Épreuve de mathématiques pratiques et informatique
Rapport du jury – Session 2016

6 octobre 2016

1 Modalités de l'épreuve

Un sujet d'oral est composé d'un unique exercice imposé au candidat, portant sur le programme de mathématiques et d'informatique de BCPST. Dans la mesure du possible, les sujets ont été choisis pour couvrir l'ensemble du programme des deux années (BCPST1 et BCPST2).

Les candidats disposent de 30 minutes de préparation dans une salle dédiée, où ils ont accès à un ordinateur sur lequel sont installés les logiciels Python 3 (Pyzo), Excel et Geogebra.

Les candidats disposent d'une clé USB fournie par le concours au début de leur préparation, sur lequel ils peuvent enregistrer leur travail.

A l'issue de la préparation, ils sont accompagnés dans une salle d'interrogation (5 minutes au maximum sont prévues pour la transition), où ils disposent d'environ 18-20 minutes pour exposer le résultat de leur travail de préparation et dialoguer avec l'examinateur. Dans cette salle de passage, ils disposent également d'un ordinateur connecté à un vidéoprojecteur, sur lequel ils peuvent exécuter les programmes sauvegardés sur la clé USB fournie

A l'issue de ce premier temps d'examination, les candidats sont accompagnés vers une deuxième salle, dans la mesure du possible voisine de la précédente, où ils présentent le résultat de leur projet informatique durant une durée de 18-20 minutes devant un second examinateur. Dans cette deuxième salle, les candidats disposent d'un ordinateur avec vidéoprojecteur sur lequel est affiché le dossier qu'ils ont au préalable envoyé au concours au début du mois de juin.

2 Éléments statistiques

Un résumé statistique est fourni ci-dessous, pour chacune des parties de l'épreuve ainsi que pour la note finale (chaque partie était notée sur 10). On notera que la corrélation entre la note d'un candidat sur la partie Mathématiques pratiques et sa note sur la partie Projet informatique progresse très légèrement ($r = 0{,}42$, $r^2 = 0{,}18$, alors que r^2 valait 0,14 l'année dernière), mais reste assez faible : en particulier, d'assez nombreux candidats ayant de grosses difficultés en mathématiques ont pu montré des compétences très intéressantes en informatique.

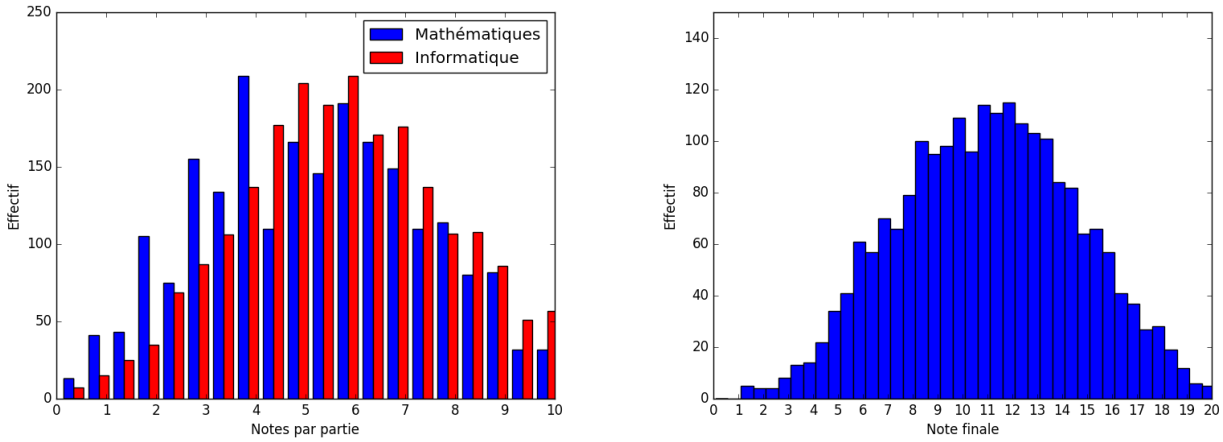
La distribution des notes est semblable à celle de l'année dernière, mais les candidats ayant obtenu des notes comprises entre 2 et 8 sur la partie informatique sont d'un meilleur niveau que ceux ayant obtenu des notes équivalentes l'année dernière.

