# Devoir Maison n°2

**Décembre 2022**

---

## Épreuve de l'enseignement de spécialité

# NUMÉRIQUE et SCIENCES INFORMATIQUES

## Classe Terminale de la voie générale

---

**Le sujet comporte 6 pages numérotées de 1/6 à 6/6.**

*Il est rappelé que la qualité de la rédaction, la clarté et la précision des raisonnements entreront pour une part importante dans l'appréciation des copies.*

*Pour vous aider dans ce DM, vous pourrez utiliser le logiciel **xampp** ou **Uwamp** sous windows, **mamp** sous macOS ou **Lamp** sous Linux.*

*Vous rendrez ce DM sous format numérique: un fichier python pour les fonctions, un fichier sql pour la base de données et un fichier de traitement de texte (**Word** ou **LibreOffice**) pour répondre aux questions qui ne nécessitent pas le fichier python ou le fichier sql.*

*Informations sur ce DM de noël:*

- *Si vous rendez le sujet pendant les vacances de Noël: +1 point.*
- *La partie 4 est optionnelle et peut rapporter jusqu'à 6 points supplémentaires.*
- *Si votre note est supérieure à 20/20, le tiers des points en surplus augmenteront la note de votre prochain DS*

# Problème

On veut créer une base de données de gestion de stocks.
Un article possède un code d'article unique et un nom.
Un rangement possède un nom.

## Partie 1 : gestion de la base de données

**Question 1.1** Créer une base de données relationnelle contenant les tables suivantes :

*table* `articles` : `code`, `nom`
où `id` est un entier, `nom` et `code` des chaînes de caractères.
*table* `rangements` : `id`, `nom`
où `id` est un entier, `nom` une chaîne de caractères.
*table* `stocks` : `id`, `#article`, `#rangement`, `quantite`
où `id` et `quantite` sont des entiers.

**Question 1.2** Quels sont les domaines des attributs `article` et `rangement` de la table `stocks` ?

**Question 1.3** Donner les clés primaires de ces trois tables.

**Question 1.4** Donner les clés étrangères de la table `stocks`.

## Partie 2 : requêtes SQL

**Question 2.1** Écrire la requête nécessaire à l'obtention de tous les noms de rangements.

**Question 2.2** Écrire la requête nécessaire à l'obtention de la liste des noms des articles et de leur code, ordonnée par ordre alphabétique des noms.

**Question 2.3** Écrire la requête nécessaire à l'insertion d'un article dans la table `articles`.

**Question 2.4** Écrire la requête nécessaire à la modification de la quantité d'un article dans la table `stocks`.

**Question 3** Écrire la requête nécessaire à l'obtention de la liste des noms des articles, de leurs codes, de l'endroit où ils sont rangés et de leurs quantités dans les stocks, ordonnés par nom de rangement déscendant.

## Partie 3 : programmation

**Question 4** On donne en annexe le code Python des classes `Article`, `Rangements` et `Stocks`.

**Question 4.1** Créer une fonction Python `affiche_stocks()` qui affiche l'ensemble des noms des articles, des codes correspondants, de l'endroit où il sont rangés et des quantités, rangés par ordre alphabétique du nom de l'article.

**Question 4.2** Créer une fonction `est_present_dans_articles(code, nom)` qui prend en paramètres deux chaînes de caractères `code` et `nom` et qui renvoie le booléen `True` si l'article est déjà présent dans la table `articles` et `False` sinon.

**Question 4.3** Créer une fonction `est_present_dans_rangements(nom)` qui prend en paramètres une chaîne de caractères `nom` et qui renvoie le booléen `True` si l'article est déjà présent dans la table `articles` et `False` sinon.

**Question 4.4** Créer une fonction `est_present_dans_stock(code)` qui prend en paramètres une chaîne de caractères `code` et qui renvoie le booléen `True` si l'article est déjà présent dans un enregistrement de la table `stocks` et `False` sinon.

**Question 4.5** Créer la fonction Python `modif_stockage(code, nom, quantite, rangement)` qui prend en paramètres :

- un code d'article (une chaine de caractère)
- un nom d'article (une chaine de caractère)
- une quantité (un entier)
- un rangement (une chaine de caractère)

Elle devra soit créer, soit modifier un enregistrement de la table `stocks` selon si l'article est déjà enregistré dans un stock ou non dans la table `stocks`.

