Quiz : Programmation Orientée Objet en Python

1. Vocabulaire de la programmation objet

Associez chaque définition au bon terme parmi: **classe, instance, attribut, méthode**

a) Une fonction définie à l'intérieur d'une classe et qui peut agir sur ses instances.
b) Une variable associée à une instance et qui stocke une valeur propre à celle-ci.
c) Un plan ou un modèle qui définit des attributs et des méthodes communes à plusieurs objets.
d) Un objet créé à partir d'une classe.

2. Création d'une classe et d'instances

Que va afficher le code suivant?

```
class Voiture:
    def __init__(self, marque, vitesse):
        self.marque = marque
        self.vitesse = vitesse

ma_voiture = Voiture("Toyota", 120)
print(ma_voiture.marque)
print(ma_voiture.vitesse)
```

a) Toyota et 120
b) Erreur car la classe n'a pas de méthode **init**
c) Rien, car on n'a pas défini de méthode pour afficher les informations
d) Erreur car self n'a pas été utilisé dans **init**

3. Utilisation de la notation pointée

(4 points)

Quel est le résultat de ce programme?

```
class Chien:
    def __init__(self, nom):
        self.nom = nom

    def aboyer(self):
        return f"{self.nom} aboie!"

chien1 = Chien("Rex")
print(chien1.aboyer())
```

a) "aboyer"
b) "Rex aboie!"
c) Erreur car self.nom est mal utilisé
d) Rien, car la méthode aboyer ne contient pas d'instruction print

4. Compréhension du paramètre self

Pourquoi utilise-t-on **self** dans une méthode de classe?

a) Pour indiquer qu'il s'agit d'une fonction globale
b) Pour faire référence à l'instance courante et accéder à ses attributs et méthodes
c) Ce n'est pas obligatoire, on peut utiliser n'importe quel mot à la place
d) Pour initialiser automatiquement tous les objets créés

Mini-projet : Gestion d’un Zoo

Objectif

Créer un programme qui permet de gérer des animaux dans un zoo en utilisant la programmation orientée objet.

Étape 1 : Définir la classe `Animal`

Crée une classe `Animal` qui a les caractéristiques suivantes :

- **Attributs** : `nom`, `espece`, `age`
- **Méthodes** :
  - `se_presenter()` : affiche une phrase du type `"Je suis Simba, un lion de 5 ans."`
  - `vieillir()` : augmente l'âge de l'animal de 1 an

Étape 2: Créer des instances et tester les méthodes

- Instancier au moins **trois animaux** différents
- Afficher leur présentation
- Faire vieillir un animal et afficher son nouvel âge

Exemple d’utilisation :

```
lion = Animal("Simba", "lion", 5)
elephant = Animal("Dumbo", "éléphant", 10)
zebre = Animal("Marty", "zèbre", 7)

print(lion.se_presenter())
print(elephant.se_presenter())
print(zebre.se_presenter())

lion.vieillir()
print(f"{lion.nom} a maintenant {lion.age} ans.")
```

Résultat attendu :

```
Je suis Simba, un lion de 5 ans.
Je suis Dumbo, un éléphant de 10 ans.
Je suis Marty, un zèbre de 7 ans.
Simba a maintenant 6 ans.
```

Étape 3 : Ajouter une classe `Zoo`

Ajouter une classe `Zoo` qui permet de gérer plusieurs animaux :

- Attribut : `animaux` (une liste d’instances `Animal`)
- Méthodes :
  - `ajouter_animal(animal)` : ajoute un animal au zoo
  - `afficher_animaux()` : affiche tous les animaux du zoo
- Ajouter une méthode pour compter le nombre total d’animaux dans le zoo.