Partie	**Moyenne**	**Médiane**	**Écart-type**	$\mathbf{0 \leq x < 2}$	$\mathbf{2 \leq x < 4}$	$\mathbf{4 \leq x < 6}$	$\mathbf{6 \leq x < 8}$	$\mathbf{8 \leq x \leq 10}$
Mathématiques	5,31	5,5	2,21	10%	27%	28%	25%	10%
Informatique	5,83	6	2,05	4%	19%	36%	27%	14%

Résumé statistique par partie de l'épreuve

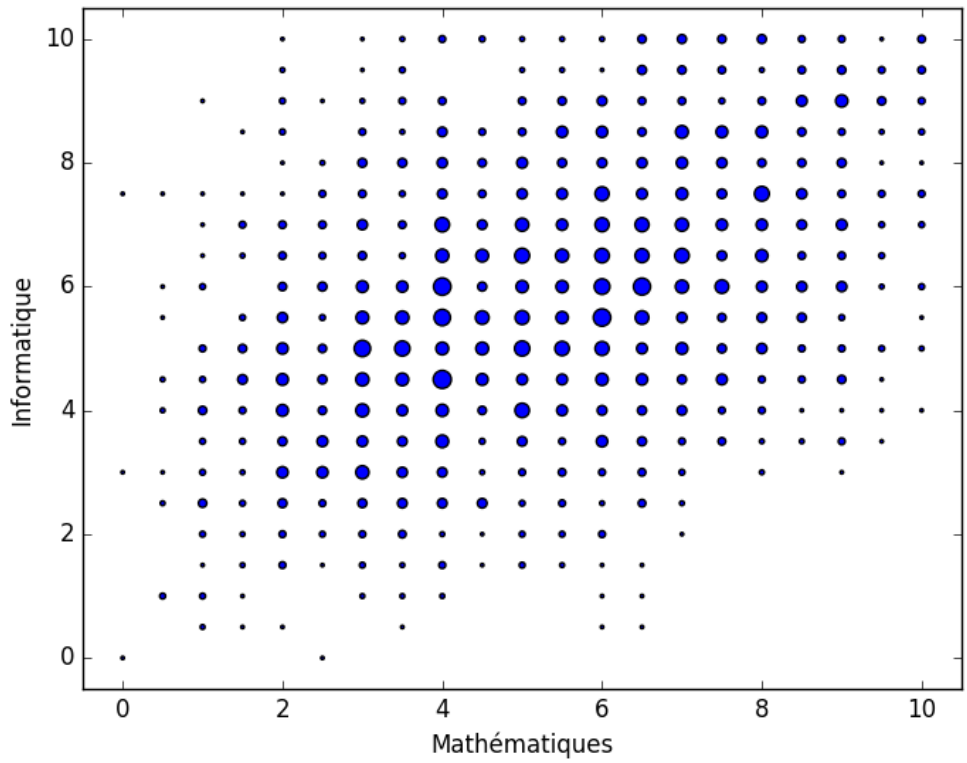
	Moyenne	**Médiane**	**Écart-type**	$\mathbf{0 \leq x < 4}$	$\mathbf{4 \leq x < 8}$	$\mathbf{8 \leq x < 12}$	$\mathbf{12 \leq x < 16}$	$\mathbf{16 \leq x \leq 20}$
Note finale	11,14	11	3,60	2%	20%	39%	31%	8%

Résumé statistique général



Histogrammes des notes

Dans le graphique suivant, la taille du disque de centre (x, y) est proportionnelle au nombre de candidats ayant obtenu la note x à la partie Mathématiques pratiques et y à la partie Informatique. Comme dit plus haut, les notes sont assez faiblement corrélées ; on pourra quand même noter que les candidats ayant excellé en mathématiques ont presque tous obtenu des notes satisfaisantes pour leur projet (l'inverse est moins vrai). Ce graphique est très similaire à celui de l'année dernière.



