

# Chapitre 1 : Programmation orientée objet

## TRAVAUX PRATIQUES

### I. TP1 : utiliser une classe existante

Utiliser la classe suivante :

```
#
class Eleve:
    #
    matiere1="Programmation"
    matiere2="Algorithmique"
    matiere3="Projet"
    #
    def __init__(self, pNom, pPrenom, pDate, pNote1, pNote2, pNote3) :
        #
        self.nom=pNom
        self.prenom=pPrenom
        self.date=pDate
        self.note_mat1=pNote1
        self.note_mat2=pNote2
        self.note_mat3=pNote3
```

#### 1) Ajouter de la documentation

Ajouter la documentation et visualiser la documentation de la classe Eleve. Si vous donnez cette classe à un autre programmeur, il faut qu'il puisse savoir comment l'utiliser sans avoir à en décortiquer le code...

#### 2) Créer trois nouveaux élèves ayant les caractéristiques suivantes

<b>Elève 1 :</b> <ul style="list-style-type: none"><li>• Nom: Dubois</li><li>• Prénom : Camille</li><li>• Date: 01/07/2003</li><li>• Programmation: 7</li><li>• Algorithmique : 14</li><li>• Projet : 11</li></ul>	<b>Elève 2 :</b> <ul style="list-style-type: none"><li>• Nom: Durand</li><li>• Prénom : Adrien</li><li>• Date: 01/11/2003</li><li>• Programmation: 13</li><li>• Algorithmique : 8</li><li>• Projet : 17</li></ul>	<b>Elève 3 : vous</b> <ul style="list-style-type: none"><li>• Nom:</li><li>• Prénom :</li><li>• Date:</li><li>• Programmation:</li><li>• Algorithmique :</li><li>• Projet :</li></ul>
--	---	---

#### 3) Ajouter une méthode moyenne

Il serait intéressant de pouvoir obtenir la moyenne de l'élève avec l'instruction `eleve1.moyenne()`. Pour cela, créer une fonction à l'intérieur de la classe Eleve qui renvoie la moyenne de l'élève

#### 4) Écrire une fonction `moy_matieres(liste_eleve)` (hors de la classe Eleve) qui prend en paramètre une liste et qui renvoie les moyennes par matières dans un dictionnaire. La liste sera ici constituée des trois élèves.

## II. TP2 : création d'un mini jeu

- 1) Dans un nouveau fichier MonPersonnage.py, créer la classe Personnage dont la documentation est la suivante:

```
class Personnage:
    """
    Personnage d'un jeu de type hack 'n slash

    Attributs d'instance:
        nom : chaine de caractères, nom du personnage
        pv : entier positif ou nul, points de vie du personnage
        degats : entier >0, dégat maximum du personnage
    Méthodes:
        init() constructeur de la classe Personnage
        attaque() : renvoie les dégâts faits à l'adversaire
        nombre aléatoire compris entre 1 et degats avec randint()
    """
```

- 2) Créer 2 objets perso1 et perso2 qui sont des instances de la classe Personnage et afficher leurs attributs.
- 3) Proposer une bataille entre ces deux personnages :

- perso1 attaque le premier, si perso2 a ses points de vie inférieurs ou égaux à 0, vous affichez : perso2.nom " est au tapis", sinon vous affichez : perso2.nom," a subi ",degats, " points de dégats et est à "perso2.pv," points de vie"

- puis perso2 attaque, si perso1 a ses points de vie inférieurs ou égaux à 0, vous affichez : perso1.nom est au tapis, sinon vous affichez : perso1.nom," a subi ",degats, " points de dégats et est à ",perso1.pv," points de vie"

Le jeu se termine lorsque l'un des joueurs est au tapis.

### III. TP3 : Mini projet POO avec attributs privés

Le but de ce mini projet est de réaliser de plusieurs façon une classe permettant d'afficher l'heure

#### 1) Première méthode

Créer une **classe** nommée **Heure1** avec un constructeur ayant comme paramètres **heureInit = 0**, **minuteInit = 0** et comme attributs privés: **\_\_heure**, **\_\_minute**.

Créer une méthode **getHeure()** qui renvoie **\_\_heure**.

Créer une méthode **getMinute()** qui renvoie **\_\_minute**.

Créer une méthode **setHeure()** ayant comme paramètre **nouvelleHeure** et change la valeur de **\_\_heure**.

Créer une méthode **setMinute()** ayant comme paramètre **nouvelleMinute** et change la valeur de **\_\_minute**.

