POO : Exercices type BAC

Exercice 1 : Une agence immobilière

Une agence immobilière développe un programme pour gérer les biens immobiliers qu'elle propose à la vente.

Dans ce programme, pour modéliser les données de biens immobiliers, on définit une classe `Bim` avec les attributs suivants :

- `nt` de type `str` représente la nature du bien (appartement, maison, bureau, commerces, ...) ;
- `sf` de type `float` est la surface du bien ;
- `pm` de type `float` est le prix moyen par m² du bien qui dépend de son emplacement.

La classe `Bim` possède une méthode `estim_prix` qui renvoie une estimation du prix du bien. Le code (incomplet) de la classe `Bim` est donné ci-dessous :

```
class Bim:
    def __init__(self, nature, surface, prix_moy):
        ...
    def estim_prix(self):
        return self.sf * self.pm
```

1. Recopier et compléter le code du constructeur de la classe `Bim`.
2. On exécute l'instruction suivante :

```
b1 = Bim('maison', 70.0, 2000.0)
```

Que renvoie l'instruction `b1.estim_prix()` ? Préciser le type de la valeur renvoyée.

3. On souhaite affiner l'estimation du prix d'un bien en prenant en compte sa nature :
   - pour un bien dont l'attribut `nt` est `'maison'` la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m² multiplié par 1,1 ;
   - pour un bien dont l'attribut `nt` est `'bureau'` la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m² multiplié par 0,8 ;
   - pour les biens d'autres natures, l'estimation du prix ne change pas.

   Modifier le code de la méthode `estim_prix` afin de prendre en compte ce changement de calcul.

4. Écrire le code Python d'une fonction `nb_maison(lst)` qui prend en argument une liste Python de biens immobiliers de type `Bim` et qui renvoie le nombre d'objets de nature `'maison'` contenus dans la liste `lst`.

Exercice 2 : cryptage de César

Dans cet exercice, on étudie une méthode de chiffrement de chaînes de caractères alphabétiques. Pour des raisons historiques, cette méthode de chiffrement est appelée "*code de César*". On considère que les messages ne contiennent que les lettres capitales de l'alphabet "ABCDEFGHIJKLMNOPQRSTUVWXYZ" et la méthode de chiffrement utilise un nombre entier fixé appelé la clé de chiffrement.

1. Soit la classe `CodeCesar` définie ci-dessous :

```
class CodeCesar:

    def __init__(self, cle):
        self.cle = cle
        self.alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

    def decale(self, lettre):
        num1 = self.alphabet.find(lettre)
        num2 = num1+self.cle
        if   num2 >= 26:
            num2 = num2-26
        if num2 < 0:
            num2 = num2+26
        nouvelle_lettre = self.alphabet[num2]
        return nouvelle_lettre
```

On rappelle que la méthode `str.find(lettre)` renvoie l'indice (index) de la `lettre` dans la chaîne de caractères `str`

Représenter le résultat d'exécution du code Python suivant :

```
code1 = CodeCesar(3)
print(code1.decale('A'))
print(code1.decale('X'))
```

2. La méthode de chiffrement du « code César » consiste à décaler les lettres du message dans l'alphabet d'un nombre de rangs fixé par la clé. Par exemple, avec la clé 3, toutes les lettres sont décalées de 3 rangs vers la droite : le A devient le D, le B devient le E, etc.

Ajouter une méthode `cryptage(self, texte)` dans la classe `CodeCesar` définie à la question précédente, qui reçoit en paramètre une chaîne de caractères (le message à crypter) et qui retourne une chaîne de caractères (le message crypté).

Cette méthode `cryptage(self, texte)` doit crypter la chaîne `texte` avec la clé de l'objet de la classe `CodeCesar` qui a été instancié.

Exemple :

```
>>> code1 = CodeCesar(3)
>>> code1.cryptage("NSI")
'QVL'
```

3. **Ecrire** un programme qui :

- demande de saisir la clé de chiffrement
- crée un objet de classe `CodeCesar`
- demande de saisir le texte à chiffrer
- affiche le texte chiffré en appelant la méthode `cryptage`

4. On ajoute la méthode `transforme(texte)` à la classe `CodeCesar` :

```
def transforme(self, texte):
        self.cle = -self.cle
        message = self.cryptage(texte)
        self.cle = -self.cle
        return message
```

On exécute la ligne suivante : `print(CodeCesar(10).transforme("PSX"))`

Que va-t-il s'afficher ? **Expliquer** votre réponse.