VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

Assignment Report
on

Data Visualization

Submitted By

K KEERTHI – 1RN21CD020

Under the Guidance of

Ms Vinutha S

Asst. Professor

Department of CSE (Data Science)

RN SHETTY TRUST®

RNS INSTITUTE OF TECHNOLOGY

Autonomous Institution Affiliated to VTU, Recognized by GOK, Approved by AICTE (NAAC 'A+ Grade' Accredited, NBA Accredited (UG - CSE, ECE, ISE, EIE and EEE)Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098
Ph:(080)28611880,28611881 URL: www.rnsit.ac.in

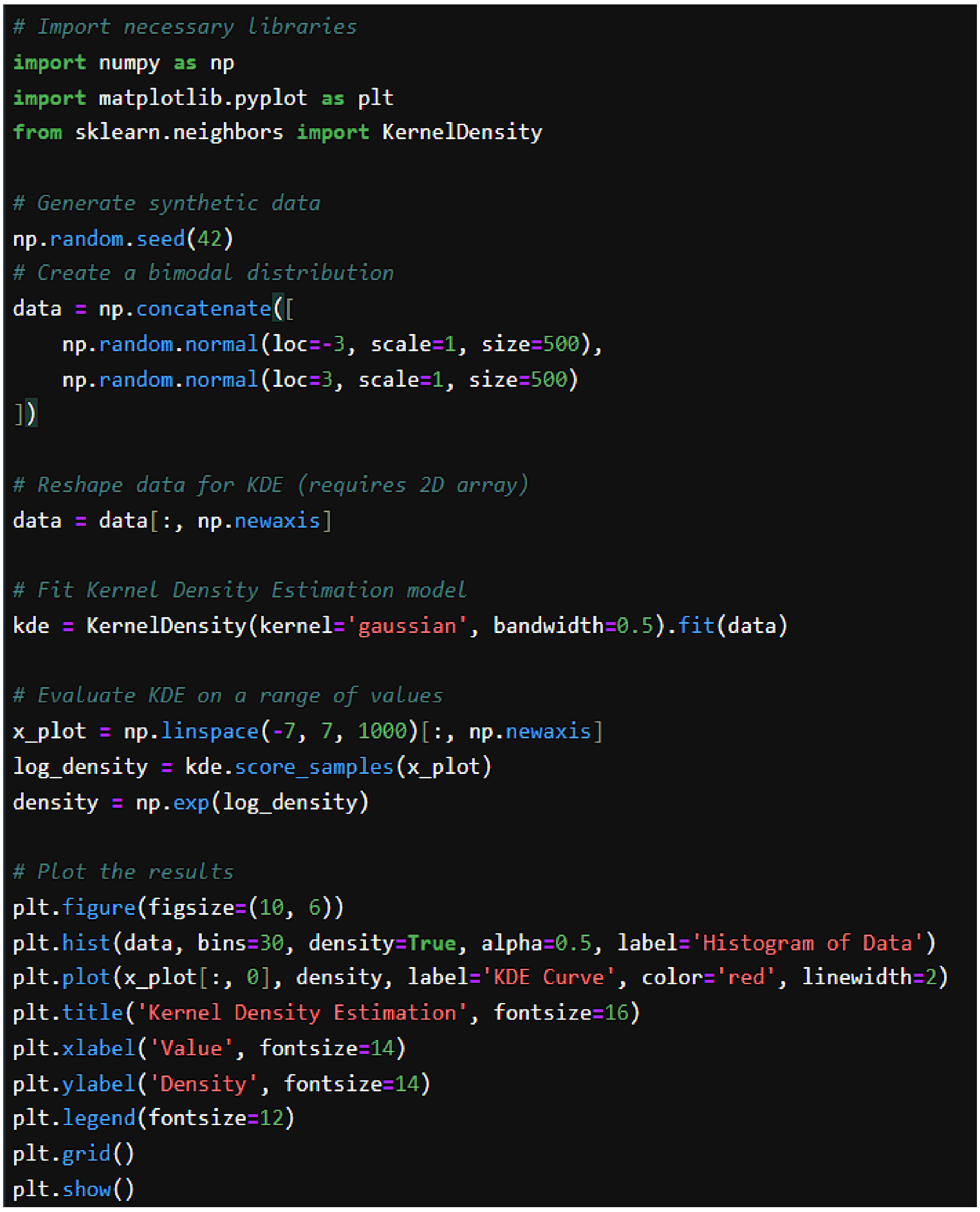
2024-2025

Table of Contents

Introduction

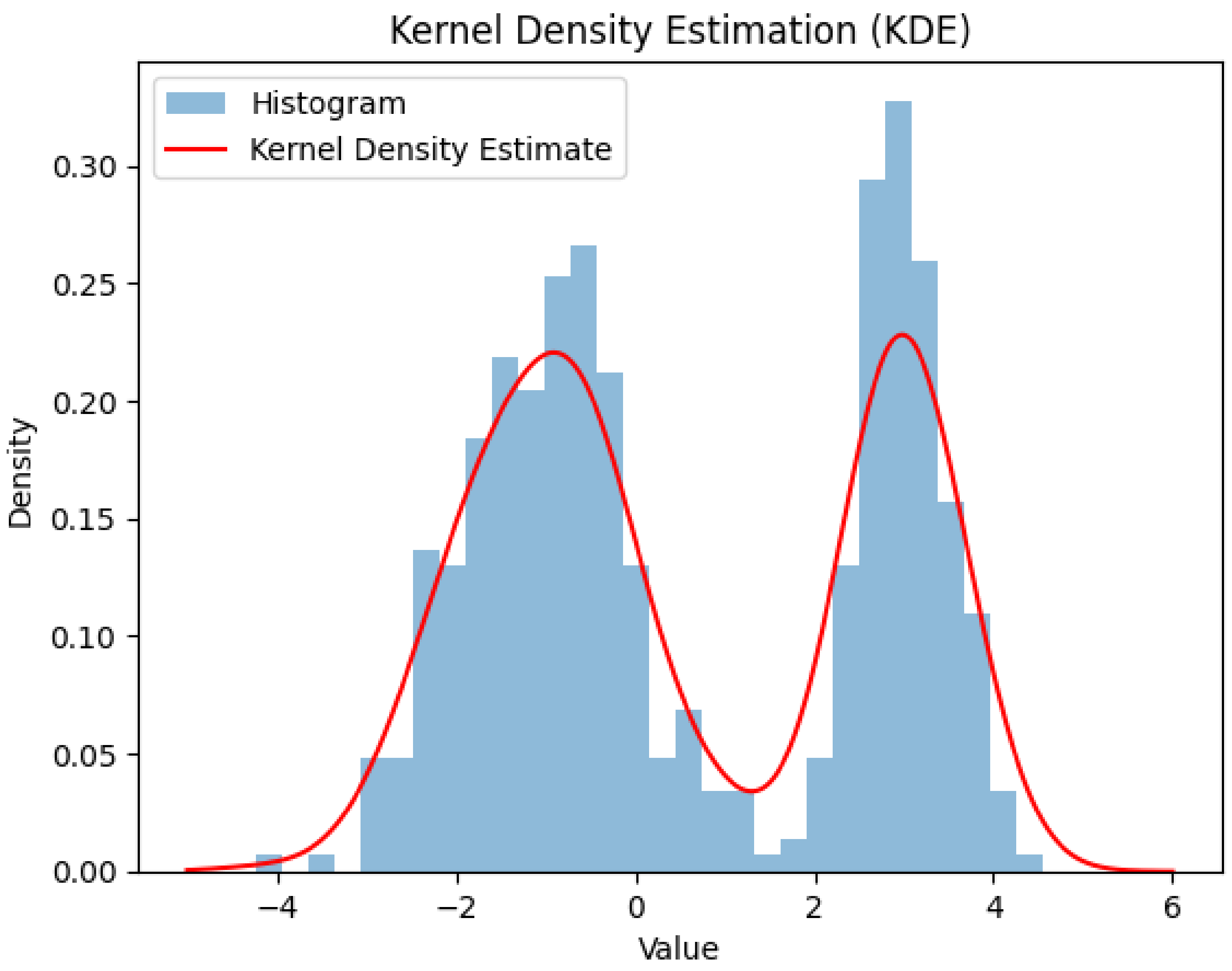
This report presents solutions to various data analysis and visualization tasks using Python libraries such as Numpy, Pandas, Matplotlib, and Seaborn. Each question addresses a specific aspect of data analysis and visualization.

Question 1: Develop a code to demonstrate Kernel Density Estimation

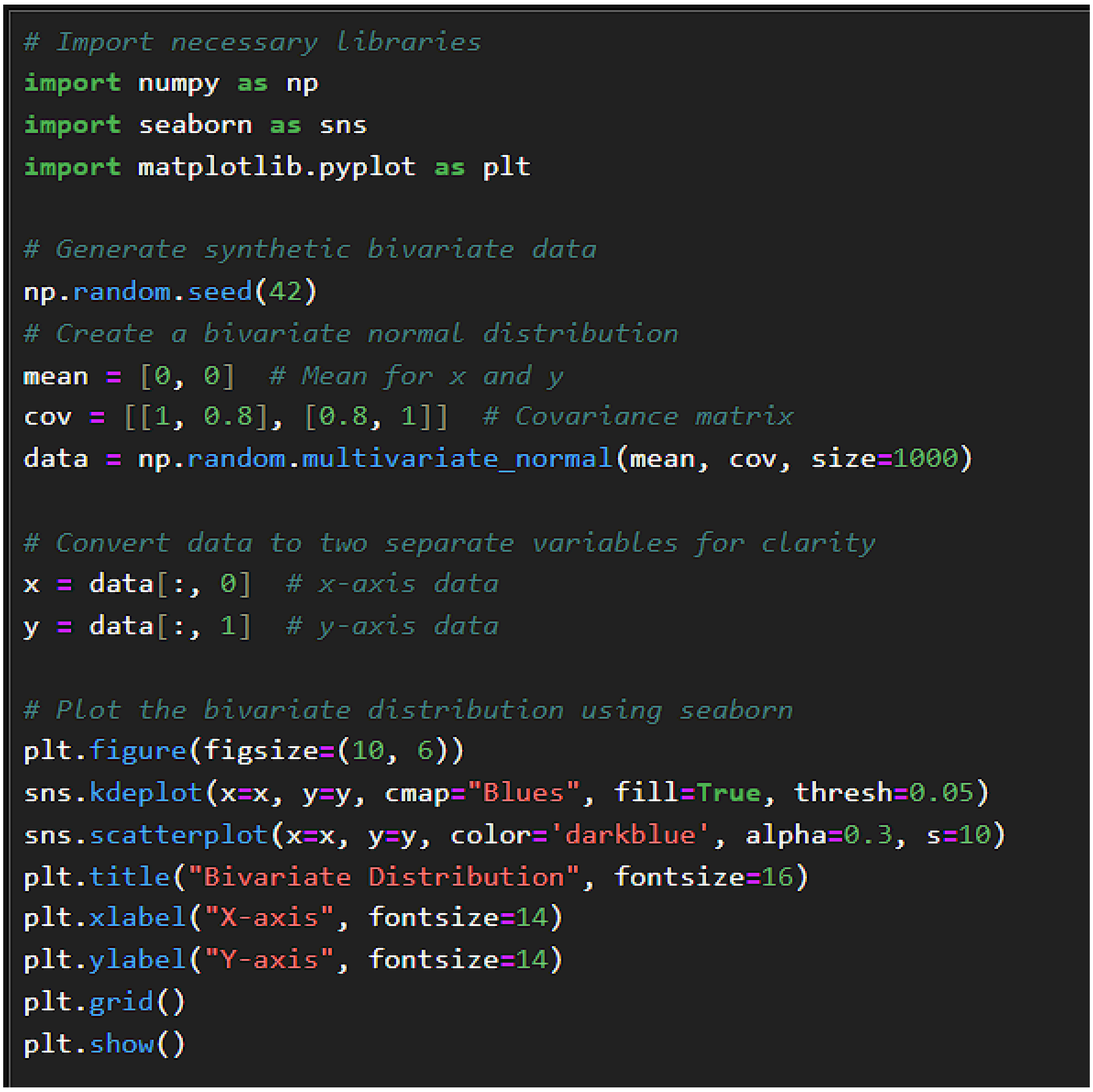


```python
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KernelDensity

# Generate synthetic data
np.random.seed(42)
# Create a bimodal distribution
data = np.concatenate([
    np.random.normal(loc=-3, scale=1, size=500),
    np.random.normal(loc=3, scale=1, size=500)
])

# Reshape data for KDE (requires 2D array)
data = data[:, np.newaxis]

# Fit Kernel Density Estimation model
kde = KernelDensity(kernel='gaussian', bandwidth=0.5).fit(data)

# Evaluate KDE on a range of values
x_plot = np.linspace(-7, 7, 1000)[:, np.newaxis]
log_density = kde.score_samples(x_plot)
density = np.exp(log_density)

# Plot the results
plt.figure(figsize=(10, 6))
plt.hist(data, bins=30, density=True, alpha=0.5, label='Histogram of Data')
plt.plot(x_plot[:, 0], density, label='KDE Curve', color='red', linewidth=2)
plt.title('Kernel Density Estimation', fontsize=16)
plt.xlabel('Value', fontsize=14)
plt.ylabel('Density', fontsize=14)
plt.legend(fontsize=12)
plt.grid()
plt.show()
```

Output:

