Étape 1 : Créer une classe abstraite Compte

Objectif : Créer une classe abstraite Compte qui servira de base pour les autres types de comptes.

Exercice :

1. Utilisez le module abc pour créer une classe abstraite appelée Compte.
2. Cette classe devra définir les attributs de base :
    - numero : le numéro du compte (entier).
    - proprietaire : le nom du propriétaire du compte (chaîne de caractères).
    - solde_initial : le solde initial du compte (float).
3. Ajoutez une méthode abstraite obtenir_informations qui retournera les informations du compte sous forme de tuple. Cette méthode devra être implémentée dans les sous-classes.

```
from abc import ABC, abstractmethod
class Compte(ABC):
    def __init__(self, numero, proprietaire, solde_initial):
        self.numero = numero
        self.proprietaire = proprietaire
        self.solde_initial = solde_initial
    @abstractmethod
    def obtenir_informations(self):
        pass
```

Étape 2 : Créer des sous-classes pour les types de comptes

Objectif : Créer deux sous-classes, CompteCourant et CompteEpargne, qui hériteront de Compte et implémenteront la méthode obtenir_informations.

4. Créez une sous-classe CompteCourant qui possède un attribut supplémentaire montant_decouvert (float), représentant le montant de découvert autorisé.
    - La méthode obtenir_informations devra retourner les informations du compte courant : numéro, propriétaire, solde, type (Courant), taux d'intérêt (vide), et montant de découvert.
5. Créez une sous-classe CompteEpargne qui possède un attribut supplémentaire taux_interet (float), représentant le taux d'intérêt du compte épargne.
    - La méthode obtenir_informations devra retourner les informations du compte épargne : numéro, propriétaire, solde, type (Épargne), taux d'intérêt, et montant de découvert (vide).

Exemple de Code :

```
class CompteCourant(Compte):

    def __init__(self, numero, proprietaire, solde_initial, montant_decouvert):
        super().__init__(numero, proprietaire, solde_initial)
```

```
        self.montant_decouvert = montant_decouvert

    def obtenir_informations(self):
        return (self.numero, self.proprietaire, self.solde_initial, "Courant", "-",
self.montant_decouvert)

class CompteEpargne(Compte):
    def __init__(self, numero, proprietaire, solde_initial, taux_interet):
        super().__init__(numero, proprietaire, solde_initial)
        self.taux_interet = taux_interet

    def obtenir_informations(self):
        return (self.numero, self.proprietaire, self.solde_initial, "Épargne", self.taux_interet, "-")
```

Étape 3 : Créer une interface graphique avec Tkinter

Objectif : Développer l'interface graphique permettant à l'utilisateur de créer des comptes bancaires.

6. Créez une classe GestionComptesApp héritée de tk.Tk qui représente l'application.
7. Ajoutez les éléments suivants à l'interface :
    - Des champs pour entrer le nom du propriétaire, le solde initial, et le type de compte (Courant ou Épargne).
    - Des champs conditionnels pour afficher ou masquer les champs montant_decouvert (pour les comptes courants) et taux_interet (pour les comptes épargne).
    - Un bouton pour créer un compte.
    - Un tableau (liste) pour afficher les comptes créés.
8. Utilisez la méthode toggle_fields pour activer ou désactiver les champs en fonction du type de compte sélectionné.

Exemple de Code :

```
import tkinter as tk

class GestionComptesApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Gestion des Comptes Bancaires")
        self.geometry("800x400")

        # Variables de l'interface
        self.numero = 1
        self.type_var = tk.StringVar(value="Courant")
        self.proprietaire_var = tk.StringVar()
        self.solde_var = tk.DoubleVar()
        self.taux_interet_var = tk.DoubleVar()
        self.montant_decouvert_var = tk.DoubleVar()
```

```
        # Liste des comptes
        self.comptes = []

        # Création des widgets
        self.creer_widgets()

    def creer_widgets(self):
        # Widgets pour le formulaire
        tk.Label(self, text="Numéro:").grid(row=0, column=0, sticky="w")
        tk.Label(self, textvariable=tk.StringVar(value=str(self.numero))).grid(row=0, column=1,
sticky="w")
        tk.Label(self, text="Propriétaire:").grid(row=1, column=0, sticky="w")
        tk.Entry(self, textvariable=self.proprietaire_var).grid(row=1, column=1)
        tk.Label(self, text="Solde Initial:").grid(row=2, column=0, sticky="w")
        tk.Entry(self, textvariable=self.solde_var).grid(row=2, column=1)
        tk.Label(self, text="Euro").grid(row=2, column=2, sticky="w")
        tk.Label(self, text="Type:").grid(row=3, column=0, sticky="w")
        tk.Radiobutton(self, text="Courant", variable=self.type_var, value="Courant",
command=self.toggle_fields).grid(row=3, column=1)
        tk.Radiobutton(self, text="Épargne", variable=self.type_var, value="Épargne",
command=self.toggle_fields).grid(row=3, column=2)
        tk.Label(self, text="Taux Intérêt:").grid(row=4, column=0, sticky="w")
        self.taux_interet_entry = tk.Entry(self, textvariable=self.taux_interet_var, state="disabled")
        self.taux_interet_entry.grid(row=4, column=1)
        tk.Label(self, text="M. Découvert:").grid(row=5, column=0, sticky="w")
        self.montant_decouvert_entry = tk.Entry(self, textvariable=self.montant_decouvert_var)
        self.montant_decouvert_entry.grid(row=5, column=1)
        tk.Button(self, text="Création Compte", command=self.creer_compte).grid(row=6, column=1)

        # Tableau des comptes
        self.tableau = tk.Listbox(self, width=100, height=10)
        self.tableau.grid(row=7, column=0, columnspan=3)

    def toggle_fields(self):
        if self.type_var.get() == "Courant":
            self.taux_interet_entry.config(state="disabled")
            self.montant_decouvert_entry.config(state="normal")
        else:
            self.taux_interet_entry.config(state="normal")
            self.montant_decouvert_entry.config(state="disabled")

    def creer_compte(self):
        proprietaire = self.proprietaire_var.get()
        solde = self.solde_var.get()

        if self.type_var.get() == "Courant":
            montant_decouvert = self.montant_decouvert_var.get()
            compte = CompteCourant(self.numero, proprietaire, solde, montant_decouvert)
        else:
            taux_interet = self.taux_interet_var.get()
            compte = CompteEpargne(self.numero, proprietaire, solde, taux_interet)
```

```
    self.comptes.append(compte)
    self.numero += 1
    self.rafraichir_tableau()
    self.proprietaire_var.set("")
    self.solde_var.set(0)
    self.taux_interet_var.set(0)
    self.montant_decouvert_var.set(0)

  def rafraichir_tableau(self):
    self.tableau.delete(0, tk.END)
    for compte in self.comptes:
      self.tableau.insert(tk.END, compte.obtenir_informations())
```

9. Appeler la méthode mainloop pour démarrer l'application graphique.
10. Testez la fonctionnalité en ajoutant des comptes courants et épargne via l'interface graphique.

Objectif final :