POO: TD

Thème 1 - Structure de données

05

TD : Programmation Orientée Objet (POO)

1. Exercice 1 : Class Eleve

Exercice 1. Exercice 1

- 1. Écrire une classe Eleve qui contiendra les attributs nom, classe et note.
- 2. Instancier trois élèves de cette classe.
- 3. Écrire une fonction compare(eleve1, eleve2) qui renvoie le nom de l'élève ayant la meilleure note.

Exemple d'utilisation de la classe

```
Script Python

>>> riri = Eleve("Henri", "TG2", 12)
>>> fifi = Eleve("Philippe", "TG6", 15)
>>> loulou = Eleve("Louis", "TG1", 8)
>>> compare(riri, fifi)
'Philippe'
```

```
class Eleve:
    def __init__(self, nom, classe, note):
        self.nom = nom
        self.classe = classe
        self.note = note

def __repr__(self):
        return self.nom + ' ' + self.classe + ' ' + str(self.note)

def compare(self, other):
    if self.note > other.note:
        return self.nom
```

```
else:
    return other.nom

Socript Python

riri = Eleve("Henri", "TG2", 12)
fifi = Eleve("Philippe", "TG6", 15)
loulou = Eleve("Louis", "TG1", 8)
riri.compare(fifi)

Texte

'Philippe'

Socript Python

riri

Texte

Henri TG2 12
```

2. Exercice 2 : Class Player

Exercice 2. Exercice 2

Écrire 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 alive 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 alive doit passer à False et ne doit plus pouvoir évoluer.

Exemple d'utilisation de la classe

```
Script Python

>>> 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.alive
```

```
% Script Python
```

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

def soin(self, h = 1):
        if self.alive: self.hp += h

def blessure(self, damage = 1):
        self.hp -= damage
        if self.hp <= 0:
            self.alive = False
            self.hp = 0</pre>
```

% Script Python

```
mario = Player()
mario.hp
mario.blessure()
mario.hp
mario.blessure(3)
mario.hp
mario.soin(5)
mario.hp
```

☐ Texte

0

Exercice 3. Exercice 3

Créer une classe CompteBancaire dont la méthode constructeur recevra en paramètres :

- un attribut titulaire stockant le nom du propriétaire.
- un attribut solde contenant le solde disponible sur le compte.

Cette classe contiendra deux méthodes retrait() et depot() qui permettront de retirer ou de déposer de l'argent sur le compte.

Exemple d'utilisation de la classe

Script Python >>> compteGL = CompteBancaire("G.Lassus", 1000) >>> compteGL.retrait(50) Vous avez retiré 50 euros Solde actuel du compte : 950 euros >>> compteGL.retrait(40000) Retrait impossible >>> compteGL.depot(10000000) Vous avez déposé 10000000 euros Solde actuel du compte : 10000950 euros

& Script Python

Exercice 3 : Class Pokemon

Exercice 4. Exercice 4 Pokemon et POO

3.1. Création d'un Pokemon

- On considère une classe Pokemon qui permet de créer des pokemons.
- Chaque Pokemon a les caractéristiques suivantes :
 - un nom
 - · un type
 - · une vitesse
 - · une attaque
 - un nombre de points de vie maximal

Question n° A - 1 : >En étudiant le code ci-dessous, modifier <u>la</u> ligne de code appropriée pour créer un Pokemon ayant les caractéristiques suivantes : * nom : Aqua * type : eau * vitesse : 35 * attaque : 75 * defense : 20 * points de vie maximal : 141

```
self.nom = nom
self.type = typ
self.vitesse = vitesse
self.attaque = attaque
self.defense = defense
self.pvMax = pvMax
self.pvActuels = pvMax

aqua = Pokemon('Aqua','eau','35,75,20,141)
assert aqua.nom == "Aqua"
assert aqua.type == "eau"
assert aqua.vitesse == 35
assert aqua.attaque == 75
assert aqua.defense == 20
assert aqua.pvActuels == 141
```

Question n° A - 2: > Modifier l'attribut defense du Pokemon aqua pour qu'il prenne la valeur 15.

