1-Practice

Theme: Introduction: Installing and configuring the programming environment

Objective of the task: Learning the basics of Python programming, installing and setting it up, and getting familiar with Python syntax.

Installing Python Environment

1. Download Python

Go to the official Python website: python.org. From the **Downloads** section, choose the version for your operating system (Windows, macOS, Linux). Run the downloaded file to start the installation.

2. Install Python

1. During installation, **check "Add Python to PATH"**. This allows you to use Python easily from the terminal or command prompt.

2. Click the **"Install Now"** button to begin the installation.

3. When the installation is complete, click **"Close"**.

PyCharm

PyCharm is a powerful and user-friendly Integrated Development Environment (IDE) designed for programming in Python. It is developed by JetBrains. There are two versions of PyCharm: Professional and Community.The Professional version is paid and supports web development and working with databases. The Community version is free and provides enough features for creating basic Python programs. The program can be downloaded from the official website: https://www.jetbrains.com/pycharm/

After opening the website in a browser, click the "Download" button and select the version you need in the pop-up window (see Figure 1.1).

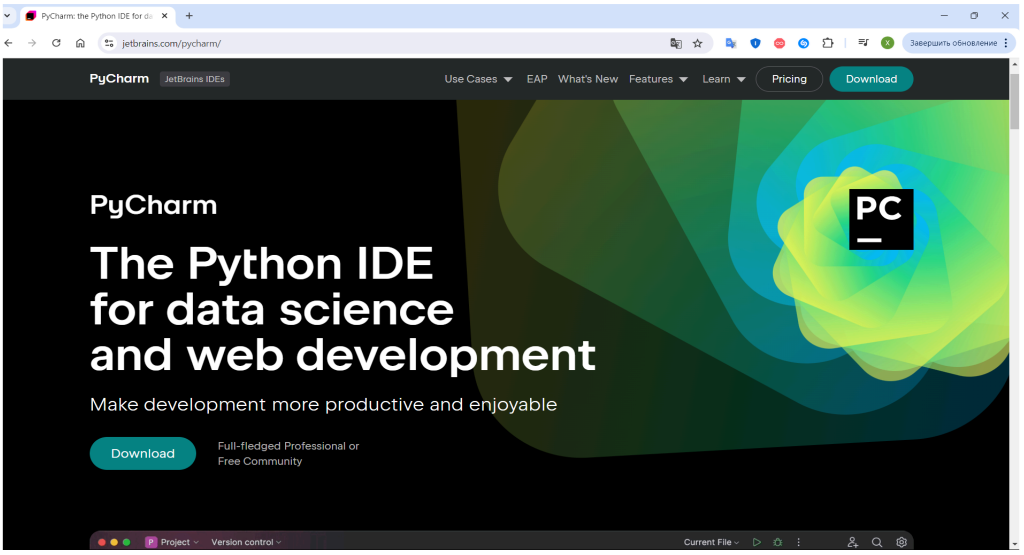

Figure 1.1: Downloading PyCharm

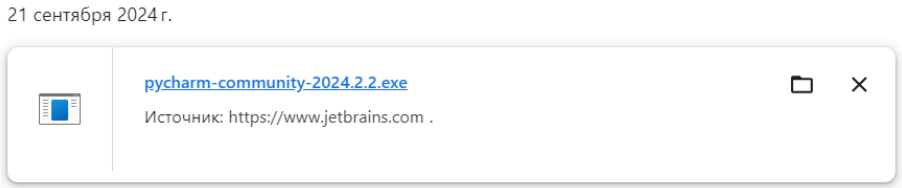

Figure 1.2: PyCharm has been downloaded

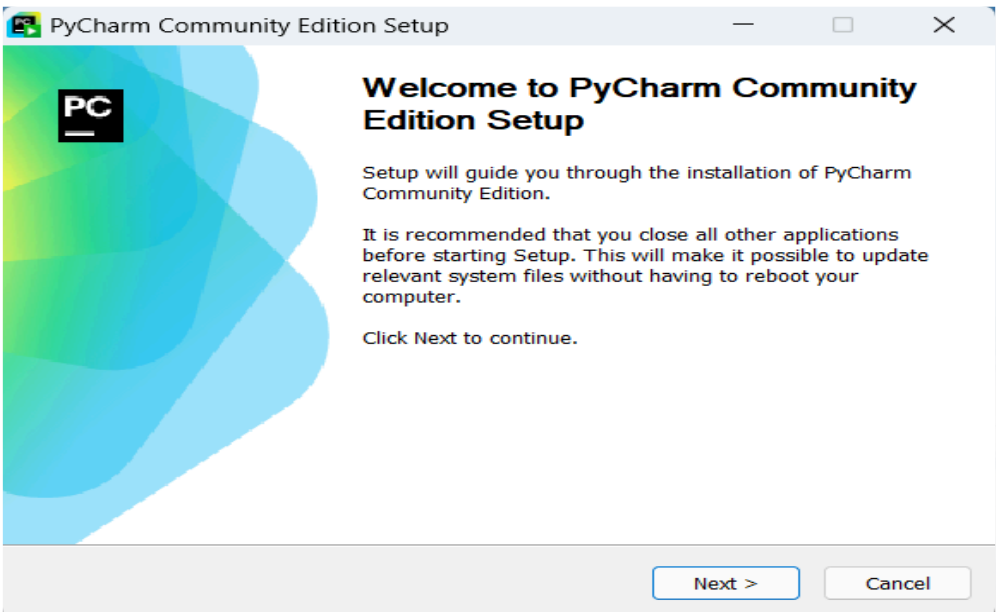

Figure 1.3: Installing PyCharm

After downloading the file, double-click it to start the installation process. In the installation window, the user is asked to choose a few settings.

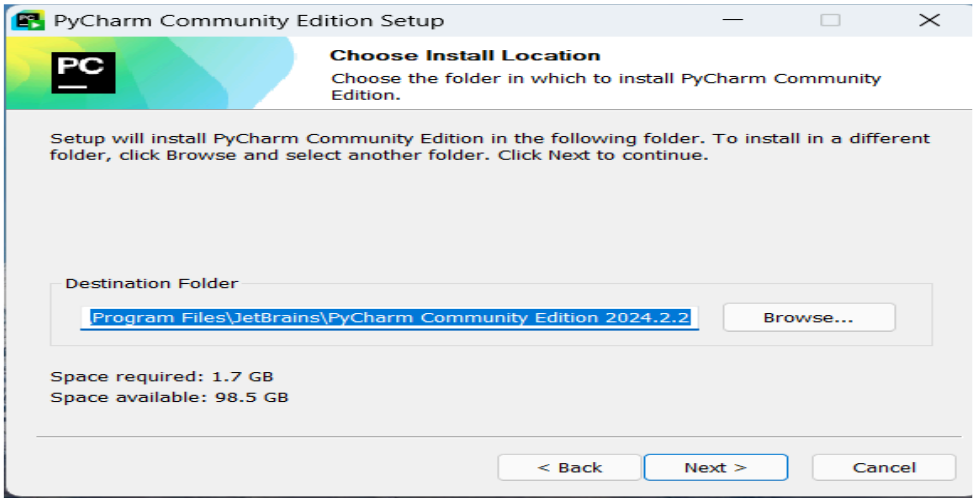

Figure 1.4: Disk space required for installing PyCharm

In the window shown in **Figure 1.3**, the user clicks the **"Next"** button. In the next window, the user selects the **folder where PyCharm will be installed** (Figure 1.4). By default, this is: C:\Program Files\JetBrains\PyCharm Community Edition. In the following window, the user is asked to choose **additional options**, such as: **"Add launchers dir to the PATH"** – allows running PyCharm from the command line. **"Create associations .py"** – automatically links Python files with PyCharm. **"Add desktop shortcut"** – creates a desktop shortcut. Select the desired options and click **"Next"** (see Figure 1.5).