3 Partie Mathématiques Pratiques

3.1 Ce qui est attendu des candidats en Mathématiques Pratiques

Dans un but d'équité, tous les candidats convoqués à une même heure sont examinés sur le même sujet, quel que soit leur jury d'interrogation. Les examinateurs connaissent les énoncés sur lesquels portent l'interrogation, il est donc inutile de les leur rappeler.

Pendant la période d'interrogation, un dialogue entre le candidat et l'examinateur s'instaure. L'oral est un échange, n'est pas un écrit, et en aucun cas n'est une conférence du candidat face à un jury muet. Les questions du jury visent à évaluer les compétences du candidat, et cherchent avec bienveillance à l'aider pour compléter son argumentation ou se rendre compte d'une erreur. Elles ne constituent en aucun cas des pièges tendus. Le jury attend de la bonne volonté et une attitude ouverte pour répondre aux questions.

Il n'est pas indispensable d'avoir traité la totalité de l'exercice pour obtenir une excellente note. Il est préférable d'avoir mené un raisonnement rigoureux et argumenté, reposant sur des connaissances solides, plutôt que d'avoir donné tous les résultats (même justes), trop vite et sans explication réelle.

Au début de l'interrogateur, le candidat est invité à préciser la liste des questions qu'il a eu le temps d'aborder pendant la préparation. Le jury tient à préciser que cette question n'a pas pour but d'évaluer a priori le candidat, mais permet de gérer au mieux le déroulement de l'oral, et éviter que certaines questions préparées par le candidat n'aient pas le temps d'être exposées.

Chaque sujet comporte au moins une question d'informatique et les examinateurs interrogent systématiquement sur au moins une de ces questions, qu'elle ait été préparée ou non par le candidat en amont. Le cas échéant, le jury peut poser une question élémentaire pour vérifier les compétences du candidat en informatique.

La durée de l'interrogation orale doit impérativement être respectée. Lorsque l'examinateur annonce la fin du temps réglementaire, il est attendu que le candidat efface efficacement le tableau et range rapidement ses affaires, afin de ne pas pénaliser le candidat suivant.

3.2 Remarques générales sur la forme de l'oral Mathématiques Pratiques

- Pour la deuxième année où cet oral est mis en place, nous avons encore une fois fortement apprécié que les candidats soient bien préparés à l'épreuve. Nous tenons à féliciter les préparateurs qui ont manifestement tenu compte des remarques du rapport 2015 dans la préparation de leurs étudiants.
- Le niveau général des candidats est plutôt bon. Peu de candidats sont en grande difficultés face aux notions du programme. La connaissance du cours est globalement satisfaisante. En revanche, certains candidats montrent toujours de grosses lacunes en calcul et en analyse (calculs de dérivées et primitives, connaissance des fonctions usuelles,...).
- L'énoncé des définitions et des théorèmes au programme doit être précis, notamment sur le rappel des bonnes hypothèses. Le jury se réserve le droit de poser une question de cours à tout moment de l'interrogation s'il le juge nécessaire, pour s'assurer de la bonne compréhension du candidat.
- Le jury tient à encourager les candidats à émettre un avis critique sur leurs résultats lorsqu'ils sont clairement incohérents (tableau de variations en désaccord avec les limites, obtention d'une probabilité supérieure à 1, etc.)
- Faire l'impasse sur une ou plusieurs parties du programme de mathématiques ou d'informatique peut avoir de lourdes conséquences sur la prestation du candidat et peut être très pénalisant pour son évaluation.
- Les candidats doivent pouvoir s'adapter à des sujets d'oraux parfois originaux, de longueur très diverse, notamment ceux où il est attendu un peu de modélisation. Les examinateurs connaissent les sujets et leur difficulté relative, et adaptent leurs attentes et la notation en conséquence. Certains candidats sortent de leur préparation en ayant traité très peu de questions, et commencent leur oral en étant déstabilisés. L'examinateur trouvera toujours le moyen d'évaluer lors de la discussion les connaissances du candidat en le guidant dans la résolution ou en lui posant des question supplémentaires.

