Robots

Travaux Pratiques 02 du module 06 – La POO

Avant de démarrer ce TP, il convient d'avoir suivi les modules 1 à 6 de ce cours.

Durée estimée
1h à 1h30

Énoncé

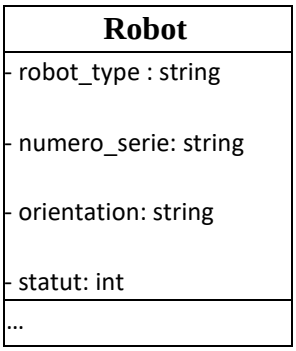
L'objectif de cet exercice est de gérer des robots, en mettant en œuvre les principes de la POO en Python, et notamment l'héritage.

Première partie : création de la classe Robot

1. Créez un dossier **Modele,** et ajoutez-y un fichier Robot.py.

2. Créer une classe **Robot**. Ajoutez-y un constructeur, que vous allez compléter au fur et à mesure des questions suivantes.

Les attributs d'un Robot peuvent-être modélisés de la manière suivante :



3. Attribut robot_type – Ajoutez cet attribut en respectant les contraintes suivantes :
   - La valeur de cet attribut peut-être précisé à la construction.
   - Par défaut, il est égal à « Générique »
   - Cet attribut peut être accédé en lecture et en écriture. Gérez cette contrainte à l'aide du décorateur @property

- Cet attribut doit faire au minimum 2 caractères. Lorsque cette contrainte n'est pas respectée, on affiche un message d'erreur, et on attribue à l'attribut sa valeur par défaut.

4. Attribut numero_serie : Ajoutez cet attribut en respectant les contraintes suivantes :
   - Généré automatiquement
   - À partir des deux caractères aléatoires (en majuscules)
   - Suivi par un nombre aléatoire entre 0 et 1 000 000 000, sur 10 caractères.
   - Exemple : ZD0000154784
   - Signalez dans le code que cet attribut ne doit pas être modifié. Il peut en revanche être consulté (via @property).

5. Ajoutez un attribut de classe permettant de compter le nombre de robot ayant été créés.

6. Attribut orientation - Ajoutez cet attribut en respectant les contraintes suivantes :
   - Valeurs possibles : NORD, EST, SUD ou OUEST.
   - Par défaut NORD à la construction du robot.
   - Cet attribut peut être accédé en lecture (via @property)
   - Cet attribut pourra être modifié via une méthode spécifique, que nous écrirons plus loin dans le TP.
   - Vous pouvez vous aider d'un attribut de classe listant les valeurs autorisées.

7. Attribut statut - Ajoutez cet attribut en respectant les contraintes suivantes :
   - Entier valant 1, 2 ou 3, pour « En service », « Hors Service » ou « En réparation ».
   - Vous pouvez vous aider d'un attribut de classe listant les valeurs autorisées, pour faire la correspondance entre l'entier et la valeur textuelle à laquelle il correspond.
   - 1 (« EN SERVICE ») par défaut, à la construction du robot.
   - Peut-être modifié, en lecture et en écriture (via @property)
   - Lorsqu'il est modifié, assurez-vous qu'une valeur valide (1,2 ou 3 est saisie)

8. Ajoutez une méthode d'affichage :
   - Redéfinir la méthode par défaut
   - Exemple :
     Robot ZD0000154784 – Générique
     Statut : EN SERVICE

Orientation : OUEST

9. Ajouter une méthode tourner, pour modifier l'orientation du robot :
    - Elle prend en paramètre un entier (1 ou -1) indiquant la direction du tour
    - Un message est affiché si le paramètre ne vaut pas 1 ou -1, et le robot ne tourne pas
    - Avec 1, le robot tourne dans le sens horaire
    - Avec -1, il tourne dans le sens anti-horaire
    - Exemple : le robot est orienté SUD. S'il tourne de 1, il se retrouve orienté OUEST. S'il tourne de -1, il se retrouve orienté EST.

10. Tester votre proposition via le Test.py fourni pour ce TP.

Deuxième partie : création de la classe RobotMobile

Un robot mobile est un robot qui, en plus de tourner, à la possibilité de se déplacer dans un espace à deux dimensions.