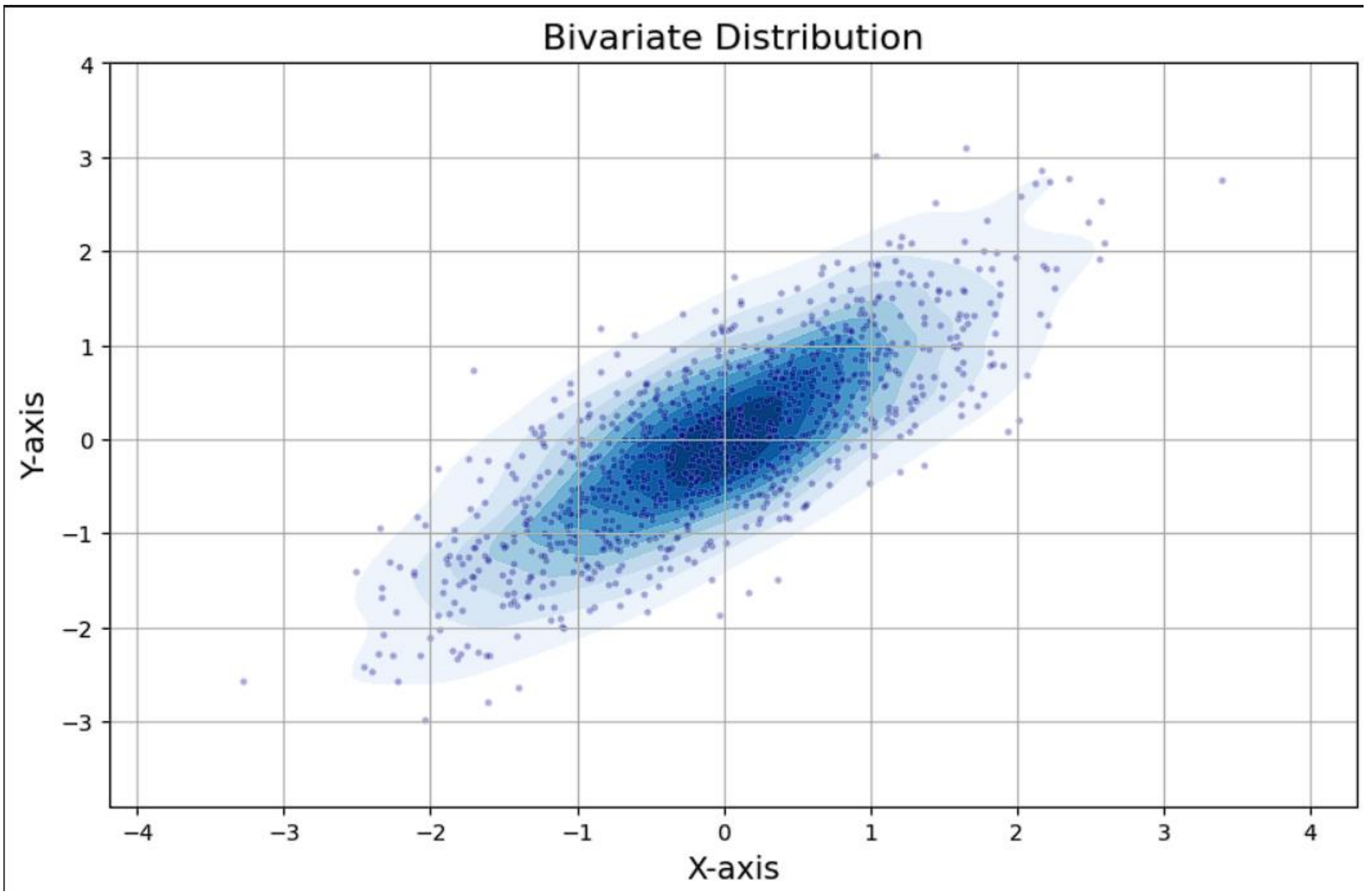

Kernel Density Estimation (KDE)
Histogram
Kernel Density Estimate
Density
Value
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−4
−2
0
2
4
6

Question 2: Develop a code to plot bivariate distribution considering asuitable data set available on the open source.

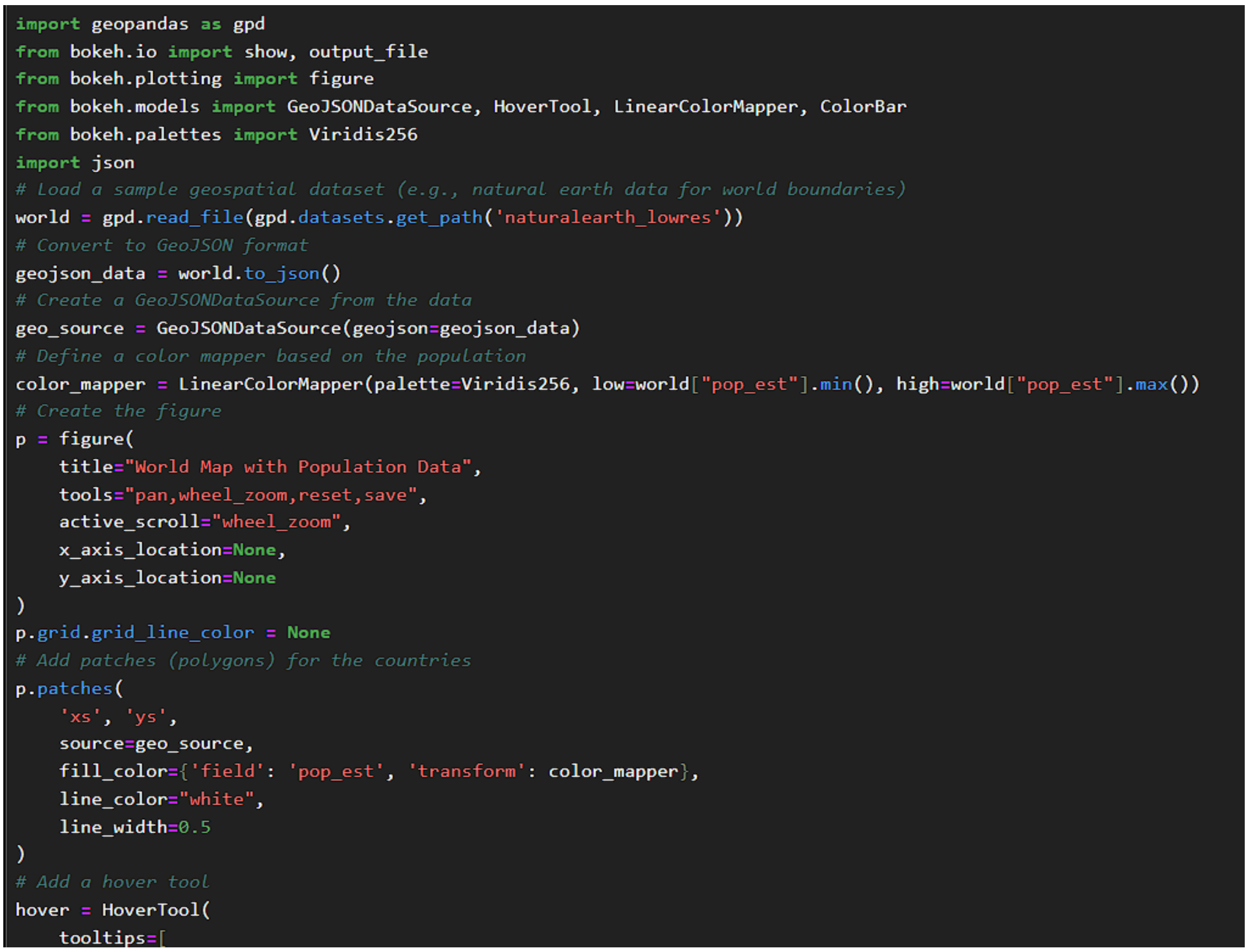


```python
# Import necessary libraries
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Generate synthetic bivariate data
np.random.seed(42)
# Create a bivariate normal distribution
mean = [0, 0]  # Mean for x and y
cov = [[1, 0.8], [0.8, 1]]  # Covariance matrix
data = np.random.multivariate_normal(mean, cov, size=1000)

# Convert data to two separate variables for clarity
x = data[:, 0]  # x-axis data
y = data[:, 1]  # y-axis data

# Plot the bivariate distribution using seaborn
plt.figure(figsize=(10, 6))
sns.kdeplot(x=x, y=y, cmap="Blues", fill=True, thresh=0.05)
sns.scatterplot(x=x, y=y, color='darkblue', alpha=0.3, s=10)
plt.title("Bivariate Distribution", fontsize=16)
plt.xlabel("X-axis", fontsize=14)
plt.ylabel("Y-axis", fontsize=14)
plt.grid()
plt.show()
```

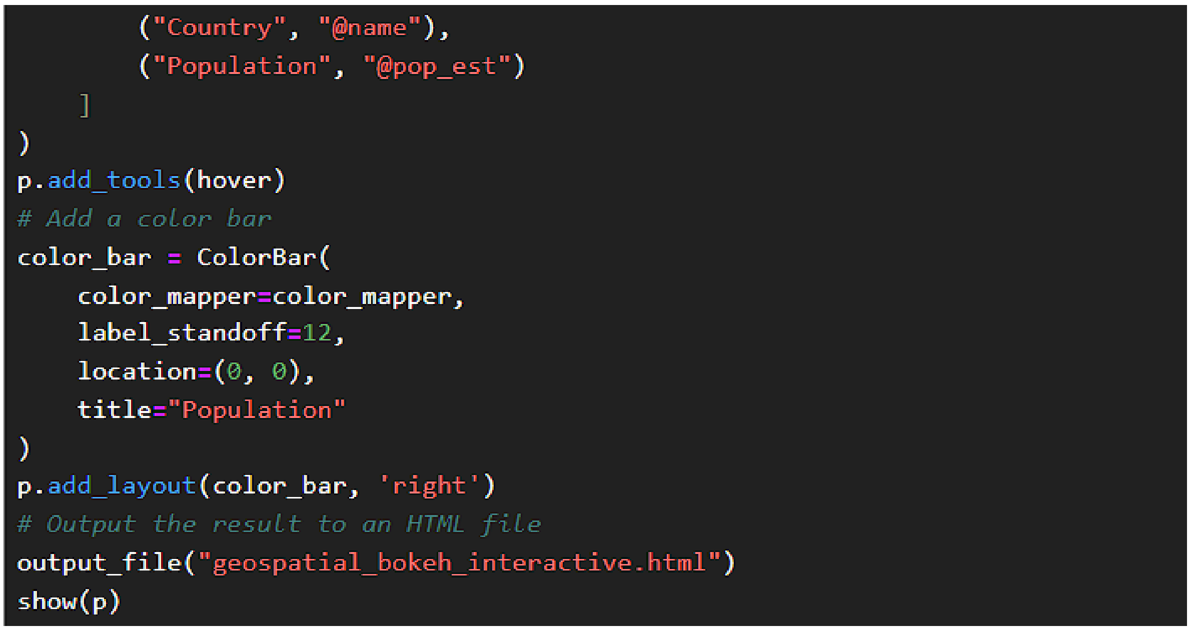
Output:



Question 3: Develop a code to showcase various Geospatial data also make use of Bokeh to make it more interactive.

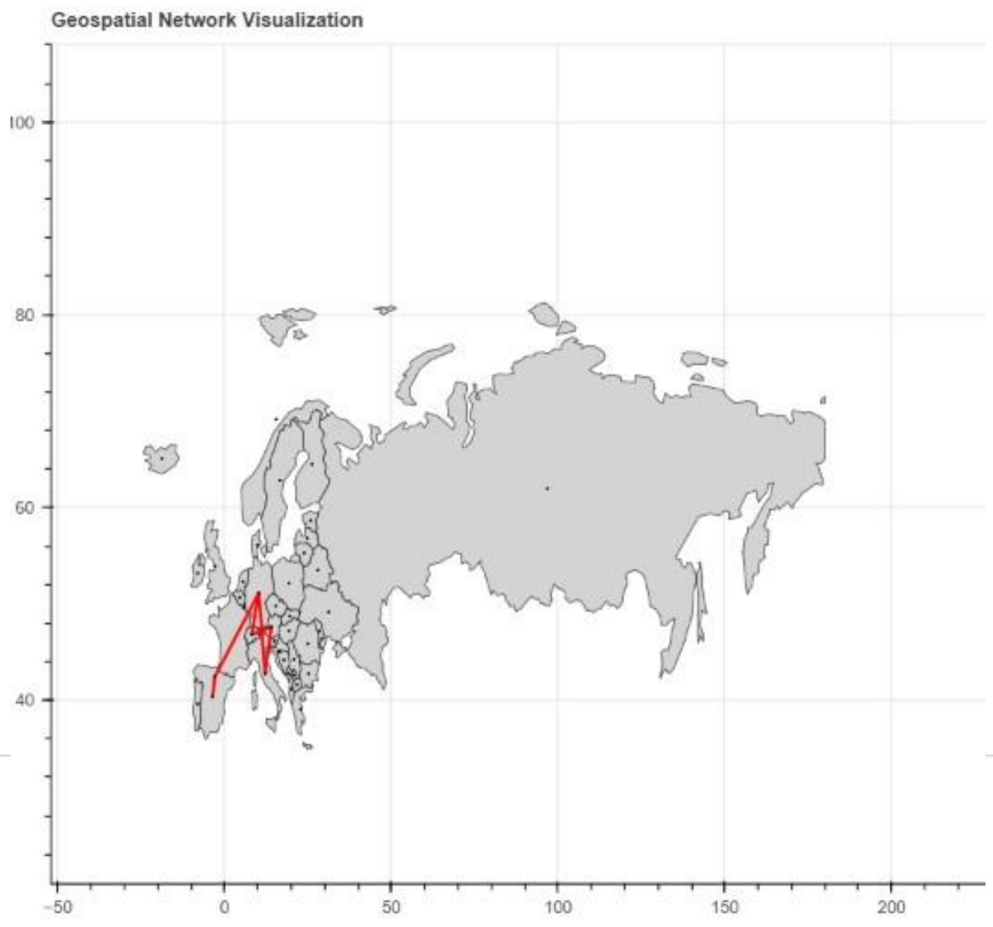


```python
import geopandas as gpd
from bokeh.io import show, output_file
from bokeh.plotting import figure
from bokeh.models import GeoJSONDataSource, HoverTool, LinearColorMapper, ColorBar
from bokeh.palettes import Viridis256
import json
# Load a sample geospatial dataset (e.g., natural earth data for world boundaries)
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
# Convert to GeoJSON format
geojson_data = world.to_json()
# Create a GeoJSONDataSource from the data
geo_source = GeoJSONDataSource(geojson=geojson_data)
# Define a color mapper based on the population
color_mapper = LinearColorMapper(palette=Viridis256, low=world["pop_est"].min(), high=world["pop_est"].max())
# Create the figure
p = figure(
    title="World Map with Population Data",
    tools="pan,wheel_zoom,reset,save",
    active_scroll="wheel_zoom",
    x_axis_location=None,
    y_axis_location=None
)
p.grid.grid_line_color = None
# Add patches (polygons) for the countries
p.patches(
    'xs', 'ys',
    source=geo_source,
    fill_color={'field': 'pop_est', 'transform': color_mapper},
    line_color="white",
    line_width=0.5
)
# Add a hover tool
hover = HoverTool(
    tooltips=[
```

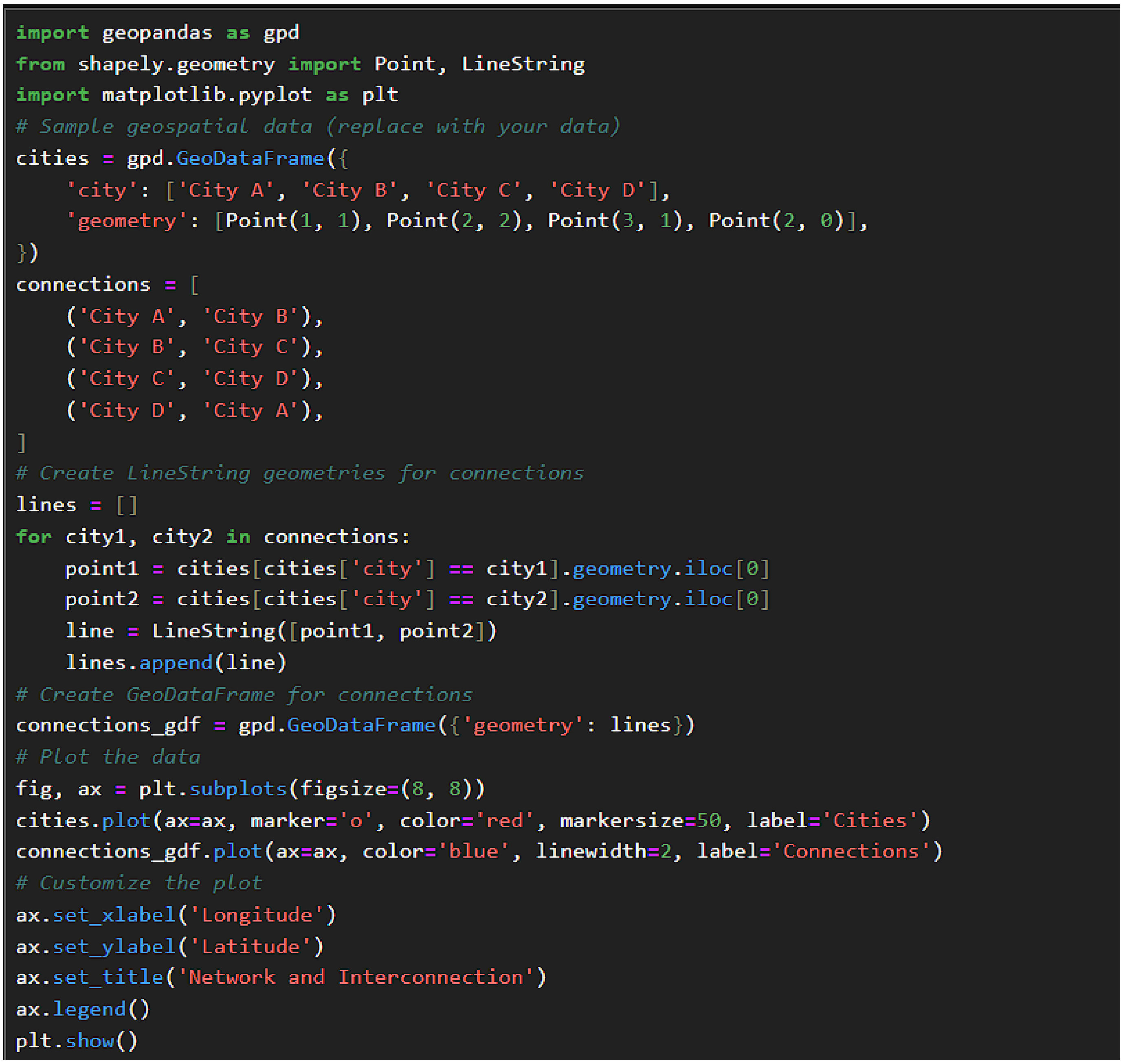


```python
        ("Country", "@name"),
        ("Population", "@pop_est")
    ]
)
p.add_tools(hover)
# Add a color bar
color_bar = ColorBar(
    color_mapper=color_mapper,
    label_standoff=12,
    location=(0, 0),
    title="Population"
)
p.add_layout(color_bar, 'right')
# Output the result to an HTML file
output_file("geospatial_bokeh_interactive.html")
show(p)
```

