Les sous-programmes: exercices corrigés en Python
Corrigé

Exercice 1 : Livres

L'objectif de cet exercice est d'écrire un programme pour gérer une bibliothèque personnelle de taille réduite. On ne développera qu'une interface homme/machine minimale.

1.1 *Définition d'un livre.* Un livre est caractérisé par son titre, son nombre de pages et son genre. Le genre est soit roman, soit BD, soit art, soit technique, etc. Il faudrait bien sûr bien d'autres informations pour décrire complètement un livre.

1.1.1 Donner le type Livre.

Solution : D'après sa caractérisation, un livre peut être représenté par un type énuméré avec des champs titre (une chaîne de caractères), son nombre de page (entier) et son genre (un type énuméré puisqu'il peut prendre une parmi plusieurs valeurs).

```
Constante
    LG_TITRE = 50       -- Longueur maximal pour le titre d'un livre
Type
    Genre = (ROMAN, BD, ART, TECHNIQUE)

    Livre =
        Enregistrement
            titre: Chaîne[LG_TITRE]
            nb_pages: Entier
            genre: Genre
        FinEnregistrement
```

```
from enum import Enum # importation de la classe Enum du module enum
GENRE = Enum('GENRE', 'ROMAN BD ART TECHNIQUE')
LONGUEUR_TITRE_MAX = int(25) # nombre maximal de caractères d'un titre

class Livre :
    def __init__ (outSelf, inTitre, inNbPages, inGenre) :
        """
        ROLE : initialise un livre avec titre, nombre de pages et genre
        ENTREE inTitre : str ; inNbPages : int (>0) ; inGenre : GENRE
        SORTIE outSelf : Livre
        """
        outSelf.titre = str(inTitre)
        outSelf.nb_pages = int(inNbPages)
        outSelf.genre = inGenre
        # affichge-test pour indiquer la création d'un nouveau livre
        print("< Nouveau Livre :",inTitre,">")
```

1.1.2 Écrire un sous-programme qui permet d'initialiser un livre à partir de son titre, de son nombre de pages et de son genre.

Solution : Ce sous-programme prend en paramètre :

— le livre à initialiser (`out`) ;

— le titre du livre (**in**) ;
— le nombre de pages (**in**)
— et le genre du livre (**in**).

Comme il s'agit d'un sous-programme d'initialisation, nous en faisons une procédure dont voici la spécification.

```
Procédure initialiser_livre(livre: out Livre;
                            titre: in Chaîne;
                            npage: in Entier;
                            genre: in Genre) Est
    -- Initialiser un livre à partir de son titre, son nombre de pages
    -- et son genre.
    --
    -- Nécessite
    --  npage > 0
    --
    -- Assure
    --  livre.titre = titre
    --  livre.nb_pages = npages
    --  livre.genre = genre
```

Le code du sous-programme est alors immédiat puisqu'il s'agit d'initialiser tous les champs de l'enregistrement.

```
Début
    livre.titre <- titre
    livre.nb_pages <- npages
    livre.genre <- genre
Fin
```

```
# L'initialisation d'un livre se fait à sa création
```

1.1.3 Écrire un sous-programme qui permet d'afficher un livre.

Solution : Ce sous-programme n'a qu'un seul paramètre, le livre qui est en entrée. De plus il fait de l'affichage. Il s'agit donc nécessairement d'une procédure.

```
Procédure afficher_livre(livre: in Livre) Est
    -- Afficher un livre
```

Le code consiste à afficher chacun des champs.

```
Début
    Afficher livre.titre
    Afficher livre.nb_pages
    Afficher livre.genre
Fin
```

Pour afficher le genre du livre, on peut écrire un nouveau sous-programme.

```
def libelle_genre(inGenre) : # return str
    """
    ROLE : renvoie le libelle (chaîne) associé au code de genre indiqué
    ENTREE inGenre : GENRE
    """
    if   inGenre == GENRE.ROMAN :     return "ROMAN"
    elif inGenre == GENRE.BD :        return "B.D.␣"
    elif inGenre == GENRE.ART :       return "ART␣␣"
    elif inGenre == GENRE.TECHNIQUE : return "TECH␣"
    else :      return "GENRE␣INCONNU"

def  afficher_livre(inLivre) :
```

```
    """
    ROLE : affiche les informations attachées à un livre
    ENTREE inLivre : Livre
    """
    # afficher en cadrant le titre à gauche sur un zone de 20 (ljust)
    # et en le tronquant éventuellement s'il est trop long
    # et le nombre de pages à droite sur une zone de 4 (rjust)
    print(inLivre.titre.ljust(LONGUEUR_TITRE_MAX)[:LONGUEUR_TITRE_MAX], \
          str(inLivre.nb_pages).rjust(4) + "␣p.", \
          libelle_genre(inLivre.genre))
```

1.2 *La bibliothèque.* Nous nous limitons à une bibliothèque qui ne peut pas contenir plus de 1000 ouvrages qui sont rangés dans l'ordre alphabétique de leur titre.

