

## ALGO2 – Algorithmique et Programmation en Python 2

---

### Fiche de TP numéro 3 - Groupe 3

**Exercice 1 :** Définissez une classe `CompteBancaire()`, qui permette d'instancier des objets tels que `compte1`, `compte2`, etc. Le constructeur de cette classe initialisera deux attributs `__nom` et `__solde`. Par défaut, le solde sera initialisé à 1000 euros.

Trois autres méthodes seront définies :

- `depot(somme)` permettra d'ajouter une certaine somme au solde.
- `retrait(somme)` permettra de retirer une certaine somme du solde. Un retrait supérieur au solde du compte ne sera pas effectué.
- `affiche()` permettra d'afficher le nom du titulaire et le solde de son compte.

```
>>> compte1 = CompteBancaire('Michu', 800)
>>> compte1.depot(350)
>>> compte1.retrait(200)
>>> compte1.affiche()
Le solde du compte bancaire de Michu est de 950 euros.
>>> compte2 = CompteBancaire('Blanchard')
>>> compte2.depot(25)
>>> compte2.affiche()
Le solde du compte bancaire de Blanchard est de 1025 euros.
```

**Exercice 2 :** Définissez une classe `Voiture` qui permet de créer des objets représentant des voitures. Une `Voiture` est caractérisée par sa couleur, son nom et sa vitesse. Par défaut, à la création d'une `Voiture`, on doit donner un nom et une couleur, qui permettront de donner des valeurs aux attributs `__nom` et `__couleur`, mais la `__vitesse` est initialisée à zéro.

Quelques autres méthodes seront définies :

- la méthode `accelere(inc)` qui augmente de `inc` la vitesse de la voiture. Nos voitures sont limitées à 130 km/h, et ne peuvent pas être accélérées de plus de 10km/heure (pour augmenter de 20km/h, il faudra donc appeler deux fois la méthode `accelere()`). C'est à `accelere()` de contrôler tout cela.
- la méthode `freine(dec)` qui diminue de `dec` la vitesse de la voiture. Une voiture ne peut pas rouler à une vitesse négative (elle est à l'arrêt lorsque sa vitesse est à zéro).
- la méthode `str()` qui retourne une chaîne de caractères avec toutes les indications utiles sur la voiture (quel est son nom ? quelle est sa couleur ? est-elle à l'arrêt ? sinon quelle est sa vitesse ?).
- la méthode `affiche()` qui affiche sur la console toutes les indications utiles sur la voiture.

Travail à faire :

- vous devez écrire et spécifier chacune des méthodes demandées
- vous devez écrire des tests unitaires pour chacune des méthodes
- vous devez écrire un script principal qui illustre l'utilisation de chacune des méthodes.

## Le jeu du démineur

Il s'agit de ce jeu où vous devez découvrir des bombes cachées dans une grille. Lorsque vous cliquez sur une cellule, celle-ci dévoile son contenu : si elle contient une bombe, *Boum!*, vous avez perdu ; si elle contient une valeur strictement positive, elle indique combien de bombes sont présentes dans les cellules voisines ; sinon, elle est vide et ses voisines sont à leur tour découvertes. Vous avez gagné lorsque les seules cases non découvertes sont celles qui contiennent une bombe.

### Exercice 3 : Une case du jeu

Définissez une classe `Case` qui va représenter une case du plateau de jeu. Elle contient deux informations. L'attribut `__valeur` vaut -1 si la case contient une bombe, et 0 (au départ) sinon. Par défaut, `__valeur` est fixée à 0. L'attribut `__cache` est initialisé à `True` dans le constructeur.

La classe `Case` propose les méthodes suivantes :

- `est_cache(self)` : indique si la case est cachée ou pas
- `est_visible(self)` : indique si la case est visible ou pas
- `est_bombe(self)` : indique si la case contient une bombe
- `est_vide(self)` : indique si la valeur de la case est nulle
- `incremente_valeur(self)` : augmente de un la valeur contenue dans la case (uniquement si la case ne contient pas de bombe)
- `montre_toi(self)` : passe l'attribut `__cache` à `False`
- `str(self)` : retourne une représentation textuelle d'une case ('-' si la case est cachée, '\*' si la case contient une bombe, ' ' si la valeur est nulle, et la valeur dans le dernier cas).

Spécifiez, écrivez des tests unitaires et implémentez ces méthodes ainsi que le constructeur de la classe `Case`.

### Exercice 4 : Le plateau de jeu

Définissez une classe `Demineur` qui va représenter le plateau de jeu. Il contient principalement trois informations : son nombre de lignes `__nb_lig`, son nombre de colonnes `__nb_col`, et le plateau de jeu qui est une liste de listes de `Case`.

