Initiation à la Programmation Orientée Objet

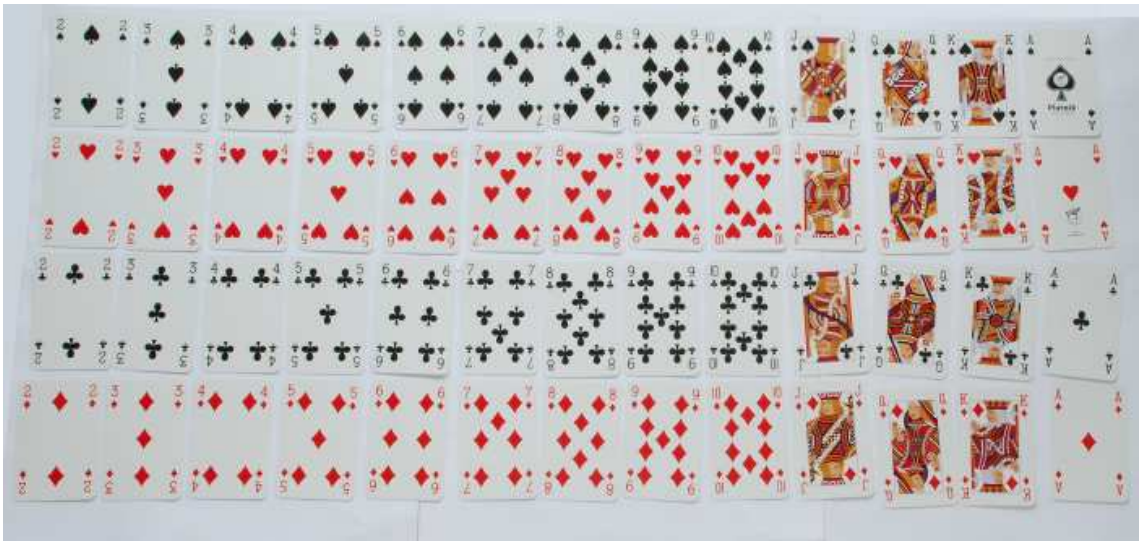
La **P**rogrammation **O**rientée **O**bjet (POO en abrégé) est un paradigme de programmation (une façon de programmer) reposant, comme son nom l'indique, sur la création et l'utilisation d'**objets**.

L'idée principale de la POO est de pouvoir représenter en machine des objets usuels, mathématiques, physiques, ... afin de pouvoir les manipuler informatiquement. En langage Python, un certain nombre de catégories d'objets (que l'on appelle **classes**) sont déjà implantées : les nombres entiers (classe **int**), les nombres réels (classe **real**), les listes (classe **list**), les chaînes de caractères (classe **str**), les dictionnaires (classe **dict**), les fonctions (classe **function**), etc.

En Python, le mot clé class, permet de définir une nouvelle classe d'objets. La définition d'une telle classe va permettre de spécifier les propriétés (que l'on nommera **attributs**) des objets de cette classe et les fonctions (que l'on nommera **méthodes**) permettant d'agir sur les objets de cette classe.

Par exemple, on peut créer une **classe** *Chaussure*. Les objets de cette classe seront donc de type *Chaussure*. On peut spécifier un certain nombre d'**attributs** pour les objets de cette classe : la pointure, la couleur, la fonction (chaussure de ville, chaussure de sport, ...), etc et un certain nombre de **méthodes** que l'on peut appliquer à ces objets : lasser, brosser, cirer, ranger, ...

Dans la suite de cette séance, nous allons, pour illustrer la notion d'objet, nous intéresser à la création et à la manipulation d'un jeu de 52 cartes à jouer.



I. Création de la classe Carte

1. Saisir dans une console Python les lignes de code suivantes :

```
>>> class Carte():
>>>     pass
```

   Nous venons de définir ainsi une nouvelle classe d'objet : Carte.

   Avec l'instruction pass, nous n'avons, pour le moment, spécifié aucun attribut ni aucune méthode pour les objets de type Carte.

Pour construire un objet de cette classe (on dira aussi **instancier un objet** ou **créer une instance** de cette classe) saisissez la ligne suivante :

```
>>> carte = Carte()
```

Remarque : Notez que l'on préférera, dans ce cours tout au moins, noter les noms des classes avec une majuscule et les instances de cette classe en minuscule.

