SCAV
Service des Concours Agronomiques et Vétérinaires
AgroParisTech

CONCOURS ABCPST - SESSION 2019

ADMISSION

RAPPORT DE L'ÉPREUVE ORALE DE MATHÉMATIQUES PRATIQUES ET INFORMATIQUE

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 Pyzo et Spyder (Python 3), Excel et Geogebra.

Les candidats disposent d'une clé USB fournie par le concours au début de leur préparation, sur laquelle il leur est demandé d'enregistrer leur travail.

À l'issue de la préparation, ils sont accompagnés dans une salle d'interrogation (5 minutes au maximum sont prévues pour la transition), où ils exposent pendant environ 18-20 minutes le résultat de leur travail de préparation et ils dialoguent 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.

À l'issue de ce premier temps d'interrogation, 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 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 reste comme les années précédentes assez faible, voire en légère baisse ($r = 0{,}40$, $r^2 = 0{,}16$). En particulier, d'assez nombreux candidats ayant de grosses difficultés en mathématiques ont pu montrer des compétences très intéressantes en informatique.

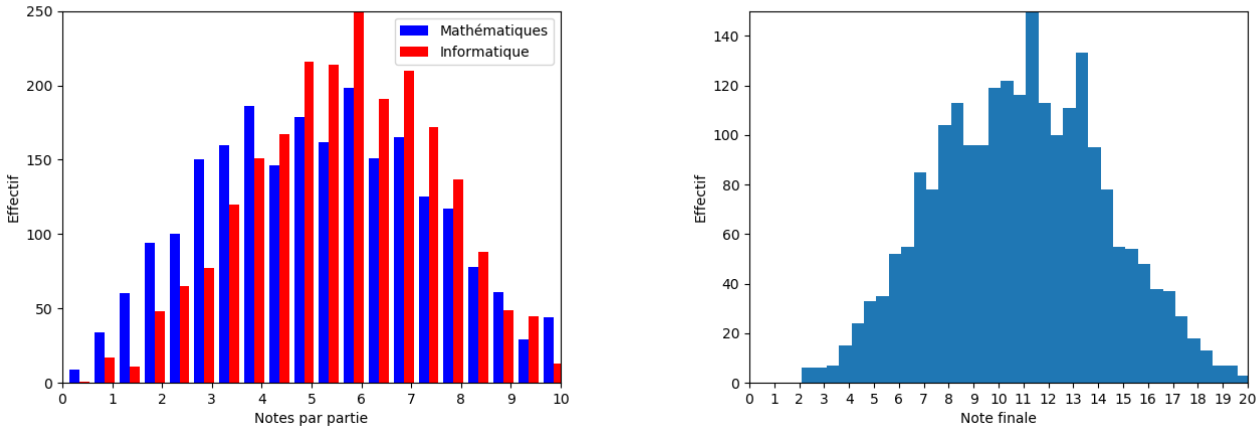
Les distributions des notes sont semblables aux années précédentes concernant chacune des parties, et les notes globales obtenues sont légèrement plus dispersées, plus de candidats ayant obtenu une note maximale dans les deux épreuves.

Partie	Moyenne	Médiane	Écart-type	$0 \leqslant x < 2$	$2 \leqslant x < 4$	$4 \leqslant x < 6$	$6 \leqslant x < 8$	$8 \leqslant x \leqslant 10$
Mathématiques	5,29	5,5	2,18	9%	27%	30%	25%	9%
Informatique	5,79	6	1,85	3%	18%	38%	32%	9%

Résumé statistique par partie de l'épreuve

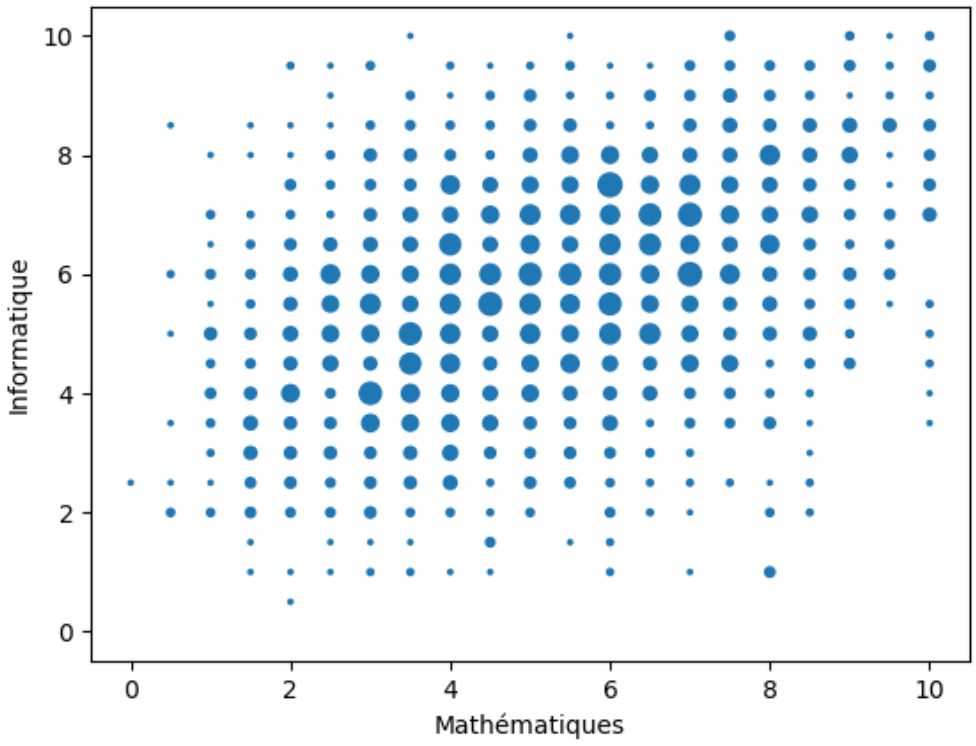
	Moyenne	Médiane	Écart-type	$0 \leqslant x < 4$	$4 \leqslant x < 8$	$8 \leqslant x < 12$	$12 \leqslant x < 16$	$16 \leqslant x \leqslant 20$
Note finale	11,08	11	3,37	1%	21%	41%	30%	7%

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 ; ce graphique est très similaire à celui des années précédentes.



