtures.

Ex:

@pytest.fixture()  
def fixturemethod():  
 print(**'This will work as the setUp method in the unittest package TESTCase class'**)  
 print(**'when ever we call this method , it will execute first before method execute it self'**)  
  
def test\_demo1(fixturemethod):  
 print(**'we are running the demo 1 method'**)

* **Pytest yield Fixture:**

with help of the yield key word , we can differentiate the what to execute the first i.e. before every test method call and what to execute after the method.

In the unittest package we have used the setUp method and teardown methods to call before and after the methods gets called.

Let us see the example , how we can achieve the same in one function in pyTEST

import pytest  
  
@pytest.yield\_fixture()  
def yield\_fixtures():  
 print(**'this will execute before evry method Gets called, like setUp method'**)  
 yield  
 print(**'this will execute after every method , like tearDown method'**)  
  
def test\_yield(yield\_fixtures):  
 print(**'this is the real test case method, this executes b/w the setUp and tearDown methods'**)

**Running or Execute the Pytest in different ways:**  
file name should start with test  
test method name should start with test

* py.test test\_mod.py # run tests in module
* py.test somepath # run all tests below somepath
* py.test test\_module.py::test\_method # only run test\_method in test\_module
* -s to print statements
* -v verbose

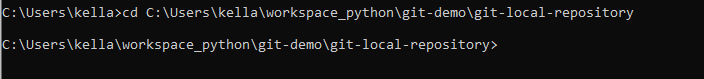
**--------------------------**

**GIT :  
Git Hub User name and password  
Email :** [**Shivakumar.kella@gmail.com**](mailto:Shivakumar.kella@gmail.com) **password:Shiva@9000**

**Username : ShivakumarKella  
1.**Create a folder

2.Make the folder as GIT local repository by executing the following command in the command promt

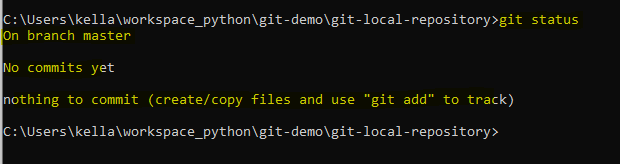
-> first we have to go the folder in command prompt  
 **cd C:\Users\kella\workspace\_python\git-demo\git-local-repository>**



->Make this folder location as GIT local repository by **‘git init’**



->To confirm this ,we can see **.git** in the next line i.e. now it was the git local repository(but it was hidden we cannot see anything in the folder)  
3. We can see what is the status of the GIT repository by **git status**

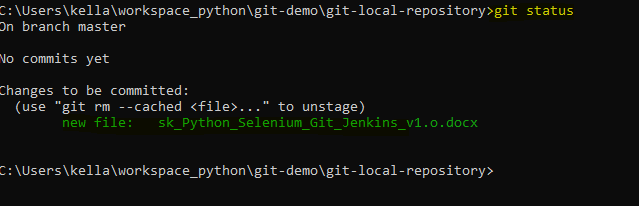


4.lets add the file to the repository by executing this in the command prompt git add filename.extension

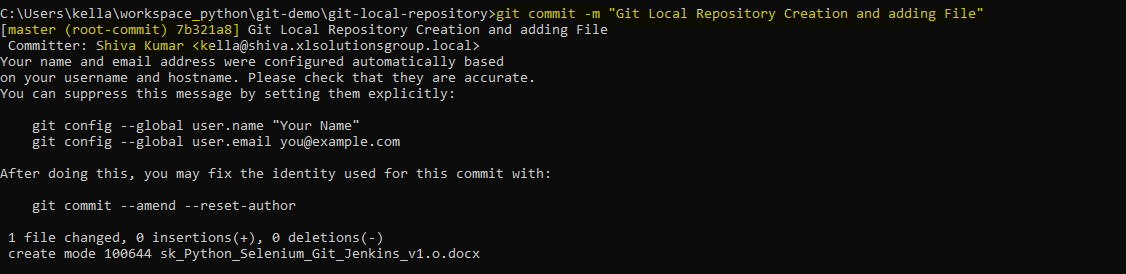
**git add sk\_Python\_Selenium\_Git\_Jenkins\_v1.o.docx**



