WELCOME !!

CodeCraft : Unleashing Langchain & LLMs

- Week 1: Day 0

- Nandan Hemanth

Prerequisites

- Basic knowledge of Python programming Language
- How legos building blocks works

What is Langchain?

🦜🔗 LangChain

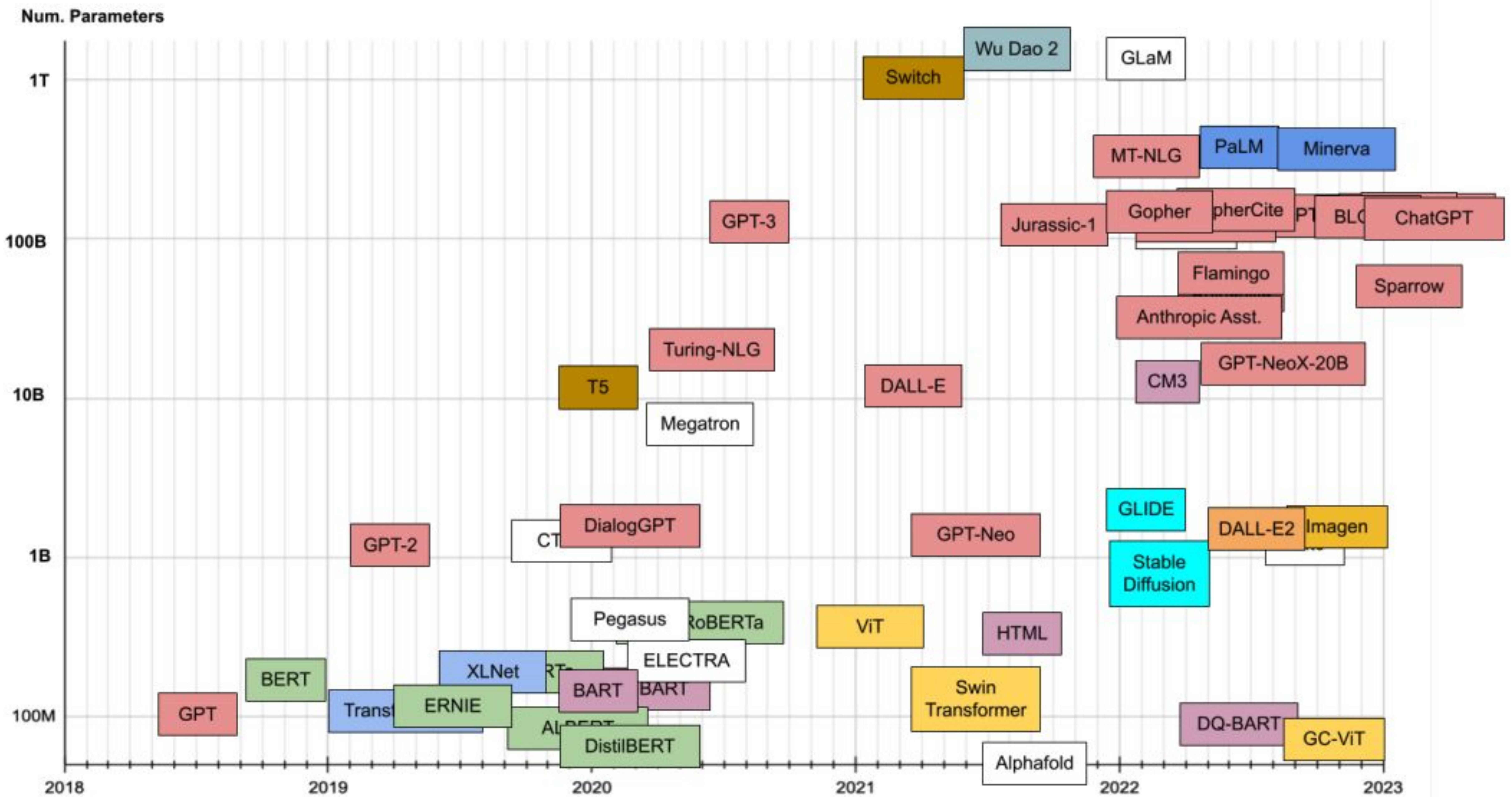
- LangChain is an open-source framework designed to simplify the creation of applications powered by Large Language Models (LLMs).

- Think of it as a set of tools (like Legos!) that allows you to connect different building blocks to achieve remarkable results in language processing tasks.

What are Large Language Models (LLMs)?

- LLMs are like language geniuses trained on massive amounts of text data.
- They are powerful AI models capable of various **language-related tasks**, including:
  - **Generating text:** Writing creative content, translating languages, crafting different writing styles.
  - **Understanding text:** Answering questions, summarizing information, analyzing sentiment.

Growth rate of LLMs:



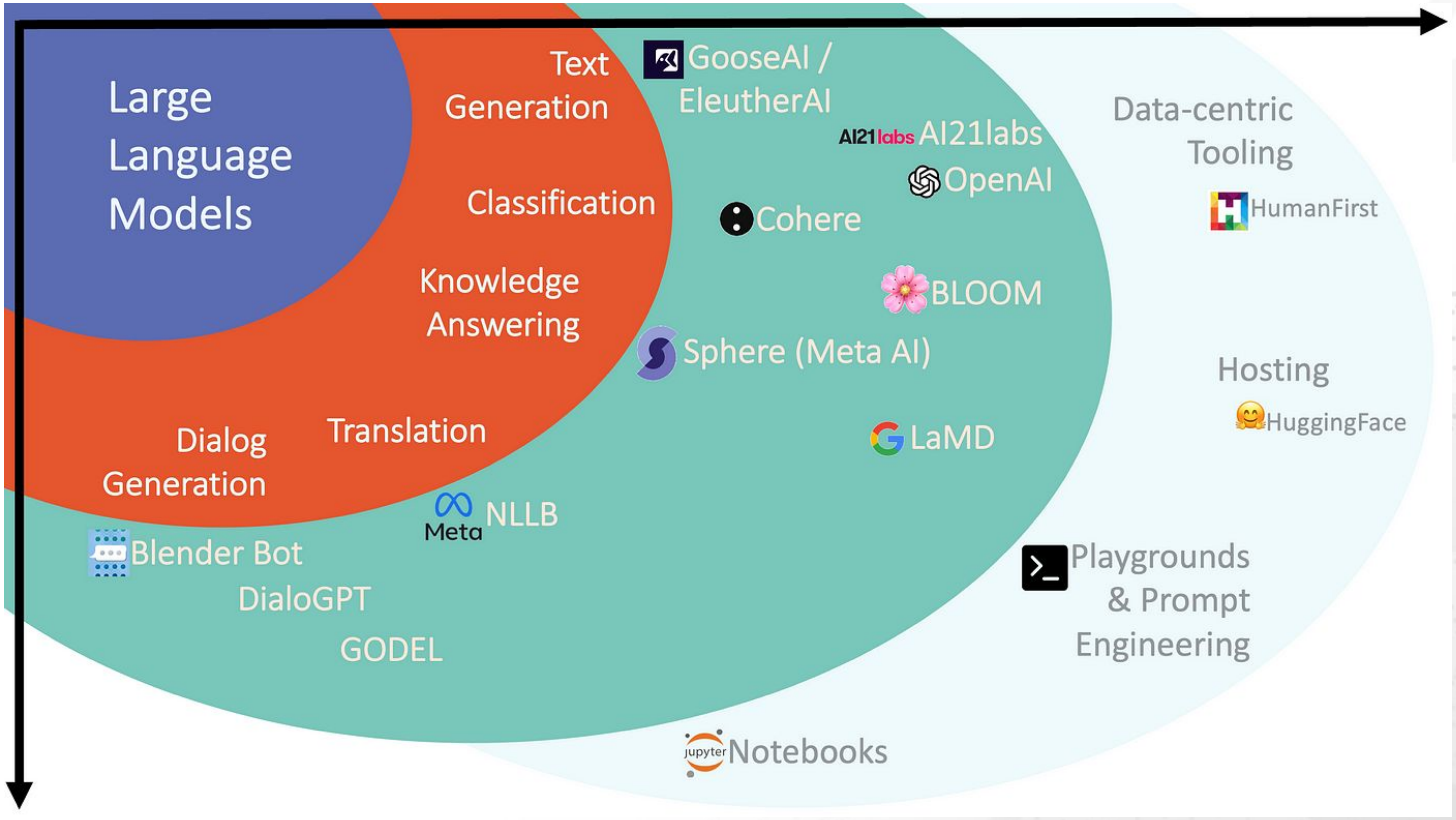
How is an LLM trained?

- LLMs are trained on **massive datasets of text and code**, similar to how a child learns language by being exposed to speech and text.
- This training allows them to:
  - **Recognize patterns** in language and identify relationships between words and phrases.
  - **Predict** what word or phrase is likely to come next, allowing them to generate text, translate languages, and answer questions.

What is the LLM architecture?

- While various architectures exist, **<u>transformers</u>** are a popular choice for LLMs due to their ability to:
  - **<u>Process information in parallel</u>**, making them efficient for handling large amounts of data.
  - **<u>Model relationships</u>** between different parts of the text, leading to better understanding and generation capabilities.

Evaluating LLMs:



Bridging the Gap of Langchain & LLMs:

- LLMs are incredibly powerful, but their raw capabilities can be **difficult to access and control** directly. Think of them as complex engines with immense potential, but not readily accessible to users.
- LangChain acts as a **user-friendly interface** that simplifies this interaction. It provides the **tools and structure** to effectively connect users with the capabilities of LLMs.

Bridging the Gap of Langchain & LLMs:

- LangChain offers building blocks called "chains", which are essentially instructions and data that you use to tell the LLM what to do. You can combine these chains in various ways to achieve specific tasks.
  - This allows users to:
    - **Provide clear instructions** to the LLM about the desired outcome.
    - **Control the flow of information**, specifying what data the LLM should access and how it should be processed.
    - **Fine-tune the LLM's behavior** for optimal performance in specific tasks.

Applications of the Synergy:

- Machine translation: Translate text from one language to another seamlessly.
- Chatbot development: Create engaging chatbots that can answer questions and hold conversations.
- Text analysis and summarization: Analyze large amounts of text and extract key information or create summaries.
- Content creation: Generate different creative text formats like poems, scripts, or marketing materials.

Real-world Scenario example:

- *Imagine you want to build a chatbot that can answer customer questions about your products.*

Using LangChain, you could:

- Create a chain that fetches relevant product information from your database.
- Design another chain that allows the LLM to understand and respond to customer queries in a natural and informative way.
- Combine these chains within LangChain to create your fully functional chatbot.

References:

- Generative AI with Langchain: https://drive.google.com/file/d/1remcULuWBlloxx8Ud9KsQldPPm0bI1ks/view?usp=sharing

- Langchain Documentation: https://python.langchain.com/docs/get_started/introduction

Join the Whatsapp Group:

CodeCraft: Langchain & LLMs

WhatsApp group

Feedback and Improvements:

- Pace of teaching? (slow / fast/ more explanation)
- Energizers and Refreshers (games/ quizes/ activity)
- Any personal issues?

