**Generative AI Labcourse Assignments**

**Assignment 1 Programs Based on Large Language Models (LLM)**

**SET A**

**1]Write a Python program using the transformers library to generate a short   
 continuation to the input prompt “Hello, I am learning LLM”.**

**Soln:**

**pip install transformers**

from transformers import pipeline

# Load a simple text generation model

model = pipeline("text-generation")

# Generate text

output = model("Hello, I am learning LLM", max\_length=30)

# Print the result

print(output[0]["generated\_text"])

**Output:**

Hello, I am learning LLM and it is a very exciting field to explore.

As I practice more

**Explanation:** This imports the pipeline() function from the Hugging Face transformers library(open-source Python library).The pipeline function makes it very easy to use pre-trained AI/LLM models.It hides all the complex steps (loading model, tokenizer, weights, etc.).2)model = pipeline("text-generation")

This line creates a text-generation model pipeline.  
"text-generation" means we want to generate text using a pre-trained LLM.  
No model name is given, so it loads a default small text-generation model automatically.The loaded model learns to predict and continue sentences.

3) output = model("Hello, I am learning LLM", max\_length=30)

This line sends text to the model for generation.  
"Hello, I am learning LLM" is the input prompt.  
max\_length=30 means the generated output will have a maximum of 30 tokens (words/sub-words).The model reads the prompt and produces the next words.

4)output[0] → fetches the first result from the list.  
["generated\_text"] → gets the actual text generated by the model.  
print() → displays the generated text on the screen.

output → list  
output[0] → dictionary  
"generated\_text" → key inside the dictionary

We are accessing first element 0 from the key generated\_text

**2]Write a Python program to load a pre-trained Large Language Model (LLM) using the transformers library to generate a simple text continuation for a given prompt.**

from transformers import pipeline

# Load pretrained GPT-2 text generation model

generator = pipeline("text-generation", model="gpt2")

prompt = "Machine learning is"

result = generator(prompt, max\_length=40, num\_return\_sequences=1)

print("Generated Output:")

print(result[0]["generated\_text"])

**Output:**

Machine learning is becoming one of the most important technologies…

Explanation:"text-generation" tells the pipeline that we want a model that continues/generates text. model="gpt2" specifies which pre-trained model we want to load.GPT-2 is a well-known Large Language Model (LLM) trained to generate human-like text.The pipeline automatically downloads and loads GPT-2 the first time you run it.

prompt = "Machine learning is"

This defines a text prompt (starting sentence).

The LLM will use this prompt and generate the next words.

result = generator(prompt, max\_length=40, num\_return\_sequences=1)

This line generates text using the GPT-2 model.

max\_length=40 → maximum total number of tokens (words/characters) in the output including the prompt.

num\_return\_sequences=1 → generate only one output text result.

num\_return\_sequences tells the model how many different text outputs you want it to generate for the same prompt.

The output is stored in the variable result.

The result is returned as a list of dictionaries.

**3]Write a program to extract an answer from a paragraph using a question answering LLM.**

from transformers import pipeline

# Load the Question-Answering model

qa\_model = pipeline("question-answering")

# Paragraph (context)

context = """The Sun is the center of our Solar System. It is a huge ball of hot gases.Earth revolves around the Sun in 365 days.

"""

# Question

question = "How many days does Earth take to revolve around the Sun?"

#question = input("Enter your question: ")

# Get answer from LLM

result = qa\_model(question=question, context=context)

print("Answer:", result["answer"])

Output:

Answer: 365

question = "Describe Sun"

Answer: It is a huge ball of hot gases

**4]Write a program to perform sentiment analysis using a Large Language Model (LLM)**

from transformers import pipeline

sentiment = pipeline("sentiment-analysis")

text = "I love using AI tools, they are amazing!"

result = sentiment(text)

print(result)

**Output:**

[{'label': 'POSITIVE', 'score': 0.9998809099197388}]

**Output:**

text = "I don't like using AI tools, they are not amazing!"

[{'label': 'NEGATIVE', 'score': 0.9997798800468445}]

**5]Write a program to perform sentiment analysis on Multiple Sentences Using an LLM.**

from transformers import pipeline

# Load the sentiment analysis model

sentiment\_analyzer = pipeline("sentiment-analysis")

# List of sentences to analyze

sentences = [

"The movie was amazing and inspiring!",

"I did not like the food at the restaurant.",

"The weather today is okay."

]

# Analyze sentiment for each sentence

results = sentiment\_analyzer(sentences)

# Display results

for i, result in enumerate(results):

print(f"Sentence: {sentences[i]}")

print(f"Sentiment: {result['label']}")

print(round(result['score'], 3))

print()

Output:

Sentence: The movie was amazing and inspiring!

Sentiment: POSITIVE

1.0

Sentence: I did not like the food at the restaurant.

Sentiment: NEGATIVE

0.999

Sentence: The weather today is okay.

Sentiment: POSITIVE

1.0

**6]Write a program to accept input from user and correct the grammar by using LLM**

from transformers import pipeline

# Load a text2text-generation model for grammar correction

grammar\_corrector = pipeline("text2text-generation", model="grammarly/coedit-large")

# Input sentence with grammatical errors

sentence = input("Enter a sentence to correct grammar: ")

# Get corrected sentence

corrected = grammar\_corrector(sentence)[0]['generated\_text']

# Display results

print("\nOriginal Sentence: ", sentence)

print("Corrected Sentence:", corrected)

**Output:**

Enter a sentence to correct grammar: He go to school yesterday.

Original Sentence: He go to school yesterday.

Corrected Sentence: He went to school yesterday.

**7]Write a program that uses a zero-shot classification LLM to classify a sentence into predefined categories.**

from transformers import pipeline

classifier = pipeline("zero-shot-classification")

text = "The Indian cricket team won the match by 5 wickets."

labels = ["Sports", "Politics", "Technology"]

result = classifier(text, candidate\_labels=labels)

print(result)

**Output:**

**{'sequence': 'The Indian cricket team won the match by 5 wickets.', 'labels': ['Sports', 'Technology', 'Politics'], 'scores': [0.9892831444740295, 0.00825184490531683, 0.002464990597218275]}**

**text = "AI is an emerging field."**

**{'sequence': 'The AI is emerging field.', 'labels': ['Technology', 'Sports', 'Politics'], 'scores': [0.9671456813812256, 0.018400633707642555, 0.014453704468905926]}**

**8)Write a program that uses an LLM to correct the spelling of a sentence provided by the user.**

from transformers import pipeline

# Load a model fine-tuned for grammar/spelling correction

corrector = pipeline("text2text-generation",model="vennify/t5-base-grammar-correction")

# Take input from the user

sentence = input("Enter a sentence with spelling mistakes: ")

# Correct the sentence

corrected = corrector("correct: " + sentence)[0]['generated\_text']

# Show results

print("Original Sentence: ", sentence)

print("Corrected Sentence:", corrected)

**9]Write a Python program using the sentence-transformers library to find the semantic similarity between the sentences given by the users.Display the similarity score.**

from sentence\_transformers import SentenceTransformer, util

model = SentenceTransformer('all-MiniLM-L6-v2')

s1 = input("Enter first prompt:")

s2 = input("Enter second prompt:")

emb1 = model.encode(s1)

emb2 = model.encode(s2)

similarity = util.cos\_sim(emb1, emb2)

print("Similarity Score:", similarity.item())

Here **SentenceTransformer** is used to load a pre-trained model that converts sentences into numerical vectors and util is used for providing utility functions, such as cosine similarity.