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 préparent le même sujet, quel que soit leur jury d'interrogation. Les examinateurs connaissent les énoncés sur lesquels portent l'interrogation, il est donc inutile **et même déconseillé** de relire l'énoncé.

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 interrogateurs posent des questions visant 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.

Chaque sujet comporte une question de cours, qui précède l'énoncé de l'exercice à préparer. Cette question est indépendante de l'exercice, dans le sens où elle peut concerner un thème différent de celui abordé dans la suite.

Au début de l'interrogation (en général, juste après la question de cours), le candidat peut être invité par l'examinateur à préciser la liste des questions qu'il a eu le temps d'aborder pendant la préparation, mais nous conseillons aux candidats de le faire spontanément. Ceci n'a pas pour but d'évaluer *a priori* le candidat, mais permet simplement à l'examinateur de gérer au mieux le déroulement de l'oral, et d'éviter que certaines questions préparées par le candidat n'aient pas le temps d'être exposées. Pour le jury, « aborder une question » pendant la préparation signifie que le candidat en a traité au moins une partie, et ne s'est pas contenté d'essayer et achopper. Certains candidats maladroits affirment avoir « abordé » beaucoup de questions, alors qu'ils ne les ont en fait que survolées, ce qui est ensuite pénalisant pour la gestion du temps de l'oral.

La présentation de la liste des questions par le candidat doit néanmoins être brève, voire réfléchie lors de la fin de la préparation avant d'entrer en interrogation (par exemple en annotant le sujet ou en surlignant les numéros des questions concernées). Beaucoup perdent trop de temps inutilement à donner cette liste de questions, une durée maximale de 10 secondes semble être suffisante.

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.

À partir de la session 2020, les candidats disposeront devant leur ordinateur (dans la salle de préparation et dans la salle d'interrogation) d'une feuille A4 aide-mémoire python, dont un exemplaire est disponible en page 15 de ce rapport. En effet, même si les sujets jusqu'à présent rappelaient les différentes fonctions à utiliser dans Python, l'aide-mémoire proposé permettra de familiariser les candidats avec différentes fonctions qui peuvent être demandées d'être utilisées par le jury dans la partie Mathématiques Pratiques. Cette liste n'est bien entendu qu'indicative, et les candidats sont libres d'utiliser d'autres modules ou fonctions de leur choix s'ils les connaissent. Toute autre fonction requise dans un sujet sera rappelée dans celui-ci, mais celles figurant sur l'aide-mémoire ne seront quant à elles pas nécessairement expliquées dans les sujets. Nous encourageons donc les candidats à se familiariser avec le document joint, et de s'habituer à l'utiliser durant l'année de préparation, puisqu'il sera imprimé tel quel lors de la session 2020. En fonction des attentes du jury, le document sera éventuellement amené à être modifié pour les sessions ultérieures.

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. Nous rappelons enfin que l'interrogation débute dès l'entrée dans la salle, l'examinateur gère le temps d'interrogation au mieux, le candidat n'a donc pas besoin de chronomètre pendant l'épreuve (l'heure étant également affichée sur l'ordinateur si besoin).

3.2 La question de cours

Depuis la session 2018, chaque sujet comporte une question de cours avant l'exercice principal. La question de cours est écrite sur le sujet (et non posée à l'oral par l'examinateur), le candidat en prend donc connaissance au début de sa préparation avec le reste du sujet.

Les examinateurs ont constaté à plusieurs reprises des candidats n'ayant pas préparé cette question et semblant la découvrir lorsque l'examinateur leur demande de l'exposer. Il serait bon, de manière générale, que les candidats lisent bien attentivement le sujet lors de leur préparation, et se préparent de manière adéquate durant l'année à cette question de cours.

Les questions de cours peuvent être diverses, cela peut être entre autres :

- l'énoncé d'une définition complète d'une notion au programme,
- l'énoncé clair et complet d'une propriété ou d'un théorème au programme (dans ce cas, la mention des hypothèses nécessaires est requise),
- le tracé de l'allure représentative d'une fonction usuelle,
- une application directe d'une formule du cours (une utilisation d'une formule de dérivée ou de primitive dans un cas précis, l'écriture d'une formule dans un cas particulier, le calcul de l'inverse d'une matrice 2×2, ...).

À part un calcul immédiat résultant de l'application directe d'une formule du cours, aucune démonstration n'est attendue par les candidats. La rédaction des questions s'appuie en priorité sur le Programme Officiel de BCPST (rentrée 2013) et peut couvrir l'ensemble du programme des deux années de classe préparatoire. Le but est simplement de vérifier brièvement la connaissance d'un point du cours, et ne doit pas donner lieu à un développement trop long.

Si cela est pertinent, le candidat peut se contenter d'une réponse orale. Il peut arriver cependant que le jury demande explicitement une réponse écrite pour lever une ambiguïté d'un énoncé oral de la propriété souhaitée. Par exemple pour la caractérisation de l'injectivité d'une application linéaire par son noyau, les réponses énoncées à l'oral sont souvent hasardeuses du type « le noyau est différent de zéro », et à l'écriture, on voit en effet apparaître souvent « $\mathrm{Ker}(f) = \{0\}$ » comme attendu, mais parfois « $\mathrm{Ker}(f) = 0$ » ou bien encore « $\mathrm{Ker}(f) = \emptyset$ ».

Dans la plupart des cas, l'examinateur se contente d'écouter la question de cours, sans la reprendre avec le candidat, même si elle est partiellement ou totalement incorrecte. Cette question a pour première ambition de pouvoir valoriser un candidat sérieux mais ayant des difficultés sur le reste du sujet. Elle a aussi pour but d'inciter les candidats à travailler régulièrement l'apprentissage de leur cours, ceci sur leurs deux années de classe préparatoire, ce qui donne bien entendu un poids supplémentaire aux consignes des professeurs de BCPST données à leurs élèves sur l'importance d'avoir une bonne connaissance du cours. Nous sommes tous convaincus qu'une meilleure maîtrise du cours aura une répercussion immédiate sur les capacités des candidats à résoudre les exercices proposés à l'oral.

