

Project – 3

Efficient Development with PyCharm: Mastering the PyCharm IDE



Submitted By:
Nishigandha Patil

INDEX

Sr. No	Topics	Page. No
1	Introduction to PyCharm	3
2	User Interface	5
3	Basic Python development	7
4	Code Management and Version Control	11
5	Debugging and Problem Solving	14
6	Advanced Python Development	16

INTRODUCTION TO PYCHARM

PyCharm is a popular integrated development environment (IDE) specifically designed for Python development. It provides features such as code editing, debugging, intelligent code completion, syntax highlighting, project navigation, version control, and support for various Python frameworks such as Django, Flask, and others. Alongside Python, it offers support for web development technologies like HTML, CSS, and JavaScript, enabling full-stack development within the same environment.

PyCharm is available in two editions:

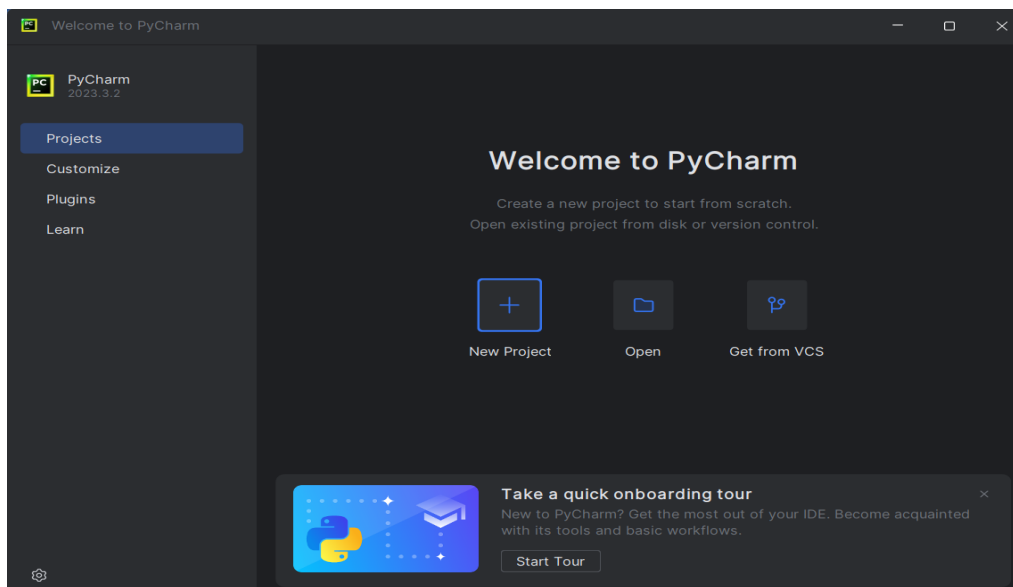
- **Community (free and open-sourced):** free and offers many powerful features for Python development.
- **Professional (paid):** for professional Python, web, and data science development. Supports popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.

1. Setting up and configuring PyCharm:

Downloading & Installing Python: Go to <https://www.python.org/downloads/> → Download Latest Version → Run the installer.

Downloading & Installing PyCharm: Go to <https://www.jetbrains.com/pycharm/download/> (Community Edition) → Click on Download → Run the installer → Launch.

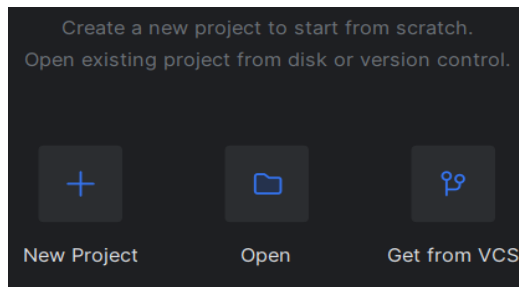
➤ Here's what you'll see when you open PyCharm:



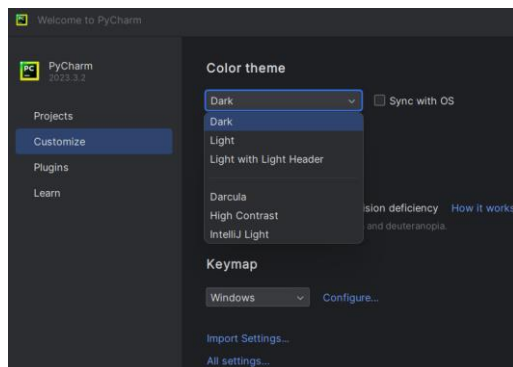
2. Familiarizing with the PyCharm:

A. Projects: You have three options to start working on a project:

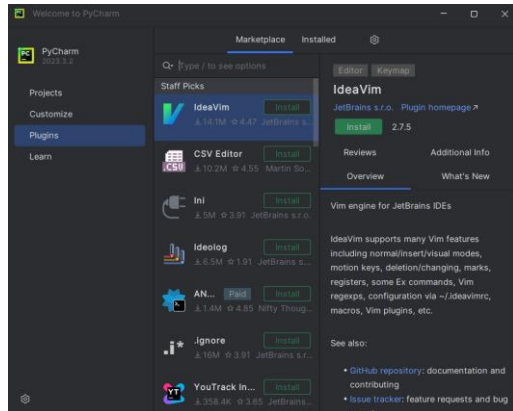
1. Create a new project
2. Open an existing project
3. Check out a project from version control



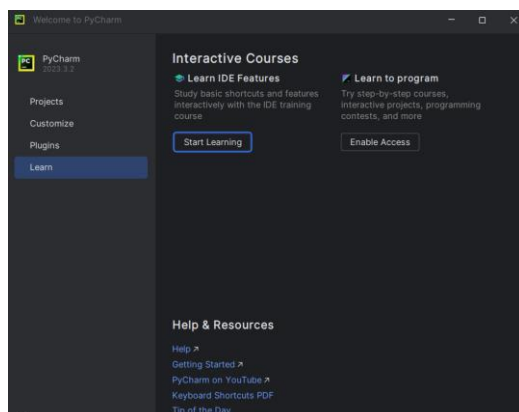
- B. Customize:** Click Customize and select another color theme or select the Sync with OS checkbox to use your system default theme. Here you can also configure accessibility settings or select another keymap.



- C. Plugins:** Click Plugins in the left pane and download and install additional plugins from JetBrains Marketplace.