Thank You!

WELCOME !!

CodeCraft : Unleashing Langchain & LLMs

- Week 1: Day 1

- Nandan Hemanth

Installing Software

🦜🔗 LangChain

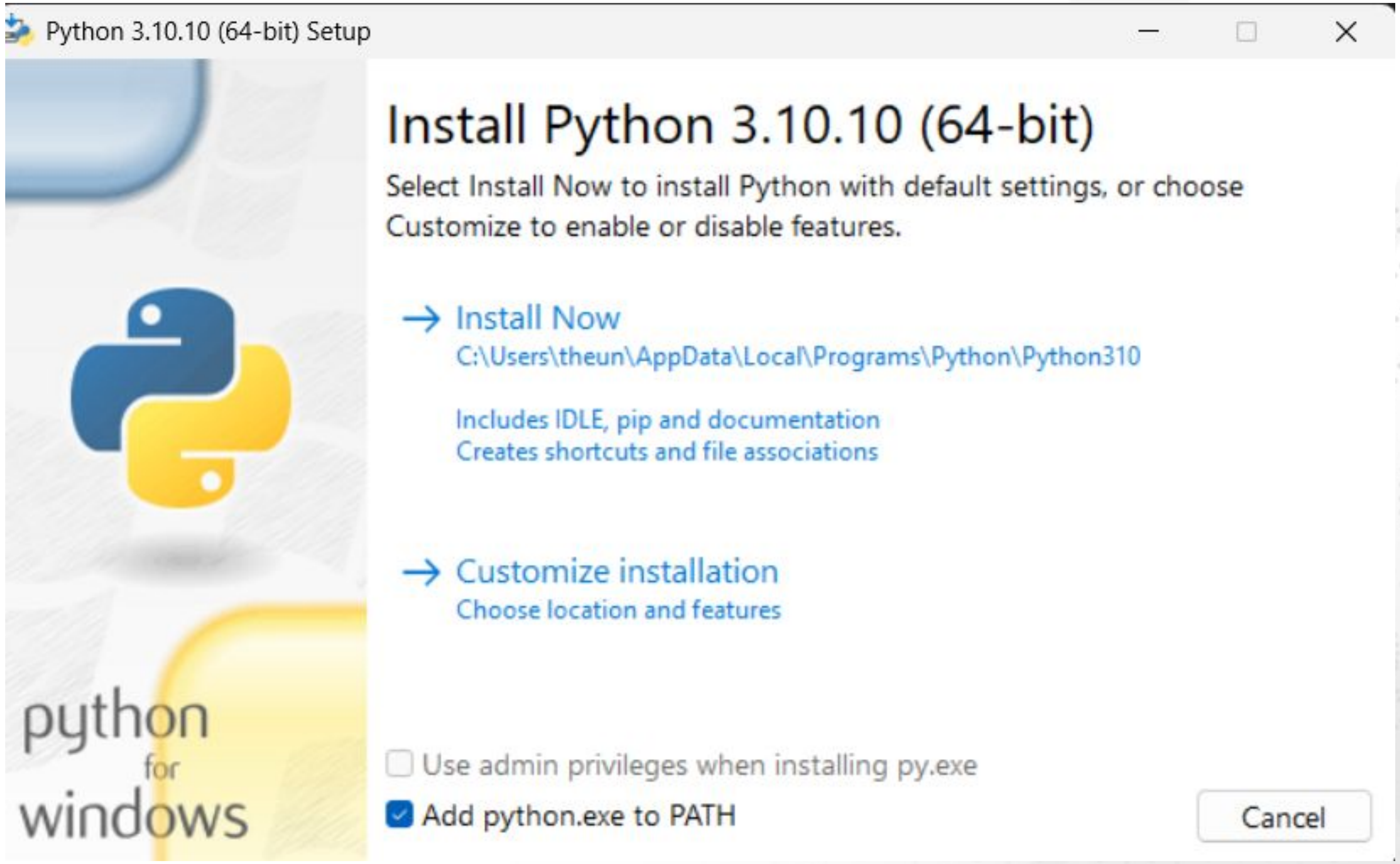
Python: https://www.python.org/downloads/

Anaconda : https://www.anaconda.com/download

VSCode: https://code.visualstudio.com/download

Installing Software - Python

Remember to check “Add python.exe to PATH” box!



Python 3.10.10 (64-bit) Setup

Install Python 3.10.10 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\theun\AppData\Local\Programs\Python\Python310

Includes IDLE, pip and documentation
Creates shortcuts and file associations

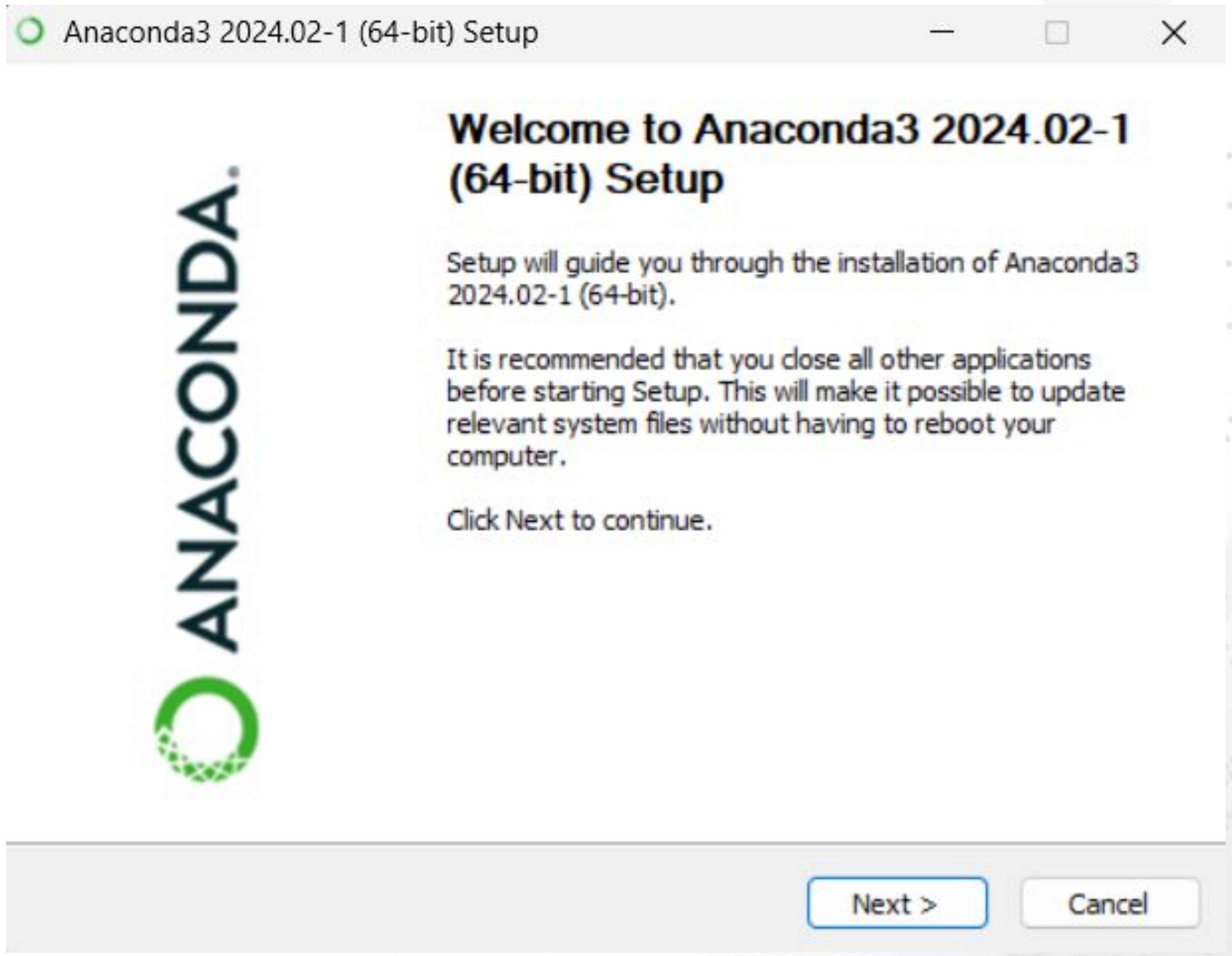
→ Customize installation
Choose location and features

☐ Use admin privileges when installing py.exe

☑ Add python.exe to PATH

Installing Software - Anaconda

You can keep all the default settings for Anaconda



Welcome to Anaconda3 2024.02-1 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2024.02-1 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

Installing Code Editor

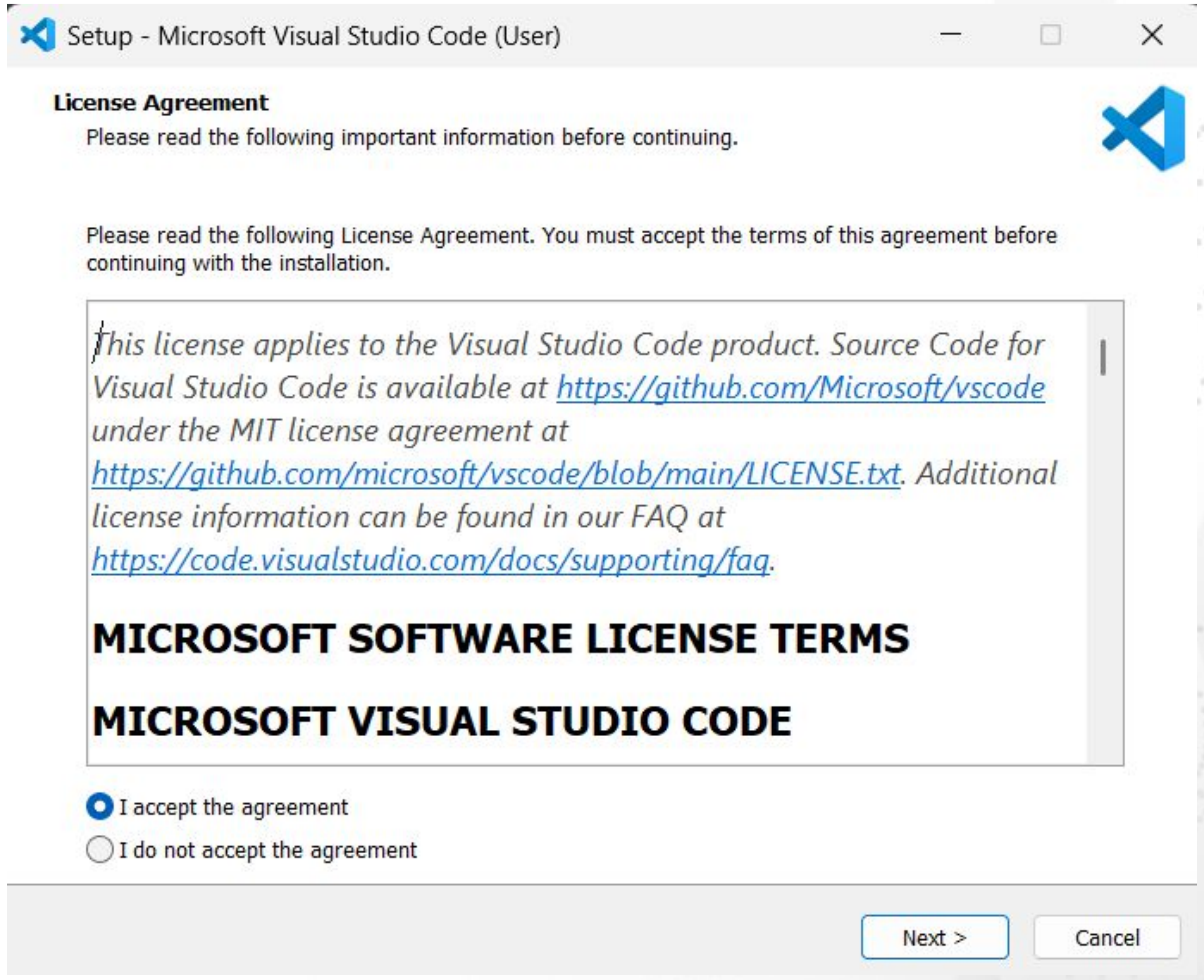
VSCode (Recomended): https://code.visualstudio.com/download

1st Alt (Zed Editor): https://zed.dev/

Alt Code Editors - Atom, Vim, Pycharm

Installing Software - Code Editor

You can keep all the default settings!



