I - TP 2

Introduction à la POO

1 Objets : attributs et création

Le libraire du coin de votre rue décide de faire appel à vous pour informatiser son système de gestion de livres : l'objectif de ce tp est de comprendre comment on peut structurer les informations se rapportant à un livre, ainsi que les opérations que l'on peut effectuer sur les livres (en calculer le prix, par exemple) dans un ensemble cohérent.

En python, le mot-clé `class` permet de définir un nouveau type : c'est une nouvelle structure de donnée permettant de représenter, structurer et manipuler des objets.

Code python

```
class Livre:
    pass

livre1 = Livre()
```

Le code ci-dessus permet de créer un nouvel objet de type `Livre` : c'est le rôle de l'instruction `Livre()` (**avec des parenthèses**). Lorsque l'on exécute l'instruction `print(livre1)`, cela indique l'adresse mémoire où est stocké l'objet désigné par `livre1`.

Code python

```
# Un nouvel objet est créé et affecté à la variable livre1
print(livre1)
print(type(livre1))
```

Résultat

```
<__main__.Livre object at 0x7f3192936fe0>
<class '__main__.Livre'>
```

1.1 Attributs d'un objet

Le code ci-dessous permet d'ajouter différents **attributs** au livre : son titre (*La peste*), son auteur (Albert Camus), et son nombre total de page (59).

Code python

```
livre1.titre = "La peste"
livre1.auteur = "Camus Albert"
livre1.nb_pages = 59
```

1. Utiliser Thonny pour inspecter l'objet `livre1` : dans le menu *Affichage*, faire afficher les panneaux *Variables* et *Inspecteur d'objets* en cliquant sur ces entrées. Une fois le panneau *Inspecteur d'objets* ouvert, cliquer sur le bouton *Attributs*. Commenter.

2. Écrire le code python permettant de représenter le livre Fondation (256 pages), écrit par Isaac Asimov en 1951.

Code python

```
livre2 = Livre()
# À compléter
livre2.titre = ...
livre2.auteur = ...
livre2.nb_pages = ...
```

1.2 Création d'un nouveau livre

Écrire une fonction `créer_livre` qui prend en argument une chaine de caractère `t`, une chaîne de caractères `a` et un entier positif `n` et qui renvoie l'objet de type `Livre` de titre `t`, d'auteur `a` et constitué de `n` pages.

Utiliser la fonction `créer_livre` pour initialiser une variable `livre3` représentant le livre de votre choix.

Code python

```
def créer_livre(t, a, n):
    """ str, str, int -> Livre
    Renvoie un nouveau livre """
    pass
```

Code python

```
print(livre3)
```

Résultat

```
<__main__.Livre object at 0x7f3192935b70>
```

1.3 Fonction `__init__`

Dans un nouveau fichier intitullé `introduction_init.py`, copier-coller le code suivant. Ouvrir ce fichier avec Thonny, puis l'exécuter.

Code python

```
class Livre:
    def __init__(self, t, a, n):
        self.titre = t
        self.auteur = a
        self.nb_pages = n
```

1. Dans la console de Thonny, exécuter l'instruction `Livre("Les misérables", "Hugo Victor", 2135")`. Observer l'effet sur l'état de la mémoire. Commenter.
2. À quoi peut server la fonction `__init__` placée dans le corps de la classe `Livre` ?

2 Premières fonctions

2.1 Afficher un livre

Le libraire est un peu perturbé par l'effet de l'instruction `print(livre1)` : il préfèrerait un affichage plus lisible. La fonction `affiche` prend en argument un objet de type `Livre` et réalise un affichage partiel des informations du livre.

Modifier la fonction `affiche` afin que l'appel `affiche(livre1)` indique également le nom de l'auteur du livre dans la description affichée. Tester votre modification sur les variables de type `Livre` déjà définies.

Code python

```
def affiche(lvr):
    """ Livre -> None
    Affiche le livre lvr """
    titre_livre = lvr.titre
    pages_tot = lvr.nb_pages
    print(f"< Livre '{titre_livre}', nombre de pages {pages_tot} >")
```

Code python

```
affiche(livre1)
```

Résultat

```
< Livre 'La peste', auteur Camus Albert, nombre de pages 59 >
```

2.2 Prix d'un livre

2.2.1 Prix de base

Ce libraire un peu simplet calcule le prix de tous ses livres de la même manière : tous ses livres coûtent 12 centimes la page. Ainsi un livre constitué de 100 pages coûte 1,2€.

Écrire une fonction `prix_base` qui étant donné un livre `lvr` et un flotant `prix_page` renvoie le prix du livre en euros.

