Version Control System

1) Local History

2) Local Git

3) Push to Git Hub

4) Check Out From Git Hub

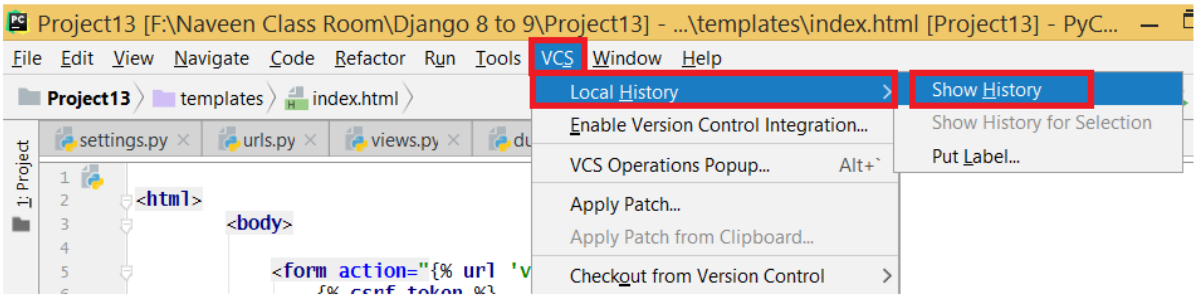
Note: Create a Project in Pycharm and write some code and do changes

1) Local History

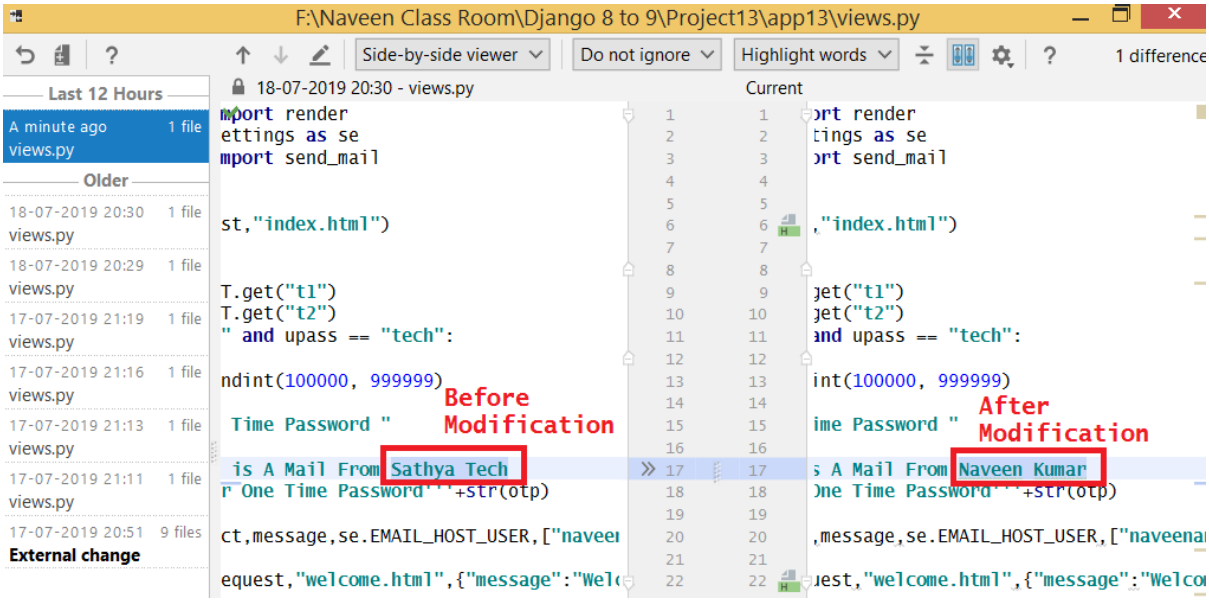
PyCharm uses **Local History** to constantly track all changes made to projects. **Local History** automatically maintains revisions for all meaningful events. This produces a detailed timeline of changes in project structure and source code, with the ability to roll back to any point if necessary.

To view Local History:

On the **VCS** menu, point to **Local History**, and then click **Show History**.

