

## ÉPREUVE ORALE D'INFORMATIQUE

### 1. Remarques générales

L'épreuve orale d'informatique, au choix avec la chimie, dure 50 minutes : 25 minutes de préparation suivies de 25 minutes d'exposé devant l'examineur.

Pendant la 1<sup>ère</sup> partie de l'exposé (10 minutes) le candidat est amené à présenter la résolution d'un exercice tiré au sort et préparé pendant les 25 minutes préalables. Pendant la seconde moitié de l'exposé, il peut au choix présenter un projet préparé tout au long de son année en classe préparatoire (10 minutes plus un temps pour des questions), ou travailler sur un exercice non préparé proposé par l'examineur.

- ✓ L'objectif des exercices proposés est de vérifier la capacité du candidat à pouvoir transformer un problème élémentaire en un algorithme, à déterminer les étapes permettant de mettre en œuvre cet algorithme et à identifier les fonctions et types de structures nécessaires à sa programmation. Le programme qui en résulte est écrit dans le langage de programmation Python.

Les exercices se présentent sous forme de problèmes généraux ne faisant pas appel nécessairement à des notions mathématiques, physiques ou biologiques...

Pendant l'exposé, le candidat est convié à présenter une solution pour résoudre le problème posé et à répondre à des questions qui peuvent être liées à la solution exposée, prolongements, variantes, efficacité de l'algorithme proposé... Les interventions de l'examineur sont destinées à obtenir des précisions, corriger des erreurs ou de mauvaises démarches, elles ne sont jamais faites pour perturber le candidat.

L'évaluation tient compte d'aspects strictement "informatiques" :

- exactitude de l'algorithme présenté
- maîtrise des concepts de programmation manipulés
- efficacité du programme, prise en compte des cas particuliers

Plus généralement d'autres qualités ont aussi été appréciées :

- vivacité et rapidité suite aux remarques de l'examineur
- aptitude à défendre les solutions proposées
- capacité à relier le problème à des problèmes plus généraux
- maîtrise du langage et "élégance" des solutions présentées

- ✓ Dans la 2<sup>ème</sup> partie de l'exposé, le candidat présente un projet réalisé pendant l'année scolaire. Les candidats peuvent s'appuyer sur une présentation projetée sur leur ordinateur ou imprimée. Certains n'ont utilisé aucun support hormis le script du programme implémenté, mais cette solution doit être évitée car elle rend l'appréhension du projet un peu plus difficile pour l'examineur.

L'exposé sur le projet a notamment pour objectif de mettre en évidence la capacité du candidat à présenter clairement :

- le sujet sur lequel il a travaillé
- les hypothèses et limites fixées pour sa résolution
- l'analyse effectuée et les solutions algorithmiques mises en œuvre pour le réaliser
- les difficultés rencontrées et les perspectives pouvant être envisagées
- éventuellement également des éléments de gestion de projet : répartition des tâches, problèmes organisationnels...

L'évaluation tient compte de :

- la qualité et la clarté de la présentation
- de l'ampleur du projet : difficulté du sujet, recherche bibliographique, nombre de méthodes implémentées, analyse des résultats ...
- d'une estimation de l'investissement apporté sur le projet (nombre de participants au projet, durée sur l'année, nombre de lignes de code ...)

- de la qualité du code : organisation en fonctions, organisation des instructions conditionnelles, des itérations, utilisation d'"outils" python tels que le "slicing", les listes en compréhension, etc., concision du code...
- de la qualité de la présentation du script, de la présence de commentaires pertinents

La palette des projets présentés a été très variée et de façon générale, les sujets étaient intéressants. Comme d'habitude, nous avons cependant constaté à nouveau une grande différence au niveau du temps consacré au projet au cours de l'année et des conditions de réalisation (nombre d'élèves impliqués, recherche biblio nécessaire, nombre de méthodes implémentées, interface graphique fournie ou non, etc.) et cela se traduit par de grosses différences dans le volume et la complexité du code présenté. Cependant, contrairement à l'année passée, nous n'avons pas ou très peu vu de projets basés pratiquement exclusivement sur la programmation d'une interface graphique qui ne permettent pas d'évaluer correctement les compétences en algorithmique, et c'est tant mieux.

Il apparaît que l'ensemble des candidats a choisi l'option informatique en connaissance de cause, et à part quelques exceptions, ils ont les compétences permettant de résoudre les exercices. Certains candidats montrent une très bonne maîtrise des concepts manipulés et une grande aisance à écrire un algorithme. Les examinateurs tiennent à souligner que même si certains candidats ont parfois été décontenancés par le sujet et n'ont pas trouvé forcément la bonne solution au départ, les interrogateurs ont tout de même pu évaluer leur capacité à rebondir aux remarques, leur réactivité pour rectifier le tir et proposer une solution au problème posé et leurs compétences en programmation.