3.3 Remarques générales sur le contenu

- La plupart des exercices proposés en algèbre était élémentaire. Cependant, le jury a regretté la mauvaise connaissance des candidats sur les notions de bases mises en jeu. Les candidats confondent régulièrement les propriétés des matrices (une matrice triangulaire n'est pas forcément inversible, une matrice inversible n'est pas forcément diagonalisable, etc.)

- Nous invitons les futurs candidats à bien maîtriser les exercices classiques d'algèbre linéaire, en particulier sur les fonctions polynomiales (factorisations, utilisation des degrés, dérivées successives) et sur la réduction des endomorphismes (bon emploi des éléments propres d'une matrice, confusions entre rang de A et rang de $A - \lambda I_n$).
- Il est rappelé que la notion de polynôme annulateur d'une matrice ou d'un endomorphisme n'est pas au programme en BCPST. Il est attendu des candidats qu'ils refassent le raisonnement qui conduit au résultat bien connu liant polynôme annulateur et valeurs propres.

- Dériver ou primitiver une fonction n'est pas un acquis de tous les candidats. L'inversion entre ces deux notions est trop souvent rencontrée.
- La résolution des équations différentielles linéaire du premier ordre avec second membre n'est pas maîtrisée. Nombreux sont les candidats qui ne savent pas utiliser à bon escient la méthode de variation de la constante.
- Les sommes de Riemann sont peu reconnues par les candidats, même lorsque l'examinateur guide énormément le candidat vers ce résultat.
- Il est rappelé que l'inégalité des accroissements finis n'est pas au programme en BCPST, de même pour le théorème du point fixe pour l'étude des suites récurrentes. Il est attendu des candidats qu'ils refassent le raisonnement qui conduit à l'un de ces résultats.
- Nous invitons les candidats étudiant une variable aléatoire (à densité ou non) à s'intéresser aux valeurs prises par celle-ci avant de démarrer toute étude. Cela peut permettre de simplifier les raisonnements, tant pour la loi que pour la recherche des moments.
- L'utilisation de la formule du produit de convolution (systématiquement rappelée dans les énoncés) n'est pas un acquis de tout candidat, même dans des cas simples.
- L'utilisation des théorèmes limites en probabilités au programme (loi faible des grands nombres, théorème central limite) est mal maîtrisée par les candidats. Même s'ils en comprennent souvent l'idée générale, peu parviennent à formuler mathématiquement ces résultats.

3.4 Remarques générales concernant les questions d'informatique

- Nous constatons à nouveau que les candidats ont été bien préparés à l'utilisation des logiciels au programme pour accompagner l'étude de problèmes mathématiques. Nous tenons une nouvelle fois à féliciter leurs professeurs dévoués qui se sont vraisemblablement investis dans l'acquisition par leurs étudiants de ces nouvelles compétences.
- Nous avons vu une très nette amélioration par rapport à 2015 dans la bonne connaissance du langage Python, notamment pour l'usage du `range`. De manière générale, sur l'informatique, toutes les remarques mises en évidence dans le rapport 2015 ont été sensiblement améliorées, ce qui a été fortement apprécié par l'ensemble des examinateurs.
- La simulation des variables suivant les lois usuelles (uniforme, bernoulli, binomiale, géométrique, exponentielle) est à présent bien maîtrisée. Certains sujets demandaient se simuler une loi de Poisson à l'aide d'une loi binomiale, ce qui a surpris la majorité des candidats interrogés. Les candidats devraient donc avoir en tête que pour simuler une loi de Poisson de paramètre λ, on peut utiliser une loi binomiale $\mathcal{B}(n, \lambda/n)$ avec n grand.