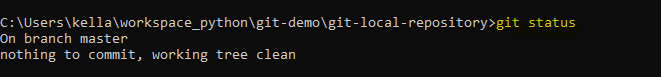
5.we can check weather file added or not by **git status**, first time it will say the new file.



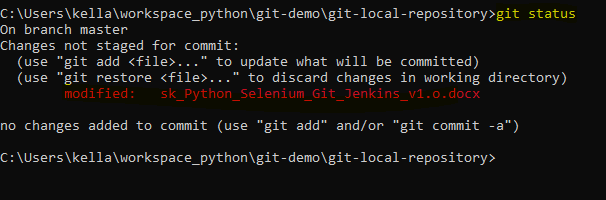
6. check-in the file to local repository i.e. commit the changes/modification/add on the files by **git commit -m ‘comment’**



7. when we have nothing to check-in or adding to the file , status will be

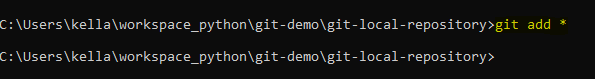


8. if we did any change/modified the file then the status will be the

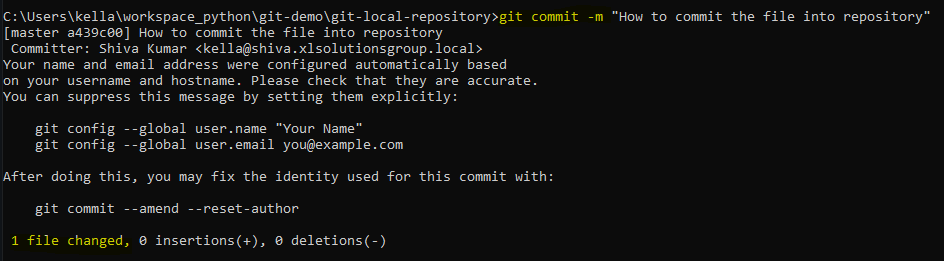


9.modified files we can add it back to the repository by

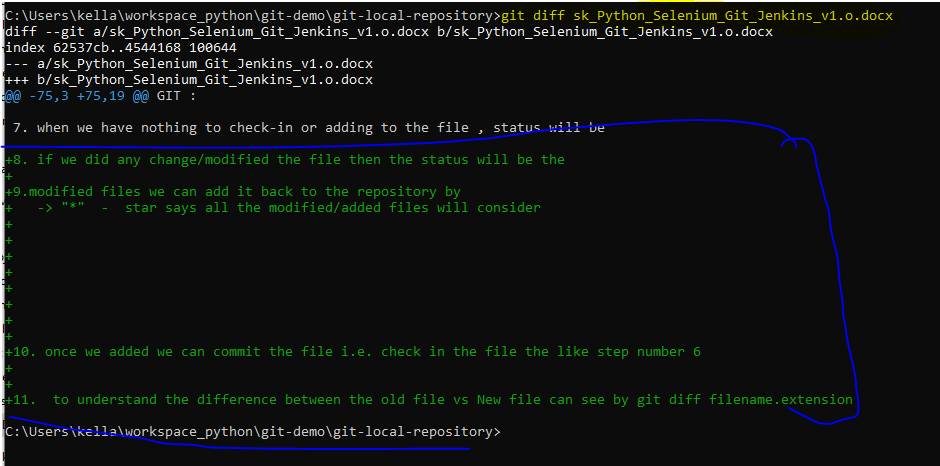
-> “\*” – star says all the modified/added files will consider



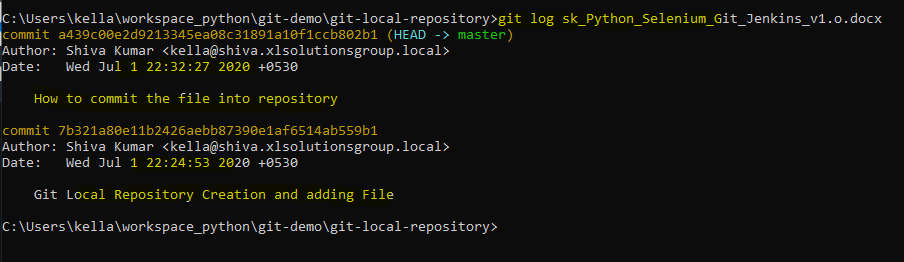
10. once we added we can commit the file i.e. check in the file the like step number 6



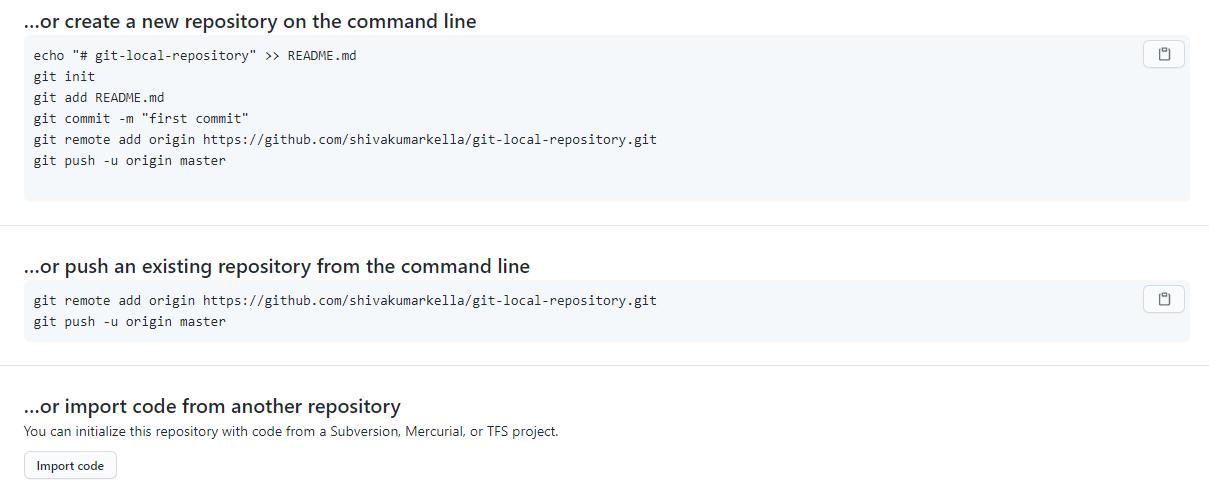
**11.**  to understand the difference between the old file vs New file can see by **git diff filename.extension**



12. in-order to see the log’s of the file we can execute the **git log filename.extension**



**GiT HUB :  
repository : https://github.com/shivakumarkella/git-local-repository.git  
1.**  Create a repository



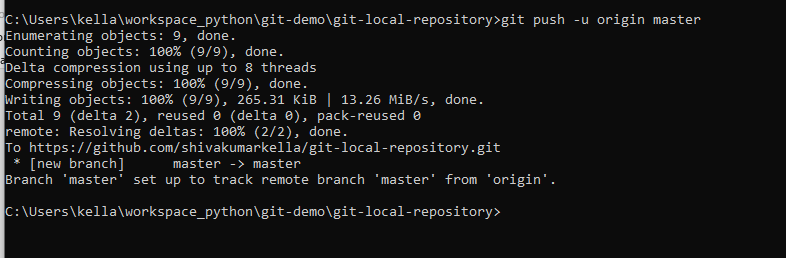
13.Now add the connection between the local repository to master GIT Hub repository by executing the

**git remote add origin https://github.com/shivakumarkella/git-local-repository.git**



14. now push the files from local repository to master repository , it will ask the username and password

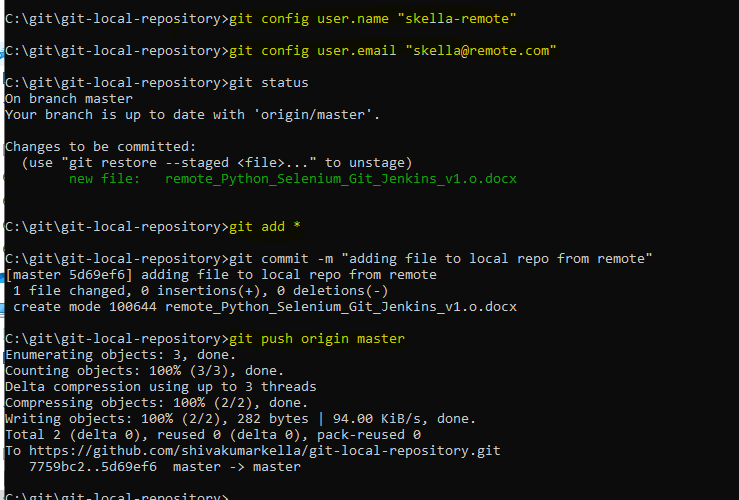
**git push -u origin master**



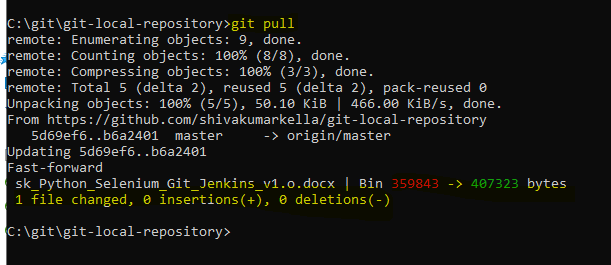
15. to clone the data or import have to use the

**git clone** [**https://github.com/shivakumarkella/git-local-repository.git**](https://github.com/shivakumarkella/git-local-repository.git)

**16.**  sometimes , we have to add the user and password before we commit ,   
 **git config user.name "skella-remote"  
 git config user.email "skella@remote.com"**



**17.**  to take the latest from the HUB/master branch have to use **git pull**



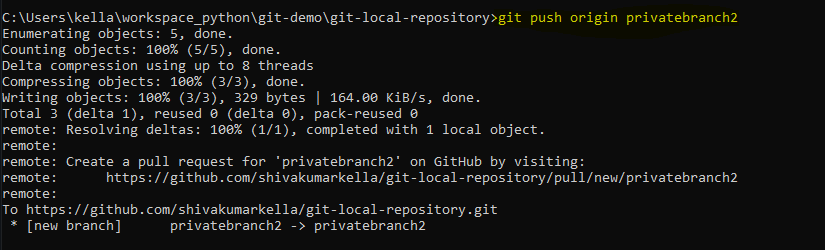
18. to know on what branch we have **git branch** -- star indicates the active branch

19. Create the branches

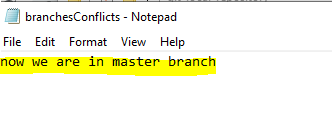
**git checkout -b branchName** 

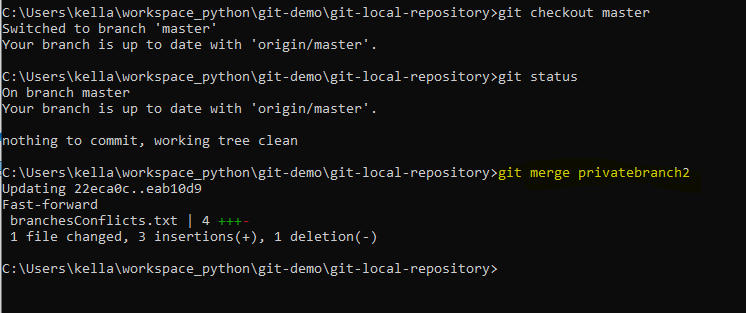
**20.** on private branch modified file push to hub into the same private branch

