About the Tutorial

PyCharm is the most popular IDE for Python, and includes great features such as excellent code completion and inspection with advanced debugger and support for web programming and various frameworks. PyCharm is created by Czech company, Jet brains which focusses on creating integrated development environment for various web development languages like JavaScript and PHP.

Audience

This tutorial has been prepared for Python developers who focus on using IDE with complete package of running, debugging and creating projects in various python frameworks. Also, interested learners with a basic knowledge of any IDE can take up this tutorial.

Prerequisites

Before proceeding with this tutorial, you need a basic knowledge of any integrated development environment of Python like Sublime Text or most popular IDE like NetBeans. If you are a beginner, we suggest you to go through tutorials related to these topics first before proceeding further on this tutorial.

Copyright & Disclaimer

© Copyright 2018 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com
# PyCharm

## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the Tutorial</td>
<td>i</td>
</tr>
<tr>
<td>Audience</td>
<td>i</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>i</td>
</tr>
<tr>
<td>Copyright &amp; Disclaimer</td>
<td>i</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>ii</td>
</tr>
<tr>
<td><strong>1. PYCHARM- INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>Features of PyCharm</td>
<td>1</td>
</tr>
<tr>
<td><strong>2. PYCHARM – INSTALLATION</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>3. PYCHARM – UNDERSTANDING BASICS</strong></td>
<td>8</td>
</tr>
<tr>
<td><strong>4. PYCHARM – KEYMAPS</strong></td>
<td>12</td>
</tr>
<tr>
<td>Finding Shortcut</td>
<td>14</td>
</tr>
<tr>
<td><strong>5. PYCHARM – SHORTCUTS</strong></td>
<td>14</td>
</tr>
<tr>
<td><strong>6. PYCHARM – OMNI</strong></td>
<td>16</td>
</tr>
<tr>
<td>Functionalities</td>
<td>16</td>
</tr>
<tr>
<td><strong>7. PYCHARM – MACROS</strong></td>
<td>18</td>
</tr>
<tr>
<td>Navigate Macro</td>
<td>18</td>
</tr>
<tr>
<td>Search Everywhere</td>
<td>19</td>
</tr>
<tr>
<td><strong>8. PYCHARM – MICROS</strong></td>
<td>20</td>
</tr>
<tr>
<td>Scroll from Source</td>
<td>20</td>
</tr>
<tr>
<td>Collapse All</td>
<td>20</td>
</tr>
<tr>
<td>Show Options menu</td>
<td>21</td>
</tr>
<tr>
<td>Hide</td>
<td>22</td>
</tr>
<tr>
<td><strong>9. PYCHARM – IMPROVING AND WRITING CODE</strong></td>
<td>24</td>
</tr>
</tbody>
</table>
Improving Code Completion ........................................................................................................24
Intention Actions ..........................................................................................................................26

10. PYCHARM – CONSOLE ........................................................................................................27

11. PYCHARM – INTERPRETERS ...........................................................................................29
Creation of Interpreter ..................................................................................................................29

12. PYCHARM – DEBUGGING AND BREAKPOINTS ...............................................................32
Understanding Breakpoints ............................................................................................................34

13. PYCHARM – INTEGRATION OF VERSION CONTROL .......................................................36
Steps Involved ............................................................................................................................36

14. PYCHARM – HTML AND CSS INTEGRATION .................................................................39
Emmet ...........................................................................................................................................39
Creating HTML and CSS files .......................................................................................................40
Creating CSS file ..........................................................................................................................41

15. PYCHARM – JAVASCRIPT SUPPORT .................................................................................43

16. PYCHARM – TIPS .................................................................................................................46
Changing the File to a Specific Changelist ..................................................................................46
Display the List of all Usages in a Class .......................................................................................47
To find Menu Command for an Action ..........................................................................................47
Running Inspection through a Code ..............................................................................................48
Specify the List of Settings ............................................................................................................48
Run / Debug the Script Files .........................................................................................................49

