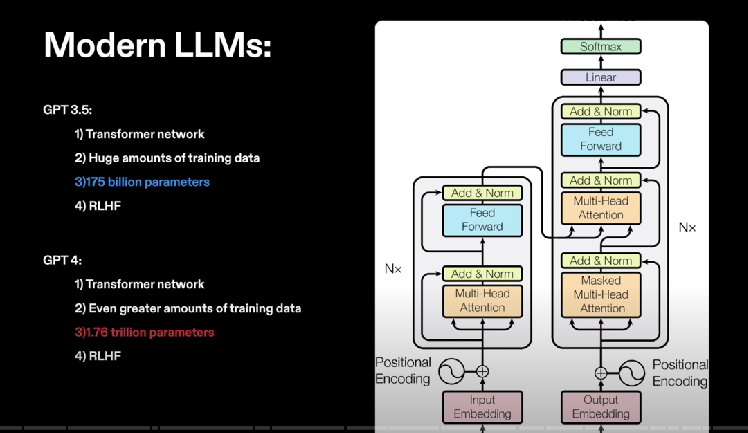
**Development with Large Language Models Tutorial – OpenAI, Langchain, Agents, Chroma**

**(** <https://www.youtube.com/watch?v=xZDB1naRUlk> **)**

LLM🡪 deep learning technique combined with huge amount of aligned to human values to create a reasoning engine.

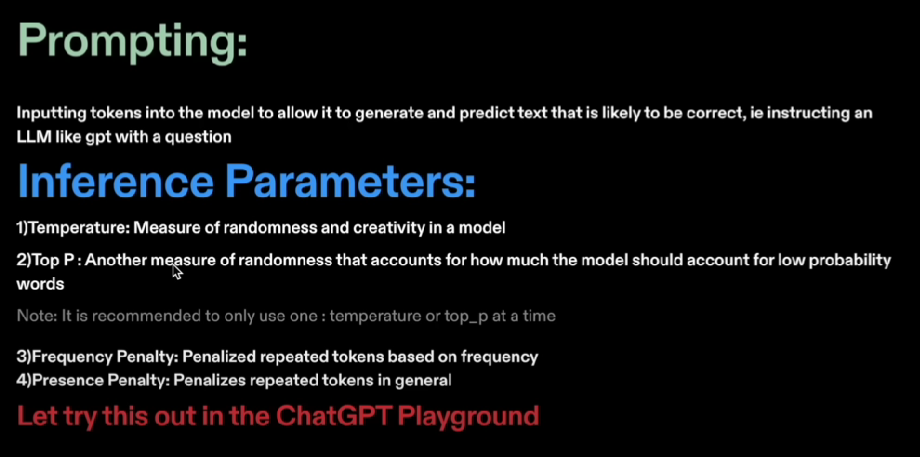


The parameters here are like weights between neurons.

**How llms work:**

1. Tokenization 🡪 breaking the sentence into vector if words. Words are assigned an unique number. There are 2 extra tokens also present. One [CLS] for start of sentence and one [SEP] for end of sentence.
2. Model/neural network/math of neural network 🡪 Each input goes into the model. Initially all the layers are like random numbers, first we predict the output using these random numbers, this is pretty much just guessing what word will be the next in the sequence. After that we take the actual word which should’ve been there and calculate the difference between the predicted and obtained word and adjust the parameters according to the error/difference obtained. This is one training set. This happens a lot of times and this is how the model gets good. The model is given millions of such sentences, this is the training data and we try to predict words in that sentence.
3. Reasoning / human values 🡪 predicted output is given back to the sentence and then we try to predict again.
4. Fine tuning 🡪 training on custom data, big data training not required. Vector dbs is more efficient than fine tuning for data retrieval.
5. RHLS(reinforcement learning from human feedback) 🡪 when we scrape data from internet to train the model, the data can be biased, so we remove this biasing from the data and then fine tune the model.

Ways of getting output from llms:



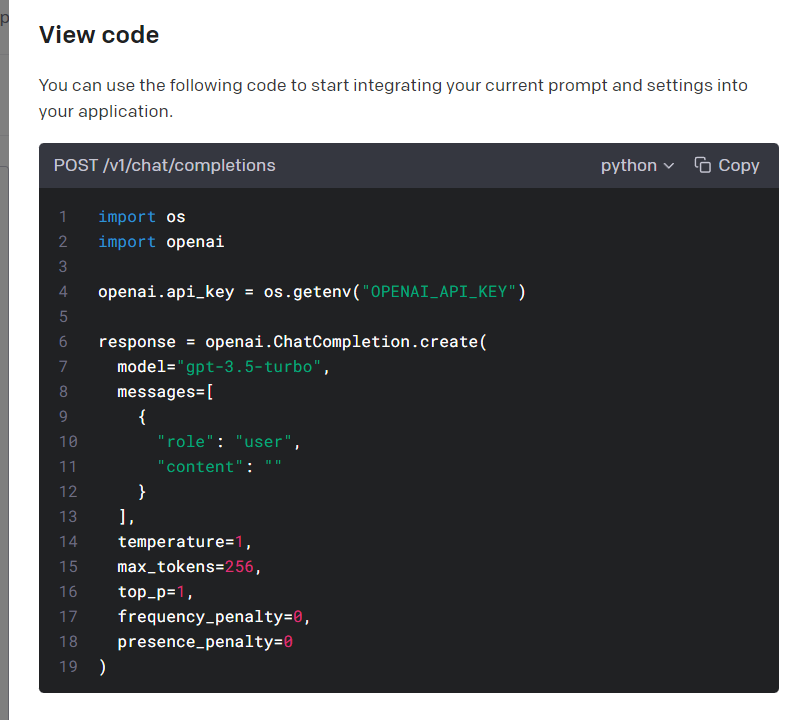
Chatgpt playground: using because it has more control parameters and closer to api working.

**Parameters that we can change in the playground:**

1. Temprature: measure of randomness/creativity. We should keep the temprature as 0 when we want to understand or get output from a given data. But when we want some creativity in predictiveness then we should increase the temprature.
2. Maximum length
3. Top P: this is also used to control the creativity of the model. Use either temprature or top P at a time.
4. Frequency penalty: if we want the model to give different answer even at same question then we use this. penalizes according to the frequency of the word
5. Presence penalty: penalizes on the basis if the word exists or not.

Streaming – predicting text as it goes instead of predicting the whole text and then displaying it.

We can use this “view code” option to view how to integrate the code with the parameter settings of the playground:



**Installing openAI:**

Download openai to desktop:

A black background with white text

Description automatically generated

Upgrade openai:



 Will let us use chatgpt

Create an api key and use it in your code. Can be disabled anytime.

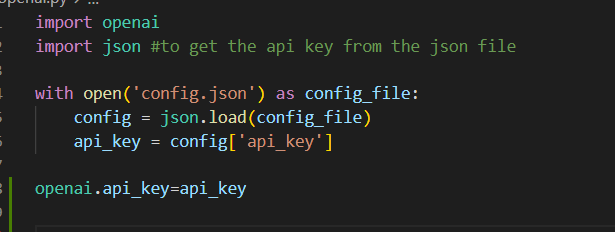
I have created a json file and stored the api key there. The json file is not being tracked by github as it is kept in a gitignore file so the api key is safe.

Storing key in the config.json file

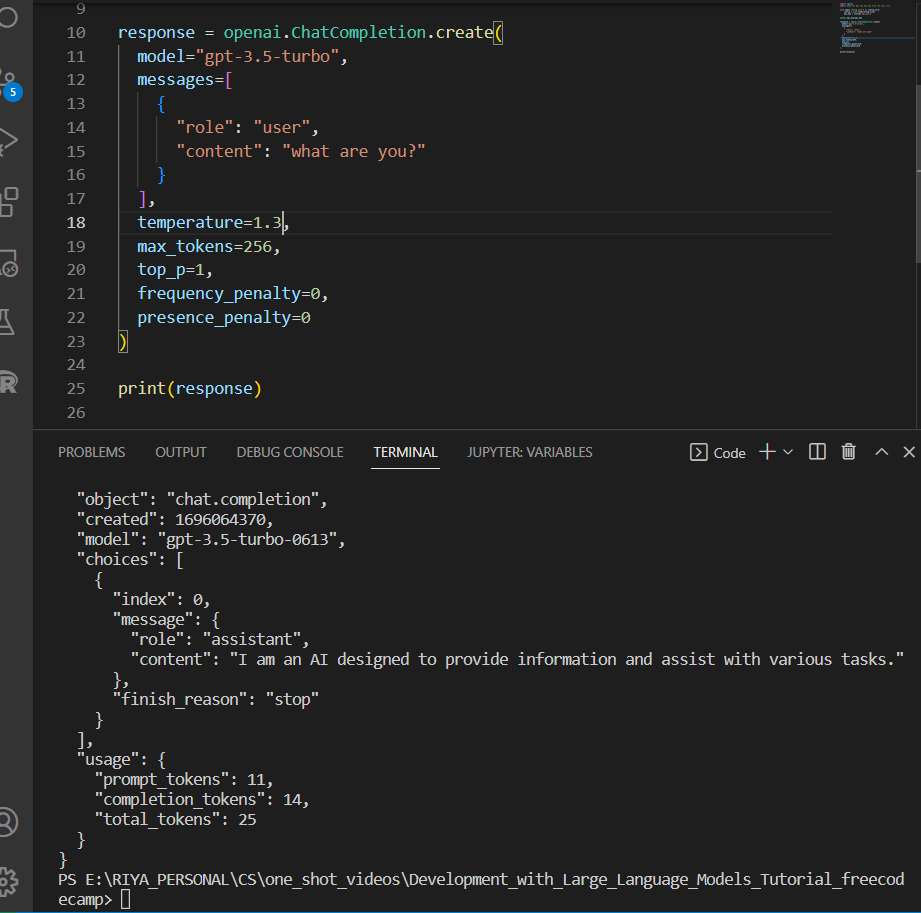
A screenshot of a computer program

Description automatically generated

Retrieving the key.



Simple question answer by user and assistant:



To print only required stuff from the returned json data:

A black background with white text

Description automatically generated

We can even give multiple inputs in the form – role: user, message: “”, then role: assistant, message:””



PROJECT 1: cloning the chatgpt interface with custom personality.

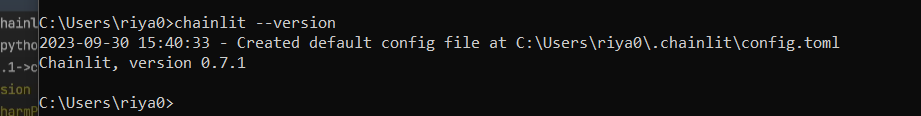
We will do this using chainlit, it is used to make online pages very easily. Chainlit is similar to streamlit but it is for LLM’s. Chainlit is a python open source package.

We will clone a user interface from chainlit and use it as out frontend.

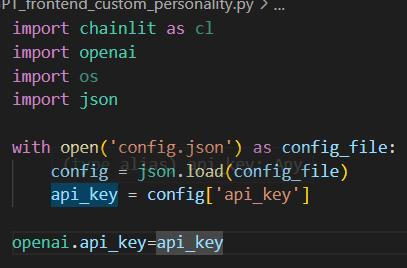
Install chainlit:

A black screen with white text

Description automatically generated



Starting code:



Making a simple user interface:

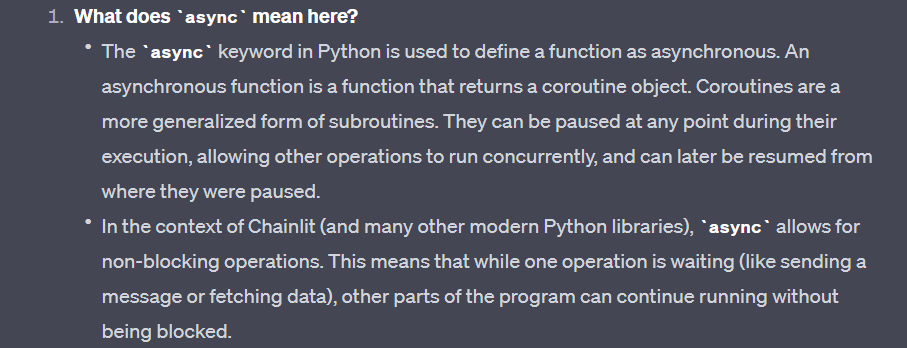
Chainlit basics: chianlit will host the page on a local port of the computer. Also chainlit.md file contains some markdown which appears at the start of the chianlit website that we’ll make, its not important we can edit or delete it.

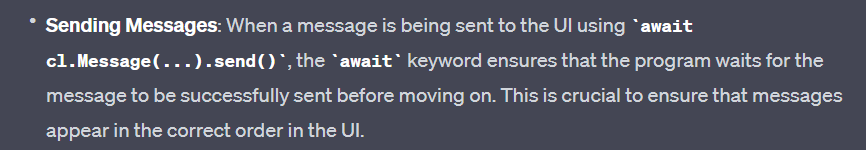
The basic syntax is:

A computer screen with text on it

Description automatically generated

Async and await meaning:



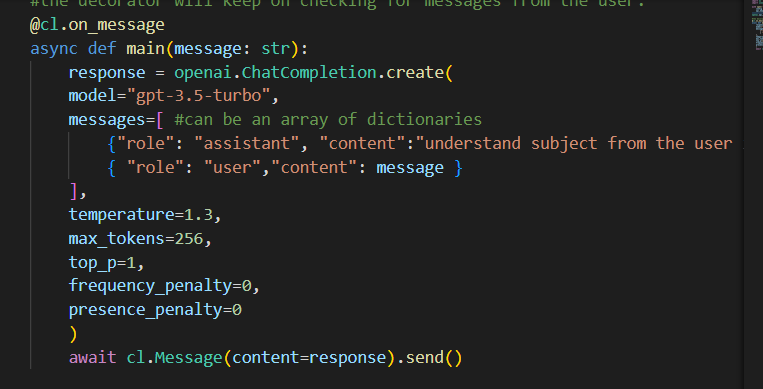


To run the chainlit file:



The w flag is used to make the site continue looking for changes, saving the file will preload the app.

Trial 1:

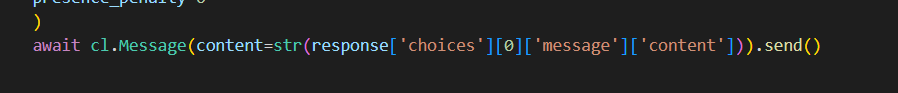


This code because the response was not edited to only show the content displays the whole json in the chat like this:

A computer screen shot of a computer code

Description automatically generated

By changing the send line to this:



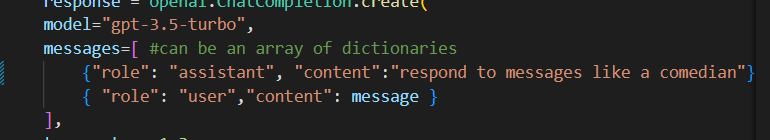
A screenshot of a white text

Description automatically generated

We can get a string response: but this took around 4-5 seconds to load.

* If I change the max\_tokens then only those numbers of tokens will be displayed n the reply/response message.

Giving a specific personality to the model:



A close-up of a message

Description automatically generated

Limitations of this method:

1. No streaming involved – no live output prediction was there, I would wait for some time and then only I could see the whole output at once.
2. No “generating” message to make the user know that yes, the model is working.
3. User does not know what kind of assistant is running on the backend.

Langchain: we can use this library to resolve all these issues. Tis is the most powerful tool with multiple functionalities to interact with gpt api.