```
!pip install langchain llama-index pinecone-client gradio
!pip install -U langchain-community
```

Show hidden output

```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URL of Analytics Vidhya Free Courses
url = "https://www.udemy.com/"

# Fetch the webpage
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")

# Parse course data
courses = []
for course in soup.find_all("div", class_="course-card"):  # Adjust the selector based on actual HTML
    title = course.find("h2").text.strip()  # Adjust selector
    description = course.find("p", class_="course-description").text.strip()  # Adjust selector
    link = course.find("a")["href"]
    courses.append({"title": title, "description": description, "link": link})

# Save data to a CSV
df = pd.DataFrame(courses)
df.to_csv("courses.csv", index=False)
print("Courses data saved to courses.csv")
```

```
Courses data saved to courses.csv
```

```
pip install langchain llama-index pinecone-client gradio
```

Show hidden output

```
!pip install -U langchain-community
```

Show hidden output

```
import os
from pinecone import Pinecone, ServerlessSpec

# Initialize Pinecone instance
pc = Pinecone(api_key = "pcsk_58zJHT_3nGMfD5V4eqVYYViJjwx77TKpiJbooD9LmBuvvnxSksSXiM1QWcGDSZHscDLUaN")

# Create Pinecone index if not already created
index_name = "course-search"
if index_name not in pc.list_indexes().names():
    pc.create_index(
        name=index_name,
        dimension=1536,  # Adjust the dimension based on your embedding model
        metric='cosine',  # Using cosine similarity for vector search
        spec=ServerlessSpec(
            cloud='aws',  # Adjust cloud provider if necessary
            region='us-east-1'  # Specify region
        )
    )

# Connect to the index
index = pc.Index(index_name)

import pandas as pd

try:
    df = pd.read_csv("courses.csv")
    print(df.head())  # Check the first few rows to confirm data
except pd.errors.EmptyDataError:
    print("The file is empty or cannot be read.")
except Exception as e:
    print(f"An error occurred: {e}")
```

```
The file is empty or cannot be read.
```

```
import gradio as gr

def search_courses(query):
    query_vector = embeddings.embed_text(query)
    results = index.query(query_vector, top_k=5, include_metadata=True)
```

```
    return [
        f"**{match['metadata']['title']}**\n{match['metadata']['link']}" for match in results["matches"]
    ]

# Gradio interface
interface = gr.Interface(
    fn=search_courses,
    inputs="text",
    outputs="text",
    title="Smart Course Search",
    description="Enter a keyword or phrase to search Analytics Vidhya's free courses."
)

interface.launch()
```

```
Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `s

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://c1aff74aa816039c76.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the worki
```