```
#Ecrire le code ici !!!
assert aqua.defense == 15
```

3.2. Méthode spéciale __str__

Question n° B - 1:

Surcharger la méthode __str__(self) pour qu'elle affiche l'ensemble des attributs du Pokemon

```
#Reprendre le code de la classe "Pokemon" et compléter !!!
from random import *
class Pokemon:
    def __init__(self, nom : str = "Anonyme",
                 typ = choice(["eau", "air", "terre", "feu"]),
                 vitesse = randint(1,51),
                 attaque = randint(51, 100),
                 defense = randint(1,50),
                 pvMax = randint(101, 150)):
        self.nom = nom
        self.type = typ
        self.vitesse = vitesse
        self.attaque = attaque
        self.defense = defense
        self.pvMax = pvMax
        self.pvActuels = pvMax
    def __str__(self):
        return self.nom + ', ' + self.type + ', ' + str(self.vitesse) + ', ' + str(self.attaque) +
   ' + str(self.defense) + ', ' + str(self.pvMax) + ', ' + str(self.pvActuels)
aqua = Pokemon('Aqua', 'eau', 35, 75, 20, 141)
print(aqua)
```

↑ Texte

```
Aqua, eau, 35, 75, 20, 141, 141
```

Question n° B - 2: > Créer une méthode etre_ko(self) qui renvoie True si le Pokemon a 0 point de vie ou moins et False sinon.

& Script Python

```
#Reprendre le code de la classe "Pokemon" et compléter !!!
from random import *
class Pokemon:
    def __init__(self, nom : str = "Anonyme",
                 typ = choice(["eau", "air", "terre", "feu"]),
                 vitesse = randint(1,51),
                 attaque = randint(51, 100),
                 defense = randint(1,50),
                 pvMax = randint(101, 150)):
        self.nom = nom
        self.type = typ
        self.vitesse = vitesse
        self.attaque = attaque
        self.defense = defense
        self.pvMax = pvMax
        self.pvActuels = pvMax
    def etre_ko(self):
        if self.pvActuels <= 0:</pre>
            return True
        else:
            return False
aqua = Pokemon('Aqua', 'eau', 35, 75, 20, 141)
assert aqua.etre_ko() == False
aqua.pvActuels = 0
assert aqua.etre_ko() == True
aqua.pvActuels = -7
assert aqua.etre_ko() == True
```

Question n° B - 3: > Créer une méthode se_reposer(self) qui redonne tous ses points de vie au Pokemon.

```
self.nom = nom
       self.type = typ
       self.vitesse = vitesse
       self.attaque = attaque
       self.defense = defense
       self.pvMax = pvMax
       self.pvActuels = pvMax
   def __str__(self):
       return self.nom + ', ' + self.type + ', ' + str(self.vitesse) + ', ' + str(self.attaque) +
', ' + str(self.defense) + ', ' + str(self.pvMax) + ', ' + str(self.pvActuels)
   def etre_ko(self):
       if self.pvActuels <= 0:</pre>
           return True
       else:
           return False
   def se_reposer(self):
       self.pvActuels = self.pvMax
```

3.3. Interaction avec un autre Pokemon

Un pokemon peut être de 4 types : * Eau * Feu * Air * Terre

On sait que : * l'eau domine le feu * le feu domine l'air * l'air domine la terre * la terre domine l'eau

Question n° C - 1: > Créer une méthode dominer(self, adversaire) prenant en paramètre une autre instance de la classe **Pokemon**, qui renvoie **True** si le Pokemon a un type qui domine celui du Pokemon donné en paramètre et **False** sinon.

