Examen — Python & IA

Sujet B — Durée : 1h30 — Aucun document autorisé†

Nom :	**Prénom :**	**Groupe :**
……	……	……

> † Une feuille **A4 recto** manuscrite autorisée si votre enseignant est sympa.
> **Consignes :** Lisez chaque question. Le barème est indiqué. Le code Python doit être correctement indenté. Les réponses vagues seront sanctionnées.

« J'ai codé une fonction qui résout tous mes problèmes… elle s'appelle main() et elle ne prend aucun argument. » — inconnu

Partie I — Questions de cours (6 points)

Réponses concises et précises.

Q1 (1 pt) — Quelle est la différence entre un `tuple` et une `list` en Python ? Dans quel cas utiliser un tuple plutôt qu'une liste ?

(Indice : l'un est une promesse, l'autre un contrat modifiable.)

Q2 (1.5 pt) — Qu'est-ce qu'une **table de hachage** (*hash table*) ? Expliquez le principe et dites comment Python l'utilise en interne.

Q3 (1.5 pt) — Expliquez le principe du **one-hot encoding**. Donnez un exemple avec les couleurs ["rouge", "vert", "bleu"].

Q4 (1 pt) — Que contient la variable `resultat` à la fin de ce code ?

```
texte = "abc abc def"
resultat = {}
for mot in texte.split():
    resultat[mot] = resultat.get(mot, 0) + 1
```

Donnez la valeur exacte de `resultat`.

Q5 (1 pt) — Vrai ou faux : « Une compréhension de liste peut contenir une condition `if` pour filtrer les éléments ». Donnez un exemple.

Partie II — Écrire du code (8 points)

Q6 (2 pts) — Écrire une fonction `freq_caracteres(texte)` qui compte le nombre d'occurrences de chaque **caractère** dans une chaîne (pas seulement les mots) et retourne un dictionnaire. Excluez les espaces.

```
def freq_caracteres(texte):
    # Votre code ici
```

Test : `freq_caracteres("hello")` → `{"h": 1, "e": 1, "l": 2, "o": 1}`

Q7 (2 pts) — Écrire une fonction `intersection(s1, s2)` qui retourne un **ensemble** contenant les éléments communs à deux listes (sans utiliser `&` ni `.intersection()`).

```
def intersection(s1, s2):
    # Votre code ici
```

Test : `intersection([1,2,3,4], [3,4,5,6])` → `{3, 4}`

Q8 (2 pts) — Écrire une fonction **récursive** `somme_chiffres(n)` qui calcule la somme des chiffres d'un entier positif n.

```
def somme_chiffres(n):
    # Votre code ici
```

Rappel : `n % 10` donne le dernier chiffre, `n // 10` enlève le dernier chiffre.
Test : `somme_chiffres(1234)` → `10` (car 1+2+3+4 = 10)

Q9 (2 pts) — Écrire une fonction `pairs_au_carre(liste)` qui retourne une nouvelle liste contenant le **carré** des **nombres pairs** de la liste d'origine. Utilisez une **compréhension de liste** avec filtre.

```
def pairs_au_carre(liste):
    # Votre code ici
```

Test : `pairs_au_carre([1, 2, 3, 4, 5, 6])` → `[4, 16, 36]`

Partie III — Analyse de code (4 points)

Q10 (2 pts) — Que va afficher ce code ? Écrivez la sortie **exacte**.

```
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
c = a & b
d = a | b
print("Intersection :", sorted(c))
print("Union :", sorted(d))
print("Dans a mais pas dans b :", sorted(a - b))
```

Réponse :

Q11 (2 pts) — Que fait la fonction suivante ? Expliquez en **une phrase**.

```
def mystere(matrice):
    n = len(matrice)
    m = len(matrice[0])
    res = [[0 for _ in range(n)] for _ in range(m)]
    for i in range(n):
        for j in range(m):
            res[j][i] = matrice[i][j]
    return res
```

(Indice : vu au cours 3. Si vous ne voyez pas, pensez à retourner… la grille.)

Réponse :

Partie IV — Petit problème (2 points)

Q12 (2 pts) — On a les notes de 3 matières stockées dans un dictionnaire de listes :

```
notes = {
    "maths":    [12, 15, 8, 17],
    "francais": [10, 14, 13, 11],
    "anglais":  [16, 9, 14, 12]
}
```

Chaque liste contient les notes de 4 étudiant·e·s (même ordre).
Écrivez le code qui calcule et affiche la **moyenne par matière**, puis la **matière avec la meilleure moyenne**.

(Indice : `sum()` / `len()` sur chaque liste. Et oui, les maths ont toujours la moyenne la plus intimidante.)

Bonus — Pour les courageux·ses (1 pt)

B1 (0.5 pt) — Le code suivant contient une erreur. Laquelle ? Proposez une correction.

```
d = {[1, 2]: "a"}
```

(Indice : les clés doivent être immutables. Les listes sont des caméléons émotionnels — elles changent tout le temps.)

B2 (0.5 pt) — En chiffrement XOR (cours 3), on a la propriété : `(m ⊕ k) ⊕ k = m`.
Complétez : « Le XOR est involutif : la même opération appliquée deux fois ……. ». (1 mot)

Partie	I	II	III	IV	Bonus	Total
Points	6	8	4	2	1	**20 + 1**

Chaque exercice est indépendant. La note maximale est 20/20 (le bonus s'ajoute au-delà de 20 si l'enseignant le souhaite).