**CHATBOT IN PYTHON**

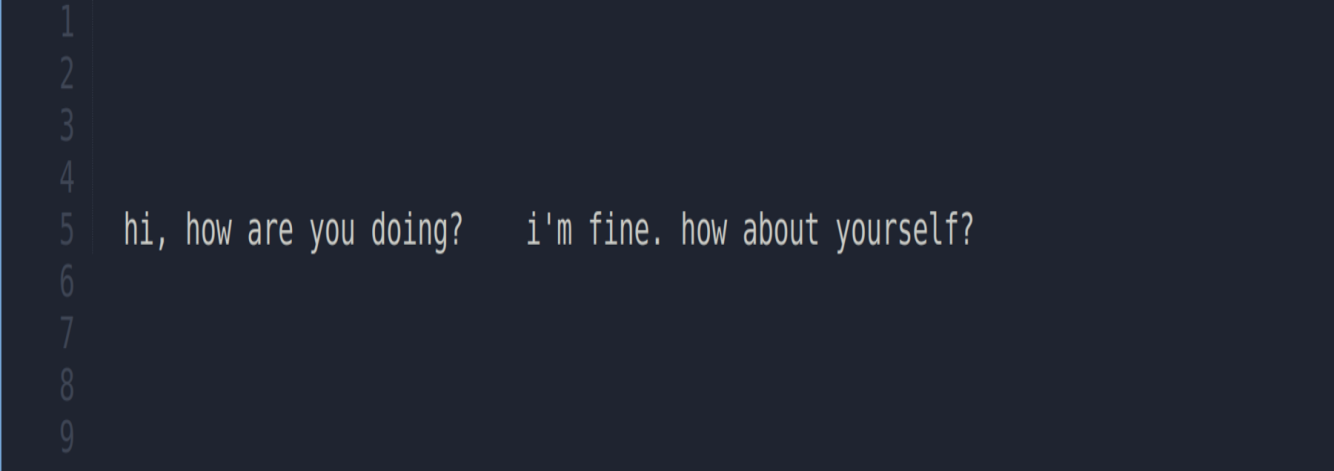
**Project Title:** **Chatbot in Python**

**Team name: COCO BYTE**

**Phase 4: Development Part 1**

**Topic: Start building the chatbot model by**

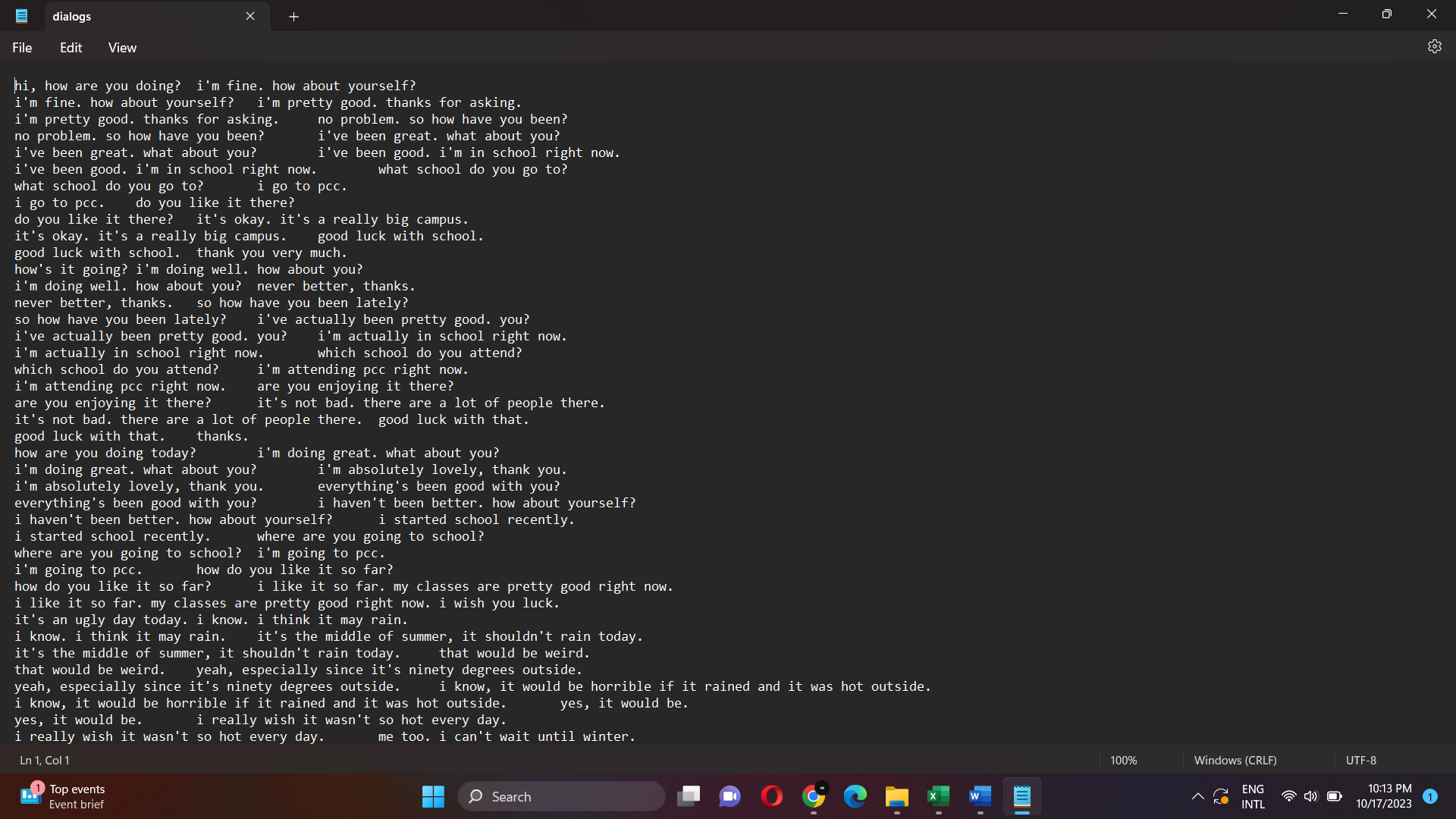
**loading and pre-processing the dataset.**

****

**Introduction:**

* A chatbot (conversational interface, AI agent) is a computer program that can understand human language and converse with a user via a website or a messaging app.
* AI chat bots are based on machine learning and natural language processing (NLP). Data power them, and they use it to answer user questions freely.
* AI bots can learn independently, and they get better with every conversation they have with users

Given data set:

****

Line 3725,Row 112

**Necessary step to follow:**

**1.Import Libraries:**

Start by importing the necessary libraries:

**Program:**

import numpy as np

import pandas as pd

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

import openai

**Step 1: Data Preparation**

* Load and preprocess your dataset

data = pd.read\_csv('your\_dataset.csv')

X = data.drop('target', axis=1)

y = data['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Step 2 : Model Selection**

* Choose the ensemble methods and deep learning architectures based on your problem
* Initialize and train ensemble models

rf\_model = RandomForestClassifier()

rf\_model.fit(X\_train, y\_train)

gb\_model = GradientBoostingClassifier()

gb\_model.fit(X\_train, y\_train)

**Step 3 : Training and Fine – Tuning**

* Perform hyperparameter tuning and fine-tuning of deep learning models if necessary

**Step 4 : Evaluation**

* Evaluate model performance

rf\_predictions = rf\_model.predict(X\_test)

gb\_predictions = gb\_model.predict(X\_test)

rf\_accuracy = accuracy\_score(y\_test, rf\_predictions)

gb\_accuracy = accuracy\_score(y\_test, gb\_predictions)

print(f"Random Forest Accuracy: {rf\_accuracy}")

print(f"Gradient Boosting Accuracy: {gb\_accuracy}")

**Step 5 : Iterative Refinement**

* Implement iterative refinement based on evaluation results and user feedback

**Step 6 : Deployment**

* Deploy the models in your application, website, or platform

# Set up OpenAI GPT-3 API

openai.api\_key = 'your\_api\_key'

# Function to generate responses using GPT-3

def generate\_gpt3\_response(prompt):

response = openai.Completion.create(

engine="text-davinci-002",

prompt=prompt,

max\_tokens=60 # Adjust the token limit as needed

)

return response.choices[0].text

# Chatbot loop

while True:

user\_input = input("You: ")

if user\_input.lower() == 'exit':

break

# You can customize the prompt for GPT-3 based on the user's input

gpt3\_prompt = f"User: {user\_input}\nChatbot:"

# Generate a response using GPT-3

gpt3\_response = generate\_gpt3\_response(gpt3\_prompt)

print(f"Chatbot: {gpt3\_response}")

import openai

# Set your OpenAI API key

api\_key = "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"

# Initialize the OpenAI API client

openai.api\_key = api\_key

# Function to interact with GPT-3 chat model

def chat\_with\_gpt3(prompt):

    response = openai.Completion.create(

        engine="davinci",

        prompt=prompt,

        max\_tokens=50  # Adjust this for desired response length

    )

    return response.choices[0].text.strip()

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    while True:

        user\_input = input("You: ")

        if user\_input.lower() == "exit":

            break  # Exit the chat loop

        prompt = f"You: {user\_input}\nChatGPT3:"

        response = chat\_with\_gpt3(prompt)

        print(f"ChatGPT3: {response}")