

□ **Exercice 1** : Une application pour un élevage de chien

10 points

On souhaite créer une application de gestion d'un élevage de chiens, un chien de cet élevage sera représenté par une instanciation de la classe **Chien** suivante dont la signification des attributs est indiqué en commentaire :

```

1  class Chien :
2
3      def __init__(self ,nom,race ,sexe ,naissance):
4          # le nom, la race, le sexe et l'année de naissance du chien
5          self.nom = nom
6          self.race = race
7          self.sexe = sexe
8          self.naissance = naissance
9
10     def __str__(self):
11         return f' {self.nom} ({self.sexe}) {self.race} né en {self.naissance}'

```

1. Quelle instruction permet de créer l'objet **medor** de la classe **Chien**, de race "caniche", femelle née en 2017 et nommé "Médor" ?
2. Quel sera l'affichage produit par **print(medor)** ?
3. Ecrire une méthode **get_race** (un *getter*) qui renvoie la race d'un objet de type **Chien**.
4. Ecrire une méthode **set_nom** (un *setter*) qui permet de modifier le nom d'un objet de type **Chien**.
5. On suppose qu'on a importé le module **time** de Python, l'expression **time.gmtime().tm_year** renvoie alors l'année en cours. Ecrire la méthode **age**, qui renvoie l'âge approximatif d'un chien (on considère qu'un chien né en 2017 a 4 ans en 2021 sans s'occuper du jour de naissance)

□ **Exercice 2** : Simulation d'un jeu de type *Pokemon*

10 points

On veut créer un jeu de type *Pokemon*, dans lesquels on fait combattre des créatures, chaque créature a :

- un nom
- un nombre de points de vie maximum
- une valeur d'attaque
- une armure qui diminue les dégâts subies
- un nombre de points de vie actuel

1. Ecrire une classe **Creature** représentant cette modélisation
2. Instancier cette classe de façon à créer les créatures suivantes :

	Nom	Vie	Attaque	Armure
toto	"Toto"	90	10	0.7
tartampion	"Tartampion"	40	70	0.5
untel	"Untel"	20	150	0.1

Chaque créature aura son nombre de points de vie maximum a sa création.

3. Ecrire la méthode spéciale **__str__** permettant d'afficher un objet de la classe **Creature** en donnant son nombre de points de vie actuel et son nombre de points de vie maximum.
4. Un objet de la classe **Creature** peut boire une potion pour récupérer 10 points de vie mais sans pouvoir dépasser son maximum. Ecrire la méthode **boit_potion** pour la classe **Creature**
5. Ecrire une méthode **subit_degats(d)** qui fait perdre **d** points de vie à un objet **Creature** mais sans tomber en dessous de 0.
6. Ecrire une méthode **attaque()** qui renvoie un nombre au hasard entre 1 et l'attribut **attaque** d'une créature.
7. Le principe d'un combat entre deux objets de type **Creature** est le suivant :
 - ① On tire au hasard celui commence puis chacun attaque à son tour
 - ② Celui qui attaque génère un nombre au hasard entre 1 et son attribut d'attaque (par exemple si Toto attaque, il tire au sort un nombre entre 1 et 10)
 - ③ Celui qui défend multiplie les dégâts reçus par son armure et soustrait le total de ses points de vie (par exemple si Toto défend et reçoit une attaque de 20, il multiplie par son armure 0.7 et perd $20 \times 0.7 = 14$ points de vie)
 - ④ Le combat prend fin lorsque l'un des deux participants atteint 0 PV.

Ecrire la méthode **combat(self,other)** entre deux créatures. On pourra afficher l'état des deux combattants durant le déroulement du combat.