**git push origin privatebranch2**

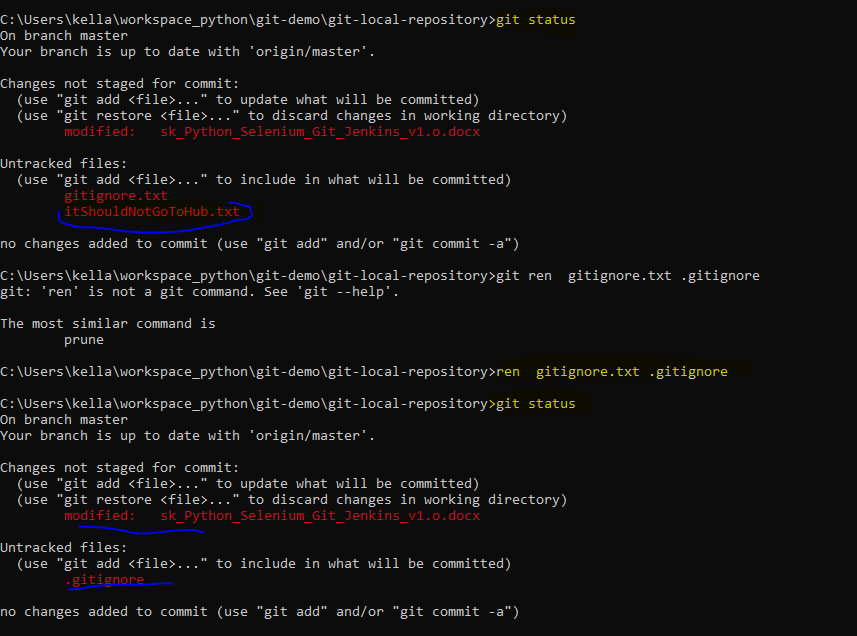


21.merge the code with the master branch -**git merge branchname(from what branch u want to merge)**





22. if we want to ignore the files check-in to GITHUB branches , we have to create the **.gitignore**  file and add the list files or paths to ignore

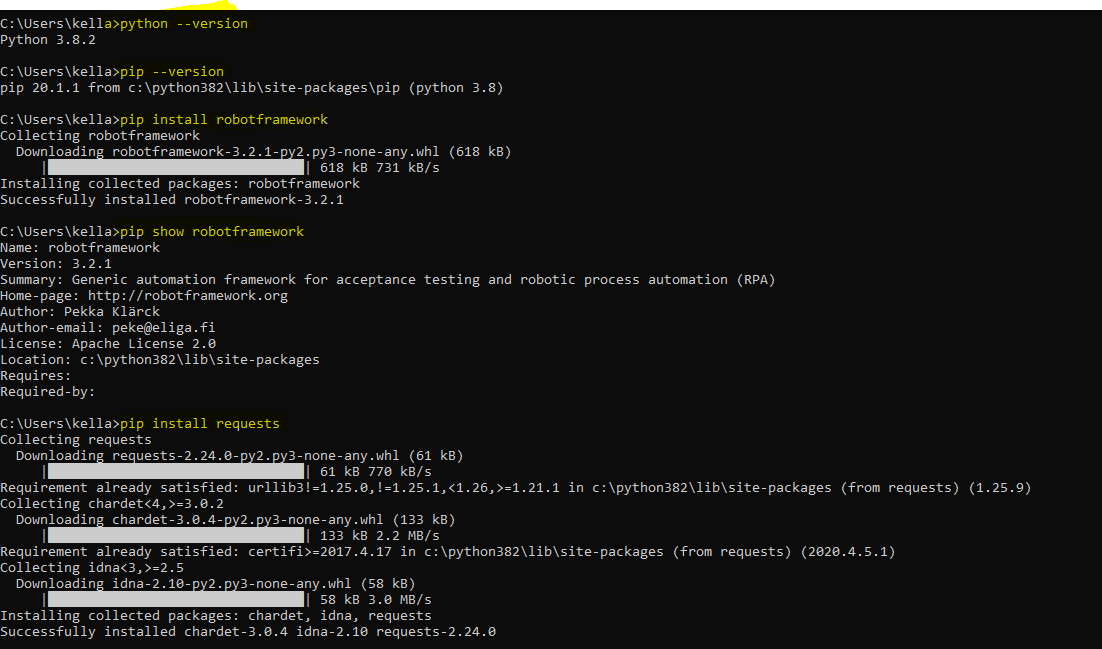
1. **Create** the text **file gitignore**.txt.
2. Open it in a text editor and **add** your rules, then save and close.
3. Hold SHIFT, right click the folder you're in, then select Open command window here.
4. Then rename the **file** in the command line, with **ren gitignore.txt .gitignore**.
5. 

