Hello everyone, this video is dedicated to the implementation aspect of the Transformer using the Python language and the PyTorch framework.

Before we start the implementation of a Python project, we need to prepare the necessary software prerequisites. There are five essential software components or tasks we need to complete before jumping into the coding process.

--

First of all, IDE, stands for Integrated Development Environment. An Integrated Development Environment (IDE) is a comprehensive software suite designed to facilitate software development. It typically includes a source code editor, build automation tools, and a debugger, allowing developers to write, compile, test, and debug their code within a single interface. Some IDEs also offer features like code completion, syntax highlighting, version control integration, and GUI builders, 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 Django, Flask, NumPy, pandas, and TensorFlow.

--

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.

Next, we will go into detail about how to install each of them.

---

Regarding IDEs, I recommend 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.

---

Regarding terminals, there are many options for Windows. I recommend PowerShell, which you can download for free from the Microsoft Store. For Linux, you can use the default terminal emulator. For macOS, you can use Terminal or iTerm2. I recommend iTerm2, and you can follow the download link to get it.

--

Regarding Python, there are two options for installing it on your operating system.

First, you can access the official download page of Python and select the appropriate installer for your operating system.

Second, you can install Anaconda, which allows you to install a specific Python version for each project within a conda environment using Anaconda's 'conda' tool. I prefer this method.

--

Next, regarding the Python environment, I present two common ways to set it up.

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

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

The second option: using ‘conda’. Follow these steps:

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

--

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, here are my recommendations for the software prerequisites. These are based on personal preference and may vary for different users. For me, while working on a Python project, I use the following:

* **IDE:** I use VS Code.
* **Terminal:**
  + On Windows, I use PowerShell.
  + On Linux, I use the default Terminal.
  + On macOS, I use iTerm2.
* **Python Installation and Environment Creation:** I prefer using the **conda** tool from the Anaconda distribution.
* **Python Package Installation:** I usually use **pip**.