Les questions sont choisies de manière aléatoire, avec remise éventuelle lors d'une même session. Il est donc inutile pour certains candidats de faire l'impasse sur une question déjà tombée lors d'une session (passée ou en cours), cette question pouvant être réutilisée de manière immédiate ou future. La liste des questions de cours utilisées lors de la session 2018 a été publiée (voir rapport 2018). Nous avons apprécié que les questions qui avaient été mal comprises en 2018 aient été nettement améliorées en 2019, preuve que les candidats ont travaillé avec soin la liste des questions de cours qui avaient été données.

Cependant, nous précisons une nouvelle fois que cette liste n'était pas exhaustive, et que la connaissance du cours doit porter sur l'intégralité du programme. Sur toutes les nouvelles questions de cours qui ont fait leur apparition cette année, les réponses étaient beaucoup plus hasardeuses, et les candidats semblaient plus déstabilisés.

Nous publions ci-après une liste de quelques questions de cours ajoutées en 2019 par rapport à celles de 2018.

Quelques exemples de questions de cours ajoutées en 2019
Définition du produit scalaire dans $\mathbb{R}^n$.
Définition de la distance d'un vecteur x de $\mathbb{R}^n$ à un sous-espace vectoriel F de $\mathbb{R}^n$.
Développement limité à l'ordre 5 au voisinage de 0 de la fonction sinus.
Développement limité d'ordre 5 au voisinage de 0 de la fonction cosinus.
Si f est la fonction définie sur $]0,1[$ par : $f(x) = \sqrt{1-x}$, déterminer l'expression d'une de ses primitives sur l'intervalle $]0,1[$.
Si f est la fonction définie sur $]0,1[$ par : $f(x) = \sqrt{1-x}$, déterminer l'expression de sa dérivée f'.
Si α est un réel quelconque, déterminer sur $]0,+\infty[$ l'expression d'une primitive de la fonction $x \mapsto \dfrac{1}{x^\alpha}$.
Allure de la représentation graphique de la fonction sin sur l'intervalle $[-\pi,\pi]$.
Allure de la représentation graphique de la fonction cos sur l'intervalle $[-\pi,\pi]$.
Allure des représentations graphiques des fonctions $x \mapsto \ln(x)$ et $x \mapsto \ln(1+x)$
Allure des représentations graphiques des fonctions $x \mapsto \ln(x)$ et $x \mapsto \lvert \ln(x) \rvert$
Donner la définition de la fonction de répartition d'une variable aléatoire réelle. Donner son tableau de variation.
Variance de la différence de deux variables aléatoires. Cas particulier où les deux variables sont indépendantes.
Formule de l'espérance de $Z = u(X,Y)$ où X et Y sont des variables aléatoires discrètes à valeurs dans $[\![1,n]\!]$ et u une fonction de $\mathbb{R}^2$ dans $\mathbb{R}$.

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

- Comme les années précédentes (l'oral sous cette forme entre dans sa cinquième année d'existence), nous avons encore une fois fortement apprécié que les candidats soient bien préparés à l'épreuve. Nous tenons une nouvelle fois à féliciter les enseignants de BCPST qui tiennent vraisemblablement compte de nos remarques issues des rapports précédents dans la préparation de leurs étudiants.

- Le niveau général des candidats est plutôt bon, notamment en probabilités discrètes. Peu de candidats sont en grande difficulté face aux notions du programme. La connaissance du cours est globalement bonne, peut-être grâce à la liste des questions de cours qui avait été publiée l'an passé. 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. L'évaluation du cours n'est pas limitée à la question posée en début de sujet.

- Le jury rappelle aux candidats que les calculs qui ont pu être réalisés lors de la préparation ne doivent pas forcément être totalement développés au tableau pendant l'interrogation. Les candidats peuvent avoir une part de **recul et de synthèse** vis-à-vis de leurs brouillons, afin de gagner

du temps lors de leur passage dont la durée est limitée. Le candidat peut alterner entre l'écriture au tableau et la formulation à l'oral des hypothèses et des méthodes mises en oeuvre. Par exemple, un candidat qui applique la formule des probabilités totales peut se contenter d'expliquer à l'oral le système complet d'événements qu'il utilise et écrire directement le résultat qu'il obtient. Il ne faut pas confondre interrogation orale et copie d'écrit.

- 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.). S'aider d'un graphique ou d'un schéma est fortement valorisé dans certaines situations par les examinateurs. On peut déplorer qu'en particulier les tracés de courbes qui peuvent être demandés spontanément à l'oral, notamment sur les fonctions usuelles, sont trop souvent erronés, alors qu'ils peuvent permettre à des candidats de visualiser des erreurs manifestes dans leurs raisonnements ou leurs conclusions. Le jury demandera par conséquent aux candidats plus souvent dans les années à venir de faire des tracés de représentations graphiques, par exemple dans les questions de cours.

- 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. C'est donc bien entendu une stratégie à prohiber.

- Les candidats doivent enfin pouvoir s'adapter à des sujets d'oraux parfois originaux, de longueurs très diverses, 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 questions supplémentaires.
  En particulier, lorsque le sujet comporte une modélisation ou plusieurs définitions, et que le sujet comporte un en-tête un peu long, nous encourageons les candidats à consacrer du temps à essayer d'assimiler au mieux les notations introduites afin de bien comprendre le sujet, et ne pas se ruer sur la résolution des questions au risque de passer à côté d'informations importantes, voire d'avoir mal compris le cadre de l'exercice. Il vaut mieux aborder moins de questions pendant la préparation, mais s'être bien imprégné des notions mises en jeu pour bien suivre les indications de l'examinateur lors de l'interrogation.

3.4 Remarques générales sur le contenu

Algèbre. Comme pour les années précédentes, la plupart des exercices proposés en algèbre linéaire et bilinéaire étaient élémentaires et leur résolution a été une fois de plus trop souvent décevante pour un trop grand nombre de candidats. Le manque de méthodes ou de discernement face aux différentes méthodes possibles pour répondre aux questions a conduit les candidats sur de mauvais raisonnements, alors que d'autres auraient souvent été plus adaptés.

Nous encourageons cette année encore les futurs candidats à améliorer leurs connaissances en algèbre linéaire. L'étude d'un endomorphisme de $\mathbb{R}^3$ ou de la diagonalisation d'une matrice 3×3 explicite doit être un attendu minimal pour tout candidat admissible, mais le jury sera vigilant à vérifier si les étudiants interrogés comprennent l'ordre des questions et les moyens mis en oeuvre pour leur éviter des calculs fastidieux. Par exemple dans de nombreux exercices, l'étude du rang de $A - \lambda I_n$ n'est pas nécessaire pour déterminer les valeurs propres de la matrice A. Mais les candidats ne lisant pas le sujet correctement ne voient pas la manière dont le sujet est construit, et ne font donc pas les liens entre les différentes questions et les diverses méthodes à leur disposition.

