In [ ]:
```
n = input("Enter a number: ")
if n == n[::-1]:
    print("This is a palindrome: ")
else:
    print("This is not palindrome: ")
```

In [ ]:
```
!pip install pymongo
```

In [ ]:
```
from pymongo import MongoClient

# Connect to MongoDB (Ensure MongoDB is running)
client = MongoClient("mongodb://localhost:27017")

# Create/select a database
db = client["Mani_database"]

# Create/select a collection (like a table)
collection = db["my_collection"]

print("Connected to MongoDB successfully!")
```

In [ ]:
```
# Insert multiple records
users = [
    {"name": "Alice", "age": 25, "city": "Los Angeles"},
    {"name": "John", "age": 28, "city": "Chicago"},
    {"name": "Emma", "age": 22, "city": "Houston"},
    {"name": "David", "age": 35, "city": "Miami"}
]

collection.insert_many(users)
print("Multiple documents inserted successfully!")
```

In [ ]:
```
client.list_database_names()
```

In [ ]:
```
db.client.learn
```

In [ ]:
```
db.list_collection_names()
```

In [ ]:
```
client.list_database_names()
```

In [ ]:
```
data = {
    "name": "Mani",
    "age": 23,
    "city": "Nagpur"
}
insert_result = collection.insert_one(data)
print("Inserted ID:", insert_result.inserted_id)

# Insert multiple documents (similar to inserting multiple rows)
data_list = [
    {"name": "Alice", "age": 25, "city": "Mumbai"},
    {"name": "Bob", "age": 28, "city": "Delhi"},
    {"name": "Charlie", "age": 35, "city": "Bangalore"}
]
insert_many_result = collection.insert_many(data_list)
print("Inserted IDs:", insert_many_result.inserted_ids)

# Confirm insertion by fetching all records
print("All records in 'tables' collection:")
for doc in collection.find():
    print(doc)
```

In [ ]:
```
db.tables.count_documents({})
```

In [ ]:
```
cursor = db.tables.find({})
```

In [ ]:
```
for doc in cursor:
    print(doc)
```

In [ ]:
```
data = {
    "name": "Mani",
    "age": 23,
    "city": "Nagpur"
}
insert_result = collection.insert_one(data)
print("Inserted ID:", insert_result.inserted_id)
```

In [ ]:
```
db.tables.count_documents({})
```

In [ ]:
```
data_list = [
    {"name": "siva", "age": 25, "city": "Mumbai"},
    {"name": "jagadessh", "age": 29, "city": "Delhi"},
    {"name": "Ganesh", "age": 22, "city": "Bangalore"}
]
insert_many_result = collection.insert_many(data_list)
print("Inserted IDs:", insert_many_result.inserted_ids)
```

In [ ]:
```
db.tables.count_documents({})
```

In [ ]:
```
cursor = db.tables.find({})
```

In [ ]:
```
for doc in cursor:
    print(doc)
```

In [ ]:
```
db.create_collection('Goods')
```

In [ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

texts = ["This is a sample document.", "This is another document.", "A complete
new_text = "This is a sample document."

# TF-IDF similarity check
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(texts + [new_text])

similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()
similarity_scores = similarity_matrix[-1][:-1]  # Compare new_text with existir

max_similarity = np.max(similarity_scores) if similarity_scores.size > 0 else 0
print(f"Similarity Score: {max_similarity}")
```

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from pymongo import MongoClient

# Step 1: Connect to MongoDB
client = MongoClient("mongodb://localhost:27017")
db = client["plagiarism_db"]
collection = db["documents"]

# Step 2: Get existing texts from MongoDB
texts = [doc["content"] for doc in collection.find()]  # Fetch all stored docum

# New input text
new_text = "This is a sample document."

# Step 3: Check Plagiarism using TF-IDF
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(texts + [new_text])

similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()
similarity_scores = similarity_matrix[-1][:-1]  # Compare new_text with stored

# Step 4: Get Plagiarism Percentage
max_similarity = np.max(similarity_scores) if similarity_scores.size > 0 else 0
plagiarism_percentage = round(max_similarity * 100, 2)  # Convert to percentage

print(f"Plagiarism Percentage: {plagiarism_percentage}%")

# Step 5: Store in MongoDB if plagiarism detected (above 30%)
if plagiarism_percentage > 30:
    collection.insert_one({"content": new_text, "plagiarism": plagiarism_percer
    print("Data stored in MongoDB.")
else:
    print("Plagiarism below threshold. Not storing.")
```

In [ ]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from pymongo import MongoClient

# Step 1: Connect to MongoDB
client = MongoClient("mongodb://localhost:27017")
db = client["plagiarism_db"]
collection = db["documents"]

# Step 2: Get user input text
new_text = input("Enter text to check for plagiarism: ")

# Step 3: Get existing texts from MongoDB
texts = [doc["content"] for doc in collection.find()]  # Fetch all stored docum

# Step 4: Check Plagiarism using TF-IDF
if texts:  # Only run if there is existing data in DB
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(texts + [new_text])

    similarity_matrix = (tfidf_matrix * tfidf_matrix.T).toarray()
    similarity_scores = similarity_matrix[-1][:-1]  # Compare new_text with sto

