P.O.O : Devoir n°2

Thème 1 - Structure de données

Eval. | P.O.O

1. D'après 2021, Centres étrangers, J1, Ex. 1

Dans cet exercice, on étudie une méthode de chiffrement de chaines 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 :

🐍 **Script Python**

```
class CodeCesar:
    def __init__(self, cle):
        self.cle = cle

    def decale(self, lettre):
        indice_1 = indice_capitale(lettre)
        indice_2 = indice_1 + self.cle
        if indice_2 >= 26:
            indice_2 = indice_2 - 26
        if indice_2 < 0:
            indice_2 = indice_2 + 26
        nouvelle_lettre = lettre_capitale(indice_2)
        return nouvelle_lettre
```

On dispose aussi des fonctions `indice_capitale(indice)` et `lettre_capitale(lettre)` qui renvoient la lettre majuscule correspondant à un indice donné et l'indice (la position d'une lettre) dans l'alphabet latin usuel.

Passer d'un indice à une lettre

Les fonctions `indice_lettre(indice)` et `lettre_indice (lettre)` peuvent être définies par

🐍 **Script Python**

```
def lettre_capitale(indice: int) -> str:
    assert 0 <= indice < 26, "L'indice doit être entre 0 et 25"
    return chr(ord('A') + indice)

def indice_capitale(lettre: str) -> int:
    assert lettre in [chr(i + ord('A')) for i in range(26)], "La lettre doit être dans l'alphabet latin capital"
    return ord(lettre) - ord('A')
```

Représenter le résultat d'exécution du code Python suivant :

🐍 **Console Python**

```
>>> code = CodeCesar(3)
>>> code.decale('A')
...
>>> code.decale('X')
...
```

Réponse

La première instruction permet d'initialiser un objet `CodeCesar` correspondant à un décalage de 3 caractères. Ainsi `A`, en position 0 devient la lettre en position 3, c'est à dire `D`. Pour le `X`, si on applique le même décalage, on obtiendrai la lettre en position 23+3 = 26, ce qui ferait échouer la fonction `lettre_indice()`. On soustrait donc 26 pour trouver `A`.

🐍 **Console Python**

```
>>> code = CodeCesar(3)
>>> code.decale('A')
D
>>> code.decale('X')
A
```

2. La méthode de chiffrement du « code de 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 `chiffre(self, texte)` dans la classe `CodeCesar` définie à la question précédente, qui reçoit en paramètre une chaîne de caractères (le message à chiffrer) et qui renvoie une chaîne de caractères (le message chiffré).

Cette méthode `chiffre(self, texte)` doit chiffrer la chaîne `texte` avec la clé de l'objet de la classe `CodeCesar` qui a été instanciée.

Exemple :

🐍 **Console Python**

```
>>> code = CodeCesar(3)
>>> code.chiffre("NSI")
'QVL'
```

Réponse

🐍 **Script Python**

```
def chiffre(self, texte):
    chaine = ""
    for caractere in texte:
        chaine = chaine + self.decale(caractere)
    return chaine
```

3. Écrire une fonction qui : * prend en argument la clef de chiffrement et le message à chiffrer ; * instancie un objet de la classe `CodeCesar` ; * renvoie le texte chiffré.

Réponse

🐍 **Script Python**

```
def chiffre_texte(clef, texte):
    return CodeCesar(clef).chiffre(texte)
```

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

🐍 **Script Python**

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

On exécute la ligne suivante dans une console `CodeCesar(10).transforme("PSX")`

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

Réponse

Le message `'FIN'` va s'afficher sur la console.

La méthode proposée permet de déchiffrer le message en appliquant la transformation opposée.

2. D'après 2022, Centres étrangers, J2, Ex. 4

Simon souhaite créer en Python le jeu de cartes « la bataille » pour deux joueurs. Les questions qui suivent demandent de reprogrammer quelques fonctions du jeu.

Règles du jeu de la bataille

Préparation

- Distribuer toutes les cartes aux deux joueurs.
- Les joueurs ne prennent pas connaissance de leurs cartes et les laissent en tas face cachée devant eux.

