Exercices d'application

1. Coordonnées dans le plan

On souhaite caractériser informatiquement la notion de point telle qu'elle existe dans le plan, aussi bien en coordonnées cartésiennes (x,y) qu'en cordonnées polaires (r, theta).

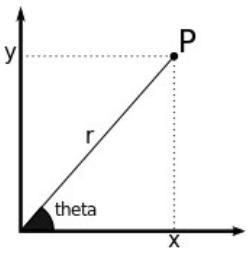


Figure 1: Coordonnées d'un point

Pour cela, nous allons construire la classe suivante :

Point
- x : float - y : float - r : float - theta : float
+ __init__(x : float, y : float) + lire_coord_cartesiennes() : (float, float) + lire_coord_polaires() : (float, float) - convertir_cart_polaire()

Q1. Relever sur le diagramme précédent :
- ➔ le nom de la classe
- ➔ le nombre d'attributs de cette classe
- ➔ le type d'accès à ces attributs
- ➔ le nom des accesseurs

Implémentation de la méthode en langage Python

Le fichier incomplet de la classe est disponible **ici**.

Les méthodes `lire_coord_cartesiennes()` et `lire_coord_polaires()` doivent renvoyer un tuple contenant respectivement les coordonnées catésiennes (x,y) et polaires (r,theta).

Q2. Compléter les méthodes `lire_coord_cartesiennes()` et `lire_coord_polaires()`.

Lors de l'instanciation d'un objet, le constructeur doit initialiser les attributs x et y avec les paramètres passés et calculer les attributs r et theta avec la méthode `__convertir_cart_polaire()`. Cette dernière calcule les coordonnées polaires du point à partir des coordonnées cartésiennes passées en argument.

Q3. Compléter le constructeur de la classe.

Q4. A l'aide de cette classe **déterminer** les coordonnées polaires des 4 points suivants. **Décrire** la démarche employée.
- ➔ A(-2,5)
- ➔ B(5,5)
- ➔ C(-2,-2)
- ➔ D(5,-2)

2.Jeu de carte

On souhaite disposer d'une classe Cartes qui représente une carte à jouer d'un jeu de 52 cartes.
Dans le monde réel, une carte est définie par :

- ➔ **sa couleur** : carreau, cœur, pique, trèfle
- ➔ **son poids** : de 2 à 10 pour les cartes numérotées, de 11 à 13 pour les figures et 14 pour l'as
- ➔ **sa figure** : 2 à 10, Valet, Dame, Roi, As

Le diagramme de classe de la classe Carte dont les attributs sont ceux définis au dessus peut s'écrire comme sur la figure 1.

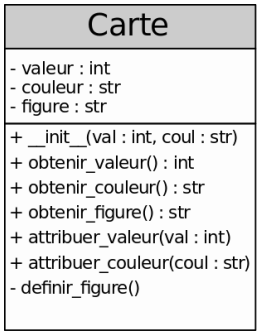


Figure 2: Diagramme de classe de la classe Carte

Le fichier incomplet de la classe Carte est disponible **ici**:

Q5. Justifier que la méthode `definir_figure()` ait un accès privé alors que toutes les autres méthodes ont un accès public.

Q6. Donner l'instruction permettant de créer une instance de la classe `Carte` nommée `carte1` dont la valeur est le valet de trèfle

Q7. Donner l'instruction permettant de recevoir la figure de `carte1`

Création d'un jeu de carte

On souhaite créer un jeu de 52 cartes sous la forme d'une liste de longueur 52 contenant l'ensemble des cartes différentes.

Q8. Donner une instruction permettant de créer une liste de longueur 52 initialisée avec des zéros

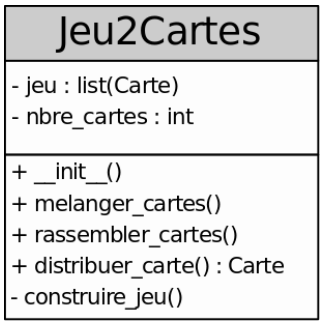
Q9. Définir un programme d'initialisation du jeu de carte avec les différentes cartes ordonnées par valeur et par couleur. Ce programme peut être composé d'une double boucle `for` balayant les valeurs et les couleurs.

Le mélange aléatoire d'une liste peut être réalisée avec la fonction shuffle de la bibliothèque random. L'appel de cette fonction s'écrit :

```
1 import random
2 random.shuffle(liste)
```

Classe Jeu2Cartes

On souhaite définir une classe Jeu2Cartes suivant :



Cette nouvelle classe devra être enregistrée dans le fichier jeu2cartes.py
Cette classe étant enregistrée dans un nouveau fichier, il sera nécessaire d'importer en tête de ce fichier, la classe `Carte` avec l'instruction `import Carte`

Les méthodes et attributs de la classer jeu2Cartes devront respecter les exigences suivantes :

	Nom	Description
Attributs	`jeu`	Liste contenant les cartes du jeu. Cette liste diminue au fur et à mesure que les cartes sont distribuées.
	`nbr_cartes`	Entier indiquant le nombre de cartes restantes dans le jeu.
Méthodes	`__init__`	Constructeur de l'objet initialisant les attributs nb_cartes et jeu. L'attribut jeu doit être initialisé avec le jeu de carte complet et mélangé.
	`melanger()`	Mélange l'ordre des éléments de la liste jeu
	`rassembler_cartes()`	Ecrase le contenu de jeu avec un nouveau jeu de carte complet et mélangé
	`distribuer_carte()`	Renvoie la première carte de la liste jeu tout en la supprimant de la liste.
	`construire_jeu()`	Construit une liste de 52 cartes mélangées

Q10. Implémenter la classe jeu2cartes en langage Python.