A contrario, les candidats utilisant des notions explicitement hors programme (par exemple que la trace d'une matrice soit égale à la somme des valeurs propres) seront lourdement sanctionnés s'ils ne connaissent pas a minima l'idée de la démonstration de ces dites notions dans le cas où ils veulent les utiliser.

Probabilités. Les candidats sont plutôt à l'aise avec les exercices de probabilités discrètes, tout en présentant des difficultés dès que l'on sort du cadre fini. Les candidats semblent se référer, voire se limiter à leurs connaissances acquises en première année, et mal percevoir l'évolution étudiée dans le programme de deuxième année sur les variables aléatoires discrètes infinies.

Le point qui nous semble le plus problématique reste l'étude élémentaire des variables aléatoires à densité. En priorité les questions classiques du type « déterminer la loi de $f(X)$ » lorsque X est une variable à densité et f une fonction simple, ont posé des problèmes insurmontables à un trop grand nombre de candidats. Ce point nous semble très inquiétant car malgré nos remarques dans les rapports précédents, nous ne voyons que très peu d'améliorations. Nous rappelons que l'étude du support de $f(X)$ avant de travailler sur la fonction de répartition peut permettre aux candidats faibles de leur éviter de traiter différents cas ensuite, ce qui peut quelque part les aider à combler un manque de rigueur dans la manipulation des événements ou des variables mises en jeu.

Enfin, le théorème de transfert pour les variables aléatoires à densité a été un trop grand nombre de fois mal appliqué par les candidats. C'est le second point qu'il nous semble important de souligner cette année, et nous espérons que les futurs candidats prendront le temps de veiller à une bonne compréhension de ce thème qui nous semble fondamental.

Analyse. Cette année, nous avons remarqué une nette amélioration dans les exercices d'analyse, en comparaison aux années précédentes. Sans doute les exercices classiques étaient mieux préparés, mais nous avons apprécié que les candidats ne soient pas bloqués trop rapidement dans les sujets, à cause de calculs élémentaires (dérivées, primitives) ou de méthodes (convergence de suites, ...). Les équivalents et les développements limités restent cependant difficiles à utiliser pour les candidats de niveau intermédiaire.

Informatique. Le niveau des candidats en informatique est chaque année supérieur à la session précédente, signe que les enseignants continuent à exercer un excellent accompagnement tout au long de la formation des étudiants.

Depuis la création de l'épreuve, le jury se demandait si la création d'un résumé « aide-mémoire », comme cela se fait à d'autres concours de CPGE scientifiques, pourrait aider les candidats durant leurs années de formation, et durant la préparation de l'oral. Nous avons décidé qu'à partir de la session 2020, ce type de formulaire serait disponible pour les candidats, et nous publions ce document conjointement au rapport pour qu'il soit utilisé par les préparateurs dès la rentrée 2019 pour la formation des candidats.

Pour cette session 2019, les tracés de courbes en Python sont encore une fois un exercice difficile pour beaucoup de candidats. Cela nous semble pourtant être un objectif à atteindre par chaque étudiant ayant travaillé en Python. Nous ne pouvons que conseiller aux candidats de niveau modeste et ayant des difficultés à maîtriser Python, de savoir le cas échéant a minima utiliser le logiciel Geogebra. En effet, ce dernier, sans avoir besoin d'être expert dans son usage, permet également de tracer rapidement des courbes à moindre coût. Il permet entre autres également de calculer des dérivées, des primitives, des développements limités, pour ceux qui savent où aller chercher ces informations.

Signalons une nouvelle fois que les sujets conseillent parfois d'utiliser des bibliothèques. Ce ne sont que des indications, les candidats peuvent très bien utiliser une autre méthode que celle proposée, et peuvent montrer toute l'autonomie qu'ils ont pu acquérir en deux (ou trois) ans d'informatique. C'est d'ailleurs ce qu'il se passera également l'an prochain avec le formulaire mis à disposition. La liste des commandes indiquées n'est qu'une simple proposition, et nous encourageons les candidats à avoir utilisé au moins une fois chacune de ces commandes qui pourrait être utile dans certains exercices ; néanmoins les candidats restent bien entendu libres d'utiliser toute fonction de leur choix.

Nous tenons une nouvelle fois à signaler qu'il serait bon que les candidats se préparent pendant l'année à savoir « présenter » rapidement un programme qu'ils ont réalisé, le temps d'interrogation étant très rapide au concours. Il est par exemple inutile de lire à l'oral la liste des bibliothèques qu'ils ont importées, ou de relire le programme ligne par ligne à l'examinateur. Il vaut mieux être concis et expliquer le fonctionnement du programme de manière synthétique.
Lorsqu'un programme utilise des simulations aléatoires, il est naturel que les candidats exécutent leur programme devant l'examinateur plusieurs fois, rien que pour témoigner que les résultats peuvent varier d'une exécution à l'autre. À cette fin, nous encourageons les candidats à se familiariser de façon optimale à l'usage de Pyzo ou Spyder, en particulier la gestion de la console et les commandes rapides d'exécution de leurs programmes (par exemple CTRL+E), afin de gagner du temps lors de l'oral.

Nombreux sont les candidats qui utilisent exclusivement l'éditeur et la fonction `print`, et n'utilisent jamais la console, ce qui reste lourd en terme d'exécution. Notamment, lors de simulation d'expériences aléatoires, où nous attendons des candidats qu'ils puissent exécuter plusieurs fois de suite leur programme, l'utilisation de la console semble bien plus appropriée et conduit parfois certains candidats à perdre du temps d'interrogation de manière inutile.

Enfin, certains candidats arrivent devant l'examinateur avec une clé USB vide. Il vaut toujours mieux pour un candidat avoir un début de programme sur sa clé USB à présenter, quitte à expliquer ensuite à l'examinateur comment il aurait achevé son programme à l'oral. Les candidats doivent en tout cas avoir en tête qu'il est toujours possible de modifier ou corriger leurs programmes pendant l'interrogation elle-même, l'ordinateur restant à leur entière disposition durant tout leur passage.

