

POO : TD

Thème 1 - Structure de données

BAC

TD : Sujet BAC Programmation Orientée Objet (POO)

1. Exercice 1 : (d'après Bac) - Locations de chambres.

Principaux thèmes abordés :

Structure de données (programmation objet) et langages et programmation (spécification).

La société LOCAVACANCES doit gérer la réservation de l'ensemble des chambres de ses gîtes. Chaque chambre d'un même complexe sera différenciée par son nom.

Pour cela, d'un point de vue informatique, on a créé deux classes : `Chambre` et `Gite` dont le code ci-dessous.

Script Python

```
class Chambre:
    def __init__(self, nom: str):
        self._nom = nom
        self._occupation = [False for i in range(365)]

    def get_nom(self):
        return self._nom

    def get_occupation(self):
        return self._occupation

    def reserver(self, date: int):
```

```

        self._occupation[date - 1] = True

class Gite:
    def __init__(self, nom: str):
        self._nom = nom
        self._chambres = []

    def __str__(self):
        n = len(self._chambres)
        if n == 0:
            return "L'hôtel " + self._nom + " n'a aucune chambre."
        else:
            return "L'hôtel " + self._nom + " a " + str(n) + " chambre(s)"

    def get_chambres(self):
        return self._chambres

    def get_nchambres(self):
        return [ch.get_nom() for ch in self._chambres]

    def ajouter_chambres(self, nom_ch : str):
        self._chambres.append(Chambre(nom_ch))

    def mystere(self, date):
        l_ch = []
        for ch in self._chambres :
            if ch.get_occupation()[date - 1] == False :
                l_ch.append(ch.get_nom())
        return(l_ch)

```

1.1. Partie A - Étude de la classe Chambre :

1. Lister les attributs en donnant leur type. Préciser s'ils sont modifiables dans la classe, en explicitant la méthode associée.

 Réponse :

1. Écrire un assert dans la méthode `reserver` pour vérifier si le nombre `date` passé en paramètre est bien compris entre 1 et 365 (on ne gère pas les années bissextiles).

 Réponse :

1. Écrire la méthode `annuler_reserver(self, date : int)` qui annule la réservation pour le jour `date`.

 Réponse :

1.2. Partie B - Étude de la classe Gite :

Le gîte « BonneNuit » a 5 chambres dénommées :

'Ch1', 'Ch2', 'Ch3', 'Ch4', 'Ch5'

On définit l'objet `giteBN` par l'instruction : `giteBN = Gite("BonneNuit")`.

1. Méthode `ajouter_chambres()`

Écrire l'instruction Python pour ajouter '`Ch1`' à l'objet `giteBN`.

 Réponse :



Dans les questions suivantes 2, 3 et 4, on considère que l'objet `giteBN` contient toutes les chambres du gîte « BonneNuit ».

1. La méthode `ajouter_chambres` permet d'enregistrer une nouvelle chambre, mais elle ne teste pas si le nom de cette chambre existe déjà.

Modifier la méthode pour éviter cet éventuel doublon.

 Réponse :



1. Étude des méthodes : `get_chambres()` et `get_nchambres()`.

a. Parmi les 4 propositions ci-dessous, quel est le type renvoyé par l'instruction Python : `giteBN.get_chambres()` .

- String
- Objet Chambre
- Tableau de String
- Tableau d'objets Chambre

 Réponse :



b. Qu'affiche la suite d'instructions suivante ?

 Script Python

```
ch = giteBN.get_chambres()[1]
print(ch.get_nom())
```

 Réponse :



c. Quelle différence existe-t-il entre les deux méthodes `get_nchambres()` et `get_chambres()` ?

 Réponse :



1. Les chambres 'ch1', 'Ch3', 'ch5' sont réservées pour tout le mois de Janvier 2021.

a. Que va renvoyer l'instruction `giteBN.mystere(3)` ?

 Réponse :

> |

b. Dans la méthode `mystere` de la classe `Gite`, quel est le type des variables en paramètre et en sortie ? Quelles sont les méthodes ou attributs dont cette méthode a besoin ?

 Réponse :

> |

2. Exercice 2 : (d'après Bac) - Pots de yaourts "

Principaux thèmes abordés :

structure de données (programmation objet) et langages et programmation (spécification).