3.5 Conclusion

Le niveau des candidats est très hétérogène, avec tous les intermédiaires entre le candidat brillant et le candidat n'ayant quasiment aucune connaissance ni compétence. L'examinateur s'attache à poser des questions adaptées pour évaluer le niveau du candidat de manière exacte. Le ressenti de certains candidats n'est pas toujours en adéquation avec la note obtenus. Tous les jurys s'efforcent d'être aimables et bienveillants, mais n'en ont pas moins pour mission d'évaluer et de classer les candidats.
L'ensemble des examinateurs a apprécié l'attitude respectueuse et courtoise de tous les candidats, ce qui a facilité le bon déroulement des oraux de cette session 2016.

4 Partie Informatique

4.1 Modalités de l'épreuve

La partie de l'épreuve consacrée au projet informatique durait vingt minutes : dix minutes (au maximum) de présentation, suivies d'un entretien. Le but de l'entretien est avant tout de juger de la maîtrise du projet par le candidat, mais également d'évaluer de manière plus générale ses compétences en informatique.

Pour le premier point, le candidat doit être capable de répondre précisément aux questions qu'on peut lui poser sur le code : quels sont les arguments de cette fonction, que renvoie-t-elle si on l'appelle sur telle valeur... La compréhension à plus haut niveau du projet est également évaluée, par exemple en demandant au candidat les difficultés qu'il y aurait à rajouter une certaine fonctionnalité.

Pour évaluer le deuxième point, les examinateurs ont demandé aux candidats de traiter au tableau un très court exercice de programmation. Le plus souvent, cet exercice était basé sur le projet : rendre une fonction plus générale ou plus efficace, ré-écrire de manière plus concise une partie du projet traitée maladroitement... Si le projet ne se prêtait pas à ce type de question, l'exercice pouvait être sans rapport direct avec le sujet. Les candidats présentant un projet ambitieux et apparemment bien maîtrisé mais incapables d'écrire au tableau une fonction parfaitement élémentaire (déterminant le maximum d'une liste, par exemple) ont été fortement pénalisés.

4.2 Remarques générales

Nous reprenons ici une grande partie des points évoqués dans le rapport du concours 2015. Cependant – et nous y reviendrons plusieurs fois par la suite – il faut tout de suite noter que le niveau général des candidats était sensiblement meilleur cette année.

- La très grande majorité des candidats semble s'être investis dans leur projet, et avoir ce faisant acquis des compétences qui leur seront utiles tant dans la suite de leurs études que dans leur vie professionnelle. Les projets étaient en moyenne un peu mieux écrits et assez nettement plus ambitieux que l'année dernière : dans le cadre d'un concours, cela signifie naturellement qu'un même projet, toutes choses égales par ailleurs, aurait été moins bien évalué cette année que l'année dernière.
- Les candidats extrêmement faibles (qui n'ont pas vraiment compris ce qu'était une variable ou une fonction ou ne maîtrisent pas la sytaxe élémentaire de Python) ont presque disparu.
- De trop nombreux candidats présentent un projet qu'ils ne maîtrisent que très imparfaitement. Ils sont certes capables d'en exposer le principe et l'architecture, voire d'expliquer le rôle des différentes fonctions, mais leur incapacité à y apporter la moindre modification pose question.
- Les présentations orales, malgré quelques défauts récurrents évoqués plus bas, étaient globalement satisfaisantes. On peut même considérer que beaucoup de candidats ont passé *trop de temps* à produire des diapositives de grande qualité et apprendre par cœur leur présentation. S'il est toujours appréciable de disposer de quelques schémas, la qualité visuelle de la présentation n'est pas réellement évaluée (et il y a une limite au nombre de diapositives que l'on peut réellement exploiter pendant les dix minutes de présentation).
- Aucun problème majeur d'organisation n'a été constaté cette année.

4.3 Remarques sur la présentation

Lors de la phase de présentation du projet, le candidat dispose du diaporama qu'il a déposé préalablement par voie électronique et qui sert de support à sa présentation. L'usage de notes est proscrit.

Une bonne présentation doit être synthétique. Le jury apprécie les candidats qui présentent l'enjeu de leur projet, les hypothèses qui ont été faites dans la modélisation choisie (le cas échéant), l'architecture générale du projet et une analyse des résultats et des perspectives ultérieures. Nous attirons l'attention des candidats sur le fait que la présentation des structures de données et algorithmes utilisés est cruciale.