Calculates **cosine similarity** between the two embeddings.  
Cosine similarity measures the **angle** between two vectors.

1.0 → very similar meaning  
0.0 → unrelated  
-1.0 → opposite meaning (rare in text)

**10]Write a program that uses a translation LLM to convert English text into Hindi.**

from transformers import pipeline

# Load translation model (English → Hindi)

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-en-hi")

# Take English text from user

text = input("Enter text in English: ")

# Translate to Hindi

result = translator(text)

# Display output

print("Translated Text (Hindi):", result[0]['translation\_text'])

**Output:**

**Enter text in English: how are you**

**Translated Text (Hindi): आप कैसे हैं**

**11]Write a program that uses a translation LLM to convert English text into Marathi.**

from transformers import pipeline

# Load English → Marathi translation model

translator = pipeline("translation", model="Helsinki-NLP/opus-mt-en-mr")

# Take English text from user

text = input("Enter text in English: ")

# Translate to Marathi

result = translator(text)

# Display output

print("Translated Text (Marathi):", result[0]['translation\_text'])

**Output:**

**Enter text in English: how are you**

**Translated Text (Marathi): तुम्ही कसे आहात?**

**12]Write a Python program using the BERT-based fill-mask model to predict missing words in a sentence. Accept a sentence from the user and display the predicted words along with their probability scores.**

from transformers import pipeline

# Load BERT explicitly

fill\_mask = pipeline("fill-mask",model="bert-base-uncased")

# Use BERT's mask token: [MASK]

text = "Artificial Intelligence is an emerging [MASK]."

output = fill\_mask(text)

print("BERT Fill-Mask Predictions:\n")

for pred in output:

print(pred["token\_str"], "→", pred["score"])

**Output:**

**field → 0.6136760115623474**

**discipline → 0.18116749823093414**

**technology → 0.13998444378376007**

**science → 0.018457191064953804**

**area → 0.006910332944244146**

**Assignment 2 Programs Based on Summarization**

**1]Write a Python program that prints a short summary by selecting the first two sentences of a paragraph.**

text = """Generative AI creates new content such as text and images.

It learns patterns from data.It is used in chatbots and image creation tools."""

sentences = text.split(".")

summary = sentences[0] + "." + sentences[1] + "."

print("Summary:", summary)

* splits the input paragraph (text) into a list of strings wherever a (.) is found

**Output:**

Summary: Generative AI creates new content such as text and images.

It learns patterns from data.

**2]Write a program to count the number of sentences in a paragraph and print a short summary containing only the first sentence.**

text = input("Enter a paragraph:\n")

# Split and remove empty strings

sentences = [s for s in text.split(".") if s.strip()]

# First sentence summary

summary = sentences[0] + "."

print("Number of sentences =", len(sentences))

print("Summary =", summary)

Explanation:

* splits the input paragraph (text) into a list of strings (potential sentences) wherever a period (.) is found
* The strip() method removes leading/trailing whitespace from each potential sentence (s).

**Output:**

Enter a paragraph:

Gen AI creates new content.It learns patterns from data. It is used in chatbots and image creation tools.

Number of sentences = 3

Summary = Gen AI creates new content.

**3] Write a Python program using the SentenceTransformer library to perform extractive text summarization. The program should take a paragraph as input, split it into sentences, generate sentence embeddings using a pre-trained model, extract any two important sentences, and display them as a summary**

from sentence\_transformers import SentenceTransformer, util

import numpy as np

# Load model

model = SentenceTransformer("all-MiniLM-L6-v2")

# Take input from user

print("Enter a paragraph (press Enter twice to finish):")

lines = []

while True:

line = input()

if line == "":

break

lines.append(line)

text = "\n".join(lines)

# Split into sentences

sentences = text.split("\n")

# Generate embeddings

sentence\_embeddings = model.encode(sentences)

paragraph\_embedding = model.encode([text])

# Find similarity scores

scores = util.cos\_sim(sentence\_embeddings, paragraph\_embedding)

# Select any two important sentences

top\_two = np.argsort(scores.flatten())[-2:]

summary = [sentences[i] for i in top\_two]

# Print summary

print("\nSummary:")

print(" ".join(summary))

**Output**

Enter a paragraph (press Enter twice to finish):

Transformers are powerful models for NLP tasks.

They can generate embeddings for sentences.

These embeddings help in text summarization.

Extractive summarization selects key sentences.

Summary:

These embeddings help in text summarization. Transformers are powerful models for NLP tasks.

scores.flatten() → turns similarity matrix into a 1D array.

np.argsort(...) → returns indices sorted by similarity.

[-2:] → selects the **top 2 most similar sentences**.

Uses the indices to extract the **two most important sentences**.

**Output2:**

Machine learning is a branch of artificial intelligence.

It enables systems to learn from data.

Transformer models are widely used in generative AI.

They are effective for tasks like summarization and translation.

Summary:

Machine learning is a branch of artificial intelligence. Transformer models are widely used in generative AI.

**3]Write a Python program to remove stopwords and print only meaningful words as a simple summary.**

**pip install nltk or python -m pip install nltk**

import torch

from transformers import BertTokenizer

import nltk

from nltk.corpus import stopwords

# Download stopwords once

nltk.download('stopwords')

# Load BERT tokenizer

tokenizer = BertTokenizer.from\_pretrained("bert-base-uncased")

text = input("Enter sentence:\n")

# Tokenize input

tokens = tokenizer.tokenize(text.lower())

# Use stopwords only for final filtering (GEN AI identifies tokens)

stop\_words = set(stopwords.words('english'))

# Keep meaningful words

meaningful\_words = [word for word in tokens if word not in stop\_words]

print("\nMeaningful Words:")

print(" ".join(meaningful\_words))

**Output:**

Generative AI is a type of artificial intelligence that creates useful content

Summary:

generative ai type artificial intelligence creates useful content

**4]Write a program to summarize a paragraph by removing repeated words.**

from transformers import BertTokenizer

# Load BERT tokenizer

tokenizer = BertTokenizer.from\_pretrained("bert-base-uncased")

# Input paragraph

text = input("Enter a paragraph:\n")

# Tokenize text

tokens = tokenizer.tokenize(text.lower())

# Remove duplicate words while preserving order

unique\_words = []

for word in tokens:

if word not in unique\_words:

unique\_words.append(word)

# Print summarized output

print("\nSummarized Paragraph (After Removing Repeated Words):")

print(" ".join(unique\_words))

**Output:**

AI is powerful and AI is used in many AI applications.