```
#Reprendre le code la classe "Pokemon" et compléter !!!
from random import *
class Pokemon:
    def __init__(self, nom : str = "Anonyme",
                 typ = choice(["eau", "air", "terre", "feu"]),
                 vitesse = randint(1,51),
                 attaque = randint(51, 100),
                 defense = randint(1, 50),
                 pvMax = randint(101, 150):
        self.nom = nom
        self.type = typ
        self.vitesse = vitesse
        self.attaque = attaque
        self.defense = defense
        self.pvMax = pvMax
        self.pvActuels = pvMax
    def __str__(self):
        return self.nom + ', ' + self.type + ', ' + str(self.vitesse) + ', ' + str(self.attaque) +
   ' + str(self.defense) + ', ' + str(self.pvMax) + ', ' + str(self.pvActuels)
    def etre_ko(self):
        if self.pvActuels <= 0:</pre>
```

```
return True
        else:
            return False
    def se_reposer(self):
        self.pvActuels = self.pvMax
    def dominer(self, other):
        if self.type == 'eau':
            if other.type == 'feu': return True
           else: return False
        if self.type == 'feu':
            if other.type == 'air': return True
            else: return False
        if self.type == 'air':
           if other.type == 'terre': return True
           else: return False
        if self.type == 'terre':
            if other.type == 'eau': return True
            else: return False
aqua = Pokemon('Aqua', 'eau', 35, 75, 20, 141)
flamichou = Pokemon("Flamichou", "feu")
assert aqua.dominer(flamichou) == True
rocamon = Pokemon("Rocamon", "terre")
assert aqua.dominer(rocamon) == False
```

Lorsque qu'un Pokemon attaque un autre Pokemon, il inflige des dégâts comptabilisés de la façon suivante : * la valeur des dégâts est égale à la différence entre l'attaque du Pokemon attaquant et la défense de l'autre Pokemon (si la différence est nulle ou négative, alors les dégâts seront égaux à 1) * si le type du Pokemon attaquant domine celui de l'autre Pokemon, alors les dégâts sont multipliés par 2

Question n° C - 2 : > Créer une méthode attaquer(self, adversaire) prenant en paramètre une autre instance de la classe Pokemon et qui retire des points de vie à ce dernier.

```
& Script Python
#Reprendre le code la classe "Pokemon" et compléter !!!
from random import *
class Pokemon:
    def __init__(self, nom : str = "Anonyme",
                 typ = choice(["eau", "air", "terre", "feu"]),
                 vitesse = randint(1,51),
                 attaque = randint(51, 100),
                 defense = randint(1,50),
                 pvMax = randint(101, 150)):
        self.nom = nom
        self.type = typ
        self.vitesse = vitesse
        self.attaque = attaque
        self.defense = defense
        self.pvMax = pvMax
        self.pvActuels = pvMax
    def __str__(self):
```

```
return self.nom + ', ' + self.type + ', ' + str(self.vitesse) + ', ' + str(self.attaque) +
', ' + str(self.defense) + ', ' + str(self.pvMax) + ', ' + str(self.pvActuels)
    def etre_ko(self):
        if self.pvActuels <= 0:</pre>
            return True
        else:
            return False
    def se_reposer(self):
        self.pvActuels = self.pvMax
    def dominer(self, other):
        if self.type == 'eau':
            if other.type == 'feu': return True
            else: return False
        if self.type == 'feu':
            if other.type == 'air': return True
            else: return False
        if self.type == 'air':
            if other.type == 'terre': return True
            else: return False
        if self.type == 'terre':
            if other.type == 'eau': return True
            else: return False
    def attaquer(self, other):
        damage = self.attaque - other.defense
        if damage <= 1: damage = 1</pre>
        if self.dominer(other):
            damage *= 2
        other.pvActuels -= damage
        if other.pvActuels <= 0: other.pvActuels = 0</pre>
aqua = Pokemon('Aqua', 'eau', 35, 75, 20, 141)
flamichou = Pokemon("Flamichou", "feu", 15, 63, 32, 118)
rocamon = Pokemon("Rocamon", "terre", 44, 95, 48, 101)
aqua.attaquer(flamichou)
assert flamichou.pvActuels == 32
aqua.attaquer(rocamon)
assert rocamon.pvActuels == 74
```

3.4. Combat de Pokemons

Lorsque deux Pokemons combattent l'un contre l'autre : * le premier à attaquer est celui ayant la vitesse la plus élevée * s'est ensuite à l'autre d'attaquer et ainsi de suite ... * le combat s'arrête dès qu'un des Pokemons est ko * le vainqueur est le Pokemon n'étant pas ko

Question n° D - 1: > Créer une fonction combattre(pokemon1, pokemon2) prenant en paramètres deux instances de la classe Pokemon et renvoyant le pokemon vainqueur du combat