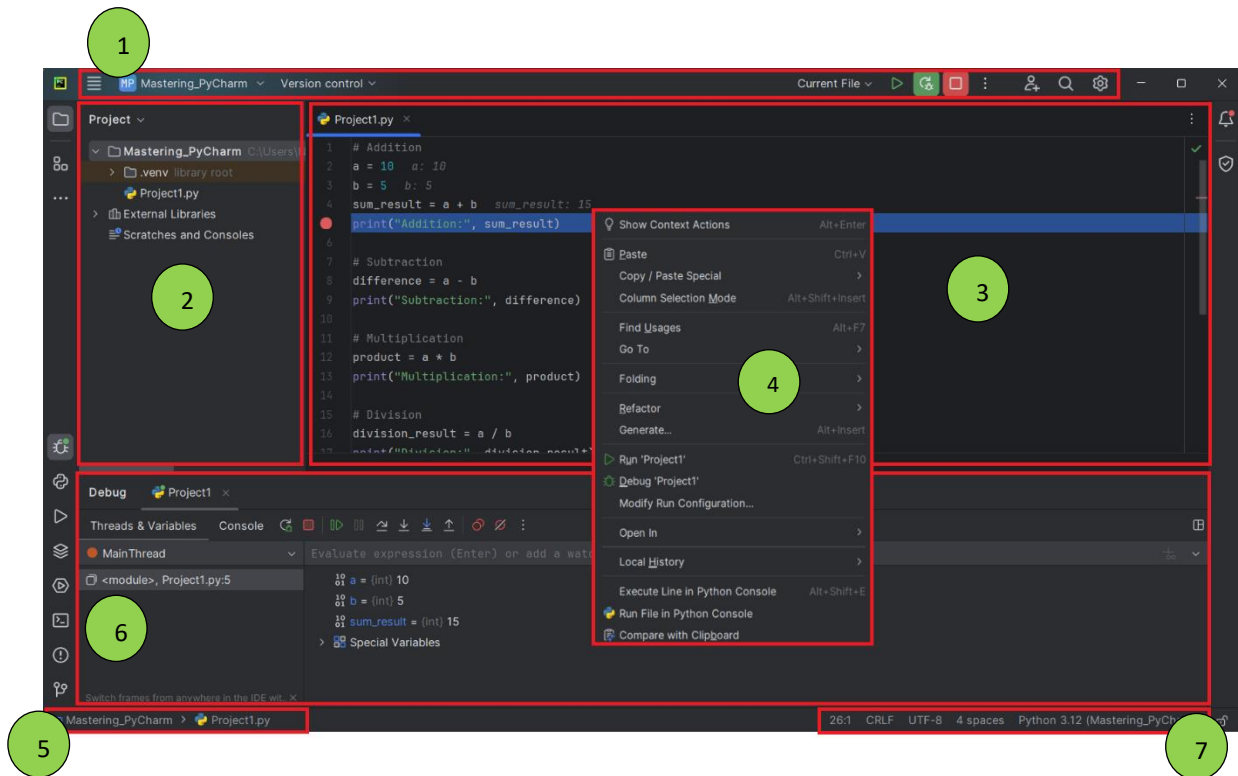


- D. Learn:** Training courses and resources to learn Features and shortcuts of IDE.



USER INTERFACE

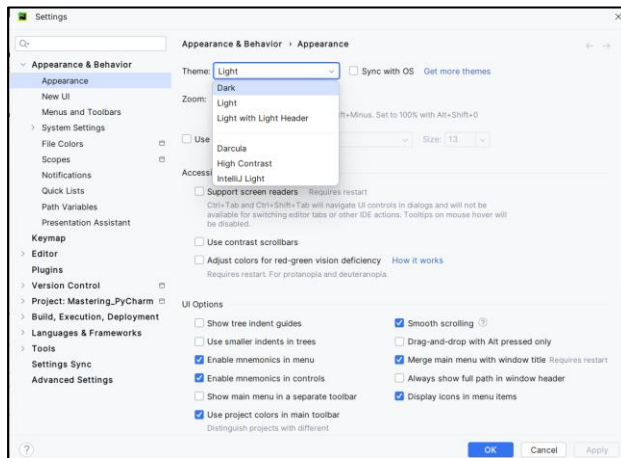
1. **UI:** PyCharm's user interface (UI) consists of several key components:



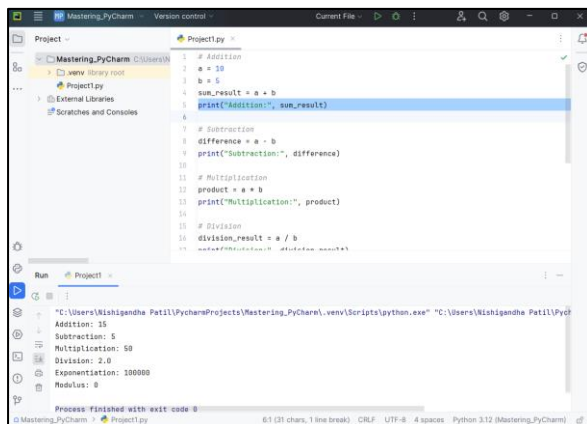
1. **Window header:** Contains various menus like File, Edit, View, Navigate, Code, Run, and more, offering access to different functionalities and options.
2. **Project Explorer:** Window on the left side displays your project folders and files.
3. **Editor:** The main area where you write your code.
4. **Context menus:** Open when you right-click an element of the interface or a code fragment and show the actions available.
5. **Navigation bar:** Allows you to quickly navigate the project folders and files.
6. **Tool windows:** Areas within the IDE that can be opened, closed, or rearranged based on your needs. Examples include the terminal, version control, database explorer, and more. They provide access to typical tasks such as project management, source code search and navigation, integration with version control systems, running, testing, debugging, and so on.
7. **Status bar:** Located at the bottom, it shows various warnings and information messages like file encoding, line separator, inspection profile, and so on. It also provides quick access to the Python interpreter settings.

2. Customizing the UI appearance:

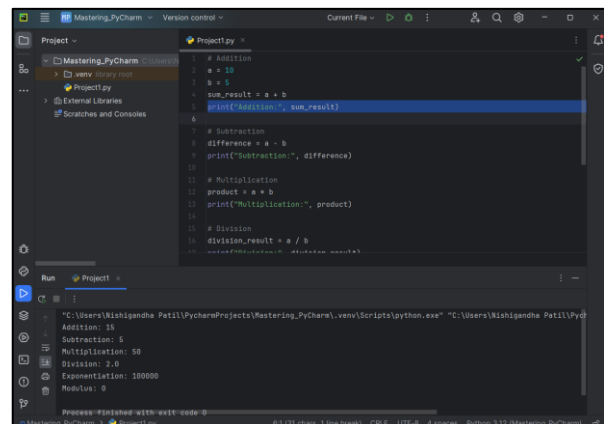
Settings (Ctrl + Alt + S) → Appearance → Choose Theme → Ok.



