Evaluation sur la création de classe

/home/phil/Documents/LMG/STS_A_2025-2027/PYTHON_STS2/01_eval2_GESTION_PARC_INFORMATIQUE_

1. Contexte

Un établissement scolaire souhaite gérer les ordinateurs présents dans ses différentes salles.

Chaque ordinateur est identifié par un identifiant unique et possède différentes caractéristiques : quantité de mémoire vive en Giga octets, capacité de stockage en Giga octets, salle dans laquelle il se trouve et état de fonctionnement.

Afin de suivre les changements effectués sur les ordinateurs, l'établissement souhaite également conserver un **historique des opérations réalisées**.

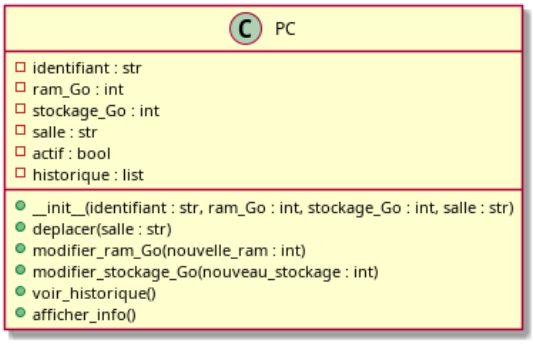
Vous allez créer une classe `PC` permettant de représenter et de gérer un ordinateur.

2. Création de la classe `PC` /5 points

Créer une classe `PC` possédant les attributs suivants :

Attribut	Type	Description
`identifiant`	`str`	Identifiant unique du PC
`ram_Go`	`int`	Quantité de mémoire vive en Go
`stockage_Go`	`int`	Capacité de stockage en Go
`salle`	`str`	Salle dans laquelle se trouve le PC
`actif`	`bool`	Indique si le PC est actif
`historique`	`list`	Liste des opérations effectuées

Travail demandé



Le but est d'écrire le constructeur `__init__` de la classe.

Le constructeur doit recevoir les informations suivantes :

- l'identifiant ;
- la quantité de RAM ;
- la capacité de stockage ;
- la salle .

► Rappeler le type de l'attribut actif. Combien d'état peut-il prendre ? L'attribut actif du PC sera fixé à True si la salle est différente de « reserve ».

► L'attribut `historique` doit être initialisé comme une liste vide.

Exemples de création d'un objet `PC` permettant de vérifier que votre classe fonctionne.:

```
pc1 = PC("PC-001", 8, 512, "E03B")
pc2 = PC("PC-002", 16, 512, "F10")
```

3. Déplacer un PC /3 points

Ajouter à la classe la méthode : `deplacer(salle: str)`

Cette méthode doit permettre de déplacer le PC dans une autre salle.

Lorsqu'un PC est déplacé :

- son attribut `salle` doit être modifié ;
- Penser à modifier la propriété *actif* du PC si celui-ci sort ou retourne dans la `"reserve"`
- une information doit être ajoutée dans l'historique.

Par exemple, si le PC était en salle `D01` et est déplacé en salle `F10`, l'historique pourra contenir : `Déplacement de salle D01 vers salle F10`

a. Question

Écrire la méthode `deplacer(salle)`.

4. Modifier la mémoire RAM /2 points

Ajouter à la classe la méthode : `modifier_ram_Go(nouvelle_ram: int)`

Cette méthode doit :

1. modifier la quantité de RAM du PC ;
2. enregistrer la modification dans l'historique.

Par exemple : `Modification de la RAM : 8 Go -> 16 Go`

a. Question

Écrire la méthode `modifier_ram_Go(nouvelle_ram)`.

5. Modifier le stockage /2 points

Ajouter à la classe la méthode : `modifier_stockage_Go(nouveau_stockage: int)`

Cette méthode doit :

1. modifier la capacité de stockage du PC ;
2. enregistrer la modification dans l'historique.

Par exemple : `Modification du stockage : 512 Go -> 1000 Go`

a. Question

Écrire la méthode `modifier_stockage_Go(nouveau_stockage)`.