Output:

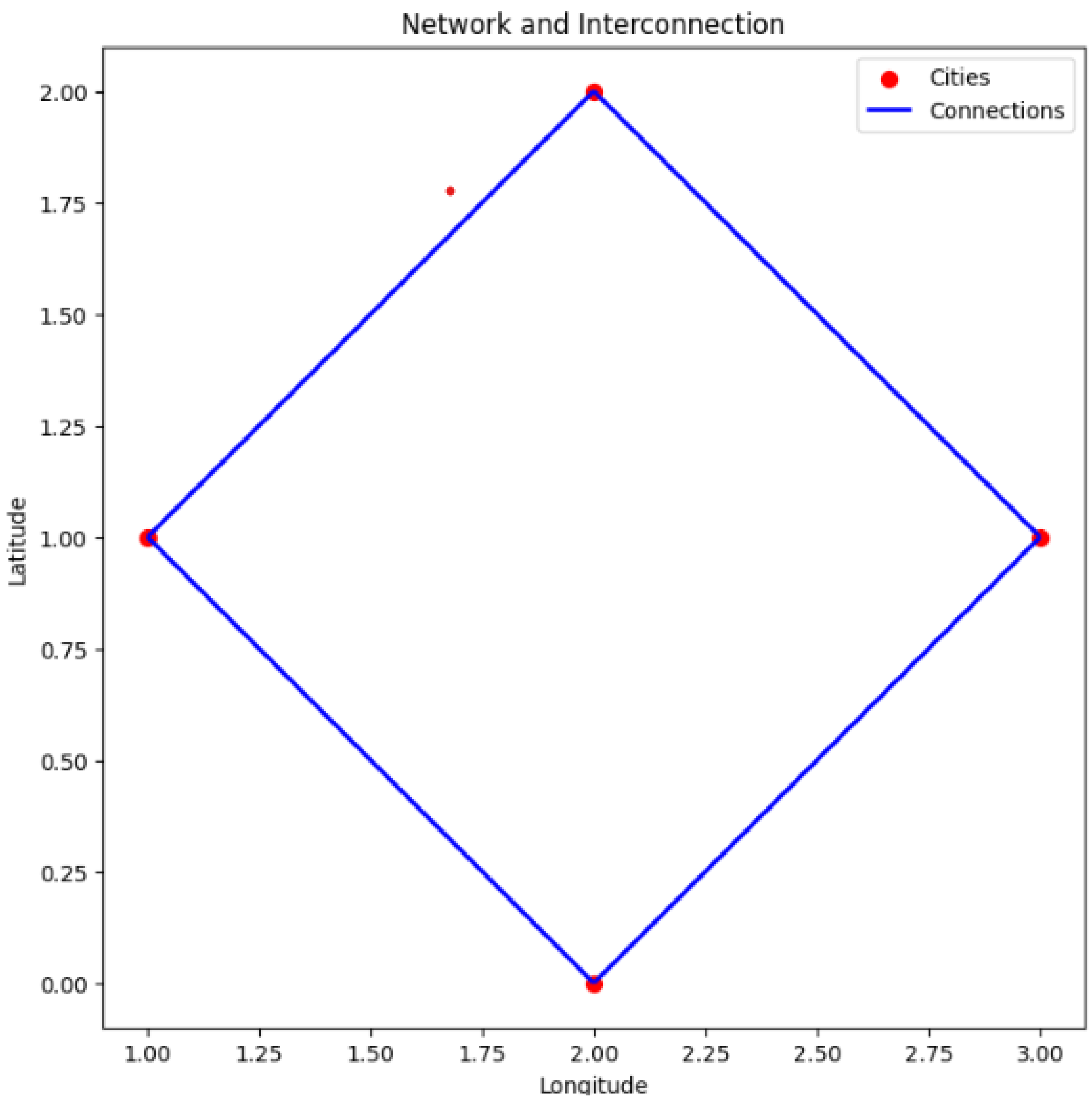

Geospatial Network Visualization
100
80
60
40
-50
0
50
100
150
200

Question 4: Develop a code to plot network and interconnection using geospatial data.