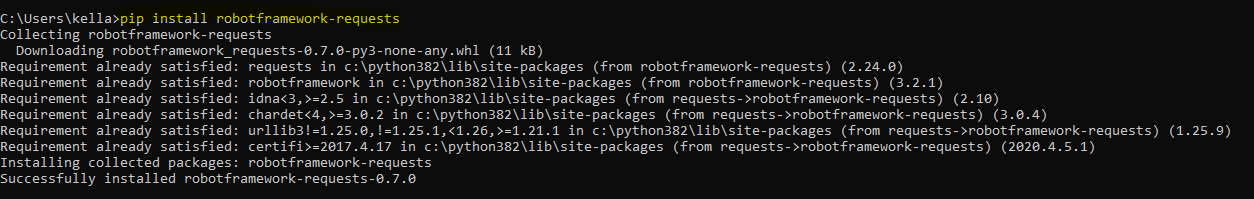
**Rest API Testing with Python by robot framework**

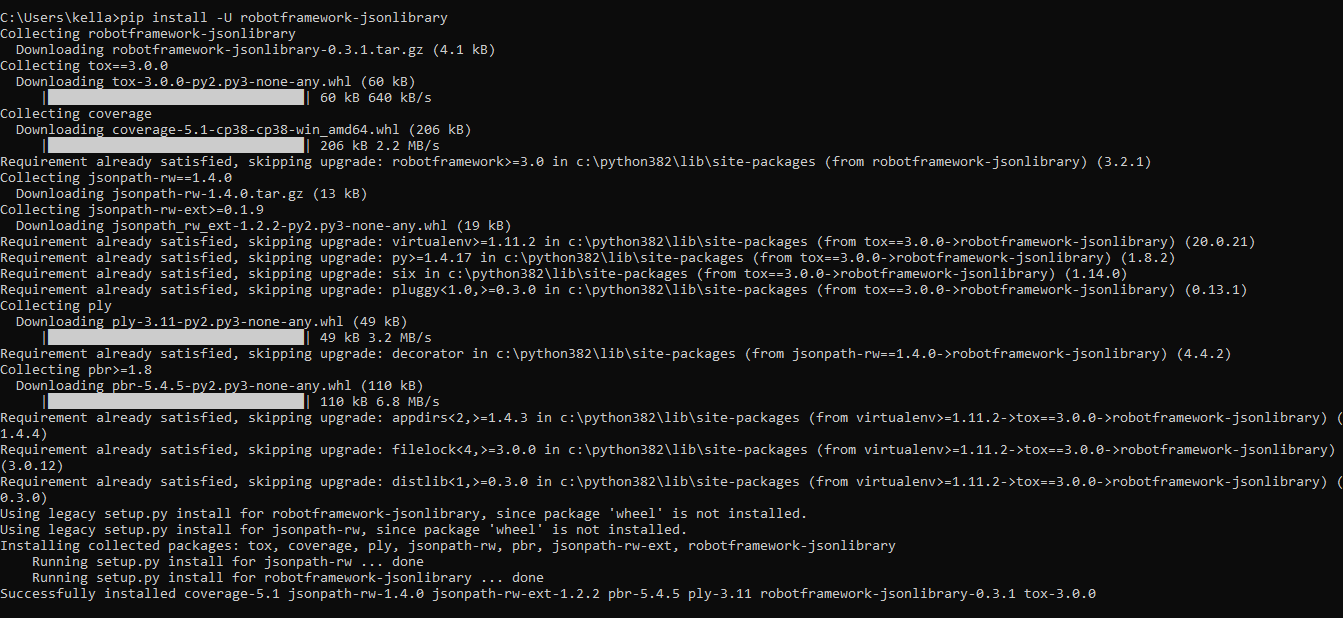
* **Environment set up:**
  + 1. Python
    2. PyCharm
    3. Robot framework
    4. Requests
    5. Robot framework-requests
    6. Robot Framework-json Library

