4 Informatique

4.1 Informatique pour tous

4.1.1 Généralités et présentation du sujet

Le sujet d'informatique commune traitait cette année de la modélisation et la simulation de l'aimantation de matériaux magnétiques. Il comportait quatre parties, chacune d'entre elles abordant un aspect différent de l'étude de cette aimantation : étude de l'aimantation moyenne (partie I), du traitement d'une base de données des propriétés des matériaux magnétiques (partie II), étude du modèle d'Ising décrivant la transition paramagnétique-ferromagnétique (partie III), puis enfin (partie IV) l'étude de l'apparition de domaines magnétiques (*domaines de Weiss*) au sein du matériau, dans lesquels les moments magnétiques (ou *spins*) sont orientés dans la même direction.

Les 24 questions de l'épreuve balayaient une partie conséquente du programme d'informatique commune : gestion de listes (et de listes de listes), résolution numérique d'équations, complexité, récursivité, pile, bases de données. Certaines questions étaient d'un niveau élémentaire (importation de modules, requête SQL simple, calcul d'une moyenne...), quand d'autres exigeaient une maîtrise et une compréhension plus fines. L'épreuve abordait ainsi un large éventail de notions étudiées durant les deux années de préparation, et a permis d'évaluer et de classer l'ensemble des candidats.

4.1.2 Commentaires généraux

Les remarques effectuées dans le rapport 2021 s'appliquent pour une large part encore cette année.

- Si certaines copies sont très faibles (voire presque vides), certaines sont excellentes et frisent parfois la perfection. La longueur et la difficulté du sujet étaient ainsi tout à fait adaptées à ce type d'épreuve, ce qui a permis de classer les candidats.
- Cette année encore, le jury souhaite souligner l'importance de la présentation des copies. Certaines sont très brouillonnes, sales, voire parfois illisibles. Un nombre trop important de ratures nuit forcément à la lecture des codes Python produits, et peut même provoquer des erreurs de syntaxe. On peut certes tolérer quelques ratures propres (correction d'un oubli, d'une erreur de syntaxe), qui ne nuisent pas à la poursuite de la lecture ni à la structure des codes proposés, mais un code trop difficile à déchiffrer (excès de ratures ou de rajouts par le biais de flèches ou d'astérisques) est forcément sanctionné.
- Ainsi, la présentation des codes Python est primordiale : les candidats doivent prêter attention au choix des noms de variables, à l'insertion de commentaires pertinents dans le corps de leurs programmes (ou en amont si ces commentaires sont trop longs).
- De la même façon, une erreur ponctuelle de syntaxe (oubli des `:`, d'une parenthèse fermante) peut être tolérée. En revanche, l'absence récurrente des parenthèses (en écrivant par exemple systématiquement `for i in range n` ou `len L` ou lors de l'utilisation d'une fonction déjà codée)

a été sanctionnée. Le jury note également cette année une recrudescence d'erreurs de syntaxe problématiques : `10^(-6)`, confusion entre `=` et `==` dans un test, utilisation abusive de `range` (dont les bornes et le pas doivent être entiers). Notons à ce propos la confusion fréquente entre les opérateurs `/` et `//` : par exemple, `range(a/b)` provoque une erreur, puisque `a/b` est un flottant (il convient alors d'utiliser la syntaxe `range(a//b)`). *A contrario*, `(a+b)//2` ne renvoie pas le flottant $\frac{a+b}{2}$: la syntaxe adéquate est dans ce cas `(a+b)/2`.

- L'importation des modules en Python doit être maîtrisée. Il existe plusieurs façons de procéder, qui ne doivent pas être confondues.
- Le sujet demandait explicitement de manipuler des listes en évitant expressément le module `numpy`. Les candidats doivent maîtriser la manipulation des listes, notamment :
- la construction d'une liste élément par élément. Par exemple, l'initialisation d'une liste `L = []` suivie, dans une boucle `for`, d'une affectation `L[i] = elt` provoque une erreur. De même, l'instruction `L=h*[]` initialise la liste `L` à une liste vide (et pas à une liste de h listes vides).
- l'ajout d'un élément à la fin d'une liste. Comme indiqué dans les rapports des années précédentes, la syntaxe `L.append(elt)` est à privilégier. D'une part, elle est plus efficace, mais elle est également moins source d'erreurs. L'emploi de la syntaxe `L = L + [elt]` (ou `L += [elt]`) a par exemple provoqué beaucoup d'oubli de crochets, quand `elt` était lui-même une liste.
- le parcours d'une liste a donné lieu à beaucoup d'erreurs, les candidats confondant les éléments d'une liste avec leur indice.
- la concaténation de listes (ou de listes de listes) a également donné lieu à beaucoup d'erreurs.
- la syntaxe du « slicing » des listes (pour ceux qui ont voulu l'utiliser) n'est pas toujours maîtrisée, notamment en ce qui concerne l'indice final ou l'utilisation des « : » (confondus avec des virgules).
- le caractère modifiable des listes en Python n'est pas compris par tous. Quand l'énoncé demandait par exemple de modifier une liste passée en paramètres, aucun `return` n'était attendu.

