

Année universitaire : 2025-2026

Examen de fin du premier semestre

Epreuve d'informatique

Classes : MP1, PC1 et PT1

Durée : 2h

N.B. :

- Toutes les réponses doivent être rédigées en Python. Il faut numéroté les questions.
- L'indentation dans les instructions Python sera prise en considération dans le barème.
- Vous trouvez, en Annexe, quelques méthodes prédéfinies sur les conteneurs.

Exercice 1: (6 points)

L'objectif de cet exercice est d'analyser les informations transmises par un réseau de capteurs installés dans un bâtiment intelligent. Ces capteurs mesurent les différents paramètres environnementaux comme la température, l'humidité et le mouvement. Toutes ces mesures sont regroupé dans la liste Data dont chaque élément est une structure de type dictionnaire.

Chaque dictionnaire dans la liste Data contient les clés suivantes :

- id : identifiant unique du capteur
- type : type de mesure (température, humidité, mouvement, etc.)
- valeur : valeur relevée par le capteur (ex. température en °C, % d'humidité ou indication de mouvement(0 ou 1))
- timestamp : date et heure de la mesure
- localisation : emplacement où la mesure a été effectuée

Exemple :

```
Data = [ {"id": 101, "type": "température", "valeur": 22.5,
"timestamp": "2025-01-15 10:30:00", "localisation":
"salle_serveurs"}, {"id": 102, "type": "humidité", "valeur":
45.2, "timestamp": "2025-01-15 10:30:00", "localisation":
"salle_serveurs"}, {"id": 103, "type": "température", "valeur":
24.1, "timestamp": "2025-01-15 10:30:00", "localisation":
"bureau_open_space"}, {"id": 104, "type": "mouvement",
"valeur": 1, "timestamp": "2025-01-15 10:30:00", "localisation":
"entrée_principale"} ]
```

Travail demandé :

Étant donnée la liste Data, écrire un programme en Python qui permet de :

1. Calculer et afficher la moyenne des valeurs de la température étant enregistrées dans la localisation "salle_serveurs".

Attention au cas où aucune location "salle_serveurs" n'est trouvée dans la liste Data.

2. Afficher tous les dictionnaires se trouvant dans la liste Data étant considérés comme anormaux, selon les critères suivants :

- Température $< 18^{\circ}\text{C}$ ou $> 26^{\circ}\text{C}$
- Humidité $< 35\%$ ou $> 55\%$
- Mouvement détecté (c'est-à-dire ayant valeur=1)

3. Créer puis afficher les ensembles suivants:

- L'ensemble E1 contient toutes les localisations.
- L'ensemble E2 contient les localisations où au moins un mouvement a été détecté.
- L'ensemble E3 contient les localisations où aucun mouvement n'a été détecté.

Exercice 2: (5 points)

Une entreprise souhaite classer automatiquement des commentaires de ses clients selon leur tonalité: positive, négative ou neutre. On suppose que les commentaires sont fournis dans une liste de chaînes de caractères.

Exemple :

```
Comment = [ "Livraison correcte mais un peu lente", "Le produit est mauvais et défectueux", "Le service est excellent et rapide", "Très bon rapport qualité prix", "Je suis déçu par la commande", "un bon produit mais Mauvais emballage et retard", "Tout était PARFAIT"]
```

Travail demandé

Étant donnée la liste Comment, écrire un programme en Python qui permet de :

1. Créer une nouvelle liste CommentMin contenant tous les commentaires convertis en minuscules.
2. Étant donnée les conteneurs suivants:

```
Positifs = {"excellent", "bon", "satisfait", "parfait", "rapide"}  
Negatifs = {"mauvais", "défectueux", "déçu", "retard", "lente"}  
Classification = {"positif": [], "négatif": [], "neutre": []}
```

Pour chaque commentaire de la liste CommentMin créée précédemment:

- ✓ Calculer Npos= le nombre de mots du commentaire appartenant à l'ensemble Positifs,
- ✓ Calculer Nneg= le nombre de mots du commentaire appartenant à l'ensemble Negatifs,
- ✓ Déterminer la classe du commentaire selon les valeurs trouvées dans Npos et Nneg, telles que:
 - classe="positif" si $N_{pos} > N_{neg}$
 - classe="négatif" si $N_{pos} < N_{neg}$
 - classe="neutre" sinon
- ✓ Ajouter le commentaire dans le dictionnaire Classification, selon la classe déterminée précédemment. Par exemple, si le commentaire est de classe="positif", alors on l'ajoute à la liste associée à la clé "positif" dans le dictionnaire.

