Bauvais Camille, Creuze Martin

TP3

Exercice 1 :

```
class Lieu():
    #constructeur
    def __init__(self, nom, adresse):
        self.__nom = nom
        self.__adresse = adresse
        self.__latitude, self.__longitude = Gmap.coordgps(adresse)
        #méthode coordgps() importée de la classe Gmap intégrée dans notre
code(donnée dans l'énoncé)

    #methode qui affiche l'ensemble des attributs d'un lieu
    def detail(self):
        print(f'nom : {self.__nom}, adresse : {self.__adresse}, latitude :
{self.__latitude}, longitude : {self.__longitude}')

def saisieLieu():
    print(" -- Programe qui récupère les coordonnées d'un lieu -- ")
    nom_lieu = input("Entrer le nom du lieu : ")
    adresse_lieu = input ("Entrer l'adresse du lieu : ")
    adresse1 = Lieu(nom_lieu, adresse_lieu)
    print ('Détails du lieu = ')
    adresse1.detail()
```

Essai :

```
 -- Programe qui récupère les coordonnées d'un lieu --
Entrer le nom du lieu : Benjamin Franklin
Entrer l'adresse du lieu : Rue Roger Couttolenc
Détails du lieu =
nom : Benjamin Franklin, adresse : Rue Roger Couttolenc, latitude : 49.4148346, longitude : 2.8178771
```

Exercice 2 :

```
class Personne():
    #constructeur, initialisation avec John Doe (30 ans)
    def __init__(self, nom='Doe', prenom='John', age=30, sexe='M'):
        self.__nom = nom
        self.__prenom = prenom
        self.__age = age
        self.__sexe = sexe

    #accesseurs
    def getNom(self):
        return self.__nom

    def getPrenom(self):
        return self.__prenom

    def getAge(self):
        return self.__age

    def getSexe(self):
        return self.__sexe

    #methode qui permet de savoir si les deux personnes ont même nom de fam
    def sameLastName(self, p):
        res = self.getNom() == p.getNom() #TEST D'EGALITE DES NOMS DE
FAMILLE
        return res

    #methode qui compare deux personnes et retourne la plus agée
    def oldest(self, p):
        if (self.getAge()>p.getAge()): #TEST SI PLUS AGE
            return f'{self.getPrenom()} {self.getNom()} est plus agé que
{p.getPrenom()} {p.getNom()}.'
        elif (self.getAge()<p.getAge()): #TEST SI PLUS JEUNE
            return f'{self.getPrenom()} {self.getNom()} est plus jeune que
{p.getPrenom()} {p.getNom()}.'
        elif (self.getAge() == p.getAge()): #TEST SI MEME AGE
            return f'{self.getPrenom()} {self.getNom()} et {p.getPrenom()}
{p.getNom()} ont le même âge.'


def repPersonne():
    #création de 3 personnes pour tester la classe et les méthodes
    personne1 = Personne('Bauvais', 'Camille', 19, 'F')
    personne2 = Personne('Creuze', 'Martin', 19, 'H')
    personne3 = Personne('Creuze', 'Albert', 45, 'H')

    print(" -- Initialisation par défaut et essai des getters -- ")
    personne4 = Personne()#test de la personne par défaut
    print(f"Nom = {personne4.getNom()}; Prenom = {personne4.getPrenom()};
Age = {personne4.getAge()}; Sexe = {personne4.getSexe()}") #test getters
    print()

    #tests méthodes
    print(' -- Essai de la fonction sameLastName -- ')
    print('         1) Avec le même nom de famille')
    print(personne2.sameLastName(personne3))
    print('         2) Avec un nom de famille différent')
    print(personne1.sameLastName(personne2))
    print()
```

```
print(' -- Essai de la fonction oldest -- ')
print('         1) Avec un age inférieur')
print(personne1.oldest(personne3))
print('         2) Avec un age supérieur')
print(personne3.oldest(personne1))
print('         3) Avec un age identique')
print(personne1.oldest(personne2))
```

Essai :

```
 -- Initialisation par défaut et essai des getters --
Nom = Doe; Prenom = John; Age = 30; Sexe = M

 -- Essai de la fonction sameLastName --
         1) Avec le même nom de famille
True
         2) Avec un nom de famille différent
False

 -- Essai de la fonction oldest --
         1) Avec un age inférieur
Camille Bauvais est plus jeune que Albert Creuze.
         2) Avec un age supérieur
Albert Creuze est plus agé que Camille Bauvais.
         3) Avec un age identique
Camille Bauvais et Martin Creuze ont le même âge.
```