License Agreement

Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO CODE

[✓ I accept the agreement]

I do not accept the agreement

Command Prompt commands - Windows

- Terminal commands (Windows):

  ‘cd <dir name>’ - change directory (forward)

  ‘cd ..’ - change back to previous directory

  ‘dir’ - list all directory content

  ‘mkdir’ - make new directory

  ‘cd <X>:’ - change volume

  ‘code .’ - open current directory in code editor

Terminal Commands - Ubuntu

- Terminal commands (Ubuntu):

  ‘cd <dir name>’ - change directory (forward)

  ‘cd ..’ - change back to previous directory

  ‘ls’ - list all directory content

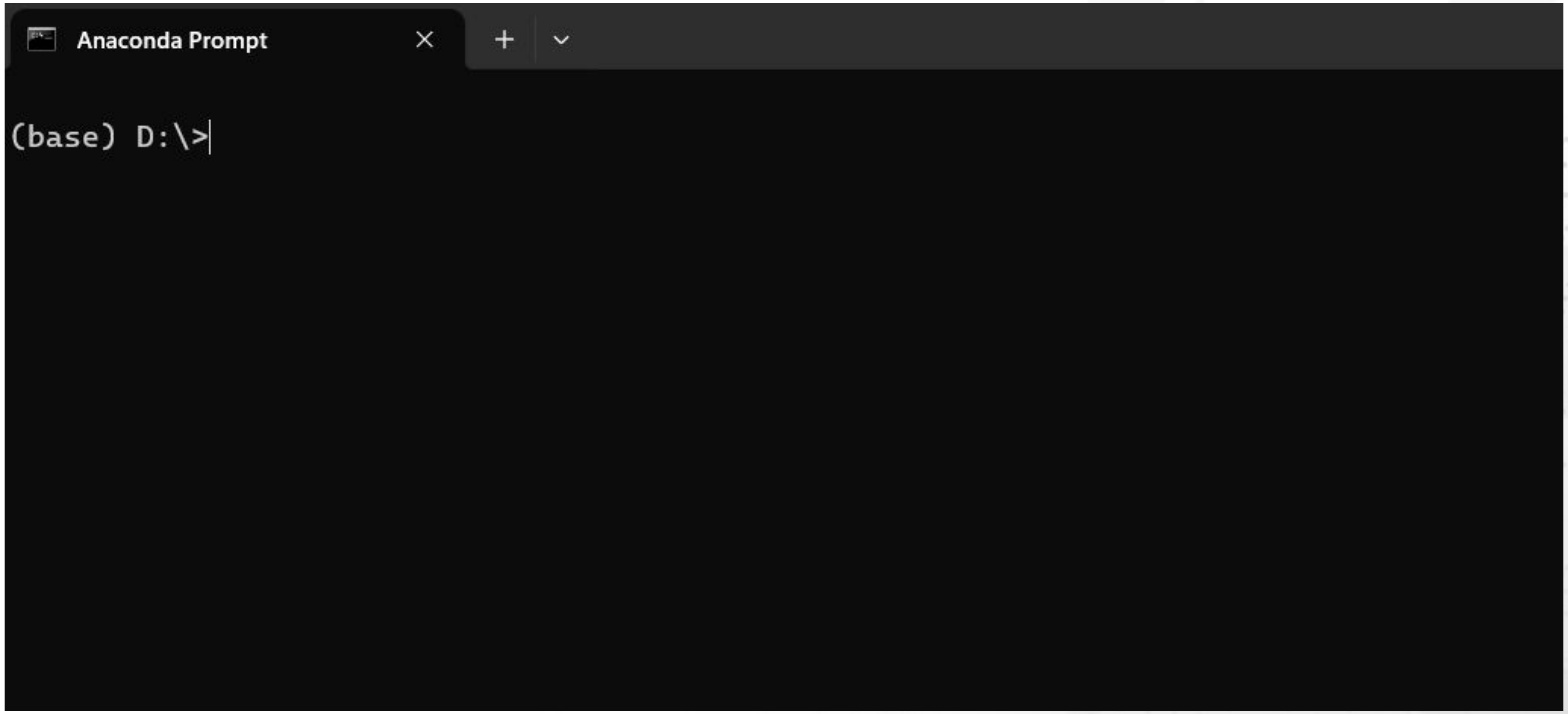
  ‘mkdir’ - make new directory

  ‘cd <X>:’ - change volume

  ‘code .’ - open current directory in code editor

Anaconda Setup - “conda”

1. Open Anaconda Prompt

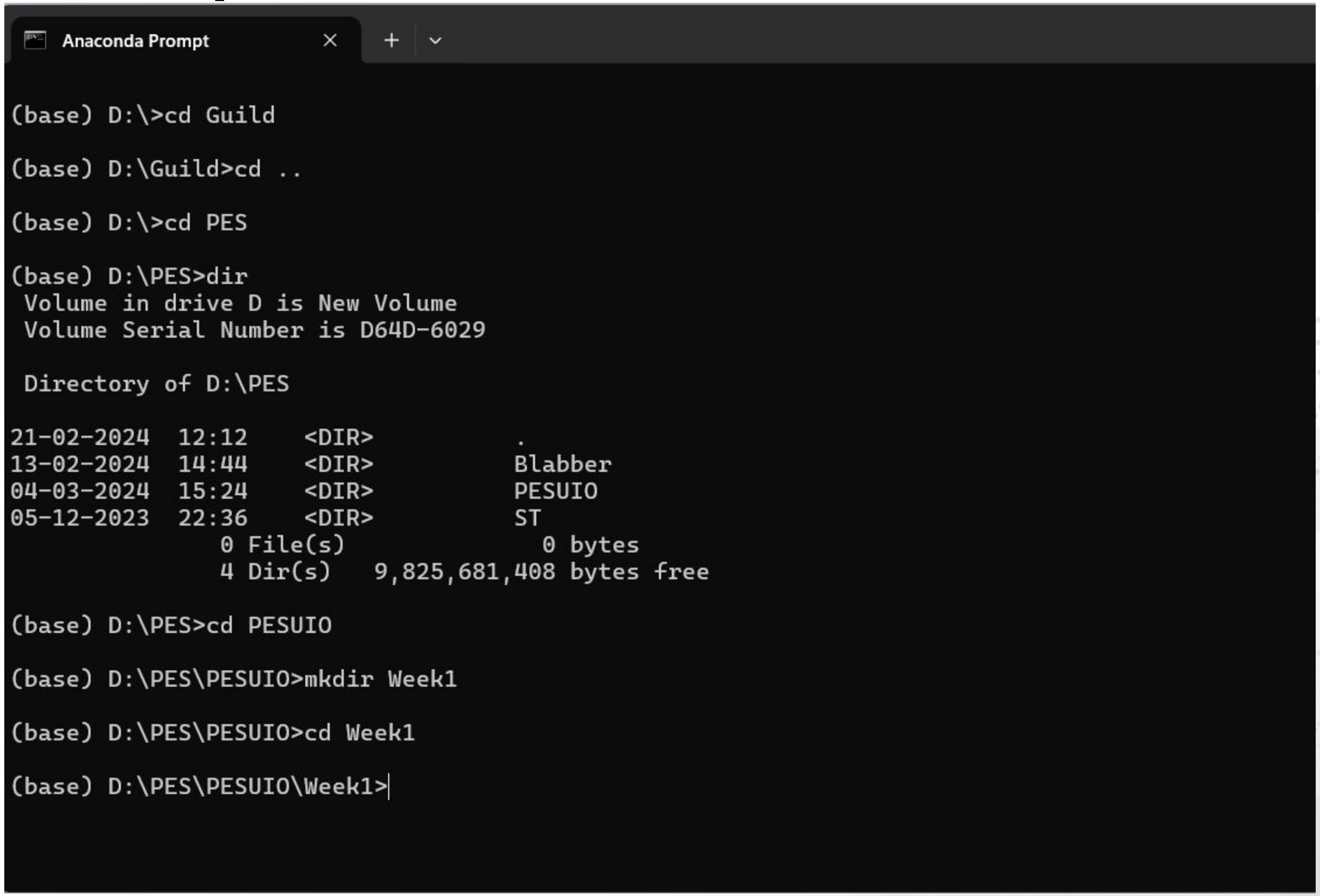


```
(base) D:\>
```

Anaconda Setup – “conda”

2. Navigate to the working directory using “Terminal Commands”
3. Activate Virtual Environment

Anaconda Setup - “conda”

