Université de Carthage Institut Préparatoire aux Études d'Ingénieur de Nabeul Département : Mathématiques	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2025/2026 Filière : SM/SP/ST Niveau d'étude : 2ème année Semestre : 1 Nombre de pages : 4 pages Date : 28/10/2025 Durée : 1h30

DS Informatique N°1

Remarque importante :
Avant de rédiger le code de chaque fonction ou méthode, consultez attentivement l'**Annexe** (en fin de sujet). Elle présente des exemples d'appels et de résultats attendus, ce qui vous aidera à mieux comprendre le comportement des fonctions et à vous assurer que votre implémentation respecte les spécifications demandées.

Problème : Modélisation de la consommation énergétique

Dans ce problème, on s'intéresse à la modélisation de la consommation d'énergie (électricité) d'appareils domestiques ou industriels.
Chaque appareil possède une consommation qui varie dans le temps, que l'on peut approximer par un polynôme du temps :

$$C(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

avec

- $C(t)$ est la consommation cumulée à l'instant t (exprimée en kilowattheures, **kWh**),
- a_i sont des coefficients réels
- t représente le temps écoulé depuis le démarrage de l'appareil (de type `float`), exprimé en millisecondes. Ainsi, $t = 0$ correspond au moment où l'appareil est mis en marche, et t augmente au cours du fonctionnement.

Afin d'approximer les variations dans le temps et de calculer facilement la consommation d'un ou plusieurs appareils, on propose de créer les trois classes suivantes :

— `Polynome` : pour représenter la consommation d'un appareil en fonction du temps ;
— `Appareil` : pour gérer la consommation complète d'un appareil (polynôme, dérivée instantanée) ;
— `GestionnaireAppareils` : pour gérer plusieurs appareils, analyser la consommation totale et détecter les appareils les plus énergivores.

Partie 1 : Classe `Polynome` (5 pts)

Définir une classe `Polynome` avec :

1.1. Constructeur et attributs :

— un attribut de classe `PRECISION = 3`;

— un constructeur `__init__(self, coeffs)` où `coeffs` est la liste des coefficients du polynôme :

$$C(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

Exemple : `coeffs = [0,2,0.5]` correspond à $C(t) = 0 + 2t + 0.5t^2$.

1.2. Définir une méthode `setCoeffs(self, coeffs)` permettant de modifier la liste des coefficients du polynôme. Cette méthode doit vérifier que la liste passée en paramètre est **non vide** et que **tous les coefficients sont des nombres réels** (de type `int` ou `float`). Si l'une de ces conditions n'est pas respectée, elle doit lever une exception appropriée de type `ValueError` ou `TypeError`.

1.3. Définir une méthode d'évaluation `__call__(self, t)` qui calcule et retourne la consommation cumulée au temps t.

1.4. Définir une méthode `derivee(self)` qui retourne un nouvel objet `Polynôme` correspondant à la consommation instantanée.

1.5. Définir une méthode de représentation `__repr__(self)` qui retourne une chaîne formatée du polynôme sous la forme $C(t) = a0 + a1 * t + a2 * t^2 + ...$, avec chaque coefficient affiché avec 3 chiffres après la virgule selon la constante PRECISION.

1.6. Définir une méthode `__eq__(self, other)` qui compare deux polynômes pour vérifier l'égalité des coefficients.

Partie 2 : Classe `Appareil` (4 pts)

Définir une classe `Appareil` avec :

2.1. Constructeur `__init__(self, nom, coeffs)` avec les attributs d'instances suivants :

- `nom` : le nom de l'appareil
- `conso` : le `Polynome` consommation cumulée de cet appareil crée à partir de la liste des coefficients passée en paramètres.
- `conso_inst` : la consommation instantanée, créée à partir de la dérivée du polynôme de consommation de cet appareil.

2.2. Méthode `evalConsommation(self, t)` qui retourne un tuple `(cumul, instant)` représentant la consommation cumulée et instantanée au temps `t`.

2.3. Méthode `setCoeffs(self, coeffs)` : qui permet de modifier la liste des coefficients du polynôme `conso` et déclenche une erreur de type `ValueError` si la liste des coefficients est invalide ou vide.

2.4. Méthode `__str__(self)` qui retourne une chaîne descriptive de l'appareil.

2.5. Méthode `__eq__(self, other)` qui vérifie si deux appareils sont égaux en comparant leurs polynômes de consommation.

Partie 3 : Classe `GestionnaireAppareils` (7 pts)

Définir une classe `GestionnaireAppareils` avec :

3.1. Constructeur `__init__(self, Dapp)` où `Dapp` est un dictionnaire sous la forme de `{nom_appareil : [coeffs]}` et avec `LA` un attribut d'instances représentant la liste d'`Appareils` à gérer.

3.2. Méthode `__len__(self)` qui retourne le nombre total d'appareils.

3.3. Méthode `__getitem__(self, nom)` qui retourne un appareil à partir de son nom.

3.4. Méthode `__contains__(self, appareil)` qui vérifie si l'appareil passé en paramètre appartient au gestionnaire d'appareils.

3.5. Méthode `__add__(self, other)` qui selon le type de other, soit elle ajoute les appareils du deuxième gestionnaires d'appareils au premier, soit elle ajoute un nouvel appareil si le nom n'existe pas déjà, ou bien elle lève une exception `TypeError` si le type de l'objet other est invalide.

3.6. Méthode `supprimerAppareil(self, nom)` : Supprime un appareil à partir du nom s'il existe.
3.7. Méthode `__sub__(self, other)` qui supprime des appareils selon le type de other : si other est un GestionnaireAppareils, elle supprime ses appareils du gestionnaire courant ; si other est une chaîne (str), elle supprime l'appareil portant ce nom ; sinon, elle lève une exception TypeError.
3.8. Méthode `getConsoMax(self, seuil, t)` qui retourne la liste des appareils dont la consommation instantanée à l'instant t dépasse un seuil donné.
3.9. Méthode `getConsoTotale(self, t)` : retourne la consommation totale à l'instant t, c'est-à-dire la somme des consommations cumulées de tous les appareils gérés par le gestionnaire.
3.10. Méthode `__str__(self)` qui retourne une description complète du gestionnaire d'appareils (voir Annexe pour un exemple d'exécution).

Partie 4 : Lecture des données et exécution du programme (4 pts)

Cette partie a pour objectif d'importer automatiquement les coefficients des polynômes de consommation depuis un fichier texte afin de créer les objets `Appareil` et le gestionnaire associé.
Cette approche rend le programme plus flexible, plus rapide à exécuter et facilement adaptable à un grand nombre d'appareils.

Chaque ligne du fichier contient le nom de l'appareil suivi des coefficients du polynôme, séparés par des points-virgules (voir annexe).

Travail demandé :

4.1. Écrire une fonction `charger_appareils(fichier)` qui lit le contenu du fichier texte dont le nom est passé en paramètres et retourne un dictionnaire de la forme `{nom_appareil: [coeffs]}`.
4.2. Écrire le script Python permettant de
a. À partir du dictionnaire retourné par la fonction précédente, créer un objet de type `GestionnaireAppareils` contenant l'ensemble des appareils.
b. Saisir le nom d'un appareil, un seuil et un instant t ;
c. Afficher la consommation cumulée et la consommation instantanée de cet appareil à l'instant t, s'il existe dans le gestionnaire.
d. Afficher la liste des appareils dont la consommation instantanée dépasse un seuil donné à l'instant t.
e. Afficher la liste des appareils triés en ordre décroissant par consommation cumulée au temps t.

Annexe : Exemples d'utilisation et résultats attendus

```
# ============================================================
# Partie 1 : Classe Polynome
# ============================================================
>>> p = Polynome([0, 2, 0.5])
>>> print(p) # C(t) = 0.000 + 2.000*t + 0.500*t^2
>>> p(3) # 10.5
>>> p_der = p.derivee()
>>> print(p_der) # C(t) = 2.000 + 1.000*t
>>> p2 = Polynome([0, 2, 0.5])
>>> p == p2 # True
>>> p == p_der # False
# ============================================================
# Partie 2 : Classe Appareil
# ============================================================
```

```
>>> a = Appareil("Frigo", [1, 0.5, 0.1])
>>> print(a)
Appareil Frigo :
Consommation cumulée : C(t) = 1.000 + 0.500*t + 0.100*t^2
Consommation instantanée : C'(t) = 0.500 + 0.200*t
>>> cumul, instant = a.evalConsommation(3) # Cumul: 2.4  Instant: 1.1
>>> a2 = Appareil("Frigo", [1, 0.5, 0.1])
>>> a == a2 # True
>>> try:
...  a.setCoeffs([2, 'x'])
 except Exception as e:
...  print(e)
Tous les coefficients doivent être des nombres réels
# ============================================================
# Partie 3 : Classe GestionnaireAppareils
# ============================================================
>>> D = {"Frigo": [1,0.5,0.1], "Chauffage": [0,2,0.2]}
>>> G = GestionnaireAppareils(D)
>>> print(len(G)) # 2
>>> "Frigo" in G # True
>>> G["Frigo"].evalConsommation(3) # (2.4, 1.1)
>>> print(G)
Gestionnaire d'appareils :
- Appareil Frigo :
 Consommation cumulée : C(t) = 1.000 + 0.500*t + 0.100*t^2
 Consommation instantanée : C'(t) = 0.500 + 0.200*t
- Appareil Chauffage :
 ...
>>> G2 = GestionnaireAppareils({"Four": [1, 1, 0.1]})
>>> G = G + G2     # ajout d'un gestionnaire
>>> print(len(G)) # 3
>>> G = G + Appareil("Climatiseur", [0, 1.5, 0.3])
>>> print(len(G)) # 4
>>> G = G - "Four"  # suppression d'un appareil
>>> print(len(G)) # 3
>>> G.supprimerAppareil("Frigo")
>>> "Frigo" in G # False
# Affiche la liste des appareils dont la conso instantanée > 1.5
>>> G.getConsoMax(seuil = 1.5, t = 3) # ['Chauffage', 'Climatiseur']
```

Données du fichier "appareils.txt"

```
Frigo; 1; 0.5; 0.1
Chauffage; 0; 2; 0.2
MachineLaver; 0.5; 0.7; 0.05
Climatiseur; 0; 1.5; 0.3
Four; 1; 1; 0.1
```