```
import csv
import pandas as pd
import sqlite3 as sql

#Fonctions du projet

#Permet de transformer les valeurs d'une colonne d'un csv en int
def toInt(colCsv):
        for i in range(0,len(colCsv)):
                colCsv.iloc[i] = colCsv.iloc[i].replace(",",".")
        colCsv = colCsv.astype(float)
        colCsv = colCsv.astype(int)

#Permet d'extraire le nom du pays pour le csv de la France
def extrairePaysFrance(csvF):
    with open(csvF) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter = '-')
        list_of_column_names = []
        for row in csv_reader:
            list_of_column_names.append(row)
            break
    pays = list_of_column_names[0][0].lstrip(',')
    pays = pays.rstrip(' ')
    return pays

#Permet d'extraire le nom du pays ou du continent d'un csv
def extrairePays(csvP):
    with open(csvP) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter = '-')
        list_of_column_names = []
        for row in csv_reader:
            list_of_column_names.append(row)
            break
    pays = list_of_column_names[0][0][3:]
    pays = pays.rstrip(' ')
    return pays

#Permet d'extraire Asie pour le csv de l'Asie et Océanie
def extraireContinentAsie(csvA):
    with open(csvA) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter = '-')
        list_of_column_names = []
        for row in csv_reader:
            list_of_column_names.append(row)
            break
    pays = list_of_column_names[0][0][3:7]
    return pays

#Permet d'extraire Océanie pour le csv de l'Asie et Océanie
def extraireContinentOcéanie(csvO):
```

```
    with open(csv0) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter = '-')
        list_of_column_names = []
        for row in csv_reader:
            list_of_column_names.append(row)
            break
    pays = list_of_column_names[0][0][12:]
    return pays

#Permet de mettre les données d'un pays dans la base de données
def
mettreDonnées(dfCarbone,dfCarboneParHabitant,dfCarboneParPIB,numeroPay
s):
    for i in range(0,28):
        d = dfCarbone.iloc[i]
        p = dfCarboneParHabitant.iloc[i]
        pib = dfCarboneParPIB.iloc[i]
        res = [numeroPays,i+28*(numeroPays-1),str(d['Date'])
[0:4],str(d['Carbon']),str(p['Carbon']),str(pib['Carbon'])]
        cursor.execute('''INSERT INTO DonnéesPays
(PaysID,DonnéesID,Date,EmpreinteCarbone,ECPopulation,ECPIB) VALUES
(?,?,?,?,?,?);''',res)

#Permet de mettre les données d'un continent dans la base de données
def
mettreDonnéesContinent(dfCarbone,dfCarboneParHabitant,dfCarboneParPIB,
numeroContinent):
    for i in range(0,28):
        d = dfCarbone.iloc[i]
        p = dfCarboneParHabitant.iloc[i]
        pib = dfCarboneParPIB.iloc[i]
        res = [numeroContinent,i+28*(numeroContinent-1),str(d['Date'])
[0:4],str(d['Carbon']),str(p['Carbon']),str(pib['Carbon'])]
        cursor.execute('''INSERT INTO DonnéesContinent
(ContinentID,DonnéesIDC,Date,EmpreinteCarboneC,ECPopulationC,ECPIBC)
VALUES (?,?,?,?,?,?);''',res)

#Rempli la base de données avec tout ce qui est lié à la production
d'énergie
def Remplire(Data,Pays,TypeEnInt):
    query = str("SELECT Type FROM ProductionEnergie WHERE EnergieID=
%d" % (TypeEnInt))
    new_df = pd.read_sql(query, conn)
    type = new_df['Type'].iloc[0]
    for i in range(0,117):
        a = Data[type].iloc[i]
        if(pd.isna(a)):
            a = 0
        date = Data['Date'].iloc[i]
        rep = [Pays,TypeEnInt,a,date[0:4]]
```

```
        cursor.execute('''INSERT INTO
Utilise(PaysID,EnergieID,Pollution,DateDeLaDonnée) VALUES
(?,?,?,?)''',rep)

#Fait un graphe qui montre la pollution de chaque énergie par pays
def graphEnergie(Type):
    query = 'Select LIBELLÉ FROM Pays'
    df = pd.read_sql(query,conn)
    for i in df['LIBELLÉ']:
        query = '''
        select Pollution FROM Utilise
        INNER JOIN Pays on Utilise.PaysID = Pays.PaysID INNER JOIN
ProductionEnergie on Utilise.EnergieID = ProductionEnergie.EnergieID
        Where Type = "'''+ Type +'" and LIBELLÉ = "'+ i + '" and
DateDeLaDonnée = 2016'
        new_df = pd.read_sql(query, conn)
        new_df["Pollution"] = new_df["Pollution"].astype(str)
        for y in range(0,len(new_df["Pollution"])):
            new_df["Pollution"].iloc[y] =
new_df["Pollution"].iloc[y].replace(",",".")
        new_df["Pollution"] = new_df["Pollution"].astype(float)

        plt.bar(i,new_df['Pollution'])
    plt.title("Pollution de L'énergie de type " + Type + " par pays en
2016")
    plt.ylabel("Mtoe")
    plt.legend()
    plt.show()

#Fait un diagramme circulaire qui montre la part des pays dans
l'empreinte carbone mondiale
def Camembert(Data):
    query = 'Select LIBELLÉ FROM Pays'
    df = pd.read_sql(query,conn)
    l = []
    a = []
    for i in df['LIBELLÉ']:
        query = '''
        select EmpreinteCarbone from DonnéesPays INNER JOIN Pays on
DonnéesPays.PaysID = Pays.PaysID
        WHERE LIBELLÉ = "''' + i + '" and Date = 2017 '
        new_df = pd.read_sql(query, conn)
        new_df["EmpreinteCarbone"] =
new_df["EmpreinteCarbone"].astype(str)
        for y in range(0,len(new_df["EmpreinteCarbone"])):
            new_df["EmpreinteCarbone"].iloc[y] =
new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
        new_df["EmpreinteCarbone"] =
new_df["EmpreinteCarbone"].astype(float)
```

```
l.append(new_df["EmpreinteCarbone"].iloc[0]/Data["Pollution"].iloc[27]
*100)
        a.append(i)
    a.append("Other")
    b=0
    for i in l:
        b = b +i
    l.append(100-b)
    b = [str(i) for i in l]
    plt.rcParams["figure.figsize"] = (10, 10)
    plt.pie(l,labels=a,autopct= lambda b: str(round(b,2)) + "%")
    plt.rcParams['text.color'] = 'white'
    plt.title("La part de chacun des pays dans l'empreinte carbone
mondiale")
    plt.show()

#Permet de régler la latitude et la longitude de la carte du monde
def regleLongEtLat(df):
    for x in range(len(df['Longitude'])):
        if str(df.loc[x, 'Longitude'])[-1] == 'E':
            df.loc[x, 'Longitude'] = str(df.loc[x, 'Longitude'])[:-1]
        if str(df.loc[x, 'Longitude'])[-1] == 'W':
            df.loc[x, 'Longitude'] = \
                '-' + str(df.loc[x, 'Longitude'])[:-1]

    for x in range(len(df['Latitude'])):
        if str(df.loc[x, 'Latitude'])[-1] == 'N':
            df.loc[x, 'Latitude'] = str(df.loc[x, 'Latitude'])[:-1]
        if str(df.loc[x, 'Latitude'])[-1] == 'S':
            df.loc[x, 'Latitude'] = \
                '-' + str(df.loc[x, 'Latitude'])[:-1]

    df['Longitude'] = pd.to_numeric(df['Longitude'], errors='coerce')
    df['Latitude'] = pd.to_numeric(df['Latitude'], errors='coerce')

    #Supprimer les erreurs des conversions de données et réinitialiser
l'index
    df.dropna()
    df = df.reset_index(drop=True)

conn = sql.connect('DBCafésPierre.db')

cursor = conn.cursor()

#Création des tables

#Création de la table Continent
cursor.execute("""
 CREATE TABLE Continent(
    ContinentID INT NOT NULL,
```

```
    NomContinent VARCHAR(100) NOT NULL,
    PRIMARY KEY(ContinentID)
 )""")


query = 'select * from Continent'
new_df = pd.read_sql(query, conn)
new_df.head(5)

#Création de la table Pays
cursor.execute("""
CREATE TABLE Pays(
    PaysID INT NOT NULL,
    LIBELLÉ VARCHAR(100) NOT NULL,
    ContinentID INT NOT NULL,
    PRIMARY KEY(PaysID),
    FOREIGN KEY (ContinentID) REFERENCES Continent(ContinentID)
)
""")

#Création de la table DonnéesContinents
cursor.execute("""
CREATE TABLE DonnéesContinent(
    ContinentID INT NOT NULL,
    DonnéesIDC INT NOT NULL,
    EmpreinteCarboneC INT,
    ECPopulationC FLOAT,
    ECPIBC FLOAT,
    Date text NOT NULL,
    PRIMARY KEY(DonnéesIDC),
    FOREIGN KEY (ContinentID) REFERENCES Continent(ContinentID)
)
""")

#Création de la table DonnéesPays
cursor.execute("""
CREATE TABLE DonnéesPays(
    PaysID INT NOT NULL,
    DonnéesID INT NOT NULL,
    EmpreinteCarbone INT,
    ECPopulation FLOAT,
    ECPIB FLOAT,
    Date text NOT NULL,
    PRIMARY KEY(DonnéesID),
    FOREIGN KEY (PaysID) REFERENCES Pays(PaysID)
)
""")

#Création de la table ProductionEnergie
cursor.execute("""
```

```
CREATE TABLE ProductionEnergie(
    EnergieID INT NOT NULL,
    Type VARCHAR(100) NOT NULL,
    PRIMARY KEY(EnergieID)
)
""")

#Création de la table Utilise
cursor.execute("""
CREATE TABLE Utilise(
    PaysID INT NOT NULL,
    EnergieID INT NOT NULL,
    Pollution INT NOT NULL,
    DateDeLaDonnée text NOT NULL,
    PRIMARY KEY(PaysID,EnergieID,DateDeLaDonnée),
    FOREIGN KEY (PaysID) REFERENCES Pays(PaysID),
    FOREIGN KEY (EnergieID) REFERENCES ProductionEnergie(EnergieID)
)
""")
```

```
<sqlite3.Cursor at 0x7f0005e230a0>
```

```
#Attribution des noms de pays
p1 = extrairePaysFrance('./ToutesDataFrance/Carbon Footprint, France,
1990-2017 (in MtCO2).csv')
p2 = extrairePays('./ToutesDataAllemagne/Carbon Footprint, Germany,
1990-2017 (in MtCO2).csv')
p3 = extrairePays('./ToutesDataCoteIvoire/Carbon Footprint, Ivory
Coast, 1990-2017 (in MtCO2).csv')
p4 = extrairePays('./ToutesDataChine/Carbon Footprint, China, 1990-
2017 (in MtCO2).csv')
p5 = extrairePays('./ToutesDataInde/Carbon Footprint, India, 1990-2017
(in MtCO2).csv')
p6 = extrairePays('./ToutesDataEtatsUnis/Carbon Footprint, United
States of America, 1990-2017 (in MtCO2).csv')
p7 = extrairePays('./ToutesDataDanemark/Carbon Footprint, Denmark,
1990-2017 (in MtCO2).csv')

#Attribution des noms de continents
c1 = extrairePays('./ToutesDataEurope/Carbon Footprint, Europe, 1990-
2017 (in MtCO2).csv')
c2 = extraireContinentAsie('./ToutesDataAsieOcéanie/Carbon Footprint,
Asia and Oceania, 1990-2017 (in MtCO2).csv')
c3 = extrairePays('./ToutesDataAfrique/Carbon Footprint, Africa, 1990-
2017 (in MtCO2).csv')
c4 = extrairePays('./ToutesDataAmériqueNord/Carbon Footprint, North
America, 1990-2017 (in MtCO2).csv')
c5 = extraireContinentOcéanie('./ToutesDataAsieOcéanie/Carbon
Footprint, Asia and Oceania, 1990-2017 (in MtCO2).csv')
```

```
#Remplissage de la table Pays
sqll = '''INSERT INTO Pays (PaysID,LIBELLÉ,ContinentID) VALUES (1,"'''
+ p1 + '''",1),(2,"''' + p2 + '''",1),(3,"''' + p3 + '''",3),(4,"''' +
p4 + '''",2),(5,"''' + p5 + '''",2),(6,"''' + p6 + '''",4),(7,"''' +
p7 + '''",1)'''
cursor.execute(sqll)
print("INSERT in pays : OK")

#Remplissage de la table Continent
sqll = '''INSERT INTO Continent (ContinentID,NomContinent) VALUES
(1,"''' + c1 + '''"),(2,"''' + c2 + '''"),(3,"''' + c3 + '''"),(4,"'''
+ c4 +'''"),(5,"''' + c5 +'''")'''
cursor.execute(sqll)
print("INSERT in Continent : OK")

#Remplissage de la table ProductionEnergie
sqll = '''INSERT INTO ProductionEnergie (EnergieID,Type) VALUES
(1,"Oil"),(2,"Coal"),(3,"Nuclear"),(4,"Gas"),(5,"Hydroelectricity"),
(6,"Biomass and Waste"),(7,"Wind"),(8,"Fuel Ethanol"),
(9,"Geothermal"),(10,"Biodiesel"),(11,"Peat"),(12,"Solar, Tide, Wave,
Fuel Cell") '''
cursor.execute(sqll)
print("INSERT in ProductionEnergie : OK")
```

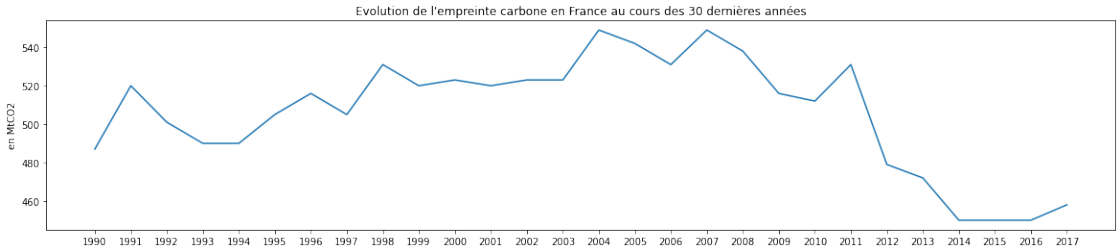
```
INSERT in pays : OK
INSERT in Continent : OK
INSERT in ProductionEnergie : OK
```

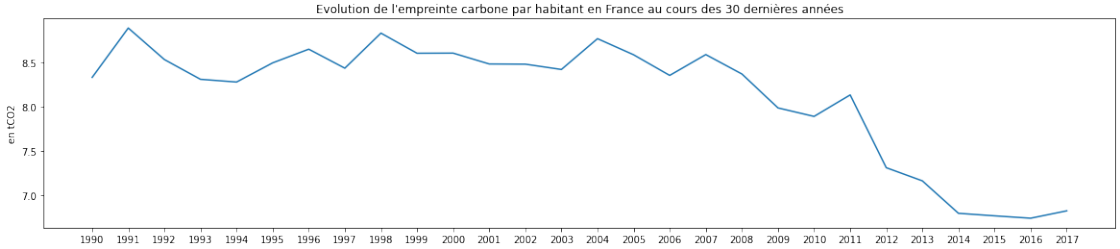
```
#Tout les csv de la France

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneFrance = pd.read_csv("./ToutesDataFrance/Carbon Footprint,
France, 1990-2017 (in MtCO2).csv",',')
dfCarboneFrance = dfCarboneFrance.rename(columns={"Unnamed: 0":
'Date',
"France - Carbon Footprint": 'Carbon'})

dfCarboneFranceParHabitant = pd.read_csv("./ToutesDataFrance/Carbon
Footprint per capita, France, 1990-2017 (in tCO2).csv",';')
dfCarboneFranceParHabitant =
dfCarboneFranceParHabitant.rename(columns={"Unnamed: 0": 'Date',
"France - Carbon Footprint": 'Carbon'})

toInt(dfCarboneFranceParHabitant['Carbon'])

dfCarboneFranceParPIB = pd.read_csv("./ToutesDataFrance/Carbon
Footprint per GDP, France, 1990-2017 (in tCO2).csv",";")
```

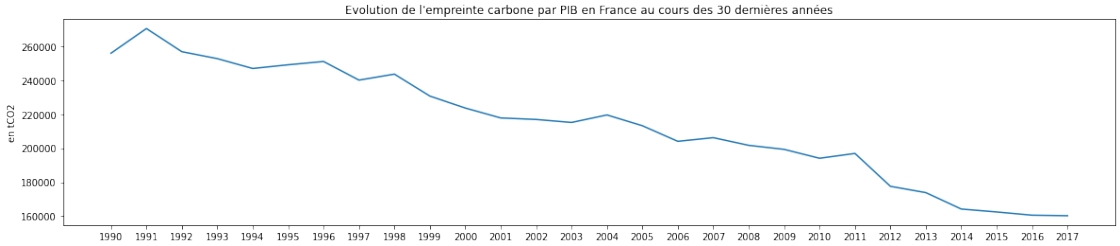
```
dfCarboneFranceParPIB =
dfCarboneFranceParPIB.rename(columns={"Unnamed: 0": 'Date', "France -
Carbon Footprint": 'Carbon'})

toInt(dfCarboneFranceParPIB['Carbon'])

mettreDonnées(dfCarboneFrance,dfCarboneFranceParHabitant,dfCarboneFran
ceParPIB,1)
query = 'select * from DonnéesPays'

#Afficher l'évolution de l'empreinte carbone en France durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone en France au cours des 30
dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en France
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant en France au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en France durant
les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB en France au cours
des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
```

```
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





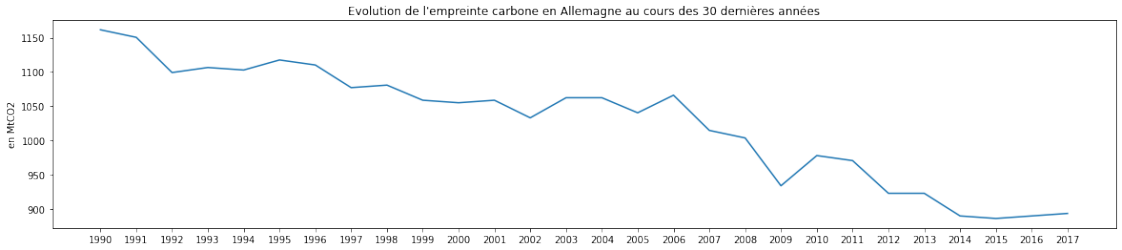


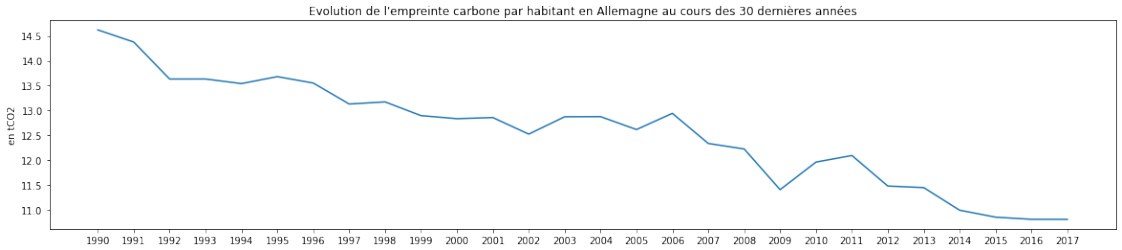

```
#Tous les csv de l'Allemagne

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneAllemagne = pd.read_csv("./ToutesDataAllemagne/Carbon
Footprint, Germany, 1990-2017 (in MtCO2).csv",';')
dfCarboneAllemagne = dfCarboneAllemagne.rename(columns={"Unnamed: 0":
'Date',
"Germany - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAllemagne['Carbon'])

dfCarboneAllemagneParHabitant =
pd.read_csv("./ToutesDataAllemagne/Carbon Footprint per capita,
Germany, 1990-2017 (in tCO2).csv",';')
dfCarboneAllemagneParHabitant =
dfCarboneAllemagneParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Germany - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAllemagneParHabitant['Carbon'])

dfCarboneAllemagneParPIB = pd.read_csv("./ToutesDataAllemagne/Carbon
```

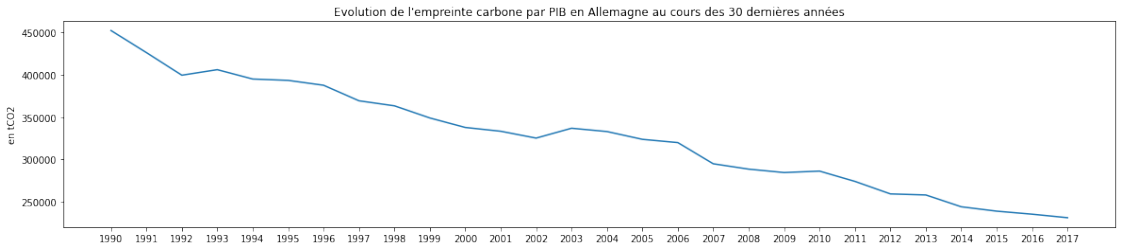
```
Footprint per GDP, Germany, 1990-2017 (in tCO2).csv",";")
dfCarboneAllemagneParPIB =
dfCarboneAllemagneParPIB.rename(columns={"Unnamed: 0": 'Date',
"Germany - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAllemagneParPIB['Carbon'])

mettreDonnées(dfCarboneAllemagne,dfCarboneAllemagneParHabitant,dfCarbo
neAllemagneParPIB,2)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "Germany"'

#Afficher l'évolution de l'empreinte carbone en Allemagne durant les
30 dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone en Allemagne au cours des
30 dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Allemagne
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant en Allemagne
au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Allemagne
durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB en Allemagne au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





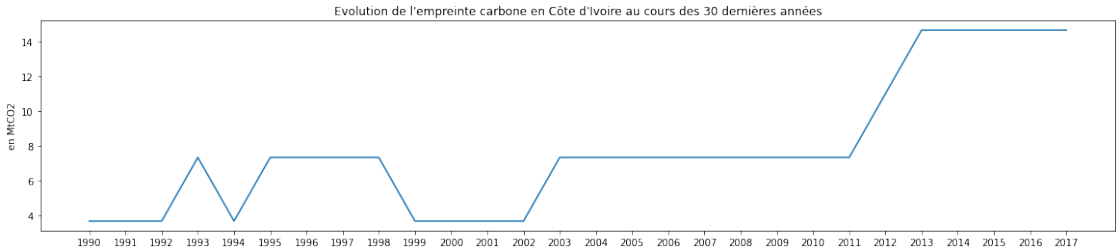


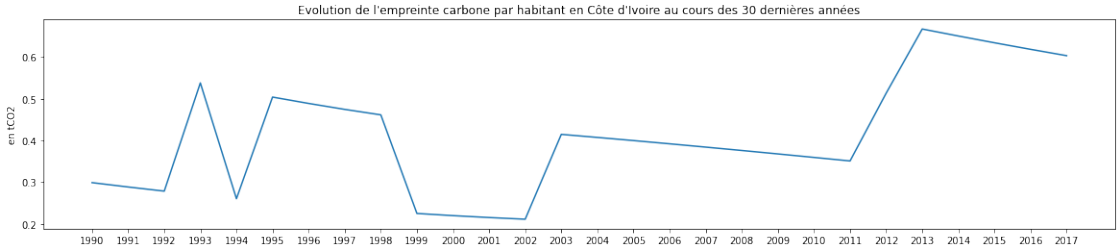

```
#Tous les csv de la Côte d'Ivoire

import matplotlib.pyplot as plt
import pandas as pd
import datetime

dfCarboneCôteIvoire = pd.read_csv("./ToutesDataCoteIvoire/Carbon
Footprint, Ivory Coast, 1990-2017 (in MtCO2).csv",';')
dfCarboneCôteIvoire = dfCarboneCôteIvoire.rename(columns={"Unnamed:
0": 'Date',
"Ivory Coast - Carbon Footprint": 'Carbon'})

toInt(dfCarboneCôteIvoire['Carbon'])

dfCarboneCôteIvoireParHabitant =
pd.read_csv("./ToutesDataCoteIvoire/Carbon Footprint per capita, Ivory
Coast, 1990-2017 (in tCO2).csv",';')
dfCarboneCôteIvoireParHabitant =
dfCarboneCôteIvoireParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Ivory Coast - Carbon Footprint": 'Carbon'})
```

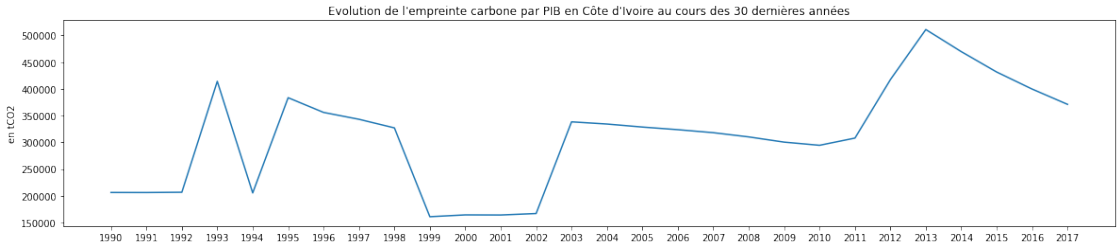
```
toInt(dfCarboneCôteIvoireParHabitant['Carbon'])

dfCarboneCôteIvoireParPIB = pd.read_csv("./ToutesDataCoteIvoire/Carbon
Footprint per GDP, Ivory Coast, 1990-2017 (in tCO2).csv",";")
dfCarboneCôteIvoireParPIB =
dfCarboneCôteIvoireParPIB.rename(columns={"Unnamed: 0": 'Date', "Ivory
Coast - Carbon Footprint": 'Carbon'})

toInt(dfCarboneCôteIvoireParPIB['Carbon'])

mettreDonnées(dfCarboneCôteIvoire,dfCarboneCôteIvoireParHabitant,dfCar
boneCôteIvoireParPIB,3)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "Ivory Coast"'

#Afficher l'évolution de l'empreinte carbone en Côte d'Ivoire durant
les 30 dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone en Côte d'Ivoire au cours
des 30 dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Côte
d'Ivoire durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant en Côte
d'Ivoire au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Côte d'Ivoire
durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB en Côte d'Ivoire
au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
```

```
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





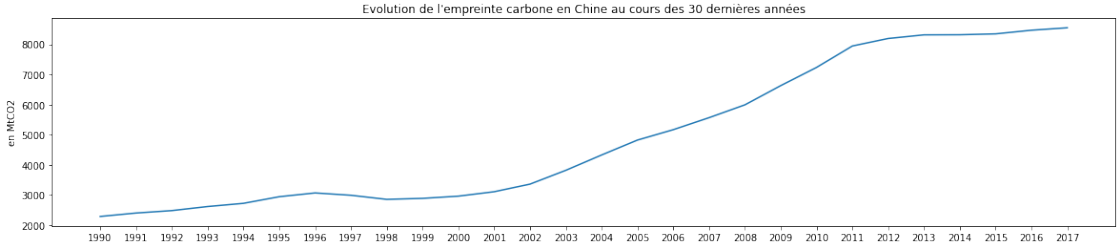


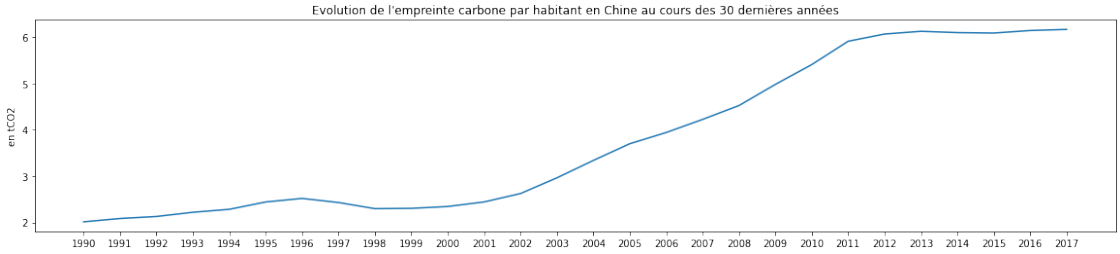

```
#Tous les csv de la Chine

import matplotlib.pyplot as plt
import pandas as pd
import datetime

dfCarboneChine = pd.read_csv("./ToutesDataChine/Carbon Footprint,
China, 1990-2017 (in MtCO2).csv",';')
dfCarboneChine = dfCarboneChine.rename(columns={"Unnamed: 0": 'Date',
"China - Carbon Footprint": 'Carbon'})

toInt(dfCarboneChine['Carbon'])

dfCarboneChineParHabitant = pd.read_csv("./ToutesDataChine/Carbon
Footprint per capita, China, 1990-2017 (in tCO2).csv",';')
dfCarboneChineParHabitant =
dfCarboneChineParHabitant.rename(columns={"Unnamed: 0": 'Date', "China
- Carbon Footprint": 'Carbon'})
```

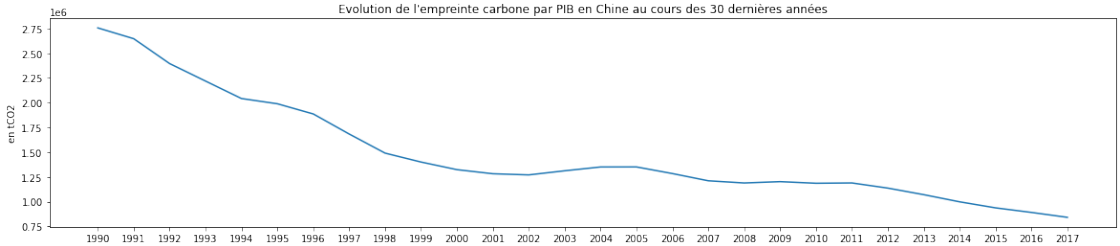
```
toInt(dfCarboneChineParHabitant['Carbon'])

dfCarboneChineParPIB = pd.read_csv("./ToutesDataChine/Carbon Footprint
per GDP, China, 1990-2017 (in tCO2).csv",";")
dfCarboneChineParPIB = dfCarboneChineParPIB.rename(columns={"Unnamed:
0": 'Date', "China - Carbon Footprint": 'Carbon'})

toInt(dfCarboneChineParPIB['Carbon'])

mettreDonnées(dfCarboneChine,dfCarboneChineParHabitant,dfCarboneChineP
arPIB,4)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "China"'

#Afficher l'évolution de l'empreinte carbone en Chine durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone en Chine au cours des 30
dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Chine
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant en Chine au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Chine durant
les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB en Chine au cours
des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
```

```
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





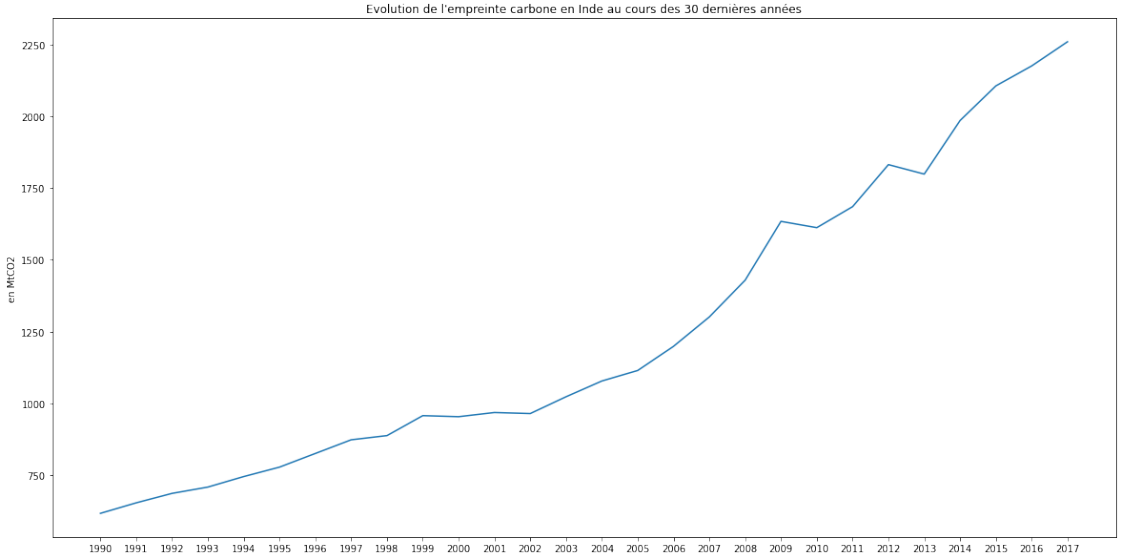


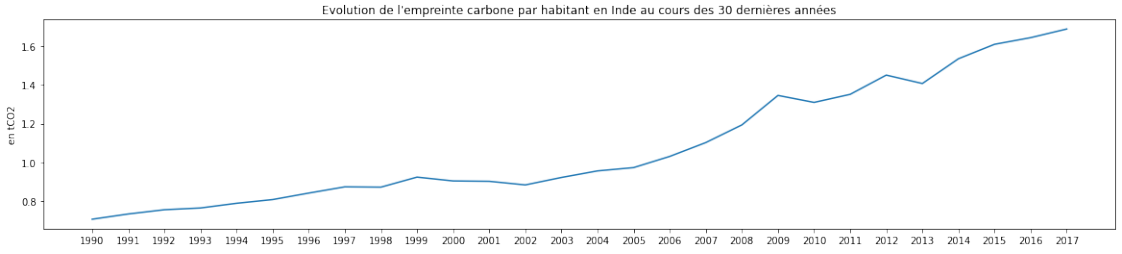

```
#Tous les csv de l'Inde

import matplotlib.pyplot as plt
import pandas as pd
import datetime

dfCarboneInde = pd.read_csv("./ToutesDataInde/Carbon Footprint, India,
1990-2017 (in MtCO2).csv",';')
dfCarboneInde = dfCarboneInde.rename(columns={"Unnamed: 0": 'Date',
"India - Carbon Footprint": 'Carbon'})

toInt(dfCarboneInde['Carbon'])

dfCarboneIndeParHabitant = pd.read_csv("./ToutesDataInde/Carbon
Footprint per capita, India, 1990-2017 (in tCO2).csv",';')
dfCarboneIndeParHabitant =
dfCarboneIndeParHabitant.rename(columns={"Unnamed: 0": 'Date', "India
- Carbon Footprint": 'Carbon'})
```

