DEVOIR 1 – CORRECTION (Programmation Orientée Objet)

EXERCICE 1 : LA POO EN QUELQUES QUESTIONS

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

```
class Identite:
    def __init__(self, nom, prenom, an):
        self.nom=nom
        self.prenom=prenom
        self.an=an
    def age(self, a):
        return a-self.an
```

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

Attributs : `nom, prenom, an`
Méthodes : `age` et la méthode spéciale `__init__`

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

b) >> **`Pires Maria`**

c) >> **`Pires 1999`**

d) >> **`16`**

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

```
class Voiture:
    def __init__(self, nom, couleur):
        self.nom=nom
        self.couleur=couleur
    def __repr__(self):
        return self.nom  +' '+ self.couleur
```

EXERCICE 2: BRIOCHES

```
class Stock:
    def __init__(self):
        self.qt_farine = 0
        self.qt_beurre = 0
        self.nb_oeufs = 0
```

Commentaire : Le constructeur initialise les attributs du stock à 0. Chaque objet possède ses propres quantités.

1. Méthode d’ajout

```
def ajouter_beurre(self, qt):
    self.qt_beurre += qt
```

2. Méthodes d’affichage

```
def afficher(self):
    print("farine :", self.qt_farine)
    print("beurre :", self.qt_beurre)
    print("oeufs  :", self.nb_oeufs)
```

Commentaires : Chaque méthode d’ajout modifie un seul attribut. La méthode afficher permet de consulter l’état du stock.

3. Méthode stock_suffisant_brioche(self)

```
def stock_suffisant_brioche(self):
    return (self.qt_farine >= 350 and
            self.qt_beurre >= 175 and
            self.nb_oeufs >= 4)
```

Commentaire : La méthode renvoie True si le stock permet de fabriquer une brioche.

4. Méthode produire(self)

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

Commentaire : La boucle while se répète tant qu’il reste assez d’ingrédients. Chaque tour retire les quantités nécessaires pour une brioche.

a) Il affichera **2**, le nombre de brioches produites.
b) Résultat :

farine : 300
beurre : 650
œufs : 2

5. Fonction nb_brioches(liste_stocks)

```
def nb_brioches(liste_stocks):
    total = 0
    for s in liste_stocks:
        total += s.produire()
    return total
```

Commentaire : La fonction parcourt chaque stock et additionne le nombre de brioches produites.