Nom :
Prénom :
Numéro d'étudiant :

13/02/2013
Durée : 40 min

Contrôle Continu
Informatique – Méthodes pour le Web

L3 AES Misashs

Tout au long de ce devoir, vous allez créer un programme permettant de gérer les livres d'une librairie.

Exercice 1 :

Créez une classe `Livre` avec son constructeur prenant en argument son titre, le nom de son auteur et son prix. Définissez une méthode `afficher` qui affiche le titre, le nom de l'auteur et le prix du livre. Créez un `Livre` en appelant le constructeur et appelez la méthode `afficher` avec ce `Livre`.

```
class Livre:
      def __init__(self, t, n, p):
        self.titre, self.nom, self.prix = t, n, p
      def afficher(self):
        print "Titre : "+self.titre
        print "Nom de l'auteur : "+self.nom
               print "Prix : "+str(self.prix)

l1 = Livre("Crime et chatiment","Dostoievski",7.50)
l1.afficher()
```

Exercice 2 :

Créez deux classes `Roman` et `Philosophie` qui héritent de la classe `Livre`. En plus du titre, du nom de l'auteur et du prix, le constructeur de la classe `Roman` prendra en argument le nom du courant littéraire auquel il appartient (par exemple « humanisme », « baroque », « romantisme », « réalisme », « naturalisme », « symbolisme », etc.).
Créez ensuite un `Roman` et un livre de `Philosophie` en appelant leurs constructeurs respectifs.

```
class Roman(Livre):
      def __init__(self, t, n, p, c):
        Livre.__init__(self, t, n, p)
        self.courant = c

class Philosophie(Livre):
      def __init__(self, t, n, p):
        Livre.__init__(self, t, n, p)

l2 = Roman("La nausee","Sartre",4.50,"Existentialisme")
l3 = Philosophie("Critique de la raison pure","Kant",10)
```

Exercice 3 :

Créez une classe `Librairie` dont le constructeur prend en argument une liste de `Livres` et une liste contenant les quantités disponibles en stock de ces livres dans la bibliothèque.

```
class Librairie:

      def __init__(self, ll, lq):
        self.livres, self.stock = ll, lq
```

Exercice 4 :

Ajoutez à la classe `Librairie` une méthode `acheter` qui prend un `Livre` et une quantité en argument. Si le livre est présent dans la liste des livres, alors la quantité prise en argument est ajoutée à la quantité correspondant au `Livre` dans la liste des quantités. Sinon, une exception est levée, le livre est ajouté à la liste des livres et la quantité est ajoutée à la liste des quantités. Utilisez pour cela `try` et `except`.

```
      def acheter(self, l, q):
        try:
                i = self.livres.index(l)
                self.stock[i] += q
        except ValueError:
                self.livres.append(l)
                self.stock.append(q)
```

Exercice 5 :

Dans la classe `Librairie`, définissez une méthode `afficher` qui appelle pour chaque `Livre` sa méthode `afficher` et qui affiche la quantité disponible.

```
      def afficher(self):
        for i in range(len(self.livres)):
                self.livres[i].afficher()
                print "Stock : "+str(self.stock[i])
```

Exercice 6 :

Imaginons que les classes `Livre`, `Roman` et `Philosophie` aient été créées dans un fichier livre.py et que la classe `Librairie` ait été créée dans un fichier librairie.py. Quelle(s) ligne(s) devra-t-on ajouter au fichier librairie.py pour que celui-ci puisse utiliser les classes/méthodes de livre.py ?

```
from livre import *
```