    # Step 5: Get Plagiarism Percentage
    max_similarity = np.max(similarity_scores) if similarity_scores.size > 0 el
    plagiarism_percentage = round(max_similarity * 100, 2)  # Convert to percer
else:
    plagiarism_percentage = 0  # No previous data, assume 0% plagiarism

print(f"Plagiarism Percentage: {plagiarism_percentage}%")

# Step 6: Store Data in MongoDB (Always store, even if plagiarism is 0%)
data_to_store = {"content": new_text, "plagiarism": plagiarism_percentage}
collection.insert_one(data_to_store)
print("Data stored in MongoDB:", data_to_store)
```

In [ ]:
```
pip install flask pymongo flask-cors
```

In [ ]:
```
from flask import Flask, request, jsonify
from flask_cors import CORS
from pymongo import MongoClient
import nest_asyncio
```

In [ ]:
```
nest_asyncio.apply()  # Allows Flask to run in Jupyter Notebook

app = Flask(__name__)
CORS(app)
```

In [ ]:
```
client = MongoClient("mongodb://localhost:27017")  # Connects to MongoDB
db = client["plagiarism_checker"]  # Creates or accesses the database
collection = db["docu"]  # Creates or accesses the collection (table)
```

In [ ]:
```python
@app.route('/check_plagiarism', methods=['POST'])
def check_plagiarism():
    data = request.json  # Extract JSON data sent from frontend
    text = data.get("text")  # Get the "text" field

    found = collection.find_one({"text": text})  # Check if text exists in Mong

    if found:
        return jsonify({"message": "Plagiarism detected!", "status": "plagiariz
    else:
        collection.insert_one({"text": text})  # Save the text in MongoDB
        return jsonify({"message": "No plagiarism found. Text saved!", "status"
```

In [ ]:
```python
if __name__ == "__main__":
    app.run(port=5000)
```

In [ ]:

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from pymongo import MongoClient
import nest_asyncio
from difflib import SequenceMatcher  # To calculate similarity

nest_asyncio.apply()

app = Flask(__name__)
CORS(app)

# Connect to MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["plagiarism_Editor"]
collection = db["documents_121"]

def calculate_similarity(text1, text2):
    """Calculate similarity percentage between two texts."""
    return SequenceMatcher(None, text1, text2).ratio() * 100  # Convert to perc

@app.route('/check_plagiarism', methods=['POST'])
def check_plagiarism():
    data = request.json
    text = data.get("text")

    # Fetch all stored texts
    all_texts = collection.find()
    max_similarity = 0  # Store the highest similarity score

    for record in all_texts:
        stored_text = record["text"]
        similarity = calculate_similarity(text, stored_text)
        max_similarity = max(max_similarity, similarity)  # Keep track of max s

    if max_similarity > 0:  # If some similarity is found
        return jsonify({"message": f"Plagiarism detected! Similarity: {max_simi
                        "status": "plagiarized", "percentage": max_similarity})
    else:
        collection.insert_one({"text": text})  # Store new unique text
        return jsonify({"message": "No plagiarism found. Text saved!",
                        "status": "unique", "percentage": 0})

if __name__ == "__main__":
    app.run(port=5000)
```

In [ ]:

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from pymongo import MongoClient
import nest_asyncio
from difflib import SequenceMatcher  # For similarity percentage

nest_asyncio.apply()  # Allows Flask to run in Jupyter Notebook

app = Flask(__name__)
CORS(app)

# Connect to MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["plagiarism_Guru"]
collection = db["documents_117"]

def get_similarity(text1, text2):
    return SequenceMatcher(None, text1, text2).ratio() * 100  # Convert to perc

@app.route('/check_plagiarism', methods=['POST'])
def check_plagiarism():
    data = request.json
    text = data.get("text")

    # Get all stored documents
    all_texts = collection.find({})

    highest_similarity = 0
    for doc in all_texts:
        similarity = get_similarity(text, doc["text"])
        if similarity > highest_similarity:
            highest_similarity = similarity  # Store the highest similarity fou

    if highest_similarity > 0:  # If there's some match, return similarity perc
        return jsonify({"message": "Plagiarism detected!", "status": "plagiariz
    else:
        collection.insert_one({"text": text})  # Save text if unique
        return jsonify({"message": "No plagiarism found. Text saved!", "status'

if __name__ == "__main__":
    app.run(port=5000)
```

In [ ]:
```python
from flask import Flask, request, jsonify
from flask_cors import CORS

app = Flask(__name__)
CORS(app)  # Allow cross-origin requests

@app.route('/check_plagiarism', methods=['POST'])
def check_plagiarism():
    data = request.json
    text = data.get("text", "")

    if not text.strip():
        return jsonify({"message": "No text provided", "percentage": 0.00})

    # Dummy plagiarism check logic (Replace this with actual plagiarism detecti
    import random
    plagiarism_percentage = round(random.uniform(0, 100), 2)  # Generates rando
    message = "Plagiarism detected!" if plagiarism_percentage > 0 else "No plag

    return jsonify({"message": message, "percentage": plagiarism_percentage})

if __name__ == '__main__':
    app.run(debug=True)
```

In [ ]: