La programmation orientée objet - POO

Exercice 1 :

Ecrire un programme qui va :

1. Créer la **classe** avec le nom **Voiture**.
2. Ajouter un **constructeur** vide à cette classe.
3. Ajouter un attribut **marque** au constructeur de la classe voiture avec la valeur 'BMW'.
4. Ajouter la méthode **afficher_marque** qui affiche la marque de la voiture.
5. Ajouter un objet **voiture_1** qui instancie la classe Voiture.
6. Appeler la méthode **afficher_marque** de la classe voiture via l'objet **voiture_1**

Exercice 2 :

Ecrire un programme qui va :

1. Créer la **classe** avec le nom **Animal**.
2. Ajouter un **constructeur** à cette classe avec les attributs : **anne_naisance** et **race**.
3. Ajouter la méthode **calculer_age** qui calcule l'age de l'animal.
4. Ajouter un objet **chien** qui instancie la classe **Animal**.
5. Appeler la méthode **calculer_age** de la classe Animal via l'objet **chien**.

Exercice 3 :

1. Créer la **classe** avec le nom **ADN**.
2. Ajouter un **constructeur** à la classe **ADN** avec l'attribut **chaine.** 3. Ajouter la méthode **calculer_chaine_ADN** qui retourne la langueur de la chaine ADN.
4. Ajouter la méthode **ADN_2_ARN** qui transforme ADN vers ARN (Brin codant).

5. Ajouter un objet **adn_partie** qui instancie la classe **ADN** et ayant la chaine suivante **'ATCGATCGTTGGC'**
6. Afficher le résultat de la méthode **calculer_chaine_ADN** de l'objet créer **adn_partie**
7. Afficher le résultat de la méthode **ADN_2_ARN** de l'objet créer **adn_partie**

8. Télécharger le fichier **s_protein_sars-covid2_ref.txt** ci-joint (depuis le mini projet) et le placer dans le répertoire de votre projet python.
9. Ajouter un objet **adn_file** qui instancie la classe **ADN** et qui récupère la valeur de la chaine depuis un fichier texte **s_protein_sars-covid2_ref.txt**.
10. Enregistrer le résultat de la méthode **ADN_2_ARN** de l'objet créer **adn_file,** dans un nouveau fichier avec le nom **arn_s_protein_sars-covid2_ref.txt**

Exercice 4 :

Ecrire un programme qui va :

1. Créer la classe **Calculatrice**.
2. Ajouter un **constructeur** à cette classe qui initialise deux valeurs numériques.
3. Ajouter la méthode **calculer_somme** qui retourne la somme de deux valeurs numériques.
4. Ajouter la méthode **calculer_soustraction** qui retourne la soustraction de deux valeurs numériques.
5. Ajouter la méthode **calculer_multiplication** qui retourne la multiplication de deux valeurs numériques.
6. Ajouter la méthode **calculer_division** qui retourne la division de deux valeurs numériques. Il faut ajouter l'exception qui protège la division par Zéro. 7. Pour chaque méthode des questions 3, 4, 5 et 6, ajouter une condition qui vérifie si les valeurs entrées sont des valeurs numériques.

8. Après la définition de la classe **Calculatrice**, demander à l'utilisateur de saisir deux nombres depuis son clavier.
9. Créer un nouvel objet **joli_calculatrice** à partir de la classe **Calculatrice,** qui prend en paramètre les deux valeurs saisis dans la question 8.
10. Afficher le résultat de la somme, soustraction, multiplication, division des deux de l'objet créer via objet **joli_calculatrice** de la question 9, en se basant successivement des méthodes **calculer_somme, calculer_soustraction, calculer_multiplication, calculer_division.**

Exercice 5 : (héritage)

Ecrire un programme qui va :

1. Créer la classe **Véhicule**.
2. Ajouter un **constructeur** à la classe **Véhicule** avec l'attribut **nbr_kilomètre**.
3. Ajouter la méthode **rouler_kilomètre** qui possède le paramètre **kilometre_parcourus** et retourne le total des kilomètres parcourus (**nbr_kilomètre + kilometre_parcourus**).
4. Ajouter la méthode **calculer_roue** qui retourne la chaine 'le nombre des roues est indéfini'

5. Créer une nouvelle classe **Camion** qui hérite de la classe **Véhicule**.
6. Ajouter la méthode **calculer_roue** qui retourne la valeur 6.

7. Ajouter un objet **petit_véhicule** qui instancie la classe **Véhicule** avec le **nbr_kilomètre** 30000**.**
8. Afficher le résultat de la méthode **rouler_kilomètre** de l'objet **petit_véhicule** avec le paramètre 1000**.**
9. Afficher le résultat de la méthode **calculer_roue** de l'objet **petit_véhicule.**

10. Ajouter un objet **petit_camion** qui instancie la classe **Camion** avec le **nbr_kilomètre** 900000**.**
11. Afficher le résultat de la méthode **kilometre_parcourus** de l'objet **petit_camion** avec le paramètre 20000**.**
12. Afficher le résultat de la méthode **calculer_roue** de l'objet **petit_camion.**

Exercice 6 :

Ecrire un programme qui va :

1. Créer la classe **ListeUtilitaire**.
2. Ajouter un **constructeur** à cette classe qui initialise une liste de type privée.
3. Ajouter la méthode de type **getter** qui retourne la liste.
4. Ajouter la méthode de type **setter** qui modifie la liste.
5. Ajouter la méthode qui calcule la langueur de la liste.
6. Ajouter la méthode qui affiche les éléments de la liste.
7. Ajouter la méthode qui trie les éléments de la liste.
8. Ajouter la méthode qui vide les éléments de la liste.

9. Ajouter l'objet **ma_liste** qui initialise la liste ayant les éléments : 'ADN', 'ARN', 'Covid', 'Protéine E', 'Anzime'.
10. Appeler l'ensemble des méthodes créés.

Exercice 7 :

Ecrire un programme qui va :

1. Créer la classe ARN.
2. La classe ARN devra hériter de la classe ADN (de l'exercice 3). 3. Redéfinir le constructeur de la classe ARN (méthode __init__(self)) pour initialiser une chaine ARN.
4. Ajouter la méthode de type **getter** qui retourne la chaine ADN. 5. Ajouter la méthode de type **setter** qui modifie la chaine ADN. 6. Ajouter la méthode ARN_2_ADN, qui transforme ARN vers ADN (Brin codant).

Exercice Classe Rectangle :

1. Ecrire une classe Rectangle en langage Python, permettant de construire un rectangle dotée d'attributs **longueur** et **largeur.**
2. Créer une **méthode Perimetre()** permettant de calculer le périmètre du rectangle et une **méthode Surface()** permettant de calculer la surface du rectangle
3. Créer une **classe fille Parallelepipede héritant** de la **classe Rectangle** et dotée en plus d'un **attribut hauteur** et d'une autre **méthode Volume()** permettant de calculer le volume du Parallélépipède.

(Les attributs des deux classes sont privés)
Périmètre = 2*(longueur + largeur)
Surface = longueur * largeur
Volume = longueur * largeur * hauteur

```
# 1)
class Rectangle:
    def __init__(self, longueur, largeur):
        self.longueur = longueur
        self.largeur = largeur
# 2)
    def Perimetre(self):
        return 2*(self.longueur+self.largeur)
    def Surface(self):
        return self.longueur*self.largeur


monRectangle = Rectangle(7 , 5)
print("La surface de mon rectangle est : ", monRectangle.Surface())
print("Le perimètre de mon rectangle est ", monRectangle.Perimetre())
class Parallelepipede(Rectangle):
    def __init__(self, longueur, largeur, hauteur):
        Rectangle.__init__(self, longueur, largeur)
        self.hauteur = hauteur
    def Volume(self):
        return self.longueur*self.largeur*self.hauteur

monParallelepipede = Parallelepipede(7, 5, 2)
print("Le volume de mon Parallelepipede est :",monParallelepipede.Volume())
```