Une entreprise fabrique des yaourts qui peuvent être soit nature (sans arôme), soit aromatisés (fraise, abricot ou vanille).

Pour pouvoir traiter informatiquement les spécificités de ce produit, on va donc créer une classe `Yaourt` qui possèdera un certain nombre d'attributs :

- Son genre : nature ou aromatisé
- Son arôme : fraise, abricot, vanille ou aucun
- Sa date de durabilité minimale (DDM) exprimée par un entier compris entre 1 et 365 (on ne gère pas les années bissextilles). Par exemple, si la DDM est égale à 15, la date de durabilité minimale est le 15 janvier.

On va créer également des méthodes permettant d'interagir avec l'objet `Yaourt` pour attribuer un arôme ou récupérer un genre par exemple. On peut représenter cette classe par le tableau de spécifications ci-dessous :

| Yaourt | |
|--------|--|
| | |

| ATTRIBUTS | METHODES |
|-----------|-------------------------|
| genre | construire(arome,duree) |
| arome | obtenir_arome() |

| ATTRIBUTS | METHODES |
|-----------|------------------------|
| duree | obtenir_genre() |
| | obtenir_duree() |
| | attribuer_arome(arome) |
| | attribuer_duree(duree) |
| | attribuer_genre(arome) |

1. Code partiel de la classe Yaourt, à compléter aux endroits indiqués en suivant les consignes des questions suivantes :

Script Python

```
class Yaourt:
    """ Classe définissant un yaourt caractérisé par :
        - son arôme
        - son genre
        - sa durée de durabilité minimale"""

    def __init__(self,arome,duree):
        # **** Assertions : à compléter suivant les indications de la question 1.a.
        self.__arome = arome
        self.__duree = duree
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'

    # **** Méthode get_arome(self) à compléter suivant les indications de la question 1.c.

    def get_duree(self):
        return self.__duree

    def get_genre(self):
        return self.__genre

    def set_duree(self,duree):
        # **** Mutateur de durée
        if duree > 0 :
            self.__duree = duree

    # **** Mutateur d'arôme set_arome(self,arome) - à compléter suivant les indications de la question 2.

    def __set_genre(self,arome):
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'
```

1.a. Quelles sont les assertions à prévoir pour vérifier que l'arôme et la durée correspondent bien à des valeurs acceptables? Il faudra aussi expliciter les commentaires qui seront renvoyés.

Pour rappel :

- L'arôme doit prendre comme valeur 'fraise', 'abricot', 'vanille' ou 'aucun'.
- Sa date de durabilité minimale (DDM) est une valeur positive.

Réponse :

1.b. Pour créer un yaourt, on exécutera la commande suivante :

Script Python

```
mon_yaourt = Yaourt('fraise', 24)
```

Quelle valeur sera affectée à l'attribut `genre` associé à `mon_yaourt` ?

Réponse :

1.c. Écrire en Python une fonction `get_arome(self)`, renvoyant l'arôme du yaourt créé.

Réponse :

1. On appelle mutateur une méthode permettant de modifier un ou plusieurs attributs d'un objet.

Ecrire en Python le mutateur `set_arome(self, arôme)` permettant de modifier l'arôme du yaourt.

On veillera à garder une cohérence entre l'arôme et le genre.

Réponse :

1. On veut créer une pile contenant le stock de yaourts. Pour cela il faut tout d'abord créer une pile vide :

Script Python

```
def creer_pile():
    pile = []
    return pile
```

3.a. Créer une fonction `empiler(p, yaourt:Yaourt)` qui renvoie la pile `p` après avoir ajouté un objet de type `Yaourt` au sommet de la pile.

Réponse :

3.b. Créer une fonction `depiler(p)` qui renvoie l'objet à dépiler.

 Réponse :

> |

3.c. Créer une fonction `est_vide(p)` qui renvoie `True` si la pile est vide et `False` sinon.

 Réponse :

> |

3.d. Qu'affiche le bloc de commandes ci-dessous ?

 Script Python

```
mon_yaourt1 = Yaourt('aucun',18)
mon_yaourt2 = Yaourt('fraise',24)
ma_pile = creer_pile()
empiler(ma_pile, mon_yaourt1)
empiler(ma_pile, mon_yaourt2)
print(depiler(ma_pile).get_duree())
print(est_vide(ma_pile))
```

 Réponse :

> |