6.1 jsonpath\_rw

6.2 jsonpath\_rw\_ext

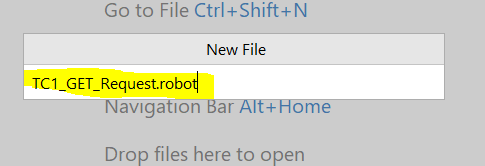


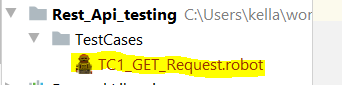




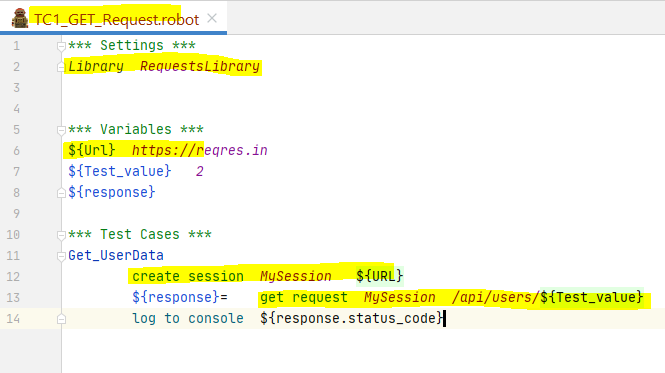


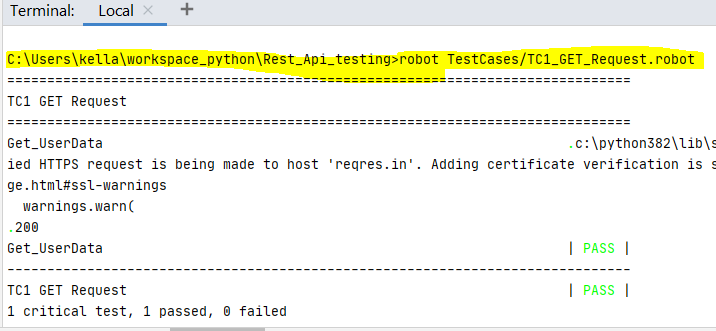
* In-order to work with robot frame work , we have to create a file extension with **.robot**

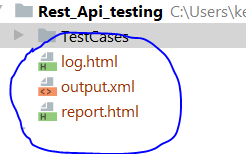
**Ex :** 



1. We need a session to run the API
2. By using **GET Request**  command we can get the response to the variables
3. With keyword **robot robotfilename.robot** will execute the code.







3 types of output report files will be generated.