Certains écueils sont à éviter : une présentation de concepts biologiques durant cinq minutes, ou un cours de physique sont à proscrire. *A contrario*, certains candidats n'explicitent pas leur modèle, implémentent des jeux dont ils n'expliquent pas les règles, ou plus généralement ne fixent pas les contours de leur projet. Nous rappelons aux candidats que l'examinateur n'est pas censé interrompre le candidat durant cette phase, et que si les examinateurs ont pris connaissance du projet du candidat, ils n'ont pas passé une année à travailler sur le projet ; il faut donc éviter de se retrouver dans une situation où l'examinateur ne voit pas où le candidat veut

en venir. Pour un jeu, demander à l'examinateur s'il connaît déjà les règles ou s'il souhaite qu'on les lui explique rapidement semble une idée raisonnable.

Il est fortement recommandé de commenter deux ou trois fonctions centrales durant cette phase de présentation. De préférence, le candidat choisira celles présentant des difficultés algorithmiques. Il est par contre inutile de lister l'ensemble des fonctions présentes dans le programme (y compris dans un diagramme d'appels qui sera en général illisible).

Une conclusion est évidemment la bienvenue, dans des proportions raisonnables : présenter les résultats de sa modélisation durant la moitié de l'oral est donc à éviter. Il peut être intéressant de proposer des améliorations possibles, mais seulement si le candidat y a effectivement réfléchi.

Une dernière remarque : la notice demande aux candidats de présenter « les éventuelles bibliothèques et outils logiciels utilisés en plus de Python et de sa bibliothèque standard » : le module `random` fait partie de la bibliothèque standard, et même si ce n'est pas le cas des modules `numpy` et `matplotlib`, il est inutile de préciser qu'on les utilise.

4.4 Thèmes et qualité des projets

Le thème du projet était complètement libre, ce qui permet en règle générale aux candidats de choisir un sujet qui les motive. Il est important de choisir un projet dont la difficulté soit en adéquation avec le niveau du candidat : comme nous l'avons dit plus haut, de nombreux candidats présentent des projets trop ambitieux pour eux et prennent ainsi le risque d'obtenir une note très faible par rapport à leur investissement. Inversement, nous attirons l'attention des préparateurs sur l'augmentation de la qualité des projets « moyens » : un minimum d'ambition est désormais nécessaire pour obtenir une bonne note. En revanche, le nombre de candidats excellents et la qualité de leurs projets n'a pas évolué de manière significative.

Le jury rappelle que le temps passé à travailler les graphismes et l'interface du projet est beaucoup moins productif d'un point de vue comptable que celui passé à améliorer les aspects algorithmiques.

Plutôt que de faire des suggestions de sujets intéressants (presque trop bien suivies cette année : nous avont vu un nombre impressionnant de projets sur l'alignement de séquences), nous pointons ici quelques écueils :

- les jeux de société ont souvent des règles assez compliquées dont la traduction informatique ne présente que peu d'intérêt mais demande du temps. Programmer un Monopoly, par exemple, demande de traiter toute une série de cas particuliers et de cartes aux effets variés mais ne permet pas vraiment de mettre en valeur les compétences du candidat ;
- de même, il est assez difficile de faire un bon projet autour de la bataille navale. Certains candidats ont montré que c'était possible, mais la plupart de ceux qui ont essayé n'ont pas été très bien payés de leurs efforts ;
- de manière plus générale, le code ne devrait pas faire apparaître de longues disjonctions de cas. Si de telles séries de `if...elif...` semblent nécessaires au candidat, c'est sans doute que le sujet est trop difficile à traiter correctement ou qu'il se prête mal à un traitement informatique ;
- à peu près aucun candidat n'est capable d'explique pourquoi l'algorithme de Dijkstra fonctionne : ce n'est pas en soi gênant, sauf si le projet se limite à un enrobage sommaire autour d'une implémentation de cet algorithme ;
- un sujet intéressant (par exemple sur le traitement du signal) peut facilement déboucher sur un mauvais projet si l'on se limite à appeler les fonctions d'une bibliothèque.

