**LA PROGRAMMATION ORIENTEE OBJECT**

* La programmation orientée objet (POO) est un paradigme de programmation basé sur les concepts de classes et d’objets.
* Les objets sont des structures qui contiennent à la fois des données (attributs) et des fonctions (méthodes) pouvant ou non modifier ces données.
* Les classes sont des modèles qui définissent les caractéristiques et le comportement d’un type d’objet.
* On peut créer des objets à partir d’une classe en utilisant le mot-clé class et en appelant le constructeur \_\_init\_\_.
* On peut accéder aux attributs et aux méthodes d’un objet en utilisant le point . suivi du nom de l’attribut ou de la méthode.
* On peut appliquer l’héritage dans le code python en utilisant les parenthèses après le nom de la classe pour indiquer la classe parente dont on hérite les attributs et les méthodes34.
* On peut surcharger les méthodes héritées en redéfinissant une méthode avec le même nom dans la sous-classe.

**Voici un exemple de code python qui illustre les concepts de base de la programmation orientée objet :**

# Définition d'une classe Animal

class Animal:

    # Constructeur qui prend en paramètre le nom et le cri de l'animal

    def \_\_init\_\_(self, nom, cri):

        # Attribut nom qui contient le nom de l'animal

        self.nom = nom

        # Attribut cri qui contient le cri de l'animal

        self.cri = cri

    # Méthode parler qui affiche le nom et le cri de l'animal

    def parler(self):

        print(f"Je suis {self.nom} et je fais {self.cri} !")

# Création d'un objet chat à partir de la classe Animal

chat = Animal("Minou", "miaou")

# Appel de la méthode parler sur l'objet chat

chat.parler()

# Définition d'une sous-classe Chien qui hérite de la classe Animal

class Chien(Animal):

    # Constructeur qui prend en paramètre le nom du chien et appelle le constructeur de la classe parente avec le cri "ouaf"

    def \_\_init\_\_(self, nom):

        super().\_\_init\_\_(nom, "ouaf")

    # Surcharge de la méthode parler pour ajouter un comportement spécifique au chien

    def parler(self):

        # Appel de la méthode parler héritée de la classe parente

        super().parler()

        # Affichage d'un message supplémentaire

        print("Je suis un bon chien !")

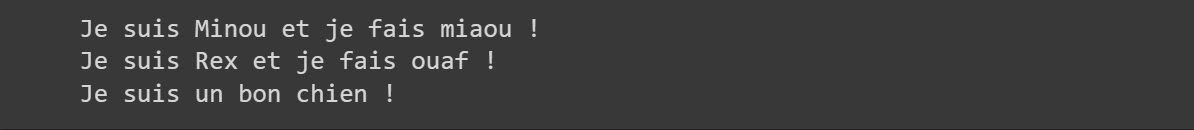
# Création d'un objet chien à partir de la sous-classe Chien

chien = Chien("Rex")

# Appel de la méthode parler sur l'objet chien

chien.parler()

**resultat :**

****

**Voici un autre exemple de code python qui illustre les concepts de base de la programmation orientée objet :**

# Définition d'une classe CompteBancaire

class CompteBancaire:

    # Constructeur qui prend en paramètre le nom du titulaire et le solde initial du compte

    def \_\_init\_\_(self, nom, solde):

        # Attribut nom qui contient le nom du titulaire du compte

        self.nom = nom

        # Attribut solde qui contient le solde du compte

        self.solde = solde

    # Méthode afficher qui affiche le nom et le solde du compte

    def afficher(self):

        print(f"Le compte de {self.nom} a un solde de {self.solde} euros.")

    # Méthode deposer qui prend en paramètre une somme à déposer sur le compte et met à jour le solde

    def deposer(self, somme):

        self.solde += somme

    # Méthode retirer qui prend en paramètre une somme à retirer du compte et met à jour le solde

    def retirer(self, somme):

        self.solde -= somme

# Création d'un objet compte1 à partir de la classe CompteBancaire avec un solde initial de 1000 euros

compte1 = CompteBancaire("Ibrahima", 100000)

# Appel de la méthode afficher sur l'objet compte1

compte1.afficher()

# Création d'un objet compte2 à partir de la classe CompteBancaire avec un solde initial de 500 euros

compte2 = CompteBancaire("Gabar", 50000)

# Appel de la méthode afficher sur l'objet compte2

compte2.afficher()

# Appel de la méthode deposer sur l'objet compte1 avec une somme de 200 euros

compte1.deposer(200)

# Appel de la méthode afficher sur l'objet compte1 pour vérifier le nouveau solde

compte1.afficher()

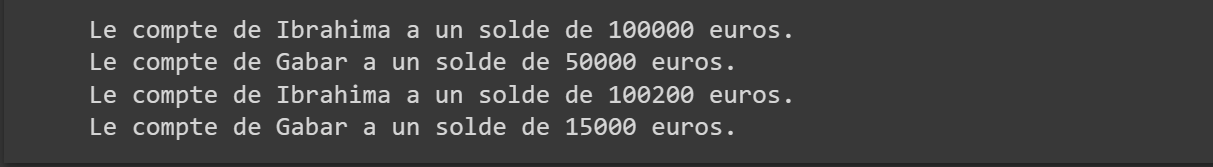
# Appel de la méthode retirer sur l'objet compte2 avec une somme de 50 euros

compte2.retirer(35000)

# Appel de la méthode afficher sur l'objet compte2 pour vérifier le nouveau solde

compte2.afficher()

**resultat :**

****