Programmation orientée objet (POO)

Table des matières

1.) Définitions 2
1.1) La POO 2
1.2) Classe 2
1.3) Méthode 2
1.4) Attribut 2
1.5) Instance 3
2) Exemple 4
3) Exercice 5

1.) Définitions

1.1) La POO

La programmation orientée objet (POO) est un paradigme de programmation qui organise le code en "objets" dont on peut créer plusieurs instances et qui possèdent des caractéristiques particulières.

1.2) Classe

Une classe est un plan pour créer des objets, définissant leurs attributs et méthodes.

```
1 class MaClasse:
2     #Ici les carctéristiques de ma classe
```

1.3) Méthode

Une méthode est une fonction définie dans une classe et qui est spécifique à cette classe. En terminale, on voit 2 types de méthodes :

Les méthodes "classiques " :

```
1 class MaClasse:
2     def maMéthode(self):
3       #Ici les actions à effectuer lors de l'appel de la méthode
4       pass
```

Les méthodes "spéciales " :

```
1 class MaClasse:
2     def __init__(self): #Méthode à connaître !
3       #Ici les instructions qui s'exécuteront lors de la création d'une
  instance de MaClasse
4       pass
```

Une méthode spéciale est une méthode dont le nom commence et se termine par "__". Il en existe beaucoup, mais seule "__init__" est au programme de terminale. "__init__" est appelée lors de la création d'une instance de la classe.

1.4) Attribut

Un attribut est une variable définie dans une classe et spécifique à cette classe.

```
1 class MaClasse:
2     attribut1 = 0
```

On peut aussi définir un attribut dans une méthode :

```
1 class MaClasse:
2     def __init__(self):
3       self.attribut1 = 0
4     def maMethode(sefl):
5       self.attribut2 = 0
```

1.5) Instance

Une instance est un objet créé à partir d'une classe. Elle représente une occurrence spécifique de cette classe, avec ses propres valeurs pour les attributs définis dans la classe.

```
1 class MaClasse:
2     def __init__(self):
3       self.attribut1 = 0
4
5 instance = MaClasse()
6 instance.attribut1 #Vaut 0
```

2) Exemple

```python
class Voiture:
    def __init__(self, proprietaire):
      self.prix = 5500
      self.poids = 1420
      self.marque = "Peugeot"
      self.proprietaire = proprietaire
    def augmenter_prix(self,valeur):
      self.prix += valeur
    def vendre(self,acheteur):
      self.proprietaire = acheteur

voiture = Voiture("Alain") #Création d'une instance de Voiture()

voiture.poids #Renvoie 1420
voiture.marque #Renvoie "Peugeot"

voiture.prix #Renvoie 5500
voiture.augmenter_prix(200) #Appel de la méthode augmenter_prix()
voiture.prix #Renvoie 5700

voiture.proprietaire #Renvoie "Alain"
voiture.vendre("Paul")
voiture.proprietaire #Renvoie "Paul"
```

3) Exercice

1. Créer une classe "Point" avec un attribut "coordonnees" qui est un tuple de la forme (x,y).

2. Créer une classe "Rectangle" avec trois attributs :

- La largeur "largeur" du rectangle
- La hauteur "hauteur" du rectangle
- La position "position" du coin inférieur gauche du rectangle. "position" est une instance de la classe "Point".

3. Ajouter une méthode qui calcule et renvoie l'aire du rectangle et une méthode qui calcule et renvoie le périmètre du rectangle (méthodes à ajouter à la classe "Rectangle").

4. Ajouter une méthode qui détermine si un point (instance de "Point") est dans le rectangle. Cette méthode renvoie True si le point est dans le rectangle et False sinon. (méthode à ajouter à la classe "Rectangle").

➔ Voir page suivante pour la correction

Si cet exercice est réussi, alors vous répondez à toutes les attentes du programme de terminale concernant la programmation orientée objet !

Correction :

```python
class Point:
  def __init__(self,x,y):
    self.coordonnees = (x,y)

class Rectangle:
  def __init__(self,largeur,hauteur,position):
    self.largeur = largeur
    self.hauteur = hauteur
    self.position = position
  def aire(self):
    return self.largeur * self.hauteur
  def perimetre(self):
    return (2 * self.hauteur) + (2 * self.largeur)
  def est_dedans(self,point):
    x_bord_droit = self.position.coordonnées[0] + self.largeur
    x_bord_gauche = self.position.coordonnées[0]
    y_bord_haut = self.position.coordonnées[1] + self.hauteur
    y_bord_bas = self.position.coordonnées[1]
    if x_bord_gauche <= point.coordonnees[0] <= x_bord_droit:
      if y_bord_bas <= point.coordonnees[1] <= y_bord_haut:
        return True
      else:
        return False
    else:
      return False
```