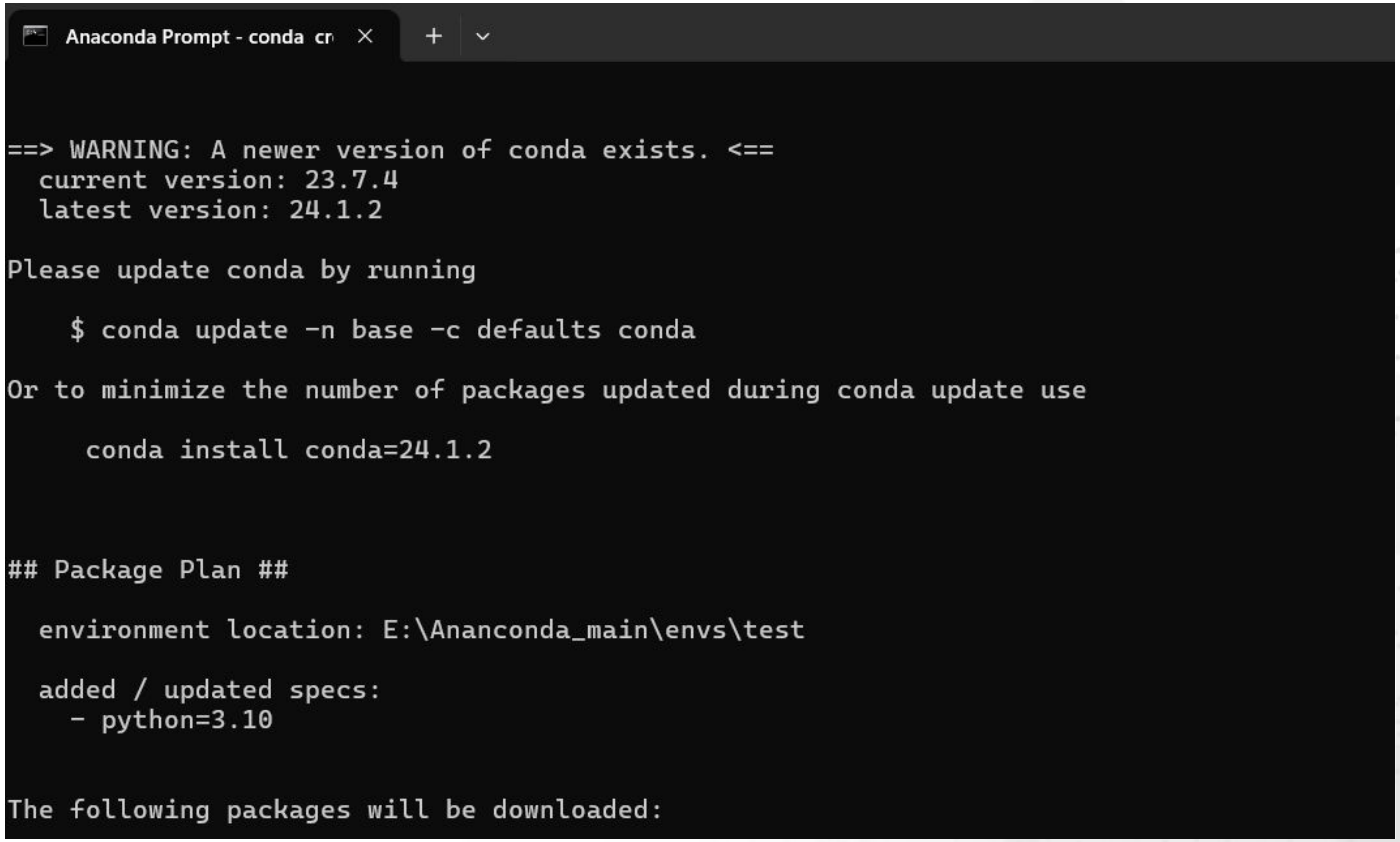

Anaconda Prompt

```
(base) D:\>cd Guild

(base) D:\Guild>cd ..

(base) D:\>cd PES

(base) D:\PES>dir
 Volume in drive D is New Volume
 Volume Serial Number is D64D-6029

 Directory of D:\PES

21-02-2024  12:12    <DIR>          .
13-02-2024  14:44    <DIR>          Blabber
04-03-2024  15:24    <DIR>          PESUIO
05-12-2023  22:36    <DIR>          ST
               0 File(s)              0 bytes
               4 Dir(s)   9,825,681,408 bytes free

(base) D:\PES>cd PESUIO

(base) D:\PES\PESUIO>mkdir Week1

(base) D:\PES\PESUIO>cd Week1

(base) D:\PES\PESUIO\Week1>
```

Anaconda Setup - Activating Virtual environment

- ‘conda env list’ - list all virtual envs
- ‘conda create -n <env_name> <python_version>’ - create a new virtual environment with the specified python version
- ‘conda activate <env_name>’ - activates the virtual environment

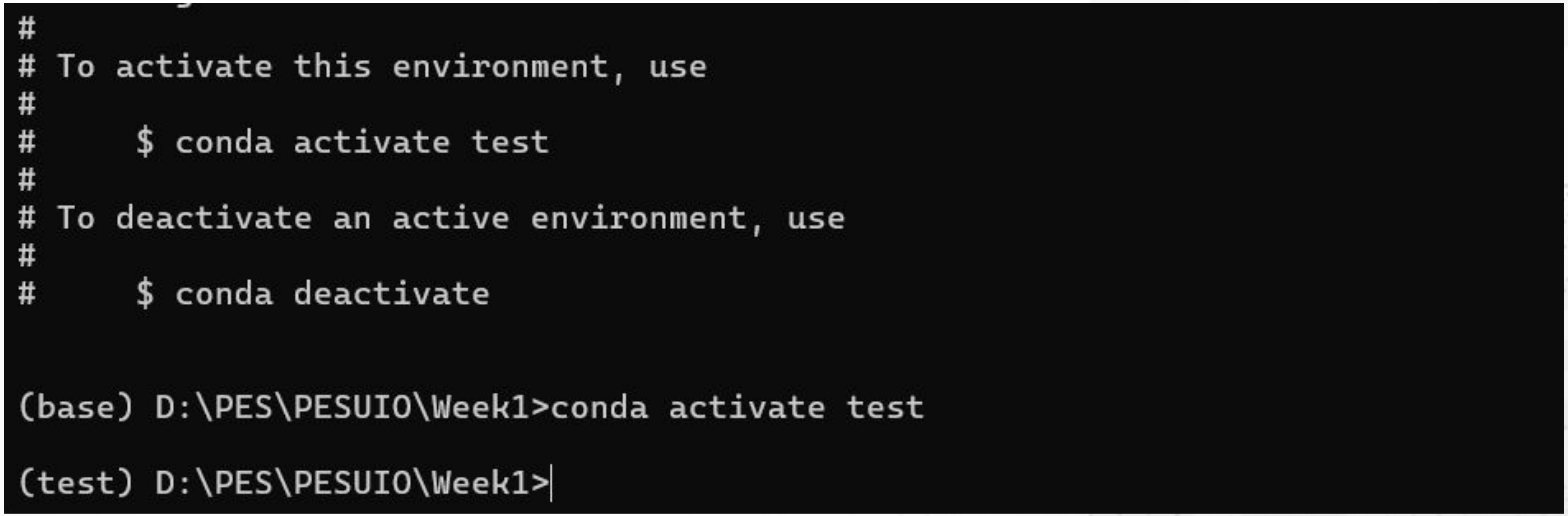
Anaconda Setup - Activating Virtual environment



Anaconda Prompt - conda cr

```
==> WARNING: A newer version of conda exists. <==
  current version: 23.7.4
  latest version: 24.1.2

Please update conda by running

    $ conda update -n base -c defaults conda

Or to minimize the number of packages updated during conda update use

     conda install conda=24.1.2



## Package Plan ##

  environment location: E:\Ananconda_main\envs\test

  added / updated specs:
    - python=3.10


The following packages will be downloaded:
```

Anaconda Setup - Activating Virtual environment



```
#
# To activate this environment, use
#
#     $ conda activate test
#
# To deactivate an active environment, use
#
#     $ conda deactivate


(base) D:\PES\PESUIO\Week1>conda activate test

(test) D:\PES\PESUIO\Week1>
```

Installing dependencies using ‘pip’

- “pip” - preferred installer program
- All libraries and frameworks can be installed using ‘pip’
- Eg: (*run these commands in the cmd or terminal*)
  - “pip install pandas”
  - “pip install numpy”

Installing langchain

- Command Prompt or Terminal:

  a. “pip install langchain”

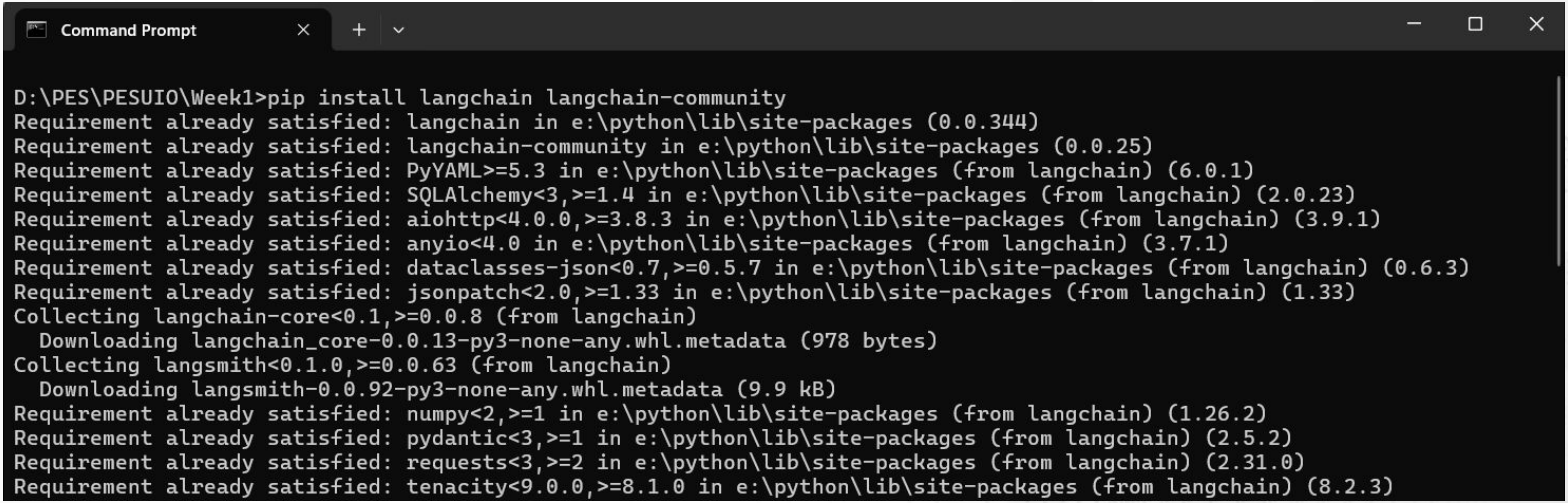
  b. “pip install langchain-community”

- Anaconda Prompt:

  c. “conda install langchain”

  d. “conda install langchain-community”

Installing langchain



```
D:\PES\PESUIO\Week1>pip install langchain langchain-community
Requirement already satisfied: langchain in e:\python\lib\site-packages (0.0.344)
Requirement already satisfied: langchain-community in e:\python\lib\site-packages (0.0.25)
Requirement already satisfied: PyYAML>=5.3 in e:\python\lib\site-packages (from langchain) (6.0.1)
Requirement already satisfied: SQLAlchemy<3,>=1.4 in e:\python\lib\site-packages (from langchain) (2.0.23)
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in e:\python\lib\site-packages (from langchain) (3.9.1)
Requirement already satisfied: anyio<4.0 in e:\python\lib\site-packages (from langchain) (3.7.1)
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in e:\python\lib\site-packages (from langchain) (0.6.3)
Requirement already satisfied: jsonpatch<2.0,>=1.33 in e:\python\lib\site-packages (from langchain) (1.33)
Collecting langchain-core<0.1,>=0.0.8 (from langchain)
  Downloading langchain_core-0.0.13-py3-none-any.whl.metadata (978 bytes)
Collecting langsmith<0.1.0,>=0.0.63 (from langchain)
  Downloading langsmith-0.0.92-py3-none-any.whl.metadata (9.9 kB)
Requirement already satisfied: numpy<2,>=1 in e:\python\lib\site-packages (from langchain) (1.26.2)
Requirement already satisfied: pydantic<3,>=1 in e:\python\lib\site-packages (from langchain) (2.5.2)
Requirement already satisfied: requests<3,>=2 in e:\python\lib\site-packages (from langchain) (2.31.0)
Requirement already satisfied: tenacity<9.0.0,>=8.1.0 in e:\python\lib\site-packages (from langchain) (8.2.3)
```

Thank You!