Summary: AI is powerful and used in many applications.

**5]Write a program that summarizes a long article into 3 lines.**

from transformers import pipeline

# Load summarization pipeline

summarizer = pipeline("summarization")

# Long article

article = """

Artificial Intelligence (AI) has rapidly transformed modern technology.

It is used in healthcare, finance, education, transportation, and many more sectors.

AI systems can analyze large datasets, detect patterns, and make predictions faster than humans.With advancements in machine learning and deep learning, AI is expected to bring even more automation and innovation. However, ethical concerns such as bias, privacy, and job displacement must be addressed to ensure responsible and beneficial use of this technology.

"""

# Generate summary (approx 3 lines)

summary = summarizer(article, max\_length=90,min\_length=60)

print("=== 3-Line Summary ===")

print(summary[0]['summary\_text'])

**Output:**

Artificial Intelligence (AI) has rapidly transformed modern technology . It is used in healthcare, finance, education, transportation, and many more sectors . However, ethical concerns such as bias, privacy, and job displacement must be addressed to ensure responsible and beneficial use of this technology . AI systems can analyze large datasets, detect patterns, and make predictions faster than humans .

**6]Write a Python program to perform abstractive text summarization using a pre-trained generative AI model.**

from transformers import pipeline

summarizer = pipeline("summarization", model="t5-small")

text = """

Machine learning enables computers to learn patterns from data.

Generative AI is a branch of ML that focuses on creating new content.

Modern generative models such as GPT and Stable Diffusion have transformed industries by enabling automation and creativity.

"""

summary = summarizer(text, max\_length=40, min\_length=20)

print(summary[0]['summary\_text'])

**Output:**

Generative AI is a branch of ML that focuses on creating new content . modern generative models such as GPT and Stable Diffusion have transformed industries .

**Pipeline is abstractive summarization**

**from transformers import pipeline**

**# Load a pre-trained summarization model (abstractive)**

**# 'facebook/bart-large-cnn' is a good choice for abstractive summarization**

**summarizer = pipeline("summarization", model="facebook/bart-large-cnn")**

**# Example text to summarize**

**text\_to\_summarize = """**

**Artificial Intelligence (AI) is a rapidly expanding field that has revolutionized various aspects of modern life. It encompasses machine learning, deep learning, natural language processing, and computer vision. AI systems are designed to perform tasks that typically require human intelligence, such as problem-solving, learning, decision-making, and understanding language. From powering virtual assistants and recommendation engines to driving autonomous vehicles and assisting in medical diagnostics, AI's applications are diverse and growing. While offering immense potential for efficiency and innovation, it also raises ethical considerations regarding data privacy, bias, and job displacement. The ongoing research in AI continues to push the boundaries of what machines can achieve.**

**"""**

**# Perform abstractive summarization**

**# max\_length and min\_length control the length of the generated summary**

**summary = summarizer(text\_to\_summarize, max\_length=100, min\_length=30, do\_sample=False)**

**# Print the original text and the generated summary**

**print("Original Text:\n", text\_to\_summarize)**

**print("\nAbstractive Summary:\n", summary[0]['summary\_text'])**

**Output**

Artificial Intelligence (AI) is designed to perform tasks that typically require human intelligence. These include problem-solving, learning, decision-making, and understanding language. From powering recommendation engines to driving autonomous vehicles, AI's applications are diverse and growing.

OR

Artificial Intelligence (AI) is designed to perform tasks that typically require human intelligence. From powering recommendation engines to driving autonomous vehicles, AI's applications are diverse and growing.

**6]Write a Program to Summarize Story Text**

from transformers import pipeline

# Load summarization model

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# Take story input from the user

print("Enter a story to summarize:")

story\_text = input()

# Generate summary

summary = summarizer(story\_text, max\_length=60, min\_length=15)

print("\n--- Original Story ---")

print(story\_text)

print("\n--- Summary ---")

print(summary[0]['summary\_text'])

**Output**

Enter a story to summarize:

Once upon a time, in a small village, there lived a kind farmer named Ramesh. He had a small farm where he grew vegetables and fruits. One day, a young boy came to him asking for food. Ramesh shared his food generously. The boy was grateful and later helped the farmer save his crops from a flood. The village praised Ramesh for his kindness.

--- Original Story ---

Once upon a time, in a small village, there lived a kind farmer named Ramesh. He had a small farm where he grew vegetables and fruits. One day, a young boy came to him asking for food. Ramesh shared his food generously. The boy was grateful and later helped the farmer save his crops from a flood. The village praised Ramesh for his kindness.

--- Summary ---

Ramesh shared his food generously with a young boy. The boy was grateful and later helped the farmer save his crops from a flood.

**7]Write a Program to Summarize Chat Conversations Using LLM.**

from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

print("Enter chat conversation (Press ENTER twice to finish):")

chat\_lines = []

while True:

line = input()

if not line:

break

chat\_lines.append(line)

chat\_text = " ".join(chat\_lines) # better than newline for BART

result = summarizer(chat\_text, max\_length=80, min\_length=40,do\_sample=False

)[0]["summary\_text"]

print("\nSummary:")

print(result)

**Output:**

Enter chat conversation (Press ENTER twice to finish):

User: Hi, I need help with my internet connection.

Agent: Sure, can you tell me what issue you are facing?

User: The speed is very slow since yesterday.

Agent: Have you tried restarting the router?

User: Yes, but nothing changed.

Agent: Okay, I will run a quick line test. Please wait.

User: Sure.

Agent: I found an issue in your connection. A technician will visit your home today.

User: Thank you!

Summary:

User: The speed is very slow since yesterday. Have you tried restarting the router? User: Yes, but nothing changed. Agent: I found an issue in your connection. A technician will visit your home today.

**Output:**

Enter chat conversation (Press ENTER twice to finish):

Arya: Hi, I need help with my shopping

Bushra: Sure, can you tell me what issue you are facing?

Arya: I dont know the areas from where to shop

Bushra: Have you tried searching them

Arya: Yes, but confused

Bushra: Okay, I will come with you tomorrow

Arya: Sure i will wait for you

Bushra: I will make a list of places for tomorrow.

Arya: Thank you!

Summary:

Arya asks Bushra for help with her shopping. Bushra says she will make a list of places for tomorrow. Arya thanks her for her help and waits for Bushra to return.