1.2.1 Définir le type bibliothèque.

Solution : Nous représentons la bibliothèque par un tableau de capacité 1000. Cependant, pour connaître le nombre d'ouvrages effectivement présents, il faut gérer la taille (entier). La bibliothèque est donc un type enregistrement.

```
Constante
    MAX_LIVRES = 1000   -- nb maximal de livres dans la bibliothèque
Type
    Bibliothèque =
        Enregistrement
            livres: Tableau [1..MAX_LIVRES] De Livre
            nb_livres: Entier
        FinEnregistrement
```

```
MAX_LIVRES = int(1000) # nombre maximal de livres dans la bibliotheque

class Bibliotheque :
    def __init__ (outSelf) :
        """
        ROLE : initialise la bibliotheque spécifiée comme vierge (0 livres)
        SORTIE outSelf : Bibliotheque
        """
        outSelf.livres=list(MAX_LIVRES*[None]) # contiendra des éléments Livre
        outSelf.nb_livres = int(0)
```

1.2.2 Écrire un sous-programme pour initialiser une bibliothèque à vide.

Solution : Ce sous-programme prend en paramètre la bibliothèque (`out`). Il s'agit d'une procédure.

```
Procédure initialiser_bibliothèque(b: out Bibliothèque) Est
    -- Initialiser la bibliothèque à vide
    --
    -- Assure
    --  b.nb_livres = 0 -- bibliothèque vide
```

Le code est donc immédiat.

```
Début
    b.nb_pages <- 0
Fin
```

Notons qu'il n'est pas nécessaire d'initialiser les livres du tableau puisque la champs `nb_pages` mis à zéro indique qu'on ne peut accéder à aucune case du tableau `livres`.

```
# L'initialisation d'une bibliothèque se fait à sa création
```

1.2.3 Écrire un sous-programme pour afficher le contenu de la bibliothèque.
Solution : Ce sous-programme prend un seul paramètre, la bibliothèque (en `in`). Il s'agit donc d'une procédure.

```
Procédure afficher_bibliothèque(b: in Bibliothèque) Est
    -- Afficher les livres de la bibliothèque.
```

Pour l'implémentation, nous choisissons d'utiliser une répétition `Pour` puisque nous connaissons le nombre de livres à afficher.

```
Variable
    i: Entier   -- parcourir les livres
Début
    Pour i <- 1 JusquÀ i = b.nb_livres Faire
        afficher_livres(b.libres[i])
    FinPour
Fin
```

```
def  afficher_bibliotheque(inBiblio):
    """
    ROLE : affiche les livres contenus dans la bibliothèque spécifiée
    ENTREE inBiblio : Bibliotheque
    """
    pluriel = str("") # marque du pluriel si plusieurs livres (vide sinon)
    if inBiblio.nb_livres > 1 : pluriel = "S"
    print("LA_BIBLIOTHEQUE_CONTIENT",inBiblio.nb_livres,"LIVRE"+pluriel,":")
    for indexLivre in range(inBiblio.nb_livres) :
        afficher_livre(inBiblio.livres[indexLivre])
    print ()
```

1.2.4 Écrire un sous-programme pour ajouter un ouvrage dans la bibliothèque.
Solution : Ce sous-programme prend en paramètre :

— la bibliothèque dans laquel le livre doit être ajouté (`in out`) ;
— le livre à ajouter (`in`).

Il s'agit donc d'une procédure.

```
Procédure ajouter(b: in out Bibliothèque; l: in Livre) Est
    -- ajouter le livre l dans la bibliothèque b
```

Les livres sont rangés dans la bibliothèque dans l'ordre lexicographique des titres. Il s'agit donc de faire un tri par insertion pour trouver la place de du nouveau livre. Nous optons pour une insertion séquentielle dont voici le premier niveau de raffinage.

```
R1 : Raffinage De « ajouter »
  | Déterminer la position du livre     position: out Entier
  | Décaler les livres compris entre position et b.nb_livres
  | Ranger le nouveau livre
```

Ces étapes correspondant à des algorithmes classiques sur les tableaux, il ne sont pas détaillés en algorithmique.

```
def  ajouter_livre(ioBiblio, inLivre) :
    """
    ROLE : ajoute le livre inLivre dans la bibliotheque ioBiblio. L'ajout se
        fait par ordre alphabetique du titre, en decalant les livres suivants
    HYPOTHESE : ioBiblio.nb_livres < MAX_LIVRES  (bibliothèque non pleine)
    ENTREE inLivre : Livre
    M.A J. ioBiblio : Bibliotheque
    """
```

```
9       indexLivre = int() # index pour parcourir les livres
10      position = int()   # position où doit être range le livre inLivre
11      aInserer = bool()  # indique si le livre doit etre inséré avant d'autres
12                         # sinon il sera ajouté apres le dernier déjà présent
13
14      #1.Déterminer la position d'insertion du livre
15      position = 0 # index de la première position possible
16      aInserer = False # a priori
17      while not aInserer and (position < ioBiblio.nb_livres) :
18          if inLivre.titre < ioBiblio.livres[position].titre :
19              aInserer = True
20          else :
21              position = position + 1
22
23      #2.Décaler les livres suivants si besoin est, en commençant par la fin
24      if aInserer : # la boucle est en sens rétrograde (le pas vaut -1)
25          for indexLivre in range(ioBiblio.nb_livres, position-1, -1) :
26              ioBiblio.livres[indexLivre + 1] = ioBiblio.livres[indexLivre]
27
28      #3.Ranger le livre à la position définie
29      ioBiblio.livres[position] = inLivre
30      ioBiblio.nb_livres = ioBiblio.nb_livres + 1
```