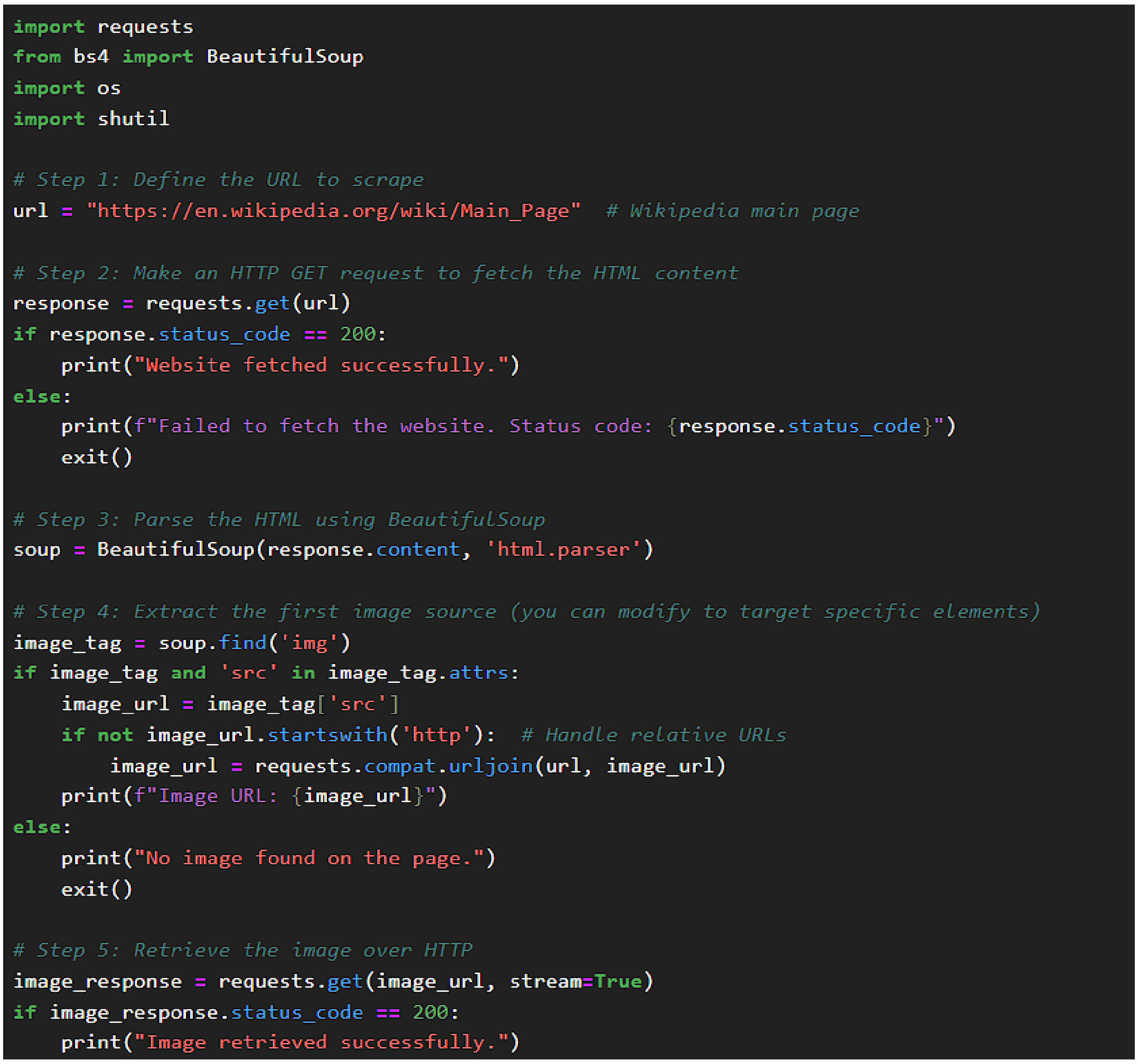


```
import geopandas as gpd
from shapely.geometry import Point, LineString
import matplotlib.pyplot as plt
# Sample geospatial data (replace with your data)
cities = gpd.GeoDataFrame({
    'city': ['City A', 'City B', 'City C', 'City D'],
    'geometry': [Point(1, 1), Point(2, 2), Point(3, 1), Point(2, 0)],
})
connections = [
    ('City A', 'City B'),
    ('City B', 'City C'),
    ('City C', 'City D'),
    ('City D', 'City A'),
]
# Create LineString geometries for connections
lines = []
for city1, city2 in connections:
    point1 = cities[cities['city'] == city1].geometry.iloc[0]
    point2 = cities[cities['city'] == city2].geometry.iloc[0]
    line = LineString([point1, point2])
    lines.append(line)
# Create GeoDataFrame for connections
connections_gdf = gpd.GeoDataFrame({'geometry': lines})
# Plot the data
fig, ax = plt.subplots(figsize=(8, 8))
cities.plot(ax=ax, marker='o', color='red', markersize=50, label='Cities')
connections_gdf.plot(ax=ax, color='blue', linewidth=2, label='Connections')
# Customize the plot
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_title('Network and Interconnection')
ax.legend()
plt.show()
```

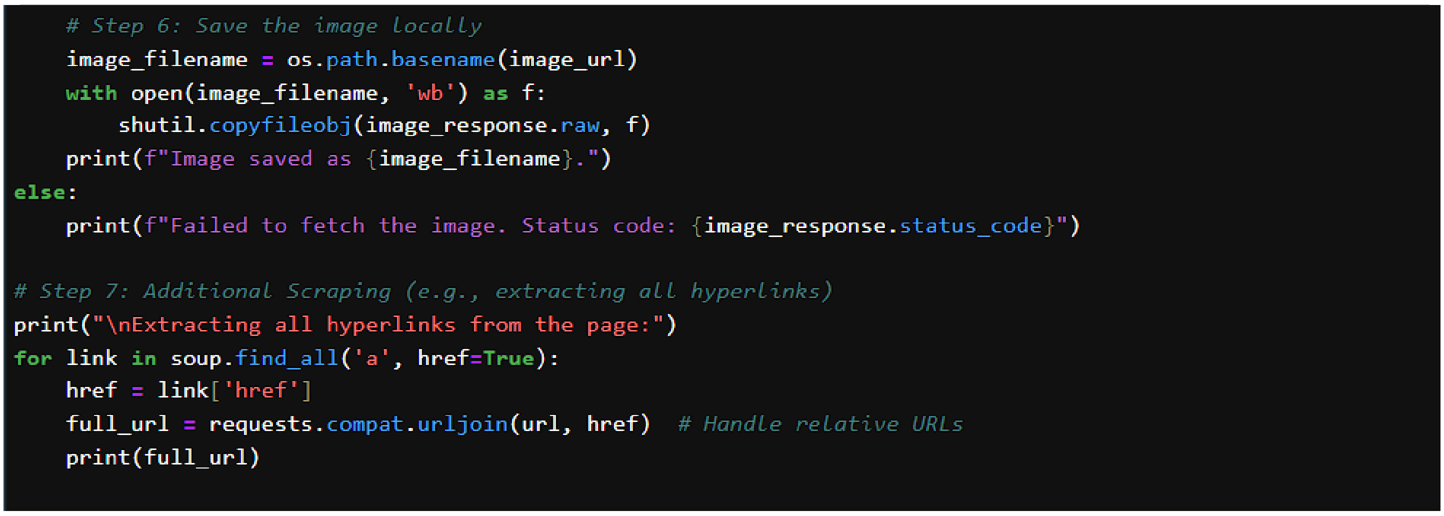
Output:


Network and Interconnection
Cities
Connections
Latitude
Longitude
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
1.00
1.25
1.50
1.75
2.00
2.25
2.50
2.75
3.00

Question 5: Develop a code showcase networked program including retrieving image over HTTP, parsing HTML and scraping the web.