**8]Write a Python program to read text from a file and generate an abstractive summary using a pre-trained transformer model.**

from transformers import pipeline

summarizer = pipeline("summarization")

# Reading text file

text = open("sample.txt","r").read()

summary = summarizer(text, max\_length=50, min\_length=20, do\_sample=False)[0]['summary\_text']

print("\nSummary:\n", summary)

**Output:**

**Output:sample.txt**

The Generative AI Revolution: A New Era of Creation

Generative Artificial Intelligence (GenAI) marks a profound shift in technology, moving beyond the traditional AI focus on analysis and classification to the creation of novel content. This technology, which gained widespread public attention with the release of consumer-facing tools like ChatGPT and DALL-E in late 2022, can produce human-like text, unique images, music, and videos in response to simple user prompts.At its core, GenAI operates using sophisticated deep learning models, particularly large language models (LLMs) and diffusion models, which are trained on vast datasets of existing digital content. Through this training, the models learn underlying patterns, structures, and relationships within the data, allowing them to generate original outputs that mirror the style and coherence of human-created work. This process is not simply copying; it's a form of advanced pattern recognition and prediction, essentially guessing the most probable next word or pixel in a sequence to build a coherent output.The applications of GenAI span across numerous industries, promising massive boosts in productivity and innovation. In healthcare, it accelerates drug discovery by suggesting novel molecular structures and aids in analyzing medical images. For software development, it generates code snippets, automates testing, and translates between programming languages. In creative fields, it serves as a powerful assistant, helping artists and marketers brainstorm ideas, generate drafts, and create personalized content at scale, freeing humans to focus on higher-level strategy and creativity.

Despite its potential, the rise of GenAI also brings significant challenges and ethical concerns. Key issues include the potential for spreading misinformation through "deepfakes" and AI hallucinations (generating convincing but factually incorrect information). There are also legal dilemmas regarding data privacy and intellectual property, as models are trained on vast amounts of internet data, leading to copyright infringement lawsuits. Furthermore, the massive computational power required for training and running these models raises environmental and cost concerns, and the fear of job displacement is a legitimate societal issue that requires careful management and worker retraining efforts.

Ultimately, the relationship between humans and generative AI is evolving into a collaborative one. Rather than a replacement, GenAI acts as a powerful tool that augments human capabilities. Responsible development and a focus on human oversight remain critical to ensure that its benefits are harnessed while mitigating its inherent risks. As the technology continues to advance rapidly, its integration into daily life and work is inevitable, promising to reshape how we create, communicate, and solve problems.

**Summary:**

Generative Artificial Intelligence (GenAI) marks a profound shift in technology, moving beyond the traditional AI focus on analysis and classification to the creation of novel content . This technology can produce human-like text, unique images, music, and videos in response to simple user prompts . The applications of GenAI span across numerous industries, promising massive boosts in productivity and innovation .

**9]Write a Python program using a Generative AI Transformer model to summarize a story and display the summary in bullet points.**

from transformers import pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

story = """

Once upon a time in a small village, there lived a boy named Raju.

He loved to explore forests and spend time with animals.

One day, he found an injured bird and took it home.

Raju cared for the bird until it healed and could fly again.

The villagers appreciated his kindness, and Raju became known as a loving and helpful boy.

"""

summary = summarizer(story,max\_length=80,min\_length=40, do\_sample=False

)[0]["summary\_text"]

sentences = summary.split(". ")

print("Story Summary in Bullet Points:")

for s in sentences:

if s.strip():

print("•", s.strip())

**Output:**

Story Summary in Bullet Points:

• Raju cared for the bird until it healed and could fly again

• The villagers appreciated his kindness, and Raju became known as a loving and helpful boy

• He loved to explore forests and spend time with animals.

**10]Write a Python program to read a pdf file and generate a summary using a pre-trained transformer model.**

**# pip install PyPDF2**

import PyPDF2

from transformers import pipeline

# Load summarization pipeline (Generative AI model)

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# Input PDF file

pdf\_path = input("Enter name of pdf: ")

# Read PDF content

pdf\_file = open(pdf\_path, 'rb')

pdf\_reader = PyPDF2.PdfReader(pdf\_file)

text = ""

for page in pdf\_reader.pages:

page\_text = page.extract\_text()

if page\_text:

text += page\_text + " "

pdf\_file.close()

# Split text into chunks of ~1000 characters for long PDFs

chunks = [text[i:i+1000] for i in range(0, len(text), 1000)]

# Summarize each chunk

summaries = []

for chunk in chunks:

summary\_chunk = summarizer(chunk, max\_length=150, min\_length=50, do\_sample=False)

summaries.append(summary\_chunk[0]['summary\_text'])

# Combine all chunk summaries into final summary

final\_summary = " ".join(summaries)

# Display summarized output

print("\nGenerated Summary:")

print(final\_summary)

**# Install required libraries if not already installed:**

**# pip install PyPDF2 transformers torch**

**import PyPDF2**

**from transformers import pipeline**

**# Load summarization pipeline (Generative AI model)**

**summarizer = pipeline("summarization", model="facebook/bart-large-cnn")**

**# Input PDF file**

**pdf\_path = input("Enter name of pdf: ")**

**# Read PDF content**

**pdf\_file = open(pdf\_path, 'rb')**

**pdf\_reader = PyPDF2.PdfReader(pdf\_file)**

**text = ""**

**for page in pdf\_reader.pages:**

**page\_text = page.extract\_text()**

**if page\_text:**

**text += page\_text + " "**

**pdf\_file.close()**

**# Split text into chunks of ~1000 characters for long PDFs**

**chunks = [text[i:i+1000] for i in range(0, len(text), 1000)]**

**# Summarize each chunk**

**summaries = []**

**for chunk in chunks:**

**summary\_chunk = summarizer(chunk, max\_length=150, min\_length=50, do\_sample=False)**

**summaries.append(summary\_chunk[0]['summary\_text'])**

**# Combine all chunk summaries into final summary**

**final\_summary = " ".join(summaries)**

**# Display summarized output**

**print("\nGenerated Summary:")**

**print(final\_summary)**

Explanation

* Rb-binary mode
* Loops over each page:  
  Extracts the text using extract\_text()  
  Appends it to a single string text with a space  
  Closes the PDF file to free resources.

chunks = [text[i:i+1000] for i in range(0, len(text), 1000)]

Splits the text into chunks of ~1000 characters.

Without chunking, very long PDFs might throw an error or truncate the text.

Loops over all chunks.  
summarizer(chunk, max\_length=150, min\_length=50, do\_sample=False) → Generates a summary:  
max\_length=150 → Maximum words in the summary  
min\_length=50 → Minimum words in the summary  
do\_sample=False → Ensures deterministic output (same input → same summary)

Appends the summary of each chunk to summaries list.

**Output:**

Machine Learning (ML) is a foundational subset of artificial intelligence (AI) focusing on the development of algorithms that allow computers to learn from data and improve their performance over time without being explicitly programmed for every specific task. It enables systems to analyze large-scale data, identify hidden patterns, and make autonomous, data-driven predictions.

Core Fundamentals and Types of Machine Learning Supervised Learning: Algorithms are trained on labeled data, meaning the input data is paired with the correct output. The goal is to learn a mapping function from input to output to predict labels for new, unseen data.

Unsupervised Learning: Algorithms are given unlabeled data and must discover hidden structures, patterns, or relationships on their own, such as clustering similar data points together.

Reinforcement Learning: Agents learn to make decisions by performing actions in an environment and receiving rewards or penalties, aiming to maximize cumulative rewards.

Deep Learning: A specialized subset of ML using neural networks with multiple layers to model complex, high-level abstractions in data, often used for image recognition and natural language processing.

Key Algorithms and Techniques

Linear Regression: Used for predicting numerical values (regression) by finding the best-fit line through data points.

Logistic Regression: Used for binary classification tasks to determine the probability of a categorical outcome.

Decision Trees: A flowchart-like structure used for classification and regression tasks by mapping observations to outcomes.

K-Nearest Neighbors (KNN): A classification algorithm that classifies data points based on the classes of their closest neighbors.

Random Forest: An ensemble learning method that builds multiple decision trees to improve accuracy and reduce overfitting.

Support Vector Machines (SVM): Finds the optimal hyperplane that separates classes in a dataset.

Neural Networks: Computational models inspired by the human brain, crucial for complex pattern recognition.

The Machine Learning Life Cycle (7 Stages)

1. Data Collection and Preparation: Gathering relevant data and cleaning it for analysis.

2. Feature Engineering: Selecting and transforming variables to improve model performance.

3. Model Selection: Choosing the appropriate algorithm for the task.

4. Training: Fitting the model to the training dataset.

5. Validation: Tuning hyperparameters to optimize performance.

6. Testing: Evaluating the final model on unseen data to ensure accuracy.

7. Deployment & Monitoring: Implementing the model and continuously updating it.

Real-World Applications of Machine Learning

Healthcare: Detecting diseases, such as analyzing ultrasound images for nerve segmentation or identifying cancer cells.

Finance: Detecting fraudulent transactions and predicting stock prices.

E-commerce: Recommending products and analyzing customer sentiment to improve user experience.

Manufacturing: Predictive maintenance for machinery, reducing downtime by forecasting equipment failures.

Transportation: Optimizing logistics, such as predicting ride demand for services like Ola.

Natural Language Processing (NLP): Building chatbots, language translation, and text summarization tools.

Machine Learning Projects and Tools

Python Libraries: Scikit-learn (general ML), TensorFlow and Keras (deep learning), and Pandas (data manipulation).

Popular Projects: Sentiment analysis on Twitter, digit recognition (MNIST), house price prediction (Boston Housing), and customer segmentation.

Data Sources: Kaggle and the US government's data portal (Data.gov) provide datasets for building models.

Machine learning is essential for digital transformation, allowing organizations to automate, optimize, and innovate by extracting insights from data.

Enter name of pdf: example.pdf

Generated Summary:

Machine Learning (ML) is a subset of artificial intelligence (AI) focusing on the development of algorithms that allow computers to learn from data. Algorithms are given unlabeled data and must discover hiddenstructures, patterns, or relationships on their own. Deep Learning is a specialized subset of ML using neural networks with multiple layers to model complex, high -level abstractions in data. It is often used for image recognition and natural language processing. Key Algorithms and Techniques includeLinear Regression, Logistic Regression

and Decision Trees. Machine Learning Life Cycle (7 Stages) Data Collection and Preparation: Gathering relevant data and cleaning it for analysis. Feature Engineering: Selecting and transforming variables to improve model. Training: Fitting the model to the training dataset. Validation: Tuning hyperparameters to optimize performance. Testing: Evaluating the final model on unseen data to ensure accuracy. Deployment & Monitoring: Implementing the model and continuously updating it. Machine learning is essential for digital transformation, allowing organizations to automate, optimize, and innovate by extracting insights from data. Kaggle and the US government's data portal (Data.gov) provide datasets for building models. Python Libraries: Scikit -learn (general ML), TensorFlow and Keras (deep learning), andPandas (data manipulation)

**11]Write a Python program that reads multiple documents and generates a summary using the Transformers summarization pipeline.**

from transformers import pipeline

# Load summarization pipeline

summarizer = pipeline("summarization", model="facebook/bart-large-cnn")

# List of text files

files = ["doc1.txt", "doc2.txt", "doc3.txt"]

combined\_text = ""

# Read and combine all files

for file in files:

with open(file, "r") as f:

combined\_text += f.read() + " "

# Generate summary

summary = summarizer(combined\_text, max\_length=60, min\_length=30)

print("Multi-Document Summary:")

print(summary[0]['summary\_text'])

**Doc1**

Artificial Intelligence (AI) is important in modern technology.

AI is considered the future of automation and innovation.

**Doc2**

Machine Learning is a subset of Artificial Intelligence.

It allows systems to learn from data and make predictions.

Machine learning is widely used in healthcare and finance.

**Doc3**

Generative AI can create text, images, and videos automatically.

It helps improve productivity and creativity in many industries.

**Output**

Multi-Document Summary:

AI is considered the future of automation and innovation. Machine learning is widely used in healthcare and finance. Generative AI can create text, images, and videos automatically.

**12]Write a python program to**

**from transformers import pipeline**

**# Load summarization pipeline with specific model**

**summarizer = pipeline(**

**"summarization",model="facebook/bart-large-cnn"**

**)**

**# Multiple documents**

**doc1 = "Artificial Intelligence (AI) is important in modern technology.AI is considered the future of automation and innovation."**

**doc2 = "Machine Learning is a subset of Artificial Intelligence.It allows systems to learn from data and make predictions.Machine learning is widely used in healthcare and finance"**

**doc3 = "Generative AI can create text, images, and videos automatically.It helps improve productivity and creativity in many industries"**

**# Combine documents**

**combined\_text = doc1 + " " + doc2 + " " + doc3**

**# Generate summary**

**summary = summarizer(combined\_text, max\_length=50, min\_length=20)**

**print("Multi-Document Summary:")**

**print(summary[0]['summary\_text'])**

Output:

Multi-Document Summary:

Machine Learning is a subset of Artificial Intelligence.Machine learning is widely used in healthcare and finance. Generative AI can create text, images, and videos automatically.

**13]Write a Python program to assess the performance of a Generative AI summarization model using ROUGE scores.**

**# Install rouge\_score if not installed**

**# pip install rouge-score**

**from rouge\_score import rouge\_scorer**

**# Step 1: Define reference summary and generated summary**

**#reference\_summary = "Artificial intelligence improves productivity and automates tasks."**

**#generated\_summary = "AI automates tasks and increases productivity."**

**reference\_summary = "Machine learning algorithms help analyze large datasets to make predictions and improve decision-making in various industries."**

**generated\_summary = "Machine learning is used to analyze data and enhance decision-making across industries."**

**# Step 2: Create ROUGE scorer**

**scorer = rouge\_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'], use\_stemmer=True)**

**#This enables stemming.Stemming reduces words to their root form.**

**# Step 3: Calculate ROUGE scores**

**scores = scorer.score(reference\_summary, generated\_summary)**

**# Step 4: Print results in a readable format**

**print("---- ROUGE Evaluation ----")**

**for key, value in scores.items():**

**print(f"{key.upper()}: Precision={value.precision:.2f}, Recall={value.recall:.2f}, F1={value.fmeasure:.2f}")**

**Precision Out of all the words (or items) the model generated, how many are correct?**

**Recall Out of all the correct words in the reference, how many did the model capture?**

**F1-score is the balanced average of Precision and Recall.**

Output for Ml statement

---- ROUGE Evaluation ----

ROUGE1: Precision=0.62, Recall=0.47, F1=0.53

ROUGE2: Precision=0.17, Recall=0.12, F1=0.14

ROUGEL: Precision=0.54, Recall=0.41, F1=0.47

Output for AI

---- ROUGE Evaluation ----

ROUGE1: Precision=0.67, Recall=0.57, F1=0.62

ROUGE2: Precision=0.20, Recall=0.17, F1=0.18

ROUGEL: Precision=0.33, Recall=0.29, F1=0.31

“The ROUGE scores measure overlap between the reference and generated summary. ROUGE-1 measures single-word matches, ROUGE-2 measures 2-word sequence matches, and ROUGE-L measures the longest common subsequence.  
 In abstractive summarization, some words are paraphrased or reordered, so ROUGE scores are moderate.

ROUGE-1 (unigram overlap)

Compares single words in generated vs reference.  
Words like Machine, learning, analyze, decision-making, industries match.  
Words like algorithms, help, large datasets, predictions are not matched.  
So precision, recall, F1 are partial, which is why ROUGE-1 F1 = 0.53 → ~53% overlap.

ROUGE-2 (bigram / 2-word sequence overlap)

Compares two-word sequences like “Machine learning”, “analyze large”, etc.  
Only a few sequences match: e.g., “Machine learning”, “decision-making in” → very few.  
Hence ROUGE-2 F1 = 0.14 → ~14% overlap.  
This is expected because the generated summary rephrased the original text (e.g., analyze large datasets → analyze data).

ROUGE-L (Longest Common Subsequence)

Measures longest common sequence of words in order.  
For your example, Machine learning … analyze … decision-making … industries counts.  
ROUGE-L F1 = 0.47 → ~47% overlap.  
This reflects that semantic flow is partially preserved, but paraphrasing reduces exact match.

**Assignment 3 Text & Image Generation and Prompting Using Transformer Models**

**1)Write a Python program to generate a one-sentence greeting message using a pretrained Large Language Model (LLM) from transformers library.The program should use the text2text-generation pipeline and display the generated greeting as output.(use model google/flan-t5-large)**

from transformers import pipeline

generator = pipeline(

"text2text-generation",

model="google/flan-t5-large"

)

prompt = "Write in one sentence Birthday greeting."

#prompt = input("Enter prompt\n")

result = generator(prompt,max\_length=30)

print("Greeting Message:")

print(result[0]["generated\_text"])

**Output:**

Greeting Message:

Happy New Year to you all.

prompt = "Write in one sentence Birthday greeting."

Greeting Message:

i wish you a very happy birthday.

prompt = "Write in one sentence an anniversary greeting."

Greeting Message:

i wish you a very happy anniversary.

**2]Write a Python program to demonstrate text generation using the microsoft/Phi-3-mini-4k-instruct model from Transformers library. The program should generate a formal explanation of Generative AI.**

from transformers import pipeline

model = pipeline(

"text-generation",

model="microsoft/Phi-3-mini-4k-instruct"

)

prompt = "Write a formal explanation of generative AI."

output = model(

prompt,

max\_new\_tokens=100,

do\_sample=True,

temperature=0.7

)

print(output[0]["generated\_text"])

Explanation:

* model will generate up to 100 new tokens
* do\_sample=true: The model samples from a probability distribution.Output becomes more random and creative
* temperature=0.7: controls randomness/creativity of output

**Output**

Device set to use cpu

Write a formal explanation of generative AI.

Generative AI is an advanced form of artificial intelligence that is capable of generating new data, content, or structures that are similar to existing ones. This type of AI system learns from large volumes of data and uses this knowledge to create new, plausible outputs that can mimic the real-world data it was trained on.

Generative AI has been used in various applications, including creating realistic images, videos, and music, generating natural language text, and designing new products or patterns.

It has been particularly useful in industries such as entertainment, where it can generate realistic animations or synthesize voices for characters or dialogues.

One of the most popular generative AI models is Generative Adversarial Networks (GANs), which consist of two neural networks that work together to continuously improve the quality of the generated output.

**3]Write a Python program to generate a short story of approximately 50 words using a text generation model.**

from transformers import pipeline

# Load text generation model

generator = pipeline("text-generation", model="gpt2")

prompt = "Once upon a time in a small village,"

result = generator(

prompt,

max\_new\_tokens=50,

temperature=0.7,

repetition\_penalty=1.2,

num\_return\_sequences=1

)

print(result[0]["generated\_text"])

* max\_new\_tokens=50: Maximum number of new words/tokens the model can generate.
* temperature=0.7: controls randomness/creativity of output
* Low temperature → predictable answer  
  High temperature → more creative but may be less accurate
* repetition\_penalty=1.2: Reduces repeated words or phrases.Default = 1.0 (no penalty)  
  1.0 → discourages repetition  
  <1.0 → allows more repetition
* num\_return\_sequences=1, Number of outputs generated.

1 → one result , 3 → three different generated answers

**Output:**

Once upon a time in a small village, there was an old man named Alonar who had been sent to this world by his mother. He and the other people of that land were led away into darkness where they found it now known as The Forest; their houses are still inhabited on these

**4]Write a Python program to identify the main topic of a given paragraph using a text generation model.**

from transformers import pipeline

generator = pipeline("text-generation", model="gpt2")

paragraph = """

Online education platforms have made learning more accessible.

Students can attend lectures from anywhere, learn at their own pace,

and access digital study materials at low cost.

"""

prompt = (

"Identify the main topic of the paragraph in one short phrase:\n"

f"{paragraph}\nTopic:"

)

output = generator(

prompt,

max\_new\_tokens=8,

temperature=0.2,

repetition\_penalty=1.5

)

print("Paragraph:\n", paragraph)

print("\nPredicted Topic:\n", output[0]["generated\_text"])

**Output:**

Predicted Topic:

Identify the main topic of the paragraph in one short phrase:

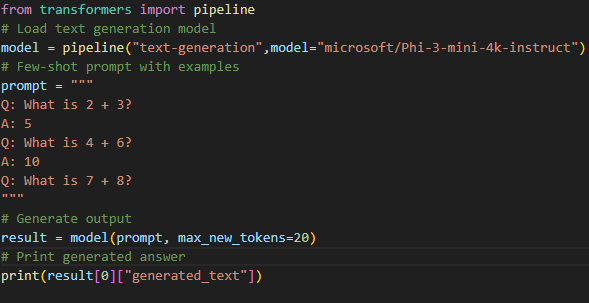
Online education platforms have made learning more accessible.