2. Notre nouvel objet **carte** n'a pour l'instant aucun *attribut* particulier. Nous allons en préciser deux en lui attribuant une *hauteur* (nombre entier entre 2 (deux) et 14 (As)) et une *couleur* (chaine de caractère parmi 'T' pour Trèfle, 'K' pour Carreau, 'C' pour Coeur et 'P' pour Pique).

   Ajoutez des attributs *hauteur* et *couleur* en saisissant les lignes suivantes :

```
>>> carte.hauteur = 7
>>> carte.couleur = 'C'
```

   Si l'on saisit maintenant les instructions **carte.hauteur** et **carte.couleur** dans la console, les réponses données sont 7 et 'C' :

```
>>> carte.hauteur
7
>>> carte.couleur
'C'
```

   Il est en fait possible (et souvent préférable) de spécifier les attributs de l'objet dès sa création. Pour cela, nous allons définir une méthode d'initialisation de la classe **Carte** dans la définition de la classe. Il existe un mot clé pour cette méthode : **___init___**

3. Saisissez, dans la zone de programmation, les lignes suivantes :

```
class Carte():
    def __init__(self, hauteur, couleur):
        self.hauteur = hauteur
        self.couleur = couleur
```

   Sauvez et exécutez votre programme puis, dans la console, créez un nouvel objet **carte** :

```
>>> carte = Carte(7,'C')
```

   Nous venons de créer un nouvel objet **carte** de classe **Carte** dont les attributs *hauteur* et *couleur* ont été spécifiés dès la création de l'objet.

   Les instructions **carte.hauteur** et **carte.couleur** saisies dans la console, donneront comme précédemment les réponses 7 et 'C' :

```
>>> carte.hauteur
7
>>> carte.couleur
'C'
```

   Remarque : le mot **self** permet de désigner l'objet *lui-même*. On utilise la syntaxe **self**.*attribut* pour désigner la valeur d'un attribut de l'objet. **self** est un argument nécessaire pour toutes les méthodes d'une classe.

4. Saisissez, dans la console, la ligne suivante :

```
>>> print(carte)
```

   La réponse donnée devrait apparaitre sous la forme suivante :

```
<__main__.Carte object at 0x000001F23FB571D0>
```

Alors que nous aurions souhaité voir apparaitre le contenu de notre objet **carte**, c'est-à-dire ses attributs, la console nous renvoie le type de l'objet Carte et l'adresse mémoire où il est stocké.

Pour maitriser l'affichage obtenu avec l'instruction **print**, il faut définir une méthode de représentation de l'objet dans la définition de la classe, grâce au mot clé : **___repr___**

Dans la zone de programmation, complétez la définition de la classe Carte en ajoutant la méthode **___repr___** suivante :

```
class Carte():
    def __init__(self, hauteur, couleur):
        self.hauteur = hauteur
        self.couleur = couleur

    def __repr__(self):
        return str(self.hauteur)+couleur
```

Sauvez et exécutez votre programme puis, dans la console, créez l'objet **carte** :

```
>>> carte = Carte(7,'C')
```

cette fois l'instruction **print(carte)** va automatiquement faire appel à la méthode **___repr___** et renvoyer la chaine de caractère **7C** :

```
>>> print(carte)
7C
```

Modifiez maintenant la méthode **___repr___** de la classe Carte pour qu'elle renvoie l'identité d'une carte quelconque sous la forme d'une chaine de caractère.
Par exemple, l'instruction **print(Carte(14, 'P'))** doit afficher : As de Pique.

5. Dans les documents envoyés pour cette séance vous trouverez un dossier **Cartes** dans lequel figure les images, au format GIF, des 52 cartes du jeu. Copiez ce dossier dans le répertoire de travail de cette séance.

   Nous allons créer une méthode **image**, à l'aide de la bibliothèque Tkinter, qui permette d'afficher l'image de la carte choisie.

   En premier lieu, saisir en première ligne de votre programme l'instruction :

```
import tkinter as tk
```

   Ensuite, copiez ou saisissez le code de la méthode **image** de la classe Carte :

```
def image(self):
    fichier = "chemin d'acces au dossier"+str(self.hauteur)+self.couleur+".GIF"
    fenetre = tk.Tk()
    fenetre.geometry('135x192+1000+400')
    image_carte = tk.PhotoImage(file=fichier)
    label = tk.Label(fenetre, image=image_carte)
    label.pack()
    fenetre.mainloop()
```

   Testez cette nouvelle méthode après avoir sauvé et exécuté votre programme.

II. Création de la classe PaquetDeCartes

Dans cette section nous allons construire une nouvelle classe d'objets **PaquetDeCartes** permettant de définir un paquet de cartes.