```
def combattre(pokemon1, pokemon2):
   if pokemon1.vitesse > pokemon2.vitesse:
      p1 = pokemon1
      p2 = pokemon2
```

```
else:
        p1 = pokemon2
        p2 = pokemon1
    while pokemon1.etre_ko() == False and pokemon2.etre_ko == False:
        p1.attaquer(p2)
        print(pokemon1.nom + ': ' + pokemon1.pvActuels + 'hp' + ' | ' + pokemon2.nom + ': ' +
pokemon2.pvActuels + 'hp')
        p2.attaquer(p1)
        print(pokemon1.nom + ': ' + pokemon1.pvActuels + 'hp' + ' | ' + pokemon2.nom + ': ' +
pokemon2.pvActuels + 'hp')
    if pokemon1.etre_ko() == True:
       return pokemon1.nom
    else:
        return pokemon2.nom
vaingueur = combattre(agua, flamichou)
print(vainqueur)
```

☐ Texte

Flamichou

Question n° D - 2 : > Compléter la fonction combattre pour que les Pokemons se reposent avant d'engager le combat

```
#Reprendre le code de la fonction "Combattre" et le compléter
def combattre(pokemon1, pokemon2):
    pokemon1.se_reposer()
    pokemon2.se_reposer()
    if pokemon1.vitesse > pokemon2.vitesse:
        p1 = pokemon1
        p2 = pokemon2
    else:
        p1 = pokemon2
        p2 = pokemon1
    while pokemon1.etre_ko() == False and pokemon2.etre_ko() == False:
        print(pokemon1.nom + ': ' + str(pokemon1.pvActuels) + 'hp' + ' | ' + pokemon2.nom + ': ' +
str(pokemon2.pvActuels) + 'hp')
        p1.attaquer(p2)
        print('\t' + pokemon1.nom + ': ' + str(pokemon1.pvActuels) + 'hp' + ' | ' + pokemon2.nom +
': ' + str(pokemon2.pvActuels) + 'hp')
        p2.attaquer(p1)
        print('\t\t' + pokemon1.nom + ': ' + str(pokemon1.pvActuels) + 'hp' + ' | ' + pokemon2.nom +
': ' + str(pokemon2.pvActuels) + 'hp')
    if pokemon1.etre_ko() == True:
       return pokemon2.nom
    else:
       return pokemon1.nom
aqua = Pokemon('Aqua','eau',35,65,20,141)
flamichou = Pokemon("Flamichou", "feu", 15, 63, 32, 118)
rocamon = Pokemon("Rocamon", "terre", 20, 35, 48, 101)
vaingueur = combattre(agua, rocamon)
print(vainqueur)
```

☐ Texte

Aqua: 141hp | Rocamon: 101hp Aqua: 141hp | Rocamon: 84hp Aqua: 111hp | Rocamon: 84hp Aqua: 111hp | Rocamon: 84hp Aqua: 111hp | Rocamon: 67hp Aqua: 81hp | Rocamon: 67hp Aqua: 81hp | Rocamon: 67hp Aqua: 81hp | Rocamon: 50hp Aqua: 51hp | Rocamon: 50hp Aqua: 51hp | Rocamon: 50hp Aqua: 51hp | Rocamon: 33hp Aqua: 21hp | Rocamon: 33hp Aqua: 21hp | Rocamon: 33hp Aqua: 21hp | Rocamon: 16hp Aqua: Ohp | Rocamon: 16hp Rocamon