Hello everyone,

This course uses Python for the practical and implementation parts. Before we start implementing a Python project, we need to prepare the necessary software prerequisites. This section will help you set up your development environment and get comfortable with the tools and libraries you'll be using throughout the course.

---

What we will learn in this part.

In the first session, we will get a quick overview of all of them, explaining what they are, what they are used for, and providing some major providers for each software component.

Then, in the second session, we will go into detail on how to install each of them.

---

To help you onboard the implementation part of the course, I would like to propose five essential software components or tasks that we need to prepare before starting the coding process.

--

First of all, IDE, stands for Integrated Development Environment. An IDE is a comprehensive software suite designed to facilitate software development.

It typically includes a source code editor, build automation tools, a debugger, code completion, syntax highlighting, version control integration, etc.

It allows developers to write, compile, test, and debug their code within a single interface., enhancing productivity and streamlining the development process.

Popular IDEs include Visual Studio, Eclipse, IntelliJ IDEA, and PyCharm, each catering to different programming languages and development needs.

--

Secondly, Terminal.

A Terminal, also known as a command-line interface (CLI), is a text-based interface used to interact with an operating system.

It allows users to execute commands by typing them manually and provides direct access to various system functions and utilities.

The terminal is commonly used for file manipulation, program execution, and system administration tasks.

It is a powerful tool for developers and system administrators, offering greater control and flexibility compared to graphical user interfaces (GUIs).

Examples of terminal applications include Bash (Linux and macOS), Command Prompt (Windows), and PowerShell (Windows).

--

Thirdly, of course, Python.

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum and first released in 1991.

Python emphasizes code readability with its clear syntax and indentation structure.

It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Python is widely used for web development, data analysis, artificial intelligence, scientific computing, and automation.

Its extensive standard library and active community contribute to a vast ecosystem of frameworks and tools, making it a versatile choice for both beginners and experienced developers.

Popular Python frameworks and libraries include NumPy, pandas, Scikit-learn, TensorFlow, PyTorch, Django, Flask, Requests, etc.

--

Fourthly, Python environment,

A Python environment refers to a setup where Python and its related tools and libraries are installed and configured to develop and run Python applications.

There are different types of Python environments, including:

* **Virtual Environment**: An isolated environment that allows for separate installations of Python and its packages for different projects, avoiding conflicts. Tools like **venv** (built-in) and **virtualenv** help create these environments.
* **Conda Environment:** Provided by the Anaconda distribution, Conda environments manage dependencies and packages for Python and other languages, offering more robust control over the environment.
* **Integrated Development Environment (IDE):** Many IDEs like PyCharm, VSCode, and Jupyter Notebook provide built-in support for creating and managing Python environments, integrating development tools and facilitating seamless coding workflows**.**

Using these environments ensures that projects have the necessary dependencies without interference from other projects or the global system environment.

--

Lastly, Python packages or libraries.

Python packages or libraries are collections of modules and functions that extend the capabilities of the Python programming language.

They allow developers to reuse code and implement complex functionality without having to write it from scratch.

These libraries are typically installed using the Python package manager, **pip**, and are integral to various fields such as data science, web development, automation, and artificial intelligence.

---

Well, we have an overview of all the software components for building a Python project. Let’s go into detail to see how to install each of them.

---

In this session I will present some popular IDEs, I would recommend Visual Studio Code or VS Code and PyCharm.

Regarding VS Code. You can visit its official webpage to download and install it based on your operating system: Windows, Linux, or macOS.

Another popular option for an IDE is PyCharm. Similarly, you can visit the download page on jetbrains.com, select your operating system, and download the appropriate PyCharm installer.

---

Now, let’s see what we can have for Terminal.

For Window environment, there are many options for terminal, like PowerShell, Command Prompt, Windows Terminal, Git Bash, WSL. For example, with PowerShell, you can download it for free from the Microsoft Store.

For Linux, you can use the default terminal emulator.

For macOS, you can use Terminal or iTerm2. For example, with iTerm2, you can follow the download link to get it.

--

Now, let’s look at how to install Python, our programming language for this course.

To install Python, I recommend two options: installing it via the Python official page or via the Anaconda distribution.

For the first option, installing Python via its official page, you can access the official download page of Python using this link. Then, select the appropriate installer for your operating system (Windows, macOS, or Linux).

For the second option, you can also install Python by installing Anaconda. You can access the installation page of Anaconda, choose the installer corresponding to your operating system (Windows, macOS, or Linux), then download the distribution and follow the installation steps.

With Anaconda, you can install a specific Python version for each project within a conda environment using the 'conda' tool. I prefer this method.

--

Next, we will see how to set up Python environment.

For Python environment, I recommend two common ways to set it up, using ‘venv’ or using ‘conda’

For the first option: we can use **venv**, a module included since Python 3.3, to create a virtual Python environment. Follow these steps:

1. Step 1, navigate into the project directory because the Python environment will be created in this working directory.
2. Step 2, Create a Python environment, for example, named ‘.MY\_ENV’. We use a ‘.’ before ‘MY\_ENV’ to distinguish it from other source code folders. Use this command: **python -m venv .MY\_ENV**. For Linux or macOS, you may need to use ‘python3’ instead of ‘python’.
3. Step 3, Activate the created environment. For Windows, use this command: **.\.MY\_ENV \Scripts\activate**, and for Linux or macOS, use this command: **source .MY\_ENV/bin/activate**.
4. Step 4. Install the required packages and start coding. I will present in detail how to install the Python packages in the next session.
5. Last step. When you want to close the development environment, use this command: **deactivate**.

For the second option: using ‘conda’. There are four main steps:

1. Step 1: Create an environment named ‘MY\_ENV’, and optionally specify a Python version, using this command: **conda create --name MY\_ENV python=3.x**. You don't need to add a ‘.’ before the environment name ‘MY\_ENV’ because the environment directory will be created inside the Anaconda directory, not the current working directory.
2. Step 2: After creating the environment, activate it with: **conda activate MY\_ENV**.
3. Step 3: Install the packages and start working. We will discuss this in the next session.
4. Step 4: To stop the development environment, use: **conda deactivate**.

--

Next, let’s see how we can install Python package in the activated Python environment

First, we need to activate the Python environment.

Next, my preferred method for installing Python packages is using **pip**, a Python package manager, for both **venv** and **conda** environments.

To install a single package, use the command: **pip install package\_name==version**. For example: **pip install torch==2.3.0**.

To install all the required packages, create a **requirements.txt** file containing all the required packages with their specific versions. Each line should list a package, like this:

package\_name==version

Then, use a single line of code to install all of them: **pip install -r requirements.txt**.

--

Finally, I would like to show you which pieces of software I used in this course. For me, while working on a Python project, I use the following:

* **For IDE:** I use VS Code.
* **For Terminal:**
  + As I work on Windows, I use PowerShell.
* **For Python Installation and Environment Creation:** I prefer using the **conda** tool from the Anaconda distribution.
* **Python Package Installation:** I use **pip**.

These choices are based on personal preference and may vary for different users.