Nous allons considérer que les objets de cette classe ne possèdent qu'un seul attribut, que l'on notera **contenu**, égal à la liste des cartes du paquet.

6. Créez, à la suite de votre programme, une nouvelle classe **PaquetDeCartes** dont la méthode **___init___** permet de définir l'attribut **contenu** des instances de cette classe.
   Par exemple, l'instruction :

   ```
   >>> paquet = PaquetDeCarte([Carte(7,'C'),Carte(14,'P')])
   ```

   doit permettre de créer un objet *paquet* dont le *contenu* est le 7 de Coeur et l'as de Pique.

7. Ajoutez une méthode **___repr___** à la classe **PaquetDeCartes** qui permet d'afficher le contenu du paquet sous forme d'une chaine de caractère. Cette méthode pourra faire appel à la méthode **___repr___** de la classe **Carte**.
   Par exemple, en considérant l'objet *paquet* de l'exemple précédent, lors de l'utilisation de la fonction **print**, on doit voir apparaitre :

   ```
   >> print(paquet)
   7 de Coeur
   As de Pique
   ```

8. Ajoutez une méthode **est_vide** à la classe **PaquetDeCartes** qui renvoie un booléen égal à True si le paquet ne contient aucune carte et False sinon.

9. Ajoutez une méthode **taille** à la classe **PaquetDeCartes** qui renvoie le nombre de cartes contenue dans un paquet de cartes.

10. Ajoutez une méthode **battre** à la classe **PaquetDeCartes** qui permet de battre le paquet de cartes, c'est-à-dire de le mélanger. On pourra pour cela utiliser la fonction **shuffle** de la bibliothèque random.

11. Ajoutez une méthode **tirer_carte** à la classe **PaquetDeCartes** qui tire la première carte du paquet : cela signifie que la carte est retirée du paquet et qu'elle est renvoyée en réponse de la méthode **tirer_carte**. Si cette méthode est invoquée sur un paquet de cartes vide, elle doit renvoyer la valeur **None**.

12. Ajoutez une méthode **ajouter_carte** à la classe **PaquetDeCartes** qui permet d'ajouter une carte à un paquet de cartes.

13. Ajoutez une méthode **ajouter_paquet** à la classe **PaquetDeCartes** qui permet d'ajouter un paquet de cartes à un paquet de cartes. Vous pourrez pour cela vous appuyer sur la méthode **ajouter_carte**.

14. Ajoutez enfin une méthode **distribuer** à la classe **PaquetDeCartes** qui permet de distribuer un paquet de cartes entre un certain nombre de joueurs. La méthode devra prendre en argument le nombre de joueurs et renvoyer la liste des paquets de cartes créés pour chacun des joueurs. Vous pourrez pour cela vous appuyer sur les méthodes **tirer_carte** et **ajouter_carte**.

III. Création d'un jeu de Bataille

La *bataille* est un jeu de cartes qui se joue habituellement à deux. Le jeu commence en distribuant aux joueurs, de manière équitable, l'intégralité du paquet de 52 cartes. Les joueurs disposent devant eux, face cachée, leurs paquets de cartes. À chaque tour, chaque joueur retourne la première carte de son paquet. Celui qui a la carte de plus grande hauteur *remporte le pli* : il prend les cartes posées et les range face cachée sous son paquet.

En cas d'égalité de hauteurs, on dit qu'il y a *bataille* : les joueurs commencent par placer une première carte face cachée puis une seconde carte face visible. En cas de nouvelle égalité, la procédure est répétée. À la fin, le joueur gagnant remporte toutes les cartes posées, qu'il place sous son paquet.

Lorsqu'un joueur a en sa possession toutes les cartes du jeu, la partie se termine et il est déclaré gagnant (elle peut aussi se terminer au bout d'un certain nombre de tours lorsque les deux joueurs en ont marre ! On désigne alors gagnant le joueur qui a le plus de cartes dans son paquet).

Mini-projet : Créer une simulation du jeu de la *Bataille* entre deux joueurs **A** et **B** en utilisant les classes **Carte** et **PaquetDeCartes** codées précédemment.

Conseil : au départ, pour simplifier la conception de votre jeu, commencez par exclure les cas de bataille en modifiant temporairement la règle de la *Bataille* : en cas d'égalité de hauteur des deux cartes tirées, le joueur **A** remporte le pli. Une fois que votre jeu simplifié est opérationnel, penchez-vous sur le traitement des cas de *Bataille*.