Exercice 3: (5 points)

Une société de transport routier réalise quotidiennement des livraisons entre différentes villes du pays. Pour mieux organiser leurs tournées, les chauffeurs souhaitent estimer le temps total nécessaire pour parcourir un chemin complet reliant plusieurs villes.

On se propose de représenter les routes par un dictionnaire nommé `Villes`, où :

- Les clés sont les villes de départ.
- Les valeurs correspondent à un dictionnaire où ses clés sont les villes directement accessibles à la ville de départ et ses valeurs sont décrites par des tuples. Pour chaque tuple, le premier élément correspond à la distance en km et le deuxième élément correspond à la vitesse maximale en km/h.

Exemple :

```
Villes = {'Tunis':{'Sousse': (140, 110), 'Bizerte': (70, 100)},
'Sousse':{'Tunis': (140, 110), 'Sfax': (120, 100), 'Kairouan': (60, 80)},
'Sfax':{'Sousse': (120, 100), 'Gabes': (130, 90)},
'Kairouan': {'Sousse': (60, 80), 'Sfax': (150, 90)},
'Gabes':{'Sfax': (130, 90), 'Djerba': (110, 100)},
'Djerba':{'Gabes': (110, 100)},
'Bizerte': {'Tunis': (70, 100)}}
```

Travail demandé :

Étant donnée le dictionnaire `Villes`, écrire un programme en Python qui permet de :

1. Saisir un entier strictement positif `N`, correspondant au nombre de villes que le chauffeur souhaite visiter.
2. Saisir les `N` **noms** de villes à visiter et les ajouter dans une liste appelée `Trajet`.

Exemple: `Trajet=['Tunis', 'Sousse', 'Sfax', 'Gabes', 'Djerba']`

3. Vérifier si le trajet est valide ou non. Pour vérifier la validité du trajet, on doit vérifier que pour chaque paire de villes consécutives dans la liste `Trajet`, la première ville se trouve comme étant une clé dans le dictionnaire et la deuxième ville existe dans le dictionnaire qui contient les villes qui y sont reliées. Le résultat final de cette vérification doit être stocké dans une variable booléenne appelée `valide`.

Pour l'exemple du trajet `['Tunis', 'Sousse', 'Sfax', 'Gabes', 'Djerba']`, on aura dans la variable `valide` la valeur `True`.

4. Si le trajet n'est pas valide, on affiche le message "Le trajet indiqué est invalide". Sinon, calculer le temps total du parcours en procédant les étapes suivantes :

Pour chaque transition du trajet:

- Récupérer les valeurs de la distance et la vitesse,
- Calculer `Temps = distance / vitesse`
- Additionner pour obtenir le temps total.

Pour l'exemple du trajet ['Tunis', 'Sousse', 'Sfax', 'Gabes', 'Djerba'], on aura:

- Tunis $\rightarrow$ Sousse : $140 / 110 = 1,27$ h
- Sousse $\rightarrow$ Sfax : $120 / 100 = 1,20$ h
- Sfax $\rightarrow$ Gabes : $130 / 90 = 1,44$ h
- Gabes $\rightarrow$ Djerba : $110 / 100 = 1,10$ h

Le temps total du parcours est 5,02 heures.

Exercice4: (4 points)

1. Mettre en ordre les complexités suivantes, de la plus rapide à la plus lente et ce en utilisant la relation d'ordre "<" :

$O(n^3)$, $O(n^n)$, $O(\log(n))$, $O(n \log(n))$, $O(e^n)$, $O(\sqrt{n})$

2. Soit n un entier strictement positif, Pour chacun des deux programmes suivants, déterminer le coût (c'est-à-dire le nombre total d'opérations effectuées), puis en déduire sa classe de complexité en O (dans le pire des cas).

```
#Programme 1 :  
k=1  
while k <=n:  
    j = 1  
    while j<=2**(n):  
        j=j*2  
    k = k + 1
```

```
#Programme 2 :  
r = 0  
i = 1  
while i <= n:  
    j = 1  
    while j <=n:  
        r = 2 * (i + j) + r  
        j = j + 1  
    k = 1  
    while k < n**(3):  
        r = r + k  
        k = k *2  
    print(r)  
    i = i + 1
```