Questionnaire d'auto-évaluation — L'héritage en Python

/home/phil/Documents/LMG/STS_A_2025-2027/PYTHON_STS2/C_classe_(heritage)/02_heritage_introduction

1. QCM de compréhension (1 seule bonne réponse par question)

Consigne : Identifier la bonne réponse sans exécuter le code.

Question 1

Que signifie `class Chien(Animal):` ?

- **A.** `Chien` est une copie de `Animal`.
- **B.** `Chien` hérite de `Animal` : il récupère ses attributs et méthodes.
- **C.** `Animal` hérite de `Chien`.
- **D.** `Chien` remplace `Animal`.

Question 2

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

class Chien(Animal):
    pass

rex = Chien("Rex")
```

Que se passe-t-il quand on écrit `rex = Chien("Rex")` ?

- **A.** Erreur : `Chien` n'a pas de `__init__`.
- **B.** `rex` est créé mais `rex.nom` n'existe pas.
- **C.** Python utilise le `__init__` de `Animal`, donc `rex.nom == "Rex"`.
- **D.** Il faut obligatoirement écrire un `__init__` dans `Chien`.

Question 3

```
class Animal:
    def faire_du_bruit(self):
        return "un bruit"

class Chien(Animal):
    def faire_du_bruit(self):
        return "un aboiement"

rex = Chien()
print(rex.faire_du_bruit())
```

Qu'affiche ce programme ?

- **A.** `un bruit`
- **B.** `un aboiement`
- **C.** `un bruit` puis `un aboiement`
- **D.** Erreur

Question 4

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

class Chat(Animal):
    def __init__(self, nom, couleur):
        super().__init__(nom)
        self.couleur = couleur
```

À quoi sert la ligne `super().__init__(nom)` ?

- **A.** À créer un nouvel animal.
- **B.** À appeler le constructeur de la classe mère `Animal` pour initialiser `self.nom`.
- **C.** À copier la classe `Animal`.
- **D.** À rien, c'est facultatif.

Question 5

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

class Chat(Animal):
    def __init__(self, nom, couleur):
        self.couleur = couleur
        self.nom = nom
```

Cette version fonctionne, mais quelle est la différence avec l'usage de `super()` ?

- **A.** Aucune, c'est identique.
- **B.** Elle n'appelle pas le constructeur de la mère : si `Animal` change, `Chat` ne suivra pas.
- **C.** Elle est plus rapide.
- **D.** Elle provoque une erreur.

Question 6

```
class Animal:
    def faire_du_bruit(self):
        return f"{self.nom} fait un bruit."

class Chien(Animal):
    def faire_du_bruit(self):
        return f"{self.nom} aboie."

animaux = [Chien("Rex"), Chien("Médor")]
for a in animaux:
    print(a.faire_du_bruit())
```

Qu'affiche ce programme ?

- **A.** `fait un bruit.` deux fois.
- **B.** `Rex aboie.` puis `Médor aboie.`
- **C.** Erreur : on ne peut pas mettre des objets dans une liste.
- **D.** `Rex fait un bruit.` puis `Médor aboie.`

Question 7

Que renvoie `type(rex)` si `rex = Chien("Rex")` ?

- **A.** `<class 'Animal'>`
- **B.** `<class 'Chien'>`
- **C.** `"Chien"`
- **D.** Une erreur

Question 8

```
class Chien(Animal):
    def faire_du_bruit(self):
        return super().faire_du_bruit() + " Et il remue la queue."
```

Que fait `super().faire_du_bruit()` ici ?

- **A.** Elle remplace la méthode de la mère.
- **B.** Elle appelle la méthode `faire_du_bruit` de `Animal` en plus de la redéfinition.
- **C.** Elle crée une nouvelle méthode.
- **D.** Elle provoque une erreur.

2. Vrai / Faux justifié

Consigne : Répondre V ou F, puis justifier en une phrase. La justification compte autant que la réponse.

1) Une classe fille doit obligatoirement redéfinir toutes les méthodes de sa mère.
2) super().__init__()` peut être appelé n'importe où dans la classe fille.
3) Si une classe fille définit un `__init__`, elle doit appeler `super().__init__()` pour conserver les attributs de la mère.
4) Une méthode redéfinie dans la classe fille remplace celle de la classe mère pour les objets de la classe fille.
5) Une classe peut hériter de plusieurs classes mères en Python (`class X(A, B):`)
6) pass` dans une classe fille signifie qu'elle n'hérite de rien.
7) Animal.__init__(self, nom)` et `super().__init__(nom)` sont équivalents dans une classe fille.

3. Auto-évaluation par compétences

Consigne : Pour chaque compétence, coche la case qui correspond à ton niveau **réel** (pas à ce que tu aimerais). Puis note la question du QCM qui t'a fait douter. Placer ✔️ dans une des 3 cases.

Compétence	Je ne sais pas	Je sais avec de l'aide	Je sais seul	❓ Question liée
Écrire class Fille(Mere):				
Expliquer ce dont une classe fille hérite				
Utiliser `pass` quand je ne veux rien ajouter				
Redéfinir une méthode				
Étendre un constructeur avec `super().__init__()`				
Ajouter un attribut spécifique à la classe fille				
Appeler une méthode de la classe mère depuis la classe fille				
Reconnaître le polymorphisme dans un code				
Utiliser `type()` pour vérifier la classe				

4. Exercice « miroir » (auto-correction)

Consigne : Écris **de mémoire** une classe `Oiseau` qui hérite de `Animal`, possède un attribut `envergure`, et une méthode `faire_du_bruit()` qui renvoie `"<nom> chante."`.

Ne regarde pas tes notes. À compléter :

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

    def faire_du_bruit(self):
        return f"{self.nom} fait un bruit."
```

```
class ……..:
    ……
    ……
    ……
    ……
    ……..
    ……..
    ……..
    ……..
    ……..
    ……..
```