17. PYCHARM – DATABASE TOOLS ......................................................................................50
Adding a Data Source ....................................................................................................................50

18. PYCHARM – EXPORTING DATA ......................................................................................53
Export to HTML feature .................................................................................................................53
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>PYCHARM – WEB FRAMEWORKS</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>PYCHARM – DJANGO</td>
<td>57</td>
</tr>
<tr>
<td>21</td>
<td>PYCHARM – PYRAMID</td>
<td>59</td>
</tr>
<tr>
<td>22</td>
<td>PYCHARM – FLASK</td>
<td>61</td>
</tr>
</tbody>
</table>
1. PyCharm - Introduction

PyCharm is the most popular IDE used for Python scripting language. This chapter will give you an introduction to PyCharm and explains its features.

PyCharm offers some of the best features to its users and developers in the following aspects:

- Code completion and inspection
- Advanced debugging
- Support for web programming and frameworks such as Django and Flask

Features of PyCharm

Besides, a developer will find PyCharm comfortable to work with because of the features mentioned below:

Code Completion

PyCharm enables smoother code completion whether it is for built in or for an external package.

SQLAlchemy as Debugger

You can set a breakpoint, pause in the debugger and can see the SQL representation of the user expression for SQL Language code.

Git Visualization in Editor

When coding in Python, queries are normal for a developer. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one.

Code Coverage in Editor

You can run .py files outside PyCharm Editor as well marking it as code coverage details elsewhere in the project tree, in the summary section etc.

Package Management

All the installed packages are displayed with proper visual representation. This includes list of installed packages and the ability to search and add new packages.

Local History

Local History is always keeping track of the changes in a way that complements like Git. Local history in PyCharm gives complete details of what is needed to rollback and what is to be added.
**Refactoring**

Refactoring is the process of renaming one or more files at a time and PyCharm includes various shortcuts for a smooth refactoring process.

**User Interface of PyCharm Editor**

The user interface of PyCharm editor is shown in the screenshot given below. Observe that the editor includes various features to create a new project or import from an existing project.

From the screenshot shown above, you can see the newly created project Demo and the site-packages folder for package management along with various other folders.

You can download the PyCharm Editor and read its official documentation at this link:

[https://www.jetbrains.com/pycharm/](https://www.jetbrains.com/pycharm/)
In this chapter, you will learn in detail about the installation process of PyCharm on your local computer.

**Steps Involved**

You will have to follow the steps given below to install PyCharm on your system. These steps show the installation procedure starting from downloading the PyCharm package from its official website to creating a new project.

**Step 1**

Download the required package or executable from the official website of PyCharm [https://www.jetbrains.com/pycharm/download/#section=windows](https://www.jetbrains.com/pycharm/download/#section=windows). Here you will observe two versions of package for Windows as shown in the screenshot given below:

![PyCharm Download](image)

Note that the professional package involves all the advanced features and comes with free trial for few days and the user has to buy a licensed key for activation beyond the trial period. Community package is for free and can be downloaded and installed as and when required. It includes all the basic features needed for installation. Note that we will continue with community package throughout this tutorial.
Step 2

Download the community package (executable file) onto your system and mention a destination folder as shown below:
Step 3

Now, begin the installation procedure similar to any other software package.
Step 4

Once the installation is successful, PyCharm asks you to import settings of the existing package if any.
This helps in creating a new project of Python where you can work from the scratch. Note that unlike other IDEs, PyCharm only focusses on working with projects of Python scripting language.
This chapter will discuss the basics of PyCharm and make you feel comfortable to begin working in PyCharm editor.

When you launch PyCharm for the first time, you can see a welcome screen with entry points to IDE such as:

- Creating or opening the project
- Checking out the project from version control
- Viewing the documentation
- Configuring the IDE

Recall that in the last chapter, we created a project named **demo1** and we will be referring to the same project throughout this tutorial. Now we will start creating new files in the same project to understand the basics of PyCharm Editor.
The above snapshot describes the project overview of demo1 and the options to create a new file. Let us create a new file called `main.py`.

