Corrections 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 possèder au moins un argument**
   (b) doit possèder 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'intéreur d'une classe.**
   (c) n'est pas déclaré dans le constructeur.
   (d) **peut être modifié à l'exterieur 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 innstancié 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.

```
c1 = Carte('Valet','COEUR')
print("nom : ",c1.getNom())
print("valeur : ",c1.getValeur())
print("couleur : ",c1.getCouleur())

c2 = Carte('As','PIQUE')
print("nom : ",c2.getNom())
print("valeur : ",c2.getValeur())
print("couleur : ",c2.getCouleur())

c2.setNom('Roi')
print("nom : ",c2.getNom())
print("valeur : ",c2.getValeur())
print("couleur : ",c2.getCouleur())

c3 = Carte('8','TREFLE')
print("nom : ",c3.getNom())
print("valeur : ",c3.getValeur())
print("couleur : ",c3.getCouleur())

#Comparaison c1 et c3
if c1.estSuperieureA(c3):
        print("c1 > c3)
elseif c1.estInferieureA(c3):
        print("c1 < c3")
else:
        print("c1=c3")

#Comparaison c3 et c2
if c3.estSuperieureA(c2):
        print("c3 > c2)
elseif c3.estInferieureA(c2):
        print("c3 < c2")
else:
        print("c3=c2")

#Comparaison c1 et c2
if c1.estSuperieureA(c2):
        print("c1 > c2)
elseif c1.estInferieureA(c2):
        print("c1 < c2")
else:
        print("c1=c2")
```

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.

Classe	`Piece`	`Appartement`
Constructeur	`__init__(self,nom,surface)`	`__init__(self,nom)`
Accesseur	`getSurface(self)`, `getNom(self)`	`getNom(self)`, `getListePieces(self)`
Mutateur	`setSurface(self,s)`	`ajouter(self,piece)`
Autre		`nbPieces(self)`, `SurfaceTotale(self)`

2. Écrire (sur feuille) un programme principal utilisant ces deux classes qui va :
   (a) créer une pièce « chambre1 » , de surface 20 m 2 et une pièce « chambre2 » », de surface 15 m 2 ,

(b) créer une pièce « séjour », de surface 25 m 2 et une pièce « sdb », de surface 10 m 2 ,

(c) créer une pièce « cuisine », de surface 12 m 2 ,

(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.

```
sejour = Piece('Sejour',25)
sdb = Piece('sdb',10)
cuisine = Piece('cuisine',12)

appart205 = Appartement('appart205')
appart205.ajouter(sejour)
appart205.ajouter(sdb)
appart205.ajouter(cuisine)

print("Surface total : ',appart205.SurfaceTotale())
for p in appart205.getListePieces():
        print(p.getNom(), " - ", p.getSurface())
```

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

```
class Voiture:
  def __init__(self,marque, couleur = 'blanc', vol_essence):
    self.marque = marque
    self.couleur = couleur
    self.vol_essence = vol_essence

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

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

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

1. Corrigez les erreurs.
2. "Ma voiture" est une Volskwagen noire ayant 26 litres d'essence. Comment instancier l'objet `ma_voiture` de la classe `Voiture` qui représente "Ma voiture" ?
   `ma_voiture = Voiture('Volskwagen','noire',26)`
3. Comment afficher dans la couleur de `ma_voiture` ? Peut-on afficher toutes ses caractèristiques avec la fonction `print` ?
   ```
   print(ma_voiture.couleur)
   print(ma_voiture.marque)
   print(ma_voiture.vol_essence)
   ```
4. On veut ajouter 15 litres d'essence à `ma_voiture`, quelle instruction faut-il écrire ?
   `ma_voiture.ajoute_essence(15)`

5. Quelle instruction faut-il écrire pour connaîte le volume d'essence restant si on a roulé pendant 160 km ?
```
ma_voiture.rouler(160)
print(ma_voiture.vol_essence)
```

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`.

```
class Point:
  def __init__(self,x, y):
    self.abs = x
    self.ord = y
```

2. Ecrivez les instructions permettant de créer deux instances p1 et p2 de cette classe ayant pour coordonnées respectives (2 ; 3) et (-1 ; 4).

```
p1 = Point(2, 3)
p2 = Point(-1, 4)
```

3. Quelles instructions permettent d'accéder aux attributs de l'objet p1 ?

```
p1.abs
p1.ord
```

4. Le point p1 a changé de coordonnées qui sont maintenant (3, 4). Modifiez les attributs du point p1, en utilsant la notation pointée, afin qu'elles correspondent à ce changement.

```
p1.abs = 3
p1.ord = 4
```

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`.

```
class Point:
  def __init__(self,x, y):
    self.abs = x
    self.ord = y

  def translater(self, dx,dy):
    self.abs +=dx
    self.ord +=dy

  def egal(self,p):
    return (self.abs ==p.abs) and (self.ord == p.ord)
```

Exercice 6 Ecrire une classe Player qui :

- ne prendra aucun argument lors de son instanciation.
- 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.

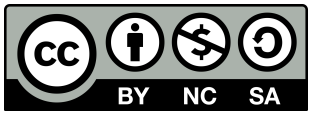
Exemple d'utilisation de la classe :

```
>>> 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
```

```
class Player:
  def __init__(self):
    self.energie = 3
    self.alive = True

  def blessure(self):
    self.energie -= 1
    if self.energie <= 0:
      self.alive = False

  def soin(self):
    if self.energie > 0:
      self.energie += 1
```



Source : www.math93.com