Langchain

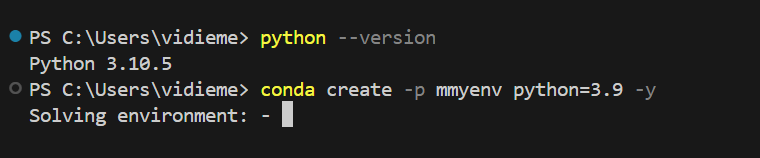
LLM: Large Language Model

LLMs are very large deep learning models that are pre-trained on vast amounts of data. The underlying transformer is a set of neural networks that consist of an encoder and a decoder with self-attention capabilities.

Used Hugginface cloud: It provides linux server (avoids aws paid server)

Python version should be greater than 3.8.1

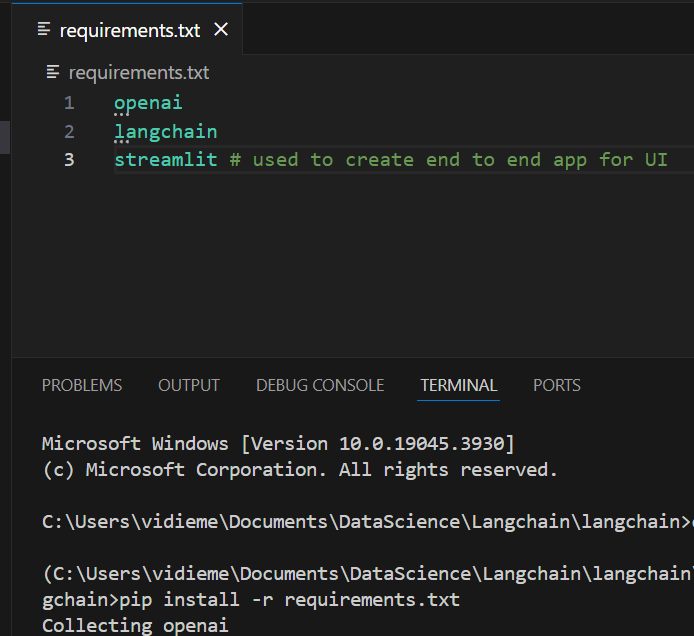
Create virtual enviroment



Activate virtual environment

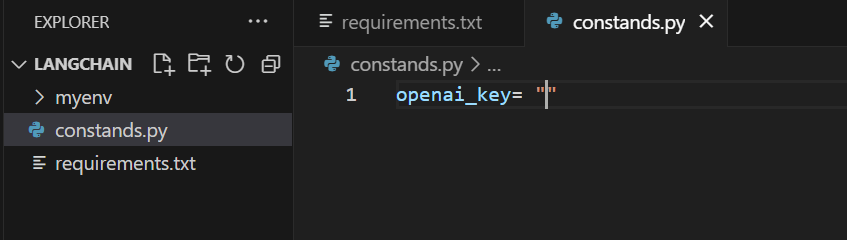
Conda activate myenv/

Create requirements.txt and install



Configure open ai api

Create constants.py: here define variable for key and paste the secret key here.



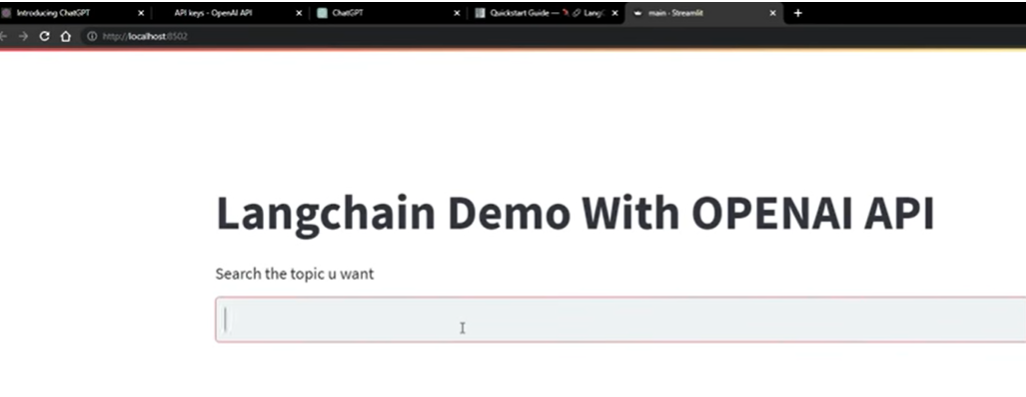
Create a new file main.py. its main file where I am integrating langchain

Documentation: <https://python.langchain.com/docs/get_started/introduction/>

Guide: <https://python.langchain.com/docs/get_started/quickstart/>

To run:

Streamlit run main.py (in cmd)



**Prompts**:

Used to solve custom usecase, where the searching would be parameter wise and not generic type.