Light theme:

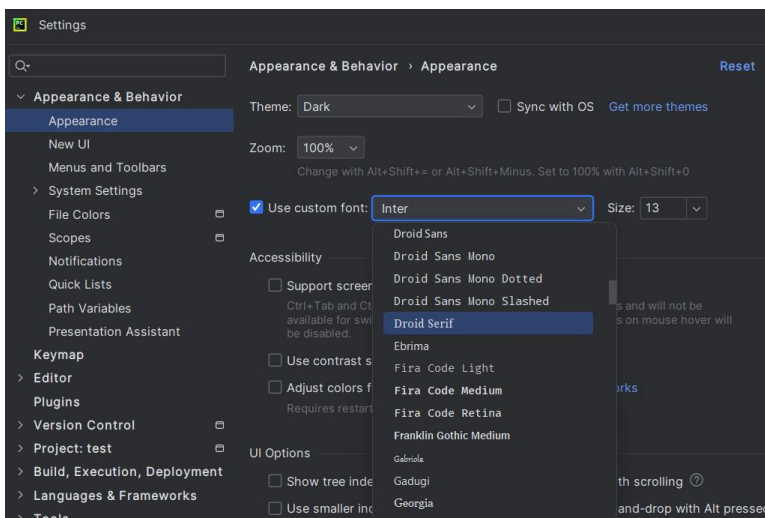


Dark Theme:



Customizing Font and size:

Settings → Appearance → Use custom font → Select Font → Select Size

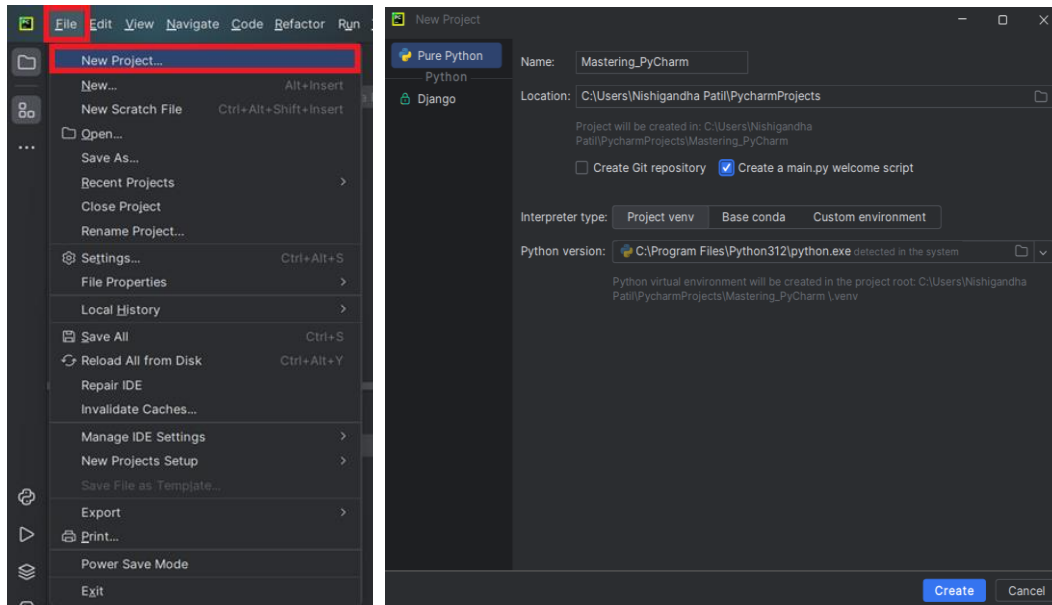


BASIC PYTHON DEVELOPMENT

1. Creating and managing Python projects:

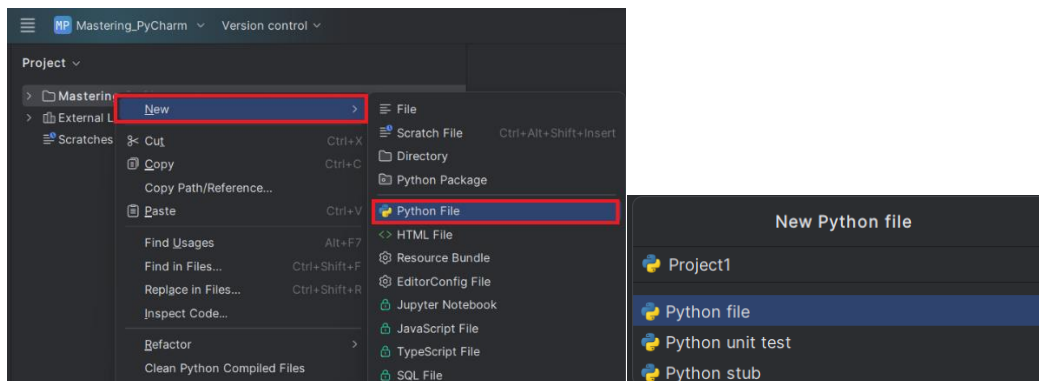
Creating New Project:

Click on File → New Project → Add project name → Create.



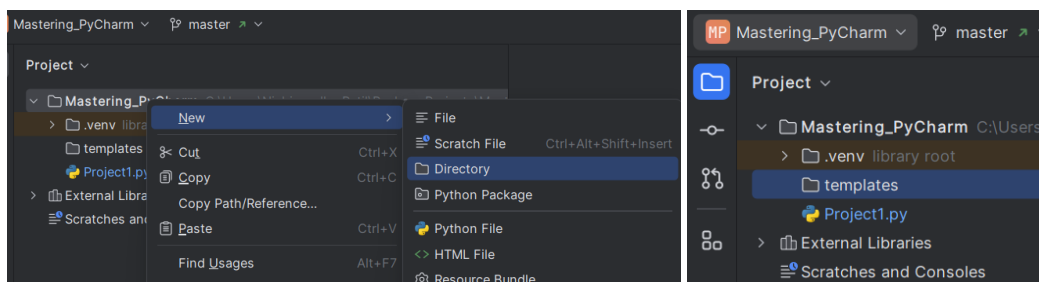
Creating new file within the project:

Right click on Project → New → Select Python File → Give name to the file.



Creating new folder within the project:

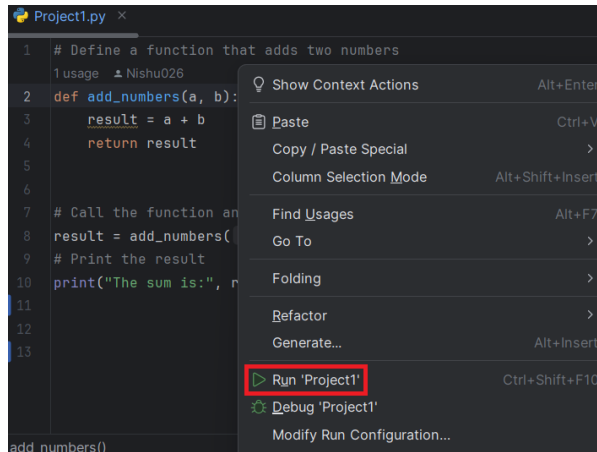
Right click on Project → New → Directory → Give directory name (e.g. templates).



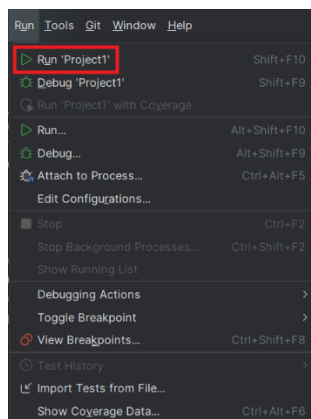
2. Running Python Code:

There are different ways to run the code in PyCharm:

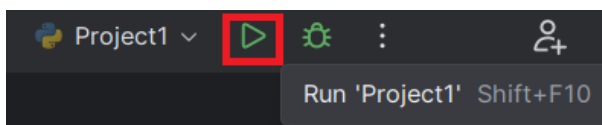
- A. Right-click an open file in the editor and choose Run <filename> from the context menu.