6. Consulter l'historique /3 points

Ajouter à la classe la méthode : `voir_historique()`

Cette méthode doit afficher toutes les opérations enregistrées dans l'historique du PC.

Si aucune opération n'a encore été effectuée, le programme doit afficher un message adapté, par exemple : `Aucune opération enregistrée.`

a. Question

Écrire la méthode `voir_historique()`.

7. Afficher les informations du PC /2 points

Ajouter à la classe la méthode : `afficher_info()`

Cette méthode doit afficher de manière lisible les informations du PC. Par exemple :

```
----- Informations du PC -----
Identifiant : PC-001
RAM : 16 Go
Stockage : 1000 Go
Salle : E03B
Actif : Oui
```

a. Question

Écrire la méthode `afficher_info()`.

8. Programme de test /3 points

Écrire un programme permettant de tester votre classe `PC`. Votre programme doit :

1. créer un PC pc1 avec les caractéristiques suivantes :
   - identifiant : `PC-001`
   - RAM : `8 Go`
   - stockage : `512 Go`
   - salle : `D22`
2. afficher ses informations ;
3. déplacer le PC dans la salle `F21` ;
4. modifier sa RAM à `16 Go` ;
5. modifier son stockage à `1000 Go` ;
6. afficher à nouveau ses informations ;
7. afficher son historique.

Exemple de déroulement attendu

```
----- Informations du PC -----
Identifiant : PC-001
RAM : 8 Go
Stockage : 512 Go
Salle : D22
Actif : Oui
----- Informations du PC -----
Identifiant : PC-001
RAM : 16 Go
Stockage : 1000 Go
Salle : F21
Actif : Oui
----- Historique -----
Déplacement de D22 vers F21
Modification de la RAM : 8 Go -> 16 Go
Modification du stockage : 512 Go -> 1000 Go
```

9. Bonus – Validation des données (+2 points)

Objectif : améliorer la classe afin d'empêcher les valeurs incorrectes. Par exemple :

- la RAM doit être un entier strictement positive ;
- le stockage doit être un entier strictement positif ;
- une salle doit être une chaîne non vide.

Lorsqu'une valeur est incorrecte, le programme doit afficher un message d'erreur et ne pas modifier l'objet. Exemple : `Erreur : la quantité de RAM doit être supérieure à 0.`

Choisir une des trois amélioerations proposées et indiquer les lignes à ajouter dans le code (et où).

Tableau rappelant les instructions du langage Python

Instruction / élément	Utilité	Exemple
`class`	Déclarer une classe	`class PC:`
`__init__`	Constructeur appelé lors de la création d'un objet	`def __init__(self, ...):`
`self`	Désigner l'objet courant	`self.ram_Go`
`=`	Affecter une valeur	`self.ram_Go = ram_Go`
`str`	Type chaîne de caractères	`"PC-001"`
`int`	Type entier	`16`
`bool`	Type booléen	`True` / `False`
`list`	Type liste	`[]`
`def`	Déclarer une méthode ou fonction	`def afficher_info(self):`
`if`	Effectuer un test conditionnel	`if self.salle == "reserve":`
`else`	Cas contraire d'un `if`	`else:`
`elif`	Ajouter une autre condition	`elif ...:`
`==`	Tester l'égalité	`self.salle == "reserve"`
`!=`	Tester la différence	`self.salle != "reserve"`
`>` / `<`	Comparer des nombres	`ram_Go > 0`
`append()`	Ajouter un élément à une liste	`self.historique.append(...)`
`for`	Parcourir une liste	`for operation in self.historique:`
`in`	Tester l'appartenance	`if "PC" in texte:`
`print(f"" )`	Afficher une information	`print(f"{self.identifiant}")`
`str()`	Convertir une valeur en chaîne	`str(self.ram_Go)`
`len()`	Obtenir la longueur d'une liste	`len(self.historique)`
`and`	Combiner deux conditions	`ram > 0 and stockage > 0`
`or`	Tester l'une ou l'autre condition	`salle == "r" or salle == "D01"`
`not`	Inverser une condition	`not self.actif`