Par ailleurs, pour les candidats ayant réalisé un programme complet, il est dommage que certains d'entre eux n'essaient pas de l'exécuter au moins une fois lors de leur préparation. Cela leur permettrait peut-être de prendre conscience d'une erreur simple de syntaxe manifeste, facile à corriger, et qui fait toujours mauvais effet devant l'examinateur.
Nous espérons que les candidats sauront faire usage à partir de la session 2020 de l'aide-mémoire fourni ici en page 15 et que les prestations s'en trouveront améliorées.

3.5 Conclusion

Le niveau des candidats reste assez hétérogène, mêlant au sein du concours des candidats brillants et des candidats n'ayant acquis en deux (ou trois) ans de formation aucune connaissance ni compétence en mathématiques. L'examinateur s'attache dans tous les cas à poser des questions adaptées pour évaluer le niveau du candidat de manière circonstanciée.
Nous rappelons cette année encore que le ressenti de certains candidats n'est pas toujours en adéquation avec la note obtenue. 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. Les examinateurs sont tenus à un devoir de réserve quant à l'évaluation du candidat, ce qui peut parfois être interprété à tort par certains candidats pour de la froideur ou de l'indifférence. Un candidat peut ressortir déçu de son interrogation et peut néanmoins obtenir une note tout à fait satisfaisante, et inversement.
L'ensemble des examinateurs a une fois de plus apprécié l'attitude respectueuse et courtoise de tous les candidats, ainsi que l'organisation parfaitement ficelée et l'efficacité de l'équipe de surveillants, qui ont une nouvelle fois facilité le bon déroulement des oraux de cette session 2019.

4 Partie Informatique

Le jury est très satisfait du niveau atteint par les candidats et de la qualité globale des projets.

4.1 Modalités de l'épreuve

La partie de l'épreuve consacrée au projet informatique dure vingt minutes. Les candidats disposaient de sept minutes, au maximum, pour présenter leur projet, suivies d'un entretien. Le jury est très satisfait de ce format, qui permet de bien juger de la maîtrise du projet par le candidat, mais également d'évaluer de manière plus générale ses compétences en informatique, conformément au programme des classes de BCPST.

Le jury constate que les candidats sont, pour une très large majorité, très bien préparés au format de l'épreuve. Le temps dévolu à l'exposé est, dans l'immense majorité des cas, respecté et l'exposé est bien préparé. Mentionnons toutefois qu'accélérer le débit de parole pour tenir dans les 7 minutes peut nuire à la compréhension par le jury du propos tenu.

Pour la partie présentation, dans le cadre explicité par la notice du concours, il appartient au candidat d'insister sur ce qui fait l'intérêt scientifique et algorithmique de son projet :

- intérêt des structures de données utilisées, en regard de la situation étudiée (modélisation) ou des algorithmes utilisés,
- solution algorithmique élégante,
- qualité des résultats obtenus,
- etc.

Pendant l'entretien, les examinateurs cherchent à évaluer la maîtrise du candidat et son implication dans son projet. Cela peut se faire, suivant les cas, en posant des questions d'ordre général sur le contexte algorithmique ou scientifique du projet, ou des questions précises sur le code python présenté par le candidat :

- quels sont les arguments de cette fonction ?
- que renvoie-t-elle si on l'appelle sur telle valeur ?
- comment pourrait-on ajouter telle fonctionnalité ?
- justifier et/ou décrire les structures de données.
- où dans le code est implémentée telle fonctionnalité ?
- etc.

La maîtrise du programme d'informatique enseigné en BCPST a régulièrement été évaluée en demandant aux candidats de traiter au tableau un exercice de programmation. Le plus souvent, cet exercice était basé sur le projet :

- écrire une fonction parcourant une structure de données du projet,
- écrire une fonction à l'aide des fonctions du projet,
- reprogrammer dans un cas plus simple une fonction du projet,
- rendre une fonction du projet plus générale ou plus efficace,
- ré-écrire de manière plus concise une partie du projet traitée maladroitement,
- etc,

mais il pouvait aussi être sans rapport direct si celui-ci ne se prêtait pas à ce type de question :

- programmer récursivement la fonction factorielle (si la récursivité a été utilisée dans le projet),
- écrire la définition d'une classe très simple (si celles-ci ont été utilisées),
- écrire une fonction testant une propriété des matrices ou des chaînes de caractère (si de tels objets ont été utilisés dans le projet),
- programmer une fonction relative strictement au programme de BCPST (le plus souvent exercice basique sur les listes),
- écrire une fonction qui prend en entrée une liste de n entiers et renvoie `True` si ces entiers sont exactement les entiers de 0 à $n-1$.

Les candidats étaient, dans l'ensemble, bien préparés à ce type d'exercices.

Les candidats présentant un projet ambitieux et apparemment bien maîtrisé, mais incapables d'écrire au tableau une fonction élémentaire, comme par exemple déterminer le maximum d'une liste ou compter le nombre d'occurrences d'une valeur dans une structure de données, ont été fortement pénalisés.

4.2 Remarques générales

Nous reprenons ici une grande partie des points évoqués dans les rapports précédents. Les attentes du jury sont maintenant globalement comprises et peu de projets très faibles ont été présentés cette année.

- La très grande majorité des candidats semblent s'être investis dans leurs projets, et avoir, ce faisant, acquis des compétences qui leur seront utiles tant dans la suite de leurs études que dans leur vie professionnelle.
- La maîtrise générale des projets était globalement bonne.
- Les candidats veilleront à garder un esprit critique quant à la portée des programmes qu'ils ont réalisés.
- Les présentations orales, malgré quelques défauts récurrents évoqués plus bas, étaient globalement satisfaisantes et illustrées de schémas clairs et bien choisis.
- Aucun problème majeur d'organisation n'a été constaté cette année.
- Les candidats ont parfaitement le droit de programmer un algorithme déjà connu dans la littérature, qu'ils adaptent au contexte de leur projet, dès lors qu'ils ne s'en approprient pas la paternité. Il peut être bienvenu, le cas échéant, de citer ses sources bibliographiques.
- Quelques candidats ont présenté des résultats obtenus par du code qu'ils ont supprimé de leur projet et n'ont pas envoyé au concours. Ceci est à éviter.