Déroulement

- À chaque tour, chaque joueur dévoile la carte du haut de son tas.
- Le joueur qui présente la carte ayant la plus haute valeur emporte les deux cartes qu'il place sous son tas.
- Les **valeurs des cartes** sont : dans l-ordre de la plus forte à la plus faible : As, Roi, Dame, Valet, 10, 9, 8, 7, 6, 5, 4, 3 et 2 (la plus faible)

Si deux cartes sont de même valeur, il y a « bataille ».

- Chaque joueur pose alors une carte face cachée, suivie d'une carte face visible sur la carte dévoilée précédemment.
- On recommence l'opération s'il y a de nouveau une bataille sinon, le joueur ayant la valeur la plus forte emporte tout le tas.

Lorsque l'un des joueurs possède toutes les cartes du jeu, la partie s'arrête et ce dernier gagne.

Pour cela Simon crée une classe Python `Carte`. Chaque instance de la classe a deux attributs : un pour sa `valeur` et un pour sa `couleur`. Il donne au valet la valeur \(11\), à la dame la valeur \(12\), au roi la valeur \(13\) et à l'as la valeur \(14\). La couleur est une chaine de caractères : `"trefle"`, `"carreau"`, `"coeur"` ou `"pique"`.

1. Simon a écrit la classe Python `Carte` suivante, ayant deux attributs `valeur` et `couleur`, et dont le constructeur prend deux arguments : `val` et `coul`.

1.a. Recopier et compléter les `...` des lignes 3 et 4 ci-dessous.

```
1   class Carte:
2       def __init__(self, val, coul):
3           ... .valeur = ...
4           ... = coul
```

Réponse

```
1   class Carte:
2       def __init__(self, val, coul):
3           self.valeur = val
4           self.couleur = coul
```

1.b. Parmi les propositions ci-dessous quelle instruction permet de créer l'objet « 7 de cœur » sous le nom `c7` ?

- `c7.__init__(self, 7, "coeur")`
- `c7 = Carte(self, 7, "coeur")`
- `c7 = Carte(7, "coeur")`
- `from Carte import 7, "coeur"`

Réponse

`c7 = Carte(7, "coeur")` crée une instance de la classe `Carte` de valeur \(7\) et de couleur `"coeur"`, puis l'affecte à la variable `c7`.

2. On souhaite créer le jeu de cartes. Pour cela, on écrit une fonction `initialiser` :

- sans paramètre
- qui renvoie une liste de 52 objets de la classe `Carte` représentant les 52 cartes du jeu.

Voici une proposition de code. Recopier et compléter les lignes suivantes pour que la fonction réponde à la demande :

🐍 **Script Python**

```
def initialiser() :
    jeu = []
    for coul in ["coeur", "carreau", "trefle", "pique"]:
    for val in range(...):
        carte_cree = ...
        jeu.append(carte_cree)
    return jeu
```

Réponse

🐍 **Script Python**

```
def initialiser() :
    jeu = []
    for coul in ["coeur", "carreau", "trefle", "pique"]:
    for val in range(2, 15):
        carte_cree = Carte(val, coul)
        jeu.append(carte_cree)
    return jeu
```

3. On rappelle que dans une partie de bataille, les deux joueurs tirent chacun une carte du dessus de leur tas, et celui qui tire la carte la plus forte remporte les deux cartes et les place en dessous de son tas.

Parmi les structures linéaires de données suivantes : Tableau, File, Pile, quelle est celle qui modélise le mieux un tas de cartes dans ce jeu de la bataille ? Justifier votre choix.

Réponse

On a besoin d'une structure linéaire pour

- extraire une carte à une seule extrémité ;
- ajouter une carte à l'autre extrémité (elle sera donc loin d'être piochée).

FILO : *First In Last Out* ; premier entré, dernier sorti, pour la File.

C'est la **File** qui répond le mieux à la modélisation souhaitée.

4. Écrire une fonction `comparer` qui prend en paramètres deux objets de la classe Carte : `carte_1`, `carte_2`. Cette fonction renvoie :

- \(0\) si la valeur des deux cartes est identique ;
- \(1\) si la carte `carte_1` a une valeur strictement plus forte que celle de `carte_2` ;

• \(-1\) si la carte `carte_2` a une valeur strictement plus forte que celle de `carte_1` .

Réponse

🐍 Script Python

```
def comparer(carte_1, carte_2):
    if carte_1.valeur > carte_2.valeur:
        return 1
    elif carte_1.valeur < carte_2.valeur:
        return -1
    else:
        return 0
```