

# STRUCTURES DE DONNEES – DEVOIR

## EXERCICE 1 : LA POO EN QUELQUES QUESTIONS

1) On définit la classe `Identite` de la manière suivante :

```
1 class Identite:
2     def __init__(self, nom, prenom, an):
3         self.nom=nom
4         self.prenom=prenom
5         self.an=an
6     def age(self, a):
7         return a-self.an
```

a) Quels sont les attributs et méthodes de cette classe ?

Qu'affichera la console à la suite de l'exécution des instructions suivantes ?

- b) `maria=Identite('Pires','Maria',1999)`  
`print(maria.nom + ' ' + maria.prenom)`
- c) `print(maria.nom,maria.an)`
- d) `hugo=Identite('Dupont','Hugo',2008)`  
`print(hugo.age(2024))`

2) Ecrire le code de la classe `Voiture` qui permet d'afficher 'Ferrari Rouge' après avoir saisi les instructions suivantes:

```
>>> F430=Voiture('Ferrari','rouge')
>>> print(F430)
Ferrari rouge
```

## EXERCICE 2 : BRIOCHES

Un fabricant de brioches décide d'informatiser sa gestion des stocks. Il écrit pour cela un programme en langage Python. Une partie de son travail consiste à développer une classe `Stock` dont la première version est la suivante :

```
class Stock:
    def __init__(self):
        self.qt_farine = 0 # quantité de farine initialisée à 0 g
        self.nb_oeufs = 0 # nombre d'œufs (0 à l'initialisation)
        self.qt_beurre = 0 # quantité de beurre initialisée à 0 g
```

1. Écrire une méthode `ajouter_beurre(self, qt)` qui ajoute la quantité `qt` de beurre à un objet de la classe `Stock`.

On admet que l'on a écrit deux autres méthodes `ajouter_farine` et `ajouter_oeufs` qui ont des fonctionnements analogues.

2. Écrire une méthode `afficher(self)` qui affiche la quantité de farine, d'œufs et de beurre d'un objet de type `Stock`. L'exemple ci-dessous illustre l'exécution de cette méthode dans la console :

```
>>> mon_stock = Stock()
>>> mon_stock.afficher()
farine: 0
oeuf: 0
beurre: 0
>>> mon_stock.ajouter_beurre(560)
>>> mon_stock.afficher()
farine: 0
oeuf: 0
beurre: 560
```

3. Pour faire une brioche, il faut 350 g de farine, 175 g de beurre et 4 œufs. Écrire une méthode `stock_suffisant_brioche(self)` qui renvoie un booléen : `VRAI` s'il y a assez d'ingrédients dans le stock pour faire une brioche et `FAUX` sinon.
4. On considère la méthode supplémentaire `produire(self)` de la classe `Stock` donnée par le code suivant :

```
def produire(self):
    res = 0
    while self.stock_suffisant_brioche():
        self.qt_beurre = self.qt_beurre - 175
        self.qt_farine = self.qt_farine - 350
        self.nb_oeufs = self.nb_oeufs - 4
        res = res + 1
    return res
```

On considère un stock défini par les instructions suivantes :

```
>>> mon_stock=Stock()
>>> mon_stock.ajouter_beurre(1000)
>>> mon_stock.ajouter_farine(1000)
>>> mon_stock.ajouter_oeufs(10)
```

- a) On exécute ensuite l'instruction : `>>> mon_stock.produire()`  
Quelle valeur s'affiche dans la console ? Que représente cette valeur ?
- b) On exécute ensuite l'instruction : `>>> mon_stock.afficher()`  
Que s'affiche-t-il dans la console ?

5. L'industriel possède  $n$  lieux de production distincts et donc  $n$  stocks distincts. On suppose que ces stocks sont dans une liste dont chaque élément est un objet de type `Stock`. Écrire une fonction Python `nb_brioche(liste_stocks)` possédant pour unique paramètre la liste des stocks et renvoie le nombre total de brioches produites.