4.3 Remarques sur la présentation

Lors de la phase de présentation du projet, le candidat dispose du diaporama au format PDF qu'il a déposé préalablement par voie électronique et qui sert de support à sa présentation. Le dossier déposé par le candidat est directement mis à disposition sur l'ordinateur utilisé par le candidat. L'usage de notes est proscrit ; cependant les candidats peuvent se munir d'un chronomètre s'ils le souhaitent.

Il est rappelé que les candidats doivent venir avec une clé USB de secours contenant leurs fichiers en cas de problème informatique. Quelques candidats (beaucoup moins que l'année dernière) n'ont pas déposé correctement leurs fichiers par voie électronique (fichier powerpoint au lieu de pdf, fichier python incomplet, voire absence de fichiers) : ces candidats ont été pénalisés, même s'ils ont pu en général utiliser leur clé de secours.

Une bonne présentation doit être synthétique. Nous attirons l'attention des candidats sur le fait que la présentation des structures de données et algorithmes utilisés est cruciale.

Le jury apprécie que les candidats 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, sans passer trop de temps à présenter des concepts non-informatiques (biologiques, physiques, mathématiques, etc.). L'analyse statistique des résultats obtenus est en soi intéressante, mais ne doit pas occuper une part trop importante de l'exposé.

Les candidats ont généralement pris soin d'expliciter leurs modèles et/ou les règles des jeux étudiés, ce qui est d'autant plus important que l'examinateur n'est pas censé interrompre le candidat durant cette phase ; il faut donc éviter de se retrouver dans une situation où l'examinateur ne comprend pas où le candidat veut en venir. Pour les mêmes raisons, les présentations contenant de trop nombreuses diapositives ou présentées en récitant un texte à toute vitesse pour tenir dans les sept minutes imposées sont à proscrire.

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 ... à condition de tenir compte de sa réponse.

Il est fortement recommandé de commenter une ou deux fonctions présentant un **intérêt algorithmique** durant cette phase de présentation. Il est donc souhaitable que ces fonctions :

- fassent apparaître des algorithmes non triviaux,
- soient concises ou bien découpées en blocs dans le diaporama, afin qu'elles soient facilement lisibles.

Il n'est pas recommandé de présenter seulement deux fonctions très simples en passant sous silence les fonctions plus complexes et intéressantes du projet. Les candidats ne doivent pas hésiter à mettre en valeur leur travail.

Le jury a apprécié que, dans leurs diapositives, de nombreux candidats utilisent des flèches, des encadrements, des accolades ou d'autres éléments graphiques permettant d'expliquer facilement leur code. A contrario, un grand nombre de lignes de codes ou des lignes de code écrites en tout petit rendent difficile voire impossible la lecture par le jury.

Les candidats prendront garde à ne pas cacher les difficultés dans des fonctions auxiliaires non présentées. 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. Le jury apprécie une bonne maîtrise du vocabulaire. Nous rappelons à ce titre qu'une boucle est une structure de contrôle permettant de répéter des opérations. Ainsi, la structure if n'est pas une « boucle ». Le jury préfère entendre « l'instruction if » voire « le if » plutôt que « la boucle if ».

Le jury est très satisfait de la qualité globale des présentations des candidats.

4.4 Qualité des projets

Le thème du projet est 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.

Cette année, les projets étaient globalement de bonne qualité et de taille raisonnable. Le nombre de lignes de code n'est pas une mesure de la qualité d'un projet. Par exemple, un projet de qualité avec 300 lignes de codes peut se voir attribuer la note maximale.

Le jury rappelle que le temps passé à travailler les graphismes et l'interface du projet est beaucoup moins rentable que celui passé à améliorer les aspects algorithmiques.

Une interface graphique via l'utilisation d'un module (tkinter, pygame, etc.) est bienvenue seulement si elle vient en complément d'un projet algorithmiquement intéressant mais ne peut pas s'y substituer. Les données d'entrée d'un programme doivent être passées en argument (ou via un fichier) et non demandées à l'utilisateur par des input, dont l'utilisation doit être réservée à des interactions avec l'utilisateur au cours du programme (par exemple pour un jeu). Nous présentons ici quelques écueils à éviter.

- Certains jeux de société ont des règles présentant un grand nombre de cas particuliers dont la traduction informatique est chronophage et ne présente pas ou peu d'intérêt. Elle engendre de nombreuses distinctions de cas, imbrications de `if`, etc.
- De manière plus générale, le code ne devrait pas faire apparaître de longues disjonctions de cas (quatre ou plus). Si de telles séries de `if` ... `elif` ... semblent nécessaires au candidat, c'est sans doute qu'il n'a pas suffisamment réfléchi à la structure du code.
- Lorsque le programme traite un grand nombre de données, il est pertinent de les stocker dans un fichier annexe.
- Le code doit pouvoir être exécuté sans lever d'erreur.
- On peut faire un projet informatique autour de son TIPE, mais ce projet ne doit pas se résumer à un traitement numérique de l'étude expérimentale du TIPE.

Certains projets se prêtent bien à une analyse statistique des résultats, qui a en général été faite par les candidats. Globalement, on a noté un effort 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 pénalisés.

On peut enfin s'attendre à ce qu'un candidat qui étudie un problème classique se soit un peu renseigné, ne serait-ce qu'en sollicitant un moteur de recherche ou son encyclopédie préférée :

- quelles solutions sont déjà connues ?
- quels algorithmes utilise-t-on habituellement ?
- quelles sont les structures de données pertinentes ?

4.5 Thèmes des projets

Cette année, les principaux thèmes abordés ont été les suivants :

- Algorithmes classiques
  - algorithmes inspirés de la biologie appliqués à des problèmes classiques : algorithmes de fourmis [1], algorithmes génétiques, réseaux de neurones, etc. Nous insistons sur le fait que ces algorithmes ont pour but de résoudre un problème informatique et non de modéliser des comportements réels, même s'ils s'en inspirent.
  - algorithmes pour la biologie : alignement de séquences, méthode shotgun, arbres phylogénétiques, jeu du chaos pour l'ADN, etc.
  - autres algorithmes : problèmes de graphes, labyrinthe, traitement d'image, analyse de texte, déplacements sous contraintes, cryptographie, etc.

