4 Informatique

4.1 Informatique commune aux filières MP, PC et PSI

4.1.1 Généralités et présentation du sujet

Le sujet d'informatique commune traitait cette année de la typographie informatisée. Il abordait la manipulation et le tracé des polices vectorielles, l'affichage de texte et la justification d'un paragraphe. Les 26 questions de l'épreuve balayaient une partie conséquente du nouveau programme d'informatique commune : gestion de listes (et de listes imbriquées), images et matrices de pixels, récursivité, algorithmes gloutons, programmation dynamique, mémoïsation, bases de données... Certaines questions étaient d'un niveau élémentaire (requête SQL simple, manipulations classiques de listes...), quand d'autres (essentiellement en fin de sujet) 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.

Une analyse détaillée des questions est présentée dans l'annexe R.

4.1.2 Commentaires généraux

Les remarques effectuées dans le rapport 2022 s'appliquent pour une large part encore cette année, et nous les reprenons en partie ci-après. Nous les complétons également de remarques spécifiques à l'épreuve de 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 bien 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 tracées à la règle (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.

- 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). Il est cependant rarement utile d'écrire un paragraphe de plusieurs lignes pour présenter l'idée générale d'un code Python. Une ou deux phrases suffisent en général largement.

- 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 sujet demandait explicitement de manipuler des listes imbriquées. 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). Il en va de même pour des listes formées de listes vides : les syntaxes `[[]] * n` ou `[[]*n]` ne conviennent pas.
  - Le parcours d'une liste dans une boucle `for` peut se faire éléments par éléments (`for elt in L:`) ou indice par indice (`for i in range(L):`) ; la première donnait souvent lieu à des codes plus lisibles. Attention toutefois à ne pas confondre les deux syntaxes.
  - 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'oublis de crochets, quand `elt` était elle-même une liste.
  - 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. Sauf mention du contraire, les listes rentrées en paramètres des fonctions ne doivent pas être modifiées.

4.1.3 Conseils aux futurs candidats

Nous conseillons aux futurs candidats une lecture attentive du rapport du jury ainsi que du programme officiel d'informatique commune : celui-ci peut aider à vérifier la maîtrise des points exigibles aux concours. De plus, un nombre souvent faible de questions a été traité dans de nombreuses copies, montrant un manque d'entraînement à écrire des codes, mêmes simples. Un investissement un peu plus important des candidats en informatique commune produirait certainement une nette amélioration.

4.2 Informatique option MP

4.2.1 Généralités

Le sujet s'intéresse à la résolution du jeu de hanjie, un jeu de réflexion consistant en la création d'une image sur une grille de pixels noirs ou blancs en utilisant des contraintes indiquées sur chaque ligne et colonne. Ces contraintes précisent les séries de cases noires consécutives dans la ligne ou la colonne et permettent de trouver de façon unique comment colorier la grille.
Le sujet est composé de trois parties indépendantes :

Q Informatique commune MP, PC et PSI

Q1 - Beaucoup d'erreurs d'interprétation de la représentation en base 16. De plus, ce genre de calculs simples doit pouvoir être effectué à la main sans erreur. De nombreux problèmes de conversion (cent / dollar) également.
Q2 - Question souvent réussie, à condition de bien avoir compris l'imbrication des listes.
Q3 - La syntaxe exacte de `COUNT` n'est pas toujours maîtrisée : il faut préciser l'attribut, ou utiliser à bon escient la syntaxe `COUNT(*)`.
Q4 - La syntaxe des jointures n'est pas toujours maîtrisée : le programme officiel de CPGE est très clair à ce sujet. De nombreuses erreurs de syntaxe ont été remarquées, relatives à l'interversion table / attribut : `TABLE.ATTRIBUT` est la syntaxe correcte, et non pas `ATTRIBUT.TABLE`.
Q5 - La gestion des fonctions d'agrégation a posé problème : il fallait utiliser un `GROUP BY`, et la syntaxe du tri par ordre alphabétique à l'aide d'un `ORDER BY` n'est souvent pas maîtrisée.
Q6 - Question plutôt réussie. Le parcours d'une liste par éléments plutôt que par indice était judicieux dans ce contexte.
Q7 - Question plutôt réussie. On pouvait utiliser avec profit les listes définies par compréhension.
Q8 - L'utilisation des fonctions précédentes n'a pas toujours été réussie.
Q9 - Le parcours des lettres de l'alphabet a souvent posé problème. Le jury n'attendait pas la maîtrise des fonctions `ord` ou `chr`. Quand `police` est un paramètre de la fonction, l'utiliser avec la syntaxe `'police'` au lieu de `police` est faux.
Q10 - Le jury a remarqué beaucoup d'erreurs concernant le nombre de listes imbriquées. Là encore, l'utilisation de listes définies par compréhension s'avérait plutôt judicieuse. Le jury attendait que la liste `v` ne soit pas modifiée.
Q11 - La fonction `zzz` appliquait une homothétie selon l'axe des abscisses. Le jury s'étonne de la confusion avec la notion de translation.
Q12 - Question plutôt réussie. Attention toutefois : une fonction d'entête `f([x,y])` provoque une erreur.
Q13 - Il suffisait de dérouler l'algorithme. Trop d'étourderies et d'erreurs de calcul ont été remarquées.
Q14 - Question moins bien réussie que prévue : un `range(a,b)` avec `b<a` ne provoque pas d'erreur. De plus, la notion d'assertion (officiellement au programme de CPGE) n'est connue que par une minorité de candidats.
Q15 - Peu de copies ont bien analysé le problème : l'objectif de cette partie (coloration continue d'un segment) a souvent été oublié.
Q16 - Question peu réussie : l'échange du rôle de `x` et `y` n'a pas souvent été mis en œuvre.
Q17 - Question peu réussie : il fallait avoir compris les problèmes précédents et effectuer une disjonction de cas opportune.
Q18 - Question très peu réussie. Comme à la question 12, la géométrie élémentaire (homothétie, translation) a souvent posé problème.
Q19 - Question difficile pour beaucoup de copies. L'utilisation à bon escient des fonctions précédentes a été délicate. Le calcul de la largeur a souvent été faux, voire complètement oublié.
Q20 - Question difficile. La gestion d'évolution de la largeur a posé beaucoup de problèmes syntaxiques.
Q21 - La définition d'un algorithme glouton n'est pas connue par beaucoup : il ne s'agit pas d'une question de complexité.

Q22 - Question assez réussie par les copies l'ayant abordée.
Q23 - Question peu abordée : les meilleures copies ont bien compris la notion de mémoïsation.
Q24 - Question très peu réussie, bien qu'abordée par un nombre non négligeable de copies : ce n'est pas parce qu'il y a deux boucles imbriquées que la complexité est quadratique.
Q25 - Question très peu abordée.
Q26 - *Idem.*

↑RETOUR