

Exercices - Programmation Orientée Objet (POO)

Exercice 1 QCM (Attention plusieurs bonnes réponses possibles).

1. Une classe :
 - (a) est un objet
 - (b) est une fonction particulière
 - (c) permet de déclarer des propriétés communes à un ensemble d'objets
 - (d) ne peut pas être définie sans constructeur
2. Création du constructeur, en Python, on utilise le mot-clé :
 - (a) init
 - (b) new
 - (c) class
 - (d) str
3. Le constructeur :
 - (a) doit posséder au moins un argument
 - (b) doit posséder plusieurs arguments
 - (c) ne possède pas d'argument
 - (d) permet d'initialiser les attributs
4. Une méthode :
 - (a) permet de caractériser le comportement d'un objet
 - (b) est une fonction contenue dans une classe
 - (c) peut être appliquée à n'importe quel objet quelque soit sa classe
 - (d) peut modifier les attributs d'un objet
5. Un attribut :
 - (a) représente une caractéristique particulière d'un objet.
 - (b) peut être modifié à l'intérieur d'une classe.
 - (c) n'est pas déclaré dans le constructeur.
 - (d) peut être modifié à l'extérieur d'une classe.
 - (e) doit obligatoirement être initialisé lors de la création de l'objet.
6. Le mot clé `self` :
 - (a) est utilisé pour accéder aux attributs et méthodes d'un objet instancié dans un programme principal.
 - (b) représente un objet temporaire en interne de la classe.
 - (c) est un objet que l'on doit instancier à l'aide du constructeur de la classe.
7. Instancier un objet consiste à :
 - (a) initialiser les attributs de la classe.
 - (b) créer un objet dans un programme principal.
 - (c) appeler le constructeur de la classe.

Exercice 2 On suppose écrite la classe `Carte` dont on vous donne les en-têtes de méthodes. Écrire (sur feuille) un programme principal qui va :

1. Créer la carte Valet de COEUR que l'on nommera `c1`.
2. Afficher le nom, la valeur et la couleur de `c1`.
3. Créer la carte As de PIQUE que l'on nommera `c2`.
4. Afficher le nom, la valeur et la couleur de `c2`.
5. Modifier le nom de la carte `c2` en Roi et afficher le nom, la valeur et la couleur de `c2`.
6. Créer la carte 8 de TREFLE que l'on nommera `c3`.
7. Comparer les cartes `c1` et `c2` puis `c1` et `c3`.

```
class Carte:
    def __init__(self, nom, couleur):
        # Affectation de l'attribut nom et de l'attribut couleur
        couleur = ('CARREAU', 'COEUR', 'TREFLE', 'PIQUE')
        noms = ['2', '3', '4', '5', '6', '7', '8', '9', '10', 'Valet', 'Dame', 'Roi', 'As']
        valeurs = {'2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9, '10': 10, 'Valet': 11, 'Dame': 12, 'Roi': 13, 'As': 14}

    def setNom(self, nom):
        # Mutateur de l'attribut nom (de la liste noms)

    def getNom(self):
        # renvoie le nom de la carte (de la liste noms): Accesseur

    def getCouleur(self):
        # renvoie la couleur de la carte (de la liste couleur): Accesseur

    def getValeur(self):
        # renvoie la valeur de la carte (du dictionnaire valeurs) : Accesseur

    def egalite(self, carte):
        ''' Renvoie True si les cartes self et carte ont meme valeur, False sinon
            carte: Objet de type Carte
        '''

    def estSuperieureA(self, carte):
        ''' Renvoie True si la valeur de self est superieure a celle de carte, False sinon
            carte: Objet de type Carte
        '''

    def estInferieureA(self, carte):
        ''' Renvoie True si la valeur de self est inferieure a celle de carte, False sinon
            carte: Objet de type Carte
        '''
```

Exercice 3 On suppose écrites les classes `Piece` et `Appartement`, dont on vous donne les en-têtes de méthodes :

```
class Piece:
    # nom est une string et surface est un float
    def __init__(self, nom, surface):
        self.nom=nom
        self.surface=surface

    # Accesseurs: retournent les attributs d'un objet de cette classe
    def getSurface(self):
        return self.surface

    def getNom(self):
        return self.nom

    # Mutateur
    def setSurface(self,s): # s est un float,
        ...

class Appartement:
    # nom est une string
    def __init__(self,nom):
        # L'objet est une liste de pieces (objets issus de la classe Piece)
        self.listeDePieces=[]
        self.nom=nom

    def getNom(self):
        # Accesseurs:
        return self.nom

    # pour ajouter une piece de classe Piece
    def ajouter(self,piece):
        ...

    # pour avoir le nombre de pieces de l'appartement
    def nbPieces(self):
        ...

    # retourne la surface totale de l'appartement (un float)
    def SurfaceTotale(self):
        ...

    # retourne la liste des pieces avec les surfaces
    def getListePieces(self): # sous forme d'une liste de tuples
        ...
```