  Pour ces algorithmes, il est attendu que le candidat cite les sources qu'il a éventuellement utilisées.
- Modélisation

  Dans ce paragraphe, nous ne parlons pas des sujets décrits ci-dessus, qui adaptent des modèles biologiques à la résolution de problèmes algorithmiques ou mathématiques, mais de sujets dont le but est de modéliser l'évolution d'un système (dynamique de populations, propagation d'une maladie ou d'un feu de forêt, déplacement de poissons ou d'oiseaux, gestion d'un cheptel, etc.). Même si la qualité et la pertinence de la modélisation ne sont pas les principaux éléments évalués, le jury attend du candidat qu'il ait un minimum de bon sens et qu'il y ait dans son projet matière à un développement intéressant d'un point de vue algorithmique. Ainsi, la surenchère de détails et de paramètres n'enrichit pas un modèle, mais empêche souvent l'analyse des résultats obtenus. Enfin, on attend des candidats un minimum de lucidité sur la portée et les limites de leurs modèles.
- Jeux : jeux de plateau, jeux logiques (logimages, kakuro, takuzu, etc.), jeux vidéos. C'est un domaine riche et varié. Les candidats doivent cependant faire attention au choix du jeu : le traitement algorithmique peut demander un long travail de prise en compte de règles trop nombreuses, mais sans intérêt du point de vue de la programmation (succession d'instructions conditionnelles) ; dans un tel cas, le candidat peut avoir intérêt à simplifier les règles du jeu [2], pour simplifier la mise en place initiale, et garder ainsi du temps pour programmer et comparer deux intelligences artificielles, même élémentaires (la première pouvant jouer de façon très basique, voire de façon aléatoire), ou encore faire jouer une intelligence artificielle contre une de ses variantes, obtenue en modifiant certains de ses paramètres.

Pour les sujets touchant aux mathématiques ou à la physique, il convient d'éviter de se lancer dans des sujets nécessitant la maîtrise d'outils trop éloignés du programme. Ainsi, les questions liées à la cryptographie, qui nécessitent de comprendre l'arithmétique modulaire (quand ce n'est pas la théorie des courbes elliptiques), ne donnent pas de bons sujets pour les candidats n'ayant pas, par ailleurs, une culture mathématique hors norme. Le but du jury, rappelons-le, n'est jamais de mettre en difficulté le candidat, mais de s'assurer qu'il dispose des connaissances minimales pour comprendre ce qu'il a proposé.

1. Ces algorithmes sont pertinents pour résoudre le problème du voyageur de commerce mais pas le problème du plus court chemin.
2. Il est alors recommandé d'expliciter les simplifications choisies.

4.6 Style et lisibilité du code

La lisibilité du code s'est améliorée, mais les conseils des rapports précédents restent d'actualité.

- Comme demandé dans la notice du concours 2019, les candidats étaient appelés soit à inclure à la fin de leur fichier quelques lignes de script, soit à écrire une fonction sans argument nommée `test`, permettant de tester leur projet (avec des paramètres bien choisis, pour garantir un temps d'exécution raisonnable). Cette consigne a été respectée dans une grande majorité des cas. Le jury insiste sur son importance.
- Le code doit être raisonnablement commenté : 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. Le jury félicite la plus grande partie des candidats pour le commentaire de leur code. Toutefois, il n'est pas nécessaire de commenter des instructions élémentaires.
- Le jury est très satisfait du fait que les candidats n'ont pas utilisé de chemins absolus.
  Pour rappel, 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 et on écrira à la place `f = open("especes.txt")` en mettant le fichier Python et le fichier de données dans le même répertoire.
  Lorsque plus d'une dizaine de fichiers sont présents, on pourra les regrouper dans un sous-dossier, et utiliser `f = open("Donnees/especes.txt")` [3].
- Cette année, les candidats ont fait attention à la longueur des lignes qui dépassaient rarement 80 caractères.
- Si l'on s'intéresse, par exemple, à l'évolution d'une population vivant dans un environnement en deux dimensions représenté par une matrice carrée, les fonctions devront être écrites pour des matrices de taille n « abstraite », ou bien en définissant au début du programme la constante n, ou bien en récupérant en début de fonction la taille de la matrice. On ne devrait ainsi jamais trouver une ligne telle que `for i in range(50)` dans le code d'une telle fonction. 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`. Le jury a constaté que les candidats ont fait attention à ce point.
- Les `from module import *` doivent être utilisés avec la plus grande prudence car ils conduisent, quand plusieurs bibliothèques sont chargées, à des conflits.

4.7 Notions hors-programme

Pour les candidats utilisant des structures hors programme, il n'est pas attendu qu'ils en connaissent toutes les subtilités, mais ils doivent être capables d'en justifier l'usage et il peut leur être demandé de créer une nouvelle entrée de leur dictionnaire, une instance de leur classe, voire d'y ajouter une méthode très simple.

La récursivité est à la limite du programme, les candidats sont libres de l'utiliser s'ils le souhaitent. Le jury peut éventuellement poser une question très élémentaire (programmer une fonction récursive simple, modifier la fonction récursive du projet) si elle est utilisée.

Le jury a remarqué cette année une nette amélioration à ce sujet.

4.8 Remarques spécifiques de programmation et d'algorithmique

Nous reprenons ici les remarques faites les années passées :

3. Ceci fonctionne sans problème sous Spyder, mais demande, sous Pyzo, qu'on utilise la commande *Execute file as script* (raccourci clavier Ctrl-Maj-E).