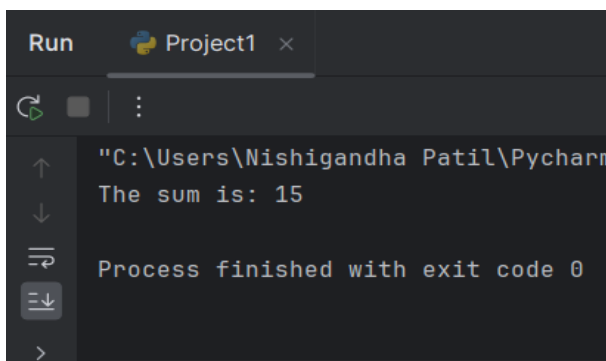
- B. Go to Run → Run <filename>



- C. In the Run widget, select Current File and click Run.

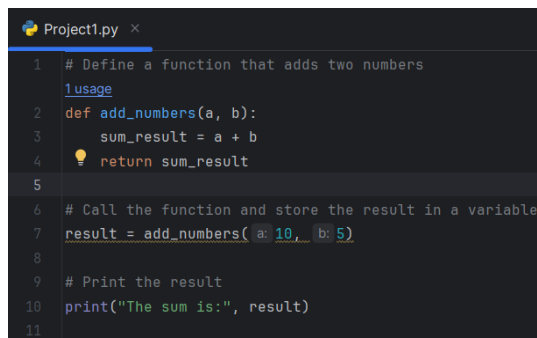


Checking the Output:



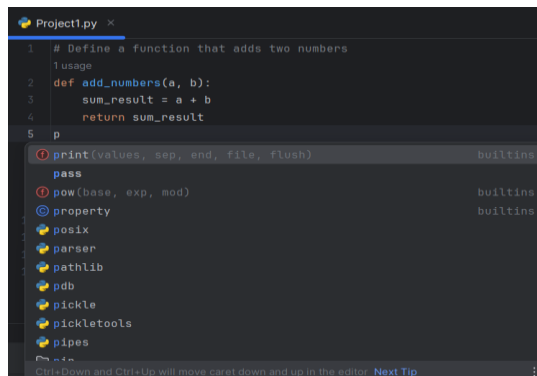
3. Features of Editor:

- A. Syntax Highlighting:** PyCharm colorizes different parts of your code to make it more readable. Keywords, strings, comments, and other elements are displayed in different colors, aiding in quick comprehension.



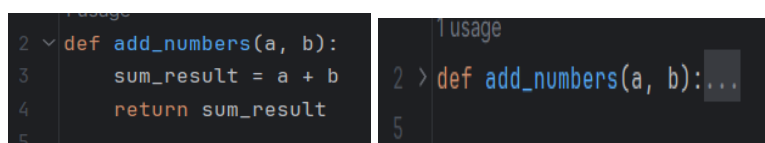
```
Project1.py x
1 # Define a function that adds two numbers
2 usage
3 def add_numbers(a, b):
4     sum_result = a + b
5     return sum_result
6
7 # Call the function and store the result in a variable
8 result = add_numbers(a=10, b=5)
9
10 # Print the result
11 print("The sum is:", result)
```

- B. Code Completion:** PyCharm offers intelligent code completion, suggesting variable names, method names, and even entire code snippets as you type. Press TAB to accept the suggestion.



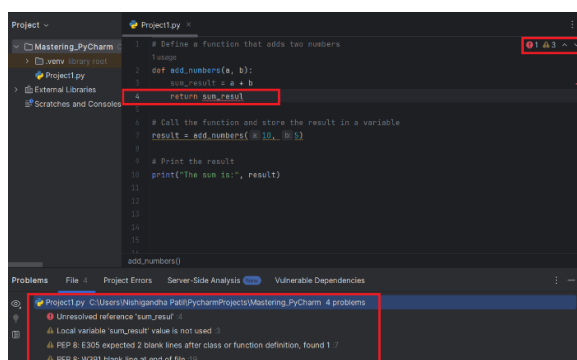
```
Project1.py x
1 # Define a function that adds two numbers
2 usage
3 def add_numbers(a, b):
4     sum_result = a + b
5     return sum_result
6
7 p
8 print(values, sep, end, file, flush) builtins
9 pass
10 pow(base, exp, mod) builtins
11 property builtins
12 posix
13 parser
14 pathlib
15 pdb
16 pickle
17 pickletools
18 pipes
19 sys
20 Ctrl+Down and Ctrl+Up will move caret down and up in the editor. Next Tip
```

- C. Code Folding:** By clicking on arrow, you can collapse sections of code (such as functions or loops) to hide them temporarily, making it easier to focus on specific parts of your code.



The left screenshot shows a function definition with a small arrow icon next to the first line of the function body. The right screenshot shows the same function definition with the first line of the function body collapsed, indicated by a small arrow icon next to the function name.

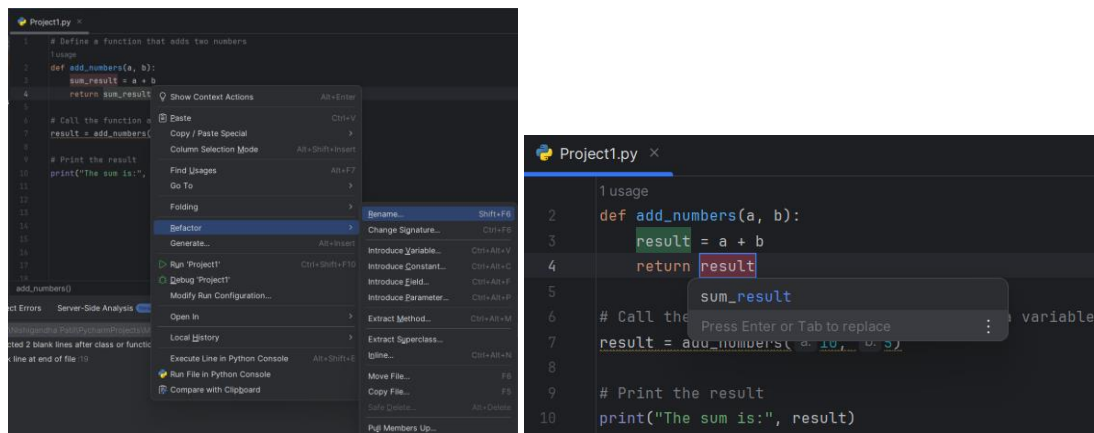
- D. Error Highlighting and Suggestions:** PyCharm highlights potential errors or warnings in your code in real-time, helping you identify and fix issues as you write.