The code included in `main.py` is as follows:

```python
y = 3
def print_stuff():
    print ("Calling print_stuff")
    print (y)
    z = 4
    print (z)
    print("exiting print_stuff")

print_stuff()  # we call print_stuff and the program execution goes to (***)
p'y
# works fine
print (z)  # NameError!!!
```
The code created in the file `main.py` using PyCharm Editor is displayed as shown below:

This code can be run within IDE environment. The basic demonstration of running a program is discussed below:
Note that we have included some errors within the specified code such that console can execute the code and display output as the way it is intended to.
4. PyCharm – Keymaps

PyCharm includes various Keymaps to show the most-used commands in the editor. This chapter discusses Keymaps in detail.

You can find the list of Keymaps available in the file menu Help -> Keymap Reference as shown in the screenshot given below:
You can find the list of Keymaps and the available shortcuts in PDF format as shown below:

Note: The default Keymap for Windows and Linux operating systems is default, while in Mac OS the default Keymap is OSX 10.5.

You can also view the list of Keymaps available using the Settings option in Windows and Linux Operating system (Preferences in Mac OS) as shown in the screenshot given below:

The default Keymap includes various sections for Editor Actions, Main Menu, Tool Windows, External tools, Version Control System, Macros, Quick Lists, Plug-ins and Other options as well.
Shortcuts are the combinations of keys being used to perform a set of activities. You can find the list of PyCharm shortcuts in Keymaps guide reference.

**Finding Shortcut**

The list of shortcuts is available in the following option **Help -> Find Action** menu where it pops up with a shortcut window.
You can see the shortcut window as shown here:

The shortcut includes a list of Identifiers, shortcuts with functions and option menu bar. For example, View Navigation Bar includes toggle ON and OFF which displays the navigation bar as per the value set (ON and OFF).
Omni is the section in PyCharm which deals into anywhere from any place. It includes various tools for a user to move from one place to another. It helps in such a scenario that you need to quickly move from one project directory into another. This chapter will familiarize you with the functionalities of Omni.

Functionalities

The Navigate menu describes the functionalities involved in Omni. This section discusses these in detail:

Class

This helps to navigate from one class to another in a mentioned project. This is very helpful to navigate through a list of classes.
**Back**

This option helps to move backwards from the existing state. The shortcut key is **Ctrl+Alt+Left**.

**Forward**

It works similar to the **back** option. However, the functionality is completely vice-versa.
The difference between a macro and Omni is subtle in PyCharm Editor. Omni allows you to go to the exact location of editor or a specified place of code with no particular significance. Macro on the other hand allows the user to navigate through functions and classes or particular class method.

**Navigate Macro**

Observe the following screenshot for a better understanding of Navigate macro:

The **Navigate -> Declaration** helps to show declaration, type declaration and to define super methods. Various attributes included in the type declaration are shown below:

However, there is an issue with this macro, if a user tries to go to the declaration of a `.so` object for example, navigating from `datetime` module to `select` module, then each time it will encounter the `stub` file.
Search Everywhere

It helps to search the classes and associated methods. It includes the option to search with Google as well.

Each of these parts includes a shortcut key combination next to its section name. **Search Everywhere** is a gateway to other search actions available in PyCharm.
8. PyCharm – Micros

Micros deal with getting places within a specified file. These tools end up using most of the development procedure. In this chapter, you will learn Micro in detail.

Consider the example of **Structure Panel** which is being used as representation of micros.

---

**Scroll from Source**

It helps to scroll from the mentioned source like the complete folder location of the specified file.

![Scroll from Source Image]

---

**Collapse All**

Consider the screenshot shown below which shows opening the file with specified location. In order to collapse the folder structure, you can use the shortcut key shown in the image.