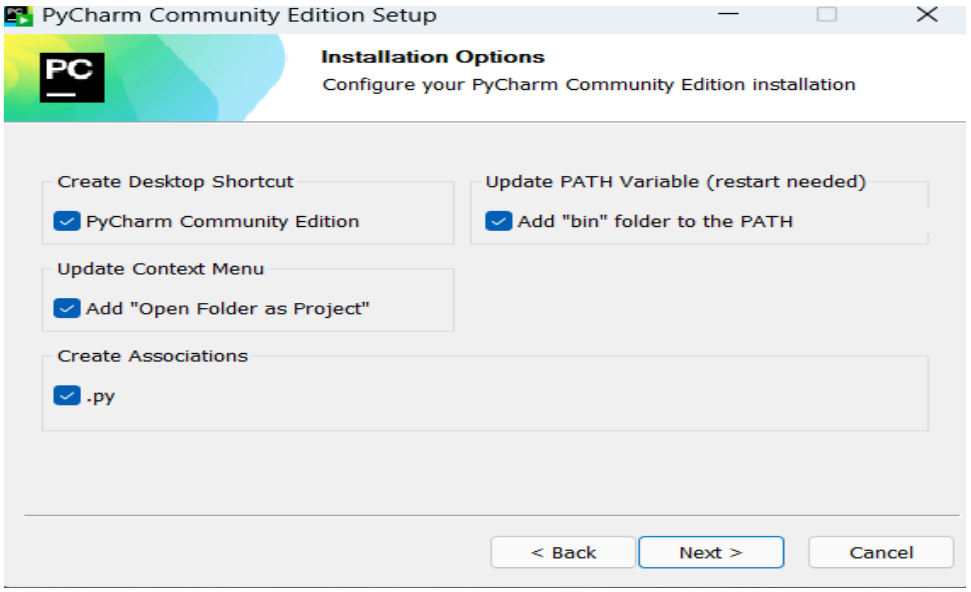

Figure 1.5: Configuration options for installing PyCharm

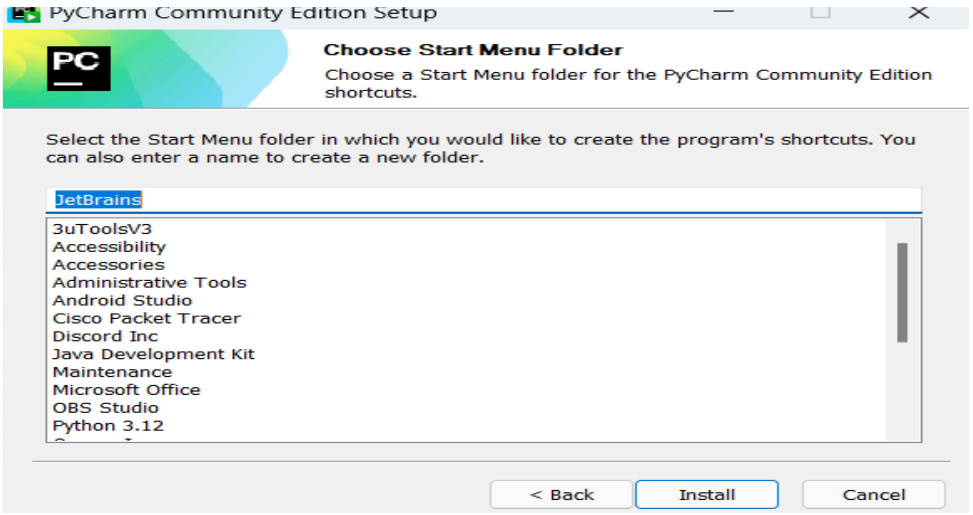

Figure 1.6: Selecting the Start Menu folder when creating program shortcuts

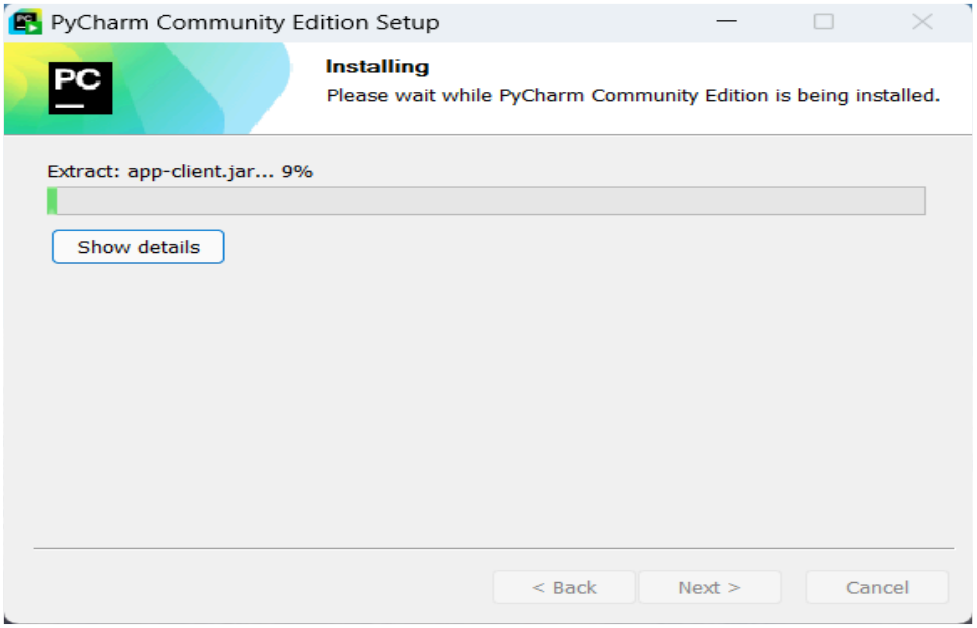

Figure 1.7: PyCharm installation in progress

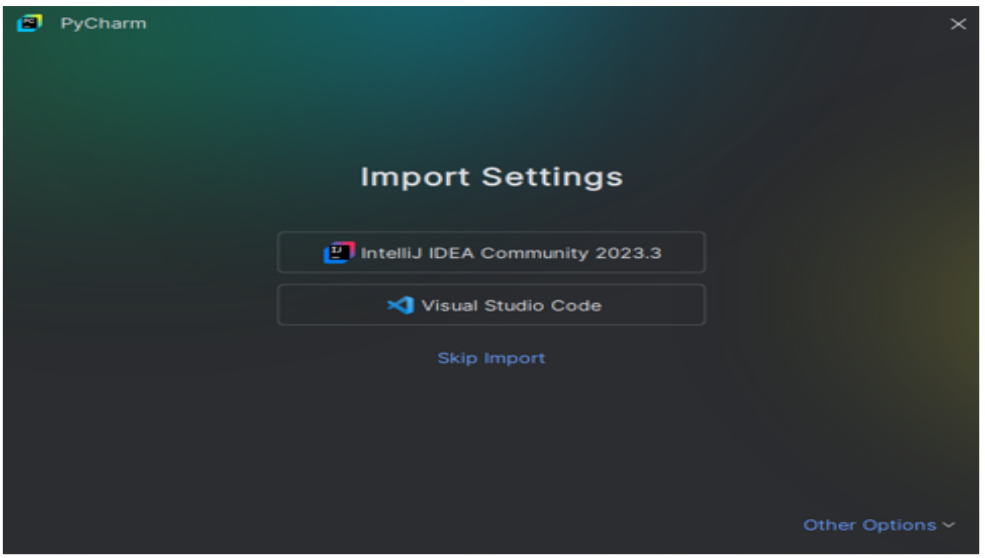

Figure 1.8: Selecting IDEA in PyCharm

Next, click the **"Start using PyCharm"** button to open the main window. Now the user can **create a new project** or **open an existing one**. To create a new project, click the **"New Project"** button (Figure 1.10). In the New Project window, the **project name, location, and Python interpreter** are selected. If Python is already installed, PyCharm detects it automatically. Otherwise, click **"Add Interpreter"** and choose either **"System Interpreter"** or **"Virtualenv Environment"**. Usually, a **Virtualenv** is used to create a separate environment for each project, which helps prevent conflicts between libraries. After selecting the interpreter, click **"Create"**. Once the project is opened, the PyCharm interface consists of several **main sections**.