4.1.3 Commentaires spécifiques à chaque question

Q1 - Question plutôt réussie. L'utilisation de la fonction `tanh` devait être cohérente avec la question précédente. L'utilisation d'une fonction anonyme (à l'aide du mot clé `lambda`) n'était pas nécessaire, et a donné lieu à beaucoup d'erreurs de syntaxe.

Q2 - L'algorithme de dichotomie n'est pas suffisamment maîtrisé. Beaucoup de candidats confondent notamment `(b-a)/2` et `(a+b)/2` pour le calcul du milieu du segment $[a, b]$. Rappelons également qu'un test d'égalité entre deux flottants (visant à tester si l'on est tombé exactement sur une solution au cours de la dichotomie) n'est pas pertinent. La condition d'arrêt précisée dans l'énoncé n'a pas toujours été respectée.

Q3 - La complexité de l'algorithme de dichotomie n'est pas toujours connue. Parmi les candidats qui la détermine, la résolution de l'inéquation $\frac{b-a}{2^n} \leqslant \varepsilon$ a donné lieu à des erreurs regrettables de manipulation du logarithme.

Q4 - Beaucoup d'erreurs ont été commises concernant le calcul du pas utilisé pour la construction d'une subdivision régulière de 500 points de l'intervalle $[t_1, t_2]$. Outre les erreurs de syntaxe évoquées dans les commentaires généraux (concernant l'utilisation de `range` et l'erreur de syntaxe `10^(-6)`), beaucoup de copies ont oublié que si $t \geqslant 1$, alors $m = 0$.

Q5 - Question plutôt réussie dans l'ensemble.

Q6 - Attention à l'oubli du coefficient `4.5`. Certains candidats ne savent pas comment obtenir deux attributs dans la même requête SQL. La syntaxe d'une jointure (et le choix des attributs adéquats) n'est pas assez maîtrisée.

Q7 - L'utilisation de `MIN` a souvent donné lieu à des erreurs de syntaxe.
Q8 - L'utilisation de `GROUP BY` et de `HAVING` (souvent confondu avec `WHERE`) est souvent erronée.

Q9 - Question facile. Les candidats doivent maîtriser l'initialisation d'une liste.

Q10 - L'alternance des `1` et `-1` a donné lieu a de nombreuses erreurs.

Q11 - La conversion d'une liste en liste de listes est souvent problématique (initialisation d'une liste de listes, gestion et calcul des indices doubles)

Q12 - Question délicate à traiter. Pour ce genre de question, il est primordial d'expliquer sa démarche avant d'écrire des formules souvent indigestes. La recherche d'une solution en une ligne n'est pas à privilégier : le code obtenu est souvent illisible, voire faux, et même quand il est correct, il n'est pas forcément plus efficace (au contraire) qu'une solution plus naturelle utilisant une disjonction de cas.

Q13 - Beaucoup de confusions entre élément et indice d'un élément dans une liste.

Q14 - Beaucoup de candidats n'ont pas su gérer le côté probabiliste du changement de signe d'un spin.

Q15 - La détermination de la complexité d'une fonction ne se résume pas à compter le nombre de boucles `for` !

Q16 - Les candidats ayant bien traité les questions précédentes ont souvent réussi cette question. L'énoncé demandait uniquement de modifier la liste placée en paramètre.

Q17 - Question plutôt réussie.

Q18 - Beaucoup de candidats ont identifié deux boucles (de `n` et `n_tests` itérations respectivement), et en ont conclu à tort que la complexité était en $\mathcal{O}$(`n*n_tests`).

Q19 - Question plutôt réussie par les copies ayant réussi la question précédente.

Q20 - Cette question a parfois donné lieu à des interprétations farfelues, parlant par exemple de pixels.

Q21 - Q22 - Q23 - Figurant en fin d'énoncé, ces questions ont permis de valoriser la prise de recul sur le sujet ainsi que la maîtrise de la récursivité et de l'utilisation des piles par les meilleurs candidats.