1. Créer un tableau qui classe les méthodes de ces deux classes selon leur type : constructeur, accesseur, mutateur ou autre.
2. Écrire (sur feuille) un programme principal utilisant ces deux classes qui va :
 - (a) créer une pièce « chambre1 » , de surface 20 m² et une pièce « chambre2 » », de surface 15 m² ,
 - (b) créer une pièce « séjour » », de surface 25 m² et une pièce « sdb » », de surface 10 m² ,
 - (c) créer une pièce « cuisine » », de surface 12 m² ,
 - (d) créer un appartement « appart205 » qui contiendra toutes les pièces créées,
 - (e) afficher la surface totale de l'appartement créé.
 - (f) afficher la liste des pièces et surfaces de l'appartement créé.

Exercice 4 On définit ci-dessous la classe voiture avec de nombreuses erreurs.

```
classe voiture:
    def init(marque, couleur = 'blanc', vol_essence):
        self.marque = ''
        self.couleur = couleur
        self.vol_essence = vol_essence

    def demarrer():
        if vol_essence > 0:
            print('on démarre')
        else:
            print('plus d\'essence')

    def rouler(nb_km, vol_essence):
        vol_essence -= 6 * nb_km / 100
        if vol_essence < 0:
            print('plus d\'essence')

    def ajoute_essence(vol_essence, volume):
        vol_essence += volume
```

1. Corrigez les erreurs.
2. "Ma voiture" est une Volkswagen noire ayant 26 litres d'essence. Comment instancier l'objet `ma_voiture` de la classe `Voiture` qui représente "Ma voiture" ?
3. Comment afficher dans la couleur de `ma_voiture` ? Peut-on afficher toutes ses caractéristiques avec la fonction `print` ?
4. On veut ajouter 15 litres d'essence à `ma_voiture`, quelle instruction faut-il écrire ?
5. Quelle instruction faut-il écrire pour connaître le volume d'essence restant si on a roulé pendant 160 km ?

Exercice 5

On souhaite créer une classe `Point` permettant de construire des objets de type `Point` définis par une abscisse et une ordonnée.

1. Créez une classe `Point` en Python et définissez le constructeur pour déclarer les attributs `x` (abscisse) et `y` (ordonnée) d'un objet `Point`.
2. Ecrivez les instructions permettant de créer deux instances `p1` et `p2` de cette classe ayant pour coordonnées respectives (2 ; 3) et (-1 ; 4).
3. Quelles instructions permettent d'accéder aux attributs de l'objet `p1` ?
4. Le point `p1` a changé de coordonnées qui sont maintenant (3, 4). Modifiez les attributs du point `p1`, en utilisant la notation pointée, afin qu'elles correspondent à ce changement.
5. On souhaite également pouvoir translater un point et vérifier si deux points sont égaux. Il est donc nécessaire d'ajouter des méthodes à notre classe.
Ecrivez les deux méthodes `translater` et `egal` de la classe `Point`.

Exercice 6 Ecrire une classe `Player` qui :

- ne prendra aucun argument lors de son instantiation.
- affectera à chaque objet créé un attribut `energie` valant 3 par défaut. affectera à chaque objet créé un attribut `en_vie` valant `True` par défaut.
- fournira à chaque objet une méthode `blessure()` qui diminue l'attribut `energie` de 1.
- fournira à chaque objet une méthode `soin()` qui augmente l'attribut `energie` de 1. si l'attribut `energie` passe à 0, l'attribut `en_vie` doit passer à `False` et ne doit plus pouvoir évoluer.

Exemple d'utilisation de la classe :

```
>>> mario = Player()
>>> mario.energie
3
>>> mario.soin()
>>> mario.energie
4
>>> mario.blessure()
>>> mario.blessure()
>>> mario.blessure()
>>> mario.alive
True
>>> mario.blessure()
>>> mario.alive
False
>>> mario.soin()
>>> mario.alive
False
>>> mario.energie
0
```



Source : www.math93.com