For instance, I created a chatbot on vec, and I have some specific commands my user can ask and not the generic question like in the above example.

This needs more import.

Streamlit run firstexample.py

Control:

Streamlit will be opened and text is entered.

 this is the main() where the control goes first.

If there is any text: example: Andy Samberg

Input\_text= Andy Samberg

This is passed in run() and object parent\_chain is called



Parent chain calls function simple sequential chain which is an inbuilt function.

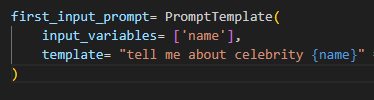
Here arguments are chain1 and chain2

Control goes to chain1



Inputs here are 1. LLM model 2. First\_input\_prompt 3. Output

Now control goes to first\_input\_prompt and with info name= Andy Samberg and template it calls promptTemplate() which accepts these input and generate prompt for language model



With all prompt and ip, it tuns LLMChain, which is nothing but library



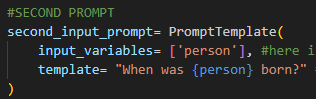
This query is run and returned.

Person’s name is set as output\_key, output\_key stores Andy Samberg

Now chain object has all these info and returned to parent chain.

Now chain2 is executed similarly.





Chain1 and chain 2 and made in sequence and stored in chains

Now Parent\_chain has information about Andy Samberg and his DOB.

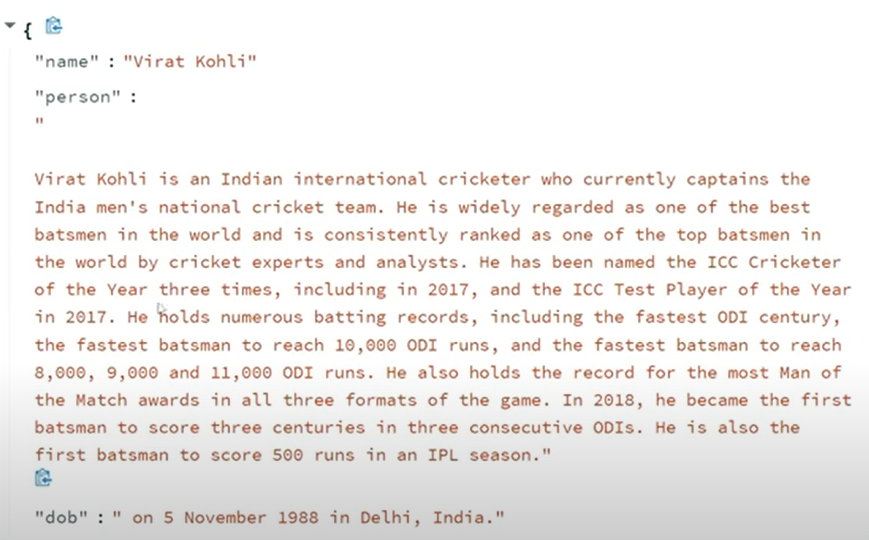
Control goes back to write() in steamlit and these info is printed in st UI.

Problem with simplesequential chain is, as it gets more input it holds the last input.

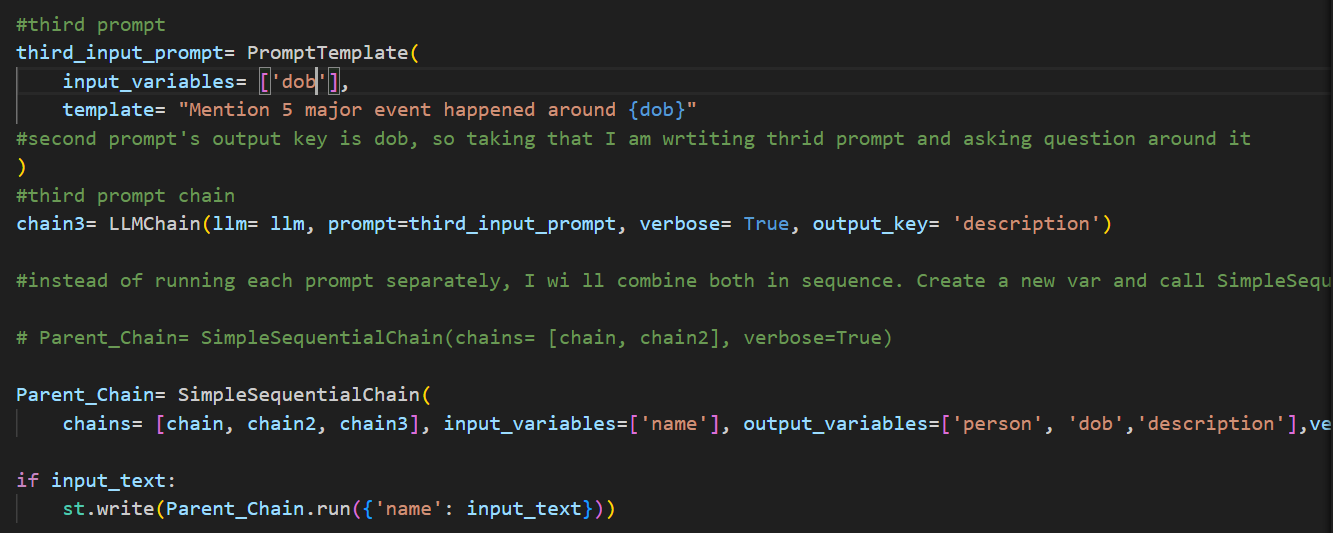
To show all the inputs, add argumnets input\_var and output\_var and pass in parent\_chain.

Instead of using run(input\_text), here I am passing key value pairs. That is entire information will be in form of json.

Now output would look like this

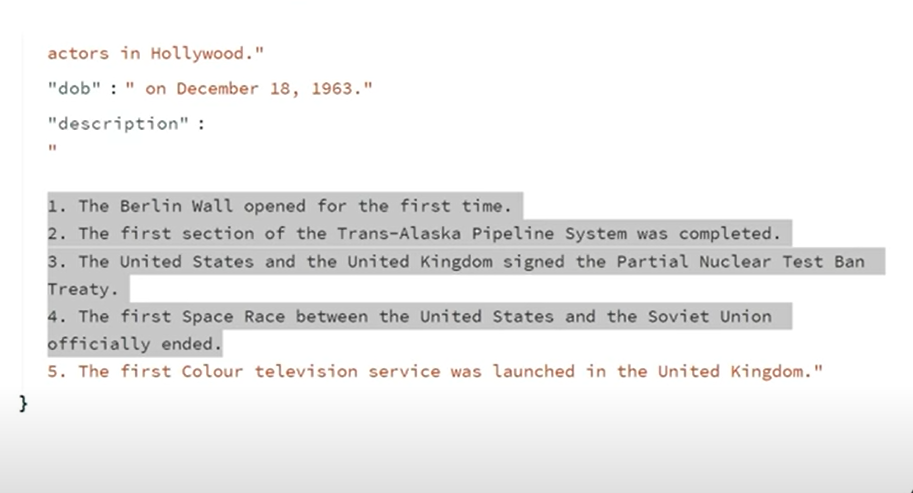


Third prompt



Now output would look like:

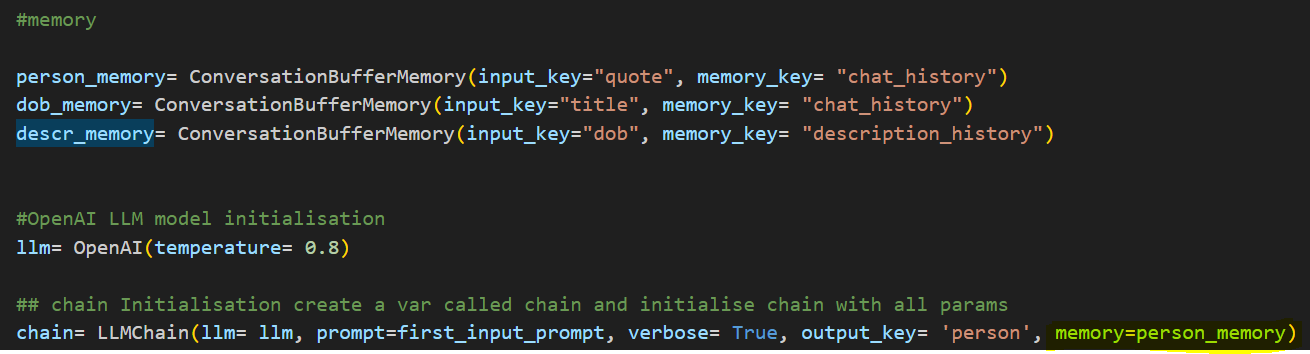
In this if I search other celeb, it also holds information of Virat Kohli which I searched first. It will be shown in terminal of VScode not in streamlit web app



Memory Buffer

Used for storing convo and to save all conversations.

Import conversationBufferMemory package

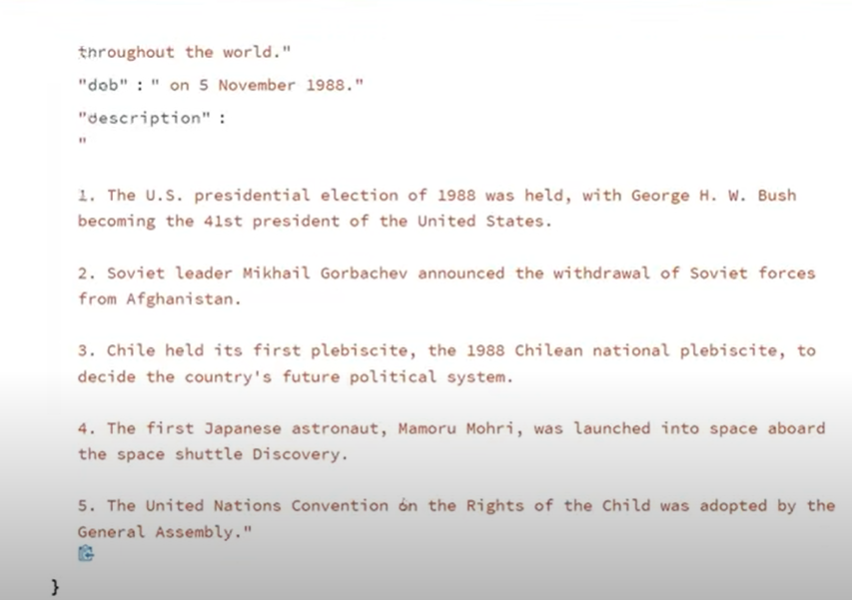


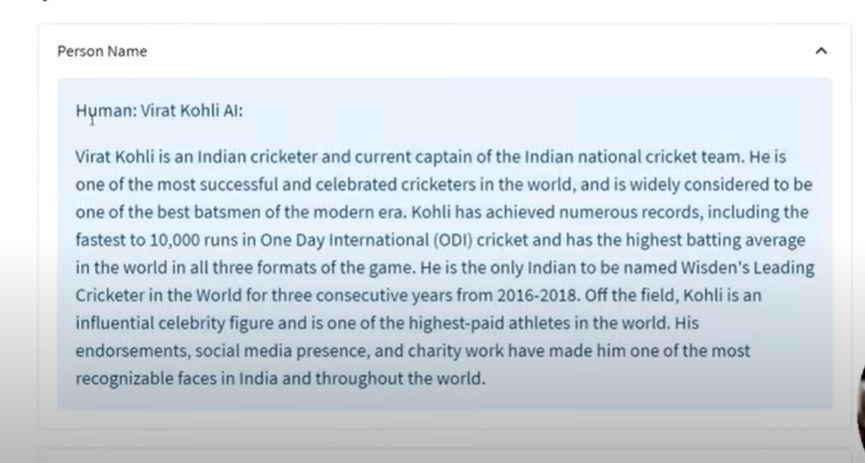
Create 3 objects for memory for person, dob and description. That is, for all the output keys.

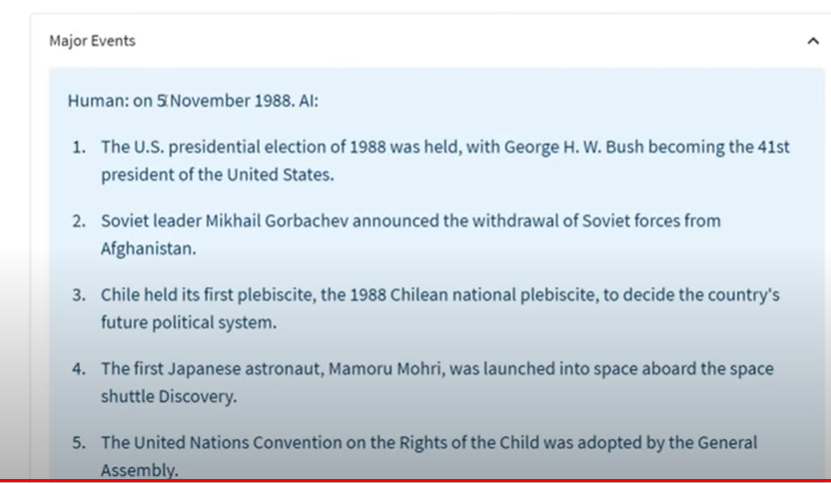
Pass input\_key, memory\_key

And pass new argument memory in chain, chain 1 and chain 2. Memory

Assign it to the memory objects created.







PROMPT ENGINEERING

<https://python.langchain.com/docs/modules/model_io/prompts/>

Prompt is the set of instructions or input provided by user to model. It helps in undetstanding the context and generate relevant responses, for answering questions or completing sentences and even in engaging in conversations.

Folder-> prompt engineering

Create a new jupyter file

Paste openai key