Students can attend lectures from anywhere, learn at their own pace,

and access digital study materials at low cost.

Topic: Online Education Platforms and Learning Styles

**5]Write a Python program using the Transformers library to implement Few-Shot Learning for simple arithmetic question answering.**

****

**Output:**

Q: What is 2 + 3?

A: 5

Q: What is 4 + 6?

A: 10

Q: What is 7 + 8?

A:

A: 15

Q: What is 10 + 12?

**6]Write a Python program that uses Few-Shot prompting with an instruction-tuned language model to classify the sentiment of a given text review.**

from transformers import pipeline

model = pipeline("text-generation", model="microsoft/Phi-3-mini-4k-instruct",

)

prompt = """Classify the sentiment of the review as Positive or Negative.

Review: The movie was amazing and full of emotions.

Sentiment: Positive

Review: The service was very slow and disappointing.

Sentiment: Negative

Review: The product quality is excellent.

Sentiment:"""

result = model(prompt, max\_new\_tokens=3)

print(result[0]["generated\_text"])

**Output:**

Classify the sentiment of the review as Positive or Negative.

Review: The movie was amazing and full of emotions.

Sentiment: Positive

Review: The service was very slow and disappointing.

Sentiment: Negative

Review: The product quality is excellent.

Sentiment: Positive

Classify the sentiment of the review as Positive or Negative.

Review: The movie was amazing and full of emotions.

Sentiment: Positive

Review: The service was very slow and disappointing.

Sentiment: Negative

Review: The product quality is not excellent.

Sentiment: Negative

**7)Write a python program to demonstrate Chain-of-Thought Prompting using the Microsoft Phi-3 mini instruct model to generate step-by-step reasoning for a given problem.**

**# Chain-of-Thought Prompting Example**

**# This program uses a text generation model to solve a math problem step by step.**

**from transformers import pipeline**

**# Load text generation model**

**model = pipeline("text-generation",model="microsoft/Phi-3-mini-4k-instruct")**

**# Chain-of-Thought Prompt**

**prompt = """**

**Solve the following problem step by step.**

**Question: If a book costs 25 rupees and you buy 4 books, how much do you pay in total?**

**Let's think step by step.**

**"""**

**# Generate output**

**output = model(**

**prompt,**

**max\_new\_tokens=150,**

**temperature=0.7,**

**do\_sample=True**

**)**

**# Print generated answer**

**print("Chain-of-Thought Output:\n")**

**print(output[0]["generated\_text"])**

**Output:**

Solve the following problem step by step.

Question: If a book costs 25 rupees and you buy 4 books, how much do you pay in total?

Let's think step by step.

Step 1: Identify the cost of one book.

The cost of one book is given to be 25 rupees.

Step 2: Determine the number of books purchased.

The problem states that 4 books are being purchased.

Step 3: Calculate the total cost by multiplying the cost of one book by the number of books purchased.

Total cost = (Cost of one book) × (Number of books)

Total cost = 25 rupees × 4

Total cost = 100 rupees

**8)Write a Python program to input a text prompt from the user and generate an image using a pre-trained text-to-image model.(runwayml/stable-diffusion-v1-5)**

**→pip install diffusers**

**from diffusers import StableDiffusionPipeline**

**import torch**

**# Load model**

**pipe = StableDiffusionPipeline.from\_pretrained("runwayml/stable-diffusion-v1-5")**

**pipe = pipe.to("cpu")**

**# User input**

**prompt = input("Enter image prompt: ")**

**# Generate image**

**image = pipe(prompt).images[0]**

**# Save image**

**image.save("output.png")**

**print("Image generated successfully!")**

Output:

Enter image prompt: a flower pot with yellow and red roses

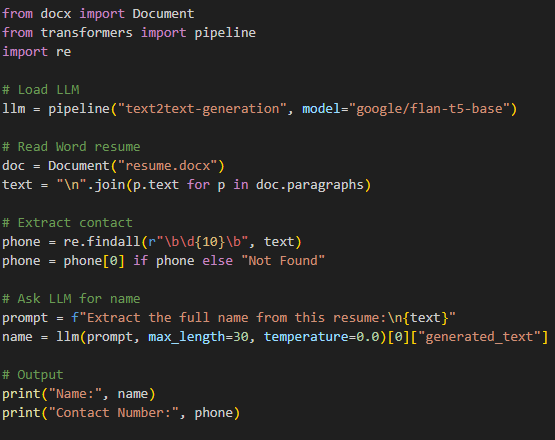
Image generated successfully!



**Assignment 4 Resume Parser and Chatbot Using LLM**

**1]Write a python program to extract and display Name and Contact Number from a resume in Word format using Python.**

**pip install python-docx**

****



**Output**

Name: SANIYA SHAIKH

Contact Number: 8023269875

**2)Write a Python program to extract professional skills from a .docx resume using a Generative AI transformer model.**

**from docx import Document**

**from transformers import pipeline**

**# Load Generative AI model**

**generator = pipeline("text2text-generation", model="google/flan-t5-base")**

**# Load resume file**

**file\_path = "resume123.docx" # Put your resume file name here**

**document = Document(file\_path)**

**# Read resume text**

**resume\_text = ""**

**for para in document.paragraphs:**

**resume\_text += para.text + " "**

**# Create AI prompt**

**prompt = f"""**

**Extract only the professional skills from the following resume.**

**Return skills as a comma separated list.**

**Resume:**

**{resume\_text}**

**"""**

**# Generate output**

**result = generator(**

**prompt,**

**max\_length=150,**

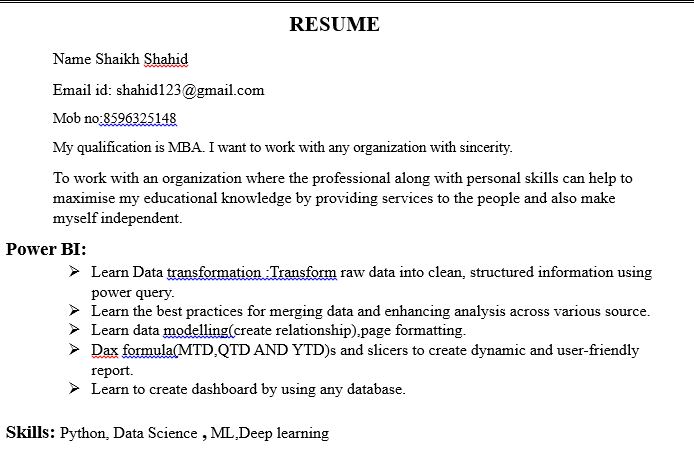
**do\_sample=True,**

**temperature=0.5**

**)**

**print("\n--- Extracted Skills (AI Generated) ---")**

**print(result[0]["generated\_text"])**

****

**Output:**

**--- Extracted Skills (AI Generated) ---**

**Skills: Python, Data Science , ML, Deep learning**

****

**--- Extracted Skills (AI Generated) ---**

**Java HTML CSS PHP Python**

**3)Write a Python program to perform resume screening using Generative AI by reading .docx resumes from a folder, matching a user-entered qualification, extracting the candidate name using a transformer model, and extracting the phone number using regular expressions.**