Code python

```
def prix_base(lvr, prix_page):
    """ Livre, int -> float
    Calcule le prix de base d'un livre, à prix_page centimes la page """
    pass
```

Code python

```
print(prix_base(livre1, 0.12))
```

Résultat

```
7.08
```

2.2.2 Prix réduit

Il arrive au libraire (qui est simplet, mais pas radin) de faire occasionnellement des réductions sur le prix des livres.

Écrire une fonction `prix_reduit` qui étant donné un livre `lvr`, un prix de vente à la page `prix_page` et un pourcentage de réduction `reduc` (un entier compris entre 0 et 100) calcule le prix du livre, après une réduction de `reduc` pourcent.

Code python

```
def prix_réduit(lvr, prix_page, reduc):
    """ Livre, float, int -> float
    Renvoie le prix du livre après réduction """
    pass
```

Code python

```
print(prix_réduit(livre1, 0.12, 50))
```

Résultat

```
3.54
```

2.3 Méthodes d'un objet

Sur ce même modèle, on peut imaginer :

- d'autres attributs pour la classe livre. Ils permettent d'ajouter de l'information à la description de l'objet.
- d'autres fonctions opérant sur les objets de type livre. Elles permettent de manipuler un livre.

Afin de structurer efficacement notre code, nous allons va *encapsuler* toutes ces informations **dans** la classe qui représente notre objet. Toutes les réponses aux questions suivantes du tp devront être écrites dans un fichier intitulé `librairie.py`.

Code python

```python
class Livre:
    def __init__(self, t, a, n):
        self.titre = t
        self.auteur = a
        self.nb_pages = n

    def affiche(self):
        """ Livre -> None """
        titre_livre = self.titre
        auteur_livre = self.auteur
        pages_tot = self.nb_pages
        print(f"< Livre '{titre_livre}', auteur {auteur_livre}, nombre de
        ↪  pages {pages_tot} >")

    def prix_base(self, prix_page):
        """ Livre -> float
        Calcule le prix de base d'un livre, à 12 centimes la page """
        return prix_page*self.nb_pages

    def prix_réduit(self, prix_page, reduc):
        """ Livre, int -> float
        Renvoie le prix du livre après réduction """
        # return (1 - reduc/100)*self.nb_pages*0.12
        return (1 - reduc/100)*self.prix_base(prix_page)
```

2.3.1 Méthodes d'un objet : utilisation

Exécuter dans la console les lignes du code suivant. Commenter la syntaxe utilisée.

Code python

```python
livre4.affiche()
livre4.prix_base(0.12)
livre4.prix_réduit(0.12, 50)
```

2.3.2 Ajout d'un attribut

Ajouter un attribut `disponible` à la classe `Livre` : il s'agit d'un entier positif ou nul qui représente combien de livres de ce type sont disponibles. L'attribut `disponible` d'un livre aura pour valeur par défaut 1.

Code python

```python
livre1 = Livre("La peste", "Camus Albert", 59)
livre2 = Livre("Fondation", "Asimoov Isaac", 265)
livre4 = Livre("Les misérables", "Hugo Victor", 2135)
```

Code python

```python
print(livre4.disponible)
```

Résultat

```
1
```

2.3.3 Reception d'un livre

Écrire une méthode `recevoir` de la classe `Livre` : cette méthode prend comme argument un entier nombre `n` correspondant au nombre de copies supplémentaires du livre `self`. Celles-ci seront ajoutées aux copies disponibles du livre : on modifiera pour cela l'attribut `disponible`.

Attention. On rappelle que le code de cette fonction devra être écrit *dans* la classe `Livre`.

Code python

```
def recevoir(self, n):
    """ Livre, int -> None
    n nouveaux exemplaires du livre sont disponibles """
    pass
```

Code python

```
livre4.recevoir(1)
print(livre4.disponible)
```

Résultat

```
2
```

2.3.4 Vente d'un livre

Écrire une méthode `vendre` de la classe `Livre` simule la vente du livre `self` :

- si aucun livre de ce type n'est disponible, la méthode renverra `0`.
- sinon : on calcule le prix du livre (avec le prix par page et la réduction passés en arguments) et on décrémente de 1 l'attribut `disponible` de `lvr`. La méthode renverra le prix du livre.

