```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.preprocessing import StandardScaler

#Q1-a Importer les données
X= load_iris().data
#Q1-b : Importer les labels des classes
y= load_iris().target
#Q2: Afficher les noms des variables
print("1.Les variables : ", load_iris().feature_names) # 4 variables
#Q3: Afficher les classes
print("2.Les classes : ",load_iris().target_names) #3 classes  ==> classification multiclasses
#Q4: Normaliation des données
scaler = StandardScaler().fit(X)
rescaledX = scaler.transform(X)
#Q5: Fonction KNN
def K_NN(X_train, X_test, y_train, y_test ,k):
   knn = KNeighborsClassifier(n_neighbors=k)
   Res_KNN=knn.fit(X_train, y_train)
   #Prédiction sur les données de test
   knn.predict(X_test)
   taux_erreur_test=(1-Res_KNN.score(X_test,y_test))
   print('3.Taux d\'erreur sur l\'ensemble de test',taux_erreur_test)
#Q6: Construire les ensembles d'apprentissage(60%) et de test(40%)
X_train, X_test, y_train, y_test = train_test_split(rescaledX, y, test_size=0.4)
#Q7: appliquer KNN avec k=5 par exemple
K_NN(X_train, X_test, y_train, y_test ,5)
#Q8: établir une validation croisée pour déterminer la meilleure valeur de k
param=[{"n_neighbors":list(range(1,15))}]
knn= GridSearchCV(KNeighborsClassifier(),param,cv=5)
Res_KNN=knn.fit(X_train, y_train)
best_k=Res_KNN.best_params_["n_neighbors"]
print("4.Meilleure valeur de k : ",best_k)
#---Classification par KNN en utilisant la meilleure valeur de k
K_NN(X_train, X_test, y_train, y_test ,best_k)
#Q9 :afficher la matrice de confusion
y_pred= knn.predict(X_test)
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import style
cm = confusion_matrix(y_test, y_pred)
sns.color_palette("Paired")
# If True, write the data value in each cell
sns.heatmap(cm,annot=True,cmap=sns.color_palette("vlag", as_cmap=True))
print(cm)

from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
print(accuracy_score(y_test,y_pred))
```