Si l'article existe déjà dans un enregistrement de la table `stocks`, la fonction devra modifier la quantité et le rangement.

Sinon la fonction devra vérifier si l'article et le rangement existent déjà et les créer éventuellement dans les tables correspondantes.

## Partie 4 : Création d'un site (optionnel)

Créer un site en PHP permettant de :

- afficher la liste des articles, leurs codes, l'endroit où ils sont rangés et leurs quantités
- pouvoir insérer un nouvel article
- pouvoir insérer un nouveau rangement
- créer un nouveau stock
- modifier la quantité d'un article dans le stock
- modifier le rangement d'un article dans le stock

# Annexe

```python

import mysql.connector

def connexion() :
    return mysql.connector.connect(
      host="localhost",
      port=8889,
      user="root",
      password="root",
      database="stock_articles"
    )

def fin_connexion(mydb) :
    mydb.close()


class Article() :
    def __init__(self,nom, code) :
        self.__nom = nom
        self.__code = code

    def insere_BDD(self) :
        """
        insère l'article dans la table article
        """
        mydb = connexion()
        curseur = mydb.cursor(buffered=True)
        requete="INSERT INTO articles(code, nom) VALUES('"+self.__code+"','"+self.__nom+"')"
        curseur.execute(requete)
        mydb.commit()
        curseur.close()
        fin_connexion(mydb)

    def get_nom(self) :
        return self.__nom

    def get_code(self) :
        return self.__code

    def __repr__(self) :
        return "" + self.__nom + " : " + self.__code


class Rangement() :
    def __init__(self,nom) :
        self.__nom = nom

    def insere_BDD(self) :
        """
        insère le rangement dans la la table rangements
        """
        mydb = connexion()
        curseur = mydb.cursor(buffered=True)
        requete="INSERT INTO rangements(id, nom) VALUES(NULL,'"+self.__nom+"')"
        curseur.execute(requete)
        mydb.commit()
        curseur.close()
```

```
58          fin_connexion(mydb)
59
60      def get_nom(self) :
61          return self.__nom
62
63      def get_id(self) :
64          """
65          récupère l'id d'un rangement
66          """
67          mydb = connexion()
68          curseur = mydb.cursor(buffered=True)
69          requete="SELECT id FROM rangements WHERE nom='"+self.__nom+"'"
70          curseur.execute(requete)
71          ligne=curseur.fetchone()
72          curseur.close()
73          fin_connexion(mydb)
74          return ligne[0]
75
76      def __repr__(self) :
77          return self.__nom
78
79  class Stocks() :
80      """
81      gère les stocks
82      """
83      def __init__(self,code_article, code_rangement, quantite) :
84
85          self.__quantite = quantite
86          self.__article = code_article
87          self.__rangement = code_rangement
88
89      def insere_BDD(self) :
90          """
91          insère le stock dans la la table stocks
92          """
93          mydb = connexion()
94          curseur = mydb.cursor(buffered=True)
95          requete = "INSERT INTO stocks(id, article, rangement, quantite) VALUES(NULL,'"+self.__article
96          curseur.execute(requete)
97          mydb.commit()
98          curseur.close()
99          fin_connexion(mydb)
100
101     def __repr__(self) :
102         return "|"+self.__article.get_nom()+"|"+self.__article.get_code()+"|"+self.__rangement.get_no
103
104     def get_quantite(self) :
105         return self.__quantite
106
107     def get_rangement(self) :
108         return self.__rangement
109
110     def modif_quantite(self, quantite) :
111         """
112         modifie la quantité d'un article
113         :param quantite : (int)
114         """
115         self.__quantite = quantite
116         mydb = connexion()
```

```
117             curseur = mydb.cursor(buffered=True)
118             requete = "UPDATE stocks SET quantite ="+str(quantite)+" WHERE article = '"+self.__article+"'
119             curseur.execute(requete)
120             mydb.commit()
121             curseur.close()
122             fin_connexion(mydb)
123 
124         def modif_rangement(self,rangement) :
125             """
126             modifie l'endroit où est rangé l'article
127             :param rangement : (int)
128             """
129             self.__rangement = rangement
130             mydb = connexion()
131             curseur = mydb.cursor(buffered=True)
132             requete = "UPDATE stocks SET rangement ="+str(rangement)+" WHERE article = '"+self.__article+
133             curseur.execute(requete)
134             mydb.commit()
135             curseur.close()
136             fin_connexion(mydb)
137 
```