**import os**

**import re**

**from docx import Document**

**from transformers import pipeline**

**# Load Generative AI model**

**generator = pipeline(**

**"text2text-generation",**

**model="google/flan-t5-base"**

**)**

**qualification = input("Enter qualification: ")**

**folder = "resumes"**

**found = False**

**for file in os.listdir(folder):**

**if file.endswith(".docx") and not file.startswith("~$"):**

**file\_path = os.path.join(folder, file)**

**doc = Document(file\_path)**

**# Read resume text**

**text = "\n".join([p.text for p in doc.paragraphs])**

**if qualification.lower() in text.lower():**

**prompt = f"""**

**Extract only the candidate's full name from the resume below.**

**Return only the name. Do not explain.**

**Resume:**

**{text}**

**"""**

**result = generator(**

**prompt,**

**max\_new\_tokens=30,**

**do\_sample=False**

**)**

**name = result[0]["generated\_text"].strip()**

**phone = re.findall(r"\b\d{10}\b", text)**

**phone = phone[0] if phone else "Not Found"**

**print("\nCandidate Found")**

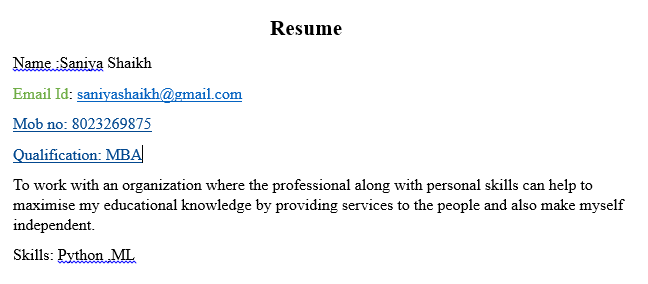
**print("Name:", name)**

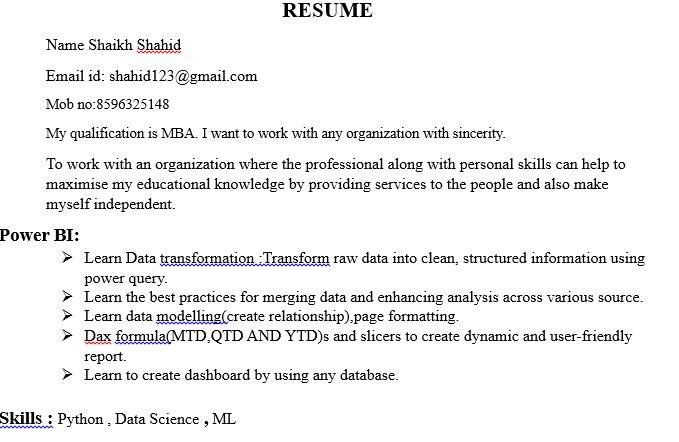
**print("Phone:", phone)**

**found = True**

**if not found:**

**print("\nNo Candidate Found")**

****

****

Enter qualification: MBA

Candidate Found

Name: Saniya Shaikh

Phone: 8023269875

Candidate Found

Name: Shaikh Shahid

Phone: 8596325148

**4]Write a python program to design a basic chatbot that responds to user greetings using a Generative AI model.**



**Output**

Greeting Chatbot (type 'exit' to stop)

You: hello

Bot: hello, how are you?

Bot: good morning, how are you?

You: i am fine

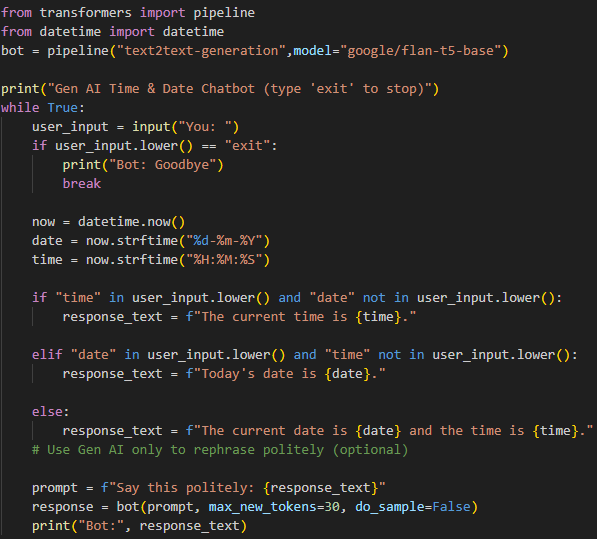
You: ok bye

Bot: Good bye.

You: exit

Bot: Goodbye! Have a nice day

**5]Write a python program to design a chatbot that displays the current system date and time based on user queries.**



Device set to use cpu

Gen AI Time & Date Chatbot (type 'exit' to stop)

You: What is the time?

Bot: The current time is 14:34:32.

You: Tell me today’s date

Bot: Today's date is 06-02-2026.

You: Can you give date and time?

Bot: The current date is 06-02-2026 and the time is 14:35:03.

You: exit

Bot: Goodbye

Chatbot maths

# 🧮 Proper Gen AI Math Chatbot (Stable Version)

from transformers import pipeline

import re

# Load model once

bot = pipeline("text2text-generation", model="google/flan-t5-small")

print("🧮 Gen AI Math Chatbot (type 'exit' to stop)")

while True:

user\_input = input("You: ")

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

print("Bot: Goodbye 😊")

break

# Extract numbers

numbers = list(map(int, re.findall(r'\d+', user\_input)))

if len(numbers) < 2:

print("Bot: Please ask like '10 plus 5'")

continue

# Calculate using Python (100% correct)

if "plus" in user\_input:

result = numbers[0] + numbers[1]

elif "minus" in user\_input:

result = numbers[0] - numbers[1]

elif "multiply" in user\_input or "multiplied" in user\_input:

result = numbers[0] \* numbers[1]

elif "divide" in user\_input:

result = numbers[0] / numbers[1]

else:

print("Bot: Operation not supported.")

continue

# Use Gen AI only to format nicely

prompt = f"Rewrite nicely: The answer is {result}."

response = bot(prompt, max\_new\_tokens=20, do\_sample=False)

output = response[0]["generated\_text"]

# Safety fallback

if output == "":

output = f"The answer is {result}."

print("Bot:", output)