Exercice 1

Classe Point

Créer une classe `Point` qui représente un point dans un plan 2D.

Attributs :

- `x` : abscisse du point
- `y` : ordonnée du point

Méthodes :

- `__init__(self, x, y)` : constructeur
- `afficher(self)` : affiche les coordonnées sous la forme "(x, y)"
- `deplacer(self, dx, dy)` : déplace le point de dx en abscisse et dy en ordonnée

1.1 Créer un point P1 de coordonnées (3, 5)

1.2 Afficher ses coordonnées

1.3 Le déplacer de (2, -1)

1.4 Afficher ses nouvelles coordonnées

Exercice 2

Classe Rectangle

Créer une classe `Rectangle` qui représente un rectangle.

Attributs :

- `longueur` : longueur du rectangle
- `largeur` : largeur du rectangle

Méthodes :

- `__init__(self, longueur, largeur)` : constructeur
- `perimetre(self)` : retourne le périmètre
- `aire(self)` : retourne l'aire
- `est_carre(self)` : retourne True si c'est un carré, False sinon

2.1 Créer un rectangle R1 de longueur 8 et largeur 5

2.2 Afficher son périmètre et son aire

2.3 Vérifier si c'est un carré

2.4 Créer un rectangle R2 de longueur 4 et largeur 4

2.5 Vérifier si c'est un carré

Exercice 3

Classe CompteBancaire

Créer une classe `CompteBancaire` qui gère un compte bancaire.

Attributs :

- `titulaire` : nom du titulaire
- `solde` : solde du compte (initialisé à 0)

Méthodes :

- `__init__(self, titulaire)` : constructeur
- `deposer(self, montant)` : ajoute un montant au solde
- `retirer(self, montant)` : retire un montant si le solde est suffisant
- `afficher_solde(self)` : affiche le solde actuel

3.1 Créer un compte pour "Alice"

3.2 Déposer 1000 euros

3.3 Retirer 300 euros

3.4 Afficher le solde

3.5 Tenter de retirer 900 euros (doit échouer)

Exercice 4

Classe Livre

Créer une classe `Livre` qui représente un livre.

Attributs :

- `titre` : titre du livre
- `auteur` : auteur du livre
- `nb_pages` : nombre de pages
- `prix` : prix du livre

Méthodes :

- `__init__(self, titre, auteur, nb_pages, prix)` : constructeur
- `afficher_info(self)` : affiche toutes les informations du livre
- `appliquer_remise(self, pourcentage)` : applique une remise sur le prix
- `est_volumineux(self)` : retourne True si le livre a plus de 500 pages

4.1 Créer un livre "1984" de George Orwell, 328 pages, 15.90 euros

4.2 Afficher ses informations

4.3 Appliquer une remise de 20%

4.4 Afficher le nouveau prix

4.5 Vérifier s'il est volumineux

Exercice 5

Classe Etudiant

Créer une classe `Etudiant` qui gère les notes d'un étudiant.

Attributs :

- `nom` : nom de l'étudiant
- `prenom` : prénom de l'étudiant
- `notes` : liste des notes (initialement vide)

Méthodes :

- `__init__(self, nom, prenom)` : constructeur
- `ajouter_note(self, note)` : ajoute une note à la liste
- `moyenne(self)` : calcule et retourne la moyenne des notes
- `mention(self)` : retourne la mention selon la moyenne
  - `< 10` : "Insuffisant"
  - `10-12` : "Passable"
  - `12-14` : "Assez bien"
  - `14-16` : "Bien"
  - `>= 16` : "Très bien"

5.1 Créer un étudiant “Dupont” “Jean”

5.2 Ajouter les notes : 12, 15, 13, 18, 14

5.3 Afficher la moyenne

5.4 Afficher la mention obtenue

5.5 Exercice 6 : Classe Voiture

On suppose que l’on a créé une classe `Voiture`, qui simule une voiture. Voici un résumé du contenu de cette classe :

Attributs :

- `marque` : marque de la voiture
- `modele` : modèle de la voiture
- `vitesse` : vitesse actuelle (initialisée à 0)
- `allumee` : état de la voiture (False par défaut)

Méthodes :