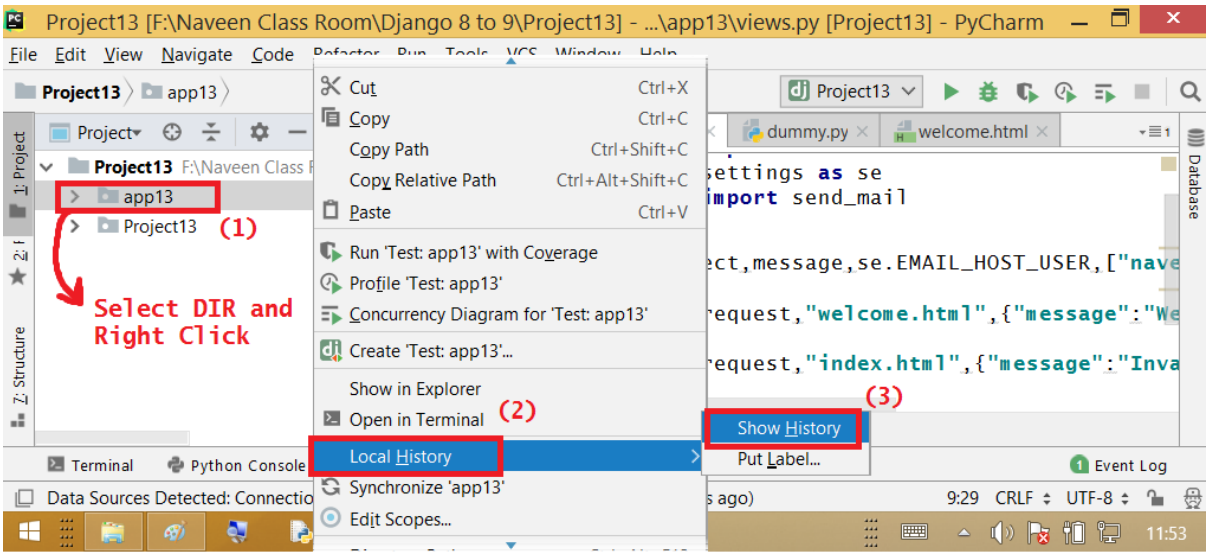


It Will Open

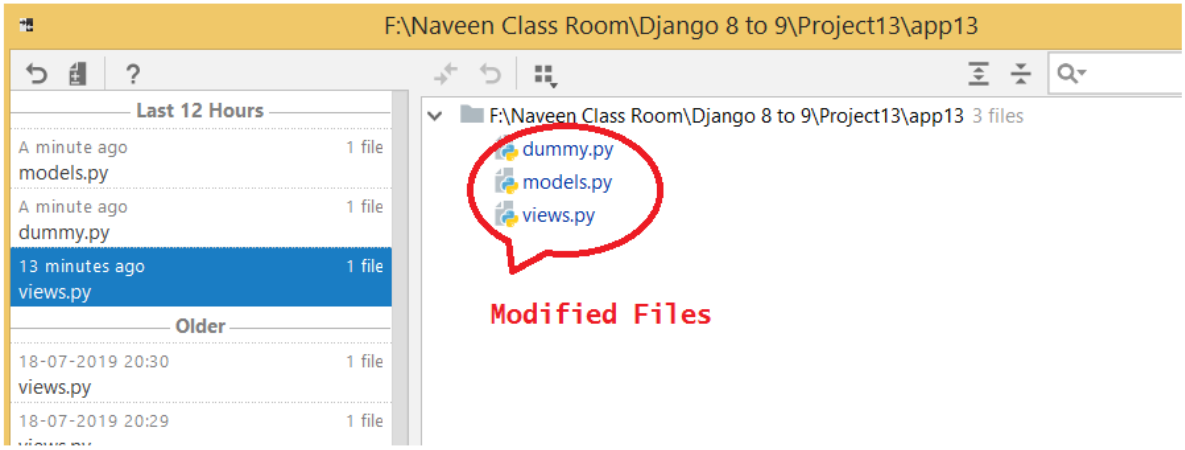


Local History contains a list of revisions and shows differences between them.

When viewing **Local History** for a directory, revisions show only names of files that changed in the directory.

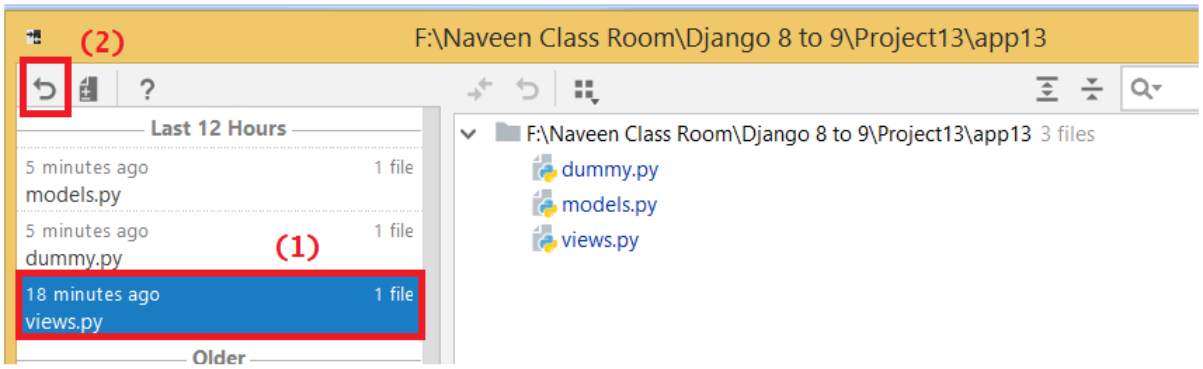


It Will Open



To revert to a specific revision:

Select it from the list and click **Revert** on the toolbar.



Viewing Local History for source code elements

In some cases, you may not want to view **Local History** for the whole file, but only for specific source code elements: classes, their members (fields and methods), or any selected fragment of text.

To view Local History for a class:

- Right-click the class name in the editor, point to **Local History**, and then click **Show History for Class**.

To view Local History for a class field:

- Right-click the field name in the editor, point to **Local History**, and then click **Show History for Field**.

To view Local History for a class method:

- Right-click the method name in the editor, point to **Local History**, and then click **Show History for Method**.

To view Local History for a source code fragment:

- Right-click the selected text in the editor, point to **Local History**, and then click **Show History for Selection**.

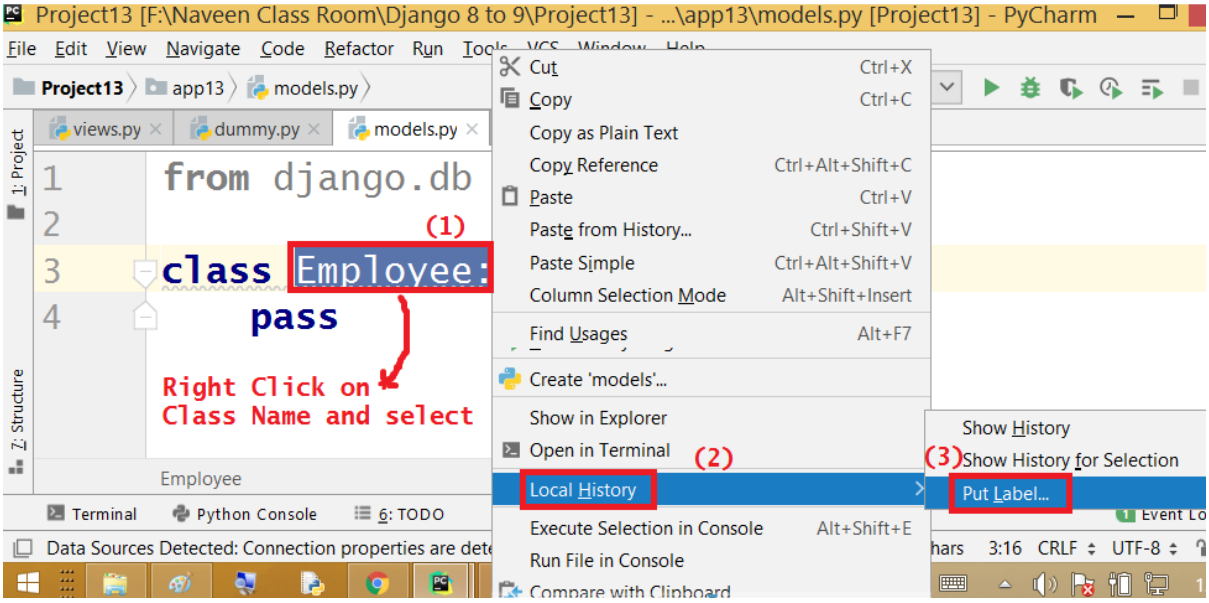
Adding labels to Local History

Local History contains revisions with timestamps, which are not easy to navigate. Some revisions are automatically marked with labels based on predefined events: running tests, deploying apps, committing changes, etc. You can also add custom labels to mark specific milestones in **Local History**.

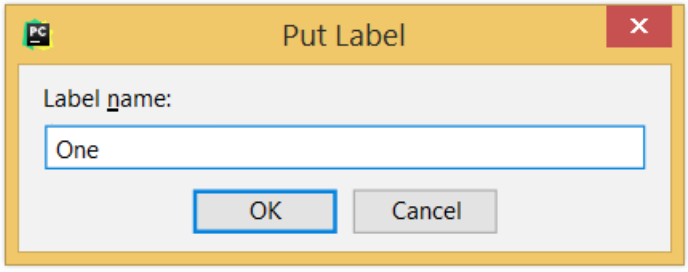
To add a label to the current revision:

1. On the **VCS** menu, point to **Local History**, and then click **Put Label**.
2. In the **Put Label** dialog box, type the label name and click **OK**.

Note: We can give Multiple labels to one element



It Will Open



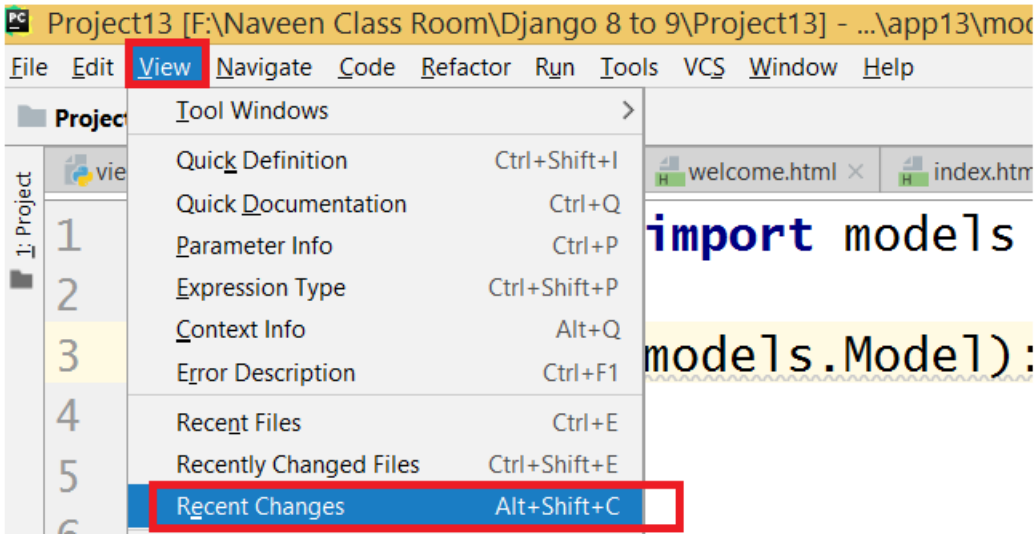
Viewing recent changes

Besides the detailed Local History, PyCharm provides a summary of major recent changes to all projects.

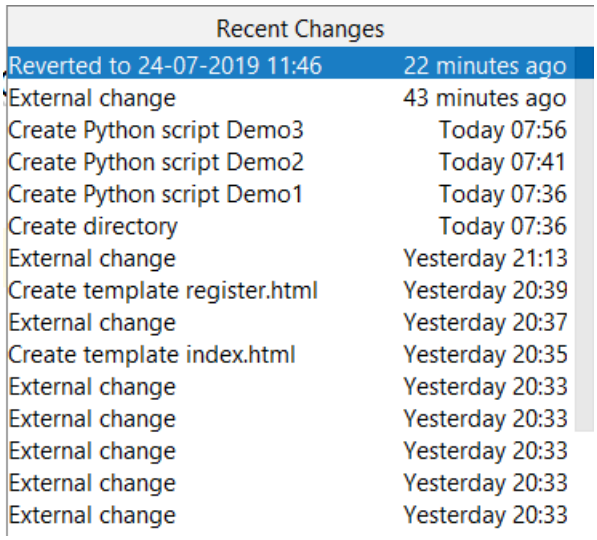
To view recent changes:

- **On the View menu, click Recent Changes.**

You can select the change you are interested in to review differences and revert it if necessary.


Project13 [F:\Naveen Class Room\Django 8 to 9\Project13] - ...\app13\mod
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Tool Windows
Quick Definition Ctrl+Shift+I
Quick Documentation Ctrl+Q
Parameter Info Ctrl+P
Expression Type Ctrl+Shift+P
Context Info Alt+Q
Error Description Ctrl+F1
Recent Files Ctrl+E
Recently Changed Files Ctrl+Shift+E
Recent Changes Alt+Shift+C
welcome.html
index.htm
import models
models.Model):

It Will Open


Recent Changes
Reverted to 24-07-2019 11:46 22 minutes ago
External change 43 minutes ago
Create Python script Demo3 Today 07:56
Create Python script Demo2 Today 07:41
Create Python script Demo1 Today 07:36
Create directory Today 07:36
External change Yesterday 21:13
Create template register.html Yesterday 20:39
External change Yesterday 20:37
Create template index.html Yesterday 20:35
External change Yesterday 20:33
External change Yesterday 20:33
External change Yesterday 20:33
External change Yesterday 20:33
External change Yesterday 20:33

Git

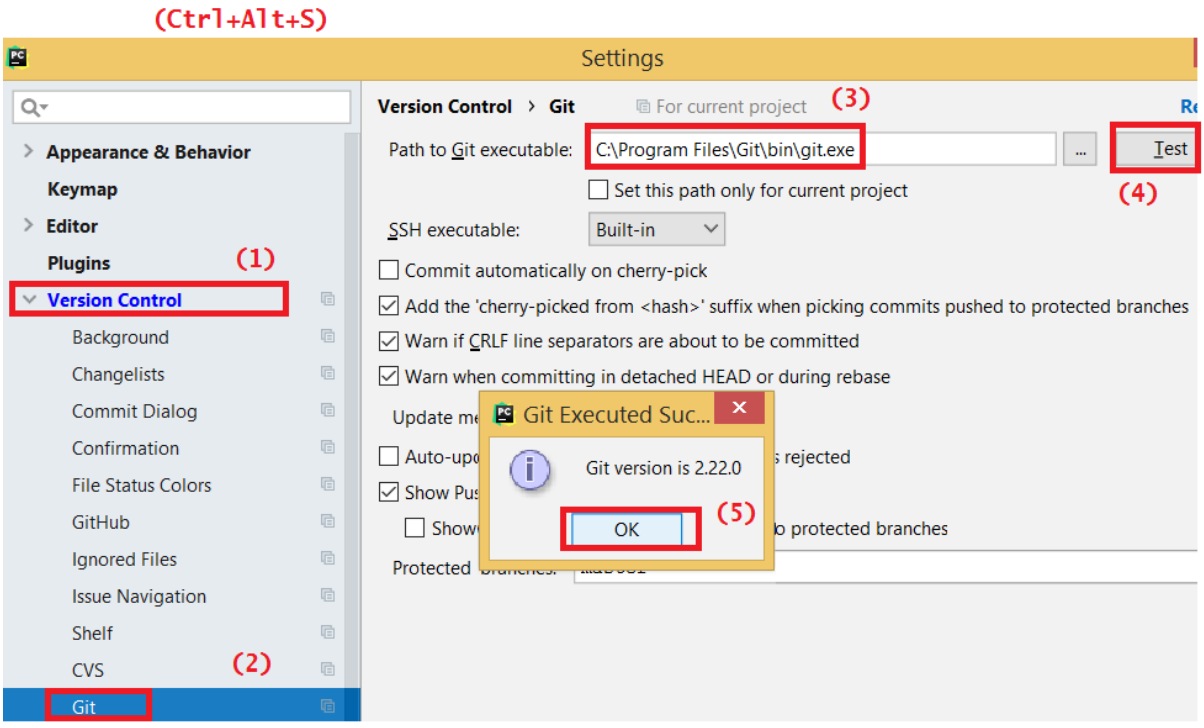
Before you can enable Git version control for an existing local project, or clone a Git project from a remote repository, do the following:

1. Download and install Git.

   From: https://git-scm.com/downloads

   The minimal supported version is 2.4 for Windows and 1.8.0.0 for Linux and MacOS.

2. In the Settings/Preferences dialog (Ctrl+Alt+S), select Version Control | Git in the left pane and specify the path to the Git executable.

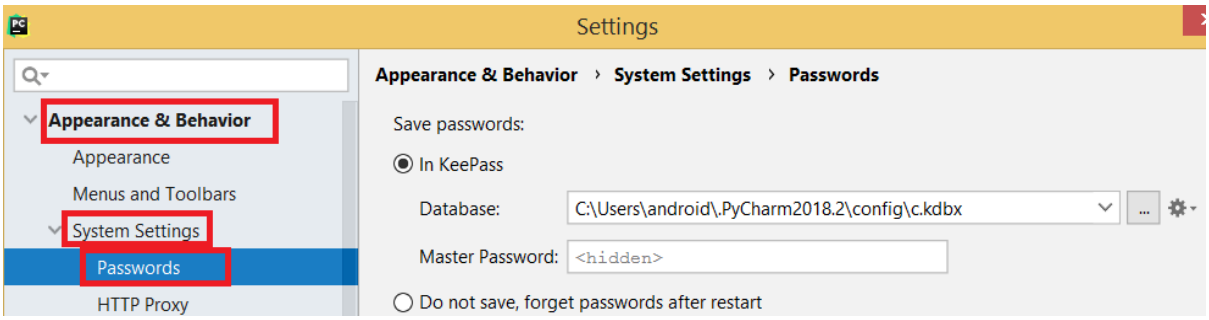


3. Set passwords for remote Git repositories

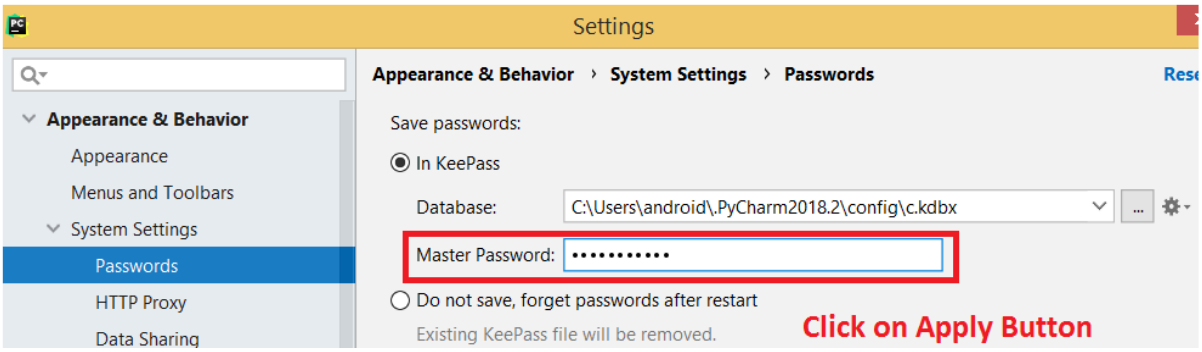
   Every time you interact with a remote Git repository (for example, during a pull, update, or push operation), it requires authorization. You can configure PyCharm to remember your passwords, so that you do not have to specify your credentials each time authorization is required.

Configure a password policy

1. In the Settings dialog (Ctrl+Alt+S), select Appearance and Behavior | System Settings | Passwords on the left.



2. Select how you want PyCharm to process passwords for Git remote repositories:



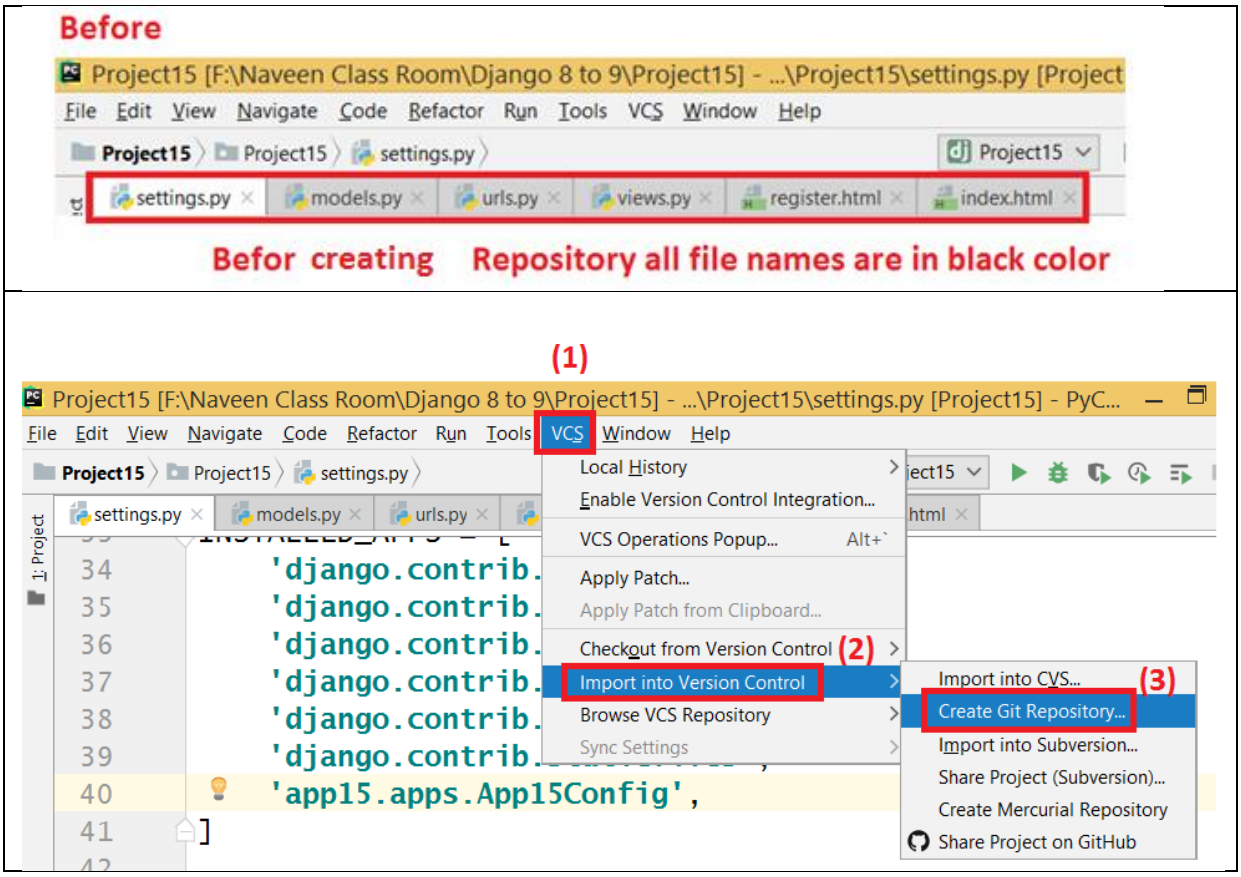
2) Local Git

Put an existing project under Git version control

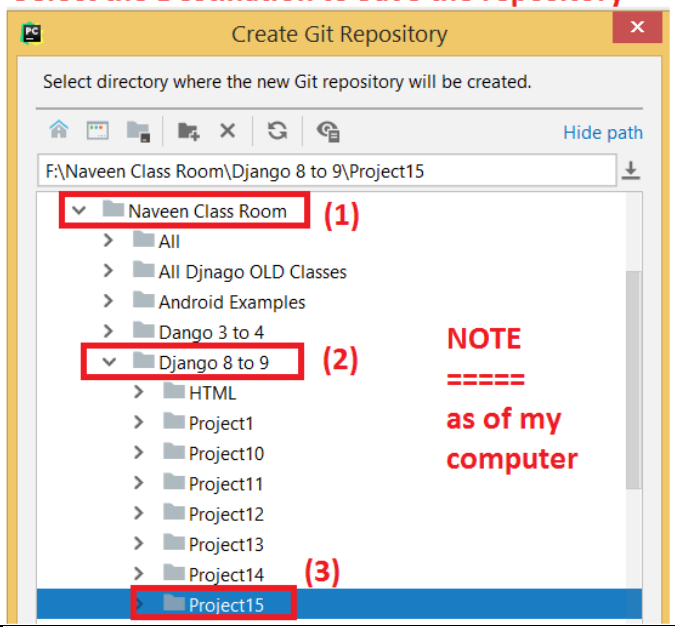
Apart from cloning a remote repository, you can create a local repository based on an existing project's sources.

Import an entire project into a single Git repository

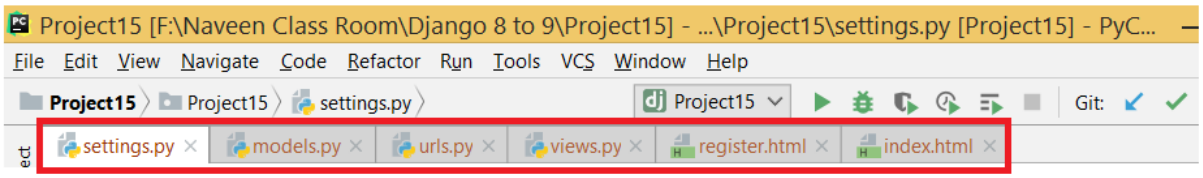
1. Open the project that you want to put under Git.
2. From the main menu, choose VCS



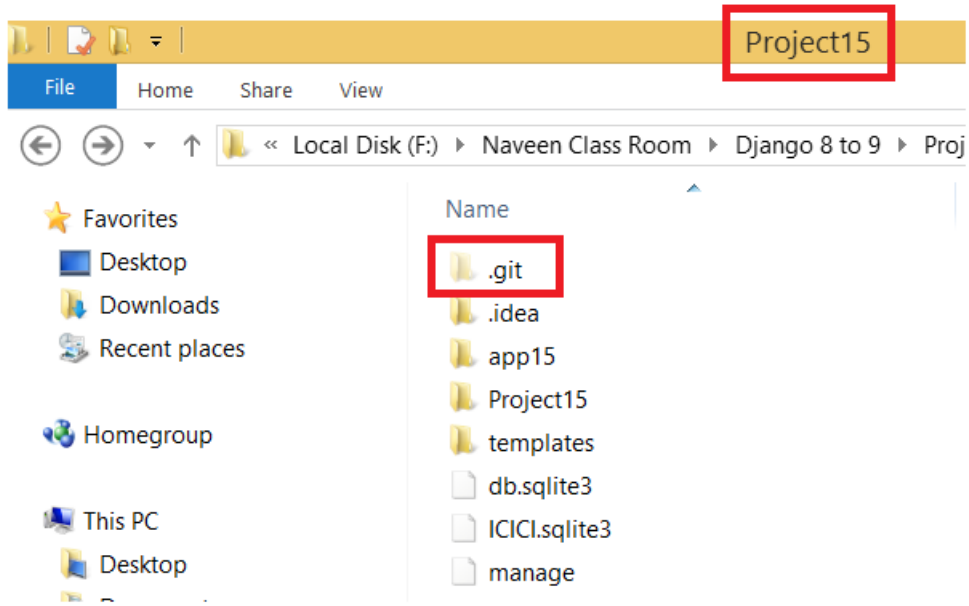
Select the Destination to Save the repository



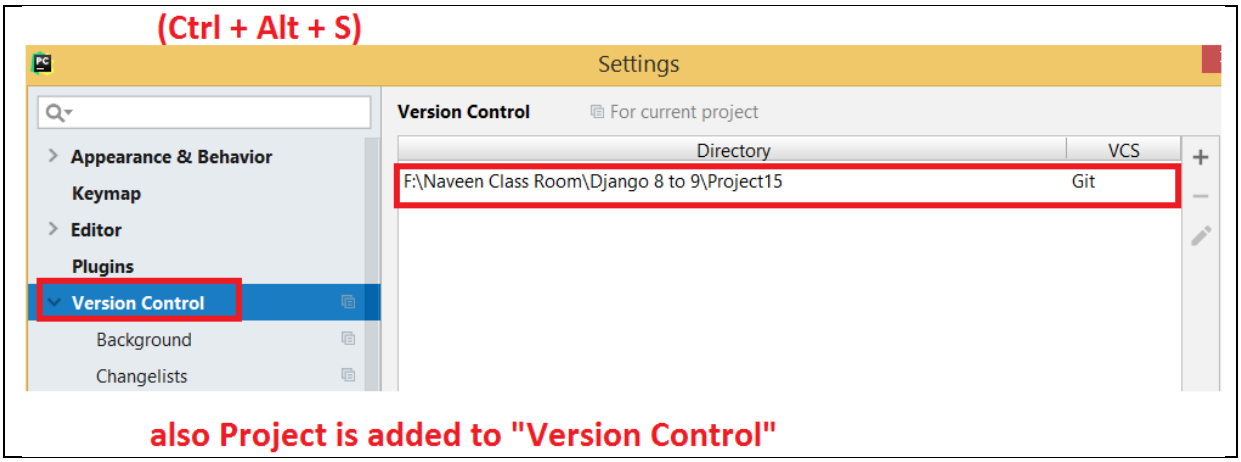
After



After Creating Git Repository we can see all fil names in red color it means Repository is created but files are not added to Repository

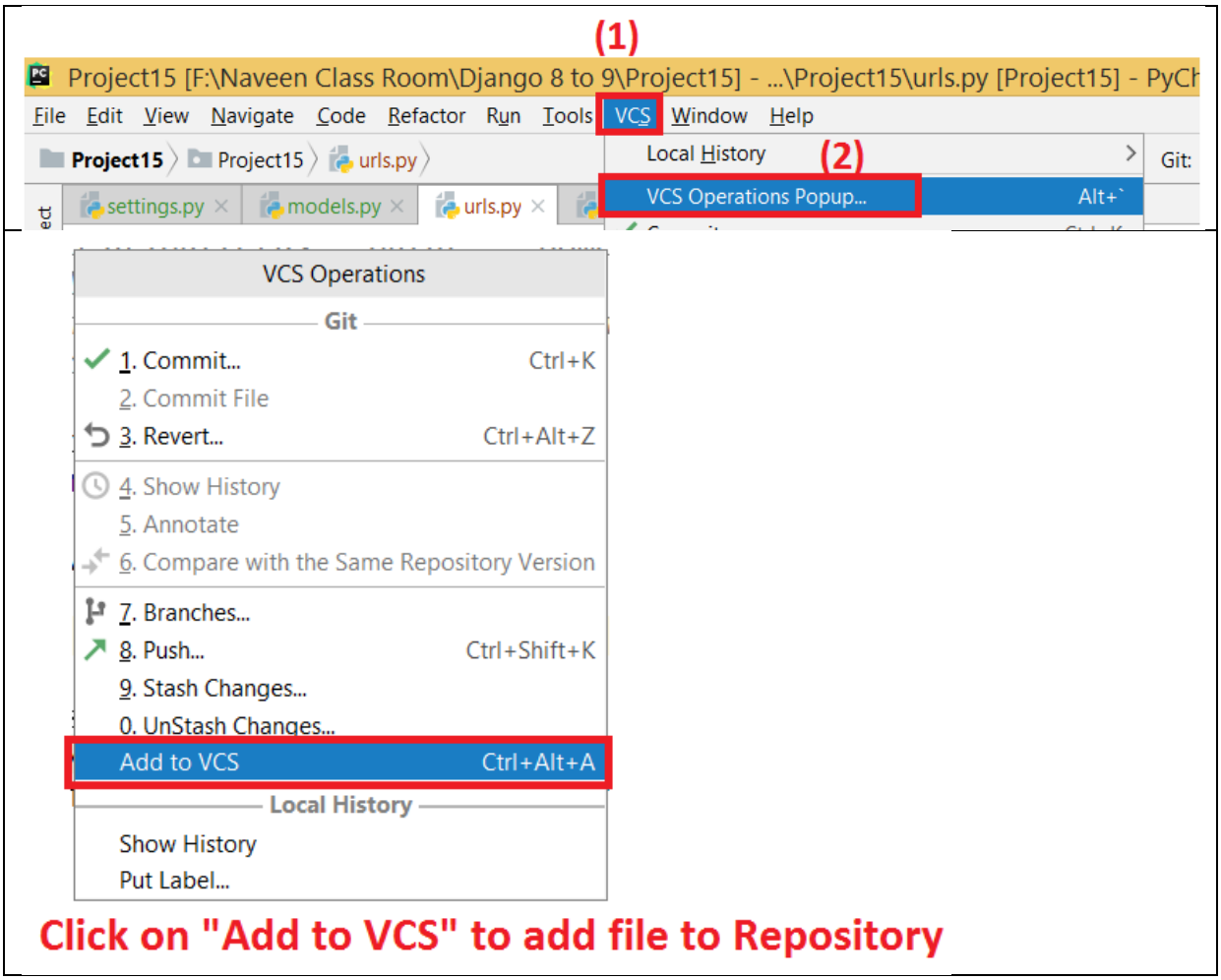


Once the Repository is Created ..... It Will create a ".get" Folder.



Adding Files to Repository

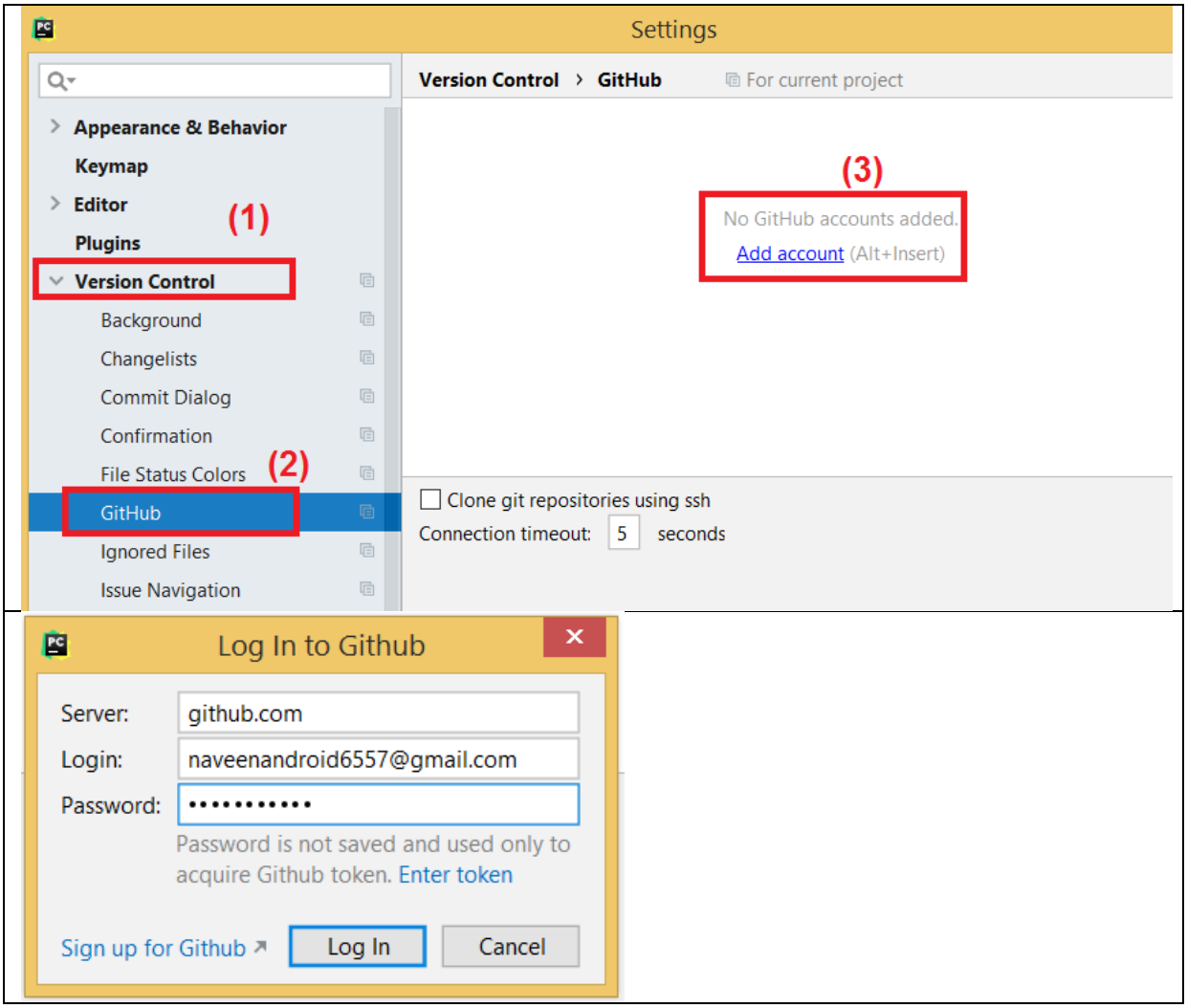
Select the File which you want to add it to Repository



3) Push Project to Git Hub

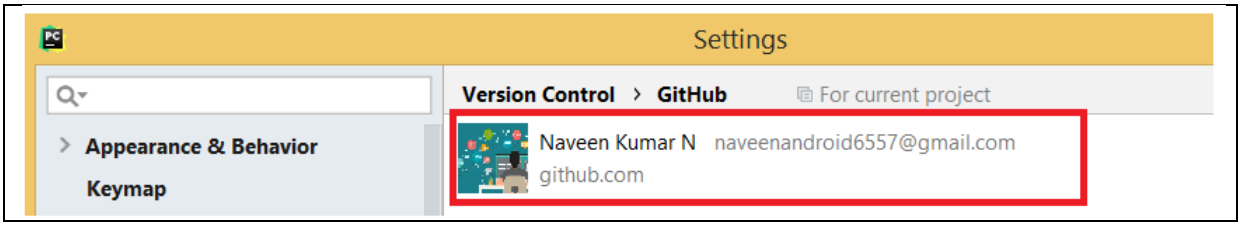
Steps to link Pycharm IDE with Github.

In Pycharm Menu select "File" and select "Settings"

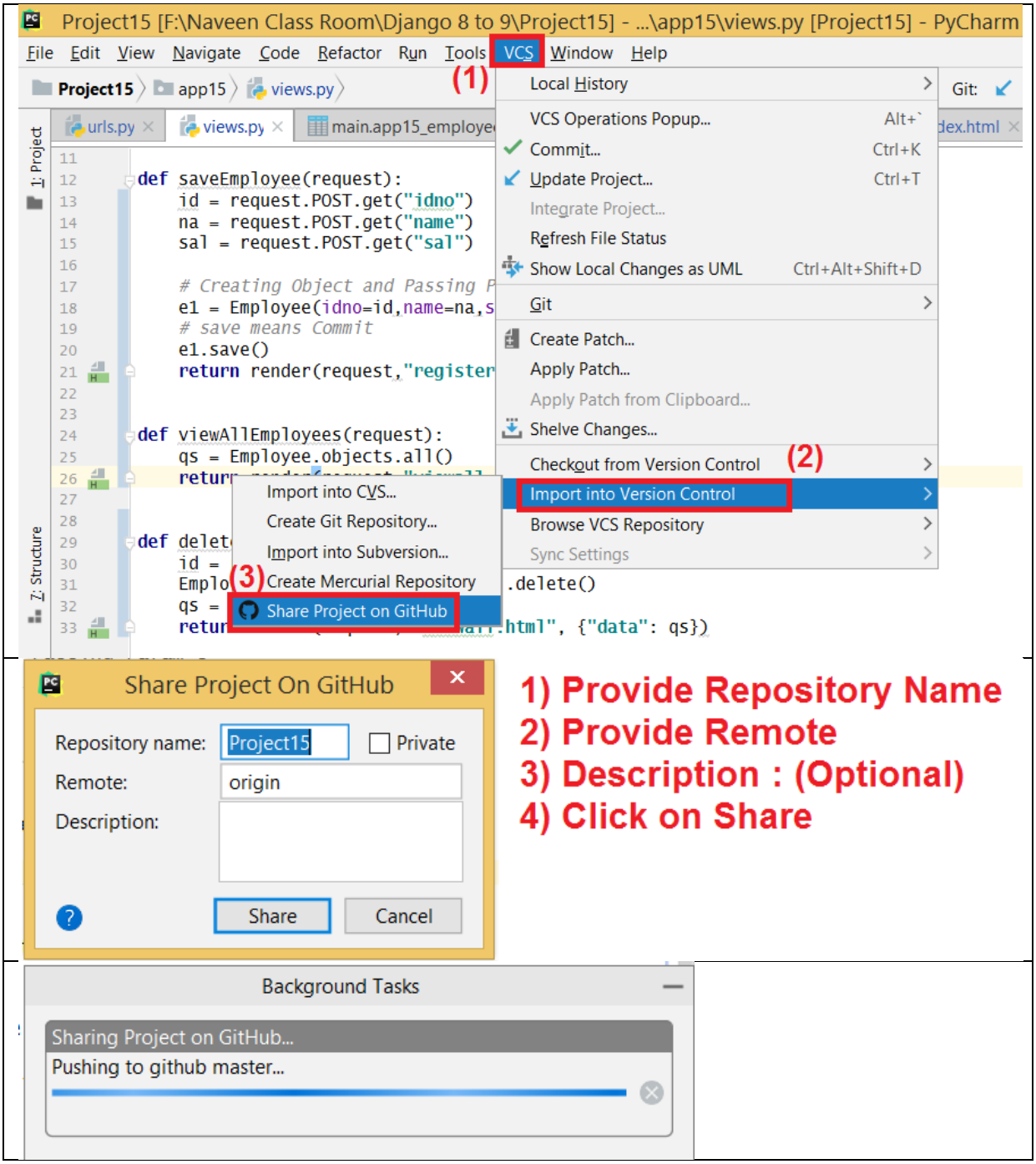


Provide Login and Password and click on Login In Button.

If Given email id and Password is valid it will link Your Github account



Steps to Push into Github



Optional Screen