- Le constructeur de `Demineur(self, nb_lig, nb_col)` initialise les attributs et crée un plateau de `nb_lig × nb_col` instances de `Case` (elles sont toutes vides). On le nommera `__plateau`. Par défaut, le plateau est construit avec 10 lignes et 20 colonnes.
- La méthode `affiche_plateau(self)` affiche sur la console le plateau. Différents exemples sont présentés plus loin dans le sujet. Pour écrire cette méthode, vous en ajouterez deux autres :
  - `affiche_ligne_traits(self)` qui affiche une ligne de '-' de la bonne dimension
  - `affiche_ligne(self, lig)` qui affiche la ligne d'indice `lig` du plateau.
- La méthode `pose_bombes(self, nb_bombes)` pose aléatoirement `nb_bombes` sur le plateau. Elle ajoute 1 à la valeur de chacune des voisines de chaque bombe posée. Par défaut, `nb_bombes` vaut 15.
- La méthode `partie_finie(self, l, c)` retourne si la partie est finie, c'est-à-dire soit si la case à la position `(l, c)` dans le plateau est une bombe, ou bien s'il n'y a plus aucune case à découvrir.
- La méthode `montre(self, l, c)` qui découvre une case, et la méthode `montre_case(self, l, c)` qui gère la découverte des cases voisines sont données ci-dessous :

```
def montre(self, l, c) :  
    '''Demineur, int, int, -> None
```

```
    découvre la case (l,c) et affiche le plateau.
```

```

    l et c doivent être positifs et inférieurs à, respectivement,
    __nb_lig et __nb_col.
    '''

    self.montre_case(l,c)
    self.affiche_plateau()

def montre_case(self,l,c) :
    '''Demineur, int, int, -> None

    découvre la case (l,c) et ses voisines le cas échéant.
    l et c doivent être positifs et inférieurs à, respectivement,
    __nb_lig et __nb_col.
    '''

    if not self.__plateau[l][c].est_visible() :
        self.__plateau[l][c].montre_toi()
    if self.__plateau[l][c].est_bombe() :
        print("Boum!!")
    elif self.__plateau[l][c].est_vide() :
        if c > 0 :
            self.montre_case(l,c-1)
        if l > 0 :
            self.montre_case(l-1,c-1)
        if l < self.__nb_lig-1 :
            self.montre_case(l+1,c-1)
        if c < self.__nb_col-1 :
            self.montre_case(l,c+1)
        if l > 0 :
            self.montre_case(l-1,c+1)
        if l < self.__nb_lig-1 :
            self.montre_case(l+1,c+1)
        if l > 0 :
            self.montre_case(l-1,c)
        if l < self.__nb_lig-1 :
            self.montre_case(l+1,c)

```

Spécifiez, écrivez des tests unitaires et implémentez ces méthodes ainsi que le constructeur de la classe Démineur.

**Exercice 5 :** Écrivez un programme principal qui permet de jouer au démineur avec un utilisateur.

---

## Annexe

Voici un exemple d'interaction avec l'utilisateur qui montre l'affichage du plateau à différentes étapes.

```

moi@maMachine:~/algo2/tp2$ python3
Python 3.4.3+ (default, Oct 14 2015, 16:03:50)
[GCC 5.2.1 20151010] on linux

```

Type "help", "copyright", "credits" or "license" for more information.

```
>>> from demineur import *
```

```
>>> dem = Demineur()
```

```
>>> dem.pose_bombes()
```

```
>>> dem.montre(6,8)
```

```

    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0 |  | 1|--|--|--|--| 1|  |  |  |  |  |  | 1|--|--|--|--| 1|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
1 |  | 1| 1| 1| 1| 1| 1|  |  |  |  |  |  | 1|--|--|--|--| 2|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
2 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1|--|--|--|--| 2|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
3 |  |  |  |  |  |  |  |  |  |  |  |  | 1| 1|--|--| 2| 1| 1|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
4 | 1| 2| 1| 1|  |  |  |  |  |  |  | 1|--|--|--| 1|  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
5 |--|--|--| 1| 1| 1| 1|  |  |  |  |  | 1| 1| 2| 1| 1|  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
6 |--|--|--|--|--|--| 1|  |  |  |  |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
7 |--|--|--|--|--|--| 1|  |  |  |  |  |  | 1| 1| 1|  |  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
8 |--|--|--|--|--|--| 1|  |  |  |  |  |  | 1|--| 1|  | 1| 1| 1|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
9 |--|--|--|--|--|--| 1|  |  |  |  |  |  | 1|--| 1|  | 1|--|--|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```
>>> dem.montre(6,5)
```

Boum !!

```

    0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
0 |  |  |  |  |  | 1| 1| 1| 1|--| 3|--| 1|  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
1 |  | 1| 1| 1| 1| 1| 1|  |  |  |  |  |  | 1|--|--|--|--| 2|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
2 |  |  |  |  |  |  |  |  |  |  |  |  |  | 1|--|--|--|--| 2|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
3 |  |  |  |  |  |  |  |  |  |  |  |  | 1| 1|--|--| 2| 1| 1|  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
4 | 1| 2| 1| 1|  |  |  |  |  |  |  | 1|--|--|--| 1|  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
5 |--|--|--| 1| 1| 1| 1|  |  |  |  |  | 1| 1| 2| 1| 1|  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
6 |--|--|--|--|--|--|**| 1|  |  |  |  |  |  |  |  |  |  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
7 |--|--|--|--|--|--| 1|  |  |  |  |  |  | 1| 1| 1|  |  |  |  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
8 |--|--|--|--|--|--| 1|  |  |  |  |  |  | 1|--| 1|  | 1| 1| 1|

```

```

+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
9 |--|--|--|--|--|--|--| 1|  |  |  |  |  | 1|--| 1|  | 1|--|--|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```