Exercice 3

```
from math import *

class Point:
    #constructeur
    def __init__(self, abs, ord):
        self.abs = abs
        self.ord = ord

    #getters et setters avec le décorateur property
    @property
    def abs(self):
        return self.__abs

    @abs.setter
    def abs(self, val):
        self.__abs = val

    @property
    def ord(self):
        return self.__ord

    @ord.setter
    def ord(self, val):
        self.__ord = val

    #AJOUT DE LA METHODE STR POUR L'AFFICHAGE DES COORDONNEES
    def __str__(self):
        return f"Coordonnées : ({self.abs};{self.ord})"

    #methode pour calculer la distance [objet courant, objet en param]
    def calculerDistance(self, p):
        print("Distance :", end=" ")
        dist = round(sqrt((p.ord - self.ord) ** 2 + (p.abs - self.abs) **
2), 3)
        return dist

    #methode pour calculer le milieu de [objet courant, objet en param]
    def calculerMilieu(self, p):
        print("Milieu -", end=" ")
        milieu = Point((p.abs + self.abs) / 2, (p.ord + self.ord) / 2)
        return milieu

class TroisPoints():
    #constructeur
    def __init__(self, point1, point2, point3):
        self.point1 = point1
        self.point2 = point2
        self.point3 = point3

    #getters et setters avec le décorateur property
    @property
    def point1(self):
        return self.__point1
    @point1.setter
    def point1(self, val):
        self.__point1 = val

    @property
    def point2(self):
```

```
        return self.__point2

    @point2.setter
    def point2(self, val):
        self.__point2 = val

    @property
    def point3(self):
        return self.__point3

    @point3.setter
    def point3(self, val):
        self.__point3 = val

    #AJOUT DE NOUVELLES METHODES CALCULER DISTANCE A UTILISER DANS ALIGNE
ET ISOCELE
    def calculerDistanceAB(self):
        dist = round(sqrt((self.point1.abs - self.point2.abs)**2 +
(self.point1.ord - self.point2.ord)**2), 3)
        return dist

    def calculerDistanceAC(self):
        dist = round(sqrt((self.point1.abs - self.point3.abs)**2 +
(self.point1.ord - self.point3.ord)**2), 3)
        return dist

    def calculerDistanceBC(self):
        dist = round(sqrt((self.point2.abs - self.point3.abs)**2 +
(self.point2.ord - self.point3.ord)**2), 3)
        return dist

    #methode qui retourne True si les 3 points sont alignés
    def sontAlignes(self):
        res = ((self.calculerDistanceAB() == self.calculerDistanceAC() +
self.calculerDistanceBC()) or (
            self.calculerDistanceAC() == self.calculerDistanceAB() +
self.calculerDistanceBC()) or (
            self.calculerDistanceBC() == self.calculerDistanceAC() +
self.calculerDistanceAB()))
        return res

    #methode qui retourne True si les 3 points forment un triangle isocele
    def estIsocele(self):
        res = ((self.calculerDistanceAB() == self.calculerDistanceBC()) or
(
            self.calculerDistanceAB() == self.calculerDistanceAC()) or (
            self.calculerDistanceBC() == self.calculerDistanceAC()))
        return res

def exo3():
    point1 = Point(0, 1)
    point2 = Point(1, 2)
    point3 = Point(2, 3)
    point4 = Point(0, 13)

    print(" -- Affichage des points -- ")
    print(f"Point 1 : {point1}")
    print(f"Point 2 : {point2}")
    print(f"Point 3 : {point3}")
    print(f"Point 4 : {point4}")
    print()
```

```
print(" -- Essai de calculerDistance --")
print("Distance entre point2 et point4 : ")
print(point2.calculerDistance(point4))
print()

print(" -- Essai de calculerMilieu -- ")
print("Milieu du point2 et point4")
print(point2.calculerMilieu(point4))
print()

lesPoints1 = TroisPoints(point1, point2, point3)
lesPoints2 = TroisPoints(point1, point2, point4)

print(" -- Essai de sontAlignes -- ")
print("Alignement des points 1, 2 et 3 (Résultat attendu = True)")
print(lesPoints1.sontAlignes())
print("Alignement des points 1, 2 et 4 (Résultat attendu = False)")
print(lesPoints2.sontAlignes())
print()

print(" -- Essai de estIsocele -- ")
print("Pour les points 1, 2 et 3 (Résultat attendu = True)")
print(lesPoints1.estIsocele())
print("Pour les points 1, 2 et 4 (Résultat attendu = False)")
print(lesPoints2.estIsocele())
```

Essai :

```
 -- Affichage des points --
Point 1 : Coordonnées : (0;1)
Point 2 : Coordonnées : (1;2)
Point 3 : Coordonnées : (2;3)
Point 4 : Coordonnées : (0;13)

 -- Essai de calculerDistance --
Distance entre point2 et point4 :
Distance : 11.045

 -- Essai de calculerMilieu --
Milieu du point2 et point4
Milieu - Coordonnées : (0.5;7.5)

 -- Essai de sontAlignes --
Alignement des points 1, 2 et 3 (Résultat attendu = True)
True
Alignement des points 1, 2 et 4 (Résultat attendu = False)
False
```

```
 -- Essai de estIsocele --
Pour les points 1, 2 et 3 (Résultat attendu = True)
True
Pour les points 1, 2 et 4 (Résultat attendu = False)
False
```