![Collapse All Image]
This shortcut key helps in collapsing the folder location of specified code as shown below.

![Image of PyCharm interface showing code and options]

**Show Options menu**

The **Show Options** menu of the structure panel of project displays the list of options available for the project created. Observe the screenshot shown below for a better understanding:
The list of options is displayed below:

- **Show Members**
- **Autoscroll to Source**
- **Autoscroll from Source**
- **Sort by Type**
- **Folders Always on Top**
- **Show Excluded Files**
- **File Nesting...**
- **Pinned Mode**
- **Docked Mode**
- **Floating Mode**
- **Windowed Mode**
- **Split Mode**
- **Remove from Sidebar**
- **Group Tabs**
  - **Move to**
  - **Resize**

---

**Hide**

This option helps to hide the structure panel of the project window. The user interface of the structure panel after being collapsed is as shown below:
You can reopen the structure panel as shown here:
PyCharm includes various standards for writing code with proper indentations valid for Python. This makes it interesting to improve the code standards and writing the complete code in PyCharm editor.

**Improving Code Completion**

Code completion in PyCharm is really unique. You can enhance it further using many other features. Note that the editor provides start and end of the code block. Consider a file named `demo.py` with the following code:

```python
message = 'GIEWIVrGMTLIVrHIQS' #encrypted message
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for key in range(len(LETTERS)):
    translated = ''
    for symbol in message:
        if symbol in LETTERS:
            num = LETTERS.find(symbol)
            num = num - key
            if num < 0:
                num = num + len(LETTERS)
            translated = translated + LETTERS[num]
        else:
            translated = translated + symbol
    print('Hacking key #%s: %s' % (key, translated))
```
The code is completed using the following construct:

```python
message = 'GIFWVKGTMVLvHIQS'  # encrypted message
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for key in range(len(LETTERS)):
    translated = ''
    for symbol in message:
        if symbol in LETTERS:
            num = 0
            if ord(symbol) + key > ord('Z'):
                num = ord(symbol) + key - ord('Z') - ord('A')
            else:
                num = ord(symbol) + key
            translated = translated + chr(num + ord('A'))
        else:
            translated = translated + symbol
    print(f'Hacking key #: {key} # translation: {translated}')

# prints Hacking key #: 0 # translation: RVIMXWPLALP
# prints Hacking key #: 1 # translation: GFWKQVYJXK
# prints Hacking key #: 2 # translation: DENVUXIYHJ
# prints Hacking key #: 3 # translation: CQQTTWGIXG
# prints Hacking key #: 4 # translation: BPPSSreh HF
# prints Hacking key #: 5 # translation: ASPQRggEDG
# prints Hacking key #: 6 # translation: 9DNGEDCBE
# prints Hacking key #: 7 # translation: 8CECBAEC
# prints Hacking key #: 8 # translation: 7BCBBAEB
# prints Hacking key #: 9 # translation: 6ACBAAED
# prints Hacking key #: 10 # translation: 599C99ED
# prints Hacking key #: 11 # translation: 488C89ED
# prints Hacking key #: 12 # translation: 377C79ED
# prints Hacking key #: 13 # translation: 266C69ED
# prints Hacking key #: 14 # translation: 155C59ED
# prints Hacking key #: 15 # translation: 044C49ED
```

If you press Ctrl + spacebar while this popup is on the screen, you can see more code completion options:
**Intention Actions**

PyCharm includes intent specific actions and the shortcut key for the same is **Alt+Enter**. The most important example of intentions at work is using language injection in strings.

The screenshot given below shows the working of intention actions:

```
message = 'GIEWIVrGMTLIVrHIQS'  #encrypted message
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for key in range(len(LETTERS)):
    translated = ''
    for symbol in message:
        if symbol in LETTERS:
            num = LETTERS.find(symbol)
            num = num - key
            if num < 0:
                num = num + len(LETTERS)
            translated = translated + LETTERS[num]
        else:
            translated = translated + symbol
print('Hacking key #s: %s' % (key, translated))
```

Note that we can insert many different languages of intention actions in PyCharm Editor.
10. PyCharm – Console

PyCharm has a full-fledged Python console with full code completion which is available in the option menu **Tools -> Run Python Console**.

Consider the code which was mentioned in the previous chapter, as shown below:

```python
message = 'GIEWIVrGMTLIVrHIQS' # encrypted message
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
for key in range(len(LETTERS)):
    translated = ''
    for symbol in message:
        if symbol in LETTERS:
            num = LETTERS.find(symbol)
            num = num - key
            if num < 0:
                num = num + len(LETTERS)
            translated = translated + LETTERS[num]
        else:
            translated = translated + symbol
    print('Hacking key #%s: %s' % (key, translated))
```

Now, let us run the code with the help of console to execute the script for getting the desired output, as shown below.

You can observe the output as shown below:
PyCharm includes interpreters to create a new project with new features as the way it is needed. You can create a virtual environment in your system as the way you need it. You can also inherit global site packages in the dialog box. Interpreters are available on Python Package Index (PyPI) and can be easily installed and accessed using `pip install`.

**Creation of Interpreter**

For creating an interpreter, it is always recommended to create a new project where desired configurations are managed. Look at the following screenshot for a better understanding:

These parameters include:

- **Location**: This describes the parameter where virtual environment is created focusing on the location on system.

- **Basic interpreter**: It defines the attributes of interpreter.
The dialog box also refers to the parameter where an existing virtual interpreter will be taken as an attribute. Once the user adds a new local interpreter, PyCharm will ask the user for the binary of interpreter. In most cases, it is always considered to be a .exe file. In case of Jython, it will be always a .bat file.
The details of Project Interpreter and the basic configuration of the existing project **demo1** can be seen as shown below:

![Project Interpreter configuration in PyCharm](image)

Remember that the interpreter also includes the basic packages which are mandatory for smooth functioning of working of project.
Running a python code comprises of two modes: running a script and debugging the script. This chapter focusses on debugging the Python script using PyCharm.

**Steps Involved**

The steps for debugging the Python project are as explained below:

**Step 1**

Start with debugging the Python project as shown in the screenshot below:
Step 2
Now, Windows firewall asks permission for debugging the Python project as the procedure involves line by line compilation.

![Windows Security Alert]

Step 3
The debugging console is created in PyCharm editor as shown below which executes the output line by line.
The run button moves from one line to another to execute the output as the way we want.

**Understanding Breakpoints**

While debugging a particular script, it is intentional to create a breakpoint. Breakpoints are intentional stopping place or the place where the code is paused in order to identify the output at specific stage.
In PyCharm, breakpoints are visible using a separate dialog in the specified editor. It includes various attributes to evaluate the breakpoints defined and tracing log for the same with a main motive to achieve better programming practice.
13. PyCharm – Integration of Version Control

PyCharm supports various subversion control systems. This feature helps in improving the code base managing various versions together. This chapter talks about this concept in detail.

Steps Involved

You will have to go through the following steps for initializing and managing version control system:

Initializing a Subversion Control System

To start the version control system in a systematic way, it is important to initialize it. Various options are available in PyCharm for different version control systems.

![PyCharm VCS Options](image)

Ignoring File

In any project of PyCharm where we set up the default project and the virtual environment on it, we should also create its management with version control system. For example, Git includes .gitignore files which are ignored during commit operation, however, includes some of its configurations. Now, go to the Settings menu and check for the following:
It includes various configurations for checking the path of Git executable and verifying if any files are ignored.
**Configuration of GitHub**

PyCharm includes settings to include configuration of GitHub repository, where a user can include username, password and other credentials, if any.

