Examen — Python & IA

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. La rigueur paie.

« Il n'y a pas de question stupide, mais certaines réponses le sont. Faites de votre mieux. » — Confucius (ou pas)

Partie I — Questions de cours (6 points)

Réponses concises et précises. Une phrase suffit dans la plupart des cas.

Q1 (1 pt) — Quelle est la différence fondamentale entre une `list` et un `set` en Python ? Donnez un cas d'usage typique pour chacun.

(Indice : un set n'a pas de double. Comme les bonnes idées pendant un examen.)

Q2 (1.5 pt) — Qu'est-ce que la **mémoïsation** ? Expliquez avec un exemple de fonction où elle est utile.

Q3 (1.5 pt) — Expliquez le principe du **Bag of Words** (BoW). Quelle information structurelle du texte perd-on avec ce modèle ?

(Exemple : « Le chat mange la souris » et « La souris mange le chat » donnent le même sac de mots. Problème ?)

Q4 (1 pt) — On exécute le code suivant :

```
 d = {}
d["pomme"] = 3
d["poire"] = 2
d["pomme"] = 5
```

Combien d'entrées contient `d` à la fin ? Quelle est la valeur de `d["pomme"]` ? Justifiez.

Q5 (1 pt) — Vrai ou faux : « La similarité de Jaccard entre deux ensembles identiques vaut toujours 1 ». Justifiez par la formule.

Partie II — Écrire du code (8 points)

Écrivez des fonctions Python complètes. L'indentation compte.

Q6 (2 pts) — Écrire une fonction `comptage_mots(texte)` qui prend une chaîne de caractères et retourne un dictionnaire où chaque clé est un mot et chaque valeur son nombre d'occurrences. Utilisez `.split()`.

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

Test avec `"le chat le chien le lapin"` → `{"le": 3, "chat": 1, "chien": 1, "lapin": 1}`

Q7 (2 pts) — Écrire une fonction `jaccard(s1, s2)` qui calcule la similarité de Jaccard entre deux ensembles.

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

Rappel : J(A, B) = |A ∩ B| / |A ∪ B| Test : `jaccard({1,2,3}, {2,3,4})` → ?

Q8 (2 pts) — Écrire une fonction **récursive** `factorielle(n)` qui calcule n! .

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

Rappel : 0! = 1, n! = n × (n-1)! Test : `factorielle(5)` → 120

(Indice : le cas de base est votre meilleur ami. Sans lui, c'est la récursion infinie, et Python déteste ça.)

Q9 (2 pts) — Écrire une fonction `normaliser(vecteur)` qui prend une liste de nombres et retourne une nouvelle liste où chaque élément est divisé par la norme euclidienne du vecteur. Utilisez une **compréhension de liste**.

```
def normaliser(vecteur):
    # Votre code ici
```

Norme euclidienne : ||v|| = √(Σ v_i^2) Test : `normaliser([3, 4])` → `[0.6, 0.8]`

Partie III — Analyse de code (4 points)

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

```
notes = {"Alice": 15, "Bob": 12, "Charlie": 18}
moy = sum(notes.values()) / len(notes)
print("Moyenne :", moy)
for nom, note in notes.items():
    if note >= 15:
        print(nom, ":", note, "★")
    else:
        print(nom, ":", note)
```

Réponse :

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

```
def mystere(texte, k):
    res = []
    for c in texte:
        if 'a' <= c <= 'z':
            res.append(chr((ord(c) - 97 + k) % 26 + 97))
        elif 'A' <= c <= 'Z':
            res.append(chr((ord(c) - 65 + k) % 26 + 65))
        else:
            res.append(c)
    return ''.join(res)
```

(Vu dans le cours 3. Si vous ne reconnaissez pas, relisez la partie cryptographie.)

Réponse :

Partie IV — Petit problème (2 points)

Q12 (2 pts) — On a deux étudiant·e·s avec leurs notes dans des dictionnaires :

```
 notes_alice = {"maths": 15, "francais": 12, "anglais": 18}
notes_bob   = {"maths": 8,  "francais": 14, "anglais": 10}
```

Écrivez le code qui calcule et affiche la moyenne de chaque étudiant·e, puis qui affiche le nom de celui ou celle qui a la meilleure moyenne.

(Réutilisez des concepts du cours 2 — et non, la réponse n'est pas « Alice » parce qu'on la préfère. Il faut le prouver par le calcul.)

Bonus — Pour les courageux·ses (1 pt)

B1 (0.5 pt) — Le code suivant contient une erreur. Laquelle ? Que faudrait-il faire pour la corriger ?

```
d = {}
d[["nom", "prenom"]] = "Alice"
```

(Indice : certaines choses ne peuvent pas être des clés de dictionnaire. Comme les listes. Et les promesses politiques.)

B2 (0.5 pt) — En cryptographie XOR (cours 3), une propriété célèbre est : `(a ⊕ k) ⊕ k = a` pour toute clé k.
Complétez la phrase : « Le XOR est son propre …….. ».

(Réponse en un mot. Si vous avez trouvé, vous venez de comprendre pourquoi on l'utilise en cryptographie.)

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).