Exercice Bac: POO

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

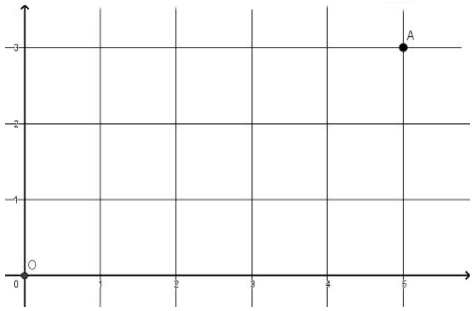
```
>>> mon_stock.afficher()
```

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_brioches(liste_stocks)` possédant pour unique paramètre la liste des stocks et renvoie le nombre total de brioches produites.

Exercice Bac: Programmation

On considère un jeu de plateforme où un personnage se déplace dans un espace à deux dimensions. Pour cela, on autorise seulement deux déplacements élémentaires : de la gauche vers la droite ou du bas vers le haut. La longueur d'un déplacement correspond au nombre de déplacements élémentaires qui le constituent. Afin de représenter ces déplacements, on se place dans un repère où les coordonnées sont des nombres entiers positifs. Le personnage au début du jeu est situé à l'origine du repère de coordonnées (0,0) et il souhaite se rendre au point A de coordonnées (5,3).



Les lignes de code de cet exercice seront écrites en langage Python.

On décide de coder les déplacements élémentaires de la manière suivante :

- le caractère `'0'` représente un déplacement élémentaire vers la droite,
- le caractère `'1'` représente un déplacement élémentaire vers le haut.

Un déplacement de longueur **n** sera donc une chaîne de caractères composée de **n** caractères `'0'` ou `'1'`. Par exemple, un déplacement possible de O à A est `'00011001'` et sa longueur est 8.

1. On considère la fonction `mystere` ci-dessous.

```
1 | def mystere(dep):
2 |     x = 0
3 |     y = 0
4 |     for c in dep:
5 |       if c == '0':
6 |         x = x+1
7 |       else:
8 |         y = y+1
9 |     return [x, y]
```

a. Que renvoie `mystere('01110111')` ?

b. De manière plus générale, que renvoie la fonction `mystere` pour une chaîne représentant un déplacement donné ?

2. Ecrire une fonction `accessible(dep, arrivee)` de sorte qu'elle renvoie `True` si le déplacement `dep` se termine sur le point `arrivee` et `False` dans le cas contraire. Les types des paramètres sont donc :
    - `dep` : une chaîne de caractères
    - `arrivee` : une liste de deux entiers

Par exemple : `accessible('00110001', [5,3])` renvoie `True` alors que `accessible('01010111', [5,3])` renvoie `False`.

On rappelle que l'expression `str(randint(0,1))` utilisant la fonction `randint` de la bibliothèque `random` renvoie aléatoirement un caractère égal à `'0'` ou `'1'`.

On décide de trouver un déplacement de longueur 8 qui permette d'atteindre le point `arrivee`. Pour cela, on se propose de créer de manière aléatoire un déplacement de longueur 8, de tester si le point `arrivee` est accessible pour ce déplacement et de recommencer tant que ce n'est pas le cas.

3. Recopier et compléter la fonction **chemin** ci-dessous qui prend en paramètre les coordonnées du point `arrivee` et qui renvoie un déplacement de 8 pas qui permette d'atteindre le point `arrivee`. Quelles sont les préconditions sur le paramètre `arrivee` ?

```
1 | from random import randint
2 | def chemin(arrivee):
3 |   deplacement = '00000000'
4 |   while ..... :
5 |     .....
6 |     for k in range(8):
7 |       pas = str(randint(0,1))
8 |       ..... = deplacement + .....
9 |   return deplacement
```

4. La fonction `int(ch,2)` renvoie l'écriture décimale d'un nombre donné en binaire sous forme d'une chaîne de caractères `ch`. Par exemple `int('00000101',2)` renvoie `5`. Quelle est la plus grande valeur possible renvoyée par `int(chem,2)` si `chem` est un déplacement de longueur 8 permettant d'atteindre le point A de coordonnées (5,3) ?