1.2.5 Écrire un sous-programme pour indiquer combien il y a d'ouvrages d'un genre donné.

Solution : Ce sous-programme a pour paramètres :

— la bibliothèque dans laquelle on veut recenser les ouvrages (**in**) ;
— le genre cherché (**in**) ;
— le nombre d'ouvrages du genre cherché (**out**).

Un seul paramètre en sortie, deux en entrée. On peut donc en faire une fonction.

```
1  Fonction nb_livres_genre(b: in Bibliothèque; genre: in Genre): Entier Est
2      -- Nombre de livres de la bibliothèque b qui sont du genre spécifié.
```

Concernant l'implantation, il faut parcourir tous les ouvrages (boucle **Pour**) et comptabiliser ceux qui sont du genre demandé.

```
1  Variable
2      i: Entier   -- parcourir les livres
3  Début
4      Résultat <- 0
5      Pour i <- 1 JusquÀ i = b.nb_livres Faire
6          Si b.livres[i].genre = genre Alors
7              Résultat <- Résultat + 1
8          FinSi
9      FinPour
10 Fin
```

```
1  def nb_livres_genre(inBiblio, inGenre) : # return int
2      """
3      ROLE : renvoie le nombre de livres de la bibliotheque inBiblio
4             qui sont du genre spécifié
5      ENTREE inBiblio : Bibliotheque ; inGenre : GENRE
6      """
7      resultat = int(0) # nombre de livres respectant le critère
8      for indexLivre in range(inBiblio.nb_livres) :
9          if inBiblio.livres[indexLivre].genre == inGenre :
10             resultat = resultat + 1
11     return resultat
```

1.3 *Programme de test.* Écrire un programme de test des sous-programmes écrits ci-dessus.

Solution : Voici un programme de test minimal qui ajoute quelques livres dans une bibliothèque et affiche le nombre de livres des différents genres.

```
# PROGRAMME PRINCIPAL DE TEST
# ROLE : tester les procedures et fonctions de gestion d'une bibliotheque
print("Test des procédures de gestion d'une bibliothèque".upper())
print()
maBiblio = Bibliotheque()
afficher_bibliotheque(maBiblio)

ajouter_livre(maBiblio, Livre("Peinture", 120, GENRE.ART))
afficher_bibliotheque(maBiblio)

ajouter_livre(maBiblio, Livre("Tintin : les sept boules de cristal", 54, GENRE.BD))
afficher_bibliotheque(maBiblio)

ajouter_livre(maBiblio, Livre("Algorithmique", 540, GENRE.TECHNIQUE))
afficher_bibliotheque(maBiblio)

ajouter_livre(maBiblio, Livre("Apprendre Python", 140, GENRE.TECHNIQUE))
afficher_bibliotheque(maBiblio)

for genre in GENRE : # afficher le nombre de livres du genre courant
    print("Nb livres", libelle_genre(genre),":", nb_livres_genre(maBiblio, genre))

"""
Résultat de l'exécution du programme

TEST DES PROCÉDURES DE GESTION D'UNE BIBLIOTHÈQUE

LA BIBLIOTHEQUE CONTIENT 0 LIVRE :

< Nouveau Livre : Peinture >
LA BIBLIOTHEQUE CONTIENT 1 LIVRE :
Peinture                  120 p. ART

< Nouveau Livre : Tintin : les sept boules de cristal >
LA BIBLIOTHEQUE CONTIENT 2 LIVRES :
Peinture                  120 p. ART
Tintin : les sept boules   54 p. B.D.

< Nouveau Livre : Algorithmique >
LA BIBLIOTHEQUE CONTIENT 3 LIVRES :
Algorithmique             540 p. TECH
Peinture                  120 p. ART
Tintin : les sept boules   54 p. B.D.

< Nouveau Livre : Apprendre Python >
LA BIBLIOTHEQUE CONTIENT 4 LIVRES :
Algorithmique             540 p. TECH
Apprendre Python          140 p. TECH
Peinture                  120 p. ART
Tintin : les sept boules   54 p. B.D.

Nb livres ROMAN : 0
Nb livres B.D.  : 1
Nb livres ART   : 1
Nb livres TECH  : 2

"""
```