- Le code doit absolument être découpé en fonctions, et il faut réfléchir soigneusement au découpage le plus judicieux. Le jury note avec satisfaction que les candidats ont, cette année, bien suivi cette recommandation déjà présente dans les rapports précédents.
- De nombreux projets nécessitent de parcourir les voisins d'une case d'un tableau bidimensionnel, ce qui a été en général bien traité : les candidats savent souvent éviter de faire de nombreux cas particuliers.
- Il est fortement déconseillé d'utiliser un algorithme que l'on n'est pas capable d'expliquer correctement lorsque l'examinateur le demande. Les projets très pauvres à l'exception d'une fonction relativement compliquée et non maîtrisée se sont vu attribuer des notes très basses.
- Au sujet de l'algorithme de Dijkstra, que le jury sait être un algorithme difficile à appréhender, le jury se contente, quand un candidat ne l'a pas utilisé alors qu'il aurait été bien utile, de lui demander s'il connaît un algorithme vu en cours qui aurait été adapté à son problème, mais sans insister ensuite sur le fonctionnement de cet algorithme ; par contre, si l'algorithme de Dijkstra a été utilisé, le jury pourra vérifier que le candidat comprend l'algorithme et son code Python. A contrario, l'algorithme de Dijkstra n'a d'intérêt que si le graphe que l'on considère est pondéré à masses positives et 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 parti de cette possibilité, mais cela ne les dispense pas de savoir écrire une boucle `while` gérant correctement les conditions de sortie.
  L'utilisation de boucles `while` reste un point délicat pour certains candidats (confusions entre `if` et `while`).
- Plus de candidats que l'année dernière maîtrisent la différence entre une fonction renvoyant une valeur et une fonction modifiant son argument. Le jury s'en réjouit.
- Le jury a constaté, encore cette année, des progrès sur la manipulation des booléens. Idéalement, on préfère, par exemple, lire `return x > 0` plutôt que :

```
if x > 0:
    return True
else:
    return False
```

PYTHON
AGRO-VETO
2020

Listes

`[]` --------	Créer une liste vide	`L.pop(k)` --	Renvoie le $k^{\text{ème}}$ élément de la liste `L` et l'enlève de `L`
`[a]*n` -----	Créer une liste avec n fois l'élément `a`	`L.remove(a)`	Enlève une fois la valeur `a` de la liste `L`
`L.append(a)`	Ajoute l'élément `a` à la fin de la liste `L`	`max(L)` ----	Renvoie le plus grand élément de la liste `L`
`L1 + L2` ---	Concatène les deux listes `L1` et `L2`	`min(L)` ----	Renvoie le plus petit élément de la liste `L`
`len(L)` ----	Renvoie le nombre d'éléments de la liste `L`	`sum(L)` ----	Renvoie la somme de tous les éléments de la liste `L`

Numpy

`import numpy as np`

`np.array()` ----------	Transforme une liste en matrice `numpy`
`np.linspace(a,b,n)` -----	Crée une matrice ligne de n valeurs uniformément réparties entre a et b (inclus)
`np.zeros([n,m])` --------	Crée la matrice nulle de taille $n \times m$
`np.eye(n)` ----------	Crée la matrice identité de taille n
`np.diag(L)` ----------	Crée la matrice diagonale dont les termes diagonaux sont les éléments de la liste `L`
`np.transpose(M)` --------	Renvoie la transposée de M
`np.dot(M,P)` ----------	Renvoie le produit matriciel MP
`np.sum(M)` ----------	Renvoie la somme de tous les éléments de M
`np.prod(M)` ----------	Renvoie le produit de tous les éléments de M
`np.max(M)` ----------	Renvoie le plus grand élément de M
`np.min(M)` ----------	Renvoie le plus petit élément de M
`np.shape(M)` ----------	Renvoie dans un couple le format de la matrice M
`np.size(M)` ----------	Renvoie le nombre d'éléments de M

Numpy.linalg

`import numpy.linalg as la`

`la.inv(M)` ----------	Renvoie l'inverse de la matrice M si elle est inversible
`la.eigvals(M)` ----------	Renvoie la liste des valeurs propres de M
`la.eig(M)` ----------	Renvoie un couple `L,P` où `L` est la liste des valeurs propres de M et `P` la matrice de passage associée
`la.matrix_rank(M)` -------	Renvoie le rang de M

Random

`import random as rd`

`rd.random()` -------	Simule une réalisation d'une variable $X \hookrightarrow \mathcal{U}([0,1])$
`rd.randint(a,b)` ---	Simule une réalisation d'une variable $X \hookrightarrow \mathcal{U}([\![a,b]\!])$
`rd.gauss(0,1)` -----	Simule une réalisation d'une variable $X \hookrightarrow \mathcal{N}(0,1)$
`rd.choice(L)` ------	Choisit aléatoirement un élément de la liste `L`

Math

`import math as m`

`m.atan(x)` -------	Renvoie $\arctan(x)$	`m.sqrt(x)` --	Renvoie $\sqrt{x}$ si $x \geq 0$
`m.floor(x)` ------	Renvoie $\lfloor x \rfloor$	`m.log(x)` ---	Renvoie $\ln(x)$ si $x > 0$
`m.factorial(n)` --	Renvoie $n!$ si $n \in \mathbb{N}$	`m.exp(x)` ---	Renvoie e^x

Logique

`a == b` ----	Teste l'égalité « $a = b$ »
`a != b` ----	Teste « $a \neq b$ »
`a < b` -----	Teste « $a < b$ »
`a <= b` ----	Teste « $a \leq b$ »
`a > b` -----	Teste « $a > b$ »
`a >= b` ----	Teste « $a \geq b$ »
`not A` -----	Renvoie la négation de A
`A and B` ---	Renvoie « A et B »
`A or B` ----	Renvoie « A ou B »
`True` ------	Constante booléenne « Vrai »
`False` -----	Constante booléenne « Faux »

Matplotlib.pyplot

`import matplotlib.pyplot as plt`

`plt.plot(X,Y,'+-r')` ----	Génère la courbe des points définis par les listes `X` et `Y` (abscisses et ordonnées) avec les options : • symbole : `'.'` point, `'o'` rond, `'h'` hexagone, `'+'` plus, `'x'` croix, `'*'` étoile, ... • ligne : `'-'` trait plein, `'--'` pointillé, `'-.'` alterné, ... • couleur : `'b'` **bleu** , `'r'` **rouge** , `'g'` **vert** , `'c'` **cyan** , `'m'` **magenta** , `'k'` **noir** , ...
`plt.bar(X,Y)` ----------	Génère l'histogramme des points définis par les listes `X` et `Y` (abscisses et ordonnées)
`plt.axis('equal')` ------	Rend le repère orthonormé
`plt.xlim(xmin,xmax)` ----	Fixe les bornes de l'axe des abscisses
`plt.ylim(ymin,ymax)` ----	Fixe les bornes de l'axe des ordonnées
`plt.show()` ----------	Affiche le graphique

Cette liste est non exhaustive. Les candidats sont libres d'utiliser les commandes de leur choix.