Créer une méthode **incrémenter()** qui permet d'incrémenter **\_\_minute** de 1. Si **\_\_minute** est égal à 60 alors mettre **\_\_minute** à 0 et ajouter 1 à **\_\_heure** ; si **\_\_heure** est égal à 24 alors mettre **\_\_heure** à 0.

Créer une méthode **\_\_str\_\_()** permettant d'afficher au format *heure : minute*.

#### 2) Utilisation de la première classe

Créer une instance nommée **h1** de la classe **Heure1** avec les arguments 22 et 12.

Puis afficher l'heure avec la méthode **getHeure()** appliquée sur l'objet h1.

Puis afficher les minutes avec la méthode **getMinute()** appliquée sur l'objet h1.

Puis mettre **\_\_heure** à 23 et **\_\_minute** à 59 et afficher h1.

Puis **incrémenter** de 1 minute l'objet h1 et afficher h1.

#### 3) Seconde méthode

Créer une nouvelle classe nommée **Heure2** avec un constructeur ayant comme paramètres **heureInit = 0**, **minuteInit = 0** et comme attribut uniquement **\_\_minutes**.

Créer une méthode **getHeure()** qui renvoie la valeur des heures.

Créer une méthode **getMinute()** qui renvoie la valeur des minutes.

Créer une méthode **setHeure()** ayant comme paramètre **nouvelleHeure** et change la valeur de **\_\_minutes**.

Créer une méthode **setMinute()** ayant comme paramètre **nouvelleMinute** et change la valeur de **\_\_minutes**.

Créer une méthode **incrémenter()** qui permet d'incrémenter **\_\_minutes** de 1.

Créer une méthode **\_\_str\_\_()** permettant d'afficher au format *heure : minute*.

## 4) Utilisation de la seconde méthode

Créer une instance nommée **h2** de la classe **Heure2** avec les arguments 22 et 12 ; puis faire comme dans la question 2 pour l'objet h2.

## 5) Troisième méthode

On va maintenant créer une classe **Heure3** avec une autre interface.

Créer une nouvelle classe nommée **Heure3** avec un constructeur ayant 3 paramètres hh =12, mm =0, ss =0 et 3 attributs **heure**, **minute**, **seconde**.

Créer une méthode **afficheHeure()** qui affiche l'heure au format heure :minute :seconde.

Créer une méthode **setHeure()** ayant comme paramètre **h** et change la valeur de **heure**.

## 6) Utilisation de la troisième méthode

Créer une instance nommée **h3** de la classe **Heure3** sans argument et appliquer la méthode **afficheHeure()** à cet objet.

Mettre l'heure à 13 puis afficher la nouvelle heure.

Créer une instance nommée **recreation** de la classe **Heure3** avec les arguments 9, 50, 18, puis afficher la nouvelle heure.

**Remarques:**

les classes **Heure1** et **Heure2** ont la **même interface** c'est-à-dire les **mêmes méthodes** **getHeure()** , **getMinute()**, **setHeure()**, **setMinute()** et **incrémenter()** mais elles sont implémentées de manière différente.

## IV. TP4 : Sujet 24-NSI-37, Exercice 2

**CODE DONNE EN PIECE JOINTE** On définit une classe gérant une adresse IPv4.

On rappelle qu'une adresse IPv4 est une adresse de longueur 4 octets, notée en décimale à point, en séparant chacun des octets par un point. On considère un réseau privé avec une plage d'adresses IP de 192.168.0.0 à 192.168.0.255.

On considère que les adresses IP saisies sont valides. Les adresses IP 192.168.0.0 et 192.168.0.255 sont des adresses réservées. Le code ci-dessous implémente la classe AdresseIP.

```
class AdresseIP:

    def __init__(self, adresse):
        self.adresse = ...

    def liste_octet(self):
        """renvoie une liste de nombres entiers,
        la liste des octets de l'adresse IP"""
        return [int(i) for i in self.adresse.split(".")]

    def est_reservee(self):
        """renvoie True si l'adresse IP est une adresse
        reservee, False sinon"""
        return ... or ...

    def adresse_suivante(self):
        """renvoie un objet de AdresseIP avec l'adresse
        IP qui suit l'adresse self
        si elle existe et False sinon"""
        if ... < 254:
            octet_nouveau = ... + ...
            return AdresseIP('192.168.0.' + ...)
        else:
            return False
```

Compléter le code ci-dessus et instancier trois objets : adresse1, adresse2, adresse3 avec respectivement les arguments suivants :

'192.168.0.1', '192.168.0.2', '192.168.0.0'

Vérifier que :

```
>>> adresse1.est_reservee()
```

False

```
>>> adresse3.est_reservee()
```

True

```
>>> adresse2.adresse_suivante().adresse
```

'192.168.0.3'