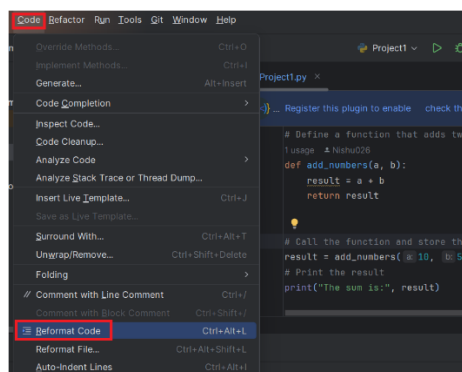
The screenshot shows the PyCharm editor with a Python file. The function definition is highlighted in red, indicating an error. The Problems panel at the bottom shows the following errors:

- Project1.py: C:\Users\Nishigandha Pali\PycharmProjects\Mastering_PyCharm: 4 problems
- Unresolved reference 'sum_result'.
- Local variable 'sum_result' value is not used.
- PEP 8: E305 expected 2 blank lines after class or function definition, found 1.
- PEP 8: W391 blank line at end of file.

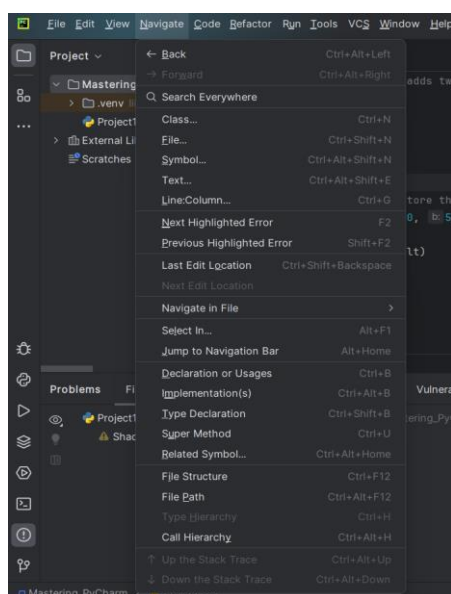
- E. Refactoring Tools:** PyCharm provides tools to easily rename variables, methods, classes, or even entire packages across your codebase. Right click on editor→Refactor→Rename.



- F. Code Formatting:** It supports automatic code formatting according to defined style guidelines, making your code look neat and consistent. Press (Ctrl + Alt + L) or go to “Code” in menu bar→Reformat.



- G. Quick Navigation:** You can quickly jump to a file, class, method, or symbol by using shortcuts like Ctrl+Click or by using the Navigate menu. This helps in moving around large projects or exploring unfamiliar codebases efficiently.

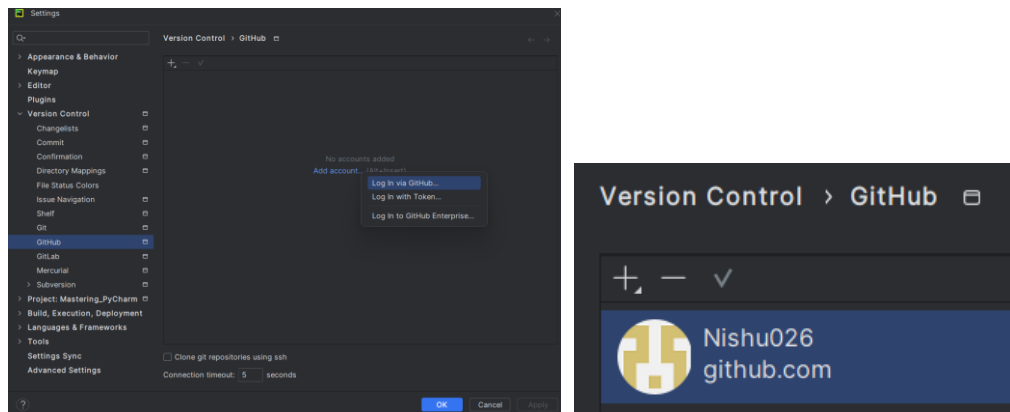


CODE MANAGEMENT AND VERSION CONTROL

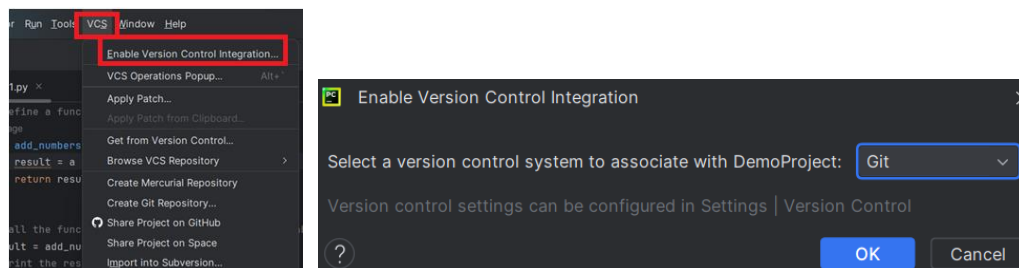
By using version control systems like Git within PyCharm, developers can manage code changes effectively, collaborate seamlessly, track project history, and revert to previous versions when needed, ensuring a more organized and efficient development workflow.

Integrating with Git:

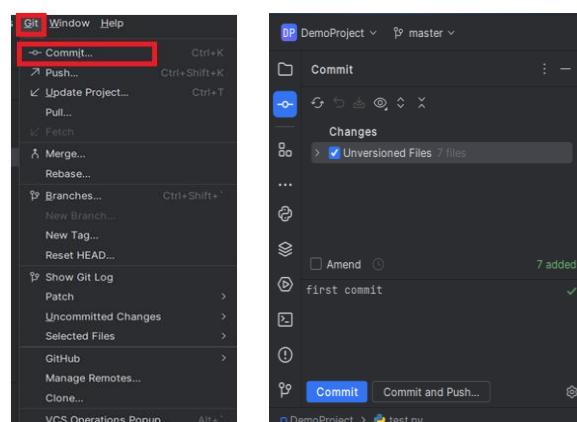
1. **Login GitHub Account:** Go to Settings → Version Control → GitHub → Add account → Login via GitHub (Authentication Process will start) → Apply.



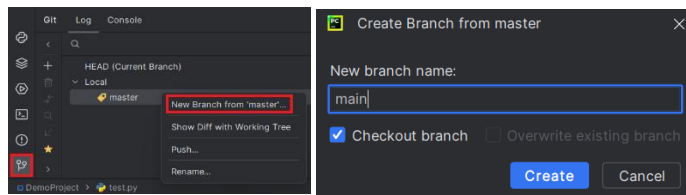
2. **Enabling VCS:** Go to VCS → Enable Version Control Integration → Select Git → Ok



3. **Commit changes:** Go to Git → Commit → Check “Unversioned Files” checkbox → Add Commit message → Commit

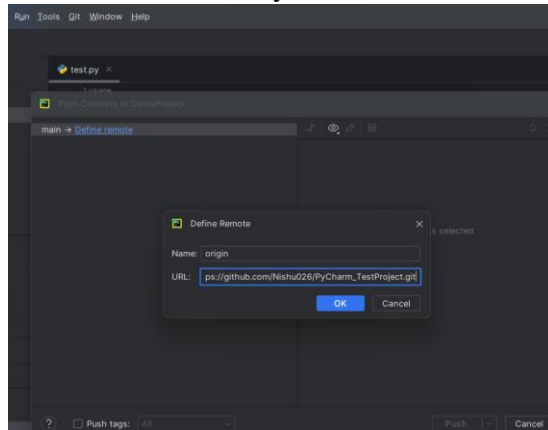


4. **Creating Branch:** Click on Git icon → Right Click on Master → New Branch → Branch name → Create.