WELCOME !!

CodeCraft : Unleashing Langchain & LLMs

- Week 1: Day 2

- Nandan Hemanth

Topics to be covered:

- Installation of the environment
- Available LLMs (open and closed source)
- LLM prompting and chatting
- Prompt Templates
- What are chains
- How to build sequential chains
- Memory

Available LLMs:

- **Open-source LLMs:** In the world of Large Language Models (LLMs), open-source LLMs stand out as those whose underlying code and architecture are accessible to the public.
  Eg: GPT2, gpt4all, llama-1, huggingface, etc
- **Closed-source LLms:** In contrast to open-source LLMs, closed-source LLMs are those where the underlying code and architecture are not publicly accessible. They are developed and maintained by private organizations or companies and often treated as proprietary technology.
  Eg: GPT4, Jurassic-1 Jumbo, etc

What are chains?

- In the world of LangChain, chains are the **<u>fundamental building blocks</u>** that enable you to connect various elements and functionalities to achieve specific tasks.
- They act like **<u>instructions or modules that work together</u>** to process information and data using Large Language Models (LLMs).

Understanding Chains

- Think of them as <u>**reusable components**</u> that perform specific actions within your LangChain application.
- Each chain <u>**encapsulates a particular functionality**</u>, like data processing, LLM interaction, or manipulation of outputs.
- You can <u>**combine and sequence multiple chains**</u> to create complex workflows and achieve more intricate results.

Types of Chains:

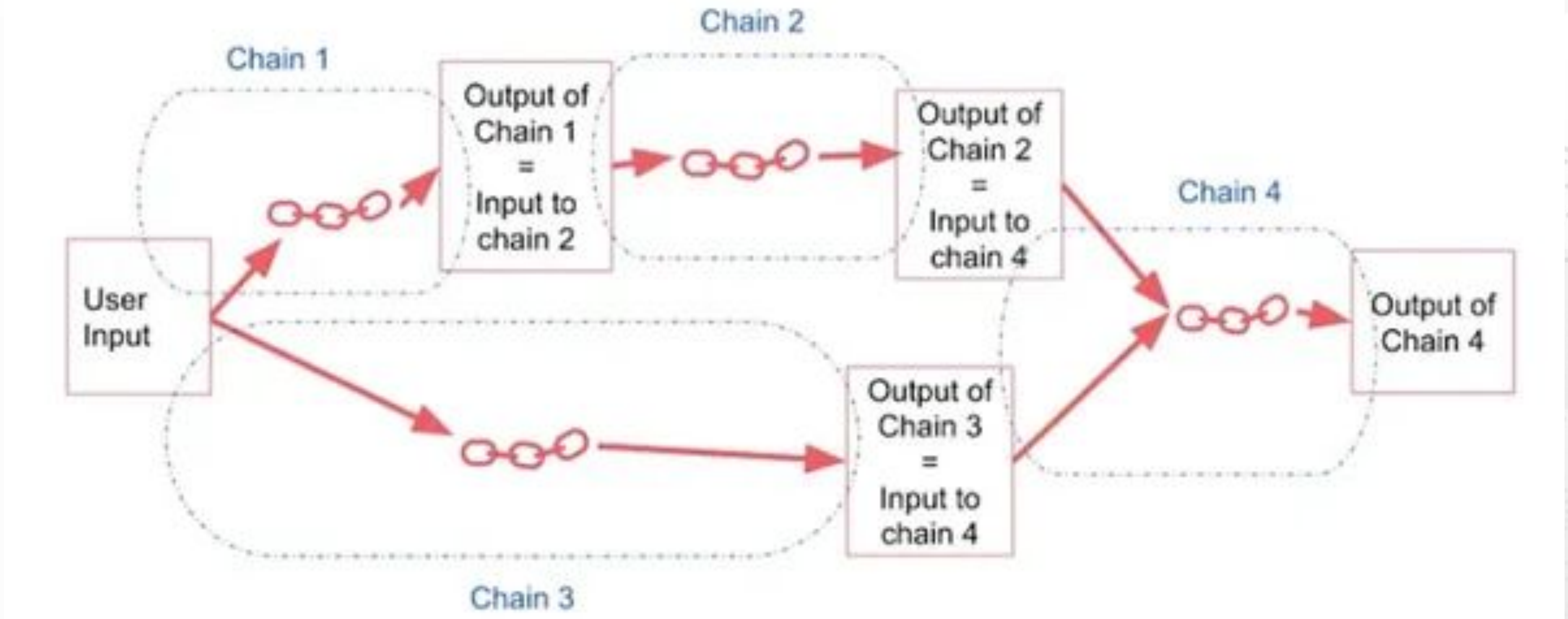
- **<u>LLM Chains:</u>** These are the core chains that interact directly with LLMs, providing instructions, data, and processing the LLM's outputs.
- **<u>Data Processing Chains:</u>** These chains handle tasks like data pre-processing, cleaning, and formatting before feeding it to the LLM.
- **<u>Utility Chains:</u>** These are pre-built chains offering functionalities like summarization, translation, or sentiment analysis, simplifying your workflow.

Real-world Example:

Imagine you want to translate a document using LangChain and an LLM:

- <u>**Data Processing Chain:**</u> This chain would read the document and format the text for the LLM.
- <u>**LLM Chain:**</u> This chain would send the formatted text to the LLM and receive the translated version.
- <u>**Output Chain:**</u> This chain might handle formatting the translated text for display.
- You would then connect these chains in the desired order to form a complete workflow for document translation.

Real-world Example:


Chain 1
Chain 2
Chain 3
Chain 4
User Input
Output of Chain 1 = Input to chain 2
Output of Chain 2 = Input to chain 4
Output of Chain 3 = Input to chain 4
Output of Chain 4

Benefits of Using Chains:

- **Modularity:** Break down complex tasks into smaller, reusable chains, promoting code organization and maintainability.
- **Flexibility:** Combine different chains in various ways to create diverse applications, catering to specific needs.
- **Readability:** Chains improve code readability by clearly separating different functionalities within your LangChain application.

What are Prompt Templates?

- In LangChain, prompt templates serve as powerful tools to <u>**standardize and streamline the process**</u> of creating prompts for Large Language Models (LLMs).
- They act as <u>**reusable blueprints**</u> that define the structure and essential elements of your prompts, reducing redundancy and ensuring consistency.

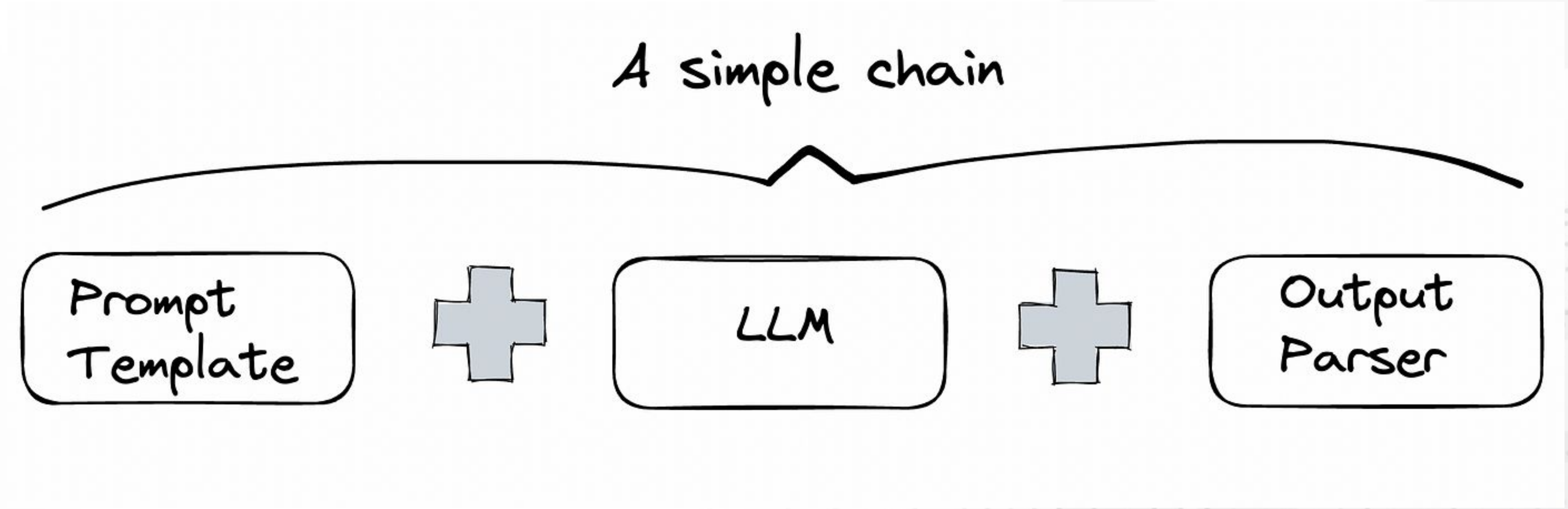
Understanding Prompt Templates:

- Imagine building prompts for a specific task, like summarizing text snippets. Instead of writing each prompt individually, you can create a template that captures the core elements.
- This template would define placeholders for specific information, such as the text snippet to be summarized.
- When needed, you can fill these placeholders with the actual data to generate individual prompts for each text snippet, maintaining a consistent structure and format.

Key Components of a Prompt Template:

- **Instructions:** These guide the LLM on how to approach the task, providing context and setting expectations.
- **Input Variables:** These represent placeholders within the template that will be filled with actual data when generating individual prompts. They can be named and typed for clarity.
- **Template String:** This combines the instructions and placeholders, forming the overall structure of the prompt.

Prompt Template:



Benefits of Using Prompt Templates:

- **Efficiency:** Saves time and effort by eliminating the need to write individual prompts from scratch.
- **Consistency:** Ensures all prompts adhere to the same structure and format, leading to consistent LLM outputs.
- **Flexibility:** Allows you to easily adapt prompts to different datasets or tasks by modifying the input variables and template string.
- **Readability:** Improves the readability and maintainability of your LangChain code by separating prompts from the core logic.

What are sequential Chains:

- Within the realm of LangChain and its modular building blocks called "chains," sequential chains play a crucial role in achieving specific tasks.
- They are a specific type of chain characterized by the ordered execution of multiple chains one after the other.

Types of Sequential Chains:

- **Simple Sequential Chain:** This is the most basic form, consisting of a linear sequence of chains where the output from one directly feeds into the next.
- **Sequential Chain:** This offers more flexibility, allowing for multiple inputs and outputs at various stages of the sequence. This enables more complex workflows with branching or merging of data streams.