## 2. Quelques points d'amélioration attendus

### 2.1 Exposé

- Il est indispensable que le candidat présente le sujet de l'exercice dans son ensemble avant de rentrer dans le détail sans aucune introduction. Certains candidats rentrent toute de suite dans le vif du sujet sans effectuer cette introduction et c'est préjudiciable à la clarté de l'exposé.
- De la même façon, chaque question doit être introduite en présentant les résultats attendus, les données fournies et brièvement la méthode mise en œuvre.
- Il faut que les candidats prennent le temps de bien lire l'énoncé et de se poser les bonnes questions avant de se lancer dans sa résolution. Pour ceux qui l'ont fait spontanément, cela traduit une certaine prise de recul et une capacité de synthèse appréciable.
- Concernant l'utilisation de noms de variables "explicites", l'amélioration constatée l'année passée se poursuit, aussi bien dans les exercices présentés que dans les projets et c'est très appréciable. Il reste encore quelques élèves récalcitrants qui continuent à utiliser des noms de variables tels que M, N, ou x, y, z, a, b, m, n, ou bien I ou L si c'est une liste même si elle représente un polynôme ou un plateau de jeu...

L'utilisation de noms plus explicites aide le candidat dans la résolution du problème et dans la présentation de la solution proposée et favorise une compréhension aisée et rapide des codes présentés à l'examineur.

### 2.2 Programmation

- Au niveau programmation, quelques améliorations peuvent être apportées :
  - Attention au vocabulaire utilisé, une instruction conditionnelle n'est pas une "boucle"...
  - programme = script et non fonction. Quand il est demandé un programme dans l'énoncé, c'est souvent afin de pouvoir tester des appels de fonctions.
  - Privilégier l'utilisation de la méthode **append** plutôt que celle de l'opérateur + quand on veut ajouter un élément dans une liste. C'est beaucoup plus efficace.
  - On peut également faciliter l'écriture de certains programmes en utilisant l'instruction **break**, qui permet souvent d'atteindre plus facilement les recommandations de *The Zen of Python*.
  - Les candidats semblent toujours peu à l'aise avec les chaînes de caractères et ont parfois été un peu perturbés par les exercices les mettant en œuvre. Ce problème est certainement dû au calendrier qui fait que les chaînes de caractères sont présentées et

utilisées en début d'année et ont été un peu mises de côté après. Pour pallier cette difficulté, nous avons ajouté dans les énoncés qui le nécessitent un petit rappel sur les manipulations de base des chaînes de caractères. Certains persistent cependant à utiliser les chaînes de caractères sous forme de listes et cela complique la programmation.

- Certains candidats ne connaissent pas l'opérateur modulo "%" qui rend pourtant de nombreux services, tester si un nombre est pair par exemple...
- Peu savent utiliser également le *slicing* (découpage) de Python permettant d'extraire des sous-chaînes ou des sous-listes très facilement et rapidement. Ce pourrait être intéressant qu'il soit un peu plus travaillé pendant l'année.
- Il y a également une sous-utilisation des instructions très intéressantes dans le langage python :

```
if v in sequence:  
    instructions
```

```
for car in sequence:  
    instructions
```

où *sequence* peut être une liste ou une chaîne de caractères.

- La notion de référence ne semble pas connue : les fonctions qui manipulent une liste passée en paramètre n'ont pas besoin de retourner la liste en résultat. Mais cette notion non triviale, pourra être approfondie ultérieurement.

### 2.3 Présentation du projet

- Contrairement aux années passées, on observe désormais la plupart du temps la présence d'un "programme principal" avec l'enchaînement des fonctions à lancer pour pouvoir faire tourner le programme. Sans ce programme principal en effet, il est difficile de connaître la succession des instructions permettant de le lancer et de le tester.
- Merci de choisir une impression adaptée qui facilite la lecture du script :
  - taille de police permettant que toutes les instructions tiennent sur une seule ligne, commentaires compris. (Obtenir 60 lignes par page donne une mesure indicative de la taille de la police pouvant être utilisée).
  - Pour éviter que les lignes soient trop longues, on peut éviter de mettre les commentaires en fin de ligne et les mettre sur la ligne précédente.
  - indentation (ne pas faire un copier-coller dans un logiciel de traitement de textes qui perd toutes les indentations).
- Imprimer les numéros de lignes.
- Les diapositives sont parfois trop "rédigées", contiennent beaucoup trop de texte, peu visible et trop petit. Ne garder que des mots clés, les idées principales. Préférer une animation ou un dessin pour illustrer une méthode ou un algorithme plutôt qu'une capture d'écran avec du code.
- Présenter les hypothèses de travail.
- On peut apprécier un graphe des appels des fonctions principales.