Code python

```
def vendre(self, prix_page, réduc):
    """ Vend le livre, renvoie l'argent généré par la vente. """
    pass
```

Code python

```
print(livre4.vendre(0.12, 0))
```

Résultat

```
256.2
```

3 Gestion d'une librairie

3.1 Introduction et motivation

Notre libraire cherche maintenant à gérer une collection de livre. Son stock de livre ne faisant qu'augmenter, il souhaite pouvoir répondre aux clients lui demandant s'il possède ou non un livre dont on lui donne le titre. Écrire une fonction `recherche` qui étant donné une liste de livres `collection` et un `titre`, renvoie le nombre d'exemplaires disponibles du livre intitulé `titre` que possède le libraire.

Code python

```
def recherche(collection, titre):
    """ [Livre], str -> int
    Nombre de livres disponible avec le titre indiqué dans la collection.
    """
    pass
```

Code python

```python
print(recherche([livre1, livre2, livre4], "Fondation"))
print(recherche([livre1, livre2, livre4], "La Peste"))
```

Résultat

```
1
0
```

3.2 La classe `Librairie`

Afin de structurer efficacement notre code, on décide de regrouper toutes les fonctions opérant sur une collection de livres dans une classe intitulée `Librairie`. Ainsi, à l'aide de cette classe, le libraire pourra efficacement gérer l'ensemble des livres de sa librairie. Dans le code ci-dessous, l'attribut `collection` est un objet de type `list` constitué d'objets de type `Livre`.

Code python

```python
class Librairie:
    def __init__(self, c):
        self.collection = c
```

1. Avec quelle instruction peut-on représenter une librairie vide (ne contenant aucun livre) ?
2. Initialiser une variable `lib` de type `Librairie` représentant une librairie contenant les livres suivants :
   - 8 exemplaires du livre La peste, d'Albert Camus (59 pages) ;
   - 2 exemplaires du livre Fondation, d'Isaac Asimov (256 pages) ;
   - 15 exemplairs du livre Les misérables, de Victor Hugo (2135 pages) ;
   - 5 exemplaires du livre Harry Potter à l'école des sorciers, de J. K. Rowling (132 pages)

3.2.1 Ajouter un livre dans une librairie

1. Écrire une méthode `ajoute` de la classe `Librairie` qui prend en argument un objet `lvr` de type `Livre` et qui l'ajoute à la librairie représentée par `self`.
2. Écrire le code python qui permettent d'ajouter à la variable `lib` définie précédemment 11 exemplaires du livre Le roi Babar (15 pages), écrit par Jean de Brunhoff.

Code python

```python
def ajoute_livre(self, lvr):
    """ Librairie, Livre -> None
    Ajoute le livre lvr à la librairie self """
    pass
```

Code python

```python
print(len(lib.collection)) #après ajout du livre Le roi Babar
```

Résultat

```
5
```

3.2.2 Afficher l'ensemble des livres de la librairie

Écrire une méthode `affiche` de la classe `Librairie` qui affiche les informations de tous les livres présents dans la librairie `self`. On pourra faire appel à la méthode `affiche` de la classe `Livre`.

Code python

```python
def affiche(self):
    """ Librairie -> None
    Affiche les livres de la librairie  """
    pass
```

Code python

```python
lib.affiche()
```

Résultat

```
< Livre 'La peste', auteur Camus Albert, nombre de pages 59 >
< Livre 'Fondation', auteur Asimoov Isaac, nombre de pages 265 >
< Livre 'Les misérables', auteur Hugo Victor, nombre de pages 2135 >
< Livre 'Harry Potter à l'école des sorciers', auteur J. K. Rowling,
 ↪  nombre de pages 132 >
< Livre 'Babar', auteur Jean de Brunhoff, nombre de pages 15 >
```

3.2.3 Calculer le bénéfice total

Écrire une méthode `bénéfice` de la classe `Librairie` qui renvoie le bénéfice total généré par la vente de tous les exemplaires de tous les livres de la librairie `self`. Cette fonction prendra comme argument `prix_page`, un flottant représentant le prix par page fixé par le libraire.

Code python

```python
def bénéfice(self, prix_page):
    """ Librairie, float -> None
    Calcule le bénéfice généré par la vente de tous les livres """
    pass
```

Code python

```python
print(lib.bénéfice(0.12))
```

Résultat

```
4062.24
```

3.2.4 À vous de jouer

1. Ajouter un nouvel attribut de votre choix à la classe `Livre`.

   Écrire une méthode de la classe `Livre` qui ajoute une fonctionnalité aux objets de type `Livre` (à vous de l'inventer !).

2. Ajouter un nouvel attribut de votre choix à la classe `Librairie`.

   Écrire une méthode de la classe `Librairie` qui ajoute une fonctionnalité aux objets de type `Librairie` (à vous de l'inventer !).