Contrôle de connaissances

Nom :..... **Prénom :**.....

Exercice 1 : (4 points)

On considère la classe suivante, représentant un élève par son nom, prénom. L'attribut note est initialement évaluer à 20 :

Maclasse :

```
class Eleve :
  def __init__(self, nom , prenom  ) :
    self. nom = nom
    self.prenom = prenom
    self.note = 20
```

Les attributs `nom` et `prenom` sont des chaines de caractères.

1. Écrire les méthodes **accesseurs** pour les deux attributs `nom` et `prenom` .

   Correction

   Correction :

   ```
   def get_nom ( self) :
       """ Methode accesseur pour nom """
       return self.nom
   def get_prenom ( self) :
       """ Methode accesseur pour prenom """
       return self.prenom
   ```

2. Écrire les méthodes **mutateurs** pour la note.

   Correction

   Correction :

   ```
   def set_note (self , nouvel_note ) :
       """ methode mutateur pour note """
       self.note = nouvel_note
   ```

3. Écrire la méthode **bonus** qui permet d'affecter un bonus, donné en argument, à la note. La note ne pourra tout de même dépasser la note de 20.

   Exemple : Si la note est de 12, et que le bonus est de 3, la note passe alors à 15.

   Correction

   Correction :

   ```
   def bonus(self, bon) :
     """ change la note"""
     self.note = self.note + bon
     if self.note > 20 :
       self.note = 20
   ```

4. Écrire la méthode `__str__` permettant l'affichage de l'objet sous la forme :
L'élève [nom] , [prenom] a eu la note de [note]

Correction

Correction :

```
def __str__(self)
  """methode d affichage"""
  return "Le␣plus␣petit␣element␣est␣{}␣,␣\
␣␣plus␣grand␣est␣{},␣et␣la␣moyenne␣est␣{}␣"\
  .format(min(self.a,self.b) , max(self.a,self.b) , self.moyenne())
```

Correction

```
class Maclasse :
    def __init__(self, a , b ) :
        self.a = a
        self.b = b

    def get_a ( self) :
        """ Methode accesseur pour a """
        return self.a
    def get_b ( self) :
        """ Methode accesseur pour b """
        return self.b
    def set_a (self , nouveau_a ) :
        """ methode mutateur pour a """
        self.a = nouveau_a
    def set_b (self , nouveau_b ) :
        """ methode mutateur pour b """
        self.b = nouveau_b
    def moyenne(self) :
        """ retourne la moyenne de a et de b """
        return ( self.a + self.b) /2
    def __str__(self) :
        mot = "Le␣plus␣petit␣est␣" + str( min(self.a , self.b) ) + \
              "␣,␣le␣plus␣grand␣est␣" + str( max(self.a , self.b) ) + \
              ",␣la␣moyenne␣est␣" + str( self.moyenne())
        return mot
```

Exercice 2 : (6 points)

On définie la classe `Montre` avec les attributs `heure` et `minute` de la façon suivante :

La classe Montre :

```
class Montre :
    def __init__ (self, heure,minute ):
        """ methode constructeur """
        self.heure = heure
        self.minute = minute
```

1. Écrire la méthode `minute_suivante`, qui augmente d'une minute.

Correction

Correction :

```
def minute_suivante(self) :
        """ augmente d'une minute """
        if self.minute == 59 :
            self.minute = 0
            if self.heure == 24 :
                self.heure = 0
            else :
                self.heure +=1
        else :
            self.minute += 1
```

2. Écrire la méthode `decalage_horaire_positif`, qui augmente l'horaire d'autant que l'argument.

Correction

Correction :

```
def decalage_horaire_positif(self , decalage ) :
        """ opere un decalage horaire positif """
        nouvelle_heure = (self.heure + decalage ) % 24
        self.heure = nouvelle_heure
```

3. Écrire la méthode `decalage_horaire_negatif`, qui diminue l'horaire d'autant que l'argument.

Correction

Correction :

```
def decalage_horaire_negatif(self , decalage ) :
        """ opere un decalage horaire negatif """
        nouvelle_heure = (self.heure - decalage ) % 24
        self.heure = nouvelle_heure
```

4. Écrire la méthode `__str__`, qui permet l'affichage de l'heure sous la forme :
il est 14 heures et 24 minutes

Correction

Correction :

```
    def __str__(self) :
            """ permet l'affichage de l'heure """
            return "il␣est␣{}␣heures␣et␣{}␣minutes".format(self.heure,self.
    minute)
```

Correction

```
class Montre :
    def __init__ (self, heure,minute ):
        """ methode constructeur """
        self.heure = heure
        self.minute = minute

    def minute_suivante(self) :
        """ augmente d'une minute """
        if self.minute == 59 :
            self.minute = 0
            if self.heure == 23 :
                self.heure = 0
            else :
                self.heure +=1
        else :
            self.minute += 1

    def decalage_horaire_positif(self , decalage ) :
        """ opere un decalage horaire positif """
        nouvelle_heure = (self.heure + decalage ) % 24
        self.heure = nouvelle_heure

    def decalage_horaire_negatif(self , decalage ) :
        """ opere un decalage horaire negatif """
        nouvelle_heure = (self.heure - decalage ) % 24
        self.heure = nouvelle_heure

    def __str__(self) :
        """ permet l'affichage de l'heure """
        return "il␣est␣{}␣heures␣et␣{}␣minutes".format(self.heure,self.minute)
```