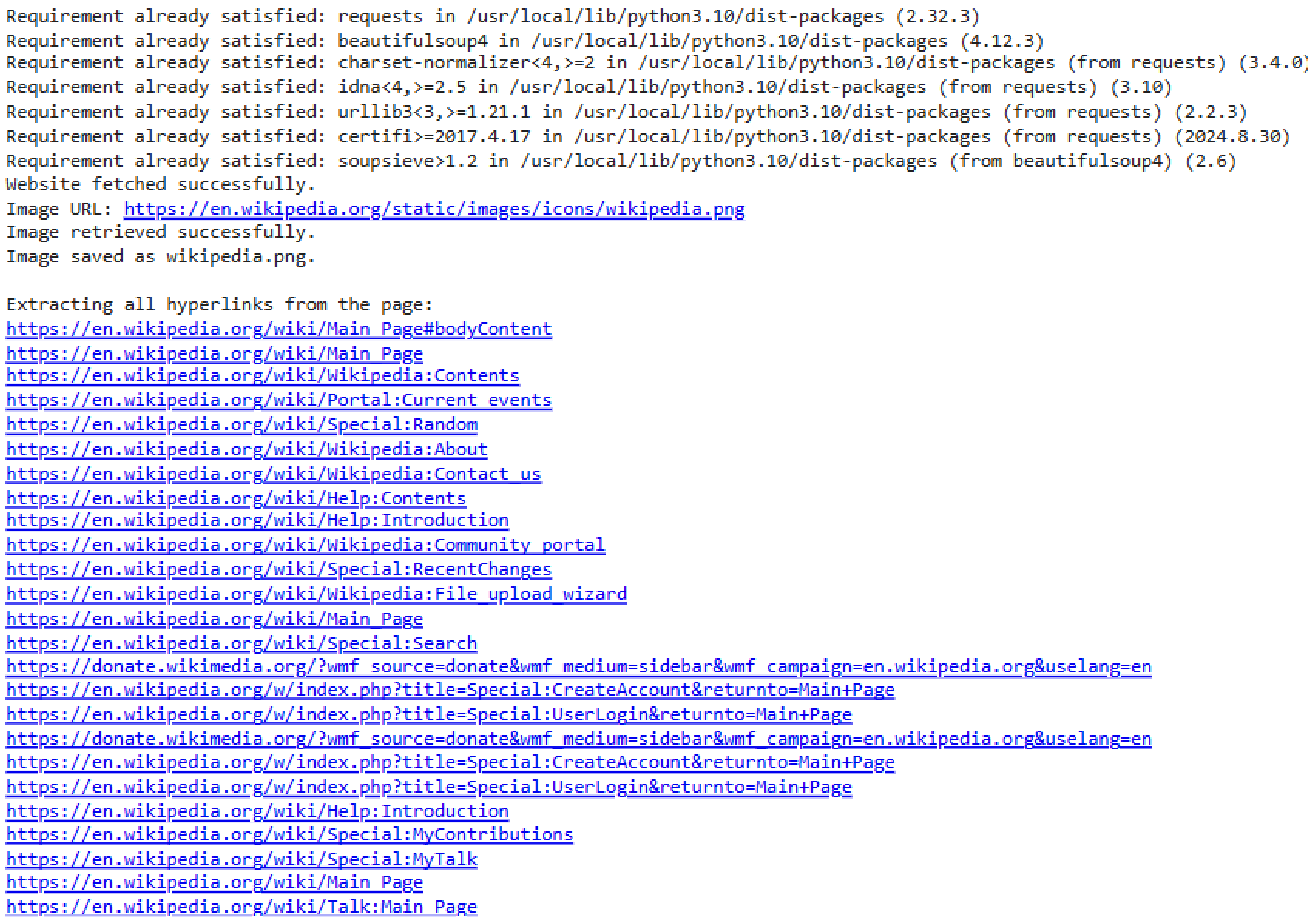


```python
import requests
from bs4 import BeautifulSoup
import os
import shutil

# Step 1: Define the URL to scrape
url = "https://en.wikipedia.org/wiki/Main_Page"  # Wikipedia main page

# Step 2: Make an HTTP GET request to fetch the HTML content
response = requests.get(url)
if response.status_code == 200:
    print("Website fetched successfully.")
else:
    print(f"Failed to fetch the website. Status code: {response.status_code}")
    exit()

# Step 3: Parse the HTML using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')

# Step 4: Extract the first image source (you can modify to target specific elements)
image_tag = soup.find('img')
if image_tag and 'src' in image_tag.attrs:
    image_url = image_tag['src']
    if not image_url.startswith('http'):  # Handle relative URLs
        image_url = requests.compat.urljoin(url, image_url)
    print(f"Image URL: {image_url}")
else:
    print("No image found on the page.")
    exit()

# Step 5: Retrieve the image over HTTP
image_response = requests.get(image_url, stream=True)
if image_response.status_code == 200:
    print("Image retrieved successfully.")
```



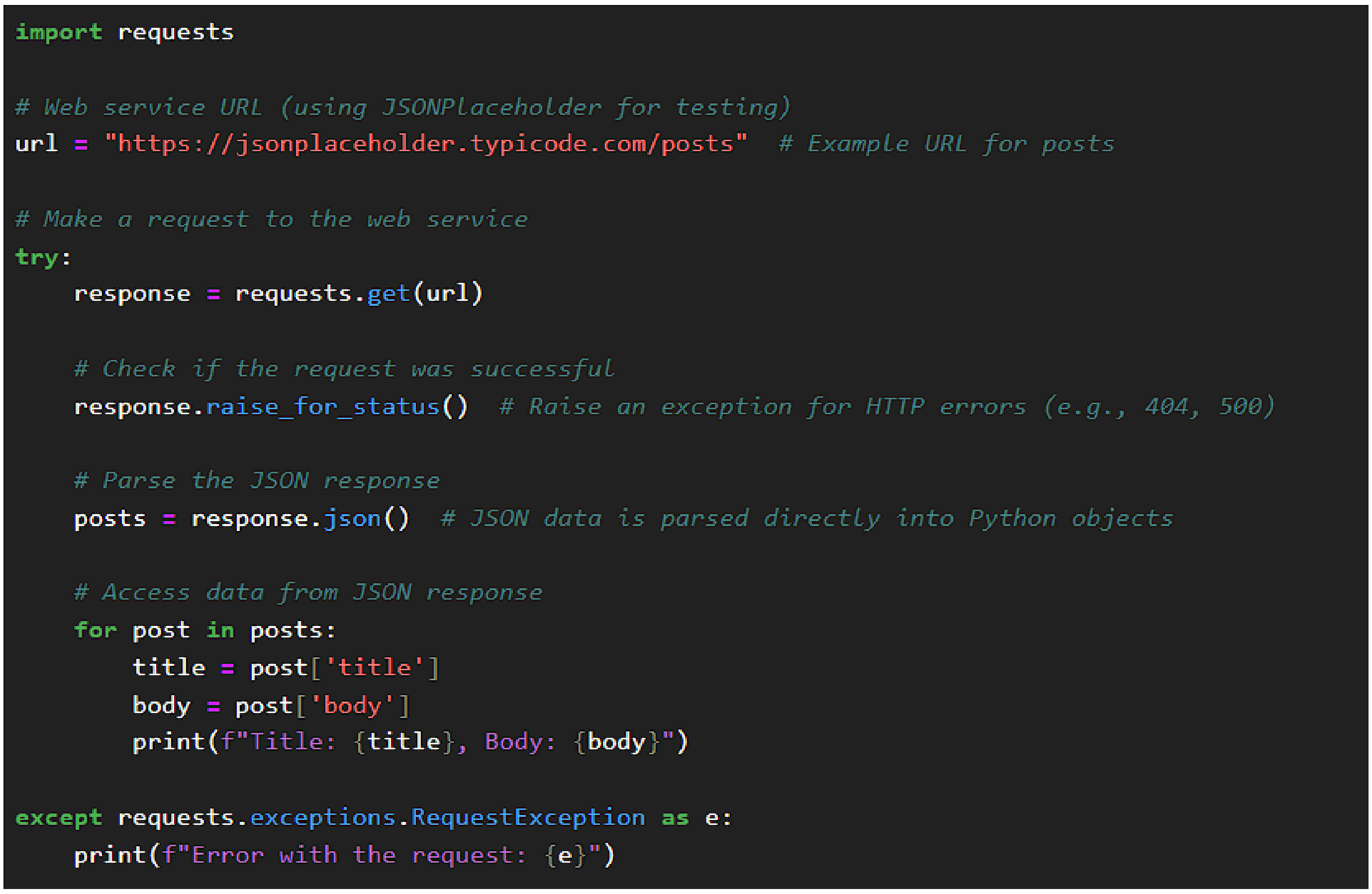
```python
    # Step 6: Save the image locally
    image_filename = os.path.basename(image_url)
    with open(image_filename, 'wb') as f:
        shutil.copyfileobj(image_response.raw, f)
    print(f"Image saved as {image_filename}.")
else:
    print(f"Failed to fetch the image. Status code: {image_response.status_code}")

# Step 7: Additional Scraping (e.g., extracting all hyperlinks)
print("\nExtracting all hyperlinks from the page:")
for link in soup.find_all('a', href=True):
    href = link['href']
    full_url = requests.compat.urljoin(url, href)  # Handle relative URLs
    print(full_url)
```

Output:



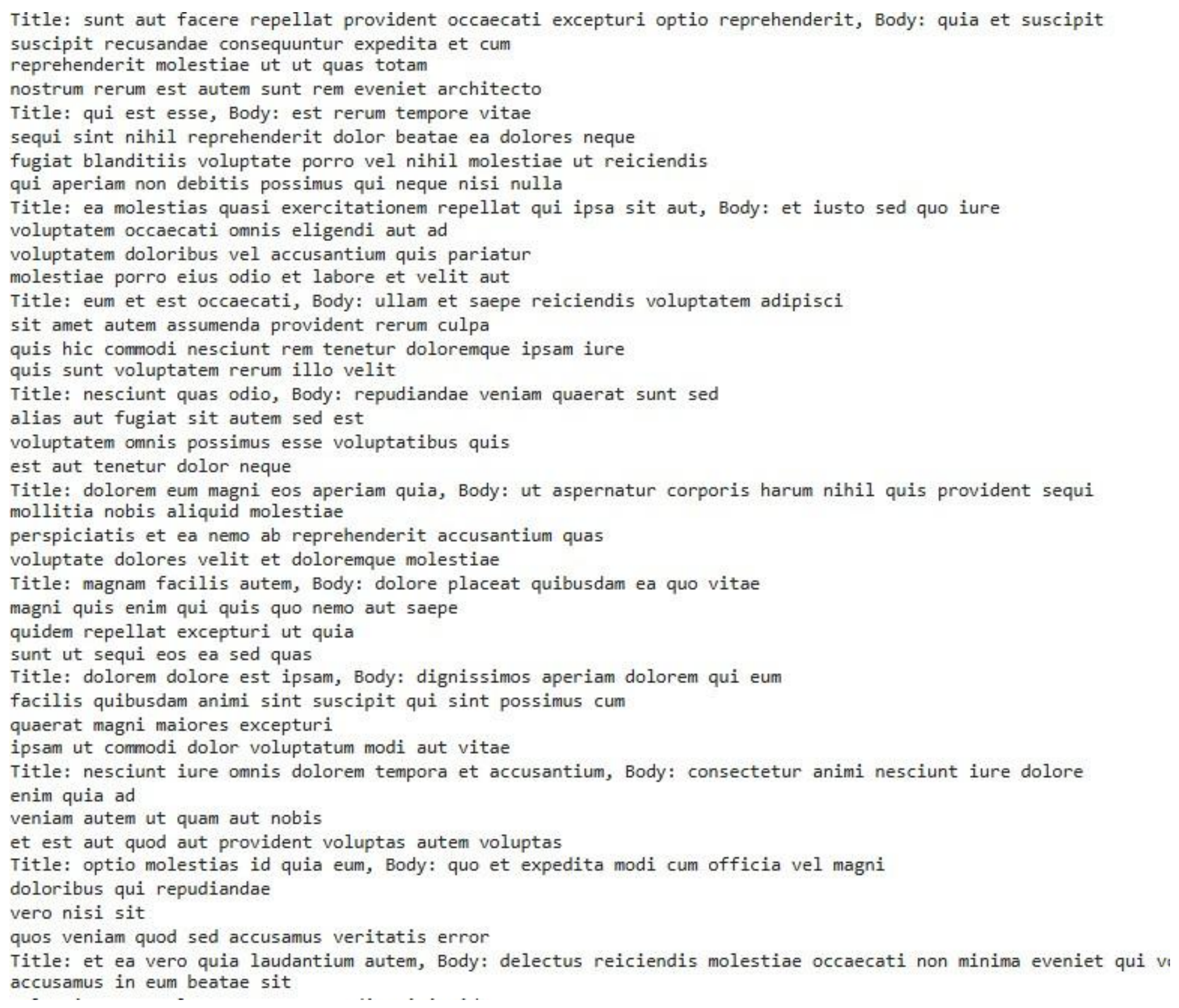
```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.32.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-packages (4.12.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (2024.8.30)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-packages (from beautifulsoup4) (2.6)
Website fetched successfully.
Image URL: https://en.wikipedia.org/static/images/icons/wikipedia.png
Image retrieved successfully.
Image saved as wikipedia.png.

Extracting all hyperlinks from the page:
https://en.wikipedia.org/wiki/Main_Page#bodyContent
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Wikipedia:Contents
https://en.wikipedia.org/wiki/Portal:Current_events
https://en.wikipedia.org/wiki/Special:Random
https://en.wikipedia.org/wiki/Wikipedia:About
https://en.wikipedia.org/wiki/Wikipedia:Contact_us
https://en.wikipedia.org/wiki/Help:Contents
https://en.wikipedia.org/wiki/Help:Introduction
https://en.wikipedia.org/wiki/Wikipedia:Community_portal
https://en.wikipedia.org/wiki/Special:RecentChanges
https://en.wikipedia.org/wiki/Wikipedia:File_upload_wizard
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Special:Search
https://donate.wikimedia.org/?wmf_source=donate&wmf_medium=sidebar&wmf_campaign=en.wikipedia.org&uselang=en
https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Main+Page
https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Main+Page
https://donate.wikimedia.org/?wmf_source=donate&wmf_medium=sidebar&wmf_campaign=en.wikipedia.org&uselang=en
https://en.wikipedia.org/w/index.php?title=Special:CreateAccount&returnto=Main+Page
https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Main+Page
https://en.wikipedia.org/wiki/Help:Introduction
https://en.wikipedia.org/wiki/Special:MyContributions
https://en.wikipedia.org/wiki/Special:MyTalk
https://en.wikipedia.org/wiki/Main_Page
https://en.wikipedia.org/wiki/Talk:Main_Page
```

Question 6: Develop a code to showcase the web services includingextensible Markup Language.



```python
import requests

# Web service URL (using JSONPlaceholder for testing)
url = "https://jsonplaceholder.typicode.com/posts"  # Example URL for posts

# Make a request to the web service
try:
    response = requests.get(url)

    # Check if the request was successful
    response.raise_for_status()  # Raise an exception for HTTP errors (e.g., 404, 500)

    # Parse the JSON response
    posts = response.json()  # JSON data is parsed directly into Python objects

    # Access data from JSON response
    for post in posts:
        title = post['title']
        body = post['body']
        print(f"Title: {title}, Body: {body}")

except requests.exceptions.RequestException as e:
    print(f"Error with the request: {e}")
```

Output:



```
Title: sunt aut facere repellat provident occaecati excepturi optio reprehenderit, Body: quia et suscipit
suscipit recusandae consequuntur expedita et cum
reprehenderit molestiae ut ut quas totam
nostrum rerum est autem sunt rem eveniet architecto
Title: qui est esse, Body: est rerum tempore vitae
sequi sint nihil reprehenderit dolor beatae ea dolores neque
fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis
qui aperiam non debitis possimus qui neque nisi nulla
Title: ea molestias quasi exercitationem repellat qui ipsa sit aut, Body: et iusto sed quo iure
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
Title: eum et est occaecati, Body: ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure
quis sunt voluptatem rerum illo velit
Title: nesciunt quas odio, Body: repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
voluptatem omnis possimus esse voluptatibus quis
est aut tenetur dolor neque
Title: dolorem eum magni eos aperiam quia, Body: ut aspernatur corporis harum nihil quis provident sequi
mollitia nobis aliquid molestiae
perspiciatis et ea nemo ab reprehenderit accusantium quas
voluptate dolores velit et doloremque molestiae
Title: magnam facilis autem, Body: dolore placeat quibusdam ea quo vitae
magni quis enim qui quis quo nemo aut saepe
quidem repellat excepturi ut quia
sunt ut sequi eos ea sed quas
Title: dolorem dolore est ipsam, Body: dignissimos aperiam dolorem qui eum
facilis quibusdam animi sint suscipit qui sint possimus cum
quaerat magni maiores excepturi
ipsam ut commodi dolor voluptatum modi aut vitae
Title: nesciunt iure omnis dolorem tempora et accusantium, Body: consectetur animi nesciunt iure dolore
enim quia ad
veniam autem ut quam aut nobis
et est aut quod aut provident voluptas autem voluptas
Title: optio molestias id quia eum, Body: quo et expedita modi cum officia vel magni
doloribus qui repudiandae
vero nisi sit
quos veniam quod sed accusamus veritatis error
Title: et ea vero quia laudantium autem, Body: delectus reiciendis molestiae occaecati non minima eveniet qui v
accusamus in eum beatae sit
```

GIT- HUB :