Once you are done with the settings mentioned, you can directly add and commit the local changes to Git repository.
HTML and CSS are well supported in PyCharm Editor. PyCharm Editor includes a special shorthand and provides tag completion for HTML.

**Emmet**

Emmet is the shorthand used in PyCharm editor. It includes various features such as abbreviation preview, automatic URL recognition and edit points, for HTML and CSS files. The user interface of the settings section is shown in the screenshot given below:
Creating HTML and CSS files

PyCharm includes an inbuilt feature for creating HTML and CSS files. The basic steps for creating new HTML and CSS files are as follows:

Now, mention the name of file while creating HTML files in the project as shown below:

```
New HTML File
Name: sample-file
Kind: HTML 5 file
```
This creates the `sample-file.html` file as shown below:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
</body>
</html>
```

## Creating CSS file

The steps for creating a CSS file are shown here:

From the **New** menu, select the **File** option as shown below:

Specify the name of CSS during its creation as shown here:

```
Enter a new file name:
```

demo.css
You can see the complete project structure with various files mentioned in different color combinations as shown below:

![Project Structure](image)

- `demo1` directory
  - `venv` directory
    - `demo.css` file
  - `demo.py` file
  - `main.py` file
  - `multiprocessing.py` file
  - `sample-file.html` file
- `External Libraries` directory
- `Scratches and Consoles` directory
In this chapter, we will focus on main features in using JavaScript in PyCharm editor. When a user implements JavaScript library through URL, PyCharm intends to download a local copy so it can be used for completion and code analysis.

Consider the sample code of our HTML file as shown below, which we created in the previous chapter:

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
</head>
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```
For each HTML file or JavaScript file, you can check the external libraries loaded through Settings configuration of PyCharm Editor. Observe the screenshot shown below for a better understanding:

![Settings configuration of PyCharm Editor](image1)

Note that you cannot see any library unless you download and implement it. PyCharm also includes JavaScript support of various libraries through a toolbox called **JS Toolbox**. The following screenshot shows this.

![JS Toolbox](image2)
It also includes various attributes which are necessary for the JavaScript file configuration. The list of attributes and configurations is shown below:

Observe that it includes various parameters such as **Unit test suffix**, **File suffix**, **View suffix**, **Search URL** and the specific **Root directory**.
PyCharm includes various tips during startup that help its user to understand its functionalities and operations. It also includes some shortcuts which are mandatory to understand.

In this chapter, you will see some of the important PyCharm tips.

**Changing the File to a Specific Changelist**

This tip shows how to change the file to a specific changelist as per the user’s choice. This helps in managing repositories as per version control system settings. Observe the following screenshot for a better understanding:

![Image of PyCharm tip](image-url)
Display the List of all Usages in a Class

This function displays the list of all usages included in a specific class, method or variable across the project. It quickly enables the user to jump to specific area. Observe the following screenshot for a better understanding:

To find Menu Command for an Action

This tip helps to find menu command for a specific action and the shortcut key for the same is Ctrl+Shift+A. A user can select desired action from the mentioned suggestion list.
Running Inspection through a Code

This tip helps in running a specific inspection through the code. The shortcut key combination for the same is **Ctrl+Alt+Shift+I**.

Specify the List of Settings

This tip is used to specify the list of desired settings. It includes smart keys for specific editor. The smart keys are shortcut keys for some operations.
Run / Debug the Script Files

This tip is very useful for running or debugging the script files which you can access through main toolbar. The shortcut key combination for same is **Alt+Shift+F10**.

By pressing **Alt+Shift+F10** you can access the Run/Debug dropdown on the main toolbar, without the need to use your mouse.
PyCharm supports interface support with various types of databases. Once a user grants access to the created database, it provides schema diagram of the database with SQL writing tools which provide code completion. In this chapter, we will focus on MySQL database connectivity which will involve following steps.

Adding a Data Source

It is important to keep a note that PyCharm supports a wide variety of database connectivity.

Step 1

