Exercice POO
Correction

Christophe Viroulaud

Terminale - NSI

Lang 06

Sommaire

Exercice 1

```python
class Personne:
    def __init__(self, nom, age):
        self.nom = nom
        self.age = age

    def se_presenter(self) -> str:
        return "Je m'appelle " + self.nom + \
        " et j'ai " + str(self.age) + " ans."

# principal
bob = Personne("Bob", 20)
jane = Personne("Jane", 18)

print(bob.se_presenter())
print(jane.se_presenter())
```

Sommaire

Exercice 2

```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 1 – Tester la classe

Sommaire

Exercice 3


```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 2 – 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 3 – Tester la classe

Sommaire

Exercice 4

```
1  class Livre:
2      def __init__(self, t: str, a: str, p: float):
3          self.titre = t
4          self.auteur = a
5          self.prix = p
```

```python
def get_titre(self) -> str:
    return self.titre

def get_auteur(self) -> str:
    return self.auteur

def get_prix(self) -> float:
    return self.prix

def set_titre(self, nouv: str) -> None:
    self.titre = nouv

def set_auteur(self, nouv: str) -> None:
    self.auteur = nouv

def set_prix(self, nouv: float) -> None:
    self.prix = nouv
```

```
1 def afficher(self) -> str:
2         return (
3             self.titre
4             + " est écrit par "
5             + self.auteur
6             + " et coûte "
7             + str(self.prix)
8             + "."
9         )
```

```
1  class Bibliotheque:
2      def __init__(self):
3          self.etagere = []
4
5      def get_etagere(self) -> list:
6          return self.etagere
7
8      def ajouter_livre(self, livre: Livre) -> None:
9          self.etagere.append(livre)
10
11     def afficher_biblio(self) -> str:
12         res = ""
13         for livre in self.etagere:
14             res = res + livre.afficher() + "\n"
15         return res
```

```
1 livre1 = Livre("Le guide du voyageur galactique",
      "Douglas Adamss", 6.42)
2 print(livre1.afficher())
3 livre2 = Livre("Germinal", "Émile Zola", "7.50")
4 biblio = Bibliotheque()
5 biblio.ajouter_livre(livre1)
6 biblio.ajouter_livre(livre2)
7 print(biblio.get_etagere()[0].afficher())
8 print(biblio.afficher_biblio())
```

Code 4 – Programme principal

Sommaire

Exercice 5

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

Choix effectués

- `ramasser_objet` renverra une chaîne de caractères parmi :
  - "Objet ajouté."
  - "Le sac est plein."
- `subir_attaque` commencera par consommer le bouclier puis l'énergie.

```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 5 – 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 6 – 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 7 – subir une attaque

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

Code 8 – Instanciation

Sommaire

Exercice 6


```
1 class Date:
2     def __init__(self, j: int, m: int, a: int):
3         self.jour = j
4         self.mois = m
5         self.annee = a
```

Code 9 – 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             nom_mois[self.mois - 1] + " / " + \
5             str(self.annee)
```

Code 10 – 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 11 – 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 12 – Tester la classe