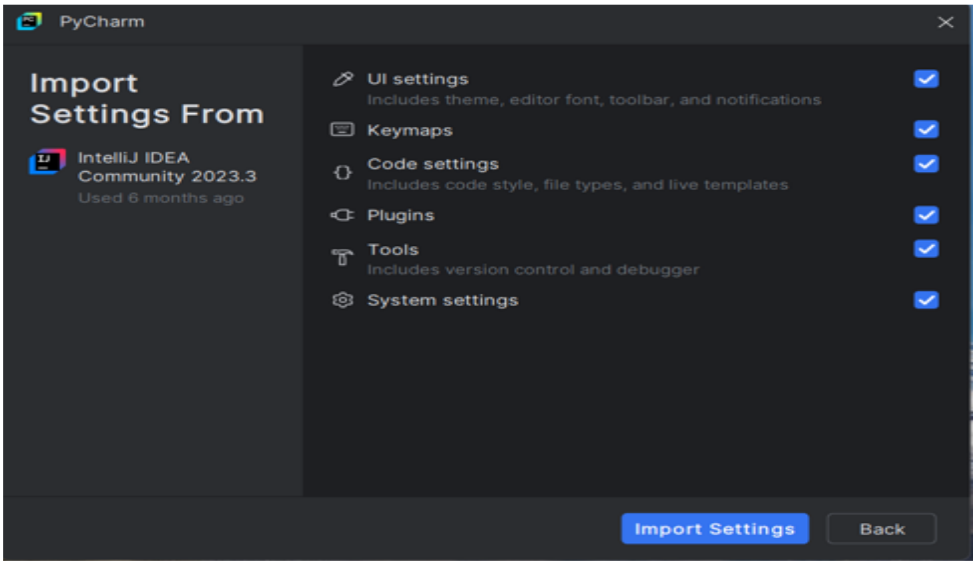

Figure 1.9: Import settings in PyCharm

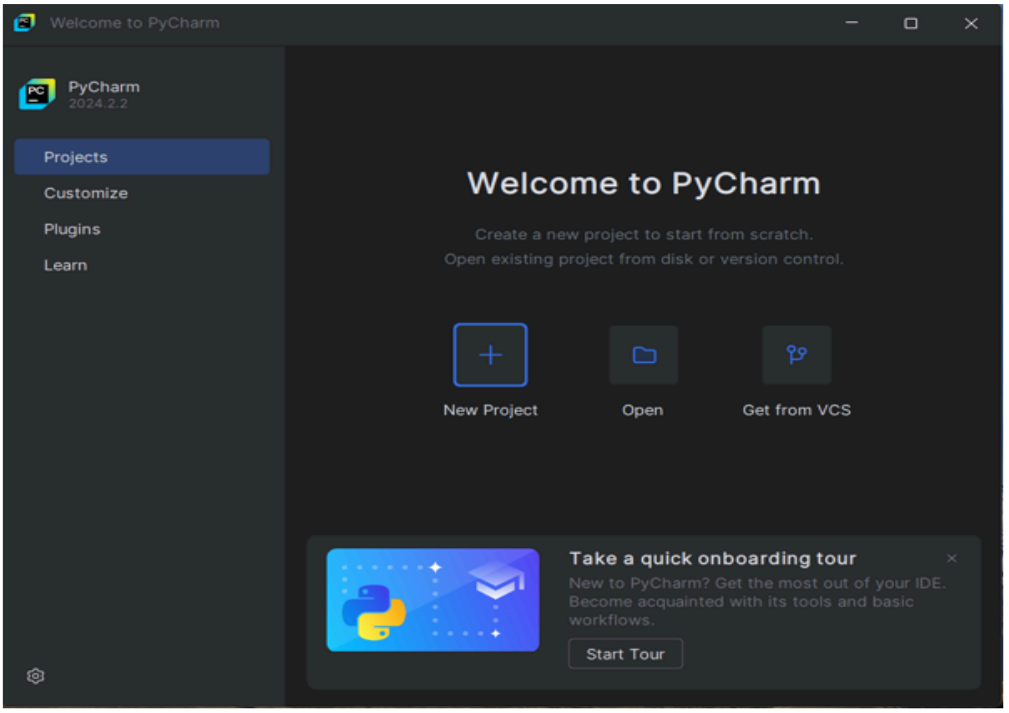

Figure 1.10: PyCharm successfully installed

Coding section

```
import mesa


class HelloAgent(mesa.Agent):
    def __init__(self, model):

        super().__init__(model)

    def step(self):

        print(f"Agent {self.unique_id}: Hello, multi agent systems!")


class HelloModel(mesa.Model):
    def __init__(self, num_agents=3):
        super().__init__()

        for i in range(num_agents):
            a = HelloAgent(self)


    def step(self):
```

```
        self.agents.shuffle_do("step")


model = HelloModel(3)
for i in range(2):
    print(f"--- {i+1}-step ---")
    model.step()
```

Output

```
--- 1-step ---
Agent 1: Hello, multi agent systems!
Agent 3: Hello, multi agent systems!
Agent 2: Hello, multi agent systems!
--- 2-step ---
Agent 3: Hello, multi agent systems!
Agent 2: Hello, multi agent systems!
Agent 1: Hello, multi agent systems!
```