Benefits of Squential Chains:

- **<u>Clear Workflow:</u>** The linear structure provides a well-defined execution order, making it easier to understand and manage complex workflows.
- **<u>Modularity and Reusability:</u>** Individual chains within the sequence remain reusable in different contexts, promoting code efficiency.
- **<u>Efficient Data Flow:</u>** The chain-to-chain data transfer optimizes the flow of information throughout the process, ensuring seamless progression towards the desired outcome.

What is memory in langchain?

In LangChain, memory plays a crucial role in <u>**enabling contextual awareness and persistence**</u> within your language-processing applications.

Properties of memory in langchain?

Maintain State Between Calls:

- Unlike traditional applications that treat each interaction as isolated, LangChain allows you to preserve information from previous interactions using memory.
- This is particularly valuable for tasks requiring contextual understanding, such as building chatbots that remember previous conversations or creating personalized experiences based on user interactions.

Properties of memory in langchain?

Information Storage:

- User inputs and responses during interactions.
- Outputs generated by LLMs in previous stages of your LangChain application.
- Intermediate data created during the processing pipeline.

Properties of memory in langchain?

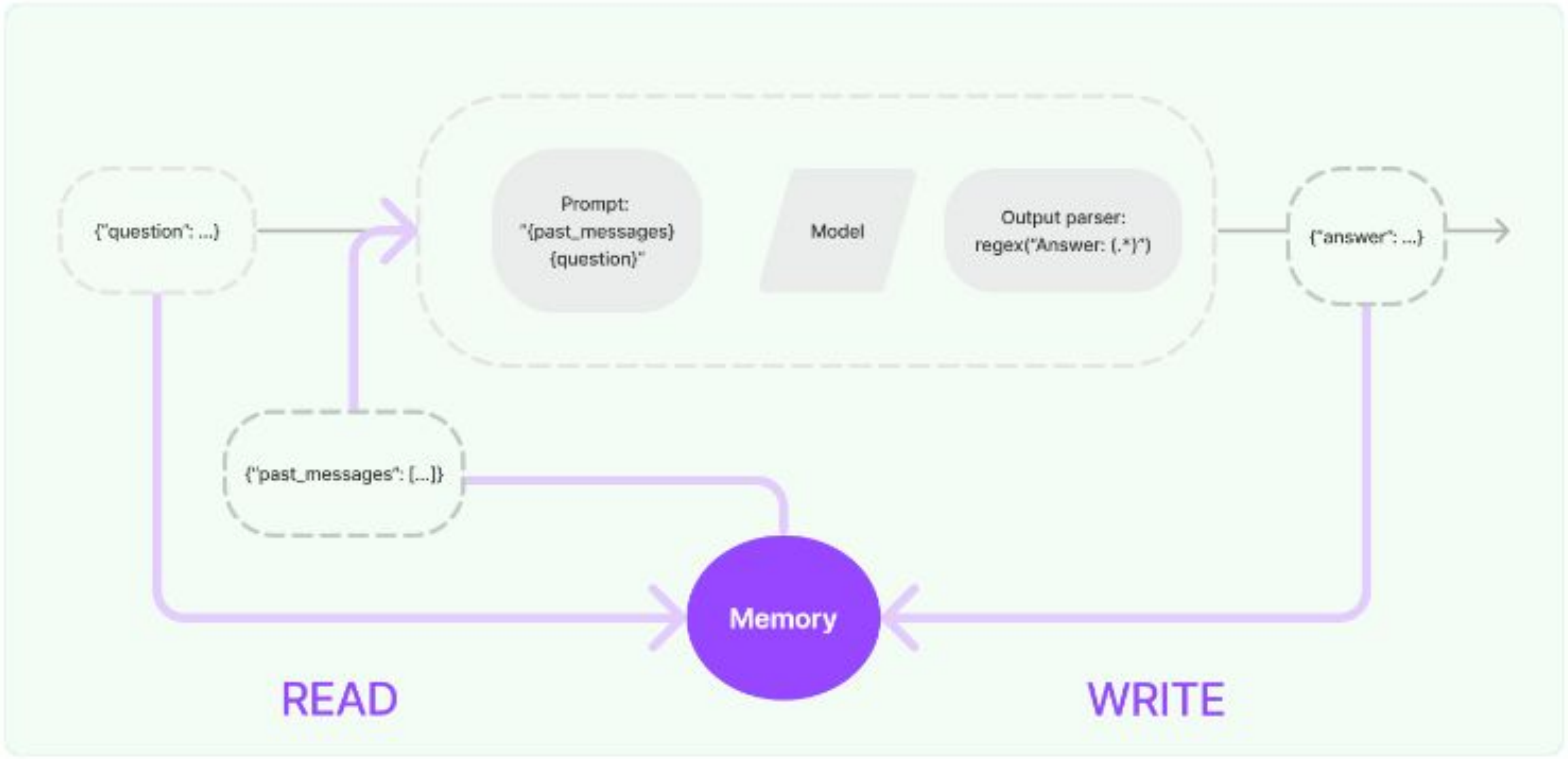
Different Memory Integrations:

- **In-memory storage:** Suitable for temporary data within a single execution.
- **Persistent storage:** Enables data retention across different executions, allowing for longer-term context building.
- **Database integrations:** Connect LangChain to external databases for more robust and scalable storage solutions.

Benefits of Using Memory in LangChain:

- <u>**Improved Contextual Understanding:**</u> Preserving information from previous interactions allows LLMs to better understand context, leading to more relevant and coherent responses.
- <u>**Personalized User Experiences:**</u> By remembering past interactions, your applications can adapt to individual users and provide a more personalized experience.
- <u>**Streamlined Workflows:**</u> Memory can simplify processing by storing intermediate data, eliminating the need to recalculate it in subsequent stages of your workflow.

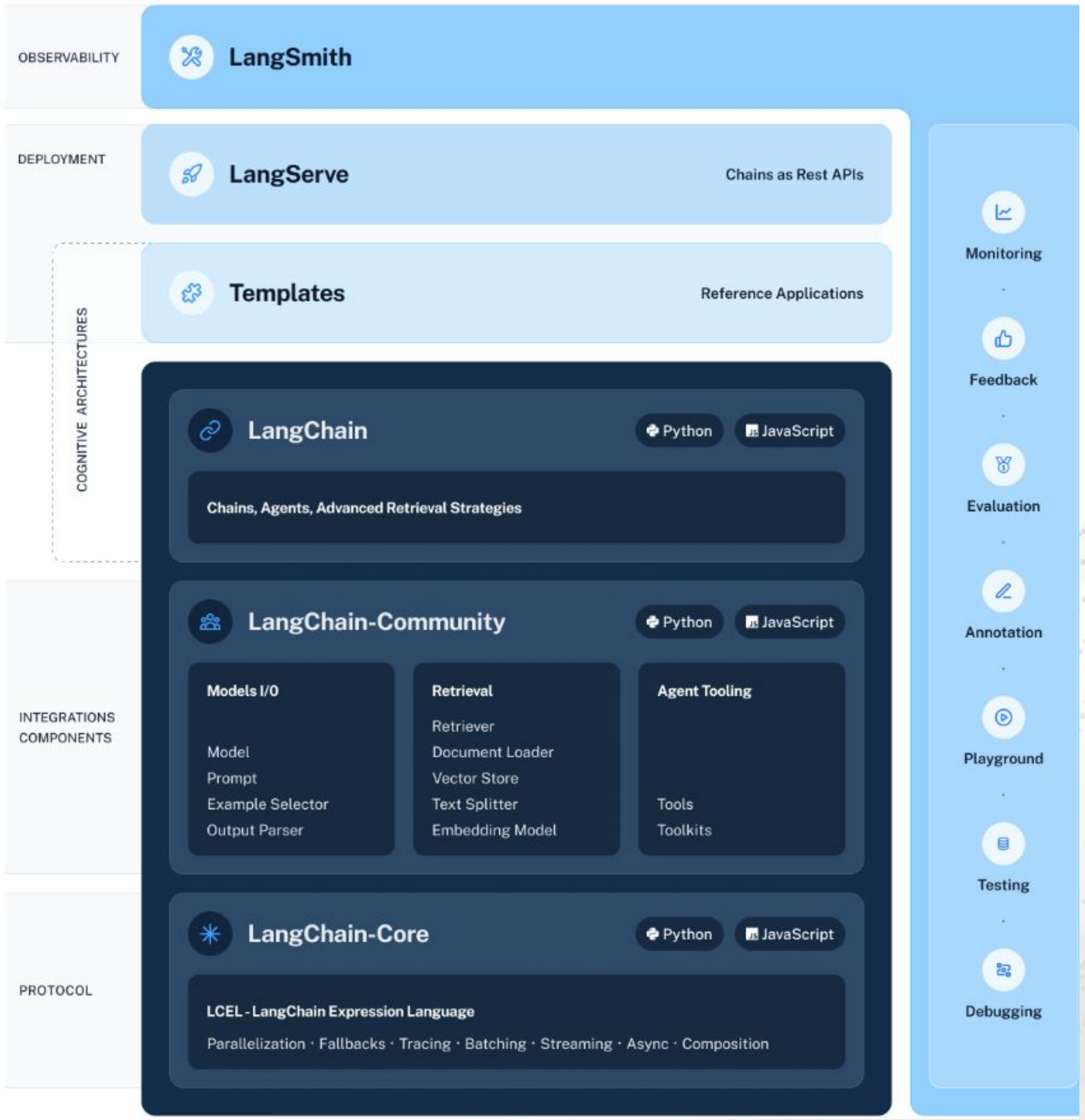
Benefits of Using Memory in LangChain:



Challenges and Considerations:

- **Data Privacy:** Carefully consider data privacy implications when utilizing memory for storing user information. Implement appropriate security measures and adhere to relevant regulations.
- **Memory Management:** Depending on the chosen integration and the amount of data stored, managing memory usage is crucial to ensure optimal performance and avoid resource exhaustion.

Langchain-


OBSERVABILITY
LangSmith
DEPLOYMENT
LangServe
Chains as Rest APIs
Templates
Reference Applications
COGNITIVE ARCHITECTURES
LangChain
Python
JavaScript
Chains, Agents, Advanced Retrieval Strategies
INTEGRATIONS COMPONENTS
LangChain-Community
Models I/O
Model
Prompt
Example Selector
Output Parser
Retrieval
Retriever
Document Loader
Vector Store
Text Splitter
Embedding Model
Agent Tooling
Tools
Toolkits
PROTOCOL
LangChain-Core
LCEL - LangChain Expression Language
Parallelization · Fallbacks · Tracing · Batching · Streaming · Async · Composition
Monitoring
Feedback
Evaluation
Annotation
Playground
Testing
Debugging

Thank You!

WELCOME !!

CodeCraft : Unleashing Langchain & LLMs

- Week 1: Day 3

- Nandan Hemanth

Topics to be covered:

- What are Agents in Langchain?
- What are indexes in langchain?
- What are document loaders?

What are Agents?