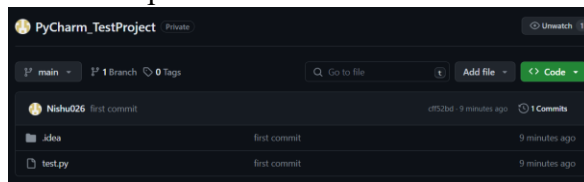


5. Push Code on GitHub:

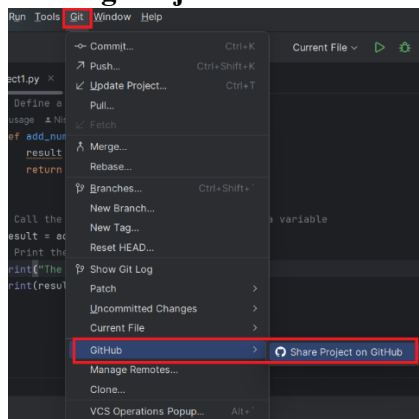
- Create GitHub Repo → Copy the URL.
- Go to “Git” in PyCharm → Push → Define remote → Paste the URL → Ok → Push.



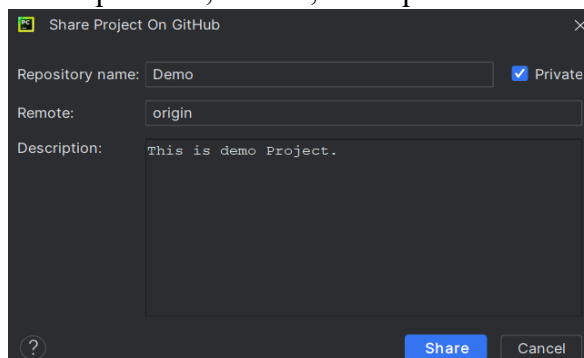
GitHub Repo:

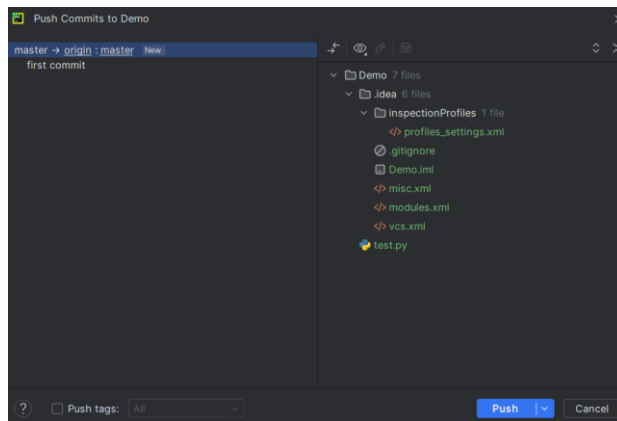


6. Sharing Project on GitHub: Enable VCS → Commit Changes → Git → GitHub → Share

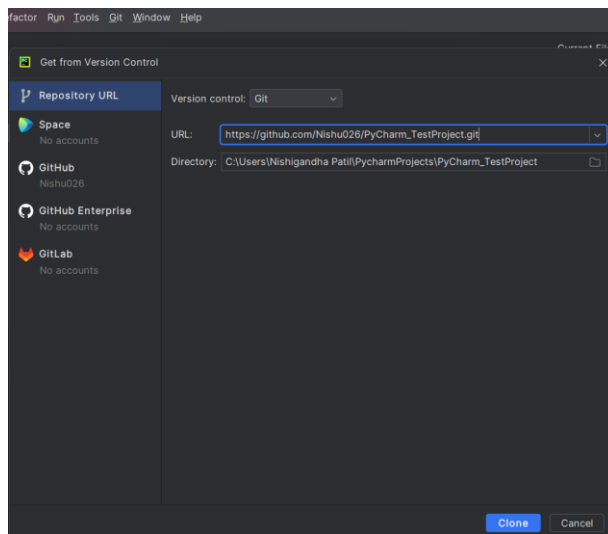


Add repo name, remote, description → Share → Push.

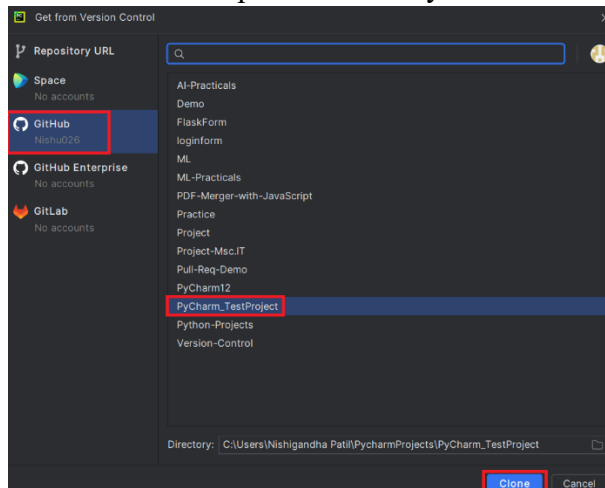




7. **Clone repository:** Git → Clone → Add URL of GitHub repository that you want to clone → Clone.



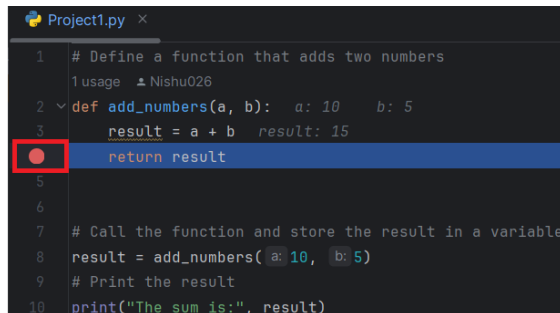
If you are already logged in to the selected hosting service, completion will suggest the list of available repositories that you can clone.



DEBUGGING AND PROBLEM SOLVING

PyCharm's debugger is a powerful tool for identifying and fixing bugs in your Python code.

