```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  7 00:50:49 2023

@author: Nisarg
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt



def read_and_prepare_world_co2_data():
    """ This function reads csv and processes the data and return new \
        filtered dataframe """

    # read csv using pandas
    data = pd.read_csv("API_EN.ATM.CO2E.KT_DS2_en_csv_v2_5994970.csv",
                       skiprows=[0, 1, 2, 3])

    years_column_list = np.arange(1990, 2021).astype(str)
    all_cols_list = ["Country Name"] + list(years_column_list)

    countries = ["China", "United States", "India", "Russian Federation",
                 "Germany", "Brazil"]

    # Filter data: select only specific countries and years
    df_selected = data.loc[data["Country Name"].isin(countries),
                           all_cols_list]
```

```
    # Transpose
    df_t = pd.DataFrame.transpose(df_selected)
    df_t.columns = df_t.iloc[0]

    # remove first row
    df_t = df_t[1:]
    df_t.index = df_t.index.astype(int)

    # scale data from kiloton to megaton
    for c in countries:
        df_t[c + " megaton"] = df_t[c]/1000

    return df_t



def read_and_prepare_ev_data():
    """ This function reads csv and processes the data and return new \
        filtered dataframe """

    # read csv using pandas
    data = pd.read_csv("IEA-EV-dataEV salesHistoricalCars.csv")
    df_selected = data.loc[(data["region"] == "World")
                           & (data["category"] == "Historical")
                           & (data["parameter"] == "EV sales")
                           & (data["mode"] == "Cars")
                           & (data["powertrain"].isin(["PHEV", "BEV"]))
                           & (data["unit"] == "Vehicles"),
                           ["powertrain", "year", "value"]]
```

```
    # scale the number of ev sales into millions
    df_selected["value_million"] = df_selected["value"]/1000000

    return df_selected


def create_and_save_line_graph(data):
    """ This function takes data as an argument and creates a line chart for \
        co2 emission using matplotlib and save png image on disk """

    # start creating line chart
    plt.figure(figsize=(10, 6))
    plt.plot(data.index, data["China megaton"], label="China")
    plt.plot(data.index, data["United States megaton"], label="United States")
    plt.plot(data.index, data["India megaton"], label="India")
    plt.plot(data.index, data["Russian Federation megaton"],
             label="Russian Federation")
    plt.plot(data.index, data["Germany megaton"], label="Germany")
    plt.plot(data.index, data["Brazil megaton"], label="Brazil")

    # set label and legend
    plt.title("CO2 emission")
    plt.xlabel("Years")
    plt.ylabel("Megatons")
    plt.xticks(np.arange(min(data.index), max(data.index)+1, 5.0))
    plt.xlim(min(data.index), max(data.index))
    plt.legend()
```

```
    # save the graph in disk
    plt.savefig("fig1.png")



def create_and_save_pi_chart(data):
    """ This function takes data as an argument and creates two pi charts for \
        co2 emission using matplotlib and save png image on disk """

    countries = ["China", "United States", "India", "Russian Federation",
                 "Germany", "Brazil"]
    # start creating a line chart
    plt.figure(figsize=(10, 6))

    # use a subplot to show two graphs in a single graph
    # create pie chart one
    plt.subplot(1, 2, 1)
    plt.pie(data.loc[data.index == 1990, countries].values.flatten().tolist(),
            labels=countries, autopct='%1.0f%%', pctdistance=1.1,
            labeldistance=1.25, textprops={'fontsize': 10}, radius=0.9)
    plt.title("1990")

    # create pie chart two
    plt.subplot(1, 2, 2)
    plt.pie(data.loc[data.index == 2020, countries].values.flatten().tolist(),
            labels=countries, autopct='%1.0f%%', pctdistance=1.1,
            labeldistance=1.25, textprops={'fontsize': 10}, radius=0.9)
    plt.title("2020")
    plt.suptitle(' CO2 emission ', fontsize=15)
```

```
    # save the graph on disk
    plt.savefig("fig2.png")



def creat_and_save_bar_chart(data):
    """ This function takes data as an argument and creates a bar chart for \
        ev sale using matplotlib and save png image on disk """

    # get unique years for the x-axis
    years = data["year"].unique()
    # prepare y-axis data
    phev_data = data.loc[data["powertrain"] == "PHEV"]
    bev_data = data.loc[data["powertrain"] == "BEV"]

    # start creating a line chart
    plt.figure(figsize=(10, 6))
    plt.bar(years, phev_data["value_million"],
            label="PHEV(plug-in hybrid electric vehicles)")
    plt.bar(years, bev_data["value_million"],
            bottom=phev_data["value_million"],
            label="BEV(battery electric vehicles)")

    # set label and legend
    plt.title("EV sales, World")
    plt.xlabel("Years")
    plt.ylabel("Vehicles(million)")
    plt.legend()

    # save the graph on disk
```

```
    plt.savefig("fig3.png")



###################### Main Program ##########################

# get co2 emission data
co2_data = read_and_prepare_world_co2_data()
# create a line graph from co2 emission data
create_and_save_line_graph(co2_data)
# create pie chart to represent  co2 emission in years 1990 and 2020
create_and_save_pi_chart(co2_data)

# get ev car sale data
ev_data = read_and_prepare_ev_data()
# create a bar chart to represent the ev data
creat_and_save_bar_chart(ev_data)

# Display graph
plt.show()
```