- In LangChain, agents act as the central decision-makers and orchestrators within your language processing applications.
- They are the brains behind the operations, responsible for planning, executing, and managing the overall workflow.

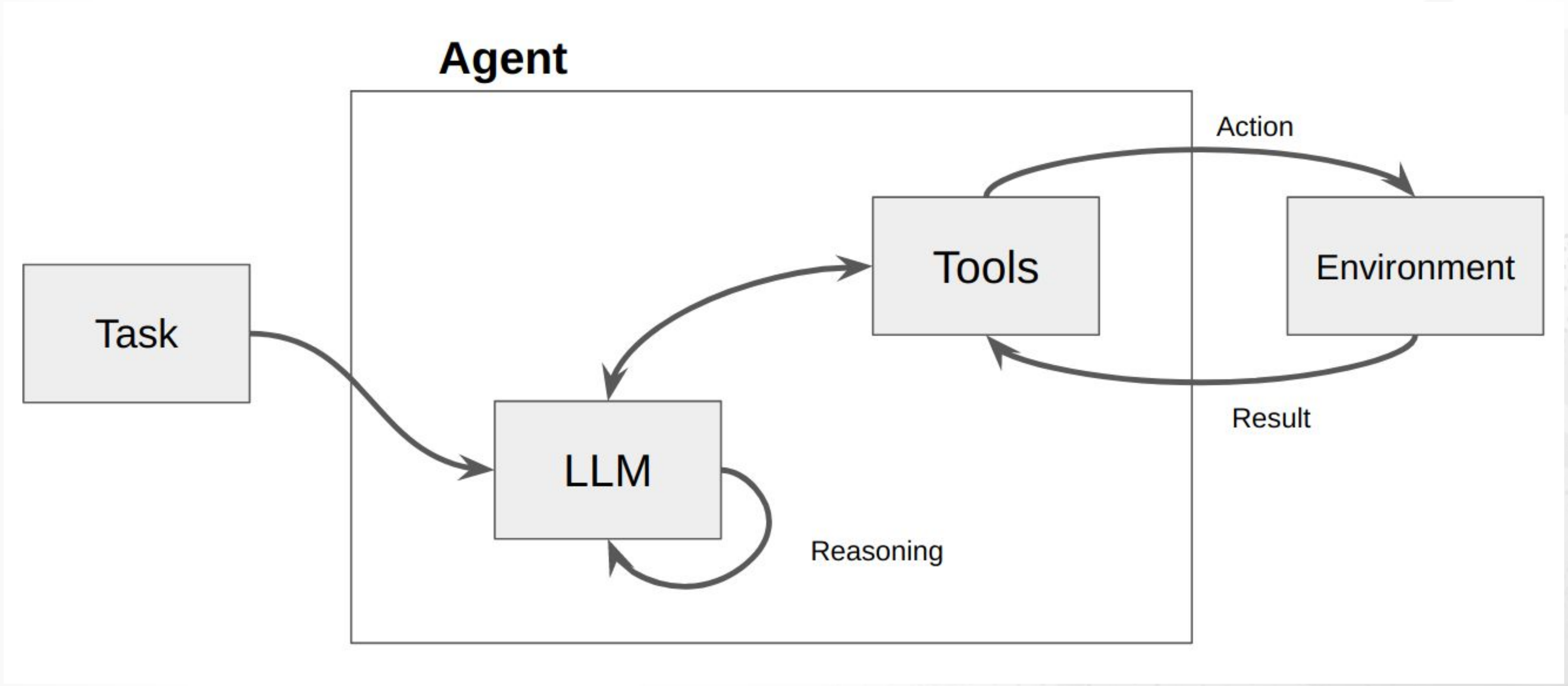
Functions of Agents:

- **Defining the Workflow:** Agents determine the sequence of actions and interactions within your application. They specify which tools to use, in what order, and what data to pass between them.
- **Interacting with Tools:** Agents utilize various tools available in LangChain to achieve specific tasks. These tools can be for:
  - Data processing: Cleaning and manipulating data before feeding it to LLMs.

Functions of Agents:

    - LLM interaction: Sending prompts to LLMs and receiving their outputs.
    - Utility functions: Performing common tasks like summarization or translation.
- **<u>Managing Memory:</u>** Agents can leverage memory to store information and maintain context across interactions. This preserved information helps LLMs better understand the current situation or personalize responses based on user history.

Functions of Agents:



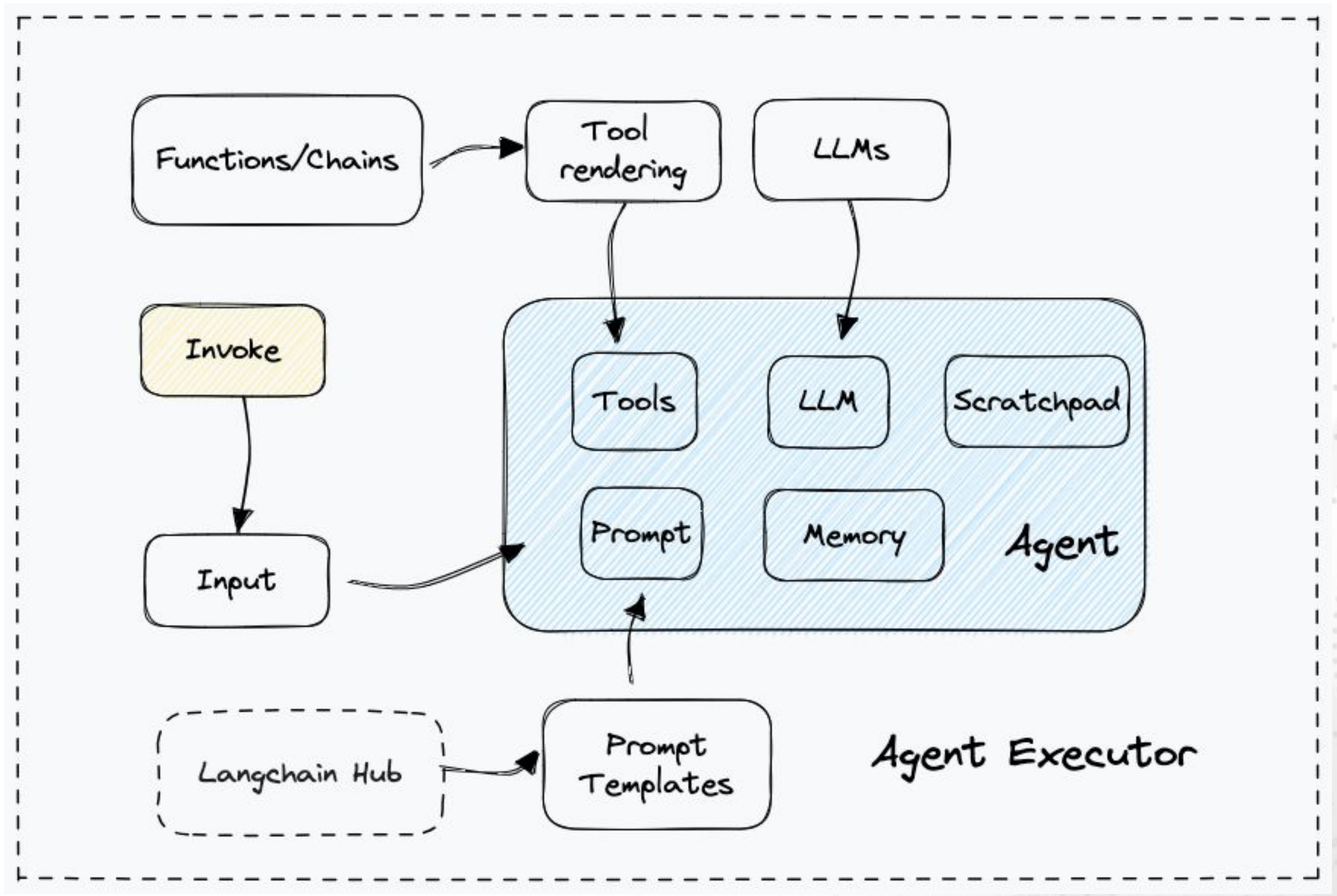
Advantages of Agents:

- <u>**Decision Making:**</u> Unlike traditional chains (sequences of calls) that follow a predefined path, agents can make decisions based on the outcome of tasks or user input. This allows for more dynamic and adaptable applications.
- <u>**Flexibility:**</u> Agents enable the creation of modular applications by combining different tools and workflows within a single agent. This promotes reusability and simplifies complex tasks.
- <u>**Efficiency:**</u> By using agents, you can create more streamlined and efficient applications, reducing the need for extensive and repetitive code for each step of the workflow.

Basic Analogy of Agents:

- Imagine building a robot that can answer your questions. The agent is the robot's control system. It decides how to access information (using data processing tools), interacts with the language model (using LLM interaction tools) to find the answer, and finally presents the answer to you (potentially using utility functions).
- By understanding the role of agents, you can unlock the full potential of LangChain and create powerful, dynamic, and adaptable language processing applications.

Basic Analogy of Agents:



What is Document loading?

In LangChain, document loading refers to the process of reading and preparing data for use within your application. It acts as the foundation for your application, as it provides the essential raw materials for the subsequent processing and interactions with Large Language Models (LLMs).

The Role of Documents:

LangChain works with data in the form of "Documents". These documents can be various types of textual data, such as:

- Plain text files
- Web pages
- Emails
- Transcripts

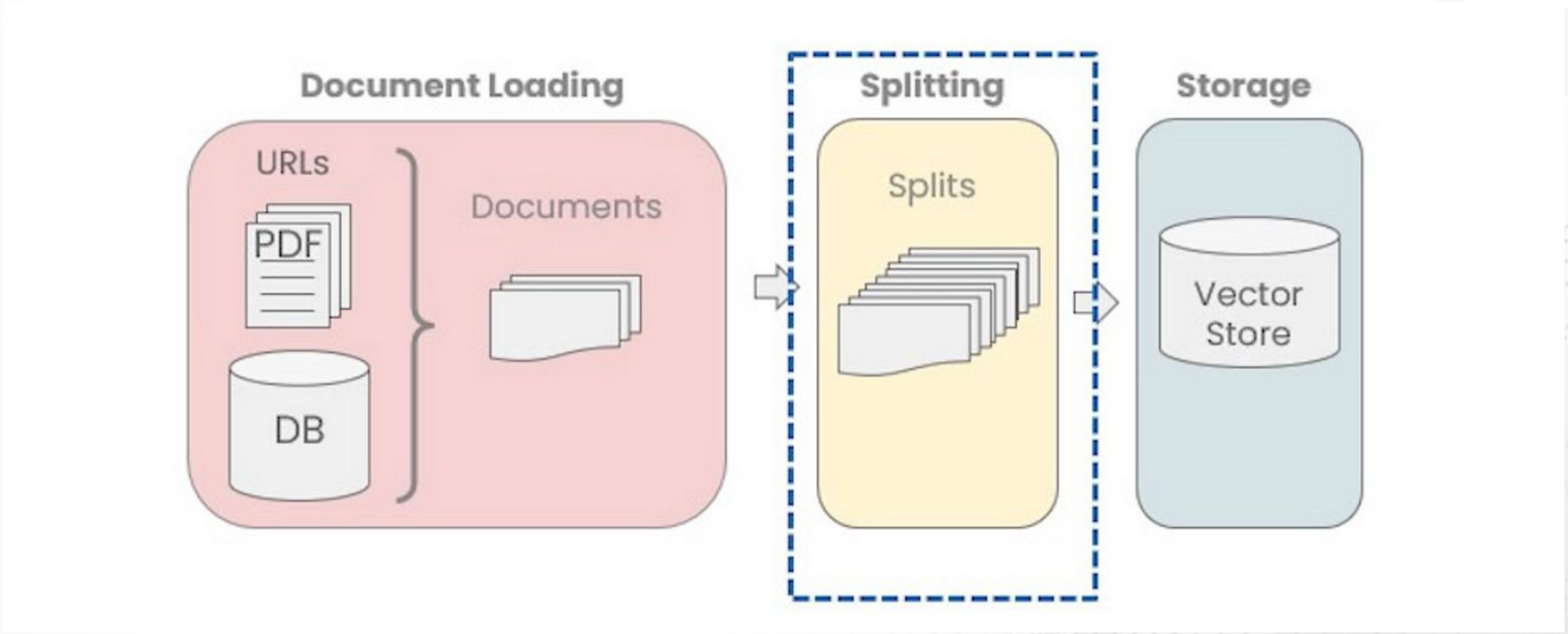
Benefits of Document Loading:

- **Flexibility:** LangChain offers a wide variety of loaders to handle diverse data sources, allowing you to work with various types of textual information.
- **Efficiency:** Loaders streamline the process of accessing and preparing data, saving you time and effort compared to manual data manipulation.
- **Consistency:** By converting data into the LangChain "Document" format, you ensure compatibility and smooth interaction with other components within your application.

Common Document Loaders:

- TextLoader: Reads text files.
- CSVLoader: Reads data from CSV (comma-separated values) files.
- JSONLoader: Reads data from JSON (JavaScript Object Notation) files.
- WebPageLoader: Fetches and processes content from web pages.
- DirectoryLoader: Loads multiple files from a directory, applying a chosen loader to each file.

Common Document Loaders:



Advanced Features in Document Loading:

- **Lazy loading:** Only load documents when explicitly needed, improving memory usage for large datasets.

- **Splitting documents:** Divide large documents into smaller chunks for efficient LLM processing.

What is Indexing in langchain?

In LangChain, indexing plays a crucial role in optimizing the performance and efficiency of your language-processing applications. It's a behind-the-scenes process that involves storing and organizing information in a way that allows for faster retrieval and processing, especially when dealing with large datasets.

Functions of Indexing:

- **Extracting key information from documents**. This could be words, phrases, or other relevant features.
- Storing this information along with document references in a special data structure called a "**vector store**".
- Building an efficient indexing system that allows for quick **searching and retrieval** of documents based on the extracted information.

Benefits of Indexing:

- **Faster Retrieval:** By using the index, LangChain can locate relevant documents in a fraction of the time compared to searching through the entire dataset. This is particularly beneficial for large datasets or applications that require frequent document access.
- **Improved Efficiency:** The indexing process helps optimize resource utilization by efficiently accessing and processing relevant information instead of scanning entire documents each time.
- **Enhanced Scalability:** As your data volume grows, the indexing system efficiently scales to handle the increased demand, ensuring consistent performance even with large datasets.

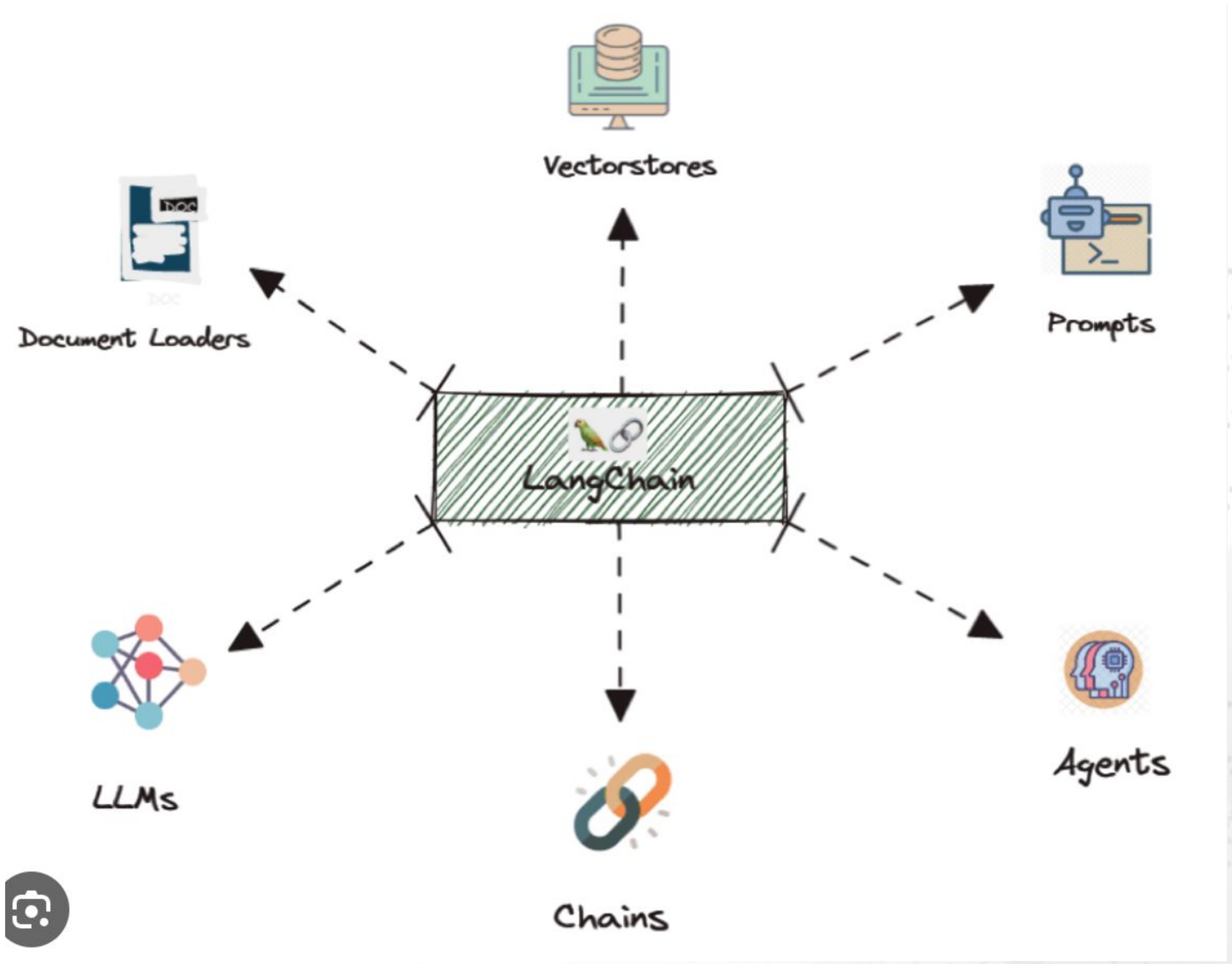
Key Components of Indexing:

- **Document Loaders:** As discussed earlier, these tools read and prepare documents, extracting essential information for indexing.
- **Vector Stores:** These specialized data structures efficiently store the extracted information and document references, enabling fast retrieval based on search queries.
- **Retrievers:** These tools leverage the index within the vector store to search for documents based on specific criteria, delivering efficient retrieval results.

Types of Indexing in LangChain:

- **Full-text indexing:** This captures all words and phrases within a document, allowing for searching based on keywords or full sentences.
- **Metadata indexing:** This focuses on capturing and indexing specific information embedded within the document (like author, date, or category), enabling searching and filtering based on these attributes.

Langchain as a whole:



Thank You!

WELCOME !!

CodeCraft : Unleashing Langchain & LLMs

- Week 2: Day 3

- Nandan Hemanth

What is streamlit?

- Streamlit is an **open-source Python library** that is designed to help developers **create web applications** for data science and machine learning with **minimal effort**.
- It simplifies the process of turning **data scripts into shareable web apps** by providing a simple and intuitive API.
- Streamlit is particularly popular among data scientists and engineers who want to **quickly prototype and deploy interactive data applications** without delving deep into web development.

Key Features of Streamlit:

- <u>**Simple and Declarative:**</u> Streamlit uses a declarative approach where you define the app's layout and functionality using Python code. This makes it easier to create UIs compared to traditional web frameworks.
- <u>**Fast Prototyping:**</u> With Streamlit, you can quickly prototype data apps and dashboards without needing extensive web development knowledge.

Key Features of Streamlit:

- **Interactive Widgets:** It provides various built-in widgets like buttons, sliders, text boxes, and charts for user interaction with your application.
- **Sharing and Deployment:** Streamlit apps can be easily shared with others by running them locally or deploying them to cloud platforms.

Use Cases for Streamlit:

- <u>**Data Exploration and Visualization:**</u> You can create interactive dashboards to explore and visualize datasets. Users can filter, sort, and interact with the data to gain insights.
- <u>**Machine Learning Model Deployment:**</u> Streamlit helps deploy machine learning models as web applications. Users can input data and see model predictions in a user-friendly interface.

Benefits of Using Streamlit:

- **Reduced Development Time:** Streamlit streamlines the UI development process, saving time compared to traditional web frameworks.
- **Improved Collaboration:** Data scientists and stakeholders can easily interact with data and models through the user-friendly interface.
- **Flexibility:** Streamlit integrates well with existing Python libraries and tools commonly used in data science.

Streamlit functions:

- **st.write(any_data):** The most versatile function, displays various data types directly on your app: text, numbers, HTML, dataframes, plots, and more.
- **st.title(title_text):** Creates a large heading for your app.
- **st.header(header_text):** Displays a smaller heading than st.title.
- **st.subheader(subheader_text):** Creates a subheading, even smaller than st.header.

Streamlit functions:

- <u>st.bar_chart(data):</u> Generates a bar chart from a list of lists or NumPy array.
- <u>st.pyplot(figure):</u> Embeds a Matplotlib figure created using pyplot into your Streamlit app.
- <u>st.video(video_data, format="video/mp4", start_time=0.0, end_time=None):</u> Embeds a video file or URL in your app.
- <u>st.audio(audio_data, format="audio/ogg", start_time=0.0, end_time=None):</u> Plays audio content.

Streamlit functions:

- **st.button(text, on_click=None, key=None):** Creates a button that triggers a callback function (on_click) when clicked.
- **st.radio(label, options, index=0, key=None):** Generates a radio button group where only one option can be selected at a time.
- **st.slider(label, min_value=None, max_value=None, value=None, key=None):** Generates a slider widget for selecting a value within a specified range.

Running Streamlit on localhost:

- Navigate to the current working directory which has the streamlit main file
- Run streamlit on localhost with this command
  - “Streamlit run <name>.py”

Thank You!

●