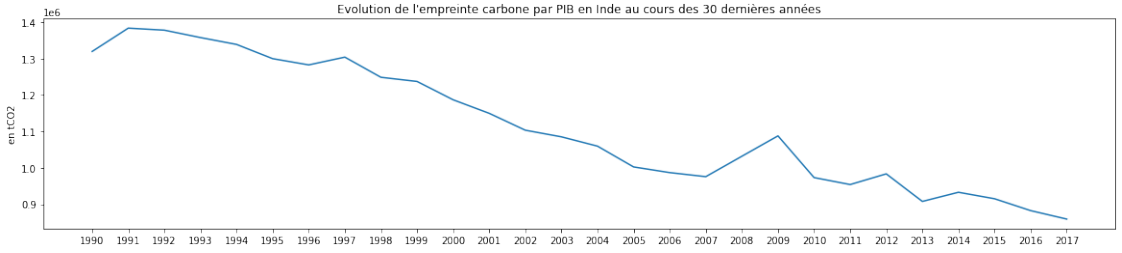
```
toInt(dfCarboneIndeParHabitant['Carbon'])

dfCarboneIndeParPIB = pd.read_csv("./ToutesDataInde/Carbon Footprint
per GDP, India, 1990-2017 (in tCO2).csv",";")
dfCarboneIndeParPIB = dfCarboneIndeParPIB.rename(columns={"Unnamed:
0": 'Date', "India - Carbon Footprint": 'Carbon'})

toInt(dfCarboneIndeParPIB['Carbon'])

mettreDonnées(dfCarboneInde,dfCarboneIndeParHabitant,dfCarboneIndeParP
IB,5)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "India"'

#Afficher l'évolution de l'empreinte carbone en Inde durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,10)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone en Inde au cours des 30
dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Inde
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant en Inde au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Inde durant
les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB en Inde au cours
des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
```

```
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





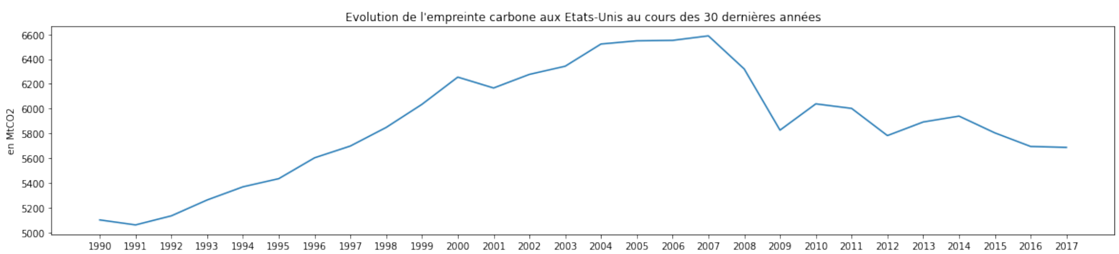


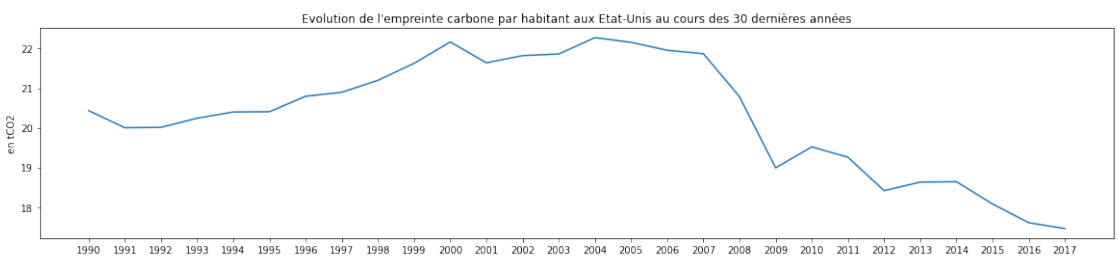

```
#Tous les csv des Etats-Unis

import matplotlib.pyplot as plt
import pandas as pd
import datetime

dfCarboneEtatsUnis = pd.read_csv("./ToutesDataEtatsUnis/Carbon
Footprint, United States of America, 1990-2017 (in MtCO2).csv",';')
dfCarboneEtatsUnis = dfCarboneEtatsUnis.rename(columns={"Unnamed: 0":
```

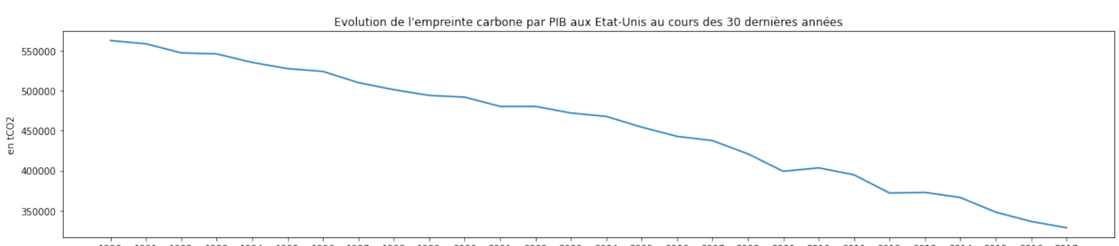
```
'Date',
"United States of America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneEtatsUnis['Carbon'])

dfCarboneEtatsUnisParHabitant =
pd.read_csv("./ToutesDataEtatsUnis/Carbon Footprint per capita, United
States of America, 1990-2017 (in tCO2).csv",';')
dfCarboneEtatsUnisParHabitant =
dfCarboneEtatsUnisParHabitant.rename(columns={"Unnamed: 0": 'Date',
"United States of America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneEtatsUnisParHabitant['Carbon'])

dfCarboneEtatsUnisParPIB = pd.read_csv("./ToutesDataEtatsUnis/Carbon
Footprint per GDP, United States of America, 1990-2017 (in
tCO2).csv",";")
dfCarboneEtatsUnisParPIB =
dfCarboneEtatsUnisParPIB.rename(columns={"Unnamed: 0": 'Date', "United
States of America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneEtatsUnisParPIB['Carbon'])

mettreDonnées(dfCarboneEtatsUnis,dfCarboneEtatsUnisParHabitant,dfCarbo
neEtatsUnisParPIB,6)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "United States
of America"'

#Afficher l'évolution de l'empreinte carbone aux Etat-Unis durant les
30 dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone aux Etats-Unis au cours
des 30 dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant aux Etat-
Unis durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant aux Etat-Unis
au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
#Afficher l'évolution de l'empreinte carbone par PIB aux Etat-Unis durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB aux Etat-Unis au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```





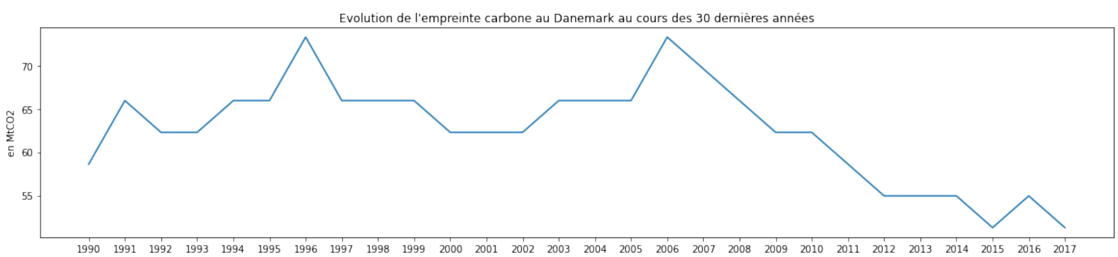


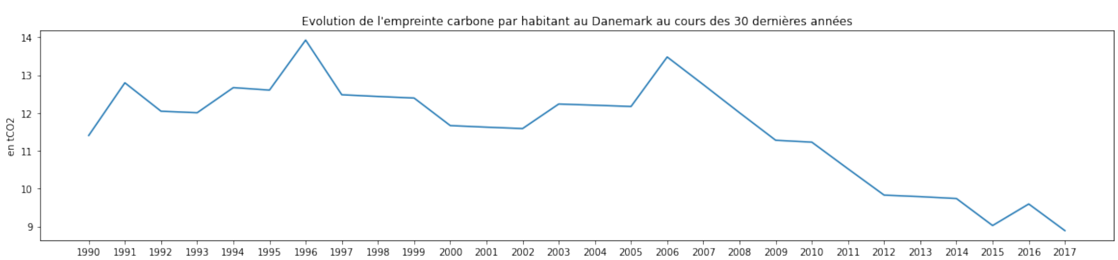

```
#Tous les csv du Danemark

import matplotlib.pyplot as plt
```

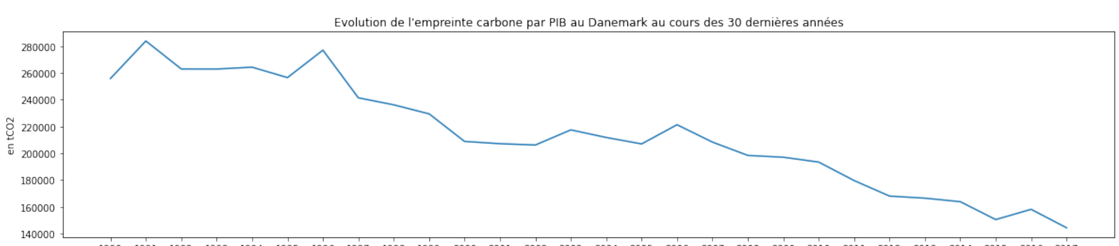
```
import pandas as pd
import datetime

dfCarboneDanemark = pd.read_csv("./ToutesDataDanemark/Carbon
Footprint, Denmark, 1990-2017 (in MtCO2).csv",';')
dfCarboneDanemark = dfCarboneDanemark.rename(columns={"Unnamed: 0":
'Date',
"Denmark - Carbon Footprint": 'Carbon'})

toInt(dfCarboneDanemark['Carbon'])

dfCarboneDanemarkParHabitant =
pd.read_csv("./ToutesDataDanemark/Carbon Footprint per capita,
Denmark, 1990-2017 (in tCO2).csv",';')
dfCarboneDanemarkParHabitant =
dfCarboneDanemarkParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Denmark - Carbon Footprint": 'Carbon'})

toInt(dfCarboneDanemarkParHabitant['Carbon'])

dfCarboneDanemarkParPIB = pd.read_csv("./ToutesDataDanemark/Carbon
Footprint per GDP, Denmark, 1990-2017 (in tCO2).csv",";")
dfCarboneDanemarkParPIB =
dfCarboneDanemarkParPIB.rename(columns={"Unnamed: 0": 'Date', "Denmark
- Carbon Footprint": 'Carbon'})

toInt(dfCarboneDanemarkParPIB['Carbon'])

mettreDonnées(dfCarboneDanemark,dfCarboneDanemarkParHabitant,dfCarbone
DanemarkParPIB,7)
query = 'select * from DonnéesPays inner join Pays on
DonnéesPays.PaysID = Pays.PaysID where Pays.LIBELLÉ = "Denmark"'

#Afficher l'évolution de l'empreinte carbone au Danemark durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarbone'])
plt.title("Evolution de l'empreinte carbone au Danemark au cours des
30 dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant au Danemark
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulation'])
plt.title("Evolution de l'empreinte carbone par habitant au Danemark
```

```
au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB au Danemark
durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIB'])
plt.title("Evolution de l'empreinte carbone par PIB au Danemark au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```



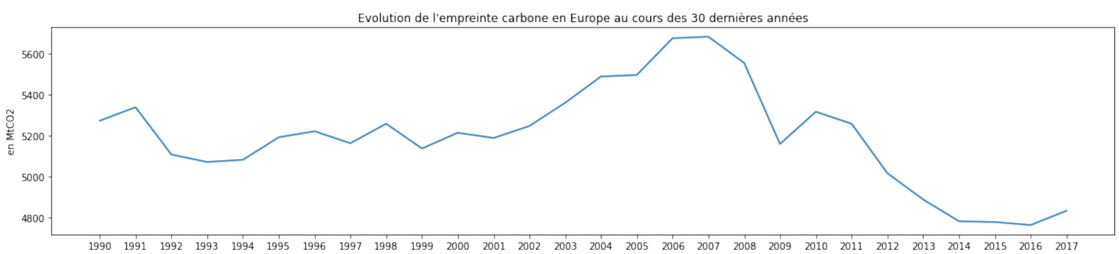


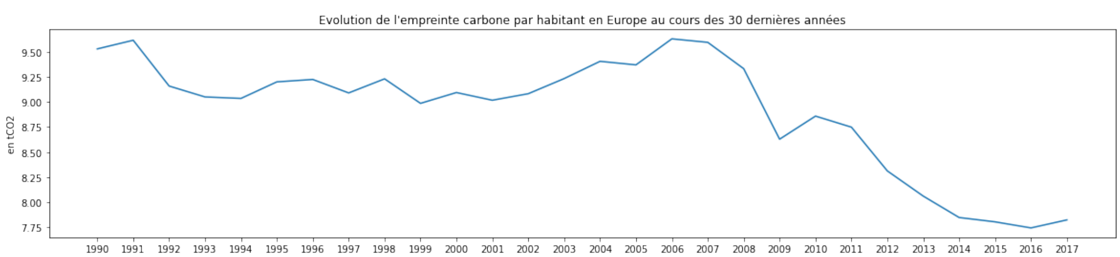


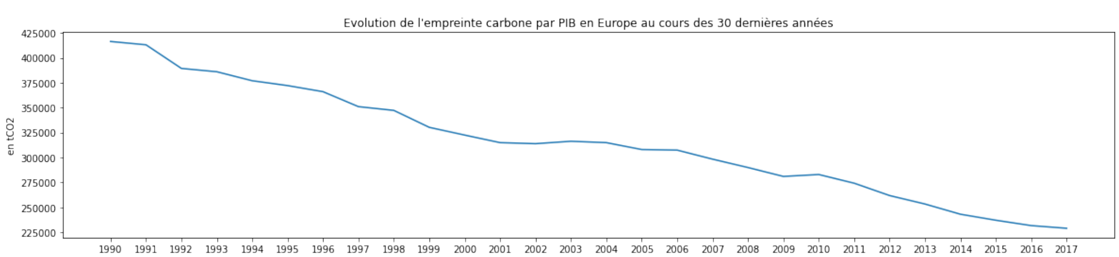
```
#Tout les csv de l'Europe

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneEurope = pd.read_csv("./ToutesDataEurope/Carbon Footprint,
Europe, 1990-2017 (in MtCO2).csv",';')
dfCarboneEurope = dfCarboneEurope.rename(columns={"Unnamed: 0":
'Date',
"Europe - Carbon Footprint": 'Carbon'})

toInt(dfCarboneEurope['Carbon'])

dfCarboneEuropeParHabitant = pd.read_csv("./ToutesDataEurope/Carbon
Footprint per capita, Europe, 1990-2017 (in tCO2).csv",';')
dfCarboneEuropeParHabitant =
dfCarboneEuropeParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Europe - Carbon Footprint": 'Carbon'})

toInt(dfCarboneEuropeParHabitant['Carbon'])

dfCarboneEuropeParPIB = pd.read_csv("./ToutesDataEurope/Carbon
Footprint per GDP, Europe, 1990-2017 (in tCO2).csv",";")
dfCarboneEuropeParPIB =
dfCarboneEuropeParPIB.rename(columns={"Unnamed: 0": 'Date', "Europe -
Carbon Footprint": 'Carbon'})

toInt(dfCarboneEuropeParPIB['Carbon'])

mettreDonnéesContinent(dfCarboneEurope,dfCarboneEuropeParHabitant,dfCa
rboneEuropeParPIB,1)
query = 'select * from DonnéesContinent'

#Afficher l'évolution de l'empreinte carbone en Europe durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarboneC'])
plt.title("Evolution de l'empreinte carbone en Europe au cours des 30
dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Europe
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulationC'])
plt.title("Evolution de l'empreinte carbone par habitant en Europe au
```

```
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Europe durant
les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIBC'])
plt.title("Evolution de l'empreinte carbone par PIB en Europe au cours
des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```