- `__init__(self, marque, modele)` : constructeur
- `demarrer(self)` : allume la voiture
- `arreter(self)` : éteint la voiture (vitesse revient à 0)
- `accelerer(self, increment)` : augmente la vitesse (si allumée)
- `freiner(self, decrement)` : diminue la vitesse (minimum 0)
- `afficher_etat(self)` : affiche l’état complet de la voiture

Ecrire les instructions qui permettent de :

5.6 Créer une voiture “Renault” “Clio”

5.7 Démarrer la voiture

5.8 Accélérer de 50 km/h

5.9 Accélérer encore de 30 km/h

5.10 Freiner de 20 km/h

5.11 Afficher l’état

5.12 Arrêter la voiture

Exercice 6

Classe Cercle

On suppose que l'on a créé une classe `Cercle` qui représente un cercle. Voici un résumé du contenu de cette classe :

Attributs :

- `rayon` : rayon du cercle
- `centre` : un objet Point (de l'exercice 1) représentant le centre

Méthodes :

- `__init__(self, rayon, centre)` : constructeur
- `perimetre(self)` : retourne le périmètre ($2 \times \pi \times$ rayon)
- `aire(self)` : retourne l'aire ($\pi \times$ rayon^2)
- `contient_point(self, point)` : vérifie si un point est dans le cercle
- `deplacer_centre(self, dx, dy)` : déplace le centre du cercle

Ecrire les instructions qui permettent de :

6.1 Créer un point P(0, 0)

6.2 Créer un cercle C de rayon 5 centré en P

6.3 Calculer son périmètre et son aire

6.4 Créer un point Q(3, 4)

6.5 Vérifier si Q est dans le cercle

6.6 Déplacer le centre de (2, 2)

Exercice 7

Classes Track et Playlist

7.1 Classe Track

Créer une classe `Track` qui représente une chanson.

Attributs : - `titre` : titre de la chanson - `auteur` : nom de l'artiste

Méthodes : - `__init__(self, titre, auteur)` : constructeur - `afficher(self)` : affiche la chanson sous la forme "titre - auteur"

7.2 Classe Playlist

Créer une classe `Playlist` qui gère une liste de lecture musicale.

Attributs :

- `nom` : nom de la playlist
- `tracks` : liste d'objets Track (vide au départ)

Méthodes :

- `__init__(self, nom)` : constructeur
- `ajouter_track(self, track)` : ajoute un objet Track
- `retirer_track(self, titre)` : retire une chanson par son titre si elle existe
- `nombre_tracks(self)` : retourne le nombre de chansons
- `afficher_playlist(self)` : affiche toutes les chansons numérotées
- `tracks_par_auteur(self, auteur)` : affiche toutes les chansons d'un auteur
- `est_vide(self)` : retourne True si la playlist est vide

7.3 Créer 5 objets Track avec des titres et auteurs de votre choix

7.4 Créer une playlist "Ma musique"

7.5 Ajouter les 5 tracks à la playlist

7.6 Afficher la playlist complète

7.7 Afficher toutes les chansons d'un auteur spécifique

7.8 Retirer une chanson par son titre

7.9 Afficher le nombre de chansons restantes

Exemple d'utilisation :

```
# Création de quelques tracks
t1 = Track("Die With A Smile", "Bruno Mars & Lady Gaga")
t2 = Track("Espresso", "Sabrina Carpenter")
t3 = Track("Beautiful Things", "Benson Boone")
t4 = Track("Apt.", "Rosé & Bruno Mars")
t5 = Track("Birds of a Feather", "Billie Eilish")

# Création de la playlist
titres_preferes = Playlist("Mes titres préférés")

# Ajout des tracks
titres_preferes.ajouter_track(t1)
titres_preferes.ajouter_track(t2)
titres_preferes.ajouter_track(t3)
titres_preferes.ajouter_track(t4)
titres_preferes.ajouter_track(t5)

# Affichage de la playlist
titres_preferes.afficher_playlist()

# Recherche des chansons avec Bruno Mars
titres_preferes.tracks_par_auteur("Bruno Mars")

# Retirer une chanson
titres_preferes.retirer_track("Espresso")

# Afficher le nombre de tracks restantes
print(f"Nombre de tracks : {titres_preferes.nombre_tracks()}")
```