Exercice POO
Correction

Christophe Viroulaud

Terminale - NSI

Lang 03

Sommaire

Exercice 1

```
1  class Livre:
2
3      def __init__(self, t: str, a: str, p: float):
4          self.titre = t
5          self.auteur = a
6          self.prix = p
7
8      def afficher(self) -> str:
9          return self.titre + " est écrit par " + \
10             self.auteur + " et coûte " + \
11             str(self.prix) + "."
12
13 livre1 = Livre("Le guide du voyageur galactique",
       "Douglas Adams", 6.42)
14 print(livre1.afficher())
```

Sommaire

Exercice 2


```python
class Menu:
    def __init__(self):
        self.entree = None
        self.plat = None
        self.accompagnement = None
        self.dessert = None

    def prendre_entree(self, choix: str) -> None:
        self.entree = choix

    def prendre_plat(self, choix: str) -> None:
        self.plat = choix

    def prendre_accompagnement(self, choix: str) -> None:
        self.accompagnement = choix

    def prendre_dessert(self, choix: str) -> None:
        self.dessert = choix
```

```python
def calculer_note(self) -> int:
    somme = 0
    if self.entree is not None:
        somme = somme + 8
    if self.plat is not None:
        somme = somme + 10
    if self.accompagnement is not None:
        somme = somme + 3
    if self.dessert is not None:
        somme = somme + 8
    return somme
```

Code 1 – Méthode supplémentaire

```
1 repas = Menu()
2 repas.prendre_plat("steak")
3 repas.prendre_accompagnement("frites")
4 repas.prendre_dessert("glace")
5 print(repas.calculer_note())
```

Code 2 – Tester la classe

Sommaire

Exercice 3

```python
class Rectangle:

    def __init__(self, L: float, l: float):
        self.lon = L
        self.lar = l

    def perimetre(self) -> float:
        return (self.lon + self.lar)*2

    def aire(self) -> float:
        return round(self.lon * self.lar, 2)

    def est_carre(self) -> bool:
        return self.lon == self.lar
```

```
1 rect = Rectangle(5.3, 2.8)
2 assert rect.aire() == 14.84
3 assert rect.est_carre() == False
```

Code 3 – Tester la classe

Sommaire

Exercice 4

Joueur
tenue parachute trainée énergie (100) bouclier (100) sac à dos (3 max)
ramasser un objet subir une attaque

```python
class Combattant:

    def __init__(self, te: str, pa: str, tr: str):
        self.tenue = te
        self.parachute = pa
        self.trainee = tr
        self.energie = 100
        self.bouclier = 100
        self.sacados = []
```

Code 4 – Initialisation

```python
def ramasser_objet(self, objet: str) -> str:
    """
    met l'objet dans le sac à dos si possible

    Args:
        objet (str): l'objet

    Returns:
        str: message
    """
    if len(self.sacados) < 3:
        self.sacados.append(objet)
        return "Objet ajouté."
    else:
        return "Le sac est plein."
```

Code 5 – ramasser

```python
def subir_attaque(self, degat: int) -> None:
    """
    modifie les PV et le bouclier du joueur
    en fonction des dégâts

    Args:
        degat (int): la valeur du dégât
    """
    if degat <= self.bouclier:
        self.bouclier = self.bouclier - degat
    else:
        # les dégâts finissent le bouclier...
        if self.bouclier > 0:
            degat = degat - self.bouclier
            self.bouclier = 0
        # ...et commencent les PV
        self.energie = self.energie - degat
```

Code 6 – subir une attaque

```
1 joueur = Combattant("jean","glider","fuméee")
2 joueur.ramasser_objet("gun")
3 joueur.subir_attaque(50)
```

Code 7

Sommaire

Exercice 5


```python
class Date:
    def __init__(self, j: int, m: int, a: int):
        self.jour = j
        self.mois = m
        self.annee = a
```

Code 8 – Initialisation

```
1 def afficher(self) -> str:
2     nom_mois = ["janvier", "février", "mars", "avril",
      "mai", "juin", "juillet", "août", "septembre", "
    octobre", "novembre", "décembre"]
3     return str(self.jour) + "/" + \
4             str(nom_mois[self.mois - 1]) + "/" + \
5             str(self.annee)
```

Code 9 – Afficher

```python
def est_avant(self, d) -> bool:
    # and est prioritaire devant or
    return self.annee < d.annee or \
        self.annee == d.annee and \
        (self.mois < d.mois or
         self.mois == d.mois and
         self.jour < d.jour)
```

Code 10 – Comparer

Remarque

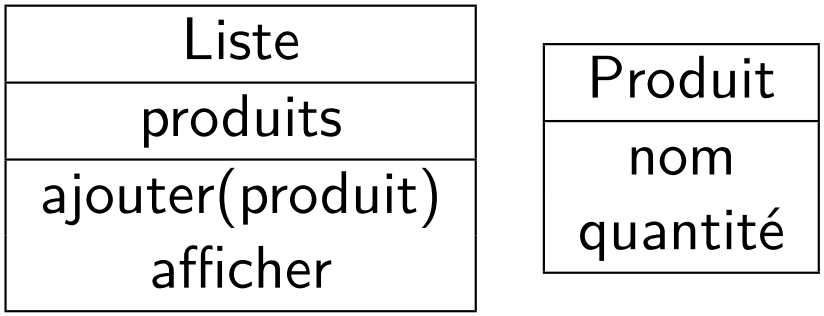
En toute rigueur, il est préférable d'utiliser les mutateurs et accesseurs pour accéder aux attributs d'un objet.

```
1 d = Date(8, 12, 1977)
2 assert d.afficher() == "8 / décembre / 1977"
3 assert d.est_avant(Date(9, 12, 1977)) == True
```

Code 11 – Tester la classe

Sommaire

Exercice 6



```python
class Produit:
    def __init__(self, n: str, q: int):
        self.nom = n
        self.quantite = q

    def get_nom(self) -> str:
        return self.nom

    def set_nom(self, n: str) -> None:
        self.nom = n

    def get_quantite(self) -> str:
        return self.quantite

    def set_nom(self, q: str) -> None:
        self.quantite = q
```

```
1  class Liste:
2      def __init__(self):
3          self.produits = []
4
5      def ajouter(self, p: Produit) -> None:
6          self.produits.append(p)
7
8      def afficher(self) -> str:
9          texte = "Ma liste: \n"
10         for p in self.produits:
11             texte = texte + p.nom + " " + \
12                 str(p.get_quantite()) + "\n"
13         return texte
```

```
courses = Liste()
courses.ajouter(Produit("raviolis", 3))
courses.ajouter(Produit("glace", 10))
print(courses.afficher())
```

Code 12 – Tester les classes