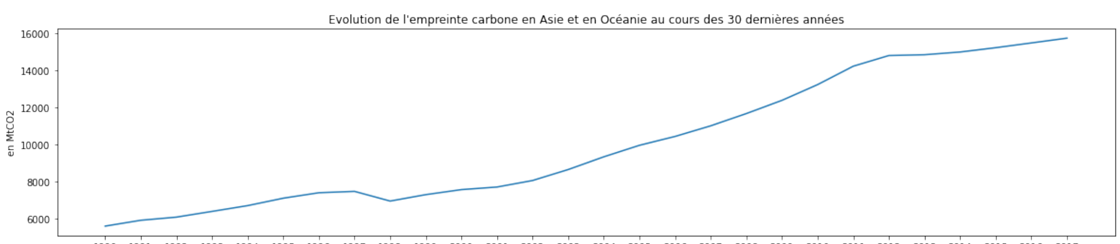


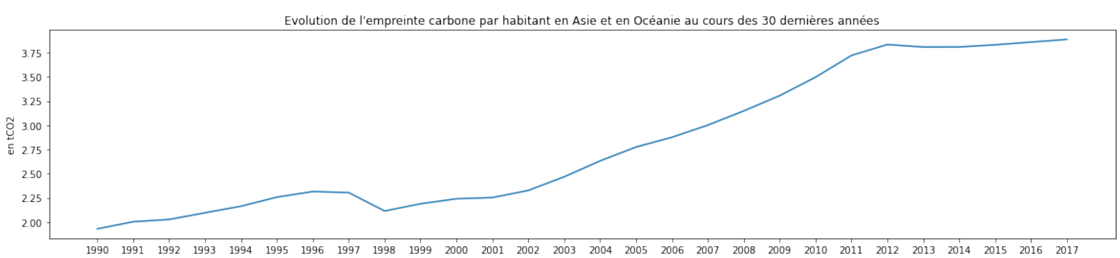


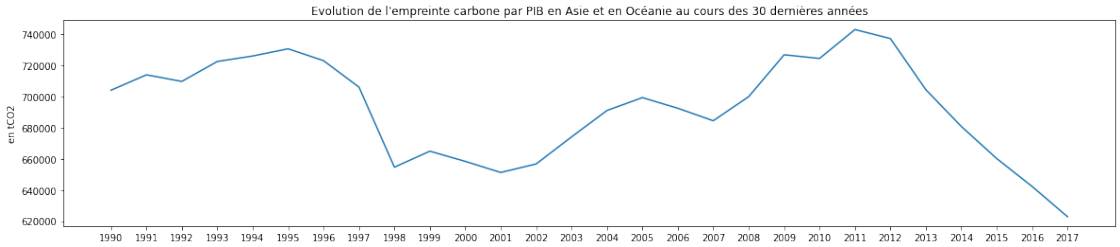
```
#Tout les csv de l'Asie et de l'Océanie

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneAsieOcéanie = pd.read_csv("./ToutesDataAsieOcéanie/Carbon
Footprint, Asia and Oceania, 1990-2017 (in MtCO2).csv",';')
dfCarboneAsieOcéanie = dfCarboneAsieOcéanie.rename(columns={"Unnamed:
0": 'Date',
"Asia and Oceania - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAsieOcéanie['Carbon'])

dfCarboneAsieOcéanieParHabitant =
pd.read_csv("./ToutesDataAsieOcéanie/Carbon Footprint per capita, Asia
and Oceania, 1990-2017 (in tCO2).csv",';')
dfCarboneAsieOcéanieParHabitant =
dfCarboneAsieOcéanieParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Asia and Oceania - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAsieOcéanieParHabitant['Carbon'])

dfCarboneAsieOcéanieParPIB =
pd.read_csv("./ToutesDataAsieOcéanie/Carbon Footprint per GDP, Asia
and Oceania, 1990-2017 (in tCO2).csv",";")
dfCarboneAsieOcéanieParPIB =
dfCarboneAsieOcéanieParPIB.rename(columns={"Unnamed: 0": 'Date', "Asia
and Oceania - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAsieOcéanieParPIB['Carbon'])

mettreDonnéesContinent(dfCarboneAsieOcéanie,dfCarboneAsieOcéanieParHab
itant,dfCarboneAsieOcéanieParPIB,2)
mettreDonnéesContinent(dfCarboneAsieOcéanie,dfCarboneAsieOcéanieParHab
itant,dfCarboneAsieOcéanieParPIB,5)
query = 'select * from DonnéesContinent inner join Continent on
DonnéesContinent.ContinentID = Continent.ContinentID where
NomContinent = "Asia" or NomContinent = "Oceania"'

#Afficher l'évolution de l'empreinte carbone en Asie et en Océanie
durant les 30 dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarboneC'])
plt.title("Evolution de l'empreinte carbone en Asie et en Océanie au
cours des 30 dernières années")
plt.ylabel('en MtCO2')
plt.show()
```

```
#Afficher l'évolution de l'empreinte carbone par habitant en Asie et en Océanie durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulationC'])
plt.title("Evolution de l'empreinte carbone par habitant en Asie et en Océanie au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Asie et en Océanie durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIBC'])
plt.title("Evolution de l'empreinte carbone par PIB en Asie et en Océanie au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```



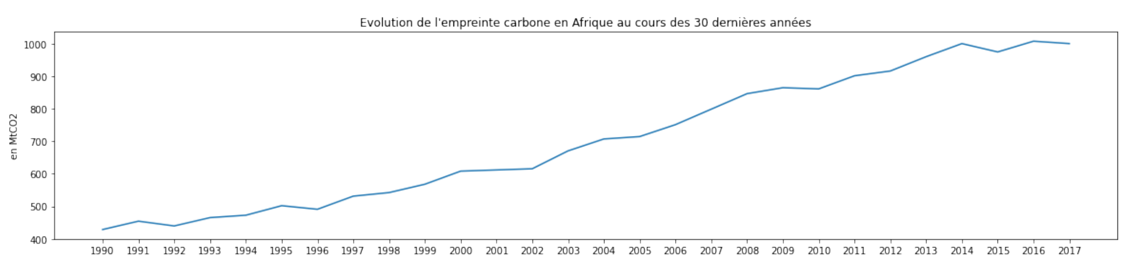


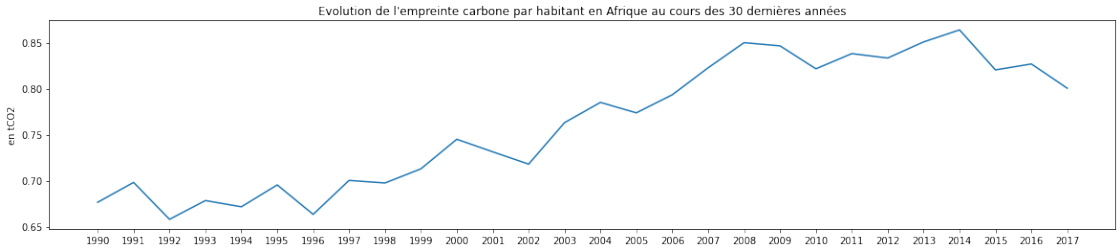


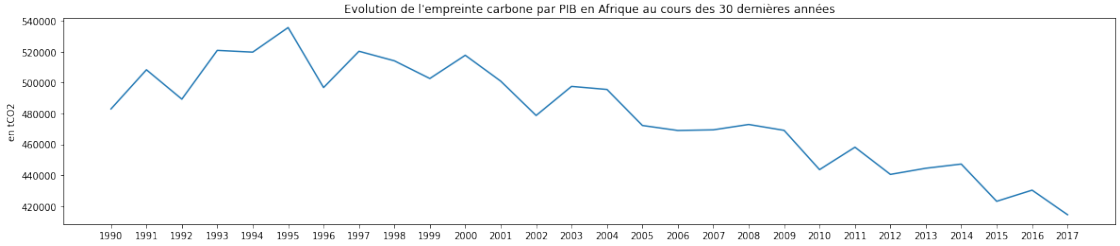

```
#Tout les csv de l'Afrique

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneAfrique = pd.read_csv("./ToutesDataAfrique/Carbon Footprint,
Africa, 1990-2017 (in MtCO2).csv",';')
dfCarboneAfrique = dfCarboneAfrique.rename(columns={"Unnamed: 0":
'Date',
"Africa - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAfrique['Carbon'])

dfCarboneAfriqueParHabitant = pd.read_csv("./ToutesDataAfrique/Carbon
Footprint per capita, Africa, 1990-2017 (in tCO2).csv",';')
dfCarboneAfriqueParHabitant =
dfCarboneAfriqueParHabitant.rename(columns={"Unnamed: 0": 'Date',
"Africa - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAfriqueParHabitant['Carbon'])

dfCarboneAfriqueParPIB = pd.read_csv("./ToutesDataAfrique/Carbon
Footprint per GDP, Africa, 1990-2017 (in tCO2).csv",";")
dfCarboneAfriqueParPIB =
dfCarboneAfriqueParPIB.rename(columns={"Unnamed: 0": 'Date', "Africa -
Carbon Footprint": 'Carbon'})

toInt(dfCarboneAfriqueParPIB['Carbon'])

mettreDonnéesContinent(dfCarboneAfrique,dfCarboneAfriqueParHabitant,df
CarboneAfriqueParPIB,3)
query = 'select * from DonnéesContinent inner join Continent on
DonnéesContinent.ContinentID = Continent.ContinentID where
NomContinent = "Africa"'

#Afficher l'évolution de l'empreinte carbone en Afrique durant les 30
dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarboneC'])
plt.title("Evolution de l'empreinte carbone en Afrique au cours des 30
```

```
dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Afrique
durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulationC'])
plt.title("Evolution de l'empreinte carbone par habitant en Afrique au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Afrique durant
les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIBC'])
plt.title("Evolution de l'empreinte carbone par PIB en Afrique au
cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```



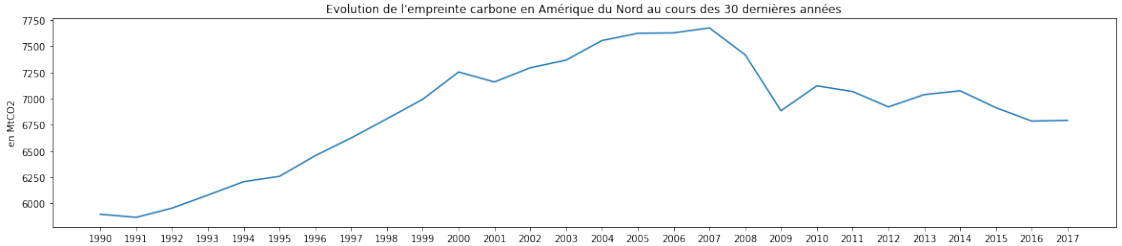


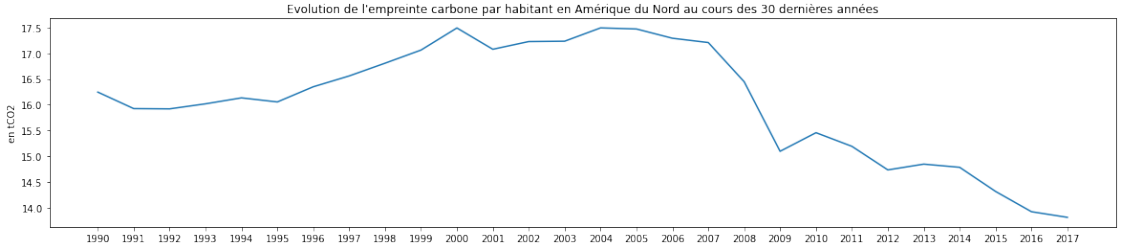


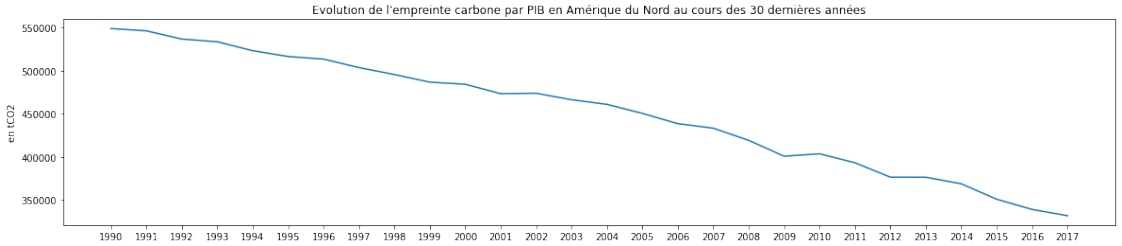

```
#Tout les csv de l'Amérique du Nord

import matplotlib.pyplot as plt
import pandas as pd

dfCarboneAmériqueNord = pd.read_csv("./ToutesDataAmériqueNord/Carbon
Footprint, North America, 1990-2017 (in MtCO2).csv",';')
dfCarboneAmériqueNord =
dfCarboneAmériqueNord.rename(columns={"Unnamed: 0": 'Date',
"North America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAmériqueNord['Carbon'])

dfCarboneAmériqueNordParHabitant =
pd.read_csv("./ToutesDataAmériqueNord/Carbon Footprint per capita,
North America, 1990-2017 (in tCO2).csv",';')
dfCarboneAmériqueNordParHabitant =
dfCarboneAmériqueNordParHabitant.rename(columns={"Unnamed: 0": 'Date',
"North America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAmériqueNordParHabitant['Carbon'])

dfCarboneAmériqueNordParPIB =
pd.read_csv("./ToutesDataAmériqueNord/Carbon Footprint per GDP, North
America, 1990-2017 (in tCO2).csv",";")
dfCarboneAmériqueNordParPIB =
dfCarboneAmériqueNordParPIB.rename(columns={"Unnamed: 0": 'Date',
"North America - Carbon Footprint": 'Carbon'})

toInt(dfCarboneAmériqueNordParPIB['Carbon'])

mettreDonnéesContinent(dfCarboneAmériqueNord,dfCarboneAmériqueNordParH
abitant,dfCarboneAmériqueNordParPIB,4)
```

```
query = 'select * from DonnéesContinent inner join Continent on
DonnéesContinent.ContinentID = Continent.ContinentID where
NomContinent = "North America"'

#Afficher l'évolution de l'empreinte carbone en Amérique du Nord
durant les 30 dernières années
new_df = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df['Date'],new_df['EmpreinteCarboneC'])
plt.title("Evolution de l'empreinte carbone en Amérique du Nord au
cours des 30 dernières années")
plt.ylabel('en MtCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par habitant en Amérique
du Nord durant les 30 dernières années
new_df2 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df2['Date'],new_df2['ECPopulationC'])
plt.title("Evolution de l'empreinte carbone par habitant en Amérique
du Nord au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()

#Afficher l'évolution de l'empreinte carbone par PIB en Amérique du
Nord durant les 30 dernières années
new_df3 = pd.read_sql(query, conn)
plt.rcParams["figure.figsize"] = (20,4)
plt.plot(new_df3['Date'],new_df3['ECPIBC'])
plt.title("Evolution de l'empreinte carbone par PIB en Amérique du
Nord au cours des 30 dernières années")
plt.ylabel('en tCO2')
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```








```
query = "SELECT * FROM ProductionEnergie"
new_df = pd.read_sql(query, conn)
new_df
```

	EnergieID	Type
0	1	Oil
1	2	Coal
2	3	Nuclear
3	4	Gas
4	5	Hydroelectricity
5	6	Biomass and Waste
6	7	Wind
7	8	Fuel Ethanol
8	9	Geothermal
9	10	Biodiesel
10	11	Peat
11	12	Solar, Tide, Wave, Fuel Cell

```
#Importer des bibliothèques

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

df = pd.read_csv("./ToutesDataFrance/Primary Energy Production by
```

```
source, France, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par la
France
for i in range(1,13):
    Remplire(df,1,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df = pd.read_csv("./ToutesDataAllemagne/Primary Energy Production by
source, Germany, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par
l'Allemagne
for i in range(1,13):
    Remplire(df,2,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df = pd.read_csv("./ToutesDataCoteIvoire/Primary Energy Production by
source, Ivory Coast, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par la
Côte d'Ivoire
for i in range(1,13):
    Remplire(df,3,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df = pd.read_csv("./ToutesDataChine/Primary Energy Production by
source, China, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par la
Chine
for i in range(1,13):
    Remplire(df,4,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df = pd.read_csv("./ToutesDataInde/Primary Energy Production by
source, India, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par
l'Inde
for i in range(1,13):
    Remplire(df,5,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
df = pd.read_csv("./ToutesDataEtatsUnis/Primary Energy Production by
source, United States of America, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par les
Etats-Unis
for i in range(1,13):
    Remplire(df,6,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

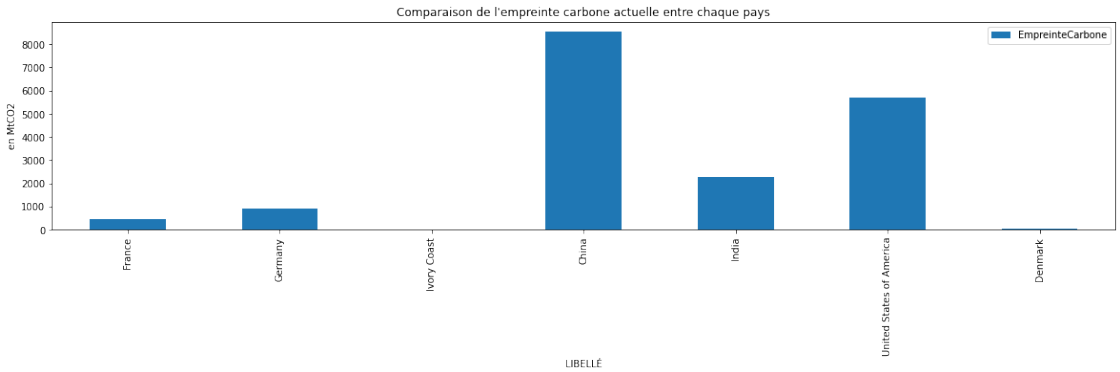
```
df = pd.read_csv("./ToutesDataDanemark/Primary Energy Production by
source, Denmark, 1900-2016 (in Mtoe).csv",";")
df = df.rename(columns={"Unnamed: 0": 'Date'})

#Rempli la base de données avec toutes les énergies produites par le
Danemark
for i in range(1,13):
    Remplire(df,7,i)
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
#Affiche l'empreinte carbone pour chaque pays

query = 'select EmpreinteCarbone,LIBELLÉ from DonnéesPays INNER JOIN
Pays on DonnéesPays.PaysID = Pays.PaysID WHERE Date = 2017'
```