Certains projets se prêtent bien à une analyse statistique des résultats (par exemple en faisant varier certains paramètres dans la simulation de la propagation d'un virus) : en général, cette analyse a été faite par les candidats.

La taille moyenne des projets a nettement augmenté : c'est plutôt une bonne chose, mais il nous semble souhaitable que cette inflation cesse. Un projet dense d'une centaine de lignes, bien présenté et bien maîtrisé, peut tout à fait obtenir la note maximale. Un projet faisant exactement la même chose en trois ou quatre cents lignes obtiendra sans doute une moins bonne note. Globalement, on a noté un effort des candidats pour éviter le code très répétitif (quatre fonctions différentes pour se déplacer dans les quatre directions par exemple) : ceux n'ayant pas fourni cet effort ont naturellement été davantage pénalisés que l'année dernière.

4.4.1 Modélisation

Nous appelons les candidats à se reporter à la section correspondante du rapport de l'année dernière : les remarques qui s'y trouvent ont été assez bien prises en compte mais restent valables.

4.4.2 Style et lisibilité du code

On n'attend bien sûr pas des candidats qu'ils écrivent du code de qualité professionnelle. Nous nous contentons de donner ici quelques conseils pour améliorer la qualité stylistique du code, qui a un impact important tant sur sa lisibilité (par les différents membres du groupe et par l'examinateur) que sur la possibilité d'y détecter facilement les erreurs et d'y apporter des améliorations ultérieures.

- Pour faciliter le travail des examinateurs, les candidats sont appelés à fournir, dans les premières lignes de leur script, une fonction ne prenant pas d'argument et permettant de tester leur projet (avec des paramètres par défaut raisonnables le cas échéant).
- Le code doit être commenté, et nous remarquons que les candidats ont fait des efforts louables sur ce point. Il n'est certainement pas nécessaire de commenter chaque ligne, mais un commentaire d'une ou deux lignes au début de chaque fonction expliquant ce qu'elle prend en argument, ce qu'elle fait et ce qu'elle renvoie est très appréciable.
- Dans les noms de variables, de fonctions *et également de fichiers*, il faut éviter tous les caractères dits *spéciaux*, c'est-à-dire se restreindre aux lettres non accentuées (minuscules et majuscules), aux chiffres et au « tiret bas » `_`. Cette remarque, déjà présente dans le rapport de l'année dernière, n'a pas eu l'effet escompté...
- Éviter d'utiliser des chemins absolus : si l'examinateur souhaite tester le programme sur sa machine, il y a peu de chance qu'un appel `f = open("D:/MonProjetInfo/Donnees/especes.txt")` fonctionne (écrire à la place `f = open("Donnees/especes.txt")`). Ces chemins relatifs ne fonctionnent sous Pyzo que si l'on utilise la commande *Execute file as script* (raccourci clavier `Ctrl-Maj-E`).
- La longueur des lignes doit être raisonnable (idéalement, pas plus de 80 colonnes). Pour faciliter cela, on pourra, dans le menu *View* de Pyzo, choisir une *Location of long line indicator* à 80 colonnes et désactiver l'option *Wrap long lines*. Au besoin, on peut placer manuellement un retour à la ligne en utilisant un `\` :
```
x = nom_de_variable_excessivement_long + \
    encore_une_variable_dont_le_nom_est_tres_long
```
- Si l'on s'intéresse, par exemple, à l'évolution d'une population évoluant dans un environnement en deux dimensions représenté par une matrice, la taille de cette matrice gagnera à être stockée dans une variable (passée en paramètres aux différentes fonctions ou globale et définie au début du programme). On ne devrait jamais trouver une ligne telle que `for i in range(50)` dans un programme.
De manière plus générale, on essaiera toujours d'écrire un code le plus générique possible : un projet ne fonctionnant que sur un exemple précis ne présente le plus souvent que peu d'intérêt.
- De même, plutôt que d'écrire `if M[i][j] == 2` avec éventuellement un commentaire précisant « on regarde si la cellule est vivante », on préférera définir `VIVANTE = 2` au début du programme et écrire ensuite `if M[i][j] == VIVANTE`.