1. Dans un fichier RobotMobile.py, créez la classe RobotMobile, qui hérite de Robot.

2. Un robot mobile possède deux attributs supplémentaires :
    - Abscisse, qui représente la position du robot sur une dimension. Dans le constructeur qui surcharge celui de la classe mère, ajoutez cet attribut, qui vaut 0 par défaut.
    - Ordonnée, sa position dans l'autre dimension. Procédez comme pour l'abscisse.
    - Ces deux attributs sont accessibles uniquement en lecture (via @property)

3. Modifier le constructeur pour qu'il puisse en option accepter une abscisse et une ordonnée spécifiées à la construction du robot.

4. Ajouter une méthode afficher_position :
    - Qui renvoie une chaîne de caractère avec les deux coordonnées du robot
    - Exemple : Position : [abs=5 ; ord=4]

5. Ajouter une méthode avancer, qui va permettre de déplacer le robot. Elle prend en paramètre le nombre m de mètres duquel avance le robot. Les règles sont les suivantes :

- Si le robot est orienté vers l'Est, l'abscisse augmente de m
- Si le robot est orienté vers l'Ouest, l'abscisse diminue de m
- Si le robot est orienté vers le Nord, l'ordonnée augmente de m
- Si le robot est orienté vers le Sud, l'ordonnée diminue de m

6. Redéfinissez la méthode d'affichage de la classe Robot :
   - Faites appel à la méthode de la classe mère
   - Utilisez la méthode afficher_position
   - Exemple :
     Robot ZD0000154784 – Générique
     Statut : EN SERVICE
     Orientation : OUEST
     Position : [abs=5 ; ord=4]

7. Tester le programme via le fichier test.py fourni?

Troisième partie : création d'un robot aspirateur

1. Dans un fichier Aspirateur.py du dossier Modele, créez la classe Aspirateur

2. Ajoutez un constructeur. Un aspirateur possède les caractéristiques suivantes :
   - Une marque, qui doit forcément être précisée
   - Une puissance, sous forme d'un entier, qui doit aussi être précisée
   - Ajouter getter et setter pour ces attributs, via @property

3. Redéfinissez la méthode d'affichage.
   - Exemple : Aspirateur Bosh, puissance=2000W

4. Ajoutez une classe AspirateurRobot, qui hérite d'Aspirateur et de RobotMobile.

5. Modifier le constructeur pour prendre en compte que :
   - Le type de ce robot est toujours « Aspirateur robot »
   - Ajoutez l'attribut distance_max, qui représente la distance maximale que peut parcourir le robot une fois chargé, qui doit être précisée à la construction. Ajouter getter et setter via @property
   - L'abscisse et l'ordonnée sont toujours initialisées à 0

6. Sans toucher à la méthode d'affichage pour l'instant, créez un aspirateur robot et affichez-le. Inversez l'ordre d'héritage, et affichez-le à nouveau. Que constatez-vous ?

7. Modifiez la méthode d'affichage pour obtenir l'affichage suivant :
   Robot ZD0000154784 – Aspirateur robot
   Statut : EN SERVICE
   Orientation : OUEST
   Position : [abs=5 ; ord=4]
   Marque : Dyson
   Puissance : 7500W

8. BONUS : prévoir le parcours de l'aspirateur robot dans une pièce. L'aspirateur part de la position 0, 0. La méthode parcours prend en paramètre la largeur et la longueur de la pièce. Vous devez faire avancer et tourner l'aspirateur dans toute la pièce, d'en haut à gauche jusqu'à en bas à droite. S'il dépasse sa distance max parcourue, il doit s'arrêter.
   Exemple : pour une pièce de 10*15
   Sens du parcours :

x	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->
<-	<-	<-	<-	<-	<-	<-	<-	<-	<-
->	->	->	->	->	->	->	->	->	->

```
Dans la console : * là où il est passé, - sinon
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

```
----******
----------
----------
----------
```

Solution

Des propositions de solution pour ce TP sont placées dans les éléments en téléchargement liés à ce module.