- 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

2. Remarques générales

Il apparaît que l'ensemble des candidats ont choisi l'option informatique en connaissance de cause, et à part quelques exceptions, ils ont les compétences permettant de résoudre les exercices, ce qui donne une moyenne de 12,46 à l'épreuve. 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. Cette année, nous avons noté une amélioration du niveau des étudiants, plusieurs d'entre eux ont pu terminer la totalité de l'exercice donné en préparation ce qui était plutôt exceptionnel les années précédentes.

La palette des projets présentés est variée, même s'il s'agit souvent de l'implantation de jeux. Nous avons eu le plaisir d'avoir la présentation de très bons projets. Tous les projets ou presque, plus ou moins ambitieux, étaient aboutis et nous souhaitons souligner la qualité de l'encadrement dans la plupart des cas. Dans l'ensemble, nous avons pu observer également une bonne structuration en plusieurs petites fonctions facilitant la compréhension du programme, et leur mise au point.

Comme d'habitude, nous avons cependant constaté une énorme disparité au niveau de l'ampleur des sujets traités, du temps consacré au projet au cours de l'année et des conditions de réalisation (nombre d'élèves impliqués, recherche biblio, nombre de méthodes implémentées, interface graphique fournie ou non, implication des encadrants, etc.) et cela se traduit par de grosses différences dans le volume et la complexité du code présenté. Nous considérons par exemple les projets de 120 lignes réalisés à 3 sur une période allant de septembre à avril, présentés souvent sans commentaires, en dessous du niveau demandé. De la même façon, nous avons sanctionné des projets type démineurs sans interface graphique, sans interaction avec l'utilisateur, où l'ordi joue aléatoirement…

Nous avons également été confrontés plusieurs fois à des représentations de données non adaptées, (utilisation de matrices quand une liste simple suffit par exemple), à des programmes mal structurés (abus de tests ou autres…), ainsi qu'à des algorithmes d'une maladresse regrettable. On nous a également présenté des algorithmes "truqués" afin de pouvoir présenter des résultats corrects. Il est nécessaire que les enseignants ne laissent pas passer de telles choses.

Dans de nombreux projets, les élèves parlent également d'IA, ce terme est souvent galvaudé. Il correspond souvent à des stratégies de jeu et même si le développement algorithmique était dans certains cas pointu, dans d'autres cas, on ne peut pas considérer que si le programme effectue un choix aléatoire, on puisse appeler cela une IA !

Les supports de présentations étaient dans l'ensemble bien préparées avec des illustrations. Pour les meilleures, on note une prise de recul vraiment intéressante. On peut déplorer cependant que certaines présentations ne contiennent que du code alors qu'on attend une explication de l'algorithme et que d'autres comportent des fautes d'orthographe…

3. Quelques points d'amélioration attendus

3.1 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 ces deux dernières années se poursuit, aussi bien dans les exercices présentés que dans les projets et c'est très appréciable

3.2 Programmation

- Au niveau programmation, quelques améliorations peuvent être apportées :
    - Attention au vocabulaire utilisé, une instruction conditionnelle n'est pas une "boucle"…
    - Si on rentre dans les détails, on observe toujours que nombre de candidats privilégient l'opérateur + pour ajouter un élément dans une liste plutôt que l'utilisation de la méthode ***append***. Ce qui peut nuire à l'efficacité quand on traire de nombreuses données.
    - On a observé l'utilisation plus régulière et maîtrisée de l'instruction *break* qui facilite l'écriture de certains programmes qui et permet souvent d'atteindre plus facilement les recommandations de *The Zen of Python*.
    - La manipulation des chaines de caractères est encore un point de difficulté pour certains mais il y a globalement un net progrès sur cet aspect.
    - Quelques candidats ne connaissent pas l'opérateur modulo "%" qui rend pourtant de nombreux services, tester si un nombre est pair par exemple…
    - Le *slicing* (découpage) de Python permettant d'extraire des sous-chaînes ou des sous-listes très facilement et rapidement semble plus connu que les années passées.
    - 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.

3.3 Présentation du projet

- On observe désormais la plupart du temps la présence d'un "programme principal" avec l'enchaînement de l'ensemble 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.

  Malheureusement il subsiste encore des projets dans lesquels il faut deviner quelle est la fonction à appeler pour lancer le programme. Même si ça a été parfois noté dans les commentaires de début de script, il est préférable de prévoir le programme principal qui permet de lancer directement la fonction en question.
- Au niveau de la présentation des fonctions, les candidats ont souvent un peu de mal à équilibrer le nombre de sauts de lignes permettant d'avoir un programme facile à lire. Entre aucune ligne blanche ou une ligne blanche par instruction, c'est tout ou rien ... Les recommandations pour la présentation des programmes python sont les suivantes :

  Les définitions de fonctions sont séparées par une ligne vide. Des sauts de ligne peuvent également être utilisés, pour délimiter des portions de code correspondant à une étape donnée du traitement appliqué.
- Merci de choisir une impression adaptée qui facilite la lecture du script, nous avons "subi" des impressions de scripts absolument impossible à lire :
    - longueur maximum recommandée pour une ligne : 79 caractères
    - taille de police permettant que toutes les instructions tiennent sur une seule ligne, commentaires compris. (Obtenir 60 lignes de code par page donne une mesure indicative de la taille de la police pouvant être utilisée).
    - pour éviter que les lignes soient trop longues, 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 perdent toutes les indentations).

- Imprimer les numéros de lignes. Si l'éditeur de code utilisé ne permet pas l'impression des numéros de ligne, le candidat pourra noter à la main dans la marge les numéros des lignes multiples de 10 par exemple.
- Certaines diapositives restent encore parfois trop "rédigées", avec 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.
- Il est important de présenter les hypothèses de travail fixées pour la réalisation du projet.