4.4.3 Notions « hors-programme »

Le jury ne pénalise pas *a priori* les candidats utilisant des notions hors-programme (ou à la limite du programme) comme les fonctions récursives, les dictionnaires ou la programmation orientée objet. Dans certains projets, leur emploi peut être très judicieux voire presque indispensable dans le cas des fonctions récursives. Cependant, un candidat utilisant ces notions doit savoir qu'il s'expose à des questions (élémentaires) portant dessus et être capable d'expliquer l'intérêt de leur utilisation dans son projet. On ne demande bien sûr pas aux candidats de savoir ce qu'est une fonction récursive terminale, mais un candidat ayant écrit une telle fonction et n'étant pas capable d'en écrire une version itérative n'a pas compris ce qu'il faisait.

4.5 Remarques spécifiques de programmation et d'algorithmique

Nous reprenons ici les remarques faites l'année dernière, avec quelques clarifications.

- Le code doit absolument être découpé en fonctions, et il faut réfléchir soigneusement au découpage le plus judicieux. Déjà rares l'année dernière, les candidats n'utilisant pas de fonction (ou mettant tout leur code dans une seule fonction, ce qui revient au même) étaient exceptionnels cette année : ils ne peuvent obtenir une note correcte.
- On sépare en règle générale les fonctions effectuant des entrées-sorties (affichage à l'écran, lecture dans un fichier...) des autres. Ainsi, on n'utilisera normalement pas une même fonction pour calculer et afficher une valeur.

- De nombreux projets nécessitent de parcourir les voisins d'une case d'un tableau bidimensionnel, ce qui avait souvent donné lieu à du code extrêmement maladroit l'année dernière. On constate une très nette amélioration cette année, ce que le jury a apprécié.
- Il est fortement déconseillé d'utiliser un algorithme que l'on n'est pas capable d'expliquer correctement lorsque l'examinateur le demande. Si le projet est riche par ailleurs, le candidat peut éventuellement préciser qu'il s'est contenté de reprendre une correction de Travaux Dirigés ou que son professeur l'a aidé. Malheureusement, de nombreux projets étaient très pauvres, à l'exception d'une fonction relativement compliquée et non maîtrisée : dans ce cas, la note sera faible.
- L'algorithme de Dijkstra n'a d'intérêt que si le graphe que l'on considère est *pondéré* ! Pour la recherche d'un plus court chemin dans un graphe non pondéré, on utilisera un parcours en largeur.
- Il est souvent possible en Python d'utiliser une boucle `for` là où une boucle `while` serait nécessaire dans certains langages (à l'aide typiquement d'un `return` dans le corps de la boucle). Les candidats sont encouragés à tirer partie de cette possibilité, mais *cela ne les dispense pas de savoir écrire une boucle* `while` *en gérant correctement les conditions de sortie.* Sur ce point déjà évoqué l'année dernière, on ne constate que très peu d'amélioration cette année.
- Très peu de candidats utilisent correctement les booléens. Idéalement, on aimerait lire plus souvent des `return x > 0` (par exemple), mais le jury est conscient que ce n'est pas évident conceptuellement. En revanche, éviter les `if f(x) == True` (ou, pire, `if f(x) == 'Ca marche !'`) semble un objectif raisonnable.
- Très peu de candidats maîtrisent la différence entre une fonction renvoyant une valeur et une fonction modifiant son argument. C'est finalement assez compréhensible, mais peut malheureusement résulter en un très grand nombre de copies inutiles (copier toute la matrice à chaque fois que l'on modifie une case par exemple) qui sont parfois la principale raison pour laquelle le programme est trop lent pour traiter des données de taille réaliste.
- Certains candidats codent des chaînes de caractères très longues, ou des matrices de « grande » taille directement dans leur programme. Dans nombre de cas, utiliser un fichier externe aurait été plus adapté, et aurait permis aux candidats d'illustrer leur maîtrise de compétences acquises durant la préparation.