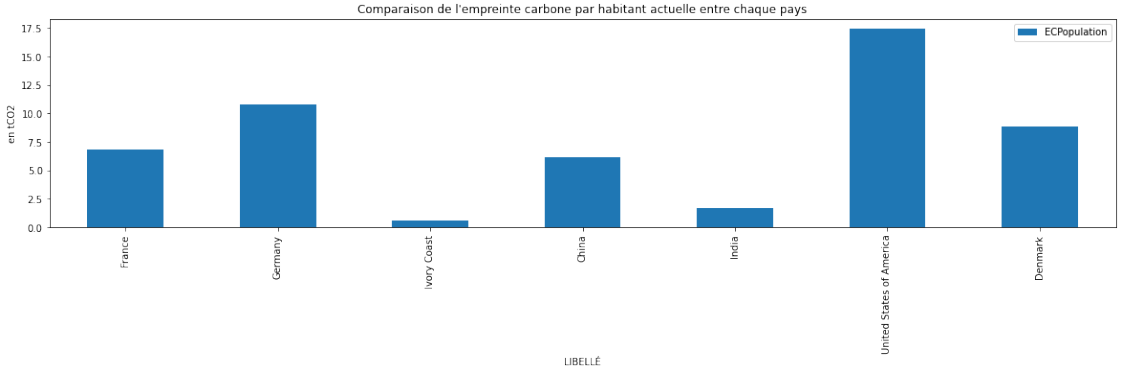
```
new_df = pd.read_sql(query, conn)
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(str)
for i in range(0,len(new_df["EmpreinteCarbone"])):
    new_df["EmpreinteCarbone"].iloc[i] =
new_df["EmpreinteCarbone"].iloc[i].replace(",",".")
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(float)
a = new_df.set_index('LIBELLÉ').plot.bar()
a.set_ylabel("en MtCO2")
a.set_title("Comparaison de l'empreinte carbone actuelle entre chaque
pays")
```

```
Text(0.5, 1.0, "Comparaison de l'empreinte carbone actuelle entre
chaque pays")
```



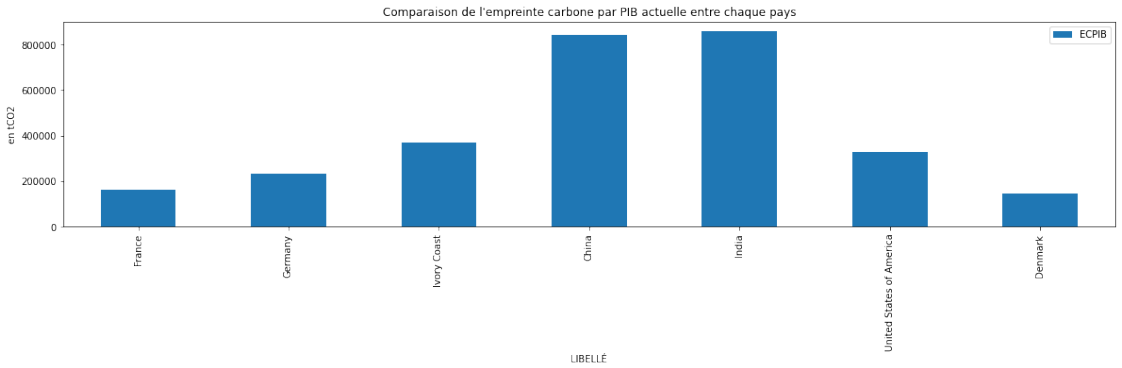

```
#Affiche l'empreinte carbone par habitant pour chaque pays

query = 'select ECPopulation,LIBELLÉ from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID WHERE Date = 2017'
new_df = pd.read_sql(query, conn)
new_df["ECPopulation"] = new_df["ECPopulation"].astype(str)
for i in range(0,len(new_df["ECPopulation"])):
    new_df["ECPopulation"].iloc[i] =
new_df["ECPopulation"].iloc[i].replace(",",".")
new_df["ECPopulation"] = new_df["ECPopulation"].astype(float)
a = new_df.set_index('LIBELLÉ').plot.bar()
a.set_ylabel("en tCO2")
a.set_title("Comparaison de l'empreinte carbone par habitant actuelle
entre chaque pays")
```

```
Text(0.5, 1.0, "Comparaison de l'empreinte carbone par habitant
actuelle entre chaque pays")
```



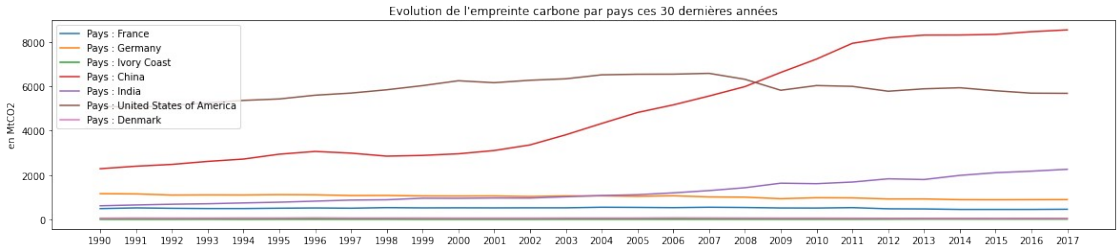

```
#Affiche l'empreinte carbone par PIB pour chaque pays

query = 'select ECPIB,LIBELLÉ from DonnéesPays INNER JOIN Pays on
DonnéesPays.PaysID = Pays.PaysID WHERE Date = 2017'
new_df = pd.read_sql(query, conn)
new_df["ECPIB"] = new_df["ECPIB"].astype(str)
for i in range(0,len(new_df["ECPIB"])):
    new_df["ECPIB"].iloc[i] = new_df["ECPIB"].iloc[i].replace(",",".")
new_df["ECPIB"] = new_df["ECPIB"].astype(float)
a = new_df.set_index('LIBELLÉ').plot.bar()
a.set_ylabel("en tCO2")
a.set_title("Comparaison de l'empreinte carbone par PIB actuelle entre
chaque pays")
```

```
Text(0.5, 1.0, "Comparaison de l'empreinte carbone par PIB actuelle
entre chaque pays")
```

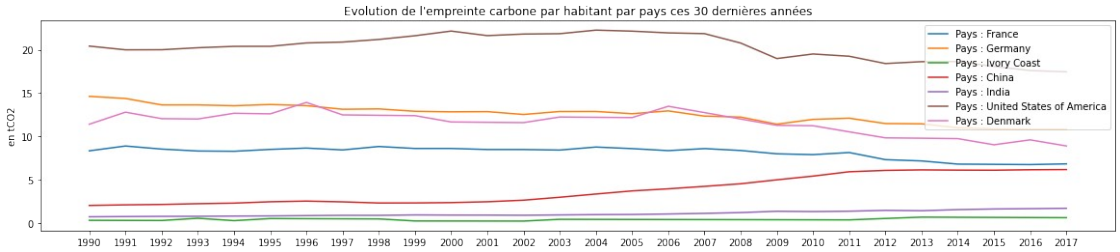



```
#Affiche l'évolution de l'empreinte carbone pour chaque pays sur les
30 dernières années

query = 'select LIBELLÉ FROM Pays'
dff = pd.read_sql(query, conn)

for i in dff["LIBELLÉ"]:
    query = 'select EmpreinteCarbone,Date from DonnéesPays INNER JOIN
Pays on DonnéesPays.PaysID = Pays.PaysID WHERE LIBELLÉ = "' + i + '"'
    new_df = pd.read_sql(query, conn)
```

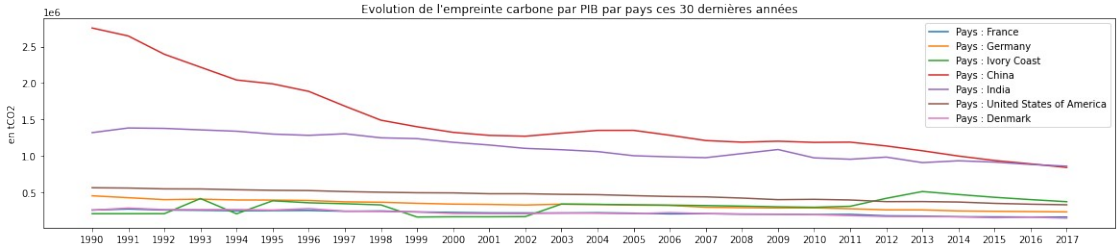
```
    new_df["EmpreinteCarbone"] =
new_df["EmpreinteCarbone"].astype(str)
    for y in range(0,len(new_df["EmpreinteCarbone"])):
        new_df["EmpreinteCarbone"].iloc[y] =
new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
    new_df["EmpreinteCarbone"] =
new_df["EmpreinteCarbone"].astype(float)
    plt.plot(new_df["Date"],new_df["EmpreinteCarbone"],label="Pays : "
+  str(i))

plt.title("Evolution de l'empreinte carbone par pays ces 30 dernières
années")
plt.ylabel("en MtCO2")
plt.legend()
plt.show()
```




```
#Affiche l'évolution de l'empreinte carbone par habitant pour chaque
pays sur les 30 dernières années

import pandas as pd

query = 'select LIBELLÉ FROM Pays'
dff = pd.read_sql(query, conn)

for i in dff["LIBELLÉ"]:
    query = 'select ECPopulation,Date from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID WHERE LIBELLÉ = "' + i + '"'
    new_df = pd.read_sql(query, conn)
    new_df["ECPopulation"] = new_df["ECPopulation"].astype(str)
    for y in range(0,len(new_df["ECPopulation"])):
        new_df["ECPopulation"].iloc[y] =
new_df["ECPopulation"].iloc[y].replace(",",".")
    new_df["ECPopulation"] = new_df["ECPopulation"].astype(float)
    plt.plot(new_df["Date"],new_df["ECPopulation"],label="Pays : " +
str(i))

plt.title("Evolution de l'empreinte carbone par habitant par pays ces
30 dernières années")
plt.ylabel("en tCO2")
plt.legend()
plt.show()
```

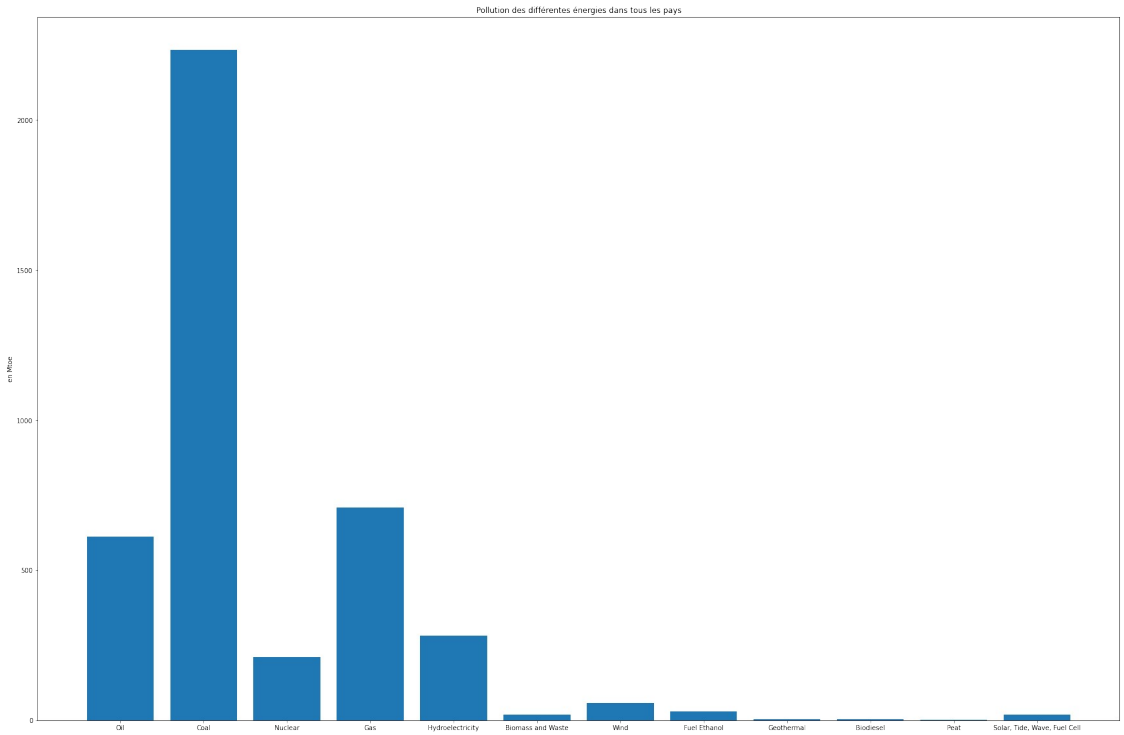



```
#Affiche l'évolution de l'empreinte carbone par PIB pour chaque pays
sur les 30 dernières années

import pandas as pd

query = 'select LIBELLÉ FROM Pays'
dff = pd.read_sql(query, conn)

for i in dff["LIBELLÉ"]:
    query = 'select ECPIB,Date from DonnéesPays INNER JOIN Pays on
DonnéesPays.PaysID = Pays.PaysID WHERE LIBELLÉ = "' + i + '"'
    new_df = pd.read_sql(query, conn)
    new_df["ECPIB"] = new_df["ECPIB"].astype(str)
    for y in range(0,len(new_df["ECPIB"])):
        new_df["ECPIB"].iloc[y] =
new_df["ECPIB"].iloc[y].replace(",",".")
    new_df["ECPIB"] = new_df["ECPIB"].astype(float)
    plt.plot(new_df["Date"],new_df["ECPIB"],label="Pays : " +  str(i))

plt.title("Evolution de l'empreinte carbone par PIB par pays ces 30
dernières années")
plt.ylabel("en tCO2")
plt.legend()
plt.show()
```



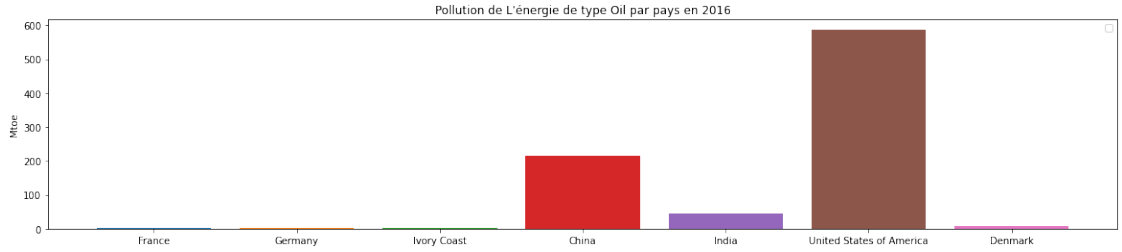

```
#Affiche la pollution par énergie dans tous les pays

query = 'select Pollution,LIBELLÉ,Type,DateDeLaDonnée FROM Utilise
INNER JOIN Pays on Utilise.PaysID = Pays.PaysID INNER JOIN
ProductionEnergie on Utilise.EnergieID = ProductionEnergie.EnergieID'
new_df = pd.read_sql(query, conn)
new_df
new_df["Pollution"] = new_df["Pollution"].astype(str)
for y in range(0,len(new_df["Pollution"])):
```

```
    new_df["Pollution"].iloc[y] =
new_df["Pollution"].iloc[y].replace(",",".")
new_df["Pollution"] = new_df["Pollution"].astype(float)
plt.figure(figsize=(30,20))
plt.bar(new_df["Type"],new_df["Pollution"])
plt.title("Pollution des différentes énergies dans tous les pays")
plt.ylabel("en Mtoe")
plt.show()
```

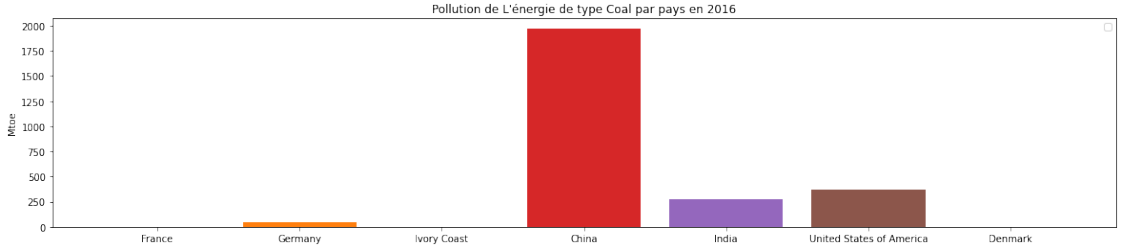



```
query = "SELECT * FROM ProductionEnergie "
new_df = pd.read_sql(query, conn)

#Affiche la pollution des différents types d'énergie pour chaque pays
for i in new_df['Type']:
    graphEnergie(i)
```

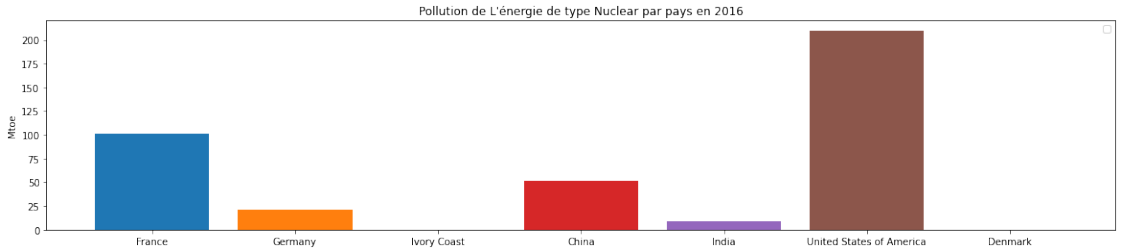
```
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



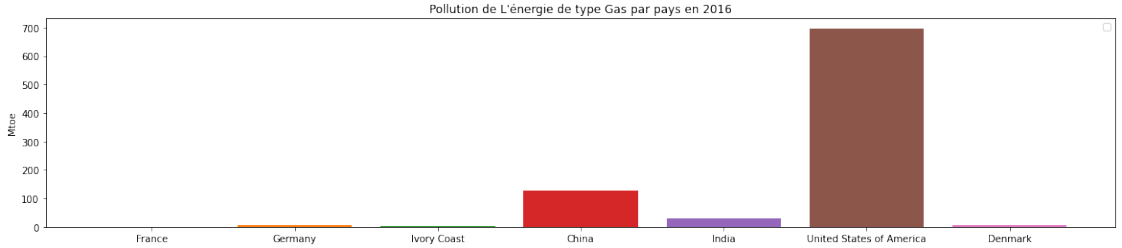
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



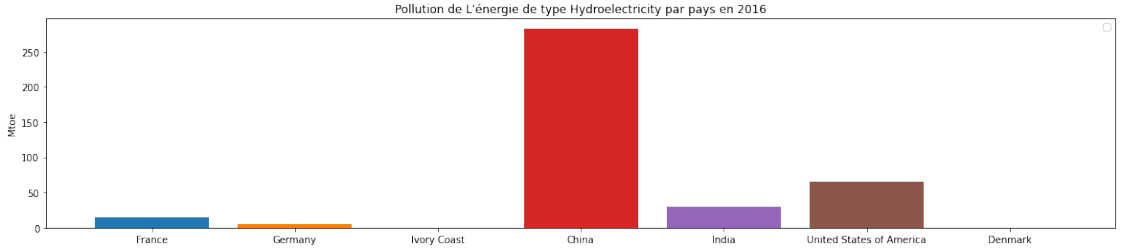
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



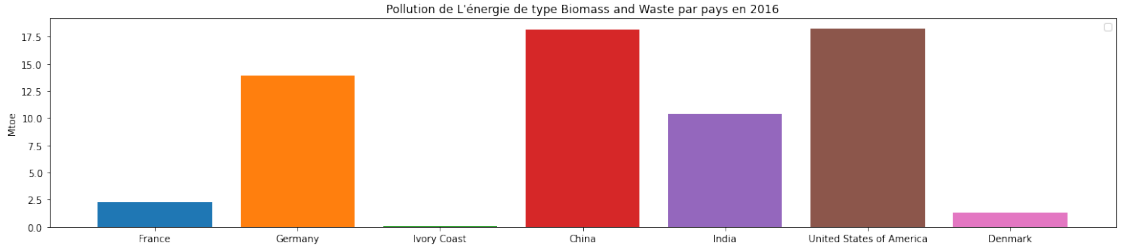
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



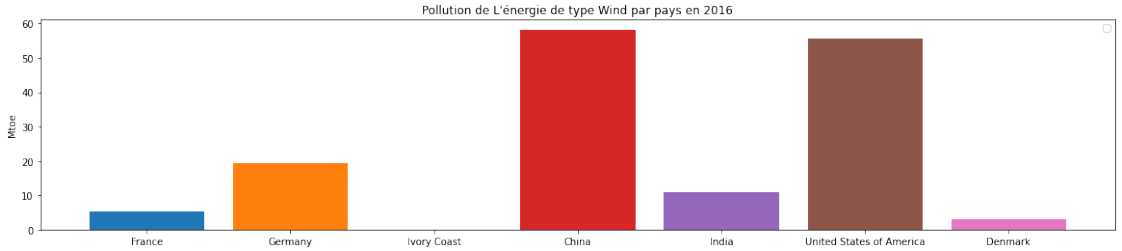
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



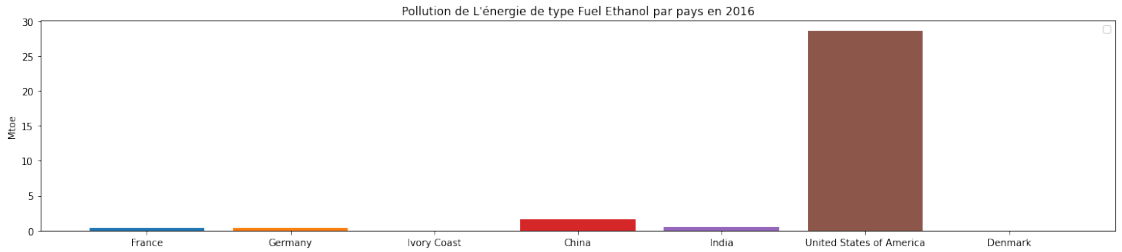
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



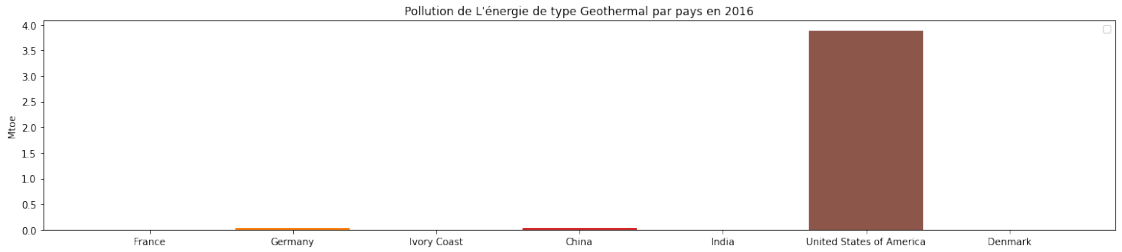
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



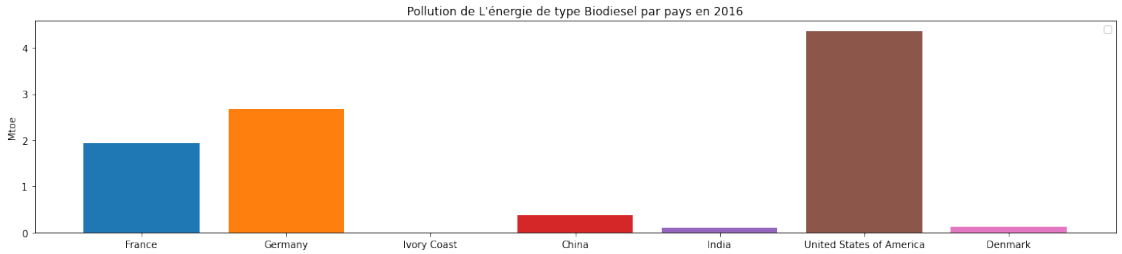
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



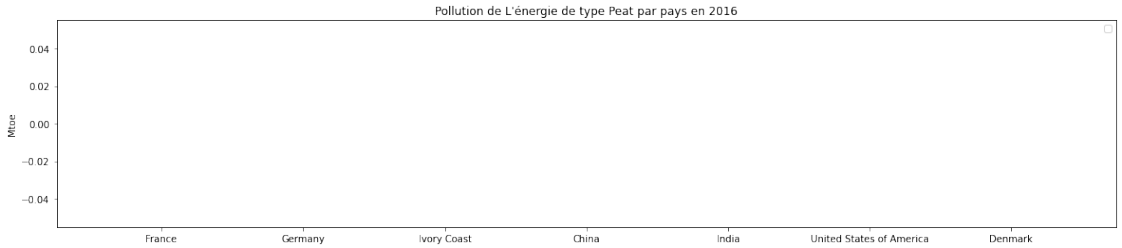
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



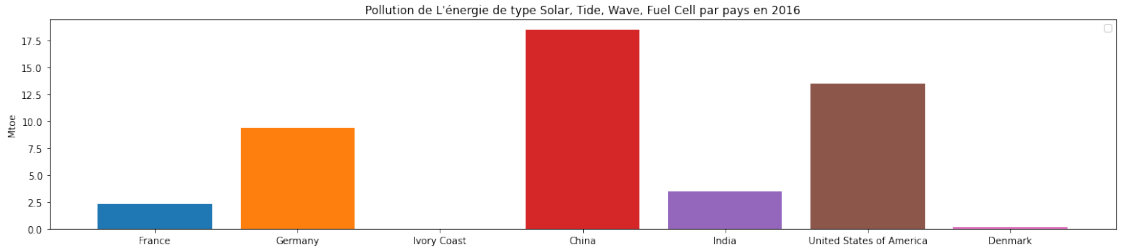
No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.



No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.

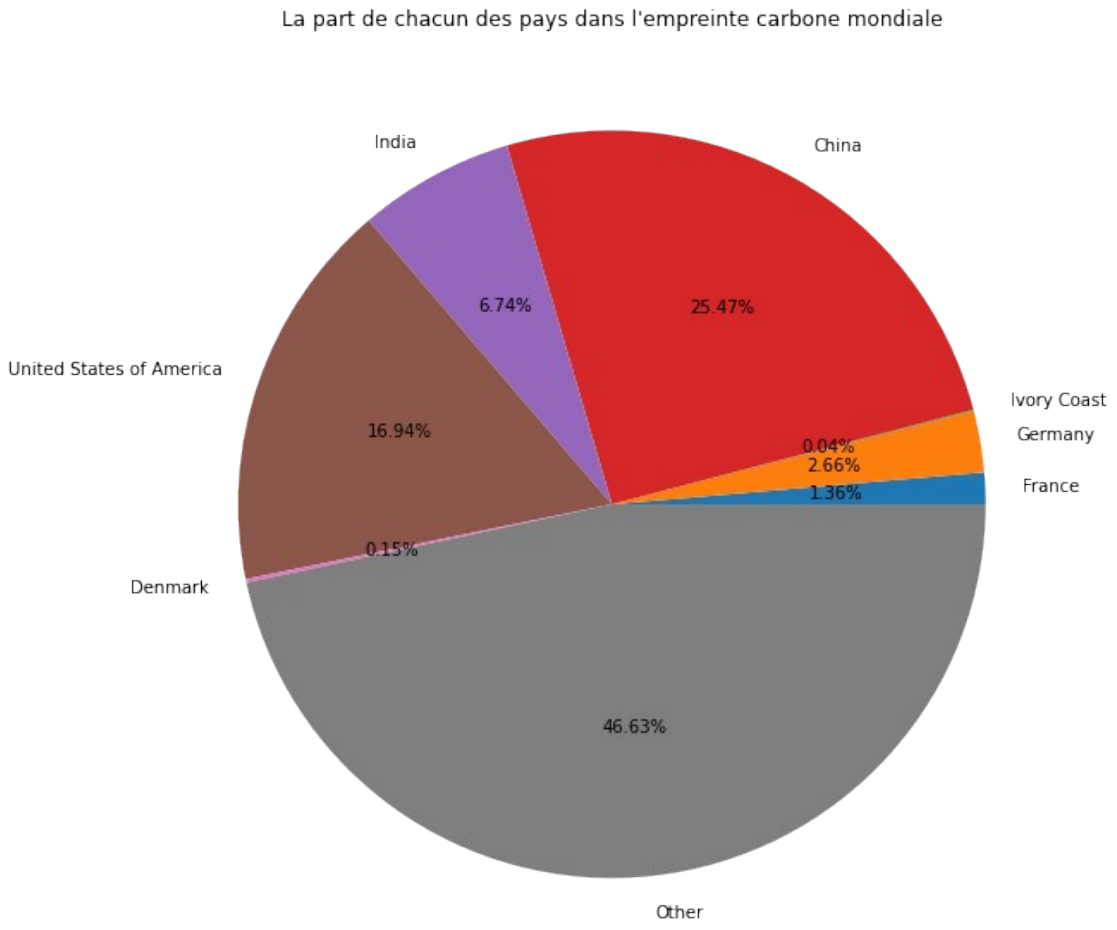


No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.




```
#Affiche la part de chaque pays dans l'empreinte mondiale

new_df = pd.read_csv("./someDataNc2Csv/Carbon Footprint, World, 1990-
2017 (in MtCO2).csv",";")
new_df = new_df.rename(columns={'Unnamed: 0':"Date",'World - Carbon
Footprint':"Pollution"})
new_df["Pollution"] = new_df["Pollution"].astype(str)
for y in range(0,len(new_df["Pollution"])):
    new_df["Pollution"].iloc[y] =
new_df["Pollution"].iloc[y].replace(",",".")
new_df["Pollution"] = new_df["Pollution"].astype(float)

Camembert(new_df)
```

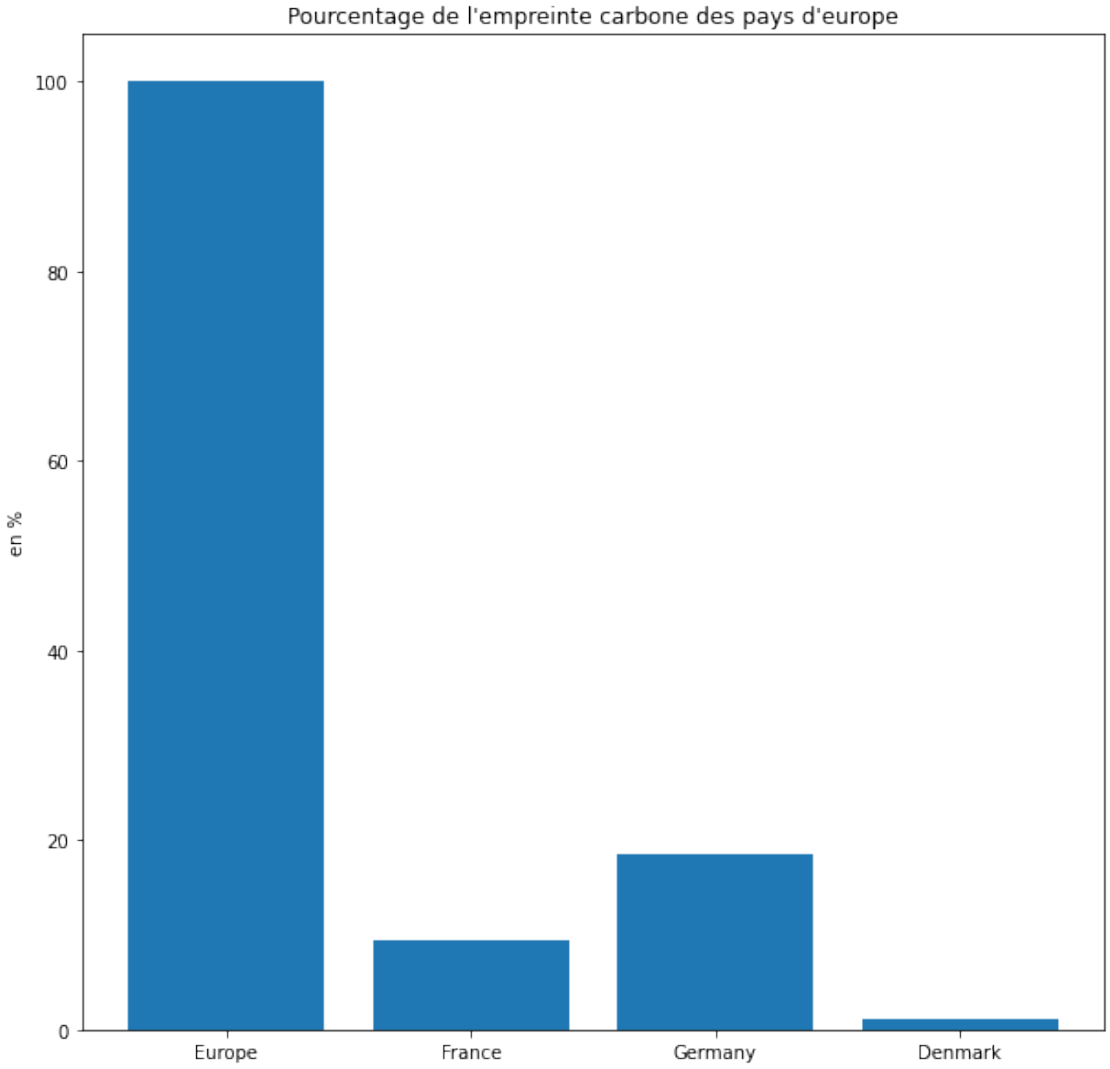
```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```




```
#Afficher le pourcentage de l'empreinte carbone de chaque pays européen

new_df = pd.read_csv("./ToutesDataEurope/Carbon Footprint, Europe, 1990-2017 (in MtCO2).csv",";")
new_df = new_df.rename(columns={'Unnamed: 0':"Date",'Europe - Carbon Footprint':"EmpreinteCarbone"})

#Transforme la donnée
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(str)
for y in range(0,len(new_df["EmpreinteCarbone"])):
    new_df["EmpreinteCarbone"].iloc[y] = 
new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
    new_df['Date'].iloc[y] = new_df['Date'].iloc[y][0:4]
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(float)
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(int)

#Prendre la donnée de l'europe
for y in range(0,len(new_df["EmpreinteCarbone"])):
    if(new_df['Date'].iloc[y] == "2017"):
```

```
        a = int(new_df["EmpreinteCarbone"].iloc[y])

query = '''
    select EmpreinteCarbone,LIBELLÉ from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID INNER JOIN Continent on
Pays.ContinentID = Continent.ContinentID
        WHERE Date = "2017" and NomContinent = "Europe"'''
df = pd.read_sql(query, conn)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(str)
for y in range(0,len(df["EmpreinteCarbone"])):
    df["EmpreinteCarbone"].iloc[y] =
df["EmpreinteCarbone"].iloc[y].replace(",",".")
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(float)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(int)
c = [100]
b = ["Europe"]
for i in range(0,len(df['LIBELLÉ'])):
    c.append(df['EmpreinteCarbone'].iloc[i]/a*100)
    b.append(df['LIBELLÉ'].iloc[i])
plt.rcParams['text.color'] = 'black'
plt.title("Pourcentage de l'empreinte carbone des pays d'europe")
plt.ylabel("en %")
plt.bar(b,c)
plt.show()
```

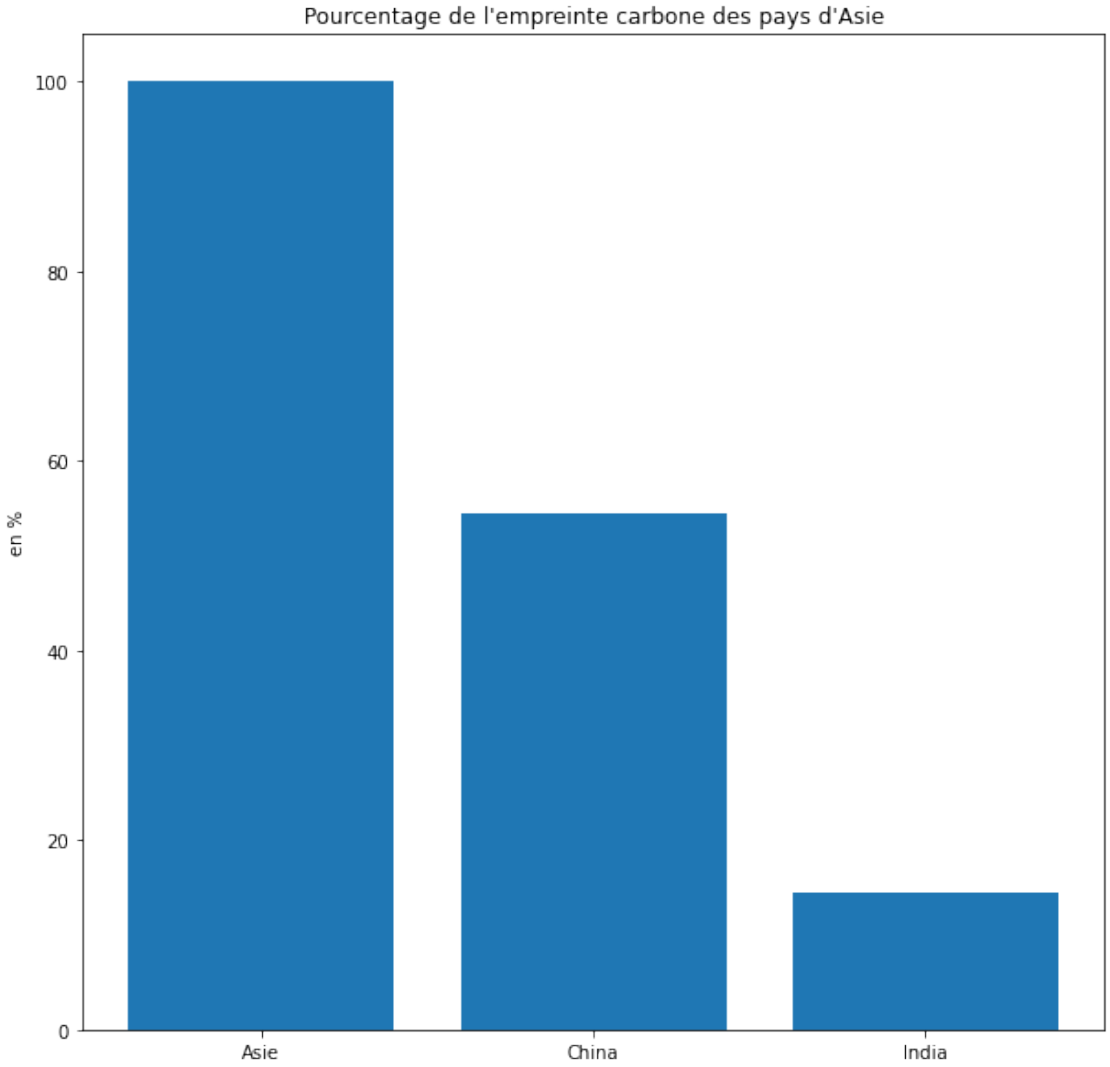
```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```




```
#Afficher le pourcentage de l'empreinte carbone de chaque pays asiatique

new_df = pd.read_csv("./ToutesDataAsieOcéanie/Carbon Footprint, Asia and Oceania, 1990-2017 (in MtCO2).csv",";")
new_df = new_df.rename(columns={'Unnamed: 0':"Date",'Asia and Oceania - Carbon Footprint':"EmpreinteCarbone"})

#Transforme la donnée
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(str)
for y in range(0,len(new_df["EmpreinteCarbone"])):
    new_df["EmpreinteCarbone"].iloc[y] = new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
    new_df['Date'].iloc[y] = new_df['Date'].iloc[y][0:4]
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(float)
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(int)
```

```
#Prendre la donnée de l'europe
for y in range(0,len(new_df["EmpreinteCarbone"])):
    if(new_df['Date'].iloc[y] == "2017"):
        a = int(new_df["EmpreinteCarbone"].iloc[y])

query = '''
    select EmpreinteCarbone,LIBELLÉ from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID INNER JOIN Continent on
Pays.ContinentID = Continent.ContinentID
        WHERE Date = "2017" and NomContinent = "Asia"'''
df = pd.read_sql(query, conn)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(str)
for y in range(0,len(df["EmpreinteCarbone"])):
    df["EmpreinteCarbone"].iloc[y] =
df["EmpreinteCarbone"].iloc[y].replace(",",".")
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(float)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(int)
c = [100]
b = ["Asie"]
for i in range(0,len(df['LIBELLÉ'])):
    c.append(df['EmpreinteCarbone'].iloc[i]/a*100)
    b.append(df['LIBELLÉ'].iloc[i])
plt.rcParams['text.color'] = 'black'
plt.title("Pourcentage de l'empreinte carbone des pays d'Asie")
plt.ylabel("en %")
plt.bar(b,c)
plt.show()
```

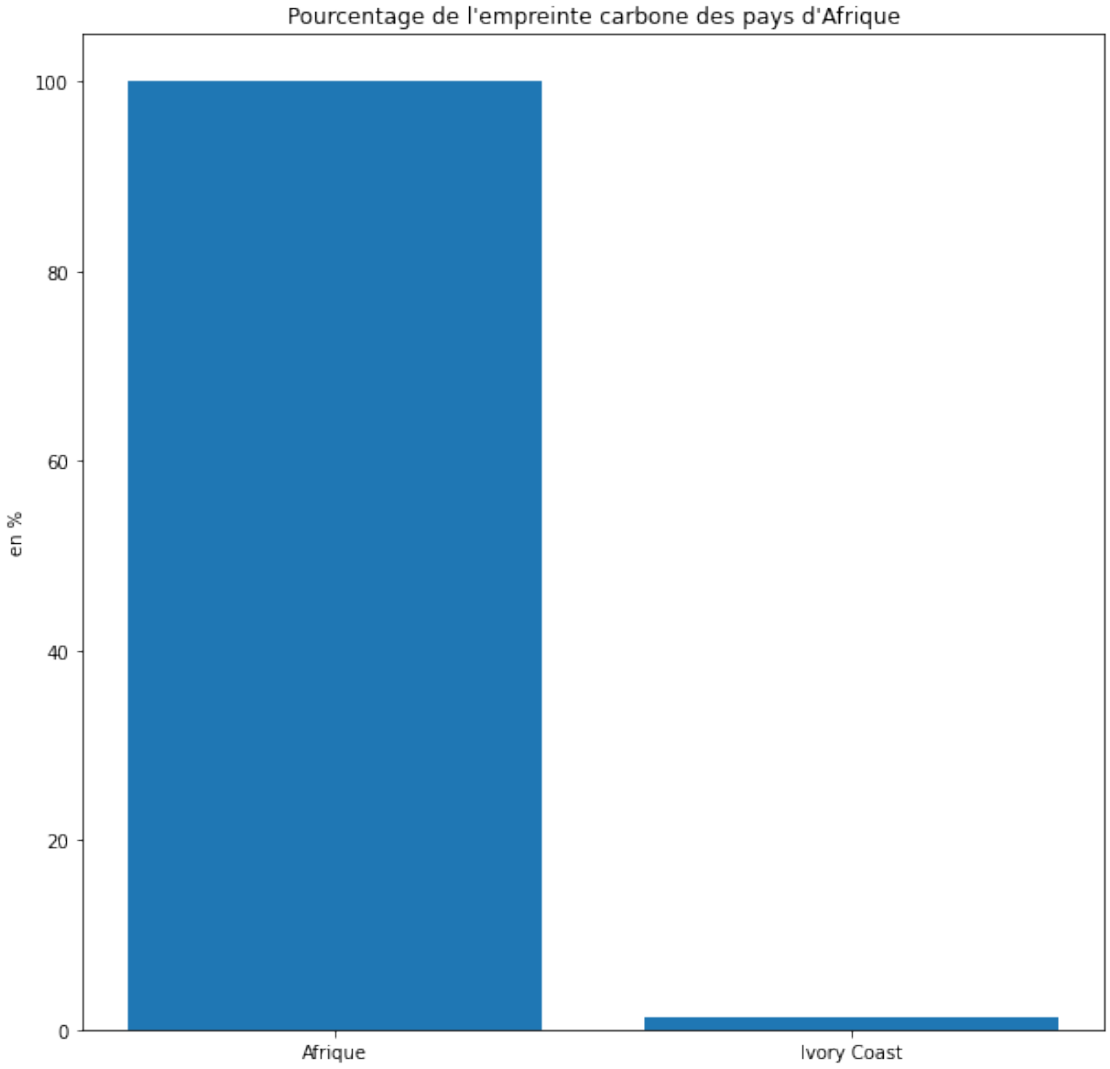
```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```




```
#Afficher le pourcentage de l'empreinte carbone de chaque pays africain

new_df = pd.read_csv("./ToutesDataAfrique/Carbon Footprint, Africa, 1990-2017 (in MtCO2).csv",";")
new_df = new_df.rename(columns={'Unnamed: 0':"Date",'Africa - Carbon Footprint':"EmpreinteCarbone"})

#Transforme la donnée
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(str)
for y in range(0,len(new_df["EmpreinteCarbone"])):
    new_df["EmpreinteCarbone"].iloc[y] = new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
    new_df['Date'].iloc[y] = new_df['Date'].iloc[y][0:4]
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(float)
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(int)
```

```
#Prendre la donnée de l'europe
for y in range(0,len(new_df["EmpreinteCarbone"])):
    if(new_df['Date'].iloc[y] == "2017"):
        a = int(new_df["EmpreinteCarbone"].iloc[y])

query = '''
    select EmpreinteCarbone,LIBELLÉ from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID INNER JOIN Continent on
Pays.ContinentID = Continent.ContinentID
        WHERE Date = "2017" and NomContinent = "Africa"'''
df = pd.read_sql(query, conn)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(str)
for y in range(0,len(df["EmpreinteCarbone"])):
    df["EmpreinteCarbone"].iloc[y] =
df["EmpreinteCarbone"].iloc[y].replace(",",".")
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(float)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(int)
c = [100]
b = ["Afrique"]
for i in range(0,len(df['LIBELLÉ'])):
    c.append(df['EmpreinteCarbone'].iloc[i]/a*100)
    b.append(df['LIBELLÉ'].iloc[i])
plt.rcParams['text.color'] = 'black'
plt.title("Pourcentage de l'empreinte carbone des pays d'Afrique")
plt.ylabel("en %")
plt.bar(b,c)
plt.show()
```

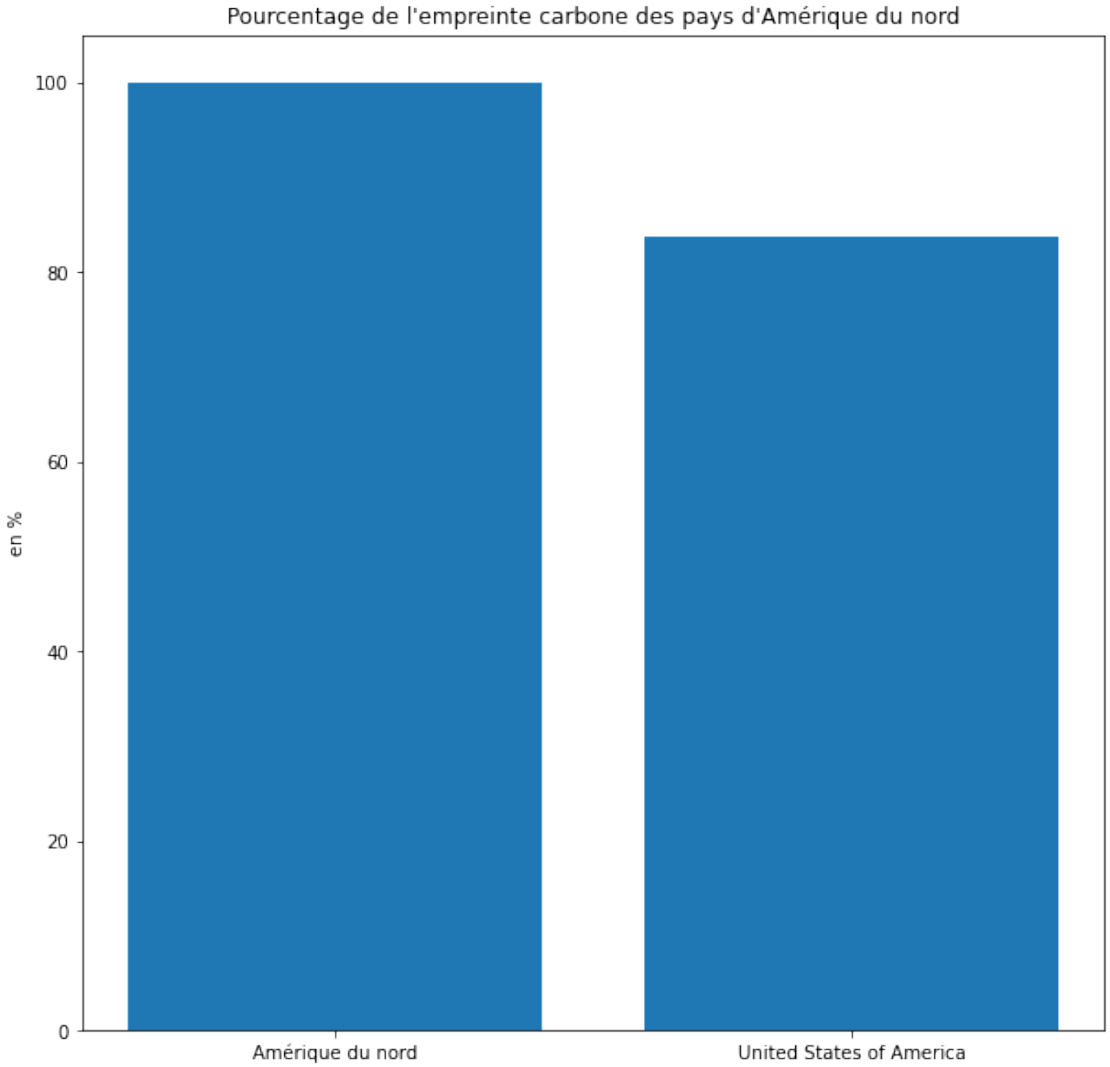
```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```




```
#Afficher le pourcentage de l'empreinte carbone de chaque pays
d'Amérique du Nord

new_df = pd.read_csv("./ToutesDataAmériqueNord/Carbon Footprint, North
America, 1990-2017 (in MtCO2).csv",";")
new_df = new_df.rename(columns={'Unnamed: 0':"Date",'North America -
Carbon Footprint':"EmpreinteCarbone"})

#Transforme la donnée
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(str)
for y in range(0,len(new_df["EmpreinteCarbone"])):
    new_df["EmpreinteCarbone"].iloc[y] =
new_df["EmpreinteCarbone"].iloc[y].replace(",",".")
    new_df['Date'].iloc[y] = new_df['Date'].iloc[y][0:4]
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(float)
new_df["EmpreinteCarbone"] = new_df["EmpreinteCarbone"].astype(int)
```

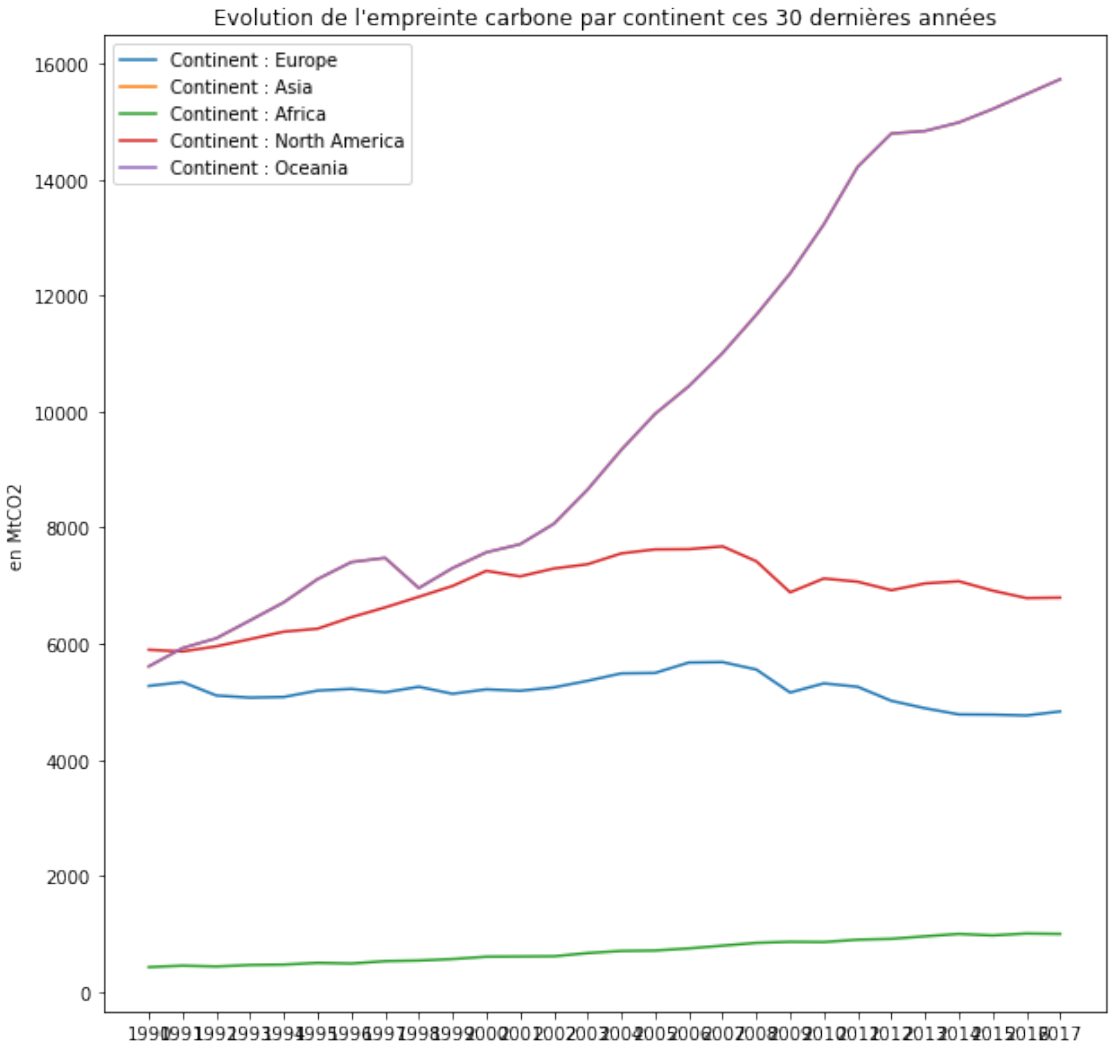
```
#Prendre la donnée de l'europe
for y in range(0,len(new_df["EmpreinteCarbone"])):
    if(new_df['Date'].iloc[y] == "2017"):
        a = int(new_df["EmpreinteCarbone"].iloc[y])

query = '''
    select EmpreinteCarbone,LIBELLÉ from DonnéesPays INNER JOIN Pays
on DonnéesPays.PaysID = Pays.PaysID INNER JOIN Continent on
Pays.ContinentID = Continent.ContinentID
        WHERE Date = "2017" and NomContinent = "North America"'''
df = pd.read_sql(query, conn)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(str)
for y in range(0,len(df["EmpreinteCarbone"])):
    df["EmpreinteCarbone"].iloc[y] =
df["EmpreinteCarbone"].iloc[y].replace(",",".")
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(float)
df["EmpreinteCarbone"] = df["EmpreinteCarbone"].astype(int)
c = [100]
b = ["Amérique du nord"]
for i in range(0,len(df['LIBELLÉ'])):
    c.append(df['EmpreinteCarbone'].iloc[i]/a*100)
    b.append(df['LIBELLÉ'].iloc[i])
plt.rcParams['text.color'] = 'black'
plt.title("Pourcentage de l'empreinte carbone des pays d'Amérique du
nord")
plt.ylabel("en %")
plt.bar(b,c)
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```



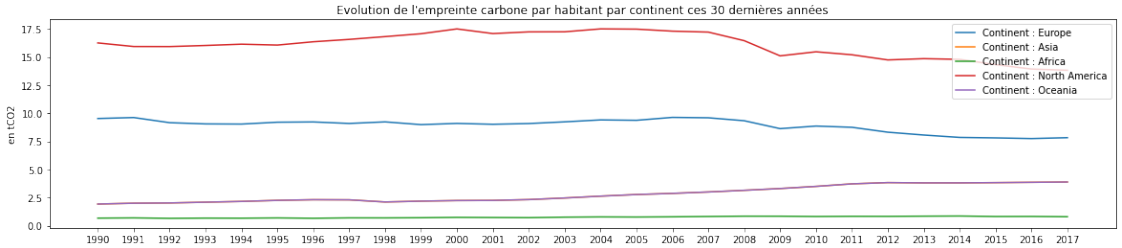

```
#Affiche l'évolution de l'empreinte carbone pour chaque continent sur
les 30 dernières années (les courbes de l'Asie et de l'Océanie
#sont confondus car le csv prend l'empreinte carbone des deux en même
temps)

query = 'select NomContinent FROM Continent'
dff = pd.read_sql(query, conn)

for i in dff["NomContinent"]:
    query = 'select EmpreinteCarboneC,Date from DonnéesContinent INNER
JOIN Continent on DonnéesContinent.ContinentID = Continent.ContinentID
WHERE NomContinent = "' + i + '"'
    new_df = pd.read_sql(query, conn)
    new_df["EmpreinteCarboneC"] =
new_df["EmpreinteCarboneC"].astype(str)
    for y in range(0,len(new_df["EmpreinteCarboneC"])):
        new_df["EmpreinteCarboneC"].iloc[y] =
new_df["EmpreinteCarboneC"].iloc[y].replace(",",".")
```

```
    new_df["EmpreinteCarboneC"] =
new_df["EmpreinteCarboneC"].astype(float)

plt.plot(new_df["Date"],new_df["EmpreinteCarboneC"],label="Continent :
" +  str(i))

plt.rcParams["figure.figsize"] = (20,4)
plt.title("Evolution de l'empreinte carbone par continent ces 30
dernières années")
plt.ylabel("en MtCO2")
plt.legend()
plt.show()
```



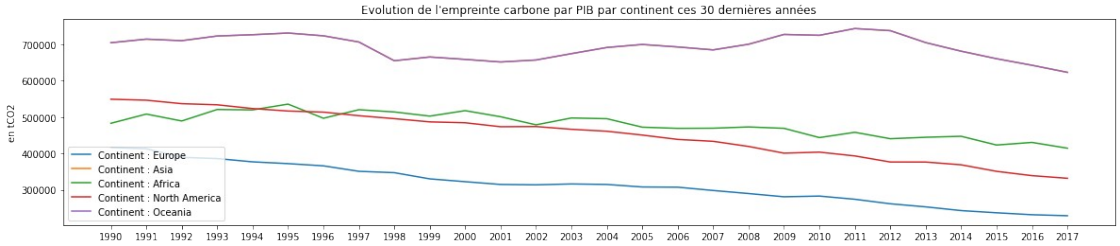

```
#Affiche l'évolution de l'empreinte carbone par habitant pour chaque
continent sur les 30 dernières années (les courbes de l'Asie et de
l'Océanie
#sont confondus car le csv prend l'empreinte carbone des deux en même
temps)
```

```
import pandas as pd

query = 'select NomContinent FROM Continent'
dff = pd.read_sql(query, conn)

for i in dff["NomContinent"]:
    query = 'select ECPopulationC,Date from DonnéesContinent INNER
JOIN Continent on DonnéesContinent.ContinentID = Continent.ContinentID
WHERE NomContinent = "' + i + '"'
    new_df = pd.read_sql(query, conn)
    new_df["ECPopulationC"] = new_df["ECPopulationC"].astype(str)
    for y in range(0,len(new_df["ECPopulationC"])):
        new_df["ECPopulationC"].iloc[y] =
new_df["ECPopulationC"].iloc[y].replace(",",".")
    new_df["ECPopulationC"] = new_df["ECPopulationC"].astype(float)
    plt.plot(new_df["Date"],new_df["ECPopulationC"],label="Continent :
" +  str(i))

plt.title("Evolution de l'empreinte carbone par habitant par continent
ces 30 dernières années")
plt.ylabel("en tCO2")
plt.legend()
plt.show()
```




```
#Affiche l'évolution de l'empreinte carbone par PIB pour chaque
continent sur les 30 dernières années (les courbes de l'Asie et de
l'Océanie
#sont confondus car le csv prend l'empreinte carbone des deux en même
temps)

import pandas as pd

query = 'select NomContinent FROM Continent'
dff = pd.read_sql(query, conn)

for i in dff["NomContinent"]:
    query = 'select ECPIBC,Date from DonnéesContinent INNER JOIN
Continent on DonnéesContinent.ContinentID = Continent.ContinentID
WHERE NomContinent = "' + i + '"'
    new_df = pd.read_sql(query, conn)
    new_df["ECPIBC"] = new_df["ECPIBC"].astype(str)
```

```
    for y in range(0,len(new_df["ECPIBC"])):
        new_df["ECPIBC"].iloc[y] =
new_df["ECPIBC"].iloc[y].replace(",",".")
    new_df["ECPIBC"] = new_df["ECPIBC"].astype(float)
    plt.plot(new_df["Date"],new_df["ECPIBC"],label="Continent : " +
str(i))

plt.title("Evolution de l'empreinte carbone par PIB par continent ces
30 dernières années")
plt.ylabel("en tCO2")
plt.legend()
plt.show()
```

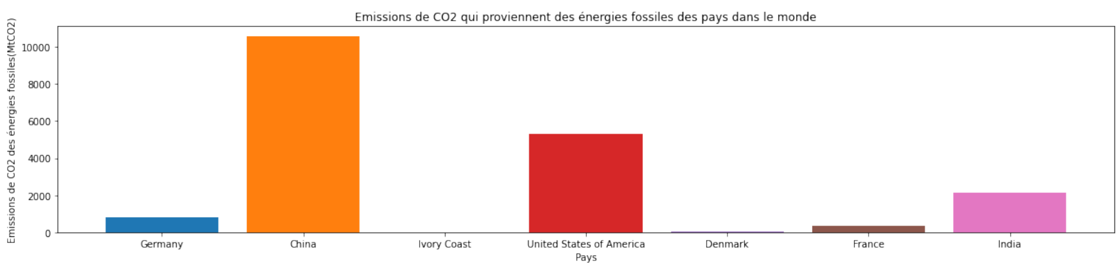



```
#take data csv C02 emission from fussil fuel

df = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels,
Germany, 2016 (in MtCO2).csv",";")
df = df.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2
emossions"})
toInt(df["CO2 emossions"])
df["CO2 emossions"] = df["CO2 emossions"].astype(float)
df["CO2 emossions"] = df["CO2 emossions"].astype(int)
plt.bar(df['Pays'],df["CO2 emossions"])

df2 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels,
China, 2016 (in MtCO2).csv",";")
df2 = df2.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2
emossions"})
toInt(df2["CO2 emossions"])
df2["CO2 emossions"] = df2["CO2 emossions"].astype(float)
df2["CO2 emossions"] = df2["CO2 emossions"].astype(int)
plt.bar(df2['Pays'],df2["CO2 emossions"])

df3 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels,
Ivory Coast, 2016 (in MtCO2).csv",";")
df3 = df3.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2
emossions"})
toInt(df3["CO2 emossions"])
df3["CO2 emossions"] = df3["CO2 emossions"].astype(float)
df3["CO2 emossions"] = df3["CO2 emossions"].astype(int)
plt.bar(df3['Pays'],df3["CO2 emossions"])
```

```
df4 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels, United States of America, 2016 (in MtCO2).csv",";")
df4 = df4.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2 emossions"})
toInt(df4["CO2 emossions"])
df4["CO2 emossions"] = df4["CO2 emossions"].astype(float)
df4["CO2 emossions"] = df4["CO2 emossions"].astype(int)
plt.bar(df4['Pays'],df4["CO2 emossions"])

df5 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels, Denmark, 2016 (in MtCO2).csv",";")
df5 = df5.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2 emossions"})
toInt(df5["CO2 emossions"])
df5["CO2 emossions"] = df5["CO2 emossions"].astype(float)
df5["CO2 emossions"] = df5["CO2 emossions"].astype(int)
plt.bar(df5['Pays'],df5["CO2 emossions"])

df6 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels, France, 2016 (in MtCO2).csv",";")
df6 = df6.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2 emossions"})
toInt(df6["CO2 emossions"])
df6["CO2 emossions"] = df6["CO2 emossions"].astype(float)
df6["CO2 emossions"] = df6["CO2 emossions"].astype(int)
plt.bar(df6['Pays'],df6["CO2 emossions"])

df7 = pd.read_csv("./DataCO2Fossile/CO2 emissions from fossil fuels, India, 2016 (in MtCO2).csv",";")
df7 = df7.rename(columns={"Unnamed: 0":"Pays","Germany":"CO2 emossions"})
toInt(df7["CO2 emossions"])
df7["CO2 emossions"] = df7["CO2 emossions"].astype(float)
df7["CO2 emossions"] = df7["CO2 emossions"].astype(int)
plt.bar(df7['Pays'],df7["CO2 emossions"])

plt.title("Emissions de CO2 qui proviennent des énergies fossiles des pays dans le monde")
plt.xlabel("Pays")
plt.ylabel("Emissions de CO2 des énergies fossiles(MtCO2)")
plt.show()
```

```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331: FutureWarning: In a future version of pandas all arguments of read_csv except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

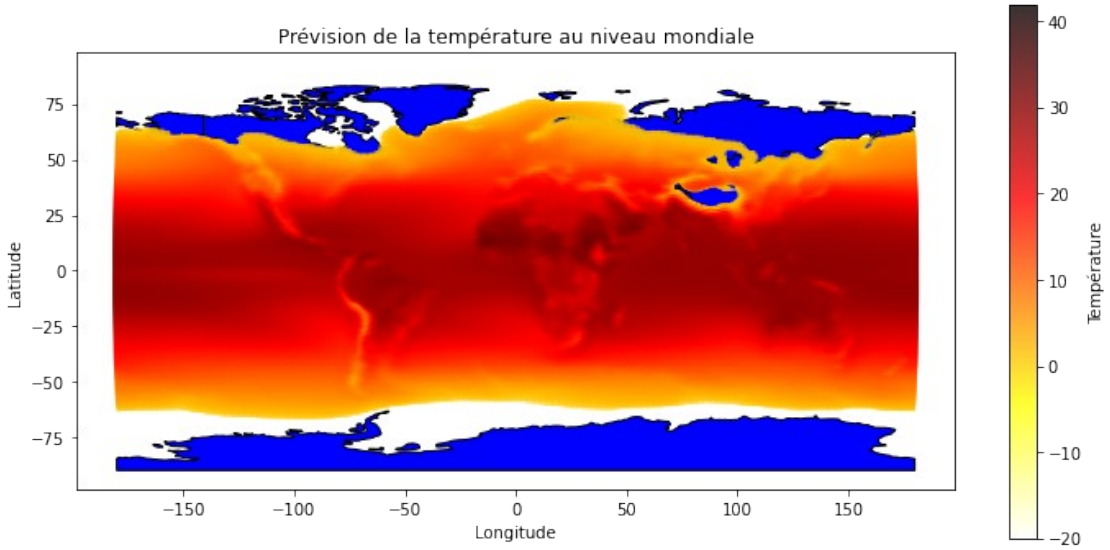
```
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
/usr/lib/python3/dist-packages/IPython/core/interactiveshell.py:3331:
FutureWarning: In a future version of pandas all arguments of read_csv
except for the argument 'filepath_or_buffer' will be keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```




```python
#Importation des bibliothèques

import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd

df = pd.read_csv("./someDataNc2Csv/CMIP6MeanTemperature.csv")
df = df.drop(columns='crs')
df = df.rename(columns={"lon":
            'Longitude',
             "lat" :
             'Latitude',
             "tas":
             'Température'})

regleLongEtLat(df)

#Depuis GeoPandas, nos données cartographiques du monde
worldmap = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))

#Créer des axes et tracer une carte du monde
```

```
fig, ax = plt.subplots(figsize=(12, 6))
worldmap.plot(color="blue",edgecolor='black', ax=ax)

#Tracer nos données avec une carte en couleur
x = df['Longitude']
y = df['Latitude']
z = df['Température']
plt.scatter(x, y, s=z, c=z, alpha=0.8, vmin=-20, vmax=42,
            cmap='hot_r')
plt.colorbar(label='Température')
plt.title("Prévision de la température au niveau mondiale")

plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

```
/home/etd/.local/lib/python3.8/site-packages/matplotlib/
collections.py:981: RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```




```
#Importation des bibliothèques

import matplotlib.pyplot as plt
import pandas as pd
import geopandas as gpd

df = pd.read_csv("./someDataNc2Csv/CMIP6 - Sea level rise (SLR) Change
meters - Long Term (2081-2100) SSP5-8.5 (rel. to 1995-2014) -
Annual .csv")
df = df.drop(columns='crs')
df = df.rename(columns={"lon":
            'Longitude',
             "lat" :
```

```
                'Latitude',
                "total":
                'Montée des eaux'})

regleLongEtLat(df)

#Depuis GeoPandas, nos données cartographiques du monde
worldmap = gpd.read_file(gpd.datasets.get_path("naturalearth_lowres"))

#Créer des axes et tracer une carte du monde
fig, ax = plt.subplots(figsize=(12, 6))
worldmap.plot(color="white",edgecolor='black', ax=ax)

#Tracer nos données avec une carte en couleur
x = df['Longitude']
y = df['Latitude']
z = df['Montée des eaux']
plt.scatter(x, y, s=z, c=z, alpha=1, vmin=-0.2, vmax=1.5,
            cmap='winter_r')
plt.colorbar(label='Montée des eaux')
plt.title("Prévision de l'élévation du niveau des eaux dans le monde
pour 2100")

plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```

```
/home/etd/.local/lib/python3.8/site-packages/matplotlib/
collections.py:981: RuntimeWarning: invalid value encountered in sqrt
  scale = np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```