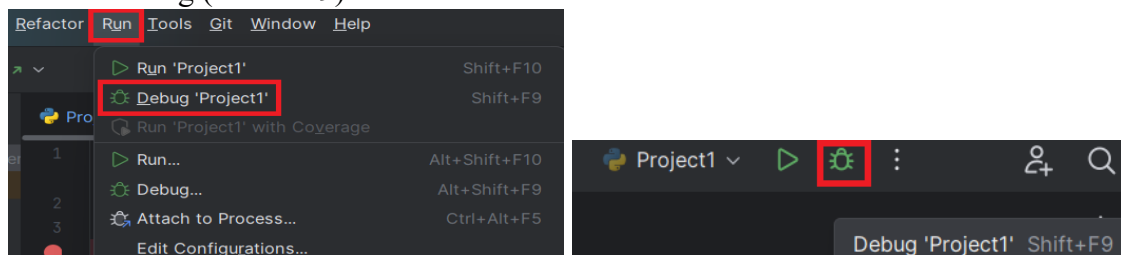
1. **Setting Breakpoints:** Click the gutter of the editor where you want to set the breakpoint. Alternatively, place the caret at the line and press Ctrl + F8.



You can remove, mute, enable, disable, move, copy or edit the breakpoints.

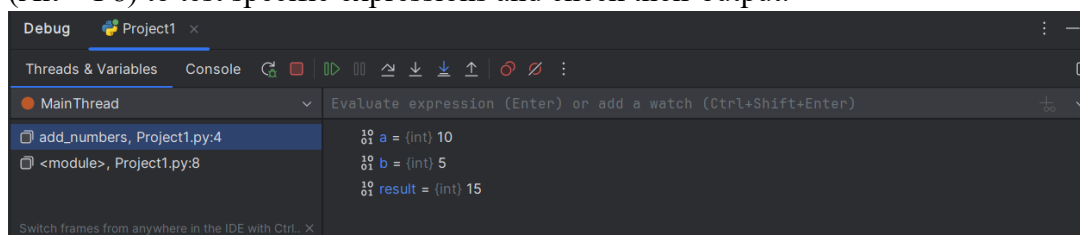
2. **Running in Debug Mode:** This mode allows you to execute your code step-by-step, pausing at breakpoints.

Run → Debug (Shift + F9)

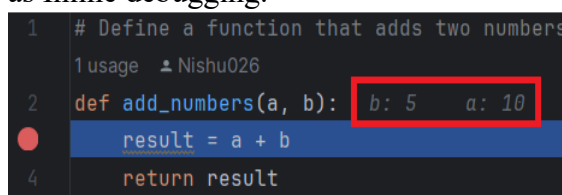


3. **Watching Variables & Evaluating Expressions:**

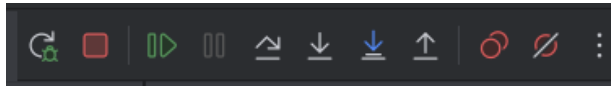
While debugging, you can add variables to the watch list. You evaluate expressions in the context of your code while debugging. You can open the "Evaluate Expression" window (Alt + F8) to test specific expressions and check their output.



4. **Inline Debugging:** In the editor, you can see text in italics next to the lines of code called as Inline debugging.

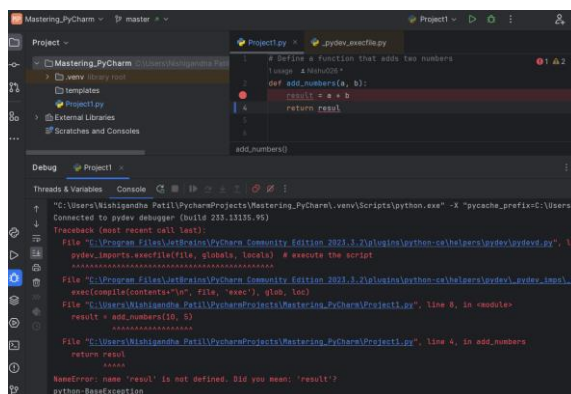


5. Debug Actions:



Sr. No	Action	Explanation
1	Rerun (Ctrl+F5)	Rerun the current Project.
2	Stop (Ctrl+F2)	Terminates the debugging process.
3	Resume (F9)	Resumes normal program execution until the next breakpoint or until the program completes.
4	Pause	Pause the execution of code.
5	Step Over (F8)	Executes the current line of code and moves to the next line.
6	Step Into (F7)	Steps into the method called on the current line, navigating into the method's code. If there is no method call, it behaves like "Step Over."
7	Step Into my Code (Alt + Shift +F7)	Steps Into entire code.
8	Step Out (Shift + F8)	Steps out of the current method and continues execution until it returns from the current method, then stops at the next line.
9	View Breakpoints	Provide the list of breakpoints set in your project. You can manage them.
10	Mute Breakpoints	Allows you to "mute" or deactivate specific breakpoints temporarily without removing them completely.

6. **Understanding Error Message:** Examine the error message displayed in the console or debugger panel. This message often provides information about the type of error, the line number, and sometimes a description of the issue.



7. Fix the Issue:

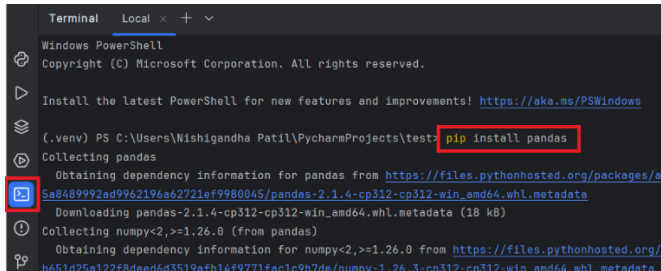
Based on your observations and analysis during debugging, make necessary changes to your code to fix the error.

ADVANCED PYTHON DEVELOPMENT

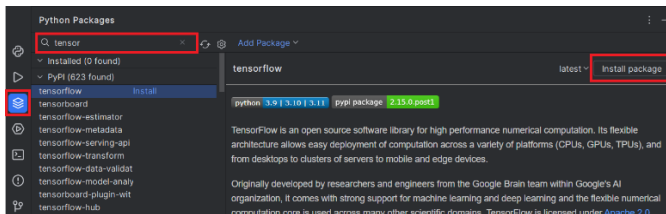
1. Installing python packages:

PyCharm allows you to install Python packages using various methods:

Using Terminal: You can use the integrated terminal within PyCharm to run pip install commands to install packages. Syntax: pip install package_name



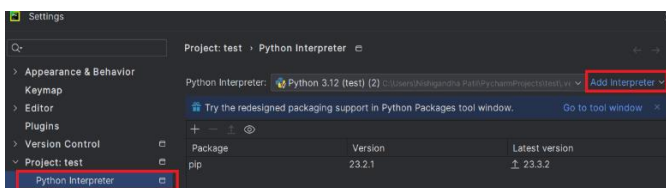
Using GUI: Go to python packages (bottom-left) → Search for package → Install package.



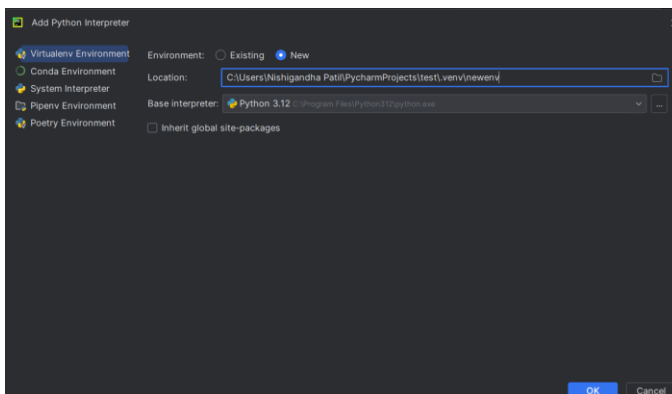
You can update packages to newer versions or remove unwanted packages from your project. PyCharm helps manage package dependencies within version control systems. It allows you to share project settings including package dependencies among team members.

2. Creating Virtual Environments:

Go to settings → Python interpreter (under Project section) → Add Interpreter



Select the type of virtual environment: Virtualenv, Pipenv, or Conda → Add location → Ok

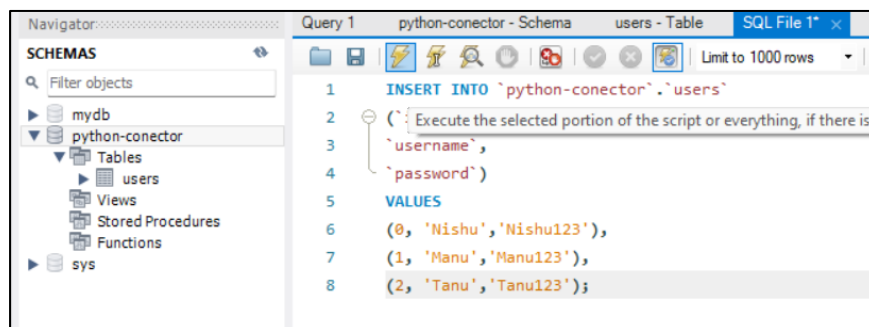


Once virtual environment is created it is automatically activated. You can deactivate the virtual environment using “deactivate” command.

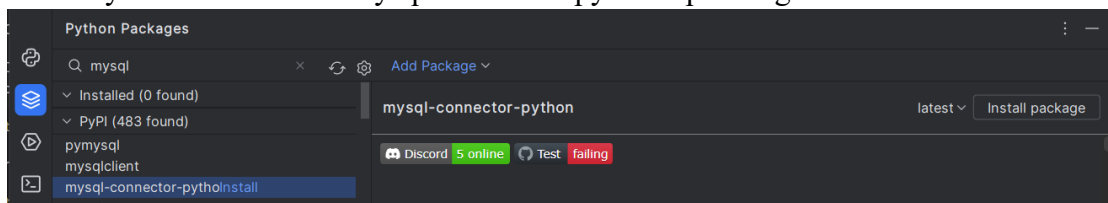
```
(newenv) PS C:\Users\Nishigandha Patil\PycharmProjects\test> deactivate
PS C:\Users\Nishigandha Patil\PycharmProjects\test> 
```

3. Working with Database (SQL):

Create Schema in SQL (python-conector) → Create table (users) → Execute.



Go to PyCharm → Install “mysql-connector-python” package.



Create python file and type the following sql-connector code:

```
1 import mysql.connector
2 mydb=mysql.connector.connect(
3     host='localhost',
4     user='root',
5     password='Nishupatil@26',
6     port='3306',
7     database_='python-conector'
8 )
9
10 mycursor= mydb.cursor()
11 mycursor.execute('SELECT * FROM users')
12
13 users= mycursor.fetchall()
14
15 for user in users:
16     print(user)
17     print('Username ' + user[1])
18     print('Password ' + user[2])
```

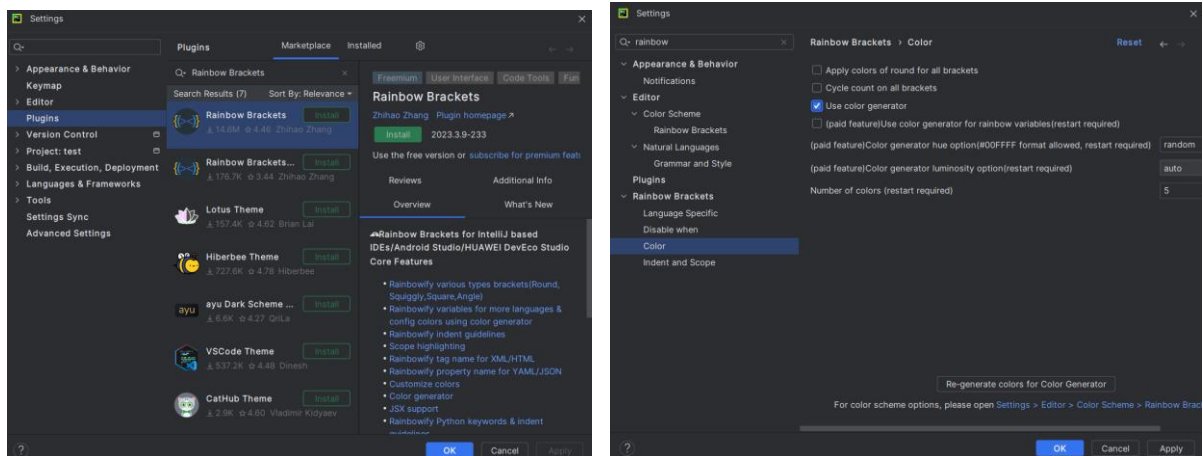
Python sql-connector Code

```
Run sql-connector
"C:\Users\Nishigandha Patil\Pycharm
(0, 'Nishu', 'Nishu123')
Username Nishu
Password Nishu123
(1, 'Manu', 'Manu123')
Username Manu
Password Manu123
(2, 'Tanu', 'Tanu123')
Username Tanu
Password Tanu123
Process finished with exit code 0
```

SQL database in PyCharm

4. Customizing the IDE with plugins and themes:

Settings → Plugins → install plugin (Rainbow Brackets) → Restart IDE → Change the settings of Rainbow Brackets → Ok



Write or open a Python file with nested brackets (parentheses, braces, brackets).

As you type or navigate through the code, you'll notice that each pair of matching brackets is colored distinctly, making it easier to identify corresponding brackets.

```
a = {1: "nishu"}
b = [1, 2, 3, 4]
print(b)
```

➤ Conclusion:

PyCharm stands out as a robust integrated development environment (IDE) designed specifically for Python. Its feature-rich platform offers a wealth of tools aimed at enhancing productivity and code quality. With a user-friendly interface, PyCharm caters to developers of varying expertise levels, providing intuitive navigation and a plethora of functionalities, including code completion, syntax highlighting, and code analysis. The built-in debugger empowers developers to efficiently identify and troubleshoot errors by setting breakpoints, inspecting variable values, and stepping through code. PyCharm also simplifies package management and virtual environment handling, ensuring smooth dependency management within projects. Its seamless integration with version control systems like Git facilitates collaborative coding and effective project versioning. Additionally, PyCharm's extensibility allows for customization through plugins, themes, and configurations, while its support for remote development further expands its utility for working on projects stored on remote servers or machines.