Open the database tool window View -> Tool Windows -> Database and open the dialog called Data Sources and Dialog.

Now, select MySQL database for adding a new data source.

Step 2

User should download the missing driver files to get proper connectivity with MySQL database.
Step 3
Now, specify the configuration settings for connectivity to be achieved.

**Host:** If your database server is on a different computer, replace localhost with the IP address of the server host, e.g. 172.20.240.163.

**Port:** The default MySQL server port is 3306. If your server uses a different port, specify that port.

**User and Password:** These are the required credentials.

Step 4
Always make sure that database connectivity is successful through **Test Connection** feature.
Testing the connection also involves creating test table through query and executing them. Once the execution is successful, you can drop the database.
PyCharm IDE includes various features for converting the existing code file into HTML format or CSV format. In this chapter, you will learn exporting data using PyCharm IDE.

The export settings of PyCharm editor are shown in the figure given below:

Export to HTML feature

This feature helps in exporting the specific file in HTML format. This is done to improve the security purposes of the given module. The following screenshot gives a better understanding:
Once the export operation is successful, the generated HTML file will display in browser output as shown below:

Now, if you check the HTML code generated after the export operation, you can observe that line numbers are also included to achieve this operation.
This chapter focusses on web frameworks and its deployment. PyCharm has a simple functionality to deploy code and files. To deploy code with PyCharm, we need to add a web server with Menu Option Settings -> Build, Execution -> Deployment.

Now, include all the settings with various configurations required for deployment of the project.
In the **Mappings** tab, a user can specify where the local code is and where it should be copied to remotely.

![Mappings tab in PyCharm](image1.png)

The code can be deployed using **Tools -> Deployment** option under the Tools menu bar. Deployment in PyCharm is very granular: a user can deploy one single file or the whole source code.

![Deployment options in PyCharm](image2.png)

PyCharm also includes various actions to compare remote and local versions. The editor is more reliable to use automatic deployments and a version control system to compare local and remote versions.
One of the features of PyCharm is that it includes a support for Django. With the ability of including JavaScript features within PyCharm, it can be considered as the best IDE for Django.

The basic steps for creating a Django project in PyCharm IDE are given below:
If the **EnableDjangoAdmin** option is enabled, PyCharm will setup the admin site for you.

**Template Debugging**
Debugging works with Django and Jinja templates. We can inspect variables, step through code, and do what we expect in a debugger.
You can create a project of **Pyramid Framework** in PyCharm editor by using its Welcome Window.

A user can set the project’s interpreter and Python location, choosing scaffold, and a template language by default. The scaffold in Pyramid framework uses URL dispatch to map URLs and to view code and SQLAlchemy for persistence properties.

PyCharm editor will ask user with list of required packages in the `setup.py` file and prompts the user to download the required packages.

Install the project in development mode (refer to official documentation of Pyramid for more details). The user should run python `setup.py` through the menu **Tools -> Run setup.py** option.
A user should select **develop task** while running a .py file as mentioned in the below window:

It is important to initialize the database using console script named `initialize <project_name>` using the following command:

```
initialize_pyramid<project_name>_db development.ini
```

The user can start the server by running the project which will display the result as shown below:
PyCharm supports Flask framework development. You can easily create a new Flask project by creating new project through welcome screen. You can set the project’s location and the virtual environment and choose a template language and where the templates will be located.

You can run a project by using Run -> Run ‘<project-name>’.

You can also add a new data source with this framework. Let us create a file called `squema.sql` and add SQL code to create some tables. PyCharm editor will recognize the files and asks you to configure a data source and set up to the database dialect.

PyCharm will let you choose the desired dialect that you want to use. You can change the properties of the SQL: Settings -> Language and Frameworks -> SQL Dialects
For flask editor, the easiest way to run the SQL query is to click somewhere in the query and click on the inspection window and click "Run Query into console".

The user interface of the Flask framework is displayed as below: