# TD - Modèle de prédiction de la température annuelle moyenne en France

## ELEMENTS DE CORRECTION :

### Q1 à Q5 :

```
# -*- coding: utf-8 -*-
import os
os.chdir('CHEMIN')
import numpy as np
import matplotlib.pyplot as plt

# Extraction des données et création des 2 jeux de données
dataset=np.loadtxt('Temp_annuelle_France.csv', skiprows=1,delimiter=';')
Annee=np.array([dataset[i][0] for i in range(len(dataset))])
Annee = Annee.reshape((len(Annee)),1)
Temp=np.array([dataset[i][1] for i in range(len(dataset))])
Temp = Temp.reshape((len(Temp)),1)

# TRACE Dataset
plt.figure(1)
plt.plot(Annee,Temp,'bo-')
plt.xlabel('Année')
plt.ylabel('Température [°C]')

# Calcul des moyennes et écrat types
mu_Annee = np.mean(Annee)
sigma_Annee = np.std(Annee)
mu_Temp = np.mean(Temp)
sigma_Temp = np.std(Temp)

# Centrage des données
Annee = (Annee - mu_Annee)/sigma_Annee
Temp = (Temp - mu_Temp)/sigma_Temp

def h(X,W):
    return X.dot(W)

def J(X,Y,W):
    N=np.shape(X)[0]
    cout = sum((h(X,W)-Y)**2)
    return (1/(2*N))*cout

def Gradient(X,Y,W):
    N = np.shape(X)[0]
    return 1/N * X.T.dot(h(X,W)-Y)

def Degre_multiple(X,d):
    N = np.shape(X)[0]
    U = np.ones((N,1))
    if d==0:
        R=U
    else:
        R=np.hstack((U,X)) # np.hstack Permet d'empiler des séquences horizontalement
        k=2
        while k<=d:
            R=np.hstack((R,X**k))
            k+=1
    return R

def Regression_Lineaire(X,Y,degre,nbre_iter=1000,alpha=0.01):
    W = np.random.randn(degre+1,1)
    X = Degre_multiple(X,degre)
    Histoire_cout = [J(X,Y,W)]
    k=0
    while k<nbre_iter :
        W = W - alpha*Gradient(X,Y,W)
        Histoire_cout.append(J(X,Y,W))
        k+=1
    return W,Histoire_cout

L_W = []
L_cout = []
```

```
for k in range(0,7):
    W,ch = Regression_Lineaire(Annee,Temp,k,nbre_iter=5000,alpha=0.02)
    L_W.append(W)
    L_cout.append(ch)

plt.figure(2)
for k in range(len(L_cout)):
    labell = 'Degré = {}'.format(k)
    plt.plot(L_cout[k],label=str(labell))
plt.xlabel("Nombre itérations")
plt.ylabel("Coût")
plt.legend()

plt.figure(3)
for k in range(len(L_cout)):
    labell = 'Degré = {}'.format(k)
    plt.plot(L_cout[k],label=str(labell))
plt.xlabel("Nombre itérations")
plt.ylabel("Coût")
plt.xlim(0,50)
plt.legend()

def Prediction(X,W):
    degre = len(W)-1
    X = Degre_multiple(X,degre)
    return h(X,W)

plt.figure(1)
for k in range(len(L_W)):
    ym = Prediction(Annee,L_W[k])
    X = Annee*sigma_Annee + mu_Annee
    ym = ym*sigma_Temp + mu_Temp
    labell = 'Degré = {}'.format(k)
    plt.plot(X,ym,label=str(labell))
plt.legend(loc="upper left")

# Un degré 4 semble être la solution offrant le modèle le plus proche pour prédire.
# prédiction températures en 2030, 2050 et 2100...

Xp = np.array([[2025,2030,2035,2040,2045,2050]]).T
Xpn = (Xp - mu_Annee)/sigma_Annee #On normalise pour coller au modèle
yp = Prediction(Xpn,L_W[3]) # On prend le modèle de degré 4
yp = yp*sigma_Temp + mu_Temp

plt.figure(1)
plt.plot(Xp,yp,'mx-')
```

**Q6 :**

```
# -*- coding: utf-8 -*-
import os
os.chdir('CHEMIN')
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
import numpy as np
import matplotlib.pyplot as plt

# Etape 1 : Recuperation du dataset
dataset=np.loadtxt('Temp_annuelle_France.csv', skiprows=1,delimiter=';')
Annee=np.array([dataset[i][0] for i in range(len(dataset))])
X = Annee.reshape((len(Annee)),1)
Temp=np.array([dataset[i][1] for i in range(len(dataset))])
Y = Temp.reshape((len(Temp)),1)

plt.figure(1)
plt.plot(X,Y,'bo-')
plt.xlabel('Année')
plt.ylabel('Température [°C]')

# Etape 2 : Preparation des donnees
degre = 4
transformation_polynomiale = PolynomialFeatures(degree = degre)
X_t = transformation_polynomiale.fit_transform(X)

# Etape 3: Definition et entraînement du modèle
Modele_RP = LinearRegression()
Modele_RP.fit(X_t, Y)

# Etape 4 : Prediction
x_new_min = 1900
x_new_max = 2050
Xp = np.linspace(x_new_min, x_new_max, 2050-1900)
Xp = Xp[:,np.newaxis]
Xp_t = transformation_polynomiale.fit_transform(Xp)
Yp = Modele_RP.predict(Xp_t)

# Etape 5 : Graphiques
plt.figure(1)
plt.plot(Xp, Yp,label=str(labell), color='coral', linewidth=3)
plt.grid()
labell = 'Degré = {}'.format(degre)
plt.xlim(x_new_min,x_new_max)
title = 'Degré = {}'.format(degre)
plt.title("Regression Lineaire multiple (polynomiale) \n " + title,
          fontsize=10)
plt.xlabel('Année')
plt.ylabel('Température [°C]')
plt.legend(loc="upper left")
plt.show()

# Etape 6 : Calculs des erreurs
Ym = Modele_RP.predict(X_t)
rmse